Universidad Autónoma de Madrid

Escuela Politécnica Superior

22 | 23

Bachelor thesis

Quantum-Assisted Generative Algorithms

Pedro Urbina Rodríguez

Escuela Politécnica Superior
Universidad Autónoma de Madrid
C\Francisco Tomás y Valiente nº 11

excelencia Campus Internacional UAM CSIC

www.uam.es

UNIVERSIDAD AUTÓNOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

Degree in Computer Science and Engineering

BACHELOR THESIS

Quantum-Assisted Generative Algorithms

Author: Pedro Urbina Rodríguez
Advisor: Dr. Jack Jacquier
Speaker: Alberto Súarez González

June 2023

Pedro Urbina Rodríguez
Quantum-Assisted Generative Algorithms

Pedro Urbina Rodríguez
C\ Francisco Tomás y Valiente Nº 11

PRINTED IN SPAIN

To my family and friends

Machines take me by surprise with great frequency.

Alan Turing

Acknowledgements

I would like to express my deepest gratitude to Dr. Jack Jacquier, whose invaluable guidance and insights greatly enriched this project. I am also grateful to Alberto Suárez for his assistance in navigating administrative tasks and his insightful comments on this final report.

I extend my sincere thanks to the faculty who have been an integral part of my academic journey over the past five years of my degree and double degree program. Their dedication and passion for teaching have left a profound impact on me, and I am immensely grateful for their instruction.

Finally, I would like to express my heartfelt appreciation to my family and friends. Their unwavering support and encouragement have immensely helped me during this journey, particularly during the development of this thesis. Their belief in my capabilities has fueled my determination and resilience, and for that, I am eternally thankful.

Resumen

La computación cuántica ha abierto nuevas posibilidades en diversos campos, incluyendo el aprendizaje automático. Una de estas potenciales aplicaciones se da en las Redes Adversarias Generativas (GANs), que suelen tener problemas como el colapso de modos y la inestabilidad durante el entrenamiento. Esta tesis se adentra en el potencial de las Redes Adversarias Asociativas Cuánticas (QAAANs) para tratar estos problemas y mejorar el aprendizaje de distribuciones unidimensionales.

Nuestro estudio ha requerido el desarrollo de Máquinas de Boltzmann Restringidas (RBMs), tanto clásicas como cuánticas, y su integración con las GANs estándar y de Discrepancia Media Máxima (MMD), lo que constituye la base del algoritmo QAAAN. Al incorporar principios cuánticos en el diseño de estos modelos, se busca aprovechar las ventajas computacionales de la computación cuántica, como la superposición, el efecto túnel cuántico y el entrelazamiento, en tareas de aprendizaje automático complejas.

A través de una serie de experimentos, la QAAAN cuántica estándar demostró un rendimiento ligeramente superior en comparación con su contraparte clásica y la GAN estándar en ciertas distribuciones unidimensionales, especialmente en el muestreo de regiones con baja probabilidad. Sin embargo, el rendimiento de la QAAAN de tipo MMD no mostró una mejora similar frente a su versión clásica.

Los resultados de este estudio resaltan el potencial de los modelos con subrutinas cuáticas en el aprendizaje de distribuciones complejas, y ponen de manifiesto la necesidad de continuar investigando en este campo. Entre las posibles áreas para futuras investigaciones se incluyen la aplicación de diferentes algoritmos cuánticos para el aprendizaje de distribuciones a priori, la ampliación de los modelos para aprender distribuciones de mayor dimensión y la optimización adicional del algoritmo QAAAN.

Palabras clave

Computación Cuántica, Recocido Cuántico, Redes Adversarias Generativas (GANs), Máquinas de Boltzmann Restringidas (RBM), Máquinas de Boltzmann Restringidas Cuánticas (QRBM), GAN de Discrepancia Media Máxima (MMD GAN), Redes Adversarias Asociativas Cuánticas (QAAANs), Aprendizaje Automático Cuántico, D-Wave Systems

Abstract

Quantum computing has brought about transformative possibilities in various fields, including machine learning. One such potential application is in enhancing Generative Adversarial Networks (GANs), which often face challenges such as mode collapse and training instability. This thesis explores the potential of Quantum-Assisted Associative Adversarial Networks (QAAANs) in addressing these issues and improving the learning of one-dimensional distributions.

The study involves the development of Quantum and Classical Restricted Boltzmann Machines (RBMs) and their integration with Vanilla and Maximum Mean Discrepancy (MMD) GANs, forming the foundation of the QAAAN algorithm. The integration of quantum principles in the design of these models aims to harness the computational advantages of quantum computing, such as superposition, tunnelling and entanglement, in complex machine learning tasks.

In a series of experiments, the Quantum Vanilla QAAAN demonstrated a slightly superior performance over its classical counterpart and the standalone Vanilla GAN in certain one-dimensional distributions, especially in sampling from low-probability regions. However, the performance of the MMD QAAAN did not show a similar improvement over its classical variant.

The results of this study underscore the potential of Quantum Assisted Models in learning complex distributions and highlight the need for continued exploration in this field. Potential areas for future research include the application of different quantum algorithms for learning priors, the extension of the models to higher-dimensional distributions, and further optimization of the QAAAN algorithm.

Keywords

Quantum Computing, Quantum Annealing, Generative Adversarial Networks, Restricted Boltzmann Machines, Quantum Restricted Boltzmann Machines, Maximum Mean Discrepancy GAN, Quantum-Assisted Associative Adversarial Networks, Quantum Machine Learning, D-Wave Systems

Table of Contents

Lists

List of equations

List of figures

List of tables

1

Introduction

Generative models play a pivotal role in machine learning, providing a powerful framework for understanding complex data distributions. These models have been applied to a broad range of tasks, from image synthesis to anomaly detection, demonstrating their versatility and importance. Among these, Generative Adversarial Networks (GANs), introduced by Goodfellow et al. in [1], have emerged as an exceptionally promising class of generative models. However, GANs face challenges such as mode collapse, unstable training, and sensitivity to hyperparameters, issues which limit their effectiveness.

Quantum computing, on the other hand, is a rapidly emerging field that holds the potential to revolutionize many areas of computation. Quantum computers leverage the principles of quantum mechanics to perform calculations in ways that are fundamentally different from classical computers. They use quantum bits, or qubits, which, unlike classical bits that can be either 0 or 1, can exist in a superposition of states, allowing for a massive increase in computational capacity. Quantum computers also take advantage of other quantum phenomena such as entanglement and quantum tunnelling, which have the potential to solve certain problems more efficiently than classical computers.

With the emergence of quantum computing and the ongoing challenges in the field of GANs, there arises an intriguing question: Could quantum computing principles be harnessed to enhance the capabilities of GANs? This question serves as the motivation behind the Quantum-Assisted Associative Adversarial Network (QAAAN) developed in [2], the primary focus of this thesis. Quantum-Assisted Associative Adversarial Networks (QAAANs) represent an attempt to leverage the unique strengths of quantum computing to overcome the limitations of classical GANs, with the objective of developing models that can more effectively learn and sample from complex distributions.

1.1. Objectives of the project

The primary aim of this project is to delve into the capabilities of QAAAN in enhancing the learning of one-dimensional distributions. GANs, despite their promising potential, often grapple with inherent challenges such as mode collapse and training instability. By integrating quantum computing principles within the GAN framework through the QAAAN approach, our goal is to address these issues, thereby

advancing the scope of quantum machine learning.

The second objective is to perform an in-depth comparison between QAAANs and their classical counterparts, including GANs and classical Associative Adversarial Networks (AANs). The aim is to understand the benefits and drawbacks of incorporating quantum principles into the GAN framework, specifically focusing on the implications of using Quantum and Classical RBMs in the QAAAN model.

The third objective of our project focuses on the practical aspects of the implementation of QAAANs. As quantum computing is still an emerging field, there are numerous practical constraints and considerations to be taken into account, such as hardware limitations and compatibility issues. Our aim is to effectively navigate these challenges to successfully implement QAAANs on actual quantum hardware. This entails adjusting and optimizing the QAAANs algorithm to align with the usage constraints of existing quantum computing platforms, such as D-Wave.

1.2. Contributions

This thesis makes several contributions to the emerging field of quantum machine learning, specifically in the application of QAAAN for learning one-dimensional distributions. These include:

- **Performing Comparative Analysis:** The research compares Quantum and Classical QAAANs with their GAN counterparts, shedding light on the advantages and limitations of quantum principles within the GAN framework.
- **Improving One-dimensional Distribution Learning:** The study demonstrates that Quantum Vanilla QAAAN can in some cases outperform Classical models, especially when sampling from low-probability regions of distributions.
- **Implementing on Quantum Hardware:** The QAAANs were successfully tested on D-Wave quantum annealers, navigating practical constraints of quantum hardware usage.

1.3. Structure of the Thesis

The thesis is organized into several distinct sections to provide a comprehensive exploration of QAAANs. The initial portion of the work is dedicated to an exhaustive review of the literature, where fundamental concepts such as Quantum Computing, Quantum Annealing, Boltzmann Machines (BMs), and GANs are explored. This section also delves into recent developments in Quantum Machine Learning, laying the theoretical groundwork for the subsequent sections. The Methodology section details the development and implementation of both classical and quantum models, focusing specifically on GANs, RBMs in the Quantum Annealer and QAAANs. Following this, a detailed presentation and analysis of the experiments conducted and their corresponding results are provided, comparing the performance of the classical and quantum models on various distributions. Finally, the thesis concludes with a reflective discussion of the project's findings, opportunities for model improvements, potential extensions, and future research prospects in the field.

2

State of the Art

In this chapter, we delve into the fundamental principles and recent developments in quantum computing, BMs, GANs, and quantum machine learning, setting the stage for our exploration of QAAANs.

2.1. Quantum Computing and Quantum Annealing

This section explores the fundamental concepts and operational paradigms of quantum computing, highlighting the distinctive advantages and potential of quantum annealing for optimization problems in the realm of machine learning.

2.1.1. Quantum Computing

Quantum Computing is a different approach to computation which uses the principles of quantum mechanics to, at least theoretically, provide more efficient ways to perform computations and solve problems. One of the underlying reasons for this theoretical speedup is the inherent parallelism embedded in the nature of quantum mechanics provided by quantum phenomena such as superposition and entanglement.

Classical computing abstractly represents information using bits, which can only be in two states, either 0 or 1. On the other hand, quantum computing uses a different information building block called a qubit. Qubits can be in state 0, state 1, or any superposition of those states, providing a whole continuum of possible states. Qubits are an abstract way of representing quantum information, which maps to an independent underlying physical reality such as the spin of an electron, the polarisation of a photon, and a photon March-Zehnder interferometer, among others. A much deeper introduction to the field of Quantum Computing can be found in [3].

Quantum entanglement is a property or a state which can be found in quantum mechanical systems, made up of a certain amount of qubits (as units of quantum information). When two or more qubits are in an entangled state, it means that the state of any of those qubits is correlated with the other qubits in the entangled system. Entanglement is not dependent on the physical distance between the qubits,

meaning that the correlation of the qubits can happen even along very long distances. This allows protocols such as quantum teleportation, as described in [4], among other cryptographic applications.

Quantum superposition refers to the principle in quantum mechanics by which a quantum system can exist in multiple states simultaneously, with each state having a certain probability until it is measured. Then, upon measurement, the system collapses to one of the possible states based on the probability distribution. To dive deeper into these quantum concepts, we refer the interested reader to [5].

Thanks to quantum superposition, there have been found theoretical exponential speedups. Some of these include Shor's algorithm for integer factorization [6] and Grover's algorithm [7] for searching unsorted databases. These theoretical speedups hold the promise to revolutionize fields such as cryptography when the hardware advancements catch up with these theoretical findings.

Quantum computing can be broadly divided into two main paradigms: gate-based quantum computing and adiabatic quantum computing or quantum annealing. Gate-based quantum computing is similar to the classical paradigm. Quantum gates, mathematically represented by unitary transformations, are applied to qubits in order to manipulate them in the desired ways, similar to classical logic gates acting on bits. This is the predominant paradigm nowadays, as it supports a wide range of algorithms, such as Grover's and Shor's algorithms, and allows for universal quantum computation. On the other hand, quantum annealing is used for solving optimization problems by trying to maintain a quantum system in a low state of energy through a process called annealing.

2.1.2. Quantum Annealing

In this subsection, we explore the concept of quantum annealing, starting from its theoretical underpinning to its practical realization in hardware solutions.

Quantum Adiabatic Theorem

The quantum adiabatic theorem of quantum mechanics is the theoretical foundation of quantum annealing. From an information-theoretical point of view, we can view any quantum system as a set of qubits. A Hamiltonian is a mathematical operator which represents the total energy of a quantum system. Hamiltonians are crucial in quantum mechanics, as they represent the time evolution of a quantum system according to Schrödinger's equation. A Hamiltonian is said to be in its ground state if it is at a global minimum of energy.

The quantum adiabatic theorem states that if a quantum system represented by a time-dependent Hamiltonian is in its ground state and is changed slowly enough, the quantum system will remain in its ground state throughout the whole process. This means that we can theoretically find the lowest energy state of any quantum system by taking any ground state quantum system and slowly evolving it towards

our target quantum system.

Introduction to Quantum Annealing

The quantum adiabatic theorem sets the foundation for quantum annealing, also providing concrete time boundaries for its performance. The theorem suggests that we can solve optimization problems whose objective is to minimize the energy of the system or the minimum energy configuration. Quantum annealing is a technique that takes advantage of this principle, allowing a theoretically efficient way of exploring the space solution of the optimization problem. It was first introduced in works such as [8] and [9].

Quantum annealing, also known as quantum adiabatic computing, is a technique for solving complex optimization problems using the theoretical power provided by the quantum adiabatic theorem. The core idea is to map the optimization problem to a quantum system whose ground state Hamiltonian is the solution to the optimization problem, H_{target}, and use the quantum adiabatic theorem to find this ground state. In order to achieve this, an initial quantum state whose ground state Hamiltonian is easy to find, H_0, is prepared in the quantum annealer. The process of annealing thus consists of slowly evolving the initial quantum state to our target quantum state, represented by:

$$H(t) = \left(1 - \frac{t}{T}\right) H_0 + \frac{t}{T} H_{target}, \quad t \in [0, T] \tag{2.1}$$

where T is the annealing time and t is the time. If the evolution of this Hamiltonian is slow enough, the quantum adiabatic theorem guarantees that the final state of the annealer after the annealing corresponds to the ground state of the target Hamiltonian, which in turn corresponds to the solution of the optimization problem.

One of the phenomena which make quantum annealing theoretically more powerful than classical annealing is a quantum phenomenon called quantum tunnelling. While in classical optimization algorithms, the search for a global solution to the problem might get stuck in local minima, quantum tunnelling allows quantum annealing to tunnel through energy barriers, avoiding local minima and thus arriving at globally optimal solutions more quickly. The quantum tunnelling property of these systems is a direct consequence of the wave-particle duality of quantum mechanics.

D-Wave Systems

As a trailblazing company in the quantum computing landscape, D-Wave Systems has focused on the creation and commercialization of quantum annealing devices. D-Wave has consistently led the quantum annealing technology sector, providing enterprises and researchers with tools to tackle complex optimization problems theoretically more efficiently than classical computing techniques. Applications of D-Wave's quantum annealers span numerous fields, such as optimization, machine learning,

finance, drug discovery, and materials science.

D-Wave's quantum annealers employ a distinctive architecture known as the Chimera graph, a bipartite graph consisting of interconnected unit cells. Each unit cell comprises a set of qubits connected by programmable couplers, which can be adjusted to represent the optimization problem being addressed. The qubits are realized as superconducting loops, and their interactions are facilitated by Josephson junctions, enabling precise control over the coupling strength between qubits.

The annealing procedure in a D-Wave quantum annealer starts with setting the qubits to their ground states and subsequently evolving the system following a time-dependent Hamiltonian, as outlined in equation (2.1). The annealer is designed to reach extremely low temperatures, ensuring that the quantum system stays near its ground state throughout the annealing process. Theoretically, this approach allows the system to navigate the solution space more effectively than classical methods, leveraging quantum tunnelling to prevent entrapment in local minima.

D-Wave's most recent generation, the D-Wave Advantage, boasts over 5000 qubits and an improved connectivity structure called the Pegasus graph, which offers increased connectivity among qubits and enables the representation of larger and more intricate optimization problems.

2.2. Boltzmann Machines and Restricted Boltzmann Machines

Boltzmann Machines (BMs) represent a category of stochastic recurrent neural networks used for unsupervised learning, drawing inspiration from statistical mechanics principles. Their primary objective is to model complex and high-dimensional data distributions. BMs are graphs composed of multiple interconnected nodes or neurons, each representing a binary variable. The symmetric connections between these neurons possess associated weights, defining the interaction strength between neurons.

In a BM, neurons can be either active or inactive, determined by the input data and the connections among them. BMs typically employ a sigmoid or logistic activation function, transforming the neuron's input into a probability of being active. The learning process in BMs consists of discovering the optimal connection weights, enabling the network to accurately represent the underlying data distribution.

The learning method for BMs involves adjusting their weights according to the Boltzmann distribution, a probability distribution originating from the energy of the system. This distribution, along with the energy function, underpins the learning process in BMs.

2.2.1. Energy Functions and Probability Distributions

In BMs, energy functions serve as the basis for determining the probability distribution of network states. As explained in [2], a BM can be represented as a logical graph $G = (V, E)$ with cardinality N, where $V = \{z_1, z_2, \ldots, z_N\} \in \{-1, 1\}^N$ are the binary nodes of the graph connected by edges in E. The model has adjustable hyperparameters $\lambda = \{w, b\}$, where weight w_{ij} is assigned to the edge connecting variables z_i and z_j and the bias b_i is assigned to variable z_i. In this setting, the energy of the BM is defined by

$$E_\lambda(\mathbf{z}) = -\sum_{z_i \in V} b_i z_i - \sum_{(z_i, z_j) \in E} w_{ij} z_i z_j, \tag{2.2}$$

The energy function is closely connected to the Boltzmann distribution, which establishes the probability of a specific configuration of the network. The Boltzmann distribution can be formulated as

$$P(\mathbf{z}) = \frac{e^{-\beta E_\lambda(\mathbf{z})}}{Z}, \tag{2.3}$$

where β is a parameter recognized as the inverse temperature in the function defining the Boltzmann distribution, and $Z = \sum_{\mathbf{z}} e^{-\beta E_\lambda(\mathbf{z})}$ is the partition function, with the sum being over all possible states.

The Boltzmann distribution indicates the probability of a particular configuration based on its energy, with configurations of lower energy exhibiting higher probabilities. The learning process in BMs concentrates on adjusting the weights and biases to approximate the data distribution with the model's distribution. This is accomplished by minimizing the Kullback-Leibler divergence, a metric quantifying the difference between two probability distributions, between the model and data distributions.

2.2.2. Restricted Boltzmann Machines and Their Architecture

Restricted Boltzmann Machines (RBMs) are a specialized form of BMs designed to improve learning efficiency and scalability. The architecture of RBMs is distinct from general BMs, as it consists of two separate layers: visible and hidden. The visible layer represents the input data, while the hidden layer is tasked with discovering underlying data patterns. A visual representation can be seen in Figure 2.1.

The structure of RBMs is based on fully connected bipartite graphs, where every visible unit connects to all hidden units. However, there are no connections within the same layer (e.g., visible-to-visible or hidden-to-hidden). This constraint enforces a conditional independence between the units in each layer, given the state of the opposite layer. Mathematically, this can be expressed as

$$P(\mathbf{v} \mid \mathbf{h}) = \prod_i P(v_i \mid \mathbf{h}) \tag{2.4a}$$

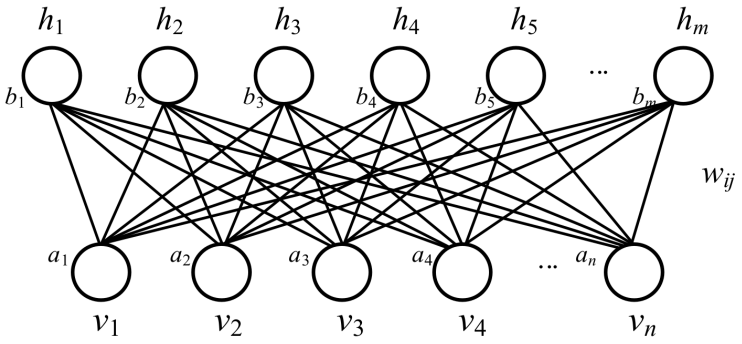

Figure 2.1: Architecture of a RBM. *Extracted from* [10]. The bipartite structure of an RBM is shown, with the visible layer containing n visible units and the hidden layer containing m hidden units.

$$P(\mathbf{h} \mid \mathbf{v}) = \prod_j P(h_j \mid \mathbf{v}) \tag{2.4b}$$

Similarly to [11], the following formulas can be used to compute these conditional probabilities

$$P(v_i = 1 \mid \mathbf{h}) = \sigma\left(a_i + \sum_j w_{ij} h_j\right) \tag{2.5a}$$

$$P(h_j = 1 \mid \mathbf{v}) = \sigma\left(b_j + \sum_i w_{ij} v_i\right) \tag{2.5b}$$

In these formulas, $\sigma(\cdot)$ represents the logistic sigmoid function, w_{ij} is the weight connecting visible unit i to hidden unit j, and a_i and b_j denote the biases for visible unit i and hidden unit j, respectively.

The restricted design enables a more manageable learning process, allowing for the application of efficient algorithms such as contrastive divergence during training. Moreover, RBMs are well-suited for handling large datasets and high-dimensional data, making them applicable in various contexts, including feature learning, dimensionality reduction, and collaborative filtering.

2.3. Generative Adversarial Networks

Generative Adversarial Networks (GANs) represent a category of generative modelling techniques that have garnered significant attention due to their capacity to generate novel data instances based on a provided dataset. They were first proposed in [1], with their architecture consisting of two neural networks, called the generator and the discriminator, which are trained simultaneously in a game theoretical manner. The main objective of the generator is to create samples similar to the ones in the dataset and trick the discriminator net into thinking they have been drowned from the real dataset while the discriminator tries to discern between real samples and the samples created by the generator.

The main idea underlying this architecture is the concept of adversarial training, by which a minimax game is designed for the generator and discriminator to compete in and learn in the process. This process drives the generator to create more realistic samples, at the same time as the discriminator gets better at discerning between real and generated samples. Thus, a positive feedback loop appears in this training process, as in order only to keep tricking the discriminator at the same rate, the generator needs to keep getting better at generating samples.

From the previous process, it is easy to see how the training process might fail. The Nash equilibrium in the minimax game has to be maintained throughout the process because if the generator at some point gets much better at discerning that the generator is training, the training process will fail, as both networks will stop being better, and the same would occur in the opposite case. However, if training is achieved, GANs are able to create sharp and varied samples resembling the original dataset in an unsupervised manner. This is the main reason GANs have achieved such success in fields like image generation.

2.3.1. GAN Architecture and Training Process

GANs are composed of two differentiated neural networks, the discriminator and the generator. The target of the generator is to transform latent variables, usually noisy inputs, through a variable amount of layers to outputs that closely resemble the distribution of the original dataset but which are not exactly samples from the real dataset. Using the notation in [2], we represent this noisy input as $z \sim q(z)$, where $q(z)$ is usually a simple uniform distribution. Thus the generator is represented by a function $x = G(z)$, which transforms this input distribution to the distribution underlying the original dataset. The discriminator takes as input real samples from the dataset and samples generated by the generator, without knowing from where it came each and tries to discriminate real from fake samples.

The training process of the GAN consists of the minimax game is described by the following expression

$$\min_G \max_D \left(\mathbb{E}_{x \sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim p_{gen}(z)}[\log(1 - D(G(z)))] \right) \tag{2.6}$$

Where $\mathbb{E}_{x \sim p_{data}(x)}$ is the expectation over the distribution of the dataset, $\mathbb{E}_{z \sim p_{gen}(z)}$ is the distribution over the latent variable distribution (the input to the generator), and G and D are the functions represented by the generator and discriminator networks respectively.

In this equation, the generator tries to minimize the likelihood that the discriminator can identify generated samples, represented by the term $\mathbb{E}_{z \sim p_{gen}(z)}[\log(1 - D(G(z)))]$. At the same time, the discriminator tries to maximize the likelihood that it correctly identifies real data, represented by term $\mathbb{E}_{x \sim p_{data}(x)}[\log D(x)]$, while at the same time minimizing the likelihood that it misclassifies a generated sample, represented by the term $\mathbb{E}_{z \sim p_{gen}(z)}[\log(1 - D(G(z)))]$. This process is illustrated in Figure 2.2.

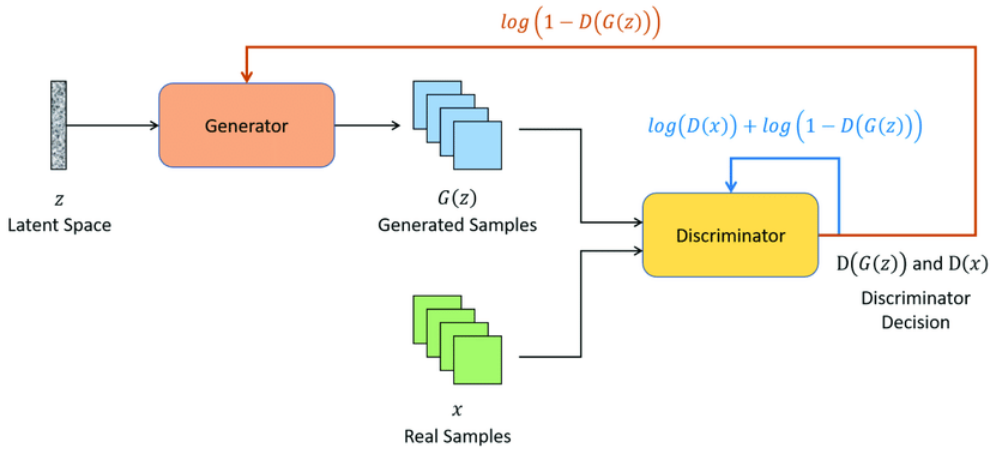

Figure 2.2: Architecture of a GAN. *Extracted from* [12]. The discriminator receives two types of inputs, real samples and generated samples, and its target is to distinguish between the two.

Some challenges can arise during this training process, the most common being mode collapse and divergence or unstable training. Mode collapse arises when the generator gets fixated on generating a small subset of samples that consistently deceive the discriminator. Consequently, the generator abandons its exploration of the entire data distribution and excessively generates these successfully deceiving samples. This leads to a diminished variety in the generated data, preventing the accurate representation of the complexity of the original data distribution. The resolution of mode collapse is an essential step towards enhancing GAN's effectiveness, with several potential solutions being currently investigated, encompassing works as [13], [14] and [15], among others.

The adversarial game between the generator and discriminator also needs a delicate balance in their learning progress. If the discriminator outlearns the generator, the valuable guidance provided by the gradients to the generator may fade, negatively impacting its learning. Additionally, if the generator advances too rapidly for the discriminator to catch up, it could lack the beneficial feedback for refinement and continue generating suboptimal samples. This unsteady equilibrium can lead to a cycle of constant oscillations where the generator and discriminator continuously strive for superiority, but without moving closer to the desired model convergence. In such a state of non-convergence, the network stops its growth in effectiveness, thus failing to fulfil the primary objective of GANs. This phenomenon has been thoroughly explored in [16], [17], [18] and [19], among others. However, when all these challenges are overcome, GANs can produce high-quality and very diverse samples from the original distribution.

2.3.2. Variants of GANs

GANs have undergone numerous adaptations and enhancements since their inception. This section introduces several well-known GAN variants, including Deep Convolutional GAN (DCGAN), Wasserstein GAN (WGAN), and Maximum Mean Discrepancy GAN (MMD GAN), and highlights their differences compared to the original GAN in aspects like architecture, training methodology, and goals. We also

briefly discuss the benefits and drawbacks of each GAN variant.

DCGAN is a prevalent GAN variant that incorporates deep convolutional layers within the generator and discriminator networks. Radford et al. [20] proposed DCGAN to enhance GAN training stability and sample quality. This variant suggests specific architectural guidelines such as using strided convolutions in place of pooling layers, implementing batch normalization, and employing distinct activation functions (Rectified Linear Unit (ReLU) for the generator and Leaky ReLU for the discriminator). These modifications contribute to more stable GAN training and enable more effective learning of intricate data distributions, particularly in image generation tasks. However, DCGAN doesn't entirely resolve challenges like mode collapse and unstable training dynamics.

WGAN, introduced by Arjovsky et al. [21], tackles the issues of mode collapse and training instability by employing a new loss function based on the Wasserstein distance. This distance metric quantifies the difference between two probability distributions and offers better convergence properties compared to the original GAN loss function. WGAN also utilizes a weight clipping strategy to impose a Lipschitz constraint on the discriminator, contributing to stable training dynamics. Generally, WGAN provides enhanced training stability and mitigates mode collapse. However, it might need more iterations for optimal performance, and the weight-clipping technique can sometimes lead to unwanted side effects.

The MMD GAN, proposed by Li et al. [22], replaces the standard GAN discriminator with an MMD critic. As a kernel-based metric, the MMD assesses the distance between two probability distributions, effectively comparing the generator's output with the target data distribution. This change streamlines adversarial training and reduces mode collapse risk. However, the MMD GAN's reliance on kernel functions can increase computational complexity, potentially impacting scalability with large, high-dimensional datasets.

2.4. Recent Developments in Quantum Machine Learning

In the past few years, quantum machine learning has witnessed remarkable progress, with a number of quantum algorithms theoretically outperforming their classical equivalents. One such development is the Quantum Support Vector Machine (QSVM), which has become a prominent quantum machine learning approach [23]. QSVMs provide benefits in terms of computational efficiency and overall performance compared to traditional support vector machines, paving the way for more effective classification tasks. The advancements in QSVM hold the potential to influence a variety of applications that require efficient classification techniques.

Another significant milestone in Quantum Machine Learning is Quantum Principal Component Analysis (QPCA), a powerful method for dimensionality reduction [24]. QPCA surpasses Classical Principal Component Analysis (PCA) regarding computational complexity and performance, positioning itself as a promising solution for tasks that demand efficient data compression and representation. This pro-

gress in QPCA has broad implications for numerous fields that rely on high-dimensional data analysis and feature extraction.

Furthermore, the Quantum Approximate Optimization Algorithm (QAOA) has emerged as a successful quantum machine learning technique for addressing combinatorial optimization problems [25]. The benefits of QAOA concerning computational effectiveness and optimization capabilities render it a valuable resource for a variety of optimization tasks. As QAOA continues to advance, its potential applications and impact on optimization issues across multiple domains are expected to expand. Other recent developments in the field are discussed in [26].

2.4.1. Quantum Annealing-based Machine Learning

In recent years, the evolution of quantum annealing hardware has played a crucial role in enhancing the performance and scalability of quantum annealing algorithms within the field of machine learning. Companies such as D-Wave Systems have made significant strides in developing increasingly sophisticated quantum annealing processors, with their most advanced system, the D-Wave Advantage, housing over 5,000 qubits. These advancements in hardware not only boost the efficiency of quantum annealing algorithms but also expand their potential applications across a broad range of machine learning tasks, including optimization and data representation.

Essentially, quantum annealers are built to address specific optimization challenges, but environmental interactions and still far from ideal hardware can instead result in sampling behaviour. This sampling ability is useful in probabilistic deep learning approaches, like BMs, which require sampling from intricate probability distributions. Quantum annealers can help explore more complex topologies beyond Restricted Boltzmann Machines (RBMs), which have bipartite connectivity due to classical limitations. With quantum annealing hardware, researchers can investigate richer topologies and graph structures, expanding the potential of BMs in deep learning applications. This is done in [27], where the authors loosened the restrictions imposed by the bipartite structure of a RBM and allowed certain connections between cells among the hidden layer, calling this new architecture a Limited Boltzmann Machine and showing that it performs slightly better than the RBM.

In [2], authors incorporate a quantum annealer into the GAN training process to train an additional BM. This quantum Boltzmann Machine aims to learn a data feature distribution from the discriminator's intermediate layer and replace the generator's noisy input with samples from this informed prior distribution. This approach hopes to foster convergence in training and reduce mode collapse. While exploring various BMs topologies, they found that despite classical settings benefiting from more network connections, quantum annealer-related improvements weren't substantial, likely due to hardware noise.

3

Development

This chapter provides a detailed overview of the research process and the implementation carried out during the course of this project. We present the methodology followed and the execution of each task within the work plan, briefly citing the results that will be further discussed later. The project is divided into two main phases: the development of classical GANs and the implementation of the QAAAN.

During the first phase of the project, our focus was on developing two types of GANs: Vanilla GAN and MMD GAN. The primary goal of this phase was to learn one-dimensional distributions, such as uniform, Gaussian, Cauchy, and Pareto distributions, by building upon the work done in [28] on GANs for learning one-dimensional distributions, in order to compare their performance on simple datasets. In the second phase, the objective was to develop a quantum-assisted GAN for learning one-dimensional distributions. To accomplish this, we referred to the quantum-assisted associative adversarial network (QAAAN) developed in [2] which employs a quantum-assisted GAN to learn the MNIST dataset.

The strategy for developing the QAAAN involved training a Boltzmann machine on a quantum annealer to learn a feature representation of the data and then using the samples generated by the quantum annealer as the prior for the generator. To implement this algorithm, we initially created a classical RBM using Markov chain Monte Carlo (MCMC) sampling for learning, followed by a quantum RBM on the D-Wave quantum annealer. Once both RBMs were developed, we integrated them with the previously coded GANs to implement the quantum-assisted RBM as a prior algorithm. This approach demonstrated the capability of learning the target distributions.

3.1. Implementation Methodology

The implementation and testing of this project followed a progressive and exploratory methodology, starting with individual components before moving to the complete system. Initially, the core components - GANs and the RBMs - were developed. This stage involved rigorous testing and refinement of each component in isolation.

Following the successful individual implementation and verification of the GANs and RBMs, a quantum version of the RBM was then created. It involved the complex task of mapping the RBM problem

onto the physical hardware of D-Wave Systems' quantum annealer using the *pyqubo* library, which aids in the efficient solution of QUBO problems.

With the separately functioning elements in place, the QAAANs were then developed, incorporating the previously tested components. This integration represented a phase akin to integration testing, as it evaluated the combined performance and interaction of the individual units within the whole system.

The final stage focused on extensive testing and optimization of the QAAANs. A series of rigorous experiments using different one-dimensional distributions were performed to assess the models' capabilities and limitations. The experiments emphasized the efficient use of computational and quantum resources, the adjustment of the discriminator, RBM, and generator parameters' update ratios, and improving learning results. This iterative testing and refinement process helped shape the continuous evolution and improvement of the QAAANs. Finally, their performance was compared to classical GANs and AANs, showcasing the potential of QAAANs in learning one-dimensional distributions.

3.2. Classical GANs

In this section, we detail the implementation of the Vanilla and MMD GAN used in this project. Our goal was to learn simple one-dimensional distributions like uniform, Gaussian, Cauchy, and Pareto. We discuss architectural choices, development processes, and encountered challenges, providing an overview of the algorithms.

3.2.1. Preliminary Experiments

Before the main project, several preliminary steps were taken. First, a strong foundation was built with TensorFlow, the machine learning library employed for the project. This involved engaging with various tutorials, which offered a clearer understanding of the structure and functionality of GANs. Next, an initial DCGAN model was implemented based on a TensorFlow tutorial. This model allowed for deeper insights into the workings of convolution and the balancing of learning rates, which would later prove vital during the primary implementation of the Vanilla and MMD GAN models.

3.2.2. Vanilla GAN Implementation

We started by implementing the Vanilla GAN, one of the first and most fundamental GAN architectures, as proposed by Goodfellow et al. in their original paper "Generative Adversarial Nets" [1]. The Vanilla GAN consists of two components: a generator, which creates synthetic data samples, and a discriminator, which evaluates the authenticity of the generated samples by comparing them to the real data. The algorithm works in a minimax game setting, where the generator and discriminator are

trained in an adversarial manner according to Equation 2.6, with the generator trying to create samples that the discriminator cannot distinguish from real data, and the discriminator aiming to correctly classify samples as real or fake. The original implementation also involves an additional hyperparameter which represents the number of discriminator training steps for each generator training step. This hyperparameter was found to be crucial for the performance of the algorithm.

A crucial architectural decision made during the implementation process was to have the generator create single samples instead of a set of points. This choice was influenced by the observation that generating sets of points for the discriminator to evaluate led to suboptimal performance. Given that the intuition at first seemed to point out that generating and discerning between sets of samples instead of single samples would facilitate training, the first implementation of the Vanilla GAN was designed in this way. However, the results were not giving good results so we had to change to the single sample implementation. By generating single samples, the discriminator could focus on evaluating the authenticity of each sample individually, leading to improved model performance.

While implementing the Vanilla GAN, it was important to carefully choose the activation functions, loss functions, and optimization algorithms for both the generator and discriminator components. We used the Exponential Linear Unit (ELU) as the activation function for the layers of both the generator and discriminator and binary cross entropy as the loss function. We also used Adam as the optimizer. We tried different number of layers with different layer sizes, which gave similar results, so we fixed as in [28] the number of layers of both the generator and the discriminator to 4. The generator consists multilayer perceptron with layer sizes 7, 13, 7, and 1 layers, while the discriminator consists of another multilayer perceptron with 11, 29, 11, and 1 units in each layer.

The training process involved alternating between updating the generator and discriminator weights based on their respective loss functions, with the ultimate goal of reaching an equilibrium where the generator produces samples that the discriminator cannot distinguish from real data. In order to optimize the hyperparameter which determines the best discriminator-generator training step ratio, we conducted a grid search on the set $\{1, 2, 3, 4, 5\}$, representing the number of discriminator updates for every generator update, and the best was found to be 5, which means that for every weight update of the generator we updated five times the weights of the discriminator.

During the implementation process, several challenges were encountered. The main challenge found at first was divergence in the training. This was solved by selecting the 5:1 discriminator-generator update ratio and choosing a smaller training rate. The optimal one was found to be 0.001. All these changes resulted in a more robust and efficient model, even though the training was slower.

After addressing the challenges and refining the Vanilla GAN implementation, the model was applied to learn one-dimensional distributions such as uniform, Gaussian, Cauchy, and Pareto distributions. The performance of the Vanilla GAN on these simple datasets served as a baseline for comparison with the more advanced MMD GAN model. The results show that the Vanilla GAN is able to learn all

the distributions when all their mass is close to 0. It is particularly good at learning uniform, Gaussian and Pareto distributions, but struggles with the Cauchy distributions. The results also indicate that it is not able to learn some distributions when the mass of the distribution is very far away from 0. These results will be further discussed in subsequent chapters.

3.2.3. MMD GAN Implementation

Kernel methods are crucial in machine learning, particularly when addressing non-linear data analysis challenges. These methods assess the similarity between data points by computing the dot product within a higher-dimensional space, without the need for explicit mapping of the data points to that space – a technique known as the "kernel trick". As explained in [29], a kernel function K takes two inputs from an input space $(x, y) \in \mathbb{R}^d$ and returns their inner product in the feature space through a mapping ϕ. In mathematical terms, this can be expressed as

$$K(x, y) = \langle \phi(x), \phi(y) \rangle \tag{3.1}$$

To qualify as a valid kernel, the function must possess certain properties. Firstly, it should be symmetric, i.e., $K(x, y) = K(y, x)$. Secondly, it should satisfy the Cauchy-Schwarz inequality: $K^2(x, y) \leq K(x, x)K(y, y)$. Lastly, for the existence of the feature space, K should be positive definite.

Generative Moment Matching Networks (GMMNs) represent a category of generative models that employ kernel methods to directly minimize the Maximum Mean Discrepancy (MMD) distance between the authentic data distribution and the generated distribution. Unlike GANs or WGANs, GMMNs do not use an auxiliary discriminator or critic. Instead, they aim to match the statistical moments (e.g., mean, variance) of the true and generated distributions, providing an alternative strategy for learning generative models. However, the choice of kernel method can sometimes restrict GMMNs' effectiveness.

MMD GANs are an extension of GMMNs. The MMD distance can be described mathematically using kernel functions. Given two distributions $\mathbb{P}$ and $\mathbb{Q}$, and a kernel k, the square of MMD distance is defined in [22] as follows:

$$M_k(\mathbb{P}, \mathbb{Q}) = ||\mu_{\mathbb{P}} - \mu_{\mathbb{Q}}||_H^2 = \mathbb{E}_{\mathbb{P}}[k(x, x')] - 2\mathbb{E}_{\mathbb{P},\mathbb{Q}}[k(x, y)] + \mathbb{E}_{\mathbb{Q}}[k(y, y')], \tag{3.2}$$

Here, $\mu_{\mathbb{P}}$ and $\mu_{\mathbb{Q}}$ are the means of the distributions $\mathbb{P}$ and $\mathbb{Q}$, respectively. x and x' are random variables which follow the probability distribution $\mathbb{P}$, while y and y' are random variables which follow the probability distribution $\mathbb{Q}$.

The key point is that we would like kernel k to result in a high MMD distance $M_k(\mathbb{P}, \mathbb{Q})$ when $\mathbb{P} \neq \mathbb{Q}$, as this would mean that we can differentiate $\mathbb{Q}$ from $\mathbb{P}$. We would like to find the kernel k which is best able to separate the real distribution of the dataset ($\mathbb{P}_{\mathcal{X}}$) and generated ($\mathbb{P}_\theta$) distribution. As explained

in [22], we would thus like to find

$$\min_{\theta} \max_{k \in \mathcal{K}} M_k(\mathbb{P}_{\mathcal{X}}, \mathbb{P}_{\theta}). \tag{3.3}$$

As the space of kernel functions $\mathcal{K}$ is too large to be searched, we can instead use a neural network parametrized by ϕ, called critic, to find the best kernel possible. Therefore, instead of training the generator G_θ via a pre-specified kernel k as in GMMNs, the MMD GAN developed in [22] considers training G_θ with the following objective

$$\min_{\theta} \max_{\phi} M_{k \circ f_\phi}(\mathbb{P}_{\mathcal{X}}, \mathbb{P}_{\theta}), \tag{3.4}$$

where k is characteristic kernel, in our case we chose $k(x, x') = \exp(-||x - x'||^2)$, and f_ϕ representing our critic network.

The MMD GAN approach thus incorporates an adversarially learned kernel function that is not fixed but instead learned by a deep neural network, the critic. By learning the kernel function, MMD GANs can more effectively compare the statistical characteristics of the actual and generated distributions, resulting in improved generative performance. In this manner, MMD GANs merge the advantages of GMMNs with adversarial training, providing a powerful alternative for learning 1D distributions.

To implement the MMD GAN, we adhered to Algorithm 1 outlined in [22], where we replaced the autoencoder employed in their algorithm with a multi-layer perceptron. The primary reason for using an autoencoder in the original paper was to meet certain injectivity conditions, which the authors later indicated might not be necessary. Additionally, [28] also constructed the MMD GAN using a basic multi-layer perceptron as the critic.

In implementing the network, the most challenging aspect was comprehending the concepts involved in MMD GANs. We aimed to construct the generator and critic networks similarly to the generator and discriminator networks in Vanilla GAN. Consequently, the generator consists of four layers with sizes 11, 29, 11, and 1, while the critic also has four layers with sizes 7, 13, 7, and 1. We conducted a range of experiments using a grid search approach for hyperparameter selection, encompassing the critic-generator update ratio, learning rate, and other novel parameters introduced in the new algorithm, such as the clipping parameter and regularization coefficient. We discovered that the optimal critic-to-generator update ratio was 3 to 1, the most effective learning rate to prevent mode collapse was 0.001, the clip parameter was 1, and the regularization coefficient was 0.01.

Upon completing the hyperparameter selection, we first performed a series of experiments with the masses of the distributions close to 0. They showed that the MMD GAN is extremely good at learning the uniform and Cauchy distributions, and relatively good at learning the Pareto and Cauchy distributions, outperforming the Vanilla GAN in all cases but the Pareto, where the results are similar. We discovered that the MMD GAN outperformed the Vanilla GAN in learning distributions whose mass is far away from

0. The MMD GAN successfully learned various distributions also when their mass is further away from 0. We will delve into these results and associated experiments in greater detail in a subsequent chapter.

3.3. Classical and Quantum RBMs

The main objective of this section is to discuss the development and implementation of classical and quantum RBMs as an essential aspect of the project. The primary motivation for developing these RBMs is to integrate them into the QAAAN algorithm, which was employed in the final experiments. This second part project can be divided into two main stages: first, the incremental construction of classical and quantum RBMs, and second, their application within the QAAAN algorithm, the topic of the next section.

Developing a classical RBM was the first step, which laid the foundation for building the quantum RBM. Constructing the quantum RBM using the D-Wave quantum annealer was the most challenging part of this project, mainly due to the intricacies of mapping the logical RBM onto the physical hardware of the D-Wave systems. To evaluate the performance of these RBMs, the MNIST dataset was used, and both the classical and quantum RBMs were able to learn and generate handwritten digit samples successfully.

3.3.1. Training Restricted Boltzmann Machines

The primary objective of training RBMs is to fine-tune the network's weights and biases. The Contrastive Divergence (CD) algorithm has become a popular method for RBM training, as it seeks to minimize the difference between the distribution generated by the model and the true data distribution.

The CD approach consists of two main stages: the positive phase and the negative phase. In the positive phase, activation likelihoods for hidden units are determined based on the input data, and hidden units are subsequently selected using these probabilities. During the negative phase, visible units are reconstructed from the chosen hidden units to produce new data samples. These reconstructed visible units are then employed to compute the activation likelihoods of hidden units once again.

The network's weights and biases are modified throughout the training to lessen the disparity between the data produced by the model and the original data. If the negative log-likelihood function is the chosen cost function and we use a gradient-based method for the training such as Stochastic Gradient Descent (SGD), it can be shown as in [11] that the update rule for the weights and biases is given by

$$w_{t+1} = w_t + \epsilon(h_t \cdot v_t^\top - h_t' \cdot v_t'^\top), \tag{3.5a}$$

$$a_{t+1} = a_t + \epsilon(v_t^\top - v_t'^\top), \tag{3.5b}$$

$$b_{t+1} = b_t + \epsilon(h_t^\top - h_t'^\top), \tag{3.5c}$$

where ϵ is the learning rate, v_t is the initial input for the visible units, h_t are the values of the hidden layer given input v_t, v_t' are the values sampled from the visible units when we have as hidden vector h_t and h_t' are the values of the hidden layer given the input vector v_t', when hyperparameter k is 1 in the CD algorithm.

CD uses MCMC as a subroutine to obtain the distribution of the RBM and update the parameters accordingly. An important hyperparameter of this algorithm is k, which refers to the number of MCMC steps which are taken in each step of the training. If k is bigger than 1, the process of obtaining h_t' and v_t' is run k times. This means that the after the hidden sample h_t is sampled from the initial sample, the process of sampling v_t' from h_t in the first iteration and from h_t' in the rest and sampling h_t' from v_t' is repeated k times. As k increases, we are guaranteed to get a sample closer to the real underlying distribution of the RBM. However, in practice, setting $k = 1$ usually gives quite good results. Given that it is computationally less expensive than bigger k's, this is the usual choice for training RBMs.

Using these methods has shown that RBMs can be effectively and efficiently trained, enabling the model to understand the underlying data distribution and generate new data samples that closely mirror the input data distribution.

3.3.2. Implementation of Classical RBM

The algorithm implemented in order to learn the MNIST data is the one explained in the previous section. Each image in the MNIST dataset consists of $28^2 = 784$ pixels, which meant that the number of visible units in the RBM was fixed at 784. Given that the units in the RBM are binary, we had to preprocess the MNIST data. Each pixel represents a grayscale value in the range $[0, 1]$, so in order to make these values binary, we selected a threshold, in our case 0.5 and pixels were assigned a 0 or 1 value depending if they are bigger than this threshold. This way the MNIST data could now be treated by the RBM.

The hyperparameter selection was carried out using grid search. The first hyperparameter to tune was the k parameter in the CD algorithm. Different configurations of this hyperparameter showed that setting $k = 1$ gives very good results, as noted in [11]. The learning rate and the number of hidden layers of the RBM were other hyperparameters to adjust. The learning rate was searched in $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}\}$ and the number of hidden layers in $\{10, 20, 50, 100\}$. The results showed that a learning rate of 0.01 and 20 hidden layers was enough to learn how to generate single digits of the data.

3.3.3. Implementation of Quantum RBM

The implementation of the Quantum RBM was one of the most challenging parts of the project. Throughout all the experiments we used D-Wave's hardware, interacting with it through the API provided by D-Wave. The experiments used the D-Wave's Advantage system, with over 5000 qubits and 15-way qubit connectivity in a Pegasus topology.

The initial approach was to implement a hybrid RBM similar to the implementation of [27], where the hidden units would be embedded in the quantum annealer, while the visible units would remain in the classical computer. The main drawback of this approach is that it required an embedding of the logical graph represented by the RBM into the physical distribution of the qubits in the quantum annealer, which varies depending on the system used. This meant that in order to specify this approach we would have to find a way to transform our problem into a Quadratic Unconstrained Binary Optimization (QUBO) problem. The main benefit of this approach is that a higher number of hidden units could be added to the RBM, as only these kinds of units were embedded to the quantum annealer. However, for our specific problem of helping GANs learn one dimensional distributions, we do not need a huge number of hidden units.

In light of this, we shifted our focus towards embedding the complete RBM within the quantum annealer, following [11]. This approach introduced us to the *pyqubo* library, which provides the capability to represent problems in Hamiltonian form and subsequently convert them into a QUBO problem. QUBO is the only format that the quantum annealer readily understands. Given that RBMs are fundamentally energy models, we could frame the RBM training problem as a Hamiltonian one, incorporating the weights and biases of the RBM. With the aid of the *pyqubo* library, this Hamiltonian problem was compiled into a QUBO problem that the quantum annealer could efficiently solve.

Employing this strategy, we succeeded in streamlining the creation of the QUBO problem within a set of dedicated methods. These methods were then invoked during the sampling of both visible and hidden layers. Consequently, this resulted in a final training procedure that bore a strong resemblance to the previously implemented classical RBM. The significant difference lay in the sampling process, which was executed on a quantum annealer as opposed to using traditional MCMC sampling methods.

A key challenge in implementing the RBM using a quantum annealer was the substantial communication time required by the API. This necessitated limiting the dataset size, the number of input images, and the epoch count. Despite the reduced number of images used for training, the RBM demonstrated the capability to learn single digits within a relatively small number of epochs.

The interaction with the D-Wave system was the primary bottleneck in our algorithm. However, we found a way to optimize this by increasing the number of problem-solving iterations the quantum annealer performed in a single API call. This approach yielded a collection of distinct solutions from the quantum annealer. Among these, we selected the best solution—identified by the lowest energy—to

guide our gradient descent. This optimization ensured a more informed gradient descent process despite the limited number of epochs, thereby enhancing the overall efficiency of our approach.

We found that the quantum annealer is able to learn single digits of the MNIST dataset using much fewer epochs thanks to this approach. This motivated us to carry on with the experiments with the QAAAN.

3.4. Quantum-Assisted Associative Adversarial Network

The purpose of this section is to explain the development procedure of the main part of the project, the implementation of the Quantum-Assisted Associative Adversarial Network (QAAAN) developed in [2] for learning one-dimensional distributions instead of the MNIST data.

The development of QAAAN is motivated by the desire to harness the potential of quantum computing in the field of machine learning, specifically for the task of learning one-dimensional distributions. Despite classical GANs and RBMs being effective tools for learning data distributions and generating new samples, they are not without their limitations. Classical GANs can suffer from issues like mode collapse and difficulty in training, while RBMs can be computationally intensive due to the need for MCMC sampling.

The QAAAN aims to overcome these challenges by leveraging the strengths of both GANs and RBMs, while utilizing the power of quantum computing. By integrating a quantum RBM with a GAN, the QAAAN can potentially provide a more efficient and robust solution for learning data distributions. The use of a quantum annealer, in particular, offers a natural and efficient way to sample from complex distributions, which is a crucial step in the training of RBMs.

Furthermore, applying the QAAAN to the task of learning one-dimensional distributions presents an interesting challenge and an opportunity to evaluate the effectiveness of quantum computing in a relatively simpler setting compared to complex datasets like MNIST.

3.4.1. Overview of QAAAN Implementation

The QAAAN serves as a unique and innovative variation of the conventional GAN model. At its core, the QAAAN algorithm is designed to effectively leverage the power of a BM to enrich the feature representation used by the generator in the GAN structure.

Traditionally, the generator in a GAN utilizes a simple prior, such as a Gaussian or uniform noise distribution. However, in the QAAAN model, this simple prior is replaced with a more expressive and informative one, specifically a BM that has been trained on feature representations extracted from the discriminator. This modified approach allows the generator to access a deeper and more comprehen-

sive understanding of the underlying data, thus enabling it to produce more accurate and high-quality generations.

Our implementation of the QAAAN is primarily based on Algorithm 1 outlined in [2]. The entire training algorithm is constituted of three distinct yet interconnected steps, forming a cycle that is iterated for a predefined number of epochs.

1.– **Training the Discriminator:** The first step involves training the discriminator. This process begins by generating samples using the BM, which are then utilized as input for the generator. The discriminator is trained using a mixture of these generated samples and real samples from the dataset. Employing traditional gradient descent optimization, the discriminator learns to differentiate between the real and artificially generated samples, improving its discerning capability with each iteration.

2.– **Training the BM:** Following the training of the discriminator, the BM is trained. The objective here is to enable the BM to learn a representative feature set, extracted from the penultimate layer of the discriminator. This raw input, however, needs to be preprocessed into binary data to make it compatible with the BM's binary nodes. Once the input is suitably transformed, the BM is trained to recognize these features, thereby enhancing its own generative capabilities.

3.– **Training the Generator:** The final step in the cycle involves training the generator. This is accomplished through gradient descent optimization, using the generator's output when it is fed with the samples generated by the BM. Over time, this training process enables the generator to produce samples that increasingly resemble the real data in the distribution.

These three steps constitute a single iteration or epoch. The entire process is repeated multiple times, with each cycle further refining the capabilities of the discriminator, the BM, and the generator. This iterative approach facilitates the gradual and balanced development of all three components, resulting in a more effective and accurate QAAAN model.

3.4.2. Implementation Process of QAAAN

Leveraging the previously implemented Vanilla GAN, MMD GAN, Classical RBM, and Quantum RBM, the implementation of QAAAN centred around the harmonious integration of these components and the fine-tuning of their respective hyperparameters. This optimized collaboration was aimed at achieving superior results through their synergistic interaction. This resulted in the creation of four distinct QAAAN models: two based on the Vanilla GAN and two on the MMD GAN. Each of these variants comprised a quantum version that utilized the previously discussed Quantum RBM and a classical version that employed the Classical RBM. These will henceforth be referred to as the Classical Vanilla QAAAN, Quantum Vanilla QAAAN, Classical MMD GAN, and Quantum MMD GAN. The implementation of all these four algorithms was very similar, as they all shared the same functionalities and their development processes were quite alike.

The first step in the implementation of the algorithms was to integrate the RBM between the generator and discriminator or critic of the Vanilla and MMD GANs. The input to the RBM had to be the feature distribution extracted from the penultimate layer of the discriminator. Given that the RBM only accepts

binary data on its nodes, we had to find a way to encode the information from the penultimate layer of the discriminator into a binary array. Following [2], the activation function of the penultimate layer was set to be the hyperbolic tangent so that the output gives values between -1 and 1. Thus, we could simply encode the output of this layer into a binary array, setting to 1 values which are bigger than 0 and 0 otherwise. Now, both the quantum and the classical RBMs could learn the feature representation extracted from the discriminator.

The second step of the integration consisted in reparametrizing the binary output of the RBM into continuous data which could be better processed by the generator. We tried several reparametrizations but the one which worked best was to map each 0 in of the output into a -1, keeping each 1 as 1 and adding some noise to make the result a continuous sample. If the range of the noise was very small the generator tended to produce very similar samples, a phenomenon known as mode collapse. However, there was a tradeoff between the probability of resulting in mode collapse and the amount of information contained in the feature layer representation. A standard deviation of $0{,}3$ in the noise was found to give good results, avoiding mode collapse and maintaining most of the information.

Once the RBM was integrated with the GAN, the training algorithm was developed. We followed Algorithm 1 developed in [2]. However, several changes were made. First of all, the main purpose of the authors of the paper is to explore how the connectivity of the layers inside a Quantum BM would improve the results of the generated samples. Even though they found that in their classical experiments, this was the case, they could not find the same results when moving to the quantum setting. The results they obtained when using a complete topology, in which every node in the BM is connected which each other, were not found to perform better than an RBM with its restricted topology. Thus we only implemented the RBM in the quantum computer, without using different topologies. Our main focus was thus comparing the performance of the Classical QAAAN (using the Classical RBM) and the Quantum QAAAN (using the Quantum RBM).

3.4.3. QAAAN Optimization and Refinement

Upon implementing the Quantum-Assisted Adversarial Network detailed in [2], we encountered several challenges that needed addressing. A notable issue was the communication bottleneck with the quantum computer through the API, which significantly slowed down our processes. Furthermore, D-Wave's systems operate with a restricted usage duration. This constraint necessitated judicious management of the jobs dispatched to the quantum computer, as the most substantial time consumption in quantum annealing arises from problem definition in the quantum annealer. Therefore, we had to strategize our operations meticulously to ensure efficient use of the quantum resources.

In response to this challenge, we devised a strategy that optimized our interaction with the Quantum RBM. Instead of generating a new sample through an API call for each step of the algorithm, we decided to generate a batch of samples from the Quantum RBM in one go, to be used until the next

training step of the Quantum RBM. These pre-sampled values were then used in the subsequent steps of the algorithm. This approach reduced the frequency of API calls, thus reducing the communication bottleneck and enabling more efficient use of the quantum resources. However, this was not enough to increase the number of epochs as much as needed to achieve good results in the training process.

Another strategy we tried involved a significant modification to the algorithm. The concept entailed initially training a GAN in a manner similar to the approaches discussed in previous sections. After a predetermined number of epochs, the training process would be paused, and the RBM would be trained on the feature distribution learned by the discriminator. Upon completing this phase, a new generator would be instantiated, and the GAN training would resume, this time with the generator being fed inputs drawn from the feature representation learned by the RBM. However, this method proved to be highly unstable and often failed to converge, leading us to eventually abandon this approach.

As mentioned, training the RBM at each step proved unfeasible due to the usage restrictions of D-Wave's quantum annealer; it only permitted the full QAAAN to be trained for approximately 200 epochs. This duration fell considerably short of what was required to generate satisfactory results. Consequently, we decided to experiment with training the RBM at intermittent epochs. We discovered that conducting RBM training every 5 steps in the training process enabled us to run the QAAAN algorithm for a more extended number of epochs without impairing the RBM's capacity to learn the distribution of the feature representation extracted from the discriminator. This approach allowed us to train both QAAAN models for 900 epochs, which proved adequate for them to learn most of the distributions we tested.

We made one final improvement to the original algorithm. Thanks to our previous experience when training GANs, we knew that it usually helps the training when the parameter updates on the discriminator are done at a higher rate than the generator. Following this principle, we added some extra flexibility to the algorithm. We created three hyperparameters, each corresponding to the discriminator, RBM and generator parameters updates each training step. We performed a grid search to select the best values for these hyperparameters. The possible values for each hyperparameter were searched in $\{1, 3, 5\}$. We found that the best-performing update ratios were $(5, 1, 1)$ in the case of the Vanilla QAAAN and $(3, 1, 1)$, where the first value represented the update ratio of the discriminator/critic. These results coincided with the ones we obtained when testing the GANs. This modification boosted significantly the performance of the four QAAAN models.

4

Experiments and Results

In this chapter, we dive into a thorough set of experiments designed to evaluate and compare the performance of all the previously developed algorithms. The experiments can be divided into two main categories, the GAN experiments and the QAAAN experiments. Our experiments centred around the task of learning simple one-dimensional distributions: Uniform, Gaussian, Pareto, and Cauchy. These distributions were selected for their diverse properties: the Uniform distribution for its simplicity and equal probability of all outcomes, the Gaussian for its prevalence in natural phenomena, the Pareto for its heavy-tailed property that models phenomena with large outliers, and the Cauchy for its heavy tails and peakedness, often used to model distributions with large variability.

The first set of experiments tested and compared the ability of the first two GANs developed, the Vanilla GAN and the MMD GAN, to learn the one-dimensional distributions. The insights gained from these experiments laid the groundwork for the development of the second set of experiments, focused on testing and comparing the performance of the QAAAN algorithm, developed in [2], and refined as explained in the previous chapter. By training a Restricted Boltzmann Machine (RBM) using quantum annealing and using the generated samples as a prior for the generator, we aimed to exploit the benefits of quantum computing for enhanced performance in learning one-dimensional distributions. The following sections detail our experimental procedure and results.

4.1. Experiments with Classical GANs

The main purpose of this set of experiments was to test and compare the performance of each of the Vanilla and MMD GANs developed following [28], [1] and [22], as described in the previous chapter.

4.1.1. Setup and Procedure for GAN Experiments

The first step of these experiments consisted in fine-tuning the hyperparameters to obtain the best performance out of each of the GANs. In all these preliminary experiments, the target distribution was chosen to be a Gaussian distribution with mean 3 and variance 1 and we measured the final Wassers-

tein distance after the training to choose the best-performing hyperparameters.

In the case of the Vanilla GAN, the main hyperparameters to optimize were the learning rate and the discriminator-to-generator update ratio. During the implementation phase, a preliminary grid search was conducted to find the best-performing hyperparameters. The best-performing learning rate was 10^{-3} and the best discriminator-to-generator update ratio was found to be 5, after searching in the pairs $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\} \times \{1, 2, 3, 4, 5\}$ over 2000 epochs. With these preliminary results, we started the final experimentation phase.

A final hyperparameter optimization was conducted now with 10000 epochs with again target distribution was Gaussian with mean 3 and variance 1. However, the learning rates $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$ were now tested with the best performing discriminator-to-generator update ratio, 5. And the discriminator-to-generator update ratios $\{1, 2, 3, 4, 5\}$ were tested with the best-performing learning rate, 10^{-3}. This resulted again in the best learning rate being 10^{-3} and the best discriminator-to-generator update ratio being 5, which became the hyperparameters used in the final experiments.

In the case of the MMD GAN, a preliminary grid search was performed in the same way as the Vanilla GAN, obtaining that the best-performing learning rate was 10^{-3} while the best critic-to-generator update ratio was now found to be 3. However, the MMD GAN has additional hyperparameters which could also be optimized, including the regularization parameter and the weight clipping parameter. The regularization parameter helps to prevent overfitting and also controls the complexity of the model, while the weight clipping parameter ensures that the weights of the critic remain within a certain range after each update. These hyperparameters were optimized, fixing the learning rate at 10^{-3} and the critic-to-generator update ratio to 3, and searching in $\{10, 5, 1, 10^{-1}, 10^{-2}, 10^{-3}\} \times \{10, 5, 1, 10^{-1}, 10^{-2}, 10^{-3}\}$. The optimal weight clipping parameter was found to be 1, while the optimal regularization parameter was 10^{-2}.

A final hyperparameter optimization was again performed, now with the clipping parameter and regularization parameter set to their optimal values, just before the final experiments in the same way as in the Vanilla GAN case. The optimal learning rate was now found to be 10^{-4} and the best critic-to-generator update ratio was found to be 3. These were the parameters which were used in the final experiments.

The final experiments were conducted on different target distributions. The distributions selected for being learnt were the following:

1.– Uniform distribution with lower bound 1 and upper bound 3.

2.– Gaussian distribution with mean 3 and variance 1.

3.– Cauchy distribution with location parameter 3 and scale parameter 1 (equivalent to the standard deviation in the Gaussian distribution).

4.– Pareto distribution with shape parameter 3 (which determines the size of the tail of the distribution) and scale parameter 1 (which determines the minimum value the distribution can take).

Each GAN was run with the best-performing hyperparameters previously identified. The experiments were run five times for each type of GAN and for each distribution. The number of epochs of the final experiments was 10,000. The results of the experiments can be seen in Figure 4.1 and Table 4.1.

Distribution	Vanilla GAN	MMD GAN
Uniform	0.079	0.027
Gaussian	0.062	0.054
Cauchy	5.278	3.843
Pareto	0.407	0.687

Table 4.1: Resulting Wasserstein Distances obtained in the comparison between Vanilla GAN and MMD GAN. These results correspond with the plots in Figure 4.1

4.1.2. Results for Vanilla GAN

The great outcomes obtained from our experiments with Vanilla GAN were unexpected, particularly when considering the findings presented in the paper by Zaheer et al. [28]. Their research posits that Vanilla GANs struggle with the task of learning simple one-dimensional distributions. However, our results depicted in Figure 4.1 and Table 4.1, show that the Vanilla GANs demonstrated a considerable aptitude for learning all of the studied distributions with a relatively high degree of accuracy.

It's essential to highlight that the experimental setup in Zaheer et al.'s research diverges from ours in a significant aspect: the placement of their distribution masses. While their distributions were generally centred much farther from 0, our distributions typically had a mean value ranging between 2 and 3. These results suggest that Vanilla GANs are capable of learning intricate distributions provided the input data is properly scaled and normalized.

In light of these findings, we ventured to conduct additional experiments, this time positioning the mean of the distributions farther from 0, akin to the setup in Zaheer et al.'s study. We concentrated all the mass of the distributions around 23, in line with their methodology. Our results were similar to those reported in their paper: the Vanilla GAN is not able to learn the distributions accurately. However, we conjecture that a substantial increase in the number of epochs could potentially enable the Vanilla GAN to learn these distributions effectively, even when they are situated far from 0.

4.1.3. Results for MMD GAN

The efficacy of MMD GAN in accurately learning various distributions is clearly seen in Figure 4.1. These results align with the observations made in [28], further validating the concept of using a critic to adversarially learn the kernel employed in the MMD during the training process.

Additionally, in the extended set of experiments where the mass of the distribution is positioned

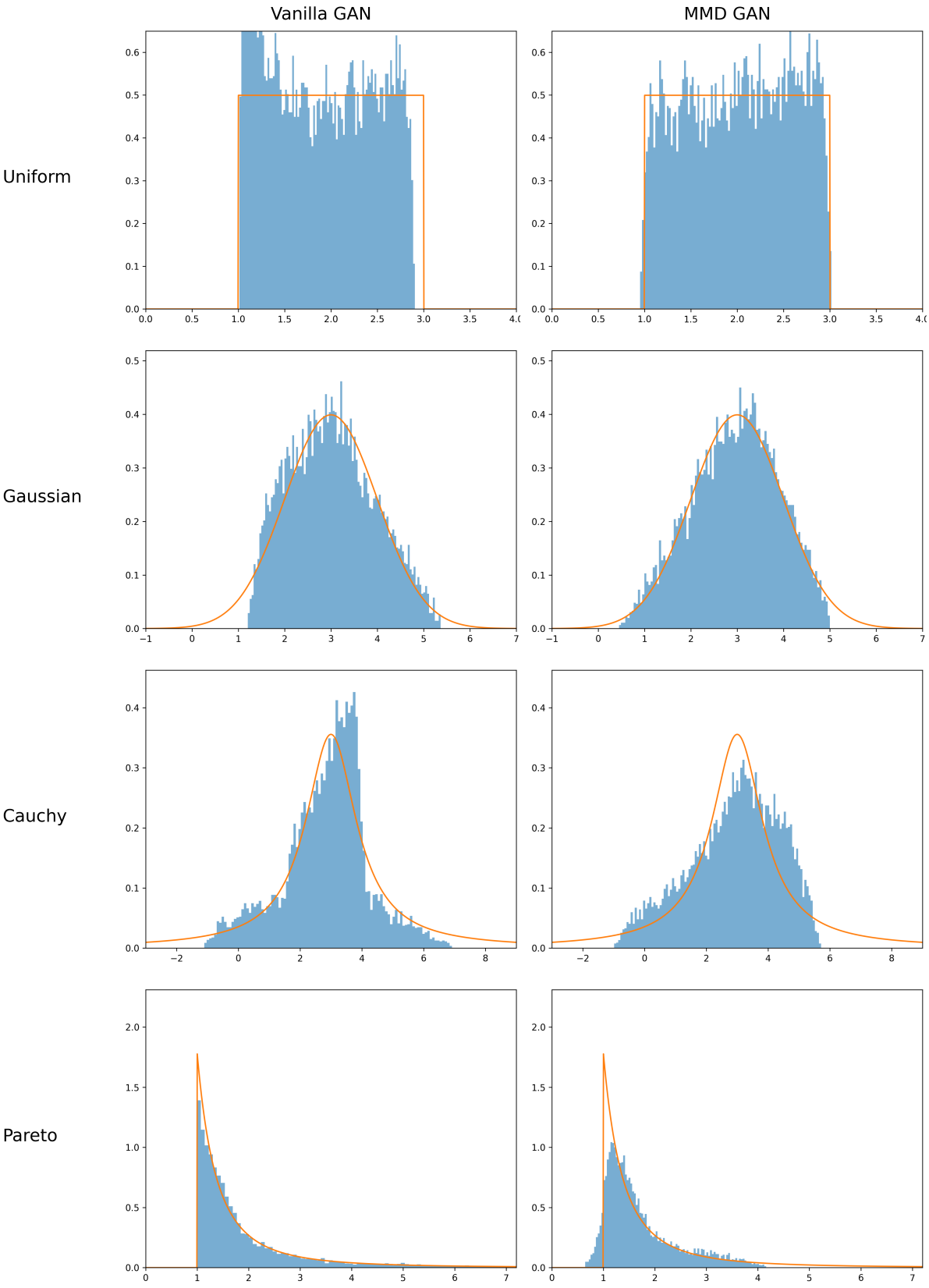

Figure 4.1: Performance of Vanilla GAN vs MMD GAN on different one-dimensional distributions. The experiments were run 5 times for 10,000 epochs, and the best results were selected. The y-axes represent the probability density function (pdf), and the x-axes represent the corresponding values.

farther from 0, specifically around 23, MMD GAN demonstrated its ability to accurately learn the distributions. This shows an improvement over the Vanilla GAN architecture, especially when dealing with distributions that are not centred close to 0. The results further affirm the robustness of MMD GAN in learning and adapting to different distribution characteristics, highlighting its potential for a wide array of applications.

4.1.4. Analysis and Discussion of GAN Experiments

The results from our initial set of experiments provide an intriguing landscape for understanding the capabilities of Vanilla and MMD GANs. Contrary to the assertions from [28], our findings reveal a rather promising performance by the Vanilla GAN architecture. However, it is important to note the difficulties of both GANs in learning the Cauchy distribution.

The Vanilla GAN's proficiency in learning various distributions - given that the data is appropriately scaled and normalized - is noteworthy. This not only introduces a compelling perspective into the abilities of Vanilla GANs but also paves the way for further exploration, specifically in the form of the Vanilla QAAAN. The prospect of comparing the Quantum Vanilla QAAAN against the Quantum MMD GAN allows for a richer and more diverse experimental design, potentially leading to a more nuanced understanding of their respective strengths and weaknesses.

Moreover, our experiments underscore the robustness of the MMD GAN when tasked with learning distributions that are situated far from 0. This characteristic is not as prominent in the Vanilla GAN, thereby establishing a clear distinction in their performance. The MMD GAN's success is primarily attributed to its unique approach in calculating the distance between two distributions. This direct calculation method not only confers a higher level of accuracy but also fosters a more inclusive learning environment, where distributions with varying characteristics can be accurately learnt and reproduced.

This exploratory phase of our research has provided us with important insights into the different GAN architectures, which have been instrumental in shaping the design and direction of our subsequent experiments. It's evident that the choice of GAN architecture, as well as the treatment of the input data, plays a crucial role in the overall performance of the model. Furthermore, it's worth noting that these observations may not be universally applicable - the optimal GAN architecture and hyperparameters can vary significantly depending on the specific task at hand.

In light of these findings, our forthcoming research will delve into the implementation of QAAANs and examine their potential in outperforming their classical counterparts.

4.2. Experiments with QAAAN

The purpose of this set of experiments was to test the performance of the different implementations of the QAAAN algorithm developed by [2], refined and adapted as explained in the previous chapter. The four different QAAANs included: the Classical Vanilla QAAAN which uses the Classical RBM and the Vanilla GAN; the Quantum Vanilla QAAAN, which uses the Quantum RBM and the Vanilla GAN; the Classical MMD QAAAN, which uses the Classical RBM and the MMD GAN; and the Quantum MMD QAAAN, which uses the Quantum RBM and the MMD GAN. All these different architectures were compared against each other, especially focusing on the comparison between the Classical and Quantum Vanilla QAAAN, and the comparison between the Classical and Quantum MMD GAN.

4.2.1. Setup and Procedure for QAAAN Experiments

The preliminary phase of our experiments focused on meticulous fine-tuning of hyperparameters. Given the large number of hyperparameters to optimize, our strategy was to begin with a base set of hyperparameters that had yielded the most effective results for the different GANs and RBMs individually. We then iteratively modified these parameters to optimize the overall performance.

Two primary hyperparameters of interest were the size of the RBM's hidden layer and the size of the discriminator or critic's penultimate layer, which represented the feature representation the RBM was to learn. We employed a grid search over potential sizes, specifically $20, 50, 70$, and found that a configuration of 50 hidden units and a penultimate layer size of 50 delivered the most promising results. This configuration achieved a balance between the learning proficiency of the QAAAN (as this parameter determined the number of epochs of the total training given the limitations of the D-Wave's hardware usage) and the RBM's ability to learn the extracted features from the discriminator or critic.

We also paid considerable attention to optimizing the RBM training rate, which determined the frequency of RBM training in relation to epochs. We examined this parameter over the set $\{1, 3, 4, 5, 7, 10\}$ and determined that a setting of 5 yielded optimal results, balancing the RBM's learning effectiveness and the constraints imposed by D-Wave's hardware usage.

We also explored an additional hyperparameter that would dictate the epoch at which the RBM's training would cease, allowing the generator to settle with the distribution learned by the RBM. However, we observed significant instability and non-convergence following this cutoff epoch, leading us to exclude this parameter from our final setup.

Another essential aspect of our fine-tuning process involved optimizing the learning rates of both the GAN and the RBM. We examined values within the set $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$ and discovered that disparate learning rates for the RBM and GAN resulted in unstable training. The most effective learning rate was found to be 10^{-4} for both the Vanilla and the MMD GAN. Larger learning rates tended

to lead to non-convergence.

The final stage of our experiments was executed similarly to previous GAN tests. We ran each of the four QAAANs five times for each of the four distributions, selecting the best result for our final results. The comparative performance of the Classical and Quantum Vanilla QAAAN can be observed in Figure 4.2. Similarly, Figure 4.3 presents the comparison between the Classical and Quantum MMD QAAAN.

Distribution	Classical Vanilla QAAAN	Quantum Vanilla QAAAN
Uniform	0.224	0.246
Gaussian	0.203	0.186
Cauchy	2.846	3.106
Pareto	0.698	0.659

Table 4.2: Resulting Wasserstein Distances obtained in the comparison between Classical and Quantum Vanilla QAAAN.These results correspond with the plots in Figure 4.2.

Distribution	Classical MMD QAAAN	Quantum MMD QAAAN
Uniform	0.557	0.348
Gaussian	1.264	0.571
Cauchy	5.71	3.721
Pareto	0.783	0.996

Table 4.3: Resulting Wasserstein Distances obtained in the comparison between Classical and Quantum MMD QAAAN. These results correspond with the plots in Figure 4.3.

4.2.2. Results for Vanilla QAAAN

Our experiments with the Vanilla QAAAN yielded promising results, as depicted in Figure 4.2 and Table 4.2. Despite the limited number of epochs used during the training process, both the Classical and Quantum variants of Vanilla QAAAN demonstrated remarkable proficiency in learning one-dimensional distributions. This performance exceeded expectations and was indicative of the robustness of the Vanilla QAAAN architecture.

A comparative evaluation of the Classical and Quantum variants exhibited a nuanced pattern of strengths. The Quantum QAAAN excelled in learning the Gaussian and Pareto distributions, surpassing the performance of its Classical counterpart. Conversely, the Classical QAAAN showcased superior performance in the case of the Cauchy distribution. Both versions demonstrated commendable performance when tasked with Uniform distribution.

These findings underscore the potential of both Classical and Quantum Vanilla QAAAN in learning diverse one-dimensional distributions, with certain advantages depending on the specific distribution in question.

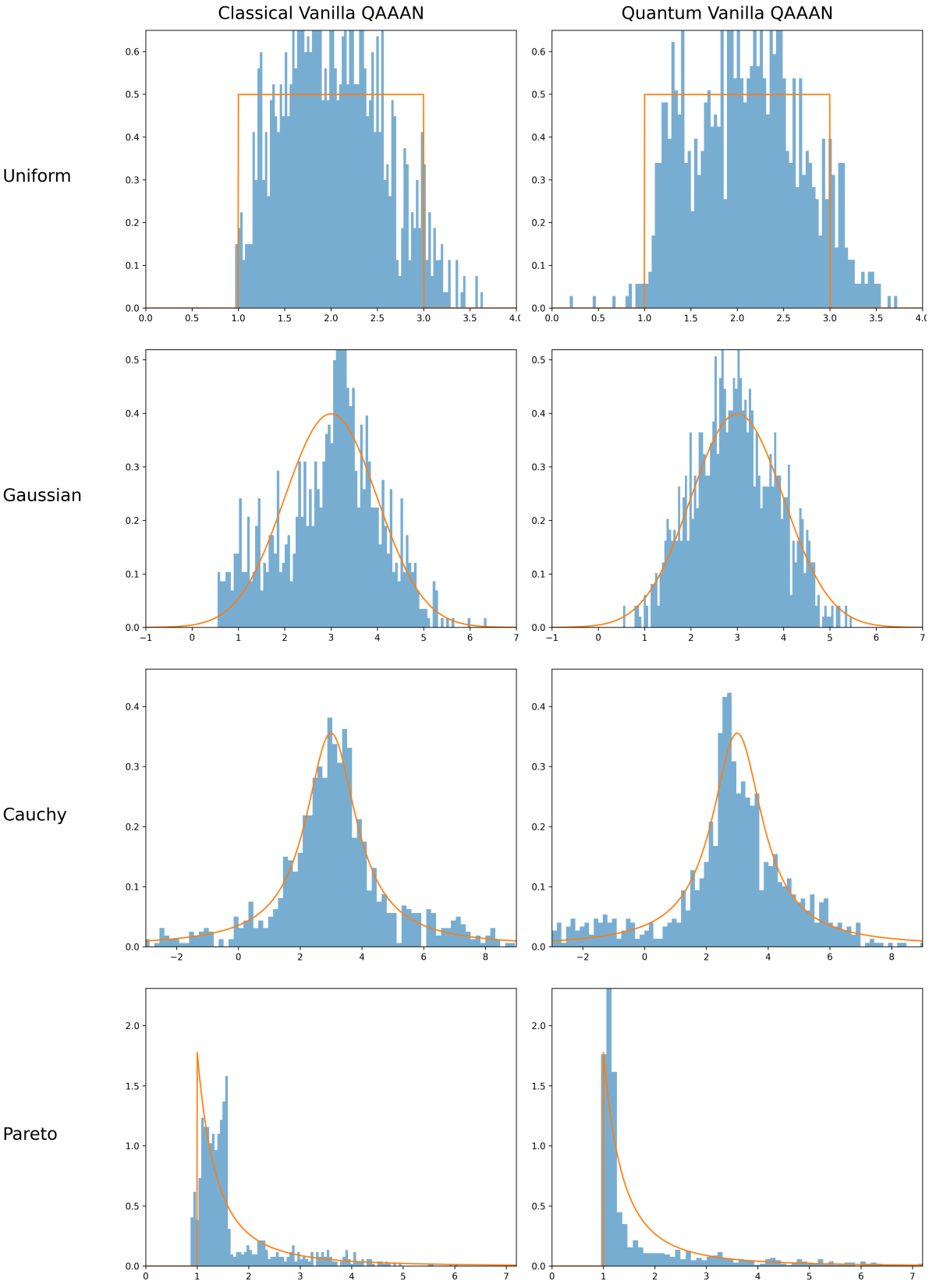

Figure 4.2: Performance of Classical vs Quantum Vanilla QAAAN on different one-dimensional distributions. The experiments were run 5 times for 900 epochs, and the best results were selected. The y-axes represent the probability density function (pdf), and the x-axes represent the corresponding values.

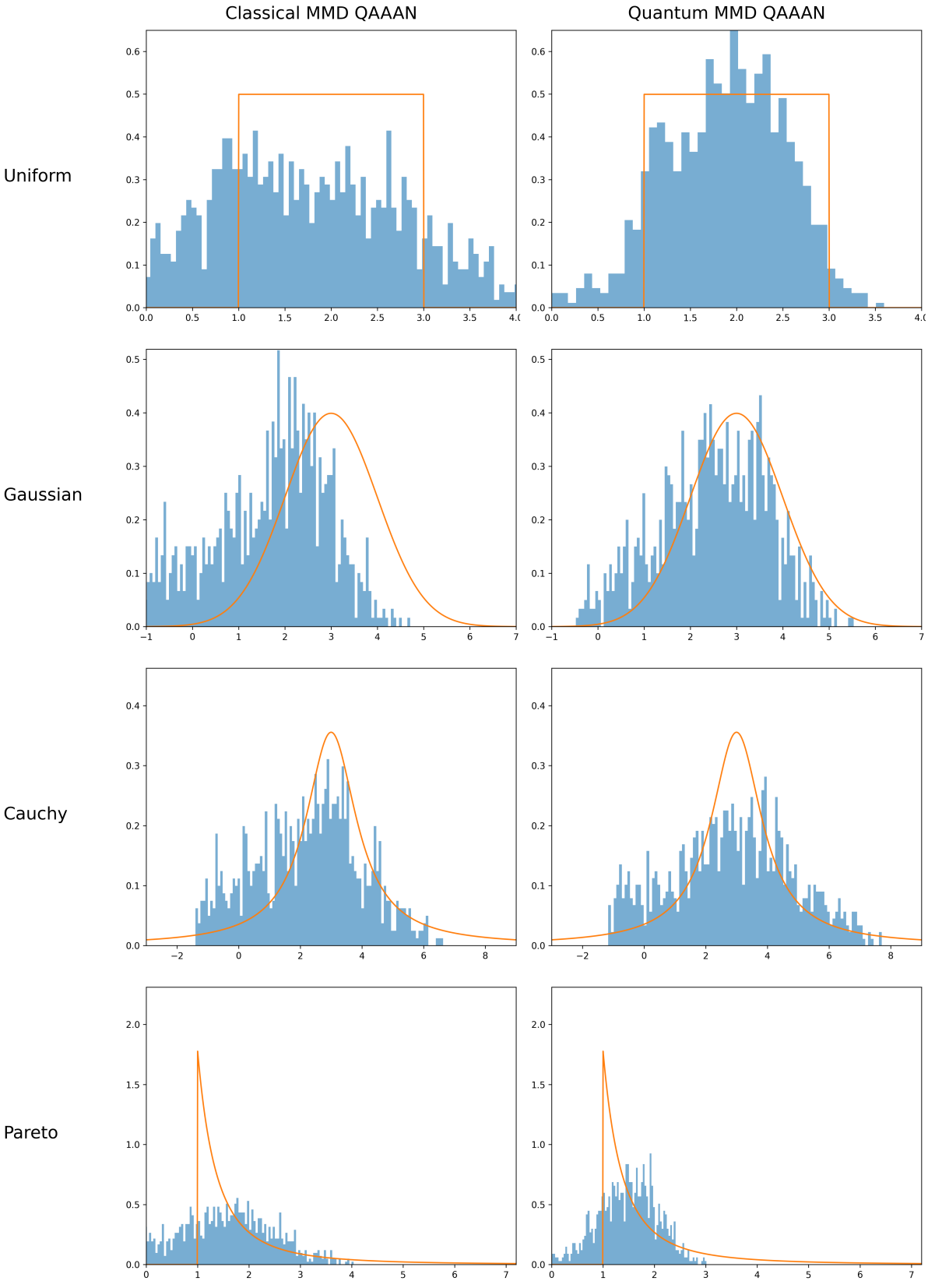

Figure 4.3: Performance of Classical vs Quantum MMD QAAAN on different one-dimensional distributions. The experiments were run 5 times for 900 epochs, and the best results were selected. The y-axes represent the probability density function (pdf), and the x-axes represent the corresponding values.

4.2.3. Results for MMD QAAAN

The outcomes of the MMD QAAAN experiments are summarized in Figure 4.3 and Table 4.3. Contrary to the Vanilla QAAAN results, the performance of MMD QAAAN was less encouraging. The Classical version of MMD QAAAN exhibited subpar performance across all tested distributions, pointing to a potential shortcoming in its ability to learn diverse distributions effectively.

On the other hand, the Quantum MMD QAAAN showcased notable performance with the Gaussian distribution and yielded satisfactory results in the Uniform case. However, it fell short when tasked with learning distributions characterized by large tails such as the Cauchy and Pareto distributions. In these instances, Quantum MMD QAAAN managed to match only the first-order moment with the target distributions, indicating a limited understanding of the underlying distributions' complexities.

A plot of the loss functions over the training period provided additional insights into the learning dynamics of the Quantum MMD QAAAN. The absence of significant learning in the final epochs suggests the presence of a performance bottleneck in the architecture. Importantly, this bottleneck does not appear to be attributable to the limited number of training epochs.

These results prompt further investigation into the inherent limitations of the MMD QAAAN architecture and the potential strategies to overcome these limitations for better performance.

4.2.4. Analysis and Discussion of QAAAN Experiments

The comprehensive series of experiments conducted offered valuable insights into the relative performances of the Vanilla and MMD QAAANs. Overall, the Vanilla QAAAN exhibited superior performance compared to the MMD QAAAN. This can be attributed to the manner in which each model's discriminator or critic learns and represents the features of the data.

The discriminator in the Vanilla GAN learns a feature representation that is directly informed by the data, creating a more accurate and detailed representation. Conversely, the critic in the MMD QAAAN learns to differentiate between real and generated data through various kernels, aiming to maximize the MMD distance associated with the learned kernel between real and generated data. Although the information contained within the penultimate layer of the critic may offer valuable insight to be used as a prior for the generator, it might not be as directly informative as the feature representation learned by the discriminator of the Vanilla GAN.

The Vanilla QAAAN demonstrated good performance in both the Classical and Quantum cases. Notably, the Quantum Vanilla QAAAN even surpassed the performance of the Classical Vanilla QAAAN in certain scenarios. This not only underscores the robustness of the Quantum Vanilla QAAAN but also emphasizes the potential advantages of quantum computing in complex machine learning tasks.

However, despite these promising results, it's essential to consider that neither variant was uniformly

superior across all tested distributions. This suggests that different GAN architectures may be better suited to different types of distributions and that it could be beneficial to tailor the architecture to the specific distribution at hand.

Future work could focus on uncovering the underlying reasons for the observed performance differences, investigating possible enhancements to the MMD QAAAN, and further exploring the potential of Quantum QAAANs in learning more complex or higher-dimensional distributions. This could further our understanding of the potential and limitations of quantum-enhanced machine learning models and provide a foundation for their practical application.

4.3. Final Comparison of GANs and QAAANs

A final round of experiments was conducted to draw a comparative analysis between the performance of Vanilla and MMD QAAAN algorithms against the Vanilla and MMD GAN algorithms. To ensure a fair comparison, we adjusted the number of epochs of the original GAN algorithms to match the 900 epochs of the QAAAN algorithms.

Our standard experimental protocol was employed, in which each algorithm was run five times, and the best run was selected for analysis. The hyperparameters deployed in this set of experiments were the ones that demonstrated optimal performance in previous sections; no additional fine-tuning was performed. For the Quantum QAAANs, the experiments were not reiterated; instead, the results from the previous sections were utilized in the comparative figures and tables.

Tables 4.4 and 4.5, and figures A.1 and A.2 (provided in the appendix) illustrate the outcomes of these comparative experiments. Table 4.4, associated with Figure A.1, contrasts the performance of the Vanilla GAN against the Vanilla QAAAN. Table 4.5, associated with Figure A.2, contrasts the performance of the MMD GAN against the MMD QAAAN.

Notably, the Quantum Vanilla QAAAN slightly outperformed the Vanilla GAN in terms of Wasserstein distances. Further, a significant disparity was observed in the shape of the generated distributions. The QAAAN seemed to possess a superior ability to learn the form of the distribution, particularly the tails representing low-probability outcomes, which can be observed in Figure A.1. In contrast, the Vanilla GAN struggled to accurately capture these low-probability samples from the distribution.

We hypothesize that this performance edge of the QAAAN could be attributed to the informed prior provided to the discriminator. This likely facilitates a more comprehensive sampling of the distribution, including the less probable outcomes, thereby avoiding under-sampling issues that the Vanilla GAN seems to encounter.

Lastly, while the MMD QAAAN's performance was not entirely satisfactory in learning certain distributions—often only matching first-order moments, as it can be seen in Figure A.2—it's worth noting that

Distribution	Vanilla GAN	Vanilla QAAAN
Uniform	0.271	0.246
Gaussian	0.209	0.186
Cauchy	4.551	3.106
Pareto	0.701	0.659

Table 4.4: Resulting Wasserstein Distances obtained in the comparison between Vanilla GAN and Quantum Vanilla QAAAN. These results correspond with the plots in Figure A.1.

Distribution	MMD GAN	MMD QAAAN
Uniform	0.073	0.348
Gaussian	0.359	0.571
Cauchy	4.432	3.721
Pareto	0.858	0.996

Table 4.5: Resulting Wasserstein Distances obtained in the comparison between MMD GAN and Quantum MMD QAAAN. These results correspond with the plots in Figure A.2.

the integration of an informed prior into the MMD GAN did not appear to detract from the algorithm's performance when looking at Figure A.2. We hypothesize that with an increase in training epochs, the MMD QAAAN could potentially match, if not surpass, the simple MMD GAN's capacity to learn different distributions.

However, the overall benefit of integrating another generative model to learn a feature representation from the critic seems to be less substantial than anticipated. This could be attributed to the critic's feature representation in the MMD QAAAN being less informative than the discriminator's feature representation in the Vanilla GAN.

5 Conclusions and Future Work

In this concluding chapter, we reflect upon the findings of our research and discuss their implications. Furthermore, we outline promising avenues for future research, intending to expand upon the discoveries made through the implementation and evaluation of QAAANs.

5.1. Conclusions

The project was conducted in two distinct stages, each contributing significantly to our overall understanding and final conclusions.

In the initial stage, our efforts were directed towards the implementation of Vanilla and MMD GANs. This stage served as an exploratory phase, designed to investigate the capacity of different GANs to learn and reproduce samples from one-dimensional distributions. By doing so, we gained invaluable insights into the operation of GANs, their effectiveness in learning one-dimensional distributions, and established the foundational knowledge required for the subsequent development of the QAAAN algorithm.

Transitioning into the second stage, our primary objective was the modification of the QAAAN algorithm, as initially developed in [2], to facilitate learning one-dimensional distributions. To accomplish this, we first developed both Classical and Quantum RBMs, subsequently integrating these with the GANs from the first stage to construct the QAAAN algorithms. This process culminated in the creation of four unique models, representing the different combinations of Classical/Quantum RBMs and Vanilla/MMD GANs. These models provided the critical foundation for our ultimate comparative study between classical and quantum algorithms.

The chief objective of this project was to conduct a performance comparison of Quantum Assisted Models against their classical counterparts. In order to facilitate this, we constructed a series of foundational models, beginning with the classical RBM. This, in turn, enabled the creation of two classical QAAAN algorithms, establishing a valuable benchmark for comparison against the quantum QAAAN algorithms.

An integral part of our investigation included a comparative analysis between the foundational GANs and the resultant quantum QAAANs. This comparison illuminated the benefits of using an informed prior for the generator, which served to enhance the overall performance of the models.

Importantly, we were successful in adapting the QAAAN algorithm to be compatible with the usage constraints of D-Wave's hardware, without compromising the RBM's ability to learn the feature layer distribution. This adaptation allowed the Vanilla QAAAN to sample from low probability regions of the distributions, which consequently improved the model's performance. In some instances, the Quantum Vanilla QAAAN outperformed both the Classical Vanilla QAAAN and the original Vanilla GAN. It is important to note though, that this required the overhead of training an additional RBM to learn a prior for the GAN when compared to the original GAN algorithm.

However, our experiments with the MMD QAAAN suggested that it did not improve upon the performance of the MMD GAN. This led us to hypothesize that the feature representation learned by the critic might not be as directly informative as the one learned by the discriminator in the Vanilla GAN, thereby resulting in less satisfactory outcomes for the MMD QAAAN.

Our early encounters with classical algorithms shed light on the capabilities and limitations of simple GANs, such as the Vanilla GAN implemented in our study. We found that Vanilla GANs could learn one-dimensional distributions effectively when the distribution's mass was not overly distanced from zero. However, their performance faltered when tasked with learning distributions where the mass was positioned significantly far from zero, highlighting the necessity of alternative approaches. In contrast, the MMD GAN demonstrated its capability to learn distributions that are far removed from zero, emphasising its versatility in handling a broader range of distributions.

Our study provides valuable insights into the advantages and potential limitations of Quantum Assisted Models in comparison to their classical counterparts. The superior performance of the Quantum Vanilla QAAAN, particularly in its ability to sample from low-probability regions of distributions, underscores the potential of leveraging quantum algorithms in capturing the complexity of distributions. This performance may be attributed to the integration of an informed prior into the generator's model, enhancing the model's efficacy, and the potential quantum advantages such as quantum tunnelling, which could contribute to more efficient exploration of the feature space.

5.2. Future Work

This project has illuminated a myriad of potential future research and development opportunities, some of which are outlined below.

One of the key areas for future investigation could be the exploration of different quantum algorithms for learning priors. In this project, we utilized Quantum RBMs to learn the prior distributions. However,

other quantum algorithms such as Quantum Principal Component Analysis (QPCA) or Quantum Auto-encoders could potentially be employed to learn these priors more effectively or offer different insights.

Another promising direction for future work is the extension of the current methodology to learning two-dimensional and higher-dimensional distributions. This project was focused on one-dimensional distributions; however, real-world problems are inherently multi-dimensional. Therefore, extending these models to encompass higher dimensions represents a natural progression for this work. This will likely introduce additional complexities and challenges which will necessitate careful exploration and understanding.

Further modifications and enhancements to the QAAAN algorithm could also be investigated. This could encompass variations to the architecture of the Quantum RBM, experimentation with different types of GANs, or exploration of the impacts of different training methodologies.

Lastly, this project has highlighted several opportunities for further research in the field of Quantum Assisted Models. For instance, a deeper understanding of why Quantum Assisted Models perform better in some cases but not in others could be explored. This could lead to more strategic use of quantum resources and help determine when it is beneficial to utilize Quantum-Assisted Models over their classical counterparts.

In conclusion, while the results of this project are promising, they represent just the initial steps in the vast landscape of potential achievements with Quantum-Assisted Models. As exploration and innovation within this field continue, we anticipate exciting advancements that will further push the boundaries of what is currently possible.

Bibliography

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, vol. 27, p. 2672–2680, Curran Associates, Inc., 2014.

[2] M. Wilson, T. Vandal, T. Hogg, and E. Rieffel, "Quantum-assisted associative adversarial network: applying quantum annealing in deep learning," *Quantum Machine Intelligence*, vol. 3, 06 2021.

[3] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*. Cambridge, U.K.: Cambridge University Press, 10th ed., 2010.

[4] S. Pirandola, J. Eisert, C. Weedbrook, A. Furusawa, and S. L. Braunstein, "Advances in quantum teleportation," *Nature Photonics*, vol. 9, pp. 641–652, 10 2015.

[5] R. Shankar, *Principles of Quantum Mechanics*. New York, NY, USA: Springer US, 1994.

[6] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM journal on computing*, vol. 26, pp. 1484–1509, 10 1997.

[7] L. K. Grover and B. W. Reichardt, "Quantum search," in *Encyclopedia of Algorithms* (M. Y. Kao, ed.), pp. 712–715, Boston, MA, USA: Springer US, 2008.

[8] T. Kadowaki and H. Nishimori, "Quantum annealing in the transverse ising model," *Phys. Rev. E*, vol. 58, pp. 5355–5363, 11 1998.

[9] E. Farhi, J. Goldstone, S. Gutmann, and M. Sipser, "Quantum computation by adiabatic evolution," *arXiv preprint arXiv:quant-ph/0001106*, 2000.

[10] M. Li, S. Wang, S. Fang, and J. Zhao, "Anomaly detection of wind turbines based on deep small-world neural network," *Applied Sciences*, vol. 10, no. 4, 2020.

[11] K. Kurowski, M. Slysz, M. Subocz, and R. Różycki, "Applying a quantum annealing based restricted boltzmann machine for mnist handwritten digit classification," *CMST*, vol. 27, no. 3, pp. 99–107, 2021.

[12] D. Vint, M. Anderson, Y. Yang, C. Ilioudis, G. Di Caterina, and C. Clemente, "Automatic target recognition for low resolution foliage penetrating sar images using cnns and gans," *Remote Sensing*, vol. 13, p. 596, 02 2021.

[13] S. Yu, K. Zhang, C. Xiao, J. Z. Huang, M. J. Li, and M. Onizuka, "Hsgan: Reducing mode collapse in gans by the latent code distance of homogeneous samples," *Computer vision and image understanding*, vol. 214, p. 103314, 01 2022.

[14] R. Durall, A. Chatzimichailidis, P. Labus, and J. Keuper, "Combating mode collapse in gan training: An empirical analysis using hessian eigenvalues," in *VISIGRAPP*, 2020.

[15] H. Thanh-Tung and T. Tran, "On catastrophic forgetting and mode collapse in generative adversarial networks," *arXiv preprint arXiv:1807.04015*, 2018.

[16] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, and X. Chen, "Improved techniques for training gans," in *Advances in Neural Information Processing Systems* (D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, eds.), vol. 29, Curran Associates, Inc., 2016.

[17] L. Mescheder, A. Geiger, and S. Nowozin, "Which training methods for gans do actually converge?," *arXiv preprint arXiv:1801.04406*, 2018.

[18] M. Arjovsky and L. Bottou, "Towards Principled Methods for Training Generative Adversarial Networks," *arXiv preprint arXiv:1701.04862*, 01 2017.

[19] Z. Li, P. Xia, R. Tao, H. Niu, and B. Li, "A new perspective on stabilizing gans training: Direct adversarial training," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 7, pp. 1–12, 02 2023.

[20] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[21] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," *arXiv preprint arXiv:1701.07875*, 2017.

[22] S. Eneji, C.-L. Li, W. Ibe, E. Eyong, M. Angib, and Udie, "Mmd gan: Towards deeper understanding of moment matching network," *International Journal of Information and Communication Technology*, p. 3, 06 2020.

[23] P. Rebentrost, M. Mohseni, and S. Lloyd, "Quantum support vector machine for big data classification," *Physical Review Letters*, vol. 113, 07 2014.

[24] S. Lloyd, M. Mohseni, and P. Rebentrost, "Quantum principal component analysis," *Nature Physics*, vol. 10, no. 9, pp. 631–633, 2014.

[25] E. Farhi, J. Goldstone, S. Gutmann, and L. Zhou, "The quantum approximate optimization algorithm and the sherrington-kirkpatrick model at infinite size," *Quantum*, vol. 6, p. 759, 07 2022.

[26] L. Gyongyosi and S. Imre, "A survey on quantum computing technology," *Computer science review*, vol. 31, pp. 51–71, 02 2019.

[27] J. Liu, F. Spedalieri, K.-T. Yao, T. Potok, C. Schuman, S. Young, R. Patton, G. Rose, and G. Chamka, "Adiabatic quantum computation applied to deep learning networks," *Entropy (Basel, Switzerland)*, vol. 20, p. 380, 05 2018.

[28] M. Zaheer, C.-L. Li, B. Póczos, and R. Salakhutdinov, "Gan connoisseur: Can gans learn simple 1d parametric distributions?," in *Proceedings of the 31st Conference on Neural Information Processing Systems*, (Red Hook, NY, USA), pp. 1–6, Curran Associates Inc., 2021.

[29] M. G. Genton, "Classes of kernels for machine learning: A statistics perspective," *J. Mach. Learn. Res.*, vol. 2, p. 299–312, 03 2002.

Acronyms

AANs Associative Adversarial Networks.

BM Boltzmann Machine.

BMs Boltzmann Machines.

CD Contrastive Divergence.

DCGAN Deep Convolutional GAN.

ELU Exponential Linear Unit.

GAN Generative Adversarial Network.

GANs Generative Adversarial Networks.

GMMNs Generative Moment Matching Networks.

MCMC Markov chain Monte Carlo.

MMD Maximum Mean Discrepancy.

MMD GAN Maximum Mean Discrepancy GAN.

MMD GANs Maximum Mean Discrepancy GANs.

MMD QAAAN Maximum Mean Discrepancy QAAAN.

PCA Principal Component Analysis.

QAAAN Quantum-Assisted Associative Adversarial Network.

QAAANs Quantum-Assisted Associative Adversarial Networks.

QAOA Quantum Approximate Optimization Algorithm.

QPCA Quantum Principal Component Analysis.

QSVM Quantum Support Vector Machine.

QUBO Quadratic Unconstrained Binary Optimization.

RBM Restricted Boltzmann Machine.

RBMs Restricted Boltzmann Machines.

ReLU Rectified Linear Unit.

WGAN Wasserstein GAN.

APPENDICES

A

GAN vs Quantum QAAAN

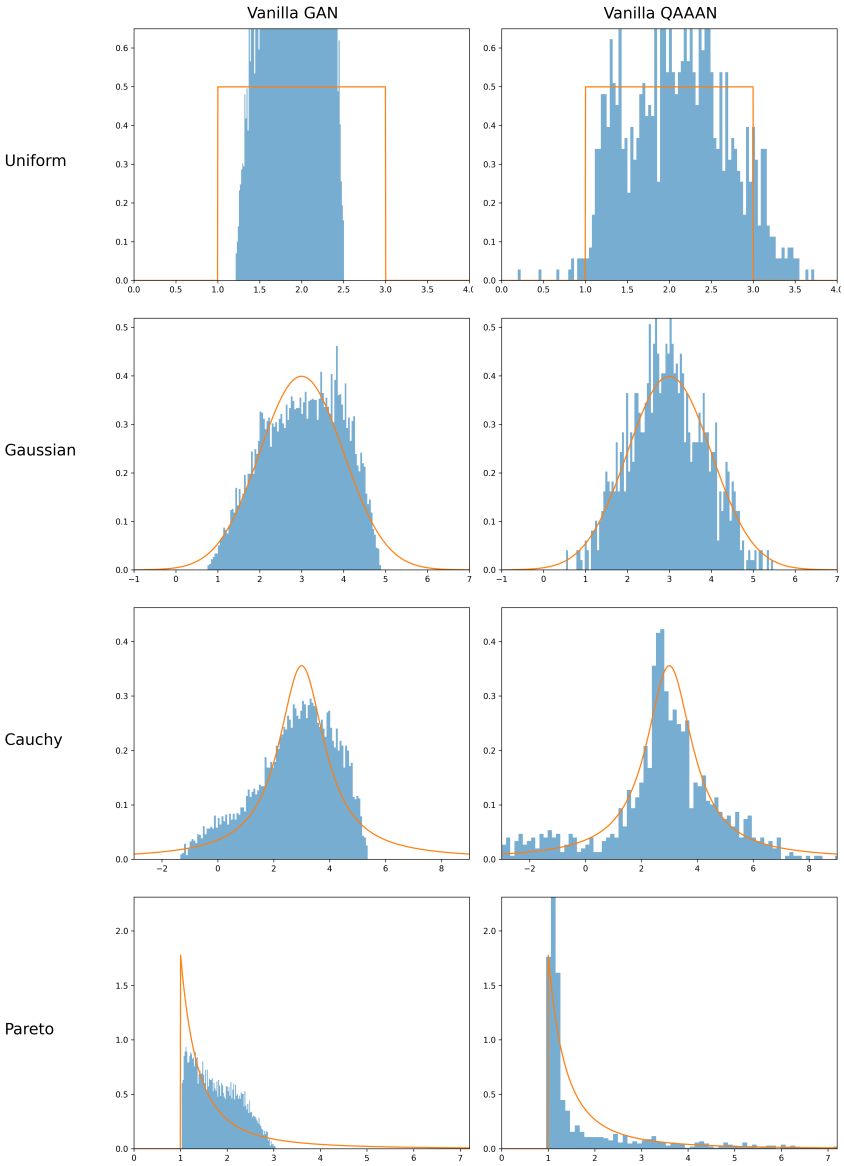

Figure A.1: Performance of Vanilla GAN vs Quantum Vanilla QAAAN on different one-dimensional distributions. The experiments were run 5 times for 900 epochs, and the best results were selected. The y-axes represent the probability density function (pdf), and the x-axes represent the corresponding values.

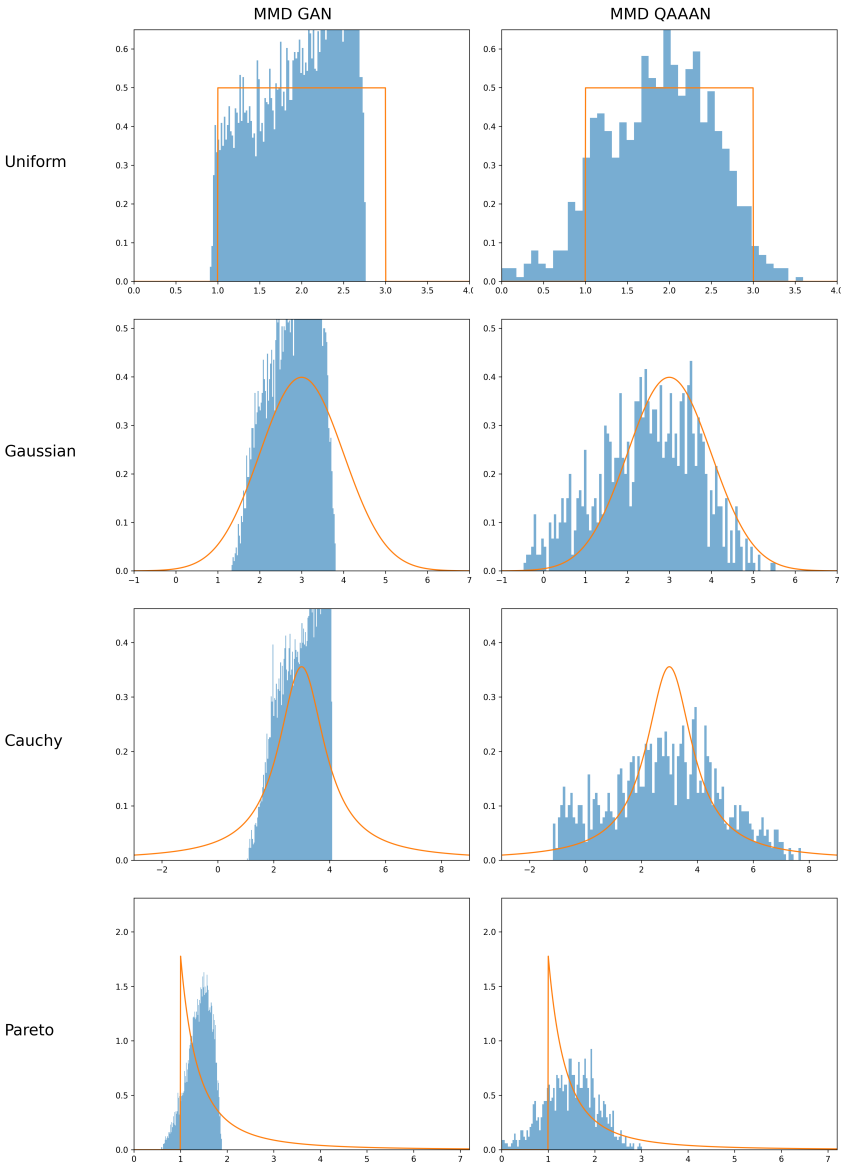

Figure A.2: Performance of MMD GAN vs Quantum MMD QAAAN on different one-dimensional distributions. The experiments were run 5 times for 900 epochs, and the best results were selected. The y-axes represent the probability density function (pdf), and the x-axes represent the corresponding values.


UÁM
Universidad Autónoma
de Madrid