Quantum Machine Learning for Enhanced Intrusion Detection

Author:

Ameel Valjee

Submitted to the Department of Computer Science at the University of Cape Town in partial fulfillment of the academic requirements for a Master of Science degree by coursework and dissertation in Computer Science

May 30, 2025

Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used the **BibTeX** convention for citation and referencing. Each contribution to, and quotation in, this essay/report/project/**Thesis** from the work(s) of other people has been attributed, and has been cited and referenced. Any section taken from an internet source has been referenced to that source.
3. This essay/report/project/**Thesis** is my own work, and is in my own words (except where I have attributed it to others).
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as his or her own work.
5. I acknowledge that copying someone else's assignment or essay, or part of it, is wrong, and declare that this is my own work.

Ameel Valjee
Signature:

Signed by candidate

Date: August 14, 2025

Acknowledgments

I would like to express my deepest gratitude to my co-supervisor, Prof. Ilya Sinayskiy, for his invaluable guidance, expertise, and consistent support throughout this project. His passion and dedication were not only motivational but character shaping. I also extend my appreciation to Prof. Tommie Meyer for his project oversights, assistance, and advice. I am thankful to my friends and family for their unwavering support, encouragement, and understanding throughout this journey. The support of the South African Department of Science, Technology and Innovation (DSTI), the South African Quantum Technology Initiative (SA QuTI), and the Centre for AI Research (CAIR) towards this research is hereby acknowledged.

Abstract

A vital component in ensuring network security is an Intrusion Detection System (IDS), and classical Machine Learning (ML) models often struggle to keep pace with sophisticated cyber threats. Quantum Computing, a probabilistic computational class utilising the principles of quantum mechanics, has shown its potential to address some of these challenges through quantum parallelism and entanglement. In this work, we explore the use of Quantum Machine Learning (QML), specifically Quantum Convolutional Neural Network (QCNN) and Quantum Support Vector Machine (QSVM), to determine the benefits of using quantum kernels over classical kernels. We investigate multiple ansatz designs, and entanglement patterns, that generate sufficient representations of vectors in the Hilbert space to distinguish normal from malicious activity. The results have shown improvements in both models compared to their classical counterparts, with QSVM being the best performing when circuit block entanglement is used.

Contents

List of Figures

List of Tables

Abbreviations

BQP	Bounded-Error Quantum Polynomial Time
CNOT	Controlled NOT
CRX	Controlled Rotation-X
CRZ	Controlled Rotation-Z
IDS	Intrusion Detection System
KL	Kullback-Leibler
ML	Machine Learning
NISQ	Noisy Intermediate-Scale Quantum
P	Polynomial Time
PCA	Principal Component Analysis
PQC	Parameterised Quantum Circuit
QCNN	Quantum Convolutional Neural Network
QML	Quantum Machine Learning
QNN	Quantum Neural Network
QSVM	Quantum Support Vector Machine
QTM	Quantum Turing Machine
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
VQC	Variational Quantum Circuit

Chapter 1

Introduction

1.1 Intrusion Detection

As cyber attacks become more sophisticated, there is an increasing need to explore alternative methods for detection and prevention. Intrusion detection is the crux of modern cybersecurity, serving as a primary mechanism for identifying unauthorised access or malicious activity within computer systems and networks. As organisations increasingly rely on interconnected systems to store, process, and transmit sensitive information, the potential attack scope has expanded significantly, leading to a growing need for robust and adaptive intrusion detection mechanisms [1].

The purpose of an Intrusion Detection System (IDS) is to monitor network traffic, system logs, and user activities to identify signs of potential security breaches, whether they are the result of external attackers or insider threats. By analysing patterns and anomalies in the network data stream, IDSs detect attacks such as malware, phishing, distributed denial-of-service. This is essential to minimise the impact of cyberattacks, preserving the confidentiality, integrity, and availability of critical assets.

IDSs predominantly makes use of two methods: signature-based detection and anomaly-based detection. Signature-based IDSs rely on predefined patterns of known threats, making them effective against well-known attacks but limited in identifying novel or evolving threats. Anomaly-based IDSs, on the other hand, use statistical models, ML algorithms, and behavioural analysis to detect deviations from normal activity, offering a more dynamic and adaptive solution for emerging threats. Continuous advances in ML have further enhanced the capability of anomaly-based IDSs to allow for the highly accurate and real-time detection of sophisticated attacks.

1.2 Machine Learning

In recent years, computational algorithms have undergone remarkable advancements, with artificial intelligence standing out as a forefront innovation. Machine learning, a type of artificial intelligence, uses a wide range of algorithms designed to extract meaningful insights from data. These algorithms aim to extract patterns and relationships from the training data that can be applied to new data. Without the ability to generalise, a model would simply memorise the training data, rendering it ineffective for practical applications. This lack of generalisation is referred to as overfitting.

Machine learning can be broadly categorised into three main approaches: supervised learning, unsupervised learning, and reinforcement learning [2]. Supervised learning involves training algorithms on labelled datasets, where both input features and corresponding outputs are provided, allowing the algorithm to learn predictive relationships. In contrast, unsupervised learning operates without labelled output, focussing instead on detecting patterns or structures within the data, such as clustering similar data points. Reinforcement learning, distinct from the other two, involves learning through an iterative process of decision-making, where feedback in the form of rewards or penalties helps refine the algorithm's performance over time. This thesis will primarily focus on supervised learning.

1.3 Quantum Computing

Classical computing is based on binary bits (1 and 0 s) to perform calculations. However, the physical limits of transistor miniaturisation and the computational inefficiencies of solving complex problems in a feasible time frame motivate the search for alternative approaches to computation [3].

Quantum computing introduces a novel way in which information can be processed. Quantum mechanics, the study of the behaviour of particles, reveals counterintuitive phenomena such as superposition, entanglement, and interference. Richard Feynman initially proposed that these phenomena provide the basis for a new computational model [3].

In a two-level system, quantum computing replaces these binary bits with qubits, or quantum bits, which are the most prevalent quantum information carriers in quantum

systems. Although there alternative representations such as qudits (multi-level generalisations of qubits) and continuous-variable states, this thesis will focus solely on qubit systems [4]. Qubit systems make use of the unique properties of quantum mechanics (superposition, entanglement, and interference), allowing quantum algorithms to explore solution spaces more effectively than their classical counterparts. For instance, on a fault-tolerant quantum computer, Shor's Algorithm can achieve exponential speed-up in factoring large numbers, which has serious implications for cryptography by potentially breaking widely used RSA encryption schemes [5]. Another example is Grover's Algorithm, which provides a quadratic speed-up for unstructured search problems, showing faster solutions to tasks like database searching [6].

Despite its potential, quantum computing is not without challenges. Physical implementations of quantum computers are complex and sensitive to their environments. Qubits are prone to errors caused by decoherence (loss of quantum information) and noise (interference from the surrounding environment due to quantum circuits not being closed systems). Current quantum devices, often referred to as Noisy Intermediate-Scale Quantum (NISQ) computers, usually have fewer than 100 qubits and are limited in their ability to perform reliable large-scale calculations [7]. Industrially, various unique physical systems are used to represent and manipulate qubits:

Superconducting quantum computers are among the most researched and widely studied implementations. In this architecture, qubits are formed using superconducting circuits based on Josephson junctions that allow electrical currents to flow without resistance. Quantum gates are implemented by applying carefully tuned microwave pulses, which manipulate the qubits by inducing transitions between discrete energy levels in the superconducting circuit [8]. Notable examples of superconducting computers include IBM's quantum processors [9] as well as Google's Sycamore processor [10] and the Willow chip [11]. However, in pursuit of the creation of fault-tolerant quantum computers, Microsoft has made promising advancements with their Majorana 1 chip [12]. Unlike standard superconducting qubits, Majorana-based qubits do not fundamentally rely on Josephson junctions. Instead, they are based on topological superconductivity (hybrid superconducting-semiconductor systems), where quantum information is stored in collective states (Majorana zero modes) and correlations between the states, which are much more noise resistant at a local level.

In trapped ion systems, qubits are represented by the quantum states of individual ions (charged atoms). These ions are held in place using electromagnetic fields within vacuum chambers. Quantum operations are performed by manipulating the ions with light stimuli, which induce changes in their quantum states [13]. Companies such as IonQ [14]

and Honeywell [15] are known to use trapped-ion quantum computers.

Photonic systems use photons (particles of light) as qubits. Information is encoded in properties of photons, such as polarisation or phase. Photonic quantum computing relies on optical components such as beam splitters and phase shifters to manipulate qubits [16] and is used in companies such as Xanadu [17].

Although demonstrations of quantum advantage, defined as the point at which a quantum computer outperforms the most powerful classical computer for a specific task, are specialised and limited, they show the potential of quantum computing. Google's Sycamore processor showed a computation that involves sampling the output of a pseudo-random quantum circuit, which is difficult for classical computers to simulate. According to their claims, this computation was performed millions of times faster than the best classical supercomputer [10]. In response to Google's 2019 claim of achieving quantum advantage, IBM disputed the assertion, arguing that the task performed by Sycamore could be completed on a classical supercomputer in a feasible time frame [18]. IBM's critique also emphasised the importance of more practical applications of quantum advantage over theoretical milestones [18]. In 2020, researchers at the University of Science and Technology of China announced the development of a photonic quantum computer named Jiuzhang. This system achieved quantum advantage by performing Gaussian boson sampling, a problem considered intractable for classical computers, significantly faster than existing classical supercomputers [19]. In 2022, Xanadu's quantum processor, Borealis, showed the ability to perform Gaussian boson sampling more efficiently than classical counterparts, marking a step toward the quantum advantage [17]. As quantum hardware improves and new algorithms are developed, quantum advantage is expected to expand to broader applications.

1.4 Problem Statement

IDSs are vital components in cybersecurity, yet they face significant challenges as cyberattacks become increasingly sophisticated and diverse. Classical ML models, while effective in many scenarios, struggle to detect subtle and complex attack patterns, especially in large-scale or high-dimensional data. As the volume of network traffic and the evolution of cyberattacks increases, classical models may not generalise well, leading to undetected threats or high false positive rates [20]. Addressing these limitations is essential to enhance the reliability and effectiveness of IDS and prevention of day 0 attacks.

Quantum computing, especially within the NISQ era, offers a potential path to overcome these challenges. Quantum Support Vector Machines (QSVM) and Quantum Convolutional Neural Networks (QCNN) are promising candidates due to their ability to explore complex feature spaces and optimise model performance in ways that classical algorithms cannot. While QSVM and QCNN are theoretically capable of handling intricate and high-dimensional feature spaces more efficiently than classical models [21–23], their actual performance is heavily influenced by how well these algorithms are optimised within the limitations of current quantum hardware. A vital component of these models lies in the design of their unitaries, transformations that define how quantum states evolve during computation. The preceding discussions in Section 1.3 entailed achieving a quantum advantage, which is primarily associated with the questionable/arguable "superiority" of quantum systems, focussing on a physical comparison against the throughput of the best classical systems. However, "superiority" in this research is based on the algorithmic perspective, commonly referred to as quantum speedup. The following literature review looks at current quantum algorithms used for intrusion detection and ways in which improvements can be made.

1.4.1 Literature review

This section presents a review of the most recent literature on QML applications in intrusion detection, with a focus on studies that use the F1 score as the choice of measure due to the metric's ability to balance true negatives and false positives, as further explained in Section 4.1.3.

The development of quantum algorithms has opened new computational capabilities for tackling complex and high-dimensional problems in numerous domains. These algorithms exploit the principles of superposition and entanglement to process information in ways that may surpass their classical counterparts, especially in scenarios where conventional methods encounter intractable time complexity.

In quantum chemistry, quantum algorithms such as the variational quantum eigensolver and quantum phase estimation have been used to simulate molecular systems to capture efficient approximations of ground state energies in many-body systems [24–28]. In finance, quantum algorithms have been used for risk assessment, fraud detection, credit scoring, and stock price prediction. These applications often involve time-series data in which quantum circuits can model intricate temporal dependencies and exploit quantum parallelism [29–31]. In healthcare, quantum algorithms have been applied to medical

imaging for the prediction and classification of Alzheimer's disease and gliomas, as well as for the detection of brain injuries [32–34]. Similarly in oncology, quantum-enhanced medical imaging models have improved the identification of multiple types of cancer [33–37]. In addition, quantum algorithms have been used to analyse medical health records, where large amounts of genetic and clinical data are used to predict how a patient will respond to specific therapies, allowing personalised treatment plans for early diagnosis of diseases and radiology treatment plans [33, 34, 38].

These contributions collectively emphasise the potential of quantum algorithms to address domain-specific challenges through improved modelling ability and computational efficiency, motivating continued exploration of QML capabilities.

The establishment of QML can be traced back to the book *Quantum Machine Learning: What Quantum Computing Means to Data Mining* [39], as well as the paper *"The quest for a Quantum Neural Network"* [40]. In this paper, Schuld et al. reviewed various proposals of quantum-enhanced learning models and noted the challenges of defining quantum analogues of neural networks . The authors concluded that, while a full Quantum Neural Network (QNN) analogue remained elusive, Variational Quantum Circuit (VQC) could be a viable path forward. Another major milestone was *"Quantum algorithms for supervised and unsupervised machine learning,"* which introduced the idea of QSVM [41]. By mapping data into high-dimensional Hilbert spaces via quantum feature maps, it exploited quantum parallelism for SVM classification. It also introduced the idea of hybrid quantum-classical workflows, which has since become a dominant paradigm in QML research.

Havlíček et al. introduced a prominent quantum kernel method in *"Supervised learning with quantum-enhanced feature spaces,"* where they demonstrated that circuits implementing classically hard-to-simulate feature maps could lead to quantum speedup [22]. The core idea was to encode the input data using a quantum feature map, evaluate the kernel using the swap test, and feed the kernel matrix into a classical SVM. The paper provided the first experimental demonstration of quantum kernel estimation on a superconducting quantum processor. The authors introduced feature maps based on data-encoding circuits whose kernel functions were hard to simulate classically. Their implementation yielded notable classification accuracy on toy datasets.

The QNN model based on VQCs optimised via gradient descent was proposed in the paper *Classification with quantum neural networks on near-term processors* [42]. It introduced a layer-wise structure that resembles classical neural networks and used stochastic gradient descent for training. This model shows the feasibility of variational training on NISQ

hardware. Following this, Killoran et al. developed a framework for continuous-variable QNNs using photonic circuits [43]. This work extended QNNs to continuous-variable quantum systems using photonic encodings. The authors defined quantum analogues of convolutional and dense layers using Gaussian gates and non-linear measurements. They applied it to image classification on small datasets and showed that quantum optics is a viable platform for QML.

QCNNs, inspired by classical convolutional neural networks, are extensions of QNNs and are designed to extract hierarchical features using quantum circuits. Cong et al. developed the QCNN architecture in "Quantum Convolutional Neural Networks," which showed success in phase recognition in many-body physics [23]. Their model demonstrated the ability to detect quantum phase transitions using far fewer qubits than the system size.

Summarising the research conducted in the IDS domain, the paper *"Quantum Machine Learning for Network Intrusion Detection Systems, a Systematic Literature Review"* [44], provides a structured analysis of the application of QML techniques to network IDSs between 2017 and 2022. Their findings reveal that QML algorithms, especially QSVMs and QNNs, have been increasingly explored for their potential to process network traffic data and have shown improved classification capabilities to distinguish normal from malicious traffic. In addition, the study indicates that the QML methods can achieve reductions in training times compared to the classical ML approaches.

In 2023, the study by Said et al. [45] introduces a QSVM framework for the detection of DDoS attacks in Smart Micro-Grids. Evaluations using the CIC-DDoS2019 dataset demonstrate that the QSVM model outperforms classical Support Vector Machine (SVM) in terms of accuracy, precision, and computational efficiency, achieving a 93% reduction in execution time. This work indicates the potential of quantum kernel methods to improve real-time detection capabilities in cyber-physical infrastructures. Extending research in cyber-physical infrastructures, in 2024 the paper titled *"Quantum-Neural Network Model for Platform Independent DDoS Detection"* [46] utilises a QNN architecture for platform-agnostic DDoS detection. The platform-independence of the model shows its applicability across heterogeneous network environments, including IoT-based and edge computing systems.

In contrast to kernel-based and variational approaches, the study *Quantum Intrusion Detection System Using Outlier Analysis* [47] identify anomalous behaviour in network traffic by embedding classical data into quantum states and using distance-based criteria in Hilbert space, effectively isolating deviations indicative of malicious activity. This

approach is especially beneficial for zero-day or stealth attacks, where known signatures are unavailable. The use of quantum-enhanced outlier detection shows that QML techniques can enhance IDS performance not only in accuracy but also in generalisation to previously unseen threats.

The paper *"Network Anomaly Detection Using Quantum Neural Networks on Noisy Quantum Computers"* by Kukliansky et al. explores the application of QNNs for intrusion detection that minimises the use of quantum resources [48]. The authors used a Netflow conversion of UNSW-NB15 dataset, namely NF-UNSW-NB15. The study compares four QNN architectures: Tree Tensor Network, Multiscale Entanglement Renormalisation Ansatz, QCNN, and a "Simple" architecture. The Simple architecture, utilising two-qubit Pauli rotation gates with minimal entanglement operations, outperformed the other configurations in both noiseless and noisy simulations. In a noiseless environment, the Simple architecture achieved an F1 score of 0.907, displaying competitive classification capabilities. The architecture's performance remained robust under IonQ's Aria-1 noisy simulation, achieving an F1 score of 0.886. The experiments extend to physical quantum hardware, including IonQ's Harmony and Aria-1 quantum computers. Despite hardware constraints, the Simple architecture achieved an F1 score of 0.86 on Aria-1, representing a state-of-the-art result for quantum-based network intrusion detection on physical devices. With regard to the hyperparameters used, an extensive grid search was conducted on multiple parameters and will be used in this research in Section 4.2.2.1. The batch sizes (16 and 32) did not reveal significant differences, with a batch size of 32 chosen for its efficiency. Learning rates ranging from 0.1 to 0.001 were tested, with 0.02 yielding the best balance between convergence and accuracy. The study compared Adam and Stochastic Gradient Descent (SGD) optimisers, with SGD (momentum 0.2, decay rate 0.001) achieving superior classification performance, while Adam showed faster convergence.

In the paper *"QML-IDS: Quantum Machine Learning Intrusion Detection System"* by Abreu et al. [49], the authors address binary and multiclass classification tasks using QML models, while also benchmarking their performance against classical ML approaches. Three QML techniques: VQC implemented classifier, QSVM, and QCNN are evaluated using three publicly available datasets: UNSW-NB15, CICIDS17, and CICIoT2023. Classical data is mapped onto quantum states via four feature maps: PauliFeatureMap, ZFeatureMap, ZZFeatureMap, and RawFeatureVector. For this, six different backends were used, specifically the noise-free quantum computing simulator QASM, and five IBM NISQ device configurations. The research explores various optimisation strategies to adapt quantum circuits for NISQ device execution while maintaining classification accuracy. While the QASM simulator provided idealised or theoretical results, IBM NISQ device

configurations such as OSAKA and BRISBANE yielded competitive outcomes. Additionally, the paper compares QML techniques with three classical ML models: SVM, Convolutional Neural Networks (CNN), and Random Forests (RF). The comparative analysis shows the potential of QML models by outperforming their classical counterparts.

Table 1.1: Overview of the Performance of QML and Classical Classifiers. The F1 scores in bold will be used as comparison against this research.

Classifier Type	**F1 Score (%)**	**Evaluation Method**	**Dataset**
Simple QNN [48]	90.7	Noiseless Simulation	NF-UNSW-NB15
Simple QNN [48]	88.6	IonQ Aria-1 (Noisy Simulator)	NF-UNSW-NB15
Simple QNN [48]	86.0	IonQ Aria-1 (Physical QPU)	NF-UNSW-NB15
QCNN [49]	**87.45**	Noiseless Simulation	UNSW-NB15
QSVM [49]	**89.34**	Noiseless Simulation	UNSW-NB15
VQC Classifier [49]	88.56	Noiseless Simulation	UNSW-NB15
QCNN [49]	87.32	IBM BRISBANE	UNSW-NB15
QSVM [49]	87.90	IBM OSAKA /IBM CAIRO	UNSW-NB15
VQC Classifier [49]	88.10	IBM KYOTO	UNSW-NB15
Classical SVM [49]	82.34	Classical Algorithm	UNSW-NB15
Random Forest [49]	82.67	Classical Algorithm	UNSW-NB15
CNN [49]	86.72	Classical Algorithm	UNSW-NB15

The notion of QML models outperforming their classical counterpart can be closely related to achieving quantum speedup, the fundamental goal of quantum algorithms. Although the concept of quantum speed up is discussed further in Section 3.2, the aforementioned paper achieves limited quantum speed up by making use of quantum feature maps that are classically intractable. To this end, this research aims to extend the expressiveness and entanglement capabilities of ansatzes that enhance the representational power of QML models. More specifically, enhancing expressibility and entanglement capabilities allows QML models to extend the classical intractability of their quantum feature maps, therefore improving the limited quantum speed up that the QML model can achieve.

The next section focusses on the findings of the paper *"Expressibility and entangling capability of parameterised quantum circuits for hybrid quantum-classical algorithms"* [50]. This will be used to discuss the concept of expressibility and entanglement capabilities, how it is determined, and which quantum circuit designs are to be considered for this research.

1.4.1.1 Expressibility and Entangling Capability

The experiments carried out in the paper involved simulating 19 Parameterised Quantum Circuit (PQC)s with different designs, qubit connectivities (for example, nearest-neighbour, ring, all-to-all) and gate configurations (for example Controlled Rotation-X (CRX), Controlled Rotation-Z (CRZ)). For each circuit, the researchers measured expressibility (closeness to Haar-random states) and entangling capability (average Meyer-Wallach entanglement) by sampling quantum states over a range of parameters.

Expressibility describes how well PQCs can explore the Hilbert space of quantum states. High expressibility requires the circuit to approximate the distribution of Haar-random states or mimic the properties of a t-design. It is commonly measured using the Kullback-Leibler (KL) divergence between the distribution of fidelities generated by the PQC and those from Haar-random states, as seen in Figure 1.1.

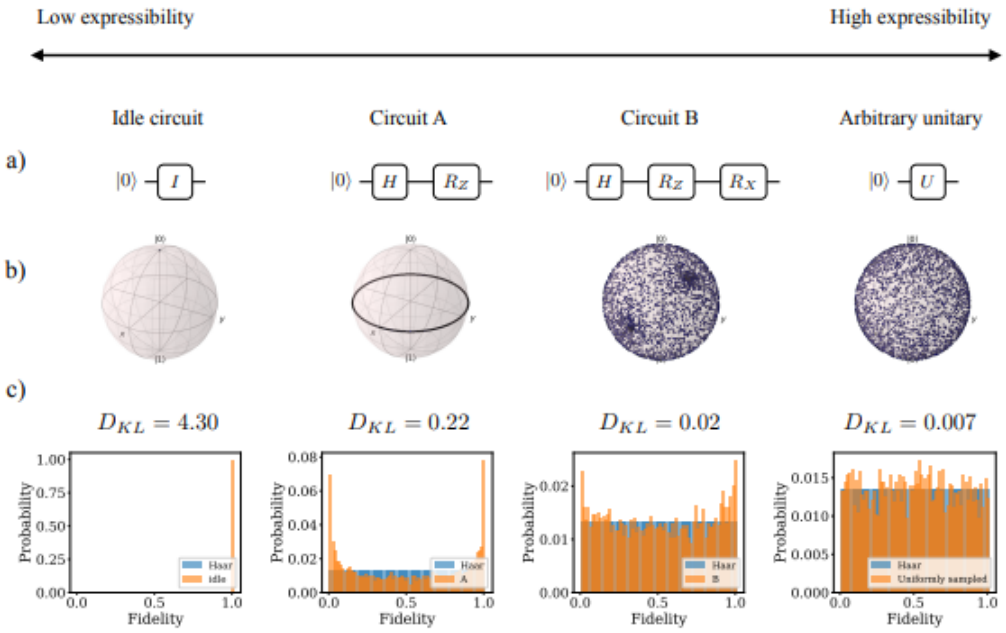


Figure 1.1: Visualisation of expressibility across quantum circuits: (a) Circuit depth and gate diversity increase expressibility, progressing from an idle circuit to a randomly sampled unitary. (b) State distributions represented on the Bloch sphere illustrate how state coverage expands with increasing expressibility. (c) The KL divergence (D_{KL}) between the fidelity distribution of the circuit and Haar-random states quantifies expressibility, with lower D_{KL} values indicating higher expressibility [50].

KL divergence measures how much one probability distribution differs from another or quantifies the "distance" between two distributions by evaluating how well the first distribution (often a reference or ideal distribution) explains or represents the second. A

smaller KL divergence indicates that the two distributions are similar, while a larger value reflects a greater divergence. Therefore, a lower KL divergence indicates that the PQC's expressibility closely approximates the ideal Haar-random behaviour, reflecting its ability to comprehensively explore the Hilbert space, or quantum state space.

Haar randomness refers to the uniform distribution of quantum states over the unitary group. Sampling from a Haar-random distribution, as represented by the arbitrary unitary circuit, ensures that every possible state in the Hilbert space, or the quantum system's state space, is equally likely. In other words, it refers to the ability of a quantum circuit to generate states that uniformly cover the entire Hilbert space associated with that quantum system. However, generating true Haar-random quantum states is computationally infeasible on quantum computers because the resources required scale exponentially with the size of the system. Instead, circuits apply a series of gates that are capable of sufficiently exploring the Hilbert space (approximating Haar randomness) while keeping computational demands manageable. The manner in which these circuits approximate Haar randomness is described by the t-design.

A t-design is an ensemble of unitary operations that approximates Haar randomness up to the t-th statistical moment. Specifically, a 2-*design* captures the second-moment statistics of Haar randomness. The first and second statistical moments refer to properties of a distribution that describe its mean and variance, respectively. The first moment represents the average overlap of states (mean fidelity) in the Hilbert space, while the second moment relates to the variance of these overlaps, capturing the diversity or spread of the states generated by the PQC. A PQC approximating a 1-design reproduces the first-moment statistics of Haar-random states, ensuring the mean overlap is correct, but it may not explore the space uniformly. Achieving a 2-design ensures the variance matches that of Haar randomness, meaning that the circuit generates states that uniformly cover the Hilbert space up to the second moment. The results of the KL divergence can be found in Figure 1.2.

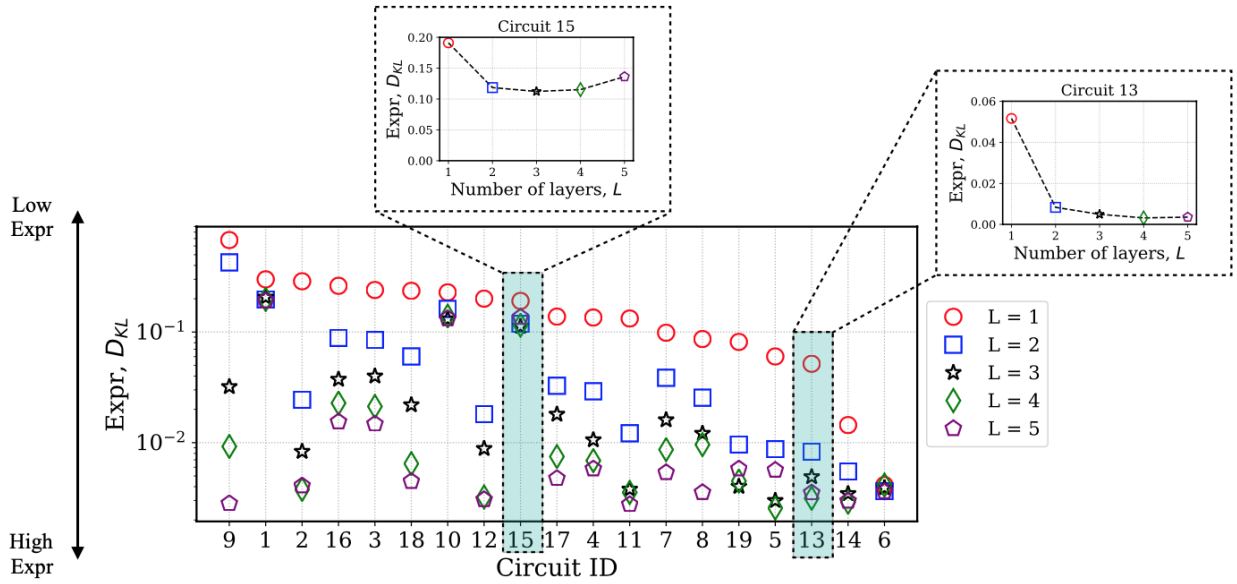


Figure 1.2: Comparison of expressibility (D_{KL}) for various parameterised quantum circuits (Circuit IDs) with different numbers of layers (L). The expressibility decreases (higher expressibility) as D_{KL} decreases. Insets highlight the dependency of D_{KL} on the number of layers (L) for selected circuits (Circuit IDs 13 and 15), demonstrating the impact of increasing depth on expressibility [50].

The entangling capability of PQCs was measured using the Meyer-Wallach (MW) entanglement measure, which calculates how much the qubits in a system are entangled on average. Entanglement is a key resource in quantum computing, allowing quantum algorithms to process and encode information in ways that classical systems cannot. To evaluate this, parameters were randomly chosen, the circuit generated quantum states, and the entanglement measure was calculated for each state. Circuits that produced mostly separate, non-entangled states had low scores, while those that generated highly entangled states scored higher. The entanglement patterns used, viewable in Figure 1.3, were nearest-neighbour, ring, all-to-all topologies. The results of the entangling capabilities of the PQCs using the three different topologies are found in Figure 1.4.

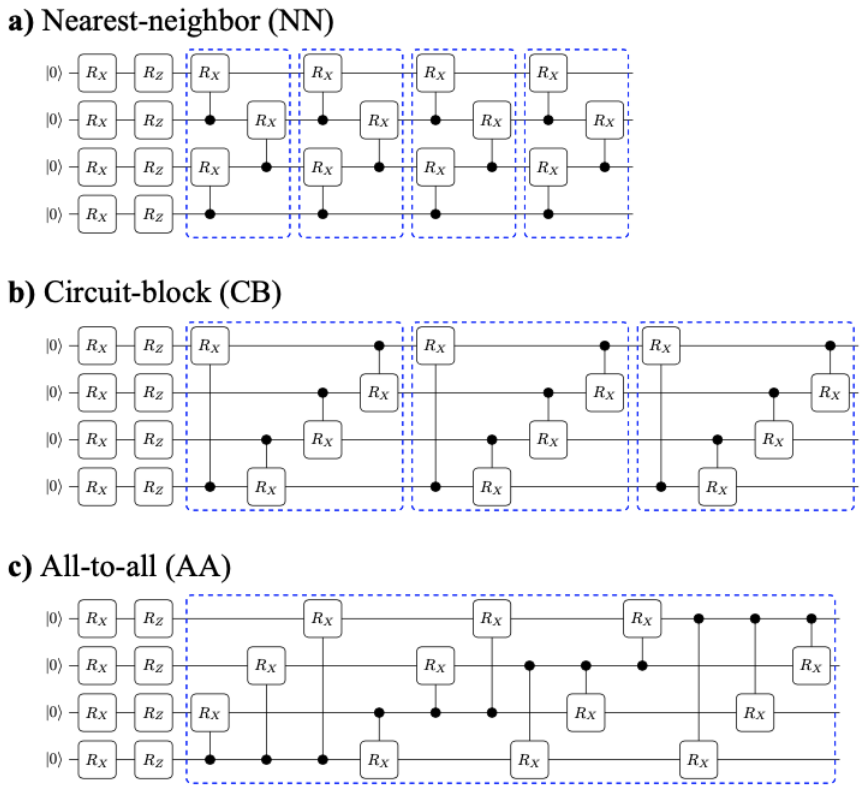


Figure 1.3: Visualisation of different parameterised quantum circuit architectures. (a) Nearest-neighbour connectivity, where gates act only on adjacent qubits within a block. (b) Circuit-block structure, consisting of repeated blocks with wider connectivity compared to nearest neighbour. (c) All-to-all connectivity, where gates can act between any pair of qubits within a block, maximizing entanglement. The dashed blue boxes indicate individual circuit blocks [50].

The study identified differences in the performance of quantum circuit topologies. All-to-all connectivity achieved the best expressibility and entangling capability due to its ability to form interactions between all pairs of qubits, making it ideal for exploring complex quantum states but highly resource-intensive. Circuit-block configurations, which combine nearest-neighbour interactions with limited non-local connections, provided a balance between performance and feasibility, offering high expressibility and entangling capability with reduced connectivity requirements. In contrast, nearest-neighbour connectivity, while being the most hardware-efficient and easier to implement, resulted in lower expressibility and entangling capability, as its limited interaction scope restricts the diversity of quantum states that can be represented.

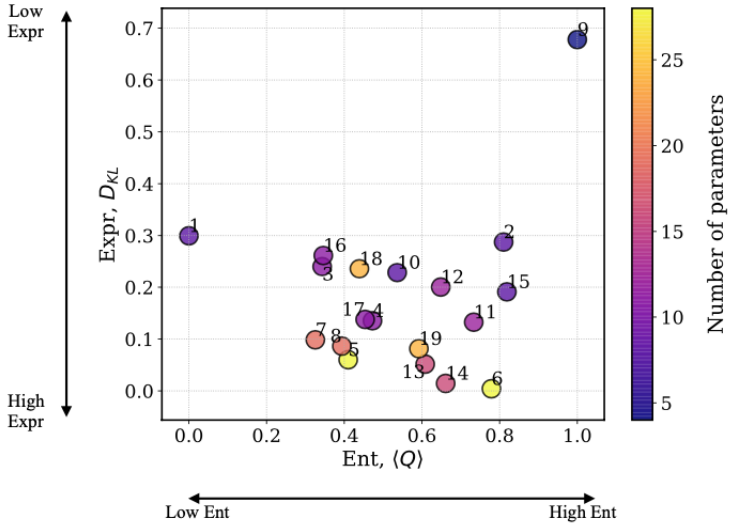


Figure 1.4: Scatter plot showing the relationship between expressibility (D_{KL}) and entanglement ($\langle Q \rangle$) for various circuit configurations. The x-axis represents entanglement (Ent, $\langle Q \rangle$) ranging from "Low Ent" (left) to "High Ent" (right), while the y-axis represents expressibility (Expr, D_{KL}) ranging from "Low Expr" (top) to "High Expr" (bottom). Each point corresponds to a labelled circuit configuration, and is coloured based on the number of parameters [50].

With regard to other testing, the study found that two-qubit gates (e.g., CNOT, CRX, CRZ) remain more error-prone and resource-intensive than single-qubit gates, and their number directly impacts circuit fidelity on current hardware. Circuits using CRX gates generally outperform those using CRZ gates in terms of both expressibility and entangling capability, as CRX gates provide greater flexibility in exploring the state space. While adding more layers to a circuit enhances expressibility, the improvement diminishes beyond a certain depth due to expressibility saturation. Additionally, while increasing the number of tunable parameters can enhance a circuit's representational power, it also complicates the optimisation process in hybrid quantum-classical algorithms.

Figure 1.5 displays the five ansatzes ($A_5, A_6, A_{13}, A_{14}, A_{19}$) chosen to be used in this research. A_5 and A_6 exhibit strong expressibility and entangling capability, making them effective in representing a wide variety of quantum states. This is especially relevant for tasks requiring high state diversity, such as ground-state energy estimation or complex data classification. Although these ansatz rely on all-to-all connectivity, which increases implementation costs in terms of circuit depth and gate count, their use of CRX gates in A_6 further enhances their representational power, outperforming CRZ gate-based designs. However, these ansatz require more parameters, with their cost scaling quadratically with the number of qubits, making them resource-intensive.

A_{13} and A_{14} offer a middle ground between performance and cost. These circuit designs use circuit-block configurations, which combine aspects of nearest-neighbour and non-local qubit interactions. This topology allows them to maintain strong expressibility and entangling capability while requiring fewer resources than fully connected configurations. This makes them suitable for hardware architectures with limited connectivity. They achieve good expressibility and entangling capability with fewer parameters compared to A_5 and A_6.

A_{19} provides a simpler design compared to the other options, utilising ring-topology circuit blocks. This structure balances connectivity requirements with sufficient expressibility and entangling capability for many practical tasks. Its design ensures lower depth and reduced two-qubit gate count compared to fully connected circuits, making it more implementable on NISQ devices while still achieving favourable performance. Additionally, its has the smallest parameter count which beneficial for optimisation.

While the above discussion points towards improvements in the expressibility and entangling capabilities of quantum circuits, does applying it to QML models translate to a significantly higher F1 score? Using these quantum feature maps can increase the exploration of vector representations in the Hilbert space of quantum states; however, the kernel methods that are used (QSVM and QCNN) fundamentally depend on its ability to differentiate between the attack and normal traffic input data.

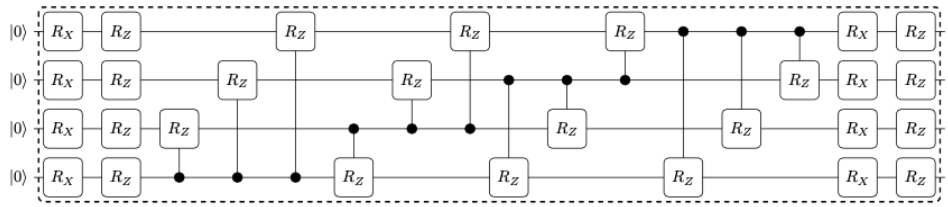
1.4.2 Research Question

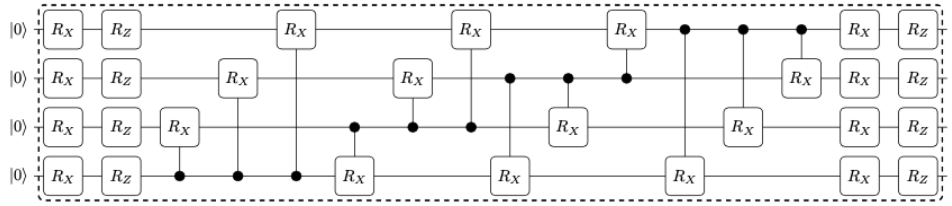
How does varying the design of the QSVM and QCNN models affect their ability to detect cyber threats compared to the less expressive methods in Table 1.1 that use the USNW-NB15 data set in a noiseless quantum device setting?

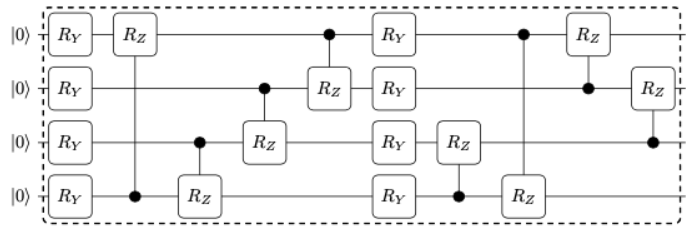
1.5 Thesis Structure

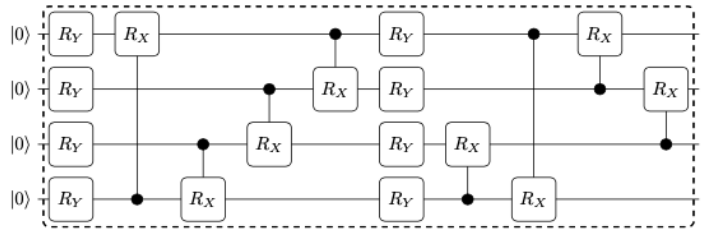
Beyond this introductory chapter, the thesis begins by providing a background on quantum systems and quantum mechanics. It then introduces the concept of a qubit system and its foundational principles and extends this understanding to systems involving multiple qubits. Following this, the quantum circuit model is discussed and how it extends to the QML, and the QML models that this thesis seeks to implement. Chapter 3 then

discusses problems solvable on a Quantum Turing Machine (QTM) and varying types of quantum speed-up. After clarifying the quantum speed up associated with this research, the risk of encountering barren plates is elaborated. In Chapter 4, the process of preparing quantum states from classical cybersecurity data is described. Thereafter, the two QML models used are described in detail, including their experimental configurations. In Chapter 5, the evaluation framework and methods for comparison with existing models are outlined. This is followed by an in-depth analysis and discussion of the experiment results. The thesis then concludes in Chapter 6 with the research insights and discussions about future work.









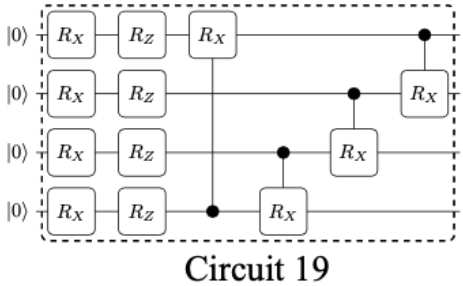


Figure 1.5: Top 5 performing ansatz from the reference paper which will be adapted in this research. Circuits 5 and 6 utilise an all-to-all connectivity with controlled-Z and controlled-X rotation gates, while Circuits 13, 14 and 19 follows a circuit-block entanglement pattern [50].

Chapter 2

Background

2.1 Quantum Mechanics

2.1.1 Introduction to Dirac Notation

Dirac notation, also known as bra-ket notation, is a mathematical tool used extensively in quantum mechanics. This notation simplifies the representation of quantum states and operators, allowing a convenient description of quantum systems [51].

A quantum state is denoted as *ket* $|\psi\rangle$, where ψ represents the state of the quantum system. The corresponding *bra* $\langle\psi|$ is the Hermitian conjugate (complex conjugate transpose) of the ket, used to compute inner products and expectations. The inner product of two quantum states $|\psi\rangle$ and $|\phi\rangle$ is written as:

$$\langle\psi|\phi\rangle, \tag{2.1}$$

which yields a scalar. The outer product $|\psi\rangle\langle\phi|$ represents an operator acting on the quantum state space.

Dirac notation also provides a convenient representation of linear operators. An operator $\hat{A}$ acting on a state $|\psi\rangle$ is denoted as $\hat{A}|\psi\rangle$. Operators are often expressed in terms of their eigenstates $|a\rangle$ and eigenvalues a:

$$\hat{A}|a\rangle = a|a\rangle. \tag{2.2}$$

2.1.2 Hilbert Space

In quantum mechanics, the underlying mathematical framework for describing quantum states and their evolution is based the relationships of vectors on *Hilbert spaces*. A Hilbert space $\mathcal{H}$ is a complete complex vector space equipped with an inner product $\langle \cdot, \cdot \rangle$, which satisfies linearity, conjugate symmetry, and positive definiteness. This inner product induces a norm,

$$\|\psi\| = \sqrt{\langle \psi, \psi \rangle}, \tag{2.3}$$

which allows measurement of distances and angles between quantum states.

The state of a quantum system is represented by a normalised vector $|\psi\rangle$ in a Hilbert space, where normalisation ensures that probability amplitudes sum to one, as required by Born's rule [52]. Physical observables, such as energy, momentum, and spin, correspond to *self-adjoint operators* $\hat{A}$ acting on these vectors. The spectral theorem ensures that such operators possess well-defined eigenvalues that represent possible measurement outcomes in an experiment. Additionally, it ensures that every self-adjoint operator can be expressed through spectral decomposition, capturing quantum measurements and wavefunction collapse [53].

Composite quantum systems are described by the vectors of the tensor products of the Hilbert space, allowing the representation of entangled states, unlike the Cartesian product of Euclidean spaces. The representation of *quantum entanglement*, where a composite state cannot be expressed as a separable product of the states of the subsystem, arises from this tensor structure.

2.1.3 Postulates of Quantum Mechanics

Quantum mechanics is founded on a set of postulates that define the behaviour of quantum systems. The postulates of quantum mechanics provide the foundation for quantum computation, defining how quantum states are represented, how they evolve through unitary operations, how measurements extract information, and how composite systems are described [3].

Postulate 1: *State Space*. The state of a quantum mechanical system is represented by a vector $|\psi\rangle$ in a complete complex Hilbert space $\mathcal{H}$. This vector contains all the

information about the system and is normalised such that:

$$\langle\psi|\psi\rangle = 1. \tag{2.4}$$

Postulate 2: *Observables.*

Observable quantities, such as position, momentum, and energy, are represented by Hermitian operators (self-adjoint operators) acting on the Hilbert space. These operators have real eigenvalues that correspond to possible measurement outcomes. Mathematically, an observable $\hat{A}$ satisfies:

$$\hat{A} = \hat{A}^\dagger, \tag{2.5}$$

where $\hat{A}^\dagger$ is the adjoint (or Hermitian conjugate) of $\hat{A}$. The eigenvalues a of $\hat{A}$ are real, and the eigenstates $|a\rangle$ form a complete orthonormal basis:

$$\hat{A}|a\rangle = a|a\rangle. \tag{2.6}$$

Postulate 3: *Measurement.* The measurement of an observable $\hat{A}$ in a quantum state $|\psi\rangle$ yields one of its eigenvalues a with probability:

$$P(a) = |\langle a|\psi\rangle|^2, \tag{2.7}$$

where $|\langle a|\psi\rangle|^2$ is the squared magnitude of the projection of $|\psi\rangle$ onto the eigenstate $|a\rangle$. After measurement, the system collapses into the eigenstate $|a\rangle$:

$$|\psi\rangle \to |a\rangle. \tag{2.8}$$

Postulate 4: *Time Evolution.* The time evolution of a quantum system is governed by the Schrödinger equation. If $\hat{H}$ is the Hamiltonian operator representing the total energy of the system, the state $|\psi(t)\rangle$ evolves as:

$$i\hbar\frac{\partial}{\partial t}|\psi(t)\rangle = \hat{H}|\psi(t)\rangle. \tag{2.9}$$

The solution to this equation for a time-independent Hamiltonian is:

$$|\psi(t)\rangle = e^{-i\hat{H}t/\hbar}|\psi(0)\rangle, \tag{2.10}$$

where $e^{-i\hat{H}t/\hbar}$ is the time-evolution operator.

Postulate 5: *Composite Systems.* For composite systems, the state space is the tensor product of the state spaces of the individual subsystems. If $\mathcal{H}_1$ and $\mathcal{H}_2$ are the state spaces of two subsystems, the composite state space is:

$$\mathcal{H}_{\text{total}} = \mathcal{H}_1 \otimes \mathcal{H}_2. \tag{2.11}$$

The state of the composite system $|\psi_{\text{total}}\rangle$ can exhibit quantum entanglement, where the state cannot be expressed as a product of individual subsystem states:

$$|\psi_{\text{total}}\rangle \neq |\psi_1\rangle \otimes |\psi_2\rangle. \tag{2.12}$$

2.2 Qubit Systems

This section discusses the concepts and properties of qubit systems, drawing on teachings from textbooks in quantum information science [2, 3, 51].

2.2.1 The Computational Basis (Z Basis)

As mentioned in Section 1.3, the fundamental unit of quantum information is the qubit, analogous to the classical bit. However, unlike a classical bit, which exists in one of two definite states, 0 or 1, a qubit exists in a superposition of two basis states, commonly referred to as the computational basis states. These states are denoted as $|0\rangle$ and $|1\rangle$, and they form an orthonormal basis for the two-dimensional Hilbert space $\mathbb{C}^2$.

A general qubit state $|\psi\rangle$ in the computational basis can be expressed as:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{2.13}$$

where $\alpha, \beta \in \mathbb{C}$ and satisfy the normalisation condition:

$$|\alpha|^2 + |\beta|^2 = 1. \tag{2.14}$$

The computational basis is equivalent to the Z-basis, where the $\hat{Z}$ operator is represented

as:

$$\hat{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \tag{2.15}$$

and the eigenvectors of Z are:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{2.16}$$

The states $|0\rangle$ and $|1\rangle$ are eigenstates of the Pauli-Z operator, $\hat{Z}$, with the following eigenvalues:

$$\hat{Z}|0\rangle = |0\rangle, \quad \hat{Z}|1\rangle = -|1\rangle. \tag{2.17}$$

The Bloch-sphere representation in Figure 2.1 is a geometric representation of the state space of a single qubit, useful for visualising qubit states and their interaction with quantum gates. The north and south poles correspond to the computational basis states $|0\rangle$ and $|1\rangle$, respectively. The superposition states lie on the surface of the sphere, and the phase relationship between the basis states determines the position on the sphere.

Any pure qubit state $|\psi\rangle$ can be uniquely represented as a point on the surface of the sphere. In terms of spherical coordinates, the qubit state is parameterised as:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle, \tag{2.18}$$

where $\theta \in [0, \pi]$ and $\phi \in [0, 2\pi)$ are the polar and azimuthal angles, respectively.

A quantum state is classified as pure if it is represented by a single vector $|\psi\rangle$ in the Hilbert space. The density matrix for a pure state is given by:

$$\rho = |\psi\rangle\langle\psi|, \tag{2.19}$$

which satisfies $\mathrm{Tr}(\rho^2) = 1$ and $\mathrm{Tr}(\rho) = 1$. Pure states are points on the surface of the Bloch sphere.

Mixed states, on the other hand, are statistical ensembles of pure states. They are represented by a density matrix:

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|, \tag{2.20}$$

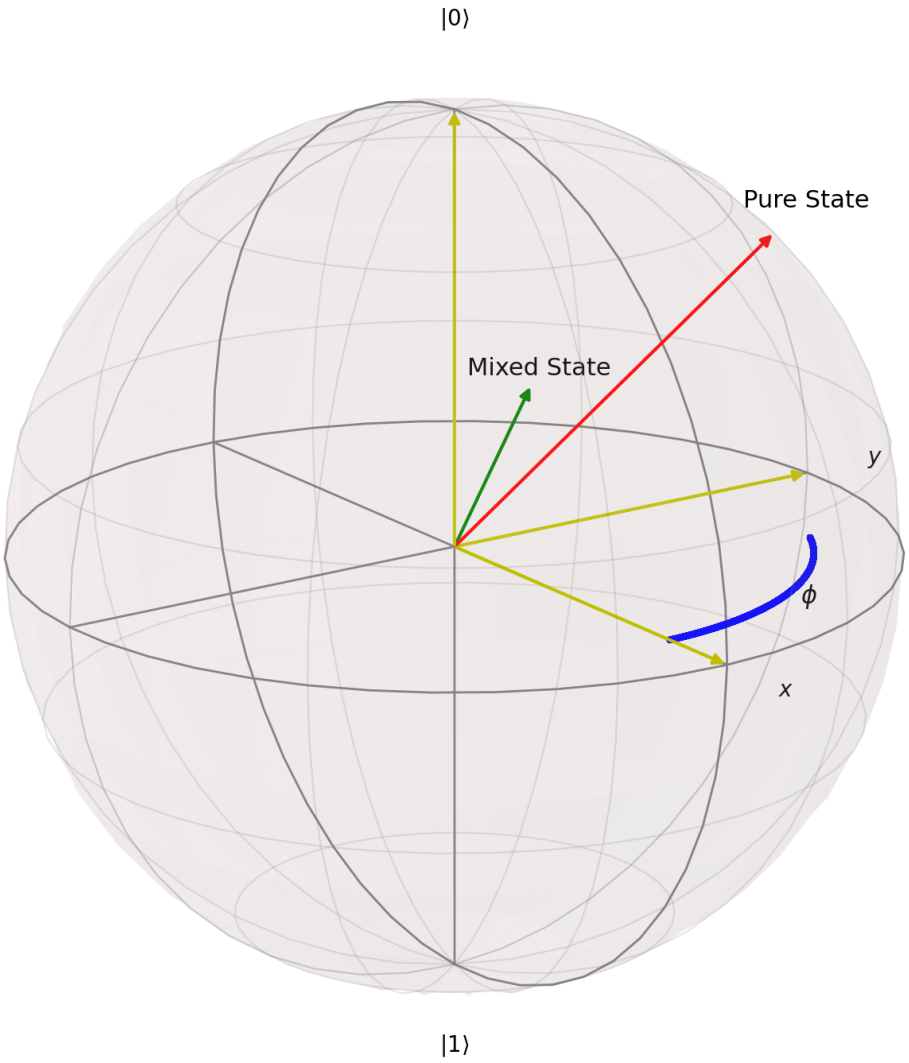


Figure 2.1: Visualisation of quantum states on the Bloch sphere. The red vector represents a pure state, which lies on the surface of the sphere. The green vector represents a mixed state, located inside the sphere. The azimuthal angle ϕ is shown in blue on the xy-plane, characterizing the phase relationship between basis states. The yellow lines represent the Cartesian coordinate axes of the Bloch sphere. The north and south poles correspond to the computational basis states $|0\rangle$ and $|1\rangle$, respectively.

where p_i are the probabilities associated with each pure state $|\psi_i\rangle$, satisfying $\sum_i p_i = 1$. Mixed states correspond to points inside the Bloch sphere, with the centre representing a completely mixed state (maximum entropy).

2.2.2 Composite Qubit Systems

For systems involving multiple qubits, the state space is the tensor product of the individual qubit state spaces. For two qubits, the composite state is described in the four-dimensional Hilbert space $\mathbb{C}^2 \otimes \mathbb{C}^2$ spanned by the basis states $|00\rangle$, $|01\rangle$, $|10\rangle$, and

$|11\rangle$. A general two-qubit state is:

$$|\psi\rangle = \sum_{i,j\in\{0,1\}} c_{ij}|ij\rangle, \tag{2.21}$$

where $c_{ij} \in \mathbb{C}$ and satisfy the normalisation condition:

$$\sum_{i,j} |c_{ij}|^2 = 1. \tag{2.22}$$

Composite systems can exhibit entanglement, a uniquely quantum phenomenon in which the state of the whole system cannot be expressed as a tensor product of individual qubit states. For example, the Bell state:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle), \tag{2.23}$$

is an entangled state.

The density matrix extends naturally to composite systems, allowing for the description of both entangled and separable states. Measurements in composite systems are described by operators acting on the tensor product space, and partial trace operations allow the analysis of subsystems.

2.2.3 Quantum Measurements

Measurements in quantum mechanics are a vital process through which quantum states collapse into one of the basis states. In the standard computational basis, the arbitary measurement operators are:

$$M_0 = |0\rangle\langle 0|, \quad M_1 = |1\rangle\langle 1|, \tag{2.24}$$

where M_0 and M_1 correspond to observing the qubit in the $|0\rangle$ or $|1\rangle$ state, respectively.

For a single qubit in the state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, the probability of measuring $|0\rangle$ is:

$$P(0) = \langle\psi|M_0|\psi\rangle = |\alpha|^2, \tag{2.25}$$

and the probability of measuring $|1\rangle$ is:

$$P(1) = \langle\psi|M_1|\psi\rangle = |\beta|^2. \tag{2.26}$$

After measurement, the qubit collapses to the observed basis state. For instance, if the outcome is 0, the post-measurement state is:

$$|\psi\rangle \to |0\rangle. \tag{2.27}$$

In multi-qubit systems, measurements can be extended to the entire computational basis. For example, the measurement operators for two qubits are:

$$M_{ij} = |ij\rangle\langle ij|, \quad i, j \in \{0, 1\}. \tag{2.28}$$

Measuring a qubit in the Z-basis requires adjustments to Equation 2.14. In addition to being a normalisation condition, it can also be interpreted as a sum of probabilities. The Pauli$Z$ operator can be written as:

$$Z = |0\rangle\langle 0| - |1\rangle\langle 1| = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \tag{2.29}$$

with eigenvalues $+1$ and -1, and eigenstates $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$, corresponding to the computational basis states. The measurement process projects the quantum state onto one of these eigenstates, and the outcome of the measurement is the eigenvalue associated with that eigenstate.

The probabilities of obtaining each eigenvalue, given by the Born rule [52], are now:

$$P(+1) = |\langle 0|\psi\rangle|^2 = |\alpha|^2, \tag{2.30}$$

$$P(-1) = |\langle 1|\psi\rangle|^2 = |\beta|^2. \tag{2.31}$$

After measurement, the quantum state collapses to the eigenstate corresponding to the measured eigenvalue:

$$\text{If } +1 \text{ is observed: } |\psi\rangle \to |0\rangle, \tag{2.32}$$

$$\text{If } -1 \text{ is observed: } |\psi\rangle \to |1\rangle. \tag{2.33}$$

The expectation value of the Pauli-Z operator for the state $|\psi\rangle$ represents the statistical average of many measurements and is calculated as:

$$\langle Z\rangle = \langle\psi|Z|\psi\rangle. \tag{2.34}$$

Substituting $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, we compute:

$$\langle Z\rangle = (\alpha^*\langle 0| + \beta^*\langle 1|)\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix}. \tag{2.35}$$

This simplifies to:

$$\langle Z\rangle = |\alpha|^2 - |\beta|^2. \tag{2.36}$$

The expectation value reflects the weighted difference between the probabilities of measuring the eigenvalues $+1$ and -1:

$$\langle Z\rangle = P(+1) - P(-1). \tag{2.37}$$

2.2.4 Quantum Circuits

Quantum circuits form the foundation of quantum computation. They are made up of quantum gates that act on qubits. They consist of a sequence of quantum gates acting on qubits, analogous to classical logic gates in classical computing, which are represented as a circuit diagram. The basic structure of a quantum circuit includes the initialisation of qubits to a known state, usually $|0\rangle$; unitary operations are applied to manipulate the qubit states; and at the end of the computation, the qubits are measured, collapsing their states to classical outcomes. The means of mapping classical data to a quantum state is referred to as encoding or embedding, used interchangeably in this thesis.

2.2.4.1 Angle Encoding

Angle encoding maps classical data to the rotation angles of quantum gates. For a single classical value x, the corresponding quantum state is encoded as:

$$|\psi\rangle = \cos(x)|0\rangle + \sin(x)|1\rangle. \tag{2.38}$$

For a vector of classical values $\mathbf{x} = (x_1, x_2, \ldots, x_n)$, the state of n qubits can be represented as:

$$|\psi\rangle = \bigotimes_{i=1}^{n} \left(\cos(x_i)|0\rangle + \sin(x_i)|1\rangle\right). \tag{2.39}$$

This method is computationally efficient for low-dimensional data and is used in VQC and ML applications.

2.2.4.2 Amplitude Encoding

Amplitude encoding embeds classical data in the amplitudes of a quantum state. For a normalised vector $\mathbf{x} = (x_1, x_2, \ldots, x_N)$, the corresponding quantum state is:

$$|\psi\rangle = \sum_{i=1}^{N} x_i|i\rangle, \tag{2.40}$$

where $|i\rangle$ represents the computational basis states.

For example, if $\mathbf{x} = (x_1, x_2)$ with $|x_1|^2 + |x_2|^2 = 1$, the quantum state for a single qubit is:

$$|\psi\rangle = x_1|0\rangle + x_2|1\rangle. \tag{2.41}$$

Amplitude encoding is suitable for high-dimensional data, but may require complex quantum circuits to prepare the desired state.

2.2.4.3 Unitary Evolution

Unitary operators govern the evolution of quantum states in a circuit. A quantum state $|\psi\rangle$ evolves under a unitary operator U as:

$$|\psi'\rangle = U|\psi\rangle, \tag{2.42}$$

where U satisfies $U^\dagger U = I$. Unitaries ensure the preservation of probability amplitudes and the reversibility of quantum operations. The evolution of the quantum state through a circuit is described by successive applications of unitary transformations. For a quantum circuit with gates $U_1, U_2, \ldots, U_n$, the final state $|\psi_f\rangle$ is given by:

$$|\psi_f\rangle = U_n U_{n-1} \cdots U_1 |\psi_0\rangle, \tag{2.43}$$

where $|\psi_0\rangle$ is the initial state.

A common method for constructing unitary operations is through the exponential mapping of a Hermitian operator H, as seen in equation 2.9. This is expressed as:

$$U = e^{-iHt}, \tag{2.44}$$

where H is a Hermitian matrix, representing the generator of the unitary transformation, t is a real scalar parameter, often associated with time or a scaling factor, and i is the imaginary unit. The exponential mapping guarantees that U is unitary since the Hermitian property of H ensures that e^{-iHt} satisfies $U^\dagger U = I$.

However, the evolution of unitaries in quantum circuits is not dependent on time and is implemented through the sequential application of quantum gates, which correspond to specific unitary operations. Specifically, quantum gates are unitary operators that act on qubits, transforming their states in the Hilbert space. The Solovay-Kitaev theorem provides a guarantee that any unitary operation on a quantum system can be efficiently approximated using a finite set of universal quantum gates. The importance of this theorem lies in its practical implications: it guarantees that even with a limited set of quantum gates, any quantum algorithm or operation can be implemented to high precision, making it feasible to build and operate universal quantum computers. This result underpins the design of quantum compilers and supports the realisation of NISQ hardware.

Quantum gates come in two forms, non-parameterised and parameterised gates. Non-

parameterised gates are fixed operations with no adjustable parameters. These fixed gates implement specific, predefined transformations in quantum states and are not modified using external inputs. 2.1 displays common non-parameterised gates:

Gate Type	**Matrix**	**Name**	**Purpose and Equation**
X	$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	NOT (Pauli-X)	Single-qubit flip: $X\|0\rangle = \|1\rangle, X\|1\rangle = \|0\rangle$.
Y	$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	Pauli-Y	Single-qubit flip with phase: $Y\|0\rangle = i\|1\rangle, Y\|1\rangle = -i\|0\rangle$.
Z	$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	Pauli-Z	Phase flip: $Z\|0\rangle = \|0\rangle, Z\|1\rangle = -\|1\rangle$.
H	$\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	Hadamard	Creates superposition: $H\|0\rangle = \frac{\|0\rangle+\|1\rangle}{\sqrt{2}}$.
$CNOT$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$	Controlled NOT	Multi-qubit gate: Flips target if control is $\|1\rangle$.
$SWAP$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	SWAP Gate	Multi-qubit gate: Swaps two qubit states.
I	$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	Identity	Leaves the state unchanged: $I\|\psi\rangle = \|\psi\rangle$.

Table 2.1: Summary of common quantum gates, their matrices, names, and purposes.

However, in certain quantum circuit designs, designing unitary operations to encode specific functions or transformations based on input data is essential. PQCs use a parameterised Hermitian generator $H(\theta)$, resulting in the unitary:

$$U(\theta) = e^{-iH(\theta)t}. \tag{2.45}$$

These parameterised quantum gates make use of variable parameters, allowing continuous

control over qubit transformations. Common parameterised gates include rotation gates:

$$R_x(\theta) = e^{-i\frac{\theta}{2}\hat{X}} = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix}, \tag{2.46}$$

$$R_y(\theta) = e^{-i\frac{\theta}{2}\hat{Y}} = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix}, \tag{2.47}$$

$$R_z(\phi) = e^{-i\frac{\phi}{2}\hat{Z}} = \begin{bmatrix} e^{-i\frac{\phi}{2}} & 0 \\ 0 & e^{i\frac{\phi}{2}} \end{bmatrix}. \tag{2.48}$$

2.2.5 Variational Quantum Circuits

VQCs are a class of hybrid quantum-classical algorithms that use parameterised quantum circuits trained using classical optimisation techniques to minimise or maximise a cost function.

An arbitrary VQC consists of the following components. Qubits are initialised to a known state, usually $|0\rangle^{\otimes n}$, where n is the number of qubits. The quantum circuit is then composed of unitary gates with trainable parameters, denoted as $U(\theta)$, where θ is a vector of parameters. At the end of the circuit, qubits are measured to extract classical information. The results of the measurements are used to compute a cost function $C(\theta)$, which quantifies the performance of the current parameters. Finally, classical optimisation techniques, such as gradient descent or stochastic methods, are used to update the parameters θ to minimise the cost function.

The quantum state produced by a VQC is:

$$|\psi(\theta)\rangle = U(\theta)|0\rangle^{\otimes n}, \tag{2.49}$$

where $U(\theta)$ is the parameterised unitary operator representing the quantum circuit.

The expectation value of an observable $\hat{O}$ with respect to the state $|\psi(\theta)\rangle$ is computed as:

$$\langle\hat{O}\rangle = \langle\psi(\theta)|\hat{O}|\psi(\theta)\rangle. \tag{2.50}$$

This expectation value is often used as the cost function:

$$C(\theta) = \langle\psi(\theta)|\hat{O}|\psi(\theta)\rangle. \tag{2.51}$$

The training of these parameters is what differentiates between variational parameterised circuits and data-driven parameterised quantum circuits.

2.3 Quantum Machine Learning

VQCs make up the essence of QML, bringing the strengths of quantum computation and classical optimisation. VQCs are used to construct models analogous to classical ML frameworks such as neural networks. The parameters of the quantum circuit, often represented as angles of rotation gates, are optimised to minimise a cost function that encodes the problem to be solved. The optimisation process relies on classical algorithms, while quantum hardware evaluates the cost function through measurements and expectation values.

An arbitrary QML pipeline utilising VQCs begins with data encoding, where classical data is mapped to quantum states using techniques such as angle encoding or amplitude encoding. This step is vital because the choice of encoding determines the effectiveness of the quantum model. Once the data is encoded, a parameterised quantum circuit acts on the quantum state, applying a sequence of unitary transformations. These transformations are designed to explore the quantum state space and capture the underlying patterns in the data.

The output of the circuit is then measured, producing probabilities associated with the computational basis states. These probabilities are interpreted as predictions or are used to compute an expectation value that serves as a cost function. For example, in classification tasks, measured probabilities can represent the likelihood that input data belong to different classes. In unsupervised learning, the cost function may represent the fidelity of the reconstructed quantum state.

The optimisation of the circuit parameters is performed iteratively, with the classical optimiser updating the parameters based on the gradients or other heuristic methods. The combination of quantum circuit evaluations and classical optimisation forms a hybrid algorithm, exploiting the strengths of both paradigms.

VQCs have been applied to a variety of ML problems, including supervised learning and unsupervised learning. In supervised learning, VQCs use parameterised quantum circuits to learn decision boundaries for classification tasks. In unsupervised learning, quantum autoencoders utilise VQCs to compress and reconstruct quantum data, identifying salient

features in the process.

However, the application of VQCs to QML faces several challenges. The issue of barren plateaus, where gradients vanish exponentially with system size, can hinder the training of large-scale quantum circuits. Noise and decoherence in current quantum devices also limit the fidelity of quantum operations, affecting the reliability of the learning process. Additionally, the scalability of VQCs remains a challenge, as larger circuits require more qubits and deeper gate sequences, increasing the susceptibility to errors. To overcome these challenges, researchers are exploring advanced techniques such as varying ansatz design, adaptive optimisation methods, and hybrid quantum-classical architectures.

2.4 Support Vector Machines

SVMs are a supervised ML technique widely applied to classification problems and occasionally used for regression tasks. The main principle behind SVMs is the identification of a decision boundary, or hyperplane, that effectively separates data into distinct categories [2, 54]. The choice of the hyperplane is important because the optimal one maximises the distance, known as the margin, between data points of different categories. This ensures better generalisation when classifying unseen data.

The following discussion first explores how SVMs work for linearly separable data and then expands to their application in more complex, non-linear scenarios.

2.4.1 Linearly Separable Support Vector Machines

In cases where data can be linearly separated, SVMs aim to find a hyperplane that divides the dataset into two distinct groups based on their labels. A hyperplane is defined mathematically as:

$$\mathbf{w}^T\mathbf{x} - b = 0, \tag{2.52}$$

where:

- $\mathbf{w}$ is the normal vector to the hyperplane.

- $\mathbf{w}^T$ is the transpose of $\mathbf{w}$, allowing the calculation of the dot product $\mathbf{w}^T\mathbf{x}$.
- $\mathbf{x}$ represents a data point in the feature space.
- b is the bias term, which shifts the hyperplane relative to the origin.

SVMs define two parallel hyperplanes on either side of the decision boundary to create a margin. These hyperplanes are expressed as:

$$\mathbf{w}^T\mathbf{x} - b = 1 \quad \text{and} \quad \mathbf{w}^T\mathbf{x} - b = -1. \tag{2.53}$$

The margin is the distance between these two hyperplanes, calculated as:

$$\frac{2}{\|\mathbf{w}\|}. \tag{2.54}$$

This formula arises from the fact that the perpendicular distance from a point to a hyperplane is given by:

$$\text{Distance} = \frac{|\mathbf{w}^T\mathbf{x} - b|}{\|\mathbf{w}\|}. \tag{2.55}$$

For the two hyperplanes defined above, their positions are symmetric about the decision boundary. The difference between these positions is:

$$\frac{1 - (-1)}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}. \tag{2.56}$$

The SVM's objective is to maximize this margin, as a larger margin generally improves the model's ability to generalize to unseen data. The data points that lie closest to the hyperplanes are known as support vectors and are critical in determining the optimal hyperplane.

The process of identifying the optimal hyperplane is mathematically formulated as an optimisation problem, aiming to minimise $\|\mathbf{w}\|$ while satisfying the margin constraints. The Lagrange method is used for this minimisation process. By minimising this Lagrangian, the optimal hyperplane can be efficiently computed using only the support vectors. This

minimisation ensures that the hyperplane achieves maximum separation between the data groups.

2.4.2 Non-Linear Support Vector Machines

When data cannot be separated linearly, SVMs rely on a technique called feature mapping. This involves transforming the data into a higher-dimensional space, where it may become linearly separable. Once the separation is achieved in this higher-dimensional space, the results can be projected back into the original space.

Instead of explicitly calculating the transformation, SVMs use the kernel trick, which computes the similarity between pairs of data points directly in the original space. The kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ measures this similarity and corresponds to the inner product of the points in the transformed space:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle, \tag{2.57}$$

where $\phi(\mathbf{x})$ represents the transformation to the higher-dimensional space. Common kernel functions include linear, polynomial, and radial basis functions, which are selected based on the nature of the problem.

2.4.3 Gram Matrix

The kernel trick in SVMs relies on the Gram matrix, a mathematical representation that captures the similarity between all pairs of data points using the selected kernel function. The Gram matrix is defined as:

$$K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) \quad \text{for } i, j = 1, \ldots, n, \tag{2.58}$$

where n is the total number of data points. The matrix is symmetric and positive semidefinite, ensuring that the kernel corresponds to a valid transformation.

The Gram matrix plays a central role in the classification process by allowing the com-

putation of the decision boundary in the transformed space. For non-linear classification, the decision function is expressed as:

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{m} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b\right), \tag{2.59}$$

where:

- α_i are the Lagrange multipliers obtained during training.
- y_i are the labels of the training points (1 or -1).
- $k(\mathbf{x}_i, \mathbf{x})$ is the kernel function, which depends on the similarity between the training point $\mathbf{x}_i$ and the input $\mathbf{x}$.
- b is the bias term.

This formulation takes advantage of the Gram matrix to compute the required kernel values $k(\mathbf{x}_i, \mathbf{x})$, allowing efficient classification without explicitly performing the transformation to a higher-dimensional space. Hence, the Gram matrix simply connects the kernel trick to the calculation of the decision boundary.

2.5 Quantum Kernels

The kernel method is vital to SVMs, allowing the model to handle non-linear relationships in the data. A kernel function transforms data into a higher-dimensional feature space, where it becomes linearly separable, without explicitly computing the transformation. This technique, known as the *kernel trick*, is both computationally efficient and mathematically elegant.

2.5.1 Kernel Function

A kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ computes the similarity between two data points $\mathbf{x}_i$ and $\mathbf{x}_j$ in the input space, corresponding to their inner product in the transformed feature space.

Popular kernel functions include:Linear Kernel, Polynomial Kernel, Radial Basis Function (RBF) Kernel

The kernel function builds the Gram matrix K, where each entry $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$ represents the similarity between two data points. This matrix plays a central role in the SVM optimization process, as it encapsulates all pairwise relationships within the dataset.

Although classical kernels have shown their effectiveness across various applications, they encounter significant limitations in specific scenarios. First, the computation and storage of the Gram matrix for large datasets can become prohibitively expensive in terms of both memory and processing time, posing a major computational bottleneck. Second, classical kernels often struggle to represent highly complex patterns or intricate relationships within the data, which can result in suboptimal model performance. Lastly, the choice of a classical kernel inherently restricts the feature space to predefined structures, limiting the adaptability of the model to diverse problem domains. These challenges drive the development of alternative approaches, such as quantum-enhanced kernels, which aim to overcome computational inefficiencies and expand the expressiveness of feature mappings.

2.5.2 Quantum Feature Mapping

Quantum computing creates a new approach to feature mapping, where classical data is encoded into quantum states in a high-dimensional Hilbert space. This process, known as *quantum feature mapping*, utilizes the properties of quantum systems, such as superposition and entanglement, to create complex and expressive representations of the data.

Quantum feature mapping is achieved by encoding a classical data point $\mathbf{x}$ into a quantum state $|\psi(\mathbf{x})\rangle$, where $|\psi(\mathbf{x})\rangle$ resides in a Hilbert space. The quantum kernel $k_Q(\mathbf{x}_i, \mathbf{x}_j)$ plays a similar role to classical kernels but takes advantage of the quantum feature mapping. This kernel is computed using a quantum circuit that encodes the data and evaluates the overlap between quantum states. The inner product in this space, corresponding to a quantum kernel, is given by:

$$k_Q(\mathbf{x}_i, \mathbf{x}_j) = |\langle\psi(\mathbf{x}_i)|\psi(\mathbf{x}_j)\rangle|^2. \tag{2.60}$$

where $|\psi(\mathbf{x})\rangle$ is the quantum state corresponding to the data point $\mathbf{x}$ after being encoded into a quantum system.

The quantum feature, and thus the quantum kernels, mapping offers several key advantages. By encoding classical data into quantum states, quantum systems naturally operate in exponentially large spaces, allowing the exploration of highly expressive and complex feature mappings that are challenging to achieve classically. Quantum devices can efficiently compute inner products in these quantum-enhanced feature spaces, significantly reducing the computational overhead associated with kernel computation for specific types of data. Furthermore, quantum circuits are highly adaptable, allowing for the design of custom feature mappings tailored to problem-specific requirements. This flexibility, combined with the ability to explore non-classical feature spaces, provides opportunities for improved classification performance. Quantum kernels thus show scalability and handle the increasing complexity of high-dimensional datasets more effectively than classical methods.

In the next section, two types of QSVMs are discussed, showing how quantum kernels and feature mappings are integrated into the SVM framework to enhance performance and scalability.

2.6 Quantum Support Vector Machine

QSVM follow the same fundamental principles as classical SVMs, such as defining a hyperplane that maximally separates the data classes. However, the key difference lies in the computation of the kernel function. Instead of relying on classical kernel functions, QSVM uses a quantum kernel function $k_Q(\mathbf{x}_i, \mathbf{x}_j)$, which is computed using a quantum circuit. This quantum kernel represents the inner product or overlap of data points mapped to a quantum-enhanced feature space. The quantum kernel function allows the SVM to explore feature spaces that are difficult or impossible to simulate classically, also known as classically intractable.

Quantum kernel methods compute the similarity between data points by comparing their quantum state representations in a high-dimensional Hilbert space. Two primary approaches for designing quantum circuits to calculate these kernel values are the quantum fidelity kernel and the swap test. Both methods rely on the fundamental principles of quantum mechanics, such as unitary operations and their adjoints, to evaluate the overlap between quantum states, but they differ in implementation and computational requirements.

2.6.1 Quantum Fidelity Kernels

The quantum fidelity kernel measures the similarity between two quantum states by evaluating their fidelity, which represents the squared magnitude of the inner product between the states. For two data points $\mathbf{x}_i$ and $\mathbf{x}_j$, the quantum states are prepared as:

$$|\psi(\mathbf{x}_i)\rangle = U(\mathbf{x}_i)|0\rangle^{\otimes n}, \quad |\psi(\mathbf{x}_j)\rangle = U(\mathbf{x}_j)|0\rangle^{\otimes n}, \tag{2.61}$$

where $|0\rangle^{\otimes n}$ is the initial state of the n-qubit system, and $U(\mathbf{x})$ is a parameterised unitary operation encoding the data. The fidelity between these states is given by:

$$F(\mathbf{x}_i, \mathbf{x}_j) = |\langle\psi(\mathbf{x}_i)|\psi(\mathbf{x}_j)\rangle|^2. \tag{2.62}$$

The fidelity kernel can be interpreted geometrically: if $|\psi(\mathbf{x}_i)\rangle$ and $|\psi(\mathbf{x}_j)\rangle$ are similar, their overlap is high, and the fidelity approaches 1. In contrast, dissimilar states have minimal overlap, and the fidelity approaches 0.

Operationally, the circuit begins with the quantum system in the initial state $|0\rangle^{\otimes n}$. The unitary operation $U(\mathbf{x}_i)$ prepares $|\psi(\mathbf{x}_i)\rangle$, while its adjoint $U^\dagger(\mathbf{x}_j)$ is applied to reverse the evolution of the state, allowing a direct comparison with $|\psi(\mathbf{x}_j)\rangle$. If the states are highly similar, the system effectively returns to $|0\rangle^{\otimes n}$, indicating a high similarity between the input data points. This circuit can be viewed schematically in 2.2 and mathematically below:

$$F(\mathbf{x}_i, \mathbf{x}_j) = \left|\langle 0^{\otimes n}|U^\dagger(\mathbf{x}_j)U(\mathbf{x}_i)|0^{\otimes n}\rangle\right|^2. \tag{2.63}$$

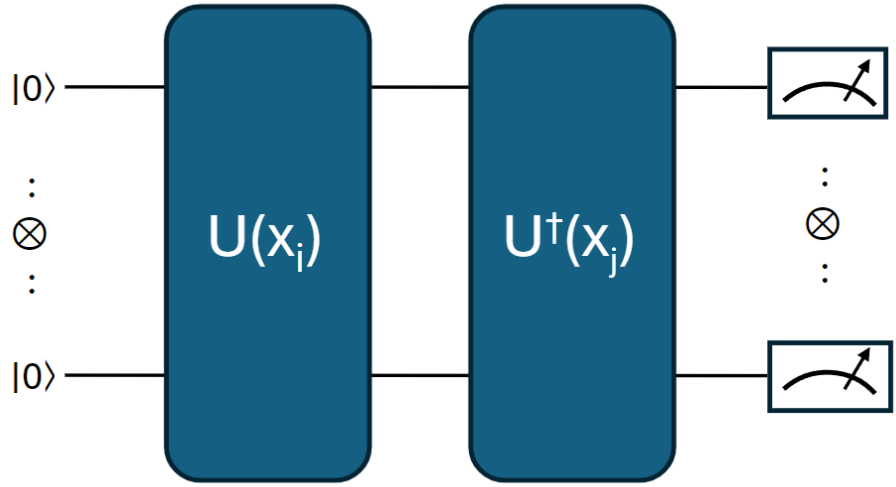


Figure 2.2: Quantum circuit representation for a fidelity kernel calculation. The circuit begins with all qubits initialised to the state $|0\rangle^{\otimes n}$, applies a unitary transformation $U(\mathbf{x}_i)$ to encode the input $\mathbf{x}_i$, followed by the adjoint transformation $U^\dagger(\mathbf{x}_j)$ for $\mathbf{x}_j$. Measurements are then performed to determine the overlap between the quantum states, allowing the computation of the fidelity kernel.

2.6.2 Swap Test

The swap test is another method of evaluating the inner product between two quantum states $|\psi(\mathbf{x}_i)\rangle$ and $|\psi(\mathbf{x}_j)\rangle$. The process begins with the quantum system initialised in the state:

$$|+\rangle \otimes |\psi(\mathbf{x}_i)\rangle \otimes |\psi(\mathbf{x}_j)\rangle, \tag{2.64}$$

where $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ is an auxiliary qubit in a superposition state. Controlled-SWAP gates are then applied to the two data states, conditioned on the auxiliary qubit. After applying the SWAP operation, the auxiliary qubit is measured. The probability of measuring the auxiliary qubit in the $|0\rangle$ state is directly related to the inner product:

$$\langle\psi(\mathbf{x}_i)|\psi(\mathbf{x}_j)\rangle. \tag{2.65}$$

If the states $|\psi(\mathbf{x}_i)\rangle$ and $|\psi(\mathbf{x}_j)\rangle$ are similar, the system remains close to the initial

configuration, effectively returning to $|0\rangle$ with high probability.

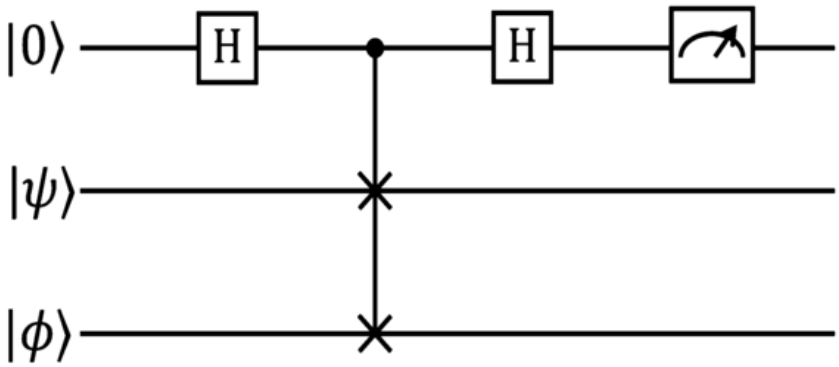


Figure 2.3: Quantum circuit for the swap test. The circuit begins with an auxiliary qubit initialised to $|0\rangle$ and two input states $|\phi\rangle$ and $|\psi\rangle$, each originally starting from state $|0\rangle^{\otimes n}$. A Hadamard gate is applied to the auxiliary qubit, followed by controlled-SWAP operations between the two input states. A final Hadamard gate is applied to the auxiliary qubit before measurement. The probability of measuring the auxiliary qubit in the $|0\rangle$ state determines the overlap $|\langle\phi|\psi\rangle|^2$ between the two input states [55].

2.6.3 Comparative Analysis

Both the quantum fidelity kernel and the swap test exploit the overlap between quantum states to compute kernel values. The fidelity kernel is computationally efficient for pure states and deterministic unitary evolutions, by avoiding the explicit computation of density matrices. The swap test, in contrast, is more versatile and can handle noisy or mixed quantum states but requires deeper circuits. Specifically, the swap test requires more than double the number of qubits required by the fidelity kernel to account for the separate encodings of x_i and x_j as well as the auxiliary qubit. The quantum fidelity kernel uses the reversibility property of unitary operations, where the adjoint (or Hermitian conjugate) of a unitary operation effectively reverses its action, allowing the comparison of two quantum states by encoding one quantum state using x_i and bringing the other quantum state back into alignment within the same quantum feature mapping.

In both approaches, the behaviour of the system relative to its initial state provides insight into the similarity of the quantum states. However, the quantum fidelity kernel will be used for its reduced circuit depth.

2.7 Quantum Convolutional Neural Network

QCNNs are hierarchical QML models designed for efficient classification tasks, inspired by classical Convolutional Neural Networks (CNNs). As described in Hur et al. [56], the QCNN architecture progressively reduces the number of qubits in a tree-like structure, analogous to the pooling operations in classical CNNs. This reduction allows QCNNs to operate with logarithmic circuit depth relative to the number of input qubits, making them better suited for NISQ devices.

A QCNN consists of layers of parameterised quantum circuits, including convolutional filters and pooling operations. The convolutional filters extract local features from the quantum states by applying two-qubit unitary operations. The pooling layers further reduce the dimensionality by tracing out subsystems after applying controlled quantum operations. This hierarchical structure ensures the model's scalability while retaining the ability to capture global correlations in data, which is often inaccessible to classical models.

The quantum state at each layer is a mixed state represented as:

$$|\psi_i(\theta_i)\rangle\langle\psi_i(\theta_i)| = \mathrm{Tr}_{B_i}(U_i(\theta_i)|\psi_{i-1}\rangle\langle\psi_{i-1}|U_i(\theta_i)^\dagger), \tag{2.66}$$

where $U_i(\theta_i)$ is the parameterised unitary operation for the i-th layer, and Tr_{B_i} represents the partial trace over subsystem B_i. The process begins with the input encoded into the quantum state $|0\rangle^{\otimes n}$ and proceeds layer by layer until a single qubit remains, which is measured to infer the class label.

Optimisation of a QCNN is achieved using classical computation by iteratively updating the parameters of PQC to minimise a predefined cost function. The cost function measures the difference between the QCNN predictions and the target labels for the training data. Mathematically, this can involve common loss functions, such as mean squared error or negative log-likelihood loss. Gradient-based optimisation methods, such as stochastic gradient descent or Adam optimiser, are used to compute the gradient of the cost function with respect to the variational parameters.

The hierarchical structure of QCNNs, inspired by tensor networks, mitigates the barren plateau problem by ensuring that the cost function gradients do not vanish exponentially with system size. Additionally, QCNNs use quantum entanglement to mimic the local

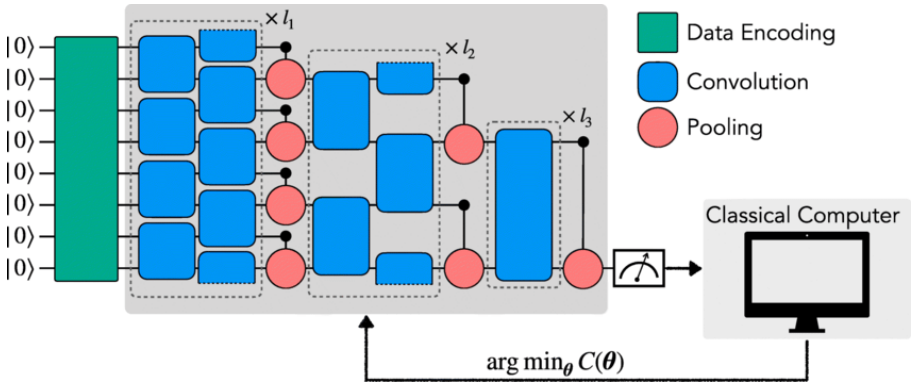


Figure 2.4: Architecture of a QCNN. The input data is encoded into quantum states (green) via a data encoding process. Convolution layers (blue) apply parameterised quantum operations to extract local features, while pooling layers (red) reduce the dimensionality of the quantum system by iteratively discarding qubits based on measurement outcomes. The classical computer optimizes the parameters θ of the quantum circuit by minimizing a cost function $C(\theta)$, allowing efficient hybrid quantum-classical training [56].

correlations captured by classical CNNs. These features, combined with their logarithmic scaling in circuit depth and parameter count, make QCNNs suitable candidates for NISQ devices.

2.7.1 QCNN as a Kernel Method

QCNNs can be interpreted as kernel methods because their mathematical structure are similar. As noted by Schuld [21, 57], QML models, including QCNNs, inherently map data into high-dimensional Hilbert spaces via quantum feature maps. The data encoding strategy in QCNNs and subsequent unitary operations facilitates this mapping, creating representations that allow inner products (kernels) between quantum states to act as similarity measures. These inner products, revealed through quantum measurements, provide the essential functionality of kernel computation.

Furthermore, QCNNs define a linear decision boundary in the feature space of quantum states, consistent with kernel methods where decision boundaries are formulated in transformed spaces. After the final layer of the QCNN, a measurement is performed on the quantum state, producing a classical outcome. This outcome is derived from the position of the quantum state in the quantum feature space. The model effectively defines a linear decision boundary in this transformed space, where the kernel evaluations have ensured the separability of the classes. This mirrors the mechanism of kernel methods in classical ML, where linear boundaries are constructed in a non-linear feature space transformed by a kernel function.

Chapter 3

Limited Quantum Speed Up

In Section 1.3, we introduce the concept of quantum advantage and quantum speed-up. To distinguish their difference, quantum advantage refers to demonstrating the efficiency and capabilities of using quantum hardware with a focus on raw computational throughput, while quantum speed-up focusses more on the performance of quantum algorithms relative to classical algorithms. While the focus of thesis looks at the enhancement of IDSs through f1 scores, one might think of how does quantum speed-up play a role in this or any detection metric? To link these two, we look at quantum speed-up as the advantage/value gained by computing classically intractable quantum feature maps. Classically intractable refers to problems that cannot be solved efficiently by classical computers. The reason for utilising classically intractable feature maps ($A_5, A_6, A_{13}, A_{14}, A_{19}$) is to generate sufficient representations of vectors in the Hilbert space of the quantum system to distinguish normal from malicious activity.

3.1 Quantum Turing Machine

Nielson et al. [3] provides an understanding of the solvability of quantum algorithms through the extensions of the Turing Machine.

The Church-Turing thesis establishes that any decision problem solvable by an algorithm can be computed on a Turing Machine. This idea was expanded into the Extended Church-Turing Thesis (ECTT), which posits that any problem efficiently solvable in a physical system can also be efficiently simulated on a classical Turing machine. Efficiency here is generally defined as execution within Polynomial Time (P), meaning the number of execution steps is $\mathcal{O}(n^k)$, where n is input size and $k \in \mathbb{N}_0$.

This classical perspective remained unchallenged until the advent of quantum computing, mainly through the discovery of Shor's algorithm [5]. This discovery revealed a class of computational tasks that are efficiently solvable on a quantum computer but intractable on a classical Turing machine, thereby challenging the ECTT.

To reconcile this shift, the Quantum-Extended Church-Turing Thesis (QECTT) was established. This thesis asserts that any computational process that can be efficiently executed in a physical system can also be efficiently simulated on a QTM, where efficiency is defined similarly to the classical case. By extending the classical thesis, the QECTT acknowledges the fundamental role of quantum mechanics in determining the limits of efficient computation in nature.

Reversely, in principle quantum computing can be simulated classically, though inefficiently, as evidenced by the use of classical simulations for small quantum systems. This concept is better understood through the use of the computational complexity class Bounded-Error Quantum Polynomial Time (BQP), which consists of decision problems solvable by a quantum computer in polynomial time with a probability of error less than $\frac{1}{3}$ for all instances. It is widely conjectured that BQP $\not\subseteq$ P, which means that there exist problems solvable in polynomial time on quantum computers that cannot be solved in polynomial time on classical computers. This conjecture is supported by the computational hardness of problems like integer factorisation, which is used in RSA encryption. Shor's algorithm, a quantum algorithm for factoring integers in polynomial time, highlights the stark contrast in efficiency between quantum and classical approaches, as no classical polynomial-time algorithm is known for this problem. The conjecture is further evidenced by quantum simulation, where simulating the evolution of a quantum system (a BQP-complete problem) on a classical Turing machine requires exponential resources $\mathcal{O}(2^n)$ in both memory and computational time.

QTMs utilise quantum superposition and entanglement to process exponentially many states simultaneously, known as quantum parallelism. Furthermore, the quantum feature maps used in this research uses structured entanglement designs to map input data into a high-dimensional quantum state space. This mapping creates highly non-local correlations between qubits, which efficiently represents complex relationships within the data. While QTMs can naturally process such entanglement, classical simulation of these correlations is highly inefficient due to the need to track exponentially many coefficients. Accurately mimicking these interactions classically would require brute-force computation of the entire quantum state vector.

3.2 Quantum Speed Up

We use the previous section as a guide to understand the relationship between classical and quantum algorithms. In the paper *"Defining and Detecting Quantum Speedup,"* Rønnow et al. [41] provides an outline of the different categories of quantum speed-ups.

A *provable quantum speedup* represents the most ideal form of quantum speed up. This occurs when a quantum algorithm's superiority over classical algorithms is mathematically proven, leaving no doubt that no classical counterpart can match its performance. Grover's search algorithm is an example of this category by demonstrating a provable quadratic speedup, reducing the complexity of searching an unsorted database from $\mathcal{O}(N)$ for the optimal classical algorithm to $\mathcal{O}(\sqrt{N})$, assuming the presence of an oracle. However, in practice, cases of provable quantum speed up is often limited to well-defined problems and assumptions.

However, when no such proof exists, the concept of a *strong quantum speedup* comes into play. This type of speedup measures the performance of a quantum algorithm against the best possible classical algorithm, even if that classical benchmark remains theoretical or unknown. Factoring is a classic example; while Shor's algorithm achieves polynomial scaling relative to the number of digits N, classical factoring algorithms are only known to have super-polynomial costs. However, the lack of a proven exponential lower bound for classical factoring complicates the certainty of this comparison.

Given the challenges of these two categories, a more practical approach drops the adjective and defines *quantum speedup* simply by comparing quantum performance with the best available classical algorithm, bypassing the need for the best possible classical algorithm. This approach acknowledges the evolving nature of the development of classical algorithms and focusses on tangible comparisons. However, it introduces subjectivity since the "best available" classical algorithm can vary over time and between implementation instances, especially in the case of intrusion detection.

When consensus on the best classical counterpart is difficult to find or the classical benchmark is deliberately chosen, *potential quantum speedup* becomes relevant. This concept evaluates quantum algorithms relative to a specific classical algorithm or a defined set of classical methods, rather than a universal standard. A notable example is the simulation of quantum systems, where quantum computing can achieve exponential speedup over brute force implementations of the Schrödinger equation.

However, the aforementioned speedups have the underlying assumption of a fully coherent, universal quantum computer capable of executing the quantum algorithms in question. In scenarios where quantum devices are limited in coherence, universality, or their ability to definitively outperform classical systems, the concept of *limited quantum speedup* emerges. This type of speedup compares quantum algorithms to classical counterparts that replicate the same algorithmic approach but on classical hardware. For instance, quantum annealing on a candidate quantum device may be compared to classical simulated annealing or other analogous classical methods. The distinction here lies in the recognition that these quantum devices might still offer measurable speedup within specific and controlled contexts. However, the results often depend on the chosen classical comparator.

To contextualise this research, which focuses on comparing different ansatzes to enhance the F1 scores of IDSs, the most appropriate categorisation would be limited quantum speedup. Firstly, provable quantum speedup does not apply because this research does not aim to provide a mathematical proof that no classical algorithm can match the performance of your quantum approach. Secondly, strong quantum speedup is also unlikely to apply because it involves benchmarking a quantum algorithm against the best possible classical algorithm, whether known or unknown. This requires knowledge of the optimal classical algorithm, which is often unavailable in practical domains like IDSs, where classical algorithms are continuously evolving. Although potential quantum speedup might seem plausible, it is an incomplete fit for these research objectives. Potential speedup involves demonstrating quantum performance improvements over specific classical benchmarks or algorithms without algorithmic equivalence. However, the focus on optimising ansatzes infers an evaluation within a shared algorithmic framework.

This research is most naturally associated with limited quantum speedup. This classification arises because the comparisons are constrained to classical algorithms that benefit through the use of quantum kernels. Specifically, optimising quantum circuits for IDSs reflects a direct focus on using quantum phenomena to improve performance within a structured algorithmic framework. The classical counterparts to the IDS—being kernel-based methods—share similar algorithmic frameworks, making limited quantum speedup the appropriate descriptor. Limited quantum speed-up accommodates scenarios where the performance advantage is contextual rather than universal and where the comparisons involve classical simulations that approximate quantum algorithms.

3.3 Barren Plateau Issue

Although limited quantum speed up is the benefit of using quantum feature mapping in QSVMs and QCNNs, quantum feature maps carry the risk of encountering barren plateaus in the case of QCNN parameter optimisation.

According to McClean et al. [58], randomly parameterised VQCs with sufficient depth can exhibit the properties of 2-designs, allowing them to approximate Haar randomness up to the second moment. However, circuits with high expressibility can suffer from *barren plateaus*-regions in the parameter space. In barren plateaus, the gradient of the cost function vanishes exponentially as a function of the number of qubits, making optimisation infeasible. This occurs because the circuit's state distribution becomes so uniform (akin to Haar randomness) that the search space is effectively flat, offering no useful gradient direction for optimisation. This presents a trade-off: while high expressibility ensures flexibility in representing quantum states, excessive expressibility risks making the parameter space untrainable, especially for deep circuits.

Practical circuit design must carefully balance this trade-off to achieve adequate state coverage without encountering barren plateaus. Cerezo et al. [59] show that short circuits can achieve local 2-design properties more efficiently than long circuits, which is critical for avoiding barren plateaus. From this finding, it was recommended that circuits with length $O(\log n)$ layers are used. The author mentions that global cost functions can attribute to the barren plateau problem, even for short circuits, and local cost functions mitigate the problem by focussing optimisation on individual qubits or small subsystems. This is where QCNNs can be advantageous. Unlike a QNN, the pooling layers in a QCNN effectively reduce the number of parameters involved in the optimisation at each stage, reducing the risk of gradients diminishing exponentially as the network deepens. Furthermore, *Quantum Advantage in Learning from Experiments* [60] shows that, unlike classical models which often need numerous epochs to achieve convergence, QCNNs exhibit strong representational power early in the training process. Experiments and simulations show that QCNNs converge to near-optimal solutions within a limited number of epochs [61].

Chapter 4

Quantum Machine Learning for Intrusion Detection

Drawing from the previous chapter, quantum feature maps fall under the BCP complexity class, which is associated with the notion of limited quantum speed-up, achieved by using quantum algorithms. This research looks into the use of highly expressive and entangled quantum feature maps to determine how IDSs perform when their quantum feature maps are more similar to haar randomness. Though resourcefully costly, it is possible to observe a theoretical understanding of any benefits gained by altering the design of the quantum feature maps through classical simulations of the quantum circuits. Small circuit depth is beneficial because the quantum state vector grows exponentially with the size of the quantum register, and in classical simulation, the full quantum state vector needs to be exhaustively computed.

In this chapter, the design considerations and implementation details of the QSVM and QCNN models for intrusion detection are presented. A minimalist approach is taken by leveraging only a few qubits and quantum operations, ensuring that the models remain classically computable for theoretical insights. The five most expressive ansatzes (Figure 1.5) from the paper "Expressibility and entangling capability of parameterised quantum circuits for hybrid quantum-classical algorithms" are used to distinguish this research from other proposed methods in Table 1.1.

The QSVM model, which relies on quantum kernel methods without trainable parameterised rotations, eliminates the need for iterative optimisation of parameters, thereby reducing the number of calls to the quantum device. The quantum fidelity kernel will be used because of its reduced circuit depth. By first examining the QSVM, we establish a baseline for evaluating any potential benefits of VQCs using QCNNs. Unlike data-driven

QSVM, QCNN use tunable parameters to enhance the model's adaptability to complex patterns. After theoretical insights are gained, quantum shots are then used for the best algorithm for further model expansion and NISQ-device adaptability.

The programming language used throughout the research is `python`. For data manipulation and analysis, `pandas` and `numpy` are used. Visualisation is handled with `matplotlib.pyplot`. Data preprocessing, including scaling, encoding, dimensionality reduction, and splitting, is implemented with modules from `sklearn`. Quantum computing components are designed with `pennylane` and its embeddings. For QCNN, optimisation is managed using `torch`, while the quantum circuits are built using the `hierarqcal` library. Finally, `sklearn.metrics` is used to calculate the F1 score for model evaluation. The full source code is available in a public GitHub repository [62].

4.1 Method

4.1.1 Data Preprocessing and Quantum State Preparation

The UNSW-NB15 dataset [63–67] obtained from the University of New South Wales is used in the training of the IDS. Each data point is a vector with dimensionality equal to the 49 features contained in the UNSW-NB15 dataset. The dataset contains both normal network traffic and synthetic attacks, which will be used in the supervised training of the binary classifiers.

In quantum computing, initialising a quantum state to the $|0\rangle$ state is standard practice because it provides a well-defined and consistent starting point for unitary evolution. By initialising n qubits to $|0\rangle$, quantum circuits ensure that any subsequent operations act predictably (or sequentially) from the known state:

$$|0\rangle^{\otimes n} = |0\rangle \otimes |0\rangle \otimes \cdots \otimes |0\rangle = |00\ldots0\rangle\,.$$

The size of the qubit register will vary depending on the encoding strategy implemented.

4.1.1.1 Angle Encoding and Principal Component Analysis

Angle encoding will encode the preprocessed features of the UNSW-NB15 dataset into the parameters of the x-rotational gates $R_x(\theta)$. Initially, the features are scaled in the range $[0, 2\pi]$. Since large-qubit registers are inefficient to classically simulate, the 49 features contained in the UNSW-NB15 dataset will require a reduction in dimensionality.

To reduce the dimensionality of the UNSW-NB15 dataset, PCA can be applied, followed by logistic regression classifier with 5 fold cross-validation to select the optimal number of principal components. PCA uses a weighted average of the features, called the principal components, that capture the most variance in the data. The first component explains the largest variance, followed by components with decreasing explained variance. Figure 4.1 below displays the cumulative proportions of the total variance in the data explained by the varying number of PCA components. Following the minimalist approach, the best trade-off between the number of PCA components and the amount of variance explained is observed to be 7 principal components.

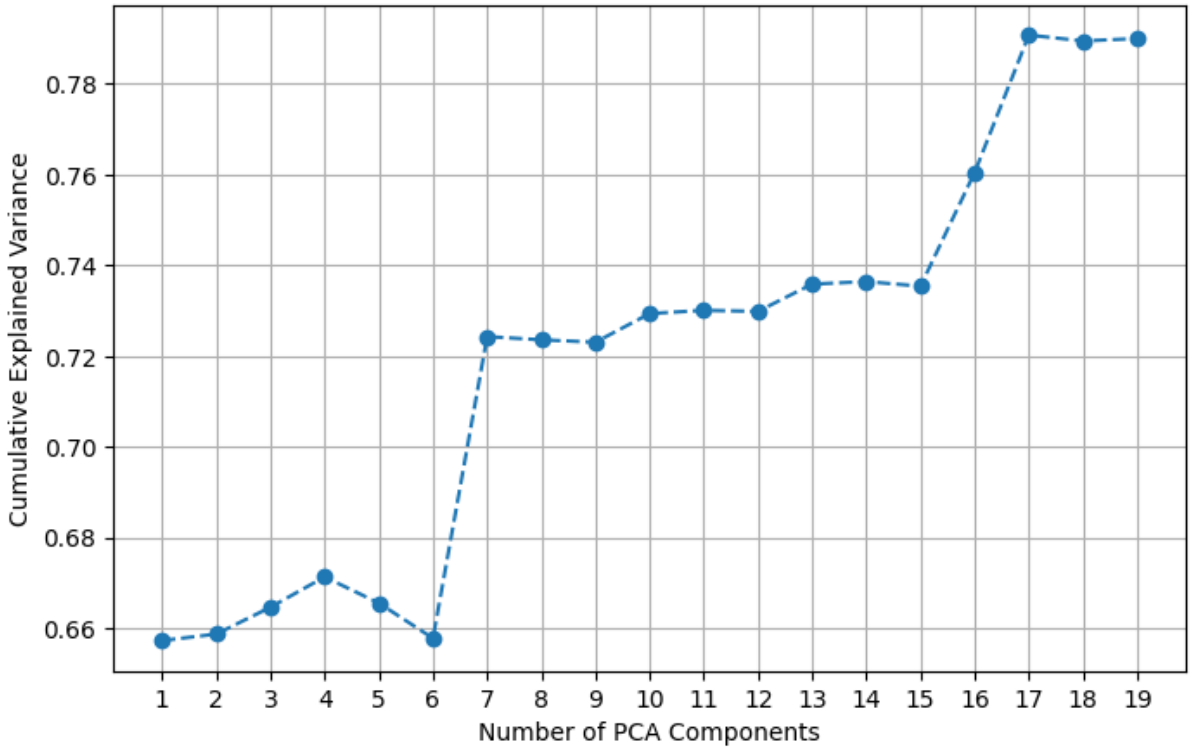


Figure 4.1: Cross-validation scores of the variances explained in the UNSW-NB15 dataset against the number of PCA components.

4.1.1.2 Amplitude Encoding

Amplitude encoding will encode a normalised feature vector of the UNSW-NB15 dataset into the amplitudes of a quantum state. Given that the UNSW-NB15 dataset has 49 features, the normalised feature vector must be padded to fit the next power of 2 greater than 49. The next power of 2 is $2^6 = 64$. Therefore, the feature vector will be padded with zeros to bring its size to 64.

The quantum system will require the initialisation of 6 qubits when amplitude encoding is used. The 49 features from the original dataset, padded with 15 zeros, will serve as the amplitude coefficients of a 6-qubit quantum state. The quantum system will have a total of $2^6 = 64$ basis states, ranging from $|000000\rangle$ to $|111111\rangle$.

4.1.2 Measurement Strategy

Two measurement strategies will be utilised: limited quantum shots and ideal statistics.

In practice, quantum computers are based on probabilistic measurements. Due to the inherent randomness in quantum mechanics, running the same quantum circuit multiple times, known as *quantum shots*, may yield different outcomes based on the probabilistic nature of quantum state collapse. The class prediction (normal traffic or attack) is determined by the most frequent outcome of the collapsed quantum system. More shots lead to a better approximation of the underlying probability distribution. The maximum number of shots available for publicly accessible IBM Quantum Processing Units is 20,000 shots, as detailed in IBM Quantum resources [9].

Ideal statistics refer to the theoretical probabilities associated with the outcomes of a quantum circuit. In other words, ideal statistics describe the statistical convergence as the number of quantum shots tends to infinity. Ideal statistics ensure the correctness of the algorithm's theoretical foundation, while quantum shots assess the practical implementation on quantum hardware.

The measurement of both QSVM and QCNN models will use ideal statistics (∞) to theoretically understand its effectiveness. The best performing algorithm, QSVM, will then undergo the initial steps to adapt it to real-world applicability by using two different shot counts: $(1024, 8196)$. Since quantum computers are still in their infancy, fewer shot counts are favoured due to the costs associated with operating a present-day NISQ device.

4.1.3 Metrics

To assess the efficacy of the proposed models, standardised metrics are used for evaluation. A *confusion matrix* will categorise the prediction outcomes, consisting of:

- True Positives *(TP)*: Attack records are correctly identified.
- True Negatives *(TN)*: Non-attack records correctly classified.
- False Positives *(FP)*: Non-attack records incorrectly classified as attacks.
- False Negatives *(FN)*: Attack records not correctly detected.

These are used to calculate accuracy, precision, recall, and the F1 score.

Accuracy will be reported as it is a widely used metric for classification tasks. It is defined as:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} = \frac{TP + TN}{\text{Total Instances}} \tag{4.1}$$

However, its effectiveness is highly dependent on the class distribution of the dataset. In imbalanced datasets, where one outcome class significantly outweighs the other, accuracy may provide a misleading evaluation of the model's performance. For example, if 95% of the traffic is normal and only 5% represents intrusions, a model that predicts all traffic as normal achieves 95% accuracy. Accuracy is a problematic metric for IDSs because it does not adequately address the imbalanced nature of intrusion detection datasets and does not differentiate the costs of false positives (FP) and false negatives (FN). Therefore, the analysis of the results will minimally focus on accuracy.

Precision measures the proportion of correctly identified positive instances out of all instances predicted as positive. It is defined as:

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

Recall, also known as sensitivity, measures the proportion of actual positive instances that were correctly identified by the model. It is defined as:

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

While precision and recall are essential for evaluating the model's performance, each has its limitations. Precision does not account for false negatives, meaning that it can be artificially high when the model avoids predicting positives to minimise false positives. In contrast, recall does not consider false positives, so it can be overly optimistic in cases where the model predicts many positives, regardless of their correctness. These weaknesses highlight the need for a balanced metric, such as the F1 score. It is defined as:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{4.4}$$

A high F1 score indicates that the model is effective in minimising false positives and false negatives, ensuring reliable threat detection.

4.2 Model Development

4.2.1 QSVM Architecture

The QSVM is a hybrid quantum-classical algorithm that maps input data into a high-dimensional Hilbert space using a quantum kernel. The quantum kernel function measures the similarity between two data points via the inner product of their corresponding quantum states, computed using a quantum circuit. The measurement at the end of the circuit collapses the quantum state and extracts information about the similarity between the encoded data points, contributing to the construction of the Gram matrix. Each entry in the Gram matrix represents the kernel value between two data points. A classical SVM is subsequently trained using this quantum kernel matrix, allowing predictions by evaluating the similarity between test and training data in the quantum-enhanced feature space.

For this study, the kernel matrix is computed using both amplitude and angle encoding strategies. In amplitude encoding, the data are normalised so that each feature vector forms a unit norm, mapping classical data to the amplitudes of quantum states. As amplitude encoding does not map data into rotation angles, no parameterised quantum circuits $A(x)$ are utilised.

To encode classical data using angle and amplitude encoding, a unitary operation $U(x)$

is applied to the initialised quantum state, transforming the input vector x into a quantum state.. This transformation maps the data into the Hilbert space through quantum feature embedding. The quantum kernel is evaluated by applying the Hermitian conjugate $U^\dagger(x_j)$ to the quantum state $U(x_i)\,|0\rangle$. When the encoded states have significant overlap, $U^\dagger(x_j)$ effectively reverses the transformation $U(x_i)$; when they are orthogonal, the overlap approaches zero. The kernel matrix for the amplitude encoding is computed as:

$$K(x_i, x_j) = |\langle x_j | x_i \rangle|^2$$

The PCA components are encoded using x-rotation gates, $U(x)$, and uses parametrisable ansatzes $A(x)$ to enhance the embedding of the features. The fidelity-based quantum kernel is expressed as:

$$K(x_i, x_j) = \left| \langle 0 | U^\dagger(x_j) A_k^\dagger(x_j) A_k(x_i) U(x_i) | 0 \rangle \right|^2$$

The kernel computations are performed with five ansatzes $(A_5, A_6, A_{13}, A_{14}, A_{19})$ and three different shot counts $(\infty, 1024, 8196)$. Each kernel computation applies the ansatz A_k and its Hermitian conjugate $A_k^\dagger$ to the encoded quantum states, deriving the kernel values from quantum state overlaps. The data set is divided into training and testing sets in 15 random states to ensure reproducibility and variability. To minimise the size of the Gram matrix produced, 100 data points will be used for the training set, and 20 data points will be used for the testing set. Finally, the quantum kernel matrix is used to train a classical SVM implemented via `scikit-learn`.

Algorithm 1 QSVM with Multiple Ansatzes and Encoding Strategies

Require: Dataset $\{(x_i, y_i)\}_{i=1}^{n}$, encoding strategy (*angle* or *amplitude*), ansatzes $\{A_5, A_6, A_{13}, A_{14}, A_{19}\}$, shot counts $\{\infty, 1024, 8196\}$, random states $\{r_1, r_2, \ldots, r_{15}\}$ for dataset splitting

Ensure: F1 score for each random state, ansatz and shot count

for each random state $r_l \in \{r_1, r_2, \ldots, r_{15}\}$ **do**
  Split dataset into training and testing sets using random state r_l.
  if encoding strategy is *angle* **then**
    Scale each feature x_i to $[0, 2\pi]$ using:
    Perform PCA
    for each ansatz $A_k \in \{A_5, A_6, A_{13}, A_{14}, A_{19}\}$ **do**
      for each shot count $s_j \in \{\infty, 1024, 8196\}$ **do**
        Encode the quantum kernel circuit using ansatz A_k, followed by $A_k^\dagger$
        Compute the quantum kernel matrix using A_k with s_j shots.
        Train the SVM classifier on the training subset via `scikit-learn`.
        Evaluate the classifier on the test subset and record F1 score.
      end for
    end for
  else if encoding strategy is *amplitude* **then**
    Normalize each feature vector x:
    for each shot count $s_j \in \{\infty, 1024, 8196\}$ **do**
      Encode the quantum kernel circuit using amplitude encoding, followed by its Hermitian conjugate.
      Compute the quantum kernel matrix with s_j shots.
      Train the SVM classifier on the training subset via `scikit-learn`.
      Evaluate the classifier on the test subset and record F1 score.
    end for
  end if
end for
Compute and save the average and best f1 scores.

4.2.2 QCNN Architecture

QCNNs are quantum analogues of classical Convolutional Neural Networks (CNNs), consisting of alternating layers of quantum convolutional operations and pooling operations. The convolutional layers apply local unitary transformations on subsets of qubits, similar to how classical convolutions operate on small patches of input data. The pooling layers reduce the number of qubits by applying CNOT gates, allowing efficient dimensionality reduction. The parameters of a QCNN are optimised using classical optimisers that

adjust the angles of the quantum gate parameters to minimise a cost function.

Therefore, QCNN is implemented within a VQC framework, where unitary operations $A(\theta)$ are governed by trainable parameters. This allows the five parameterised ansatzes $(A_5, A_6, A_{13}, A_{14}, A_{19})$ to be used in both the amplitude and angle encoding strategies in 15 random states. The parameter θ is initialised randomly and iteratively updated after every epoch. The training process uses the negative log-likelihood loss (NLL) cost function to evaluate the model performance. Because there is no Gram matrix construction required, the training and testing datasets will increase to 1000 and 200 data points, respectively.

A global cost function will be adopted because it reduces the number of quantum circuit executions compared to a layerwise (local) cost function. Although a layer-wise cost function reduces the number of parameters to be trained, it requires optimisation for each layer independently, implying the rerunning the quantum circuit as many times as there are layers for every optimisation step. This can become costly for both classical simulation and quantum hardware implementation. Instead, training a larger number of parameters minimises the number of quantum circuit executions required. Contributing to the minimalist approach is the use of a small number of epochs, with the expectation of rapid convergence to near-optimal solutions in the training of QCNN, as supported by the findings of [61].

The number of layers in the QCNN is restricted to a maximum of 2 layers, following the findings of [59] which suggests using $O(\log n)$ layers. This implies that for the case of the 1-layer QCNN, the pooling layer will also act as the fully connected layer. The number of parameters that must be trained can be seen in Table 4.1. The first pooling layer for the 2-layer QCNN will collapse 3 qubits, implying that the next convolutional layer will contain 4 qubits in the case of angle encoding and 3 qubits in the case of amplitude encoding. Since PCA ranks its components by the amount of variance explained with the first principal component explaining the most variance, the pooling layers will strategically collapse the qubits in an upward manner. This implies that the first qubit will be the fully connected qubit and, therefore, the qubit that is measured.

Kukliansky et al. [48] conducts a comprehensive grid search to explore various hyperparameter configurations. A focused grid search will be performed to analyse the impact of these hyperparameter configurations against this research's models. To gain insight on the trends of the hyperparameters such that a reduced hyperparameter configuration can be adopted in later testing, the following configurations were initially tested:

	Amplitude Encoding		Angle Encoding	
	1 Convolutional Layer	**2 Convolutional Layers**	**1 Convolutional Layer**	**2 Convolutional Layers**
Ansatz5	54	72	70	98
Ansatz6	54	72	70	98
Ansatz13	24	36	28	44
Ansatz14	24	36	28	44
Ansatz19	18	27	21	33

Table 4.1: Comparison of total Trainable Parameter sizes.

- **Epoch Sizes:** $5, 10, 20, 50$
- **Learning Rates (η):** $0.1, 0.05, 0.02, 0.015, 0.01, 0.005, 0.001$.
- **Optimisers:** Adam and SGD (Stochastic Gradient Descent) with momentum values $0, 0.2, 0.3$ and decay rates $0, 0.001, 0.01$.

The initial grid search kept the batch size constant at size 32 since the aforementioned reference paper found no significant differences as discusses in Section 1.4.1. The next section discusses the trends of the initial extensive grid search of the hyperparameters and concludes with a reduced hyperparameter set that will be used for this research. The results focus on the trends observed across key hyperparameters: learning rate (η), decay rate, optimiser type, number of epochs and momentum (for SGD).

4.2.2.1 Hyperparameter Grid Search

The learning rate significantly influenced the performance of both optimisers (SGD and Adam). For high learning rates ($\eta = 0.1$), Adam and SGD achieved their peak performance within fewer epochs. Adam, for example, reached high validation F1 scores at 10 epochs without decay. SGD required longer training (50 epochs) and slight decay ($decay = 0.01$) to achieve comparable performance. Medium Learning Rates ($\eta = 0.01 - 0.05$) resulted in more stable but slower convergence. Low Learning Rates ($\eta = 0.001$) did not provide significant updates to the model parameters, resulting in flat performance regardless of decay or epoch count. Both optimisers exhibited consistent lower F1 scores in different decay settings and epochs, indicating a plateau in learning.

The decay rates had different effects for the different optimisers. No Decay ($decay = 0$) generally provided superior results for both optimisers, especially at high learning rates ($\eta = 0.1$). Moderate Decay ($decay = 0.001$) slightly reduced performance across configurations. High Decay ($decay = 0.01$) were detrimental to Adam but occasionally

beneficial to SGD. With regard to only SGD, momentum ($momentum = 0.3$) improved the performance of SGD, especially at higher learning rates. It allowed faster convergence by stabilising parameter updates, achieving F1 scores comparable to those of Adam in longer training sessions.

The number of epochs had a predictable effect on the performance. Short training (< 10 epochs) preferred the Adam optimiser due to its faster convergence, especially with the learning rate $\eta = 0.1$. Moderate training ($10 - 20$ epochs) was found to be the best in this range, with both optimisers approaching their highest F1 scores. Long training (> 20 epochs) showed diminishing performance gains, with both optimisers showing minor improvements. However, SGD benefited more from the higher epochs when paired with high momentum.

The Adam optimiser exhibited faster convergence and higher peak performance in fewer epochs. Adam is well suited for high learning rates without decay. However, SGD maintained stability during prolonged training sessions, especially with momentum. It achieved a performance comparable to that of Adam with sufficient training and proper decay settings. Therefore, for rapid convergence, Adam with high learning rates ($\eta = 0.1 - 0.01$), $decay = 0$, and 10 epochs should be used. For long training sessions, SGD with high learning rates ($\eta = 0.1 - 0.01$), $momentum = 0.3$, $decay = 0.01$, and 20 epochs should be used.

Algorithm 2 QCNN Experiment

Require: Dataset $\{(x_i, y_i)\}_{i=1}^{n}$, encoding strategy (*amplitude* or *angle*), ansatzes $\{A_5, A_6, A_{13}, A_{14}, A_{19}\}$, random states $\{r_1, r_2, \ldots, r_{15}\}$, hyperparameter grid (layer size $\{1, 2\}$, batch size 32, learning rates $\{0.1, 0.05, 0.02, 0.01\}$, optimisers, momentum $\{0.3\}$, no decay rates)

Ensure: Model performance metrics (e.g., accuracy, F1-score) for each configuration

for each random state $r_l \in \{r_1, r_2, \ldots, r_{15}\}$ **do**
  Split the dataset into training and testing sets according to batch size using random state r_l.
  for each encoding strategy (*amplitude* or *angle*) **do**
    Preprocess the data:
    if encoding is *amplitude* **then**
      Scale each feature x_i to $[0, 1]$
      Normalize each feature vector x
    else
      Scale each feature x_i to $[0, 2\pi]$
      Apply PCA
    end if
    for each ansatz $A_k \in \{A_5, A_6, A_{13}, A_{14}, A_{19}\}$ **do**
      for each hyperparameter configuration (layer size, batch size, learning rate, optimiser, momentum) **do**
        Generate the Quantum Circuit with the given ansatz A_k, layer size and encoding strategy via `hierarqcal`
        if Optimiser is *SGD* **then**
          Initialize the QCNN with each learning rate, optimiser, momentum.
        else
          Initialize the QCNN with each learning rate, optimiser.
        end if
        for epochs in $e_p \in \{10, 20\}$ **do**
          Train the QCNN on the training set for e_j epochs.
          Evaluate the QCNN on the testing set and record performance metrics.
        end for
      end for
    end for
  end for
end for
Compute and save the average and best f1 scores.

4.3 Experiment Overview

Table 4.2 summarises the experimental configurations for the QSVM and QCNN models. Both models were evaluated using amplitude and angle encoding strategies, with the QSVM using a data-driven parameter method and the QCNN implementing a trainable quantum architecture.

	QSVM		**QCNN**	
	Amplitude	**Angle**	**Amplitude**	**Angle**
Number of Qubits	6	7	6	7
Ansatzes	N/A	A5, A6, A13, A14, A19	A5, A6, A13, A14, A19	A5, A6, A13, A14, A19
Shot Levels	$\infty, 1024, 8196$	$\infty, 1024, 8196$	∞	∞
Random States	15 random states	15 random states	15 random states	15 random states
Layer Sizes	N/A		1, 2	1, 2
Epoch Sizes	N/A		10, 20	10, 20
Batch Size	N/A		32	32
Learning Rates	N/A		0.1, 0.05, 0.02, 0.01	0.1, 0.05, 0.02, 0.01
Optimisers	N/A		Adam, SGD	Adam, SGD
Momentum (SGD)	N/A		0.3	0.3

Table 4.2: Overview of QSVM and QCNN Experiments

A subset of the quantum circuits used in the research can be viewed below. Figure 4.2 and 4.3 are implementations of the fidelity based QSVM using amiplitude encoding (raw feature map) and angle encoding respectfully. Figure 4.4 shows the implementation of QCNN architecture using the two different layer sizes.

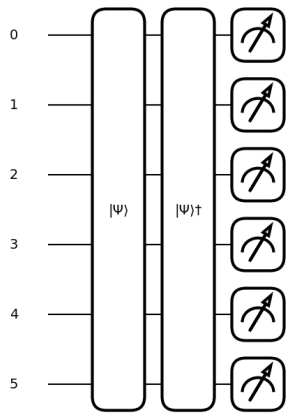


Figure 4.2: Example of the raw feature map implementation of the QSVM model using amplitude encoding. Amplitude encoding is shown through the application of $|\Psi\rangle$, followed by its conjugate transpose $|\Psi\rangle^\dagger$ before the qubits are measured.

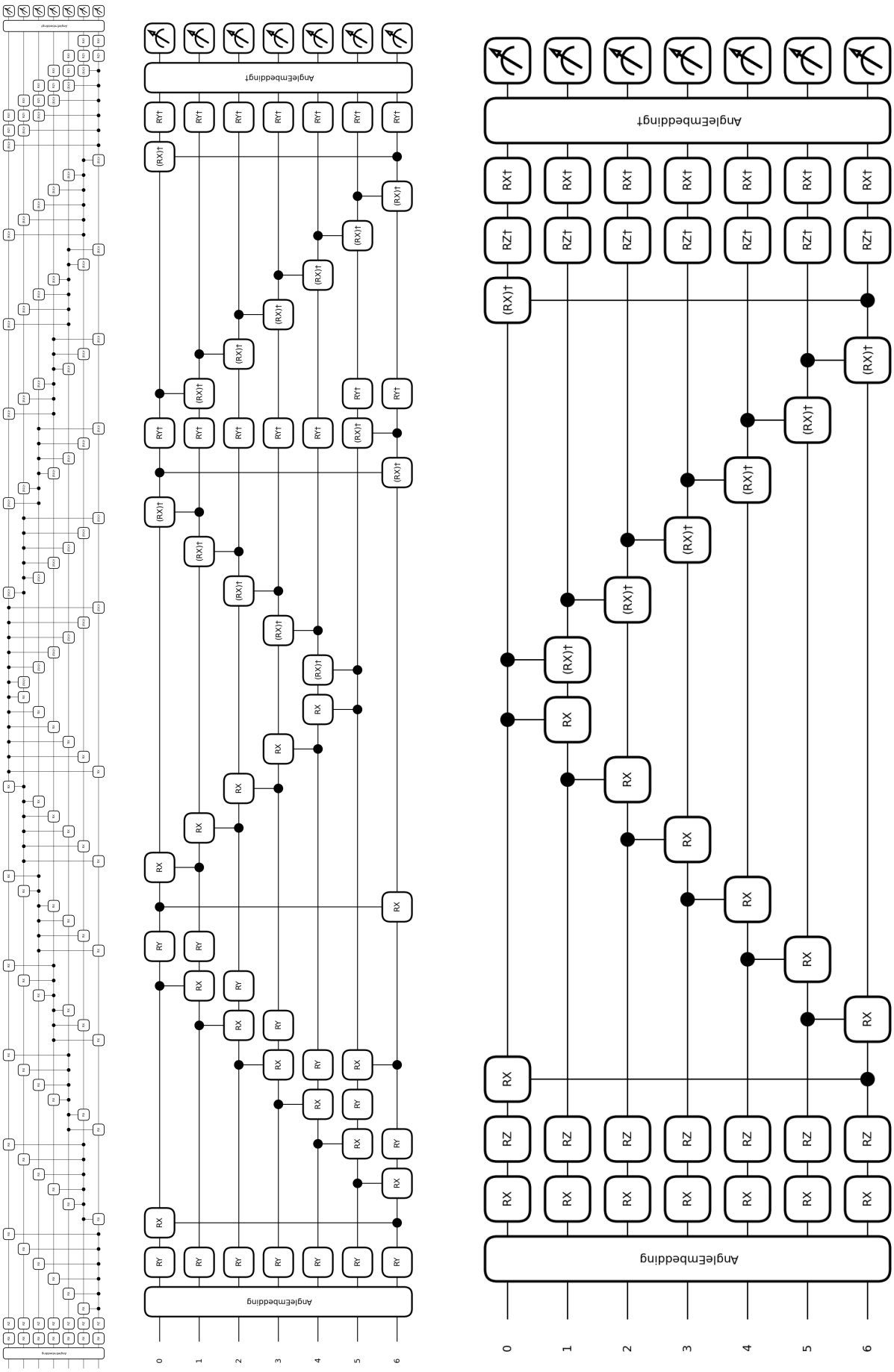


Figure 4.3: Examples of QSVM implementations used in this research. Each circuit here undergoes angle embedding using 7 qubits, followed by one or two qubit rotations, and is met by its Hermitian adjoint in the second half of the circuit before measurement. The top circuit uses A_6, the middle uses A_{14}, and the bottom circuit uses A_{19}.

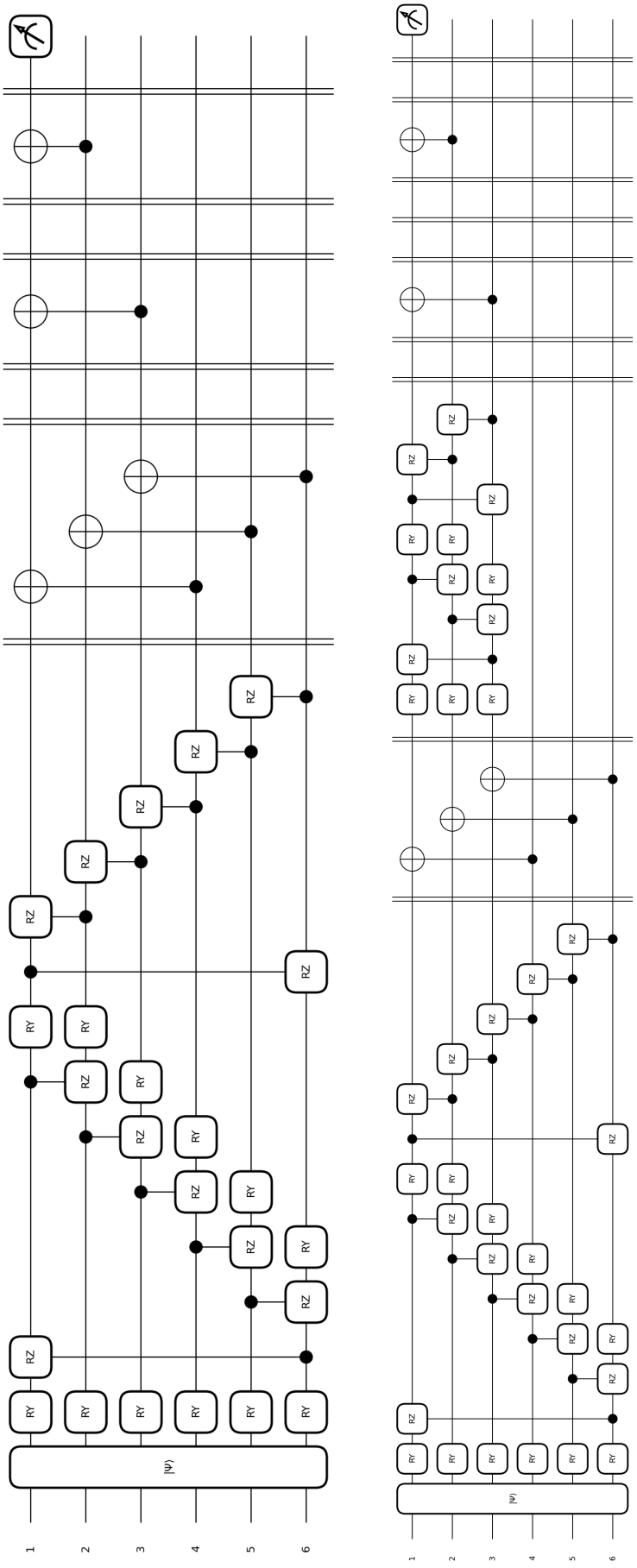


Figure 4.4: Examples of QCNN implementations used in this research. Each circuit here undergoes amplitude embedding using 6 qubits, followed by the first convolutional layer using A_{13}. The pooling layer is then applied using CNOT gates. The top circuit implements one layer, therefore, the following CNOT gates form the fully connected layer and the upper most qubit is measured. The bottom circuit uses two layers, therefore A_{13} is applied again on the remaining qubits in use. After the second convolutional layer, the circuit is met by the fully connected layer comprising of CNOT gates and the the upper most qubit is measured.

Chapter 5

Results and Discussion

For the analysis and comparison of the results, three statistical tests were used: the Shapiro-Wilk test, the Kruskal-Wallis test, and the Wilcoxon signedrank test. In all three statistical tests, a significance level, or p-value, of 0.05 was used. The Shapiro-Wilk test was first applied to determine the normality of the result data distributions. This test determines whether the data conform to a normal distribution, which is assumed to be true for many parametric statistical methods. If the p-value from this test is less than the predefined significance level , the null hypothesis is rejected, indicating that the data does not follow a normal distribution. For both the QSVM and QCNN results, the Shapiro-Wilk test revealed non-normal distributions ($p < 0.05$), prompting the use of non-parametric methods for in-depth analysis.

The Kruskal-Wallis test was used to compare multiple groups, such as different ansatz designs, hyperparameter configurations, or shot counts. This test evaluates the null hypothesis that the medians of all considered groups are equal. If the p-value is below the significance level, the null hypothesis is rejected, indicating that at least one group's median differs significantly from the others.

The Wilcoxon signed-rank test was used to compare the observed performance of the QCNN and QSVM models against benchmark values. This test evaluates the null hypothesis that the median difference between the paired observations (the model results and their corresponding benchmarks) is zero. A p-value below the significance level suggests rejecting the null hypothesis, indicating that the observed performance differs significantly from the benchmark.

5.1 QSVM Analysis

This section focusses on the results of the QSVM experiments using six ansatz structures. Among these, five ansatzes utilised angle embedding, while the sixth ansatz applied amplitude embedding without a feature map (raw implementation). The experiments were executed using shot-based sampling. Given the non-normal distribution of the test results, verified through the Shapiro-Wilk test, the Kruskal-Wallis statistical test was used to analyse differences across ansatz configurations.

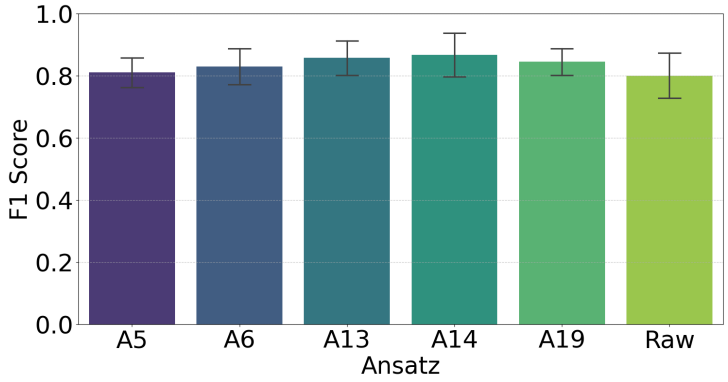


Figure 5.1: F1 Score Distribution Across Ansatz Functions

The F1 distribution for all six ansatz functions is seen in Figure 5.1. The angle-encoding ansatzes ($A_5, A_6, A_{13}, A_{14}, A_{19}$) achieve comparable F1 scores, with no statistically significant differences detected by the Kruskal-Wallis test with a p-value of 0.058. However, when amplitude embedding (raw implementation) is used, the p-value changes to 0.03 and a statistically significant difference is observed. Specifically, the amplitude embedding approach shows slightly lower F1 score compared to the angle encoding methods. This suggests that the lack of a feature map in the raw amplitude implementation may affect the model's ability to capture relevant patterns within the data, leading to reduced performance. Notably, A_{13} and A_{14} yield the highest F1 scores, suggesting that these ansatz structures are well suited for the QSVM implementation.

The analysis of the F1 scores across three different shot counts ($1024, 8192$, and ∞) reveals no observable differences in performance. This uniformity could be due to the absence of noise models, as the experiments were conducted under noiseless simulation conditions. In a noiseless environment, the shot count has minimal influence on the results, as the ideal quantum measurements are not affected by hardware-induced noise. This points to the need for further studies incorporating realistic noise models to better understand the impact of shot counts on F1 scores and the robustness of the model in practical NISQ scenarios.

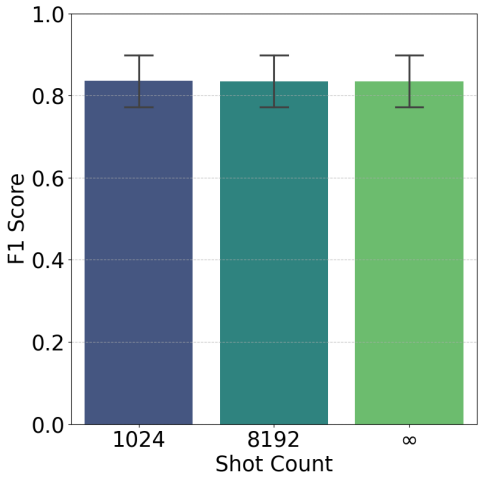


Figure 5.2: F1 Score Distribution Across Shot Counts

5.2 QCNN Analysis

This section shows the performance analysis of the QCNN using multiple configurations. The configurations include two optimisers (Adam and SGD), four learning rates (0.1, 0.05, 0.02, and 0.01), two epoch sizes (10 and 20), one-layer versus two-layer architectures, and amplitude versus angle embedding encoding methods. Similarly to QSVM, the non-normal distribution of the test results, confirmed using the Shapiro-Wilk test, required the use of the Kruskal-Wallis test for statistical analysis.

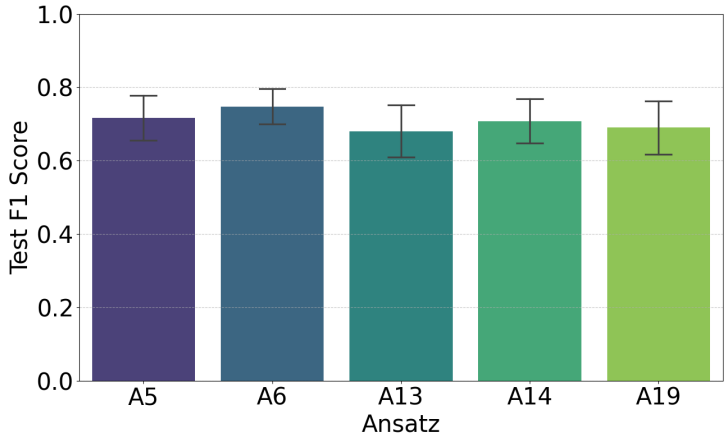


Figure 5.3: Test F1 Score Distribution Across Ansatz

Before conducting in-depth comparisons of various QCNN configurations, Figure 5.3 provides an overview of the overall performance across all ansatz designs. A Kruskal-Wallis test was performed, resulting in a p-value of 0. This result suggests that there are statistically significant differences in the performance medians among the ansatz designs configurations. Overall, A_6 achieves the highest F1 score across all hyperparameter configurations.

Figure 5.4 illustrates the impact of four different learning rates (0.1, 0.05, 0.02, and 0.01) on the test F1 scores across all ansatz configurations. Across all ansats, there is a statistical difference between the learning rates with a p-value of 0. However, when a learning rate of 0.1 is excluded, there is no significant difference (p-value = 0.839). Although no statistically significant differences were detected between the remaining learning rates, a learning rate of 0.02 tends to exhibit slightly improved test F1 scores in some cases, especially for A_{19}. The minimal variability observed in these lower learning rates suggests that they offer more stable convergence.

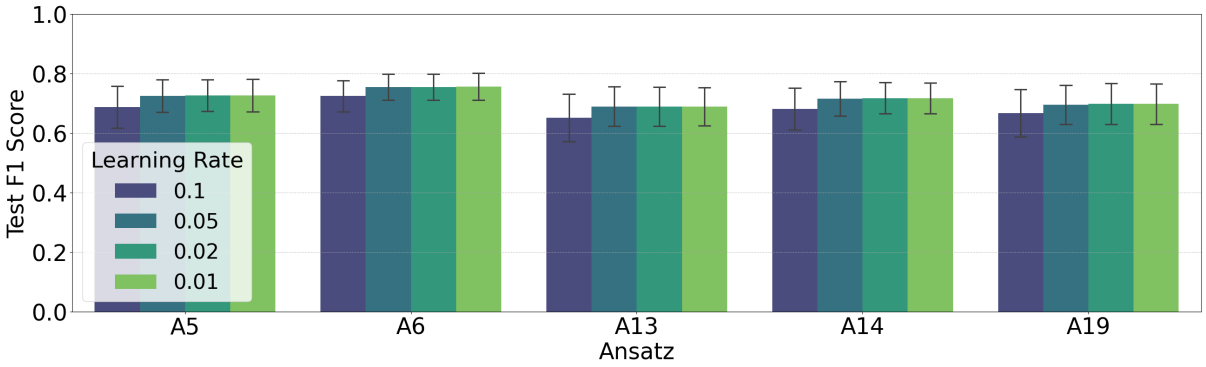


Figure 5.4: Test F1 Score Distribution Across Ansatz and Learning Rate

The comparison of the F1 test scores between the Adam and SGD optimisers is shown in Figure 5.5. Across all ansatz structures, the Adam optimiser achieves slightly higher test F1 scores than the SGD optimiser. The Kruskal-Wallis statistical test indicates a significant difference between the two optimisers with a p-value of 0, however, this is influenced by the results of learning rates. Figure 5.6 shows the F1 scores for the various learning rates when paired with a specific optimiser. When the learning rate of 0.1 is dropped from the optimiser comparisons, a p-value of 0.734 is observed, which implies that there are no statistical differences between the optimisers when low learning rates are applied.

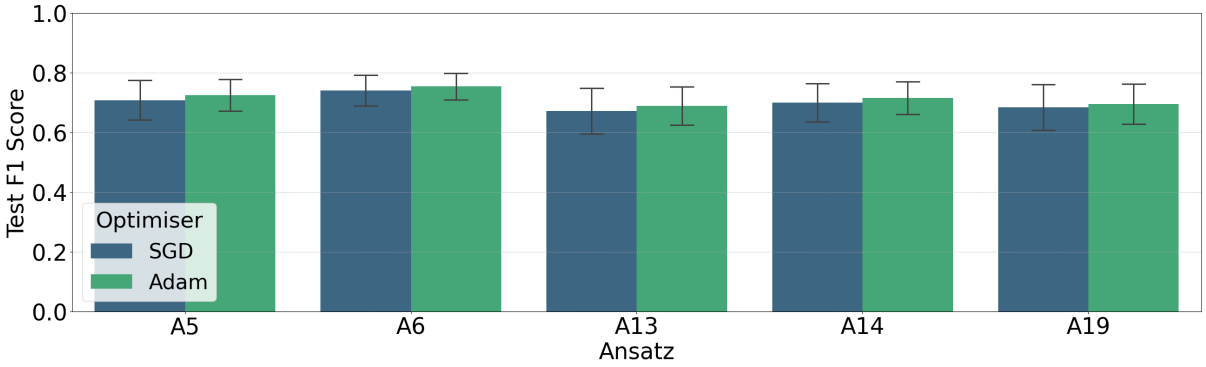


Figure 5.5: Test F1 Score Distribution Across Ansatz and Optimiser

To evaluate the impact of training duration, F1 test scores were analysed for two epoch

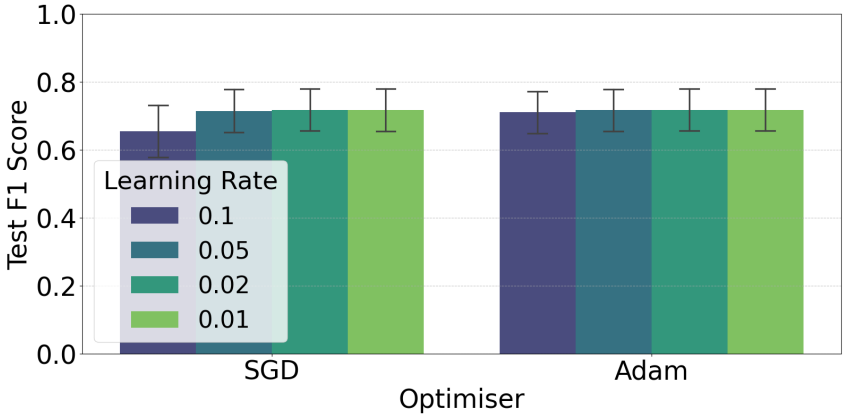


Figure 5.6: Test F1 Score Distribution Across Learning Rate and Optimiser

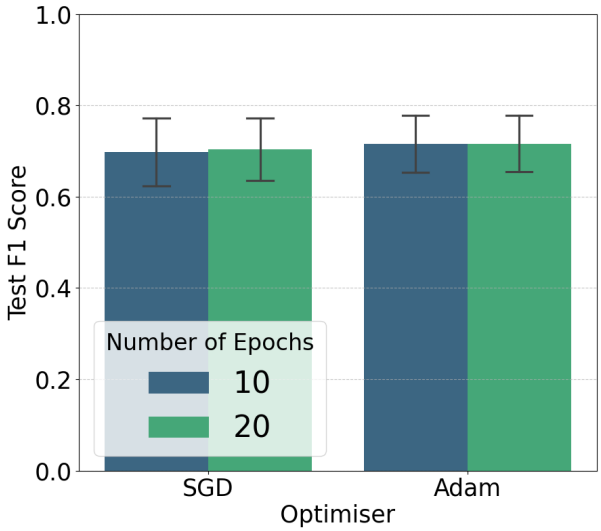


Figure 5.7: Test F1 Score Distribution Across Number of Epochs and Learning Rate

settings: 10 and 20 epochs. The results, shown in Figure 5.7, show a minor improvement in performance when the duration of the training increases to 20 epochs. However, this improvement is not statistically significant with a p-value of 0.482.

Comparison of QCNN architectures of one layer and two layers reveals a statistically significant difference (p-value = 0) in test F1 score for the two-layer configuration, as illustrated in Figure 5.8. Across all ansatz configurations, the two-layer QCNN consistently outperforms the one-layer variant, notably in A_{13}. However, when focussing specifically on A_6, a p-value of 0.126 reveals no statistical difference between the implementation of one layer and two layers.

The performance of angle embedding and amplitude embedding methods is compared in Figures 5.9. Both encoding strategies achieve comparable test F1 scores and accuracies across all ansatz configurations. While amplitude embedding shows slightly lower

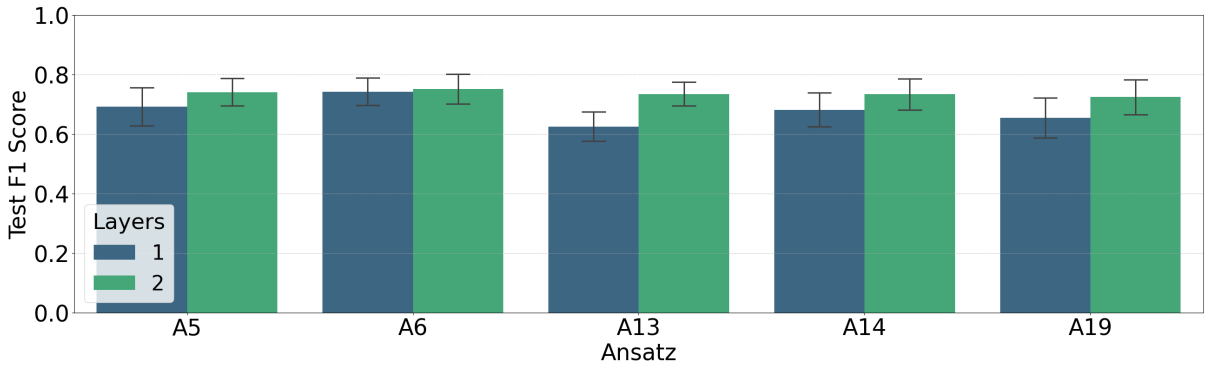


Figure 5.8: Test f1 Score Distribution Across Ansatz and Layers

variability in performance, the Kruskal-Wallis test indicates no statistically significant difference between the two encoding methods with a p-value of 0.109. However, once again focussing on A_6, a p-value of 0 reveals a statistical difference between the angle embedding and the amplitude embedding, favouring the angle embedding implementation for A_6.

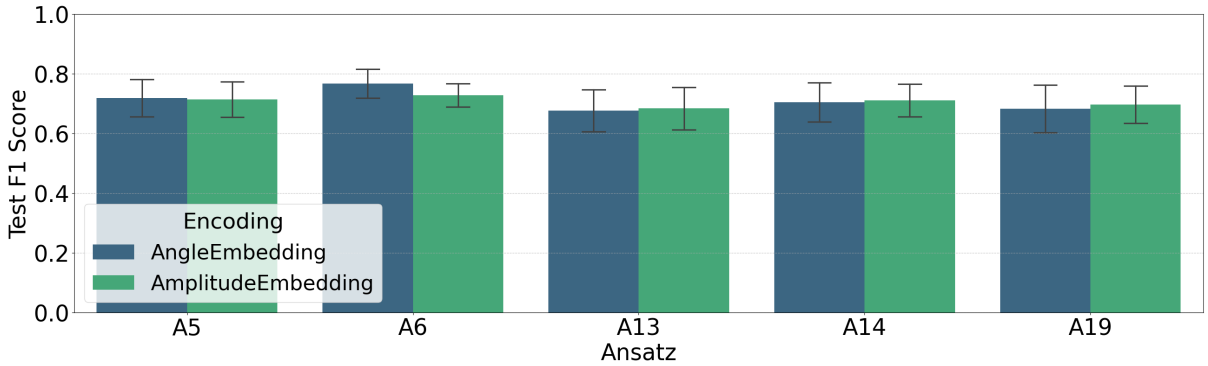


Figure 5.9: Test F1 Score Distribution Across Ansatz and Encoding

The results indicate that, while most variations in optimisers, learning rates, and encoding strategies do not produce statistically significant differences, the depth of the QCNN architecture has a considerable impact on performance. Specifically, the two-layer QCNN consistently outperforms the one-layer configuration, indicating the role of circuit depth in achieving higher F1 scores.

5.3 Overall Comparison

The results presented in Table 5.1 summarise the mean, median, standard deviation and maximum values for the F1 scores and mean, and standard deviation for accuracy across

both models. The QCNN results displayed here implemented 2 layers and used a learning rate of 0.02. The Wilcoxon signed-rank test revealed a statistically significant difference in performance between the QCNN and QSVM models. The QSVM achieved its best F1 score of **1.0000** with A_{14}, surpassing its benchmark value of **0.8934**. Furthermore, this outperformance of the QSVM model was observed in all ansatz configurations. In contrast, the QCNN achieved its highest F1 score of **0.8666** using A_{19} with angle embedding, which falls below its benchmark value of **0.8745**.

However, a simple comparison between the best F1 scores is often inadequate. To ensure consistency with the statistical analysis performed using the Wilcoxon signed-rank test, a comparison was conducted between the median F1 scores obtained in this study and the benchmark values reported in the literature. Recall that the benchmark values are the best F1 scores and not the median F1 scores. For the QSVM, the median F1 score of **0.8571** using A_{14} and **0.8421** using A_{13} is slightly below the benchmark value of **0.8934**, but showed no statistical difference using a p-value of 0.05. For the QCNN, the highest median F1 score of **0.7651** was achieved with A_6 using angle embedding, remains below the benchmark value of **0.8745**. Specifically, the difference between the QSVM median (**0.8571**) and its benchmark (**0.8934**) is approximately **4.06%**, while the QCNN median (**0.7651**) deviates from its benchmark (**0.8745**) by a more substantial margin of **12.49%**.

The amplitude embedding implementation in the QSVM model, although not utilising a feature map, achieved an F1 mean of **0.8014** with a maximum of **0.9167**, further emphasising the QSVM's robust performance across all configurations. These observations infers the QSVM's superiority to handle amplitude embedding configurations with greater stability and efficiency. Its higher maximum and mean F1 scores suggest that QSVM is better suited to achieve both peak performance and consistent results, making it a more reliable algorithm for IDS applications. Amplitude embedding, while representing all data features simultaneously, has the potential to bring in additional noise as a result of its direct encoding of the entire feature space. This comprehensive representation can amplify the effect of less informative or irrelevant features, which may contribute to the observed performance variability. Angle embedding, through PCA, may inherently filter or reduce noise due to its structured representation of individual features, thereby achieving more stable and higher performance.

These findings suggest that data-driven QML approaches, i.e. the QSVM implementation, may be more suitable than tunable parameter-based approaches. Data-driven methods reduce the reliance on the black-box methodology of highly tunable parameters that risk overfitting. The lack of changes in F1 score when varying hyperparemeter configurations

may contribute to the ineffect of hyperparameter changes against the f1 scores

In terms of the ansatz design, it was discovered that the two varying QML models favoured separate entanglement structures. More specifically, QSVM showed better F1 scores for A_{13} and A_{14}, implying that circuit block entanglement was better suited for that algorithm. On the other hand, QCNN performed best using A_5 and A_6, which corresponds to the all-to-all topology. Furthermore, both algorithms favoured the $R_x(\theta)$ gates over $R_z(\theta)$ as both A_6 and A_{14} scored higher than their respective variants A_5 and A_{13}.

Model	Ansatz	Encoding	F1 Mean	F1 Median	F1 Std	F1 Max	Accuracy Mean	Accuracy Std	Benchmark F1
QSVM	Ansatz13	N/A	0.8570	0.8421	0.0567	0.9565	0.8406	0.0688	0.8934
QSVM	Ansatz14	N/A	**0.8673**	**0.8571**	0.0716	**1.0000**	**0.8563**	0.0793	
QSVM	Ansatz19	N/A	0.8448	0.8377	**0.0439**	0.9167	0.8281	**0.0547**	
QSVM	Ansatz5	N/A	0.8101	0.8000	0.0493	0.9167	0.7875	0.0619	
QSVM	Ansatz6	N/A	0.8290	0.8182	0.0591	0.9565	0.8156	0.0651	
QSVM	Amplitude (raw)	N/A	0.8014	0.8000	0.0730	0.9167	0.7967	0.0694	
QCNN	Ansatz13	AmplitudeEmbedding	0.7475	0.7496	**0.0283**	0.8015	0.7494	**0.0276**	0.8745
QCNN	Ansatz13	AngleEmbedding	0.7430	0.7442	0.0288	0.8009	0.7486	**0.0276**	
QCNN	Ansatz14	AmplitudeEmbedding	0.7308	0.7309	0.0462	0.8015	0.7352	0.0421	
QCNN	Ansatz14	AngleEmbedding	0.7577	0.7545	0.0355	0.8377	0.7661	0.0327	
QCNN	Ansatz19	AmplitudeEmbedding	0.7294	0.7030	0.0497	0.8058	0.7331	0.0454	
QCNN	Ansatz19	AngleEmbedding	0.7405	0.7413	0.0613	**0.8666**	0.7487	0.0580	
QCNN	Ansatz5	AmplitudeEmbedding	0.7428	0.7394	0.0349	0.8056	0.7452	0.0333	
QCNN	Ansatz5	AngleEmbedding	0.7576	0.7632	0.0492	0.8284	0.7650	0.0445	
QCNN	Ansatz6	AmplitudeEmbedding	0.7343	0.7202	0.0368	0.8010	0.7379	0.0349	
QCNN	Ansatz6	AngleEmbedding	**0.7805**	**0.7651**	0.0447	0.8513	**0.7886**	0.0381	

Table 5.1: Comprehensive Statistical Results for QSVM and QCNN with Benchmarks

Chapter 6

Conclusion

As discovered in previous research, the QML algorithms have yielded improvements in traffic classification accuracy, execution time, and the ability to detect day 0 attacks. This research extends the application of the QSVMs and QCNNs algorithms for intrusion detection by using various designs of ansatzes that have high expressivity and entangling capabilities to improve the detection accuracy of IDSs. The experiments were conducted in a noiseless setting to determine their theoretical performance. This research, as a result, demonstrates and advances the limited quantum speed up achieved by the quantum kernels used in each of the QML algorithms, measurable through the increases of F1 scores yielded by the classically intractable feature maps.

When looking at the two QML models, the QSVM outperforms the QCNN which may indicate that quantum kernels work better with data driven parameterised gates rather optimisation driven parameterised gates. Circuit-Block encoding produced the best results for QSVM, yet all-to-all entanglement produced the best result for QCNN, which may indicate that different QML algorithms favour different entangling patterns. However, these are not definitive conclusions, as there would need to be other variables taken into account. A key takeaway from these results is that the best resulting ansatz designs (A_{14} for QSVM and A_6 for QCNN) are also the two most expressive ansatz discovered in previous research, as seen in Table 1.2.

This research has shown the potential of quantum algorithms, even with a minimal number of qubits, to enhance IDSs. The decision to limit the number of qubits was driven by the constraints of classical simulation, allowing for a deeper understanding of the theoretical implications and advantages of quantum algorithms, as well as their contribution to quantum speedup. As inferred using QTMs in Section 3.1, the computational capabilities of quantum algorithms are inherently constrained when simulated classically. This

limitation prompts the question: In which direction should this research be extended?

An avenue for exploration is the impact of incorporating more features into the calculation of quantum fidelity kernels on the performance of the IDS. While additional features can increase the variance captured, they may also contain unnecessary noise. Investigating the effects of using all features as qubits, varying the number of PCA components or using alternative dimensionality reduction methods, such as autoencoders, could provide valuable insights. The preprocessing of data and preparation of quantum states are critical to the success of quantum algorithms. Decisions such as scaling feature values (e.g., between $-\pi$ and π or 0 and 2π) and determining the optimal number of qubits also impact performance.

Relative to the encoding of classical data into the quantum state, the design of the remaining unitary transformations within the quantum circuit is just as important. Three distinct paths emerge: optimising the circuit design, optimising the parameters, or optimising both. Determining the most effective approach is an open question that requires further exploration.

In the first case, genetic algorithms could be used to design ansatz optimised for IDS use cases. This approach demands consideration of several factors, including the recurring question of how data is encoded into the quantum system. Genetic algorithms undergo operations such as selection, mutation and crossover that would need to be applied to the design of a quantum circuit as well as appropriate means of determining the quantum circuits' fitness value. In this research, the design of the ansatzes was based off the expressibility and entanglement capabilities that these circuits yielded, but can be improved through a guided and systematic alteration of the quantum circuit design. This process can be guided through quantum kernel alignment, where the fitness value looks at the proximity of the output kernel to the ideal kernel.

In terms of optimising tunable parameters, classical optimisation methods, such as gradient-based approaches, are commonly used to update parameters in QML models. However, these methods don't account for the sinusoidal nature of the quantum parameter search space. This characteristic often results in experiencing barren plateaus, limiting the effectiveness of classical optimisation techniques. Future research should focus on developing optimisation strategies specifically tailored to the unique properties of quantum systems, such as natural quantum gradient descent.

This research signifies the capabilities of QSVMs for intrusion detection. However, this data-driven approach can be improved by incorporating trainable parameters to align

quantum kernels with idealised kernel matrices. Although promising in theory, this approach of Quantum Kernel Alignment is prone to overfitting. However, the design of quantum circuits still plays a pivotal role in projecting input data into higher-dimensional spaces. As seen in QCNNs, the circuit depth played a role in the experiment outcomes and can be implemented in QSVMs by increasing the amount of entangling blocks used.

This research acknowledges the vital role of data encoding, quantum circuit design, and parameter optimisation in the development of quantum-enhanced IDS. While the theoretical advantages of quantum algorithms are evident, their practical implementation on NISQ devices needs to be observed across various datasets. By extending this research, the implementation of QML can progress toward realising its full potential for IDSs and other applications.

Bibliography

[1] Financial Times. "The Growing Threat of Cybercrime." In: *Financial Times* (2023). Accessed: 2024-09-21. URL: https://www.ft.com/content/8a79ab25-c902-4110-bcb8-be2fd422f6bf.

[2] Maria Schuld and Francesco Petruccione. *Supervised Learning with Quantum Computers*. 1st. Quantum Science and Technology. Springer, 2018. ISBN: 978-3-319-96423-2. DOI: 10.1007/978-3-319-96424-9.

[3] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. 10th Anniversary Edition. Cambridge University Press, 2002. ISBN: 978-1-107-00217-3. DOI: 10.1017/CBO9780511976667.

[4] Timothy Proctor et al. "Ancilla-driven quantum computation for qudits and continuous variables." In: *Physical Review A* 95.5 (2017), p. 052317. DOI: 10.1103/PhysRevA.95.052317.

[5] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer." In: *SIAM Journal on Computing* 26.5 (1997), pp. 1484–1509. DOI: 10.1137/S0097539795293172.

[6] Lov K. Grover. "A Fast Quantum Mechanical Algorithm for Database Search." In: *Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC)*. ACM, 1996, pp. 212–219. DOI: 10.1145/237814.237866.

[7] John Preskill. "Quantum Computing in the NISQ Era and Beyond." In: *Quantum* 2 (2018), p. 79. DOI: 10.22331/q-2018-08-06-79.

[8] Alvaro Ballon. *Superconducting Qubits*. Accessed: 2025-01-05. 2024. URL: https://pennylane.ai/qml/demos/tutorial_sc_qubits.

[9] IBM Quantum. *Processor types*. Accessed: 2024-05-22. 2025. URL: https://docs.quantum.ibm.com/guides/processor-types.

[10] Frank Arute, Kunal Arya, Ryan Babbush, et al. "Quantum supremacy using a programmable superconducting processor." In: *Nature* 574 (2019), pp. 505–510.

[11] Google Quantum AI. *Meet Willow, our state-of-the-art quantum chip*. Accessed: 2025-03-11. 2024. URL: https://research.google/meet-willow-quantum-chip/.

[12] Microsoft Quantum Team. *Microsoft unveils Majorana 1, the world's first quantum processor powered by topological qubits*. Accessed: 2025-03-11. 2025. URL: https://news.microsoft.com/2025/02/19/microsoft-unveils-majorana-1-quantum-processor/.

[13] Alvaro Ballon. *Superconducting Qubits*. Accessed: 2025-01-06. 2024. URL: https://pennylane.ai/qml/demos/tutorial_trapped_ions.

[14] IonQ, Inc. *IonQ Quantum Computing*. Accessed: 2025-01-06. URL: https://ionq.com/.

[15] Honeywell International Inc. *Honeywell Quantum Solutions*. Accessed: 2025-01-06. URL: https://www.honeywell.com/us/en/company/quantum.

[16] Alvaro Ballon. *Superconducting Qubits*. Accessed: 2025-01-06. 2024. URL: https://pennylane.ai/qml/demos/tutorial_photonics.

[17] Lars S Madsen, Frederik Laudenbach, Mohsen F Askarani, et al. "Quantum computational advantage with a programmable photonic processor." In: *Nature* 606 (2022), pp. 75–81.

[18] IBM Research. *On "Quantum Supremacy"*. Accessed: 2024-08-22. 2019. URL: https://www.ibm.com/blogs/research/2019/10/on-quantum-supremacy/.

[19] Han-Sen Zhong, Hui Wang, Yu-Hao Deng, et al. "Quantum computational advantage using photons." In: *Science* 370.6523 (2020), pp. 1460–1463.

[20] Ruizhi Tian, Zhiqiang Liu, Weixiang Fei, et al. "Cybersecurity data science: an overview from machine learning perspective." In: *Journal of Big Data* 7.1 (2020), pp. 1–30. DOI: 10.1186/s40537-020-00318-5.

[21] Maria Schuld and Nathan Killoran. "Quantum machine learning in feature Hilbert spaces." In: *Physical Review Letters* 122.4 (2021), p. 040504.

[22] Vojtěch Havlíček et al. "Supervised learning with quantum-enhanced feature spaces." In: *Nature* 567.7747 (2019), pp. 209–212.

[23] Iris Cong, Soonwon Choi, and Mikhail D Lukin. "Quantum convolutional neural networks." In: *Nature Physics* 15.12 (2019), pp. 1273–1278.

[24] Alán Aspuru-Guzik et al. "Simulated Quantum Computation of Molecular Energies." In: *Science* 309.5741 (2005), pp. 1704–1707. DOI: 10.1126/science.1113479.

[25] Alberto Peruzzo et al. "A variational eigenvalue solver on a photonic quantum processor." In: *Nature Communications* 5.1 (2014), p. 4213. DOI: 10.1038/ncomms5213.

[26] Abhinav Kandala et al. "Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets." In: *Nature* 549.7671 (2017), pp. 242–246. DOI: 10.1038/nature23879.

[27] Yudong Cao et al. "Quantum Chemistry in the Age of Quantum Computing." In: *Chemical Reviews* 119.19 (2019), pp. 10856–10915. DOI: 10.1021/acs.chemrev.8b00803.

[28] Sam McArdle et al. "Quantum computational chemistry." In: *Reviews of Modern Physics* 92.1 (2020), p. 015003. DOI: 10.1103/RevModPhys.92.015003.

[29] Mayra Rivera-Ruiz, Andres Mendez-Vazquez, and Mauricio Lopez Romero. "Time Series Forecasting with Quantum Machine Learning Architectures." In: Advances in Computational Intelligence, Oct. 2022, pp. 66–82. ISBN: 978-3-031-19492-4. DOI: 10.1007/978-3-031-19493-1_6.

[30] D. Balakrishnan et al. "Quantum Neural Network for Time Series Forecasting: Harnessing Quantum Computing's Potential in Predictive Modeling." In: *2023 2nd International Conference on Futuristic Technologies (INCOFT)*. 2023, pp. 1–7. DOI: 10.1109/INCOFT60753.2023.10425465.

[31] Mina Doosti et al. *A Brief Review of Quantum Machine Learning for Financial Services*. July 2024. DOI: 10.48550/arXiv.2407.12618.

[32] Pranjali Shinde, Ramesh Kumar Jena, and Bibhudatta Tripathy. "Quantum machine learning in medical image analysis: A survey." In: *Neurocomputing* 515 (2022), pp. 106–125. DOI: 10.1016/j.neucom.2022.08.084.

[33] Amanpreet Singh, Gurwinder Singh Sethi, and Saleh Albahli. "Quantum Machine Learning Revolution in Healthcare: A Systematic Review of Emerging Perspectives and Applications." In: *Journal of Healthcare Engineering* 2023 (2023), pp. 1–15. DOI: 10.1155/2023/8663982.

[34] Rajiv Mohan et al. "Quantum Computing in Medicine." In: *Journal of Medical Systems* 47.1 (2023), p. 105. DOI: 10.1007/s10916-023-02058-2.

[35] Mahir Mohammed et al. "Oncological Applications of Quantum Machine Learning." In: *Computational and Mathematical Methods in Medicine* 2023 (2023), pp. 1–9. DOI: 10.1155/2023/7277461.

[36] Philip Adebayo, Frederick Basaky, and Edgar Osaghae. "Developing a Model for Predicting Lung Cancer Using Variational Quantum-Classical Algorithm: A Survey." In: *Journal of Applied Artificial Intelligence* 3.1 (June 2022), pp. 47–60. DOI: 10.48185/jaai.v3i1.446.

[37] Rushank Goyal. "Quantum lattices for early cancer detection through machine learning." In: *Cancer Genetics* 268-269 (2022), p. 27. ISSN: 2210-7762. DOI: 10.1016/j.cancergen.2022.10.086.

[38] Tarek Haddadin, Nadin Bouri, and Nada Matta. "Quantum Machine Learning for Digital Health? A Systematic Review." In: *arXiv preprint arXiv:2410.02446* (2023).

[39] Peter Wittek. *Quantum Machine Learning: What Quantum Computing Means to Data Mining*. Academic Press, 2014. ISBN: 9780128009536. DOI: doi.org/10.1016/C2013-0-19170-2.

[40] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. "The quest for a Quantum Neural Network." In: *Quantum Information Processing* 13.11 (2014), pp. 2567–2586. DOI: 10.1007/s11128-014-0809-8.

[41] Troels F. Rønnow et al. "Defining and detecting quantum speedup." In: *Science* 345.6195 (2014), pp. 420–424.

[42] Edward Farhi and Hartmut Neven. "Classification with quantum neural networks on near term processors." In: *arXiv preprint arXiv:1802.06002* (2018). arXiv: 1802.06002.

[43] Nathan Killoran et al. "Continuous-variable quantum neural networks." In: *Physical Review Research* 1.3 (2019), p. 033063. DOI: 10.1103/PhysRevResearch.1.033063.

[44] Otavio Kiyatake Nicesio, Adriano Galindo Leal, and Vagner Luiz Gava. "Quantum Machine Learning for Network Intrusion Detection Systems, a Systematic Literature Review." In: *2023 IEEE 2nd International Conference on AI in Cybersecurity (ICAIC)*. 2023, pp. 1–6. DOI: 10.1109/ICAIC57335.2023.10044125.

[45] Daniel Said et al. "Quantum Computing and Machine Learning for Cybersecurity: Distributed Denial of Service (DDoS) Attack Detection on Smart Micro-Grid." In: *Energies* 16.9 (2023), p. 3572. DOI: 10.3390/en16093572.

[46] Muhammed Yusuf Küçükkara, Furkan Atban, and Cüneyt Bayılmış. "Quantum-Neural Network Model for Platform Independent DDoS Attack Classification in Cyber Security." In: *Advanced Quantum Technologies* 7.10 (2024), p. 2400084. DOI: 10.1002/qute.202400084. URL: https://doi.org/10.1002/qute.202400084.

[47] Tae Hoon Kim and S. Madhavi. "Quantum Intrusion Detection System Using Outlier Analysis." In: *Scientific Reports* 14.1 (2024), p. 27114. ISSN: 2045-2322. DOI: 10.1038/s41598-024-78389-0.

[48] Alon Kukliansky et al. "Network Anomaly Detection Using Quantum Neural Networks on Noisy Quantum Computers." In: *IEEE Transactions on Quantum Engineering* 5 (2024), pp. 1–11.

[49] Diego Medeiros de Abreu, Christian Esteve Rothenberg, and Antonio Abelem. "QML-IDS: Quantum Machine Learning Intrusion Detection System." In: *arXiv preprint* (Oct. 2024). DOI: 10.48550/arXiv.2410.16308.

[50] Sukin Sim, Peter D Johnson, and Alán Aspuru-Guzik. "Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms." In: *Advanced Quantum Technologies* 2.12 (2019), p. 1900070.

[51] John Watrous. *The Theory of Quantum Information*. Cambridge University Press, 2018. ISBN: 978-1-107-07329-6. DOI: 10.1017/9781316848142.

[52] Max Born. "Zur Quantenmechanik der Stoßvorgänge." In: *Zeitschrift für Physik* 37 (1926), pp. 863–867. DOI: 10.1007/BF01397477.

[53] Michael Reed and Barry Simon. *Methods of Modern Mathematical Physics, Vol. 1: Functional Analysis*. New York: Academic Press, 1972, pp. 197–210. ISBN: 978-0125850506.

[54] Vladimir Kecman. "Support Vector Machines – An Introduction." In: *Support Vector Machines: Theory and Applications*. Ed. by Lipo Wang. Studies in Fuzziness and Soft Computing. Berlin, Heidelberg: Springer, 2005, pp. 1–47. DOI: 10.1007/978-3-540-28646-2_1.

[55] Weikun Liu, Yuzhu Li, Houwen Yin, et al. "Quantum multi-state Swap Test: an algorithm for estimating overlaps of arbitrary number quantum states." In: *EPJ Quantum Technology* 11 (2024), p. 46. DOI: 10.1140/epjqt/s40507-024-00259-5.

[56] Taeho Hur, Laramie Kim, and DaeKil Park. "Quantum Convolutional Neural Network for Classical Data Classification." In: *Quantum Machine Intelligence* 4.1 (2022), p. 3. DOI: 10.1007/s42484-021-00061-x.

[57] Maria Schuld and Francesco Petruccione. "Quantum Models as Kernel Methods." In: *Machine Learning with Quantum Computers*. Springer, 2021, pp. 217–245. DOI: 10.1007/978-3-030-83098-4_6.

[58] Jarrod R. McClean et al. "Barren Plateaus in Quantum Neural Network Training Landscapes." In: *Nature Communications* 9.1 (2018), p. 4812. DOI: 10.1038/s41467-018-07090-4.

[59] M. Cerezo et al. "Cost Function Dependent Barren Plateaus in Shallow Parametrized Quantum Circuits." In: *Nature Communications* 12.1 (2021), p. 1791. DOI: 10.1038/s41467-021-21728-w.

[60] Hsin-Yuan Huang et al. "Quantum Advantage in Learning from Experiments." In: *Science* 376.6598 (2022), pp. 1182–1186. DOI: 10.1126/science.abn7293.

[61] Henry Chen et al. "Quantum Convolutional Neural Networks for High Energy Physics Data Analysis." In: *Physical Review Research* 3.L032024 (2021). DOI: 10.1103/PhysRevResearch.3.L032024.

[62] Ameel Valjee. *Quantum Machine Learning for Enhanced Intrusion Detection*. https://github.com/ameelvaljee/Quantum-Machine-Learning-for-Enhanced-Intrusion-Detection. GitHub repository. 2025.

[63] Nour Moustafa and Jill Slay. "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)." In: *2015 Military Communications and Information Systems Conference (MilCIS)*. IEEE. 2015, pp. 1–6. DOI: 10.1109/MilCIS.2015.7348942.

[64] Nour Moustafa and Jill Slay. "The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 dataset and the comparison with the KDD99 dataset." In: *Information Security Journal: A Global Perspective* 25.1-3 (2016), pp. 18–31. DOI: 10.1080/19393555.2015.1125974.

[65] Nour Moustafa, Gavin Creech, and Jill Slay. "Novel geometric area analysis technique for anomaly detection using trapezoidal area estimation on large-scale networks." In: *IEEE Transactions on Big Data* 5.4 (2017), pp. 481–494. DOI: 10.1109/TBDATA.2017.2701360.

[66] Nour Moustafa, Jill Slay, and Gavin Creech. "Big data analytics for intrusion detection system: statistical decision-making using finite dirichlet mixture models." In: *Data Analytics and Decision Support for Cybersecurity*. Springer, 2017, pp. 127–156. DOI: 10.1007/978-3-319-59439-2_7.

[67] Mohanad Sarhan et al. "NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems." In: *Big Data Technologies and Applications: 10th EAI International Conference, BDTA 2020, and 13th EAI International Conference on Wireless Internet, WiCON 2020, Virtual Event, December 11, 2020, Proceedings*. Springer, 2020, pp. 117–131. DOI: 10.1007/978-3-030-67197-6_9.