NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

BSc THESIS

Quantum Neural Networks with Qutrits

Themistoklis N. Valtinos

SUPERVISORS: **Dimitris Syvridis**, Professor
Aikaterini Mandilara, Doctor

ATHENS

JULY 2023

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Κβαντικά Νευρωνικά Δίκτυα με Qutrits

Θεμιστοκλής Ν. Βαλτινός

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Δημήτριος Συβρίδης**, Καθηγητής
Αικατερίνη Μανδηλαρά, Δόκτωρ

ΑΘΗΝΑ

ΙΟΥΛΙΟΣ 2023

BSc THESIS

Quantum Neural Networks with Qutrits

Themistoklis N. Valtinos

S.N.: 1115200600015

SUPERVISORS: **Dimitris Syvridis**, Professor
Aikaterini Mandilara, Doctor

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Κβαντικά Νευρωνικά Δίκτυα με Qutrits

Θεμιστοκλής Ν. Βαλτινός

Α.Μ.: 1115200600015

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Δημήτριος Συβρίδης**, Καθηγητής
Αικατερίνη Μανδηλαρά, Δόκτωρ

ABSTRACT

Quantum computers, leveraging the principles of quantum physics, have the potential to revolutionize various domains by utilizing quantum bits (qubits) that can exist in superpositions and entanglement, allowing for parallel exploration of solutions. Recent advancements in quantum hardware have enabled the realization of high-dimensional quantum states on a chip-scale platform, proposing another potential avenue.

The utilization of qudits, quantum systems with levels exceeding 2, not only offer increased information capacity, but also exhibit improved resilience against noise and errors. Experimental implementations have successfully showcased the potential of high-dimensional quantum systems in efficiently encoding complex quantum circuits, further highlighting their promise for the future of quantum computing.

In this thesis, the potential of qutrits is explored to enhance machine learning tasks in quantum computing. The expanded state space offered by qutrits enables richer data representation, capturing intricate patterns and relationships. To this end, employing the mathematical framework of SU(3), the Gell-Mann feature map is introduced to encode information within an 8-dimensional space. This empowers quantum computing systems to process and represent larger amounts of data within a single qutrit.

The primary focus of this thesis centers on classification tasks utilizing qutrits, where a comparative analysis is conducted between the proposed Gell-Mann feature map, well-established qubit feature maps, and classical machine learning models. Furthermore, optimization techniques within expanded Hilbert spaces are explored, addressing challenges such as vanishing gradients and barren plateaus landscapes.

This work explores foundational concepts and principles in quantum computing and machine learning to ensure a solid understanding of the subject. It also highlights recent advancements in quantum hardware, specifically focusing on qutrit-based systems.

The main objective is to explore the feasibility of the Gell-Mann encoding for multiclass classification in the SU(3) space, demonstrate the viability of expanded Hilbert spaces for machine learning tasks, and establish a robust foundation for working with geometric feature maps.

By delving into the design considerations and experimental setups in detail, this research aims to contribute to the broader understanding of the capabilities and limitations of qutrit-based systems in the context of quantum machine learning, contributing to the advancement of quantum computing and its applications in practical domains.

SUBJECT AREA: Quantum Computing, Machine Learning

KEYWORDS: quantum circuits, machine learning, quantum information, qutrits, classification, supervised learning, neural networks

ΠΕΡΙΛΗΨΗ

Οι κβαντικοί υπολογιστές, εκμεταλλευόμενοι τις αρχές της κβαντικής μηχανικής, έχουν τη δυνατότητα να μεταμορφώσουν πολλούς τεχνολογικούς τομείς, χρησιμοποιώντας κβαντικά bit (qubits) που μπορούν να υπάρχουν σε υπέρθεση και εναγκαλισμό, επιτρέποντας, μεταξύ άλλων δυνατοτήτων, την παράλληλη αναζήτηση λύσεων. Πρόσφατες εξελίξεις στο κβαντικό υλικό επέτρεψαν την υλοποίηση πολυδιάστατων κβαντικών καταστάσεων σε νέες πλατφόρμες μικροκυκλωμάτων, προτείνοντας μια ακόμη ενδιαφέρουσα προσέγγιση.

Η χρήση qudits, κβαντικών συστημάτων με υψηλότερες διάστασεις, προσφέρει αυξημένο χώρο για αναπαράστη πληροφορίας, αλλά επίσης πειραματικές υλοποιήσεις έχουν επιδείξει ανθεκτικότητα έναντι θορύβου και σφαλμάτων. Αυτό επισημαίνει περαιτέρω την θέση τους στο μέλλον του κβαντικού υπολογισμού.

Σε αυτήν τη πτυχιακή, εξετάζεται η δυνατότητα των qutrits για την επίλυση προβλημάτων μηχανικής μάθησης σε κβαντικό υπολογιστή. Ο επεκταμένος χώρος καταστάσεων που προσφέρουν τα qutrits επιτρέπει πλουσιότερη αναπαράσταση δεδομένων. Για το σκοπό αυτό, χρησιμοποιώντας το μαθηματικό πλαίσιο του SU(3), εισάγεται η χρήση των πινάκων Gell-Mann για την κωδικοποίηση σε έναν 8-διάστατο χώρο. Αυτό εξοπλίζει τα συστήματα κβαντικού υπολογισμού με τη δυνατότητα επεξεργασίας και αναπαράστασης περισσότερων δεδομένων σε ένα μόνο qutrit.

Η έρευνα επικεντρώνεται σε προβλήματα ταξινόμησης χρησιμοποιώντας qutrits, όπου διεξάγεται μια συγκριτική ανάλυση μεταξύ του προτεινόμενου χάρτη χαρακτηριστικών Gell-Mann, κυκλώματων που χρησιμοποιούν qubits και μοντέλων κλασσικής μηχανικής μάθησης. Επιπλέον, εξερευνούνται τεχνικές βελτιστοποίησης σε χώρους Hilbert υψηλών διαστάσεων, με σκοπό την αντιμετώπιση προκλήσεων, όπως τα vanishing gradients και το πρόβλημα των barren plateaus. Τέλος, καλύπτονται πρόσφατες εξελίξεις στον κβαντικό υλικό, με ειδική έμφαση σε συστήματα βασισμένα σε qutrits.

Ο κύριος στόχος αυτής της πτυχιακής εργασίας είναι να εξετάσει τη δυνατότητα κωδικοποίησης Gell-Mann για προβλήματα ταξινόμησης, να αποδείξει την εφικτότητα της επέκτασης των χώρων Hilbert για εργασίες μηχανικής μάθησης και να ορίσει μια αξιόπιστη βάση για εργασία με γεωμετρικούς χάρτες χαρακτηριστικών.

Αναλύωντας τις σχεδιαστικές επιλογές και πειραματικές διατάξεις λεπτομερώς, αυτή η έρευνα στοχεύει να συμβάλει στην ευρύτερη κατανόηση των δυνατοτήτων και των περιορισμών των συστημάτων με qutrits στο πλαίσιο της κβαντικής μηχανικής μάθησης, συνεισφέροντας στην πρόοδο του κβαντικού υπολογισμού και των εφαρμογών του σε πρακτικούς τομείς.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Κβαντική Πληροφορική, Μηχανική Μάθηση

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: κβαντικά κυκλώματα, μηχανική μάθηση, κβαντική πληροφορία, qutrits, κατηγοριοποίηση, επιβλεπόμενη μάθηση, νευρωνικά δίκτυα

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Overview

Throughout the early 20th century, physicists embarked on a quest to unravel the fundamental rules governing the universe. They encountered phenomena that defied the explanations provided by existing physics, indicating that the prevailing rules were not entirely accurate. This prompted the development of a more comprehensive framework known as ”quantum” physics, which remarkably accounted for these peculiar behaviors.

Exploiting the principles of quantum physics, quantum computers have the potential of a paradigm shift in various fields, from cryptography and optimization to materials science. Traditional computers operate on classical bits, representing information as either 0s or 1s, quantum computers utilize quantum bits, or qubits, which can exist in superpositions of both 0 and 1 simultaneously. In addition, qubits can exhibit the phenomenon of entanglement, where the states of multiple qubits become intrinsically linked. These unique characteristics empower quantum computers to explore multiple solutions in parallel, potentially providing exponential speedups for specific problems.

The future of quantum computing holds great promise, with the added prospect of transitioning from qubits to qudits, with levels exceeding 2 (d > 2), as the fundamental units of information processing. Recent breakthroughs have demonstrated the power of high-dimensional quantum systems. Scientists have successfully created a microchip capable of generating two entangled qudits, each with 10 states, resulting in a total of 100 dimensions—surpassing what could be achieved by six entangled qubits [1]. This achievement underscores the potential of high-dimensional quantum systems to offer increased information capacity and improved resilience against noise and errors compared to conventional qubit-based systems.

This experimental implementation has been realized using integrated photonics and this in addition to other approaches including qutrits, has been a subject of ongoing research. For example in the paper ”Quantum Information Scrambling on a Superconducting Qutrit Processor,” the authors explore the dynamics of quantum information in strongly interacting systems using qutrits (three-level quantum systems) instead of the conventional two-level qubits [5]. This work demonstrates the potential of higher-dimensional quantum systems like qutrits in achieving resource-efficient encoding of complex quantum circuits, serving as a proof of principle for using qutrit-based quantum processors and paves the way for building more advanced quantum information processors.

The expanded state space offered by qudits, compared to qubits, holds significant potential for enhancing machine learning tasks in a quantum computer. By increasing the number of distinct states that can be represented, qudits enable a higher-dimensional encoding of information, allowing for the representation of more complex patterns and relationships within the data. With qudits, the increased granularity of information encoding enables a quantum machine learning algorithm to capture and process finer details,

potentially leading to more accurate and nuanced models.

Furthermore, the expanded state space of qudits enhances the capacity for parallel processing. Quantum algorithms can operate on multiple states simultaneously, leveraging superposition, entanglement, and coherent manipulations. With qudits, this parallelism can be further amplified, potentially accelerating computations and enabling the exploration of a broader solution space.

Moreover, the increased state space of qudits can mitigate the impact of noise and errors during quantum computations. Error correction techniques, such as quantum error correction codes, rely on redundancy within the state space to detect and rectify errors. With a larger state space, qudits offer more room for implementing robust error correction protocols, enhancing the reliability and stability of quantum machine learning algorithms.

Quantum machine learning is an evolving field that seeks to merge the principles of quantum mechanics with machine learning. While the majority of research in this domain has predominantly concentrated on qubits, which are the prevalent quantum hardware available at present, this particular work focuses on leveraging the potential of qutrits and puts forth a modular architecture that bears resemblance to a neural network.

In the pursuit of enhancing encoding capabilities, the Gell-Mann feature map is introduced, which draws upon the mathematical framework provided by the special unitary group SU(3). The utilization of the proposed Gell-Mann feature map enables the encoding of information within an 8-dimensional space. This empowers the quantum system to capture and process significantly larger amounts of data even within a single qutrit and introduces a novel avenue for exploration within the realm of quantum machine learning.

1.2 Structure

The structure of this thesis is specifically designed to ensure clarity and accessibility for readers who may not possess prior familiarity with quantum computing. Acknowledging that the subject matter may initially appear intimidating, it is important to emphasize that only a modest level of prerequisite knowledge is required. By commencing with the underlying mathematical foundations of quantum information those who are new to quantum computing can effectively engage with the subsequent chapters.

Thus, the initial three chapters provide essential background knowledge. Chapter two delves into the theoretical aspects of quantum computing, while chapter three covers the fundamentals of machine learning. The fourth chapter serves as a bridge between the two, focusing on parameterised quantum circuits, with emphasis on the techniques used in the proposed implementation. Readers who are already well-versed in these subjects may opt to omit these sections.

The fifth chapter delves into a comprehensive analysis of the proposed feature map and quantum neural network architecture, offering code examples, benchmarks and insights into the obtained results. Following the implementation chapter, the sixth chapter provides

conclusions drawn from the study and outlines potential avenues for future research and development.

Finally, the Quantum Hardware Appendix presents an extensive collection of papers and studies exploring high-dimensional quantum hardware, with a specific focus on qutrit-based systems. These resources offer valuable insights into the capabilities and applications of qutrits, highlighting advancements and progress in high-dimensional quantum systems research.

1.3 Objective

The main objective of this thesis is to thoroughly investigate the feasibility of utilizing the Gell-Mann encoding for multiclass classification within the SU(3) space. With quantum hardware employing qutrits on the horizon, this thesis seeks to exhibit the practicality of utilizing this expanded dimensionality for information encoding and optimization in machine learning tasks.

A comprehensive comparison will be presented, evaluating the performance of the proposed Gell-Mann feature map against existing qubit feature maps that have shown promising results, as well as classical machine learning models. Given the nascent nature of quantum machine learning, it is expected that achieving superior results to classical methods is often infeasible, and any claims suggesting otherwise should be approached with skepticism.

The thesis will explore the methodology of developing a quantum machine learning model, particularly focusing on the establishment of a flexible and modular framework for the feature map that can be easily adapted to different datasets. Emphasis will be placed on the subsequent analysis of results, optimization, and fine-tuning processes, with particular attention given to addressing the challenges posed by vanishing gradients in an increased Hilbert space. The thesis aims to explore techniques to combat this issue, building upon existing advancements in qubit-based systems.

Ultimately, this research aims to explore the feasibility of the Gell-Mann encoding for classification tasks and investigate the information capacity of these models. By delving into the design considerations and experimental setups, it seeks to contribute to the broader understanding of the capabilities and limitations of qutrit-based systems in the context of quantum machine learning, contributing to the advancement of quantum computing and its applications in practical domains.

2. QUANTUM INFORMATION

2.1 Mathematical Foundations

In order to work with quantum algorithms and systems, it is essential to have a solid foundation in linear algebra, complex analysis and functional analysis. This first section of this chapter serves as an introduction to the algebraic aspects of quantum computing, focusing on vectors, linear transformations, matrix algebra, Hilbert spaces, group theory and Lie algebra. Experienced readers in these subjects may choose to omit these sections.

2.1.1 Vector Space

A vector space over a field F is a non-empty set V together with two binary operations that satisfy the eight axioms listed below. In this context, the elements of V are commonly called vectors, and the elements of F are called scalars [6].

The first operation, called vector addition or simply addition, assigns to any two vectors $\boldsymbol{v}$ and $\boldsymbol{w}$ in V a third vector in V which is commonly written as $\boldsymbol{v} + \boldsymbol{w}$, and called the sum of these two vectors.

The second operation, called scalar multiplication, assigns to any scalar a in F and any vector $\boldsymbol{v}$ in V another vector in V, which is denoted as $a\boldsymbol{v}$.

For V to be a vector space, the following eight axioms must be satisfied for every $\boldsymbol{u}$, $\boldsymbol{v}$, and $\boldsymbol{w}$ in V, and a and b in F:

1. $(\boldsymbol{u} + \boldsymbol{v}) + \boldsymbol{w} = \boldsymbol{u} + (\boldsymbol{v} + \boldsymbol{w})$ (Associativity of addition)
2. $\boldsymbol{u} + \boldsymbol{v} = \boldsymbol{v} + \boldsymbol{u}$ (Commutativity of addition)
3. $\exists \mathbf{0} \in V$ such that $\mathbf{0} + \boldsymbol{v} = \boldsymbol{v}$ (Existence of additive identity)
4. $\forall \boldsymbol{v} \in V, \exists (-\boldsymbol{v}) \in V$ such that $\boldsymbol{v} + (-\boldsymbol{v}) = \mathbf{0}$ (Existence of additive inverse)
5. $a \cdot (\boldsymbol{u} + \boldsymbol{v}) = a \cdot \boldsymbol{u} + a \cdot \boldsymbol{v}$ (Distributivity of scalar multiplication over vector addition)
6. $(a + b) \cdot \boldsymbol{v} = a \cdot \boldsymbol{v} + b \cdot \boldsymbol{v}$ (Distributivity of scalar multiplication over scalar addition)
7. $(ab) \cdot \boldsymbol{v} = a \cdot (b \cdot \boldsymbol{v})$ (Compatibility of scalar multiplication)
8. $1 \cdot \boldsymbol{v} = \boldsymbol{v}$ (Identity element of scalar multiplication)

2.1.2 Inner and Outer Products

An inner product is a function that takes two vectors from a vector space and returns a scalar, which is often used to measure similarity between vectors or to define geometric concepts such as the length of a vector and the angle between two vectors. Inner products

generalize the dot product and must satisfy specific properties such as linearity, positive definiteness, and conjugate symmetry [7].

Formally, for a vector space V over a field F, an inner product is a function $\langle \cdot, \cdot \rangle : V \times V \to F$ that satisfies the following properties for all vectors $\boldsymbol{u}$, $\boldsymbol{v}$, $\boldsymbol{w} \in V$ and scalars $a \in F$:

1. Linearity in the first argument:

$$\langle a\boldsymbol{u}, \boldsymbol{v} \rangle = a\langle \boldsymbol{u}, \boldsymbol{v} \rangle$$

$$\langle \boldsymbol{u} + \boldsymbol{w}, \boldsymbol{v} \rangle = \langle \boldsymbol{u}, \boldsymbol{v} \rangle + \langle \boldsymbol{w}, \boldsymbol{v} \rangle$$

2. Conjugate symmetry:

$$\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \overline{\langle \boldsymbol{v}, \boldsymbol{u} \rangle}$$

3. Positive definiteness:

$$\langle \boldsymbol{u}, \boldsymbol{u} \rangle > 0 \quad \text{for } \boldsymbol{u} \neq \boldsymbol{0}$$

$$\langle \boldsymbol{0}, \boldsymbol{0} \rangle = 0$$

In the context of inner products, the length (norm) of a vector $\boldsymbol{v}$ is defined as:

$$\|\boldsymbol{v}\| = \sqrt{\langle \boldsymbol{v}, \boldsymbol{v} \rangle}$$

The angle θ between two nonzero vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ can be computed using the inner product as follows:

$$\cos(\theta) = \frac{\langle \boldsymbol{u}, \boldsymbol{v} \rangle}{\|\boldsymbol{u}\|\|\boldsymbol{v}\|}$$

These properties ensure that the inner product provides a meaningful notion of similarity and geometric properties within the vector space [7].

The outer product, denoted as $\boldsymbol{u} \otimes \boldsymbol{v}$, is a mathematical operation that takes two vectors, $\boldsymbol{u}$ and $\boldsymbol{v}$, and produces a matrix. Mathematically, the outer product is defined as:

$$\boldsymbol{u} \otimes \boldsymbol{v} = \boldsymbol{u} \cdot \boldsymbol{v}^T$$

where $\cdot$ represents the dot product and $\boldsymbol{v}^T$ denotes the transpose of vector $\boldsymbol{v}$ [8].

A simple example to illustrate the concept:

Let $\boldsymbol{u} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$ and $\boldsymbol{v} = \begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}$. The outer product of $\boldsymbol{u}$ and $\boldsymbol{v}$ is calculated as:

$$\boldsymbol{u} \otimes \boldsymbol{v} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \cdot \begin{bmatrix} 4 & 5 & 6 \end{bmatrix} = \begin{bmatrix} 4 & 5 & 6 \\ 8 & 10 & 12 \\ 12 & 15 & 18 \end{bmatrix}$$

2.1.3 Matrix Algebra

Matrices are fundamental in representing and manipulating linear transformations and systems of linear equations. They are an indispensable tool for modeling real-world phenomena, allowing for the formulation and solution of intricate mathematical systems. Several basic operations can be applied to modify matrices, including matrix addition, scalar multiplication, matrix multiplication, transposition, and inverse [9].

Matrix Addition

Matrix addition is performed by adding corresponding elements of two matrices of the same size. Given matrices $\boldsymbol{A} = [a_{ij}]$ and $\boldsymbol{B} = [b_{ij}]$ of size $m \times n$, the sum $\boldsymbol{C} = \boldsymbol{A} + \boldsymbol{B}$ is a matrix of the same size, where each element c_{ij} is obtained by adding a_{ij} and b_{ij}:

$$\boldsymbol{C} = \boldsymbol{A} + \boldsymbol{B} = [a_{ij} + b_{ij}]$$

Matrix Scalar Multiplication

Matrix scalar multiplication is performed by multiplying each element of a matrix by a scalar value. Given a matrix $\boldsymbol{A} = [a_{ij}]$ and a scalar α, the scalar product $\alpha\boldsymbol{A}$ is a matrix of the same size, where each element c_{ij} is obtained by multiplying α and a_{ij}:

$$\alpha\boldsymbol{A} = [\alpha a_{ij}]$$

Matrix Multiplication

Matrix multiplication is a binary operation that combines two matrices to produce a third matrix. Given matrices $\boldsymbol{A} = [a_{ij}]$ of size $m \times n$ and $\boldsymbol{B} = [b_{ij}]$ of size $n \times p$, the product $\boldsymbol{C} = \boldsymbol{AB}$ is a matrix of size $m \times p$, where each element c_{ij} is obtained by taking the dot product of the i-th row of $\boldsymbol{A}$ and the j-th column of $\boldsymbol{B}$:

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \ldots + a_{in}b_{nj}$$

It is important to note that matrix multiplication is not commutative, meaning that $\boldsymbol{AB}$ may not be equal to $\boldsymbol{BA}$.

Matrix Transpose

The transpose of a matrix $\boldsymbol{A}$, denoted as $\boldsymbol{A}^\top$, is a new matrix obtained by interchanging its rows and columns. If $\boldsymbol{A}$ has dimensions $m \times n$, then the transpose $\boldsymbol{A}^\top$ has dimensions $n \times m$, and its elements are defined such that $[\boldsymbol{A}^\top]_{ij} = [\boldsymbol{A}]_{ji}$.

Matrix Inverse

The inverse of a square matrix $\boldsymbol{A}$, denoted as $\boldsymbol{A}^{-1}$, is a matrix such that $\boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix. Finding the inverse of a matrix involves solving a system of linear equations or using other techniques such as Gaussian elimination or the adjugate matrix method [9].

2.1.4 Eigenvalues and Eigenvectors

In quantum computing, eigenvalues and eigenvectors play a significant role for representing the possible states of the system, for determining measurement probabilities and forming the basis for the decomposition of the unitary operations.

An eigenvector of a square matrix $\boldsymbol{A}$ is a non-zero vector $\boldsymbol{v}$ such that when $\boldsymbol{A}$ is applied to $\boldsymbol{v}$, the resulting vector is parallel to $\boldsymbol{v}$. The resulting vector is represented as $\lambda\boldsymbol{v}$, where λ is the eigenvalue associated with the eigenvector $\boldsymbol{v}$.

$$\boldsymbol{A}\boldsymbol{v} = \lambda\boldsymbol{v}$$

where $\boldsymbol{A}$ is the matrix, $\boldsymbol{v}$ is the eigenvector, and λ is the eigenvalue [10].

Eigenvalues and eigenvectors have the following properties:

1. Eigenvectors are non-zero vectors.
2. Eigenvalues can be real or complex numbers depending on the matrix.
3. A matrix can have repeated eigenvalues, and each eigenvalue may correspond to multiple linearly independent eigenvectors.
4. The set of eigenvectors corresponding to distinct eigenvalues is linearly independent.
5. The sum of eigenvalues equals the trace of the matrix, and the product of eigenvalues equals the determinant of the matrix.

2.1.5 Hilbert Space

Hilbert spaces, named after the German mathematician David Hilbert, are a class of mathematical structures used in functional analysis and quantum mechanics. They provide a powerful framework for analyzing linear operators, function spaces, and various types of convergence. The most familiar example of a Hilbert space is the Euclidean vector space consisting of three-dimensional vectors, denoted by $\mathbb{R}^3$, and equipped with the dot product.

A Hilbert space is a real or complex inner product space that is also a complete metric space with respect to the distance function induced by the inner product [11]. This means that there is an inner product $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ associating a complex number to each pair of elements $\boldsymbol{x}, \boldsymbol{y}$ of $\boldsymbol{H}$ that satisfies the following properties:

1. The inner product is conjugate symmetric; that is, the inner product of a pair of elements is equal to the complex conjugate of the inner product of the swapped elements:
   $$\langle \boldsymbol{y}, \boldsymbol{x} \rangle = \overline{\langle \boldsymbol{x}, \boldsymbol{y} \rangle}.$$
   Importantly, this implies that $\langle \boldsymbol{x}, \boldsymbol{x} \rangle$ is a real number.

2. The inner product is linear in its first argument. For all complex numbers a and b:

$$\langle a\boldsymbol{x}_1 + b\boldsymbol{x}_2, \boldsymbol{y}\rangle = a\langle \boldsymbol{x}_1, \boldsymbol{y}\rangle + b\langle \boldsymbol{x}_2, \boldsymbol{y}\rangle.$$

3. The inner product of an element with itself is positive definite:

$$\langle \boldsymbol{x}, \boldsymbol{x}\rangle > 0 \quad \text{if } \boldsymbol{x} \neq \mathbf{0}, \quad \langle \boldsymbol{x}, \boldsymbol{x}\rangle = 0 \quad \text{if } \boldsymbol{x} = \mathbf{0}.$$

It follows from properties 1 and 2 that a complex inner product is antilinear, also called conjugate linear, in its second argument, meaning that:

$$\langle \boldsymbol{x}, a\boldsymbol{y}_1 + b\boldsymbol{y}_2\rangle = \bar{a}\langle \boldsymbol{x}, \boldsymbol{y}_1\rangle + \bar{b}\langle \boldsymbol{x}, \boldsymbol{y}_2\rangle.$$

The norm is the real-valued function:

$$\|\boldsymbol{x}\| = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x}\rangle},$$

and the distance d between two points $\boldsymbol{x}, \boldsymbol{y}$ in $\boldsymbol{H}$ is defined in terms of the norm by:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \|\boldsymbol{x} - \boldsymbol{y}\| = \sqrt{\langle \boldsymbol{x} - \boldsymbol{y}, \boldsymbol{x} - \boldsymbol{y}\rangle}.$$

That this function is a distance function means firstly that it is symmetric in $\boldsymbol{x}$ and $\boldsymbol{y}$, secondly that the distance between $\boldsymbol{x}$ and itself is zero, and otherwise the distance between $\boldsymbol{x}$ and $\boldsymbol{y}$ must be positive, and lastly that the triangle inequality holds:

$$d(\boldsymbol{x}, \boldsymbol{z}) \leq d(\boldsymbol{x}, \boldsymbol{y}) + d(\boldsymbol{y}, \boldsymbol{z}).$$

As a complete normed space, Hilbert spaces are by definition also Banach spaces. As such, they are topological vector spaces, in which topological notions like the openness and closedness of subsets are well-defined. Of special importance is the notion of a closed linear subspace of a Hilbert space that, with the inner product induced by restriction, is also complete and therefore a Hilbert space in its own right [11].

2.1.6 Group Theory

Group theory is a branch of mathematics that focuses on the study of an algebraic structure known as a group. A group consists of a set of elements along with a binary operation that combines any two elements from the set to produce a third element. To be considered a group, this binary operation must satisfy specific properties, including closure, associativity, identity, and inverse [12].

Within group theory, researchers delve into various properties and structures of groups, including subgroup, group homomorphism, isomorphism, and group actions. It explores the symmetries and transformations that can occur within mathematical structures and has applications in many areas, including algebra, geometry, physics, and cryptography.

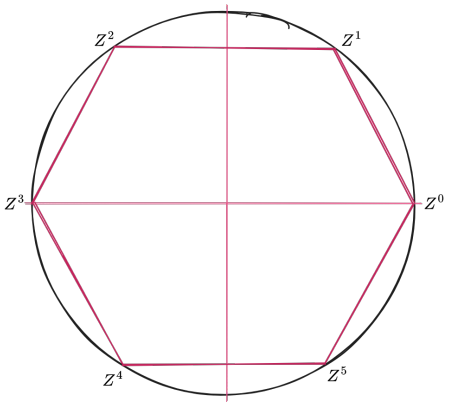


Figure 2.1: Cyclic group with 6 elements

Definition 2.1.1. *A* group *is a set G together with a binary operation $*$ that satisfies the following properties:*

1. ***Closure:*** *For all $a, b \in G$, $a * b \in G$.*
2. ***Associativity:*** *For all $a, b, c \in G$, $(a * b) * c = a * (b * c)$.*
3. ***Identity element:*** *There exists an element $e \in G$ such that for all $a \in G$, $a * e = e * a = a$.*
4. ***Inverse element:*** *For every element $a \in G$, there exists an element $a^{-1} \in G$ such that $a * a^{-1} = a^{-1} * a = e$.*

Example of a Group: The set of integers $\mathbb{Z}$ under addition.

Definition 2.1.2. *Let G be a group. A non-empty subset H of G is called a* subgroup *of G if H is itself a group under the operation inherited from G [13].*

Example of a Subgroup:

Consider the group of integers under addition, $\mathbb{Z}$. We can define a subgroup by taking the set of even integers, denoted by $2\mathbb{Z}$, which consists of all multiples of 2.

To show that $2\mathbb{Z}$ is a subgroup of $\mathbb{Z}$, we need to verify the following:

1. $2\mathbb{Z}$ is non-empty.
2. For any $a, b \in 2\mathbb{Z}$, the sum $a + b$ is also in $2\mathbb{Z}$.
3. For any $a \in 2\mathbb{Z}$, the inverse $-a$ is also in $2\mathbb{Z}$.

Indeed, $2\mathbb{Z}$ satisfies these conditions. It contains the element 0, which makes it non-empty. For any two even integers a and b, their sum $a+b$ is also even, as the sum of two multiples of 2 is still a multiple of 2. Finally, for any even integer a, its negation $-a$ is also even.

Therefore, $2\mathbb{Z}$ forms a subgroup of $\mathbb{Z}$ under addition.

2.1.7 Lie Algebra

In quantum computing, quantum gates are the fundamental building blocks of quantum circuits. These gates are represented by unitary matrices, which are elements of Lie groups. The Lie algebra associated with a Lie group provides a mathematical framework for understanding and analyzing the generators of these unitary transformations.

Lie algebras are vector spaces equipped with a Lie bracket operation that satisfies the Jacobi identity. They are closely related to Lie groups and have applications in various areas of mathematics and physics, such as quantum mechanics and particle physics.

Lie groups are groups that are also smooth manifolds, and any Lie group gives rise to a Lie algebra, which is its tangent space at the identity. Conversely, to any finite-dimensional Lie algebra over real or complex numbers, there is a corresponding connected Lie group unique up to finite coverings (Lie's third theorem) [14].

A Lie algebra is a vector space $\mathfrak{g}$ over some field F together with a binary operation $[\cdot,\cdot] : \mathfrak{g} \times \mathfrak{g} \to \mathfrak{g}$ called the Lie bracket satisfying the following axioms:

Bilinearity:

$$[a\boldsymbol{x} + b\boldsymbol{y}, \boldsymbol{z}] = a[\boldsymbol{x}, \boldsymbol{z}] + b[\boldsymbol{y}, \boldsymbol{z}]$$
$$[\boldsymbol{z}, a\boldsymbol{x} + b\boldsymbol{y}] = a[\boldsymbol{z}, \boldsymbol{x}] + b[\boldsymbol{z}, \boldsymbol{y}]$$

for all scalars a, b in F and all elements $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$ in $\mathfrak{g}$.

Alternativity:

$$[\boldsymbol{x}, \boldsymbol{x}] = 0 \quad \text{for all } \boldsymbol{x} \in \mathfrak{g}.$$

The Jacobi identity:

$$[\boldsymbol{x}, [\boldsymbol{y}, \boldsymbol{z}]] + [\boldsymbol{y}, [\boldsymbol{z}, \boldsymbol{x}]] + [\boldsymbol{z}, [\boldsymbol{x}, \boldsymbol{y}]] = 0 \quad \text{for all } \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathfrak{g}.$$

It is customary to denote a Lie algebra by a lowercase fraktur letter such as $\mathfrak{g}, \mathfrak{h}, \mathfrak{b}, \mathfrak{n}$. If a Lie algebra is associated with a Lie group, then the algebra is denoted by the fraktur version of the group: for example, the Lie algebra of SU(n) is $\mathfrak{su}(n)$ [15].

Generators and Dimension
Elements of a Lie algebra $\mathfrak{g}$ are said to generate it if the smallest subalgebra containing these elements is $\mathfrak{g}$ itself. The dimension of a Lie algebra is its dimension as a vector space over F. The cardinality of a minimal generating set of a Lie algebra is always less than or equal to its dimension.

Lie algebras exhibit rich mathematical structures and classifications. The study of semisimple and complex Lie algebras, in particular, has led to deep connections with diverse areas of mathematics, such as algebraic geometry, combinatorics, and topology. The development of techniques to analyze and classify Lie algebras has been a significant focus of research in algebraic and geometric representation theory.

2.1.8 Special Unitary Groups

The special unitary group SU(n) plays a crucial role in quantum mechanics and quantum information theory. In mathematics, the special unitary group, denoted as SU(n), is a group of $n \times n$ unitary matrices with determinant 1.

A unitary matrix $\boldsymbol{U}$ is a complex square matrix that satisfies the condition $\boldsymbol{U}\boldsymbol{U}^\dagger = \boldsymbol{U}^\dagger\boldsymbol{U} = \mathbb{I}$, where $\boldsymbol{U}^\dagger$ represents the conjugate transpose of $\boldsymbol{U}$, and $\mathbb{I}$ is the identity matrix. Unitary matrices preserve the inner product and norm of vectors.

The special unitary group consists of unitary matrices with determinant 1. This condition restricts the matrices to have a unit modulus determinant, ensuring that they have no scaling effect on quantum states.

The special unitary group matrices are orthogonal under the unitary transformation. They form a group under matrix multiplication, where the product of two unitary matrices is another unitary matrix [16].

The special unitary group is a compact Lie group of dimension $n^2 - 1$. It is a manifold with nontrivial topology, making it a subject of study in the field of algebraic topology.

2.1.9 The SU(2) Group

The SU(2) group is of great significance in the study of quantum systems, particularly in the context of quantum spin, angular momentum, and quantum entanglement. It forms the basis for the construction of quantum gates and the development of quantum algorithms. The Pauli matrices are the fundamental elements of the SU(2) group:

$$\boldsymbol{\sigma_1} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad \boldsymbol{\sigma_2} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad \boldsymbol{\sigma_3} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

The SU(2) group consists of 2×2 unitary matrices, which are complex matrices satisfying the condition $\boldsymbol{U}\boldsymbol{U}^\dagger = \boldsymbol{U}^\dagger\boldsymbol{U} = \mathbb{I}$, where $\boldsymbol{U}^\dagger$ represents the conjugate transpose of $\boldsymbol{U}$ and $\mathbb{I}$ is the 2×2 identity matrix. Unitary matrices in SU(2) preserve the inner product and norm of vectors, and have a determinant of 1.

The SU(2) group is closely related to the special orthogonal group SO(3). In fact, SU(2) is a double cover of SO(3), meaning that every element in SO(3) has two corresponding elements in SU(2) [16].

The SU(2) group provides the mathematical framework for spin representations. Spin is a fundamental property of quantum particles, and the SU(2) matrices act as the rotation operators for spin states.

2.1.10 The SU(3) Group

The SU(3) group forms the basis for the construction of quantum gates and the development of quantum algorithms for qutrit-based quantum computing. The Gell-Mann matrices are a set of eight linearly independent 3×3 traceless Hermitian matrices that span the Lie algebra of the SU(3) group (as the Pauli matrices for SU(2)):

$$\boldsymbol{\lambda}_1 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad \boldsymbol{\lambda}_2 = \begin{pmatrix} 0 & -i & 0 \\ i & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\boldsymbol{\lambda}_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix} \qquad \boldsymbol{\lambda}_4 = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

$$\boldsymbol{\lambda}_5 = \begin{pmatrix} 0 & 0 & -i \\ 0 & 0 & 0 \\ i & 0 & 0 \end{pmatrix} \qquad \boldsymbol{\lambda}_6 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

$$\boldsymbol{\lambda}_7 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -i \\ 0 & i & 0 \end{pmatrix} \qquad \boldsymbol{\lambda}_8 = \frac{1}{\sqrt{3}} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -2 \end{pmatrix}$$

The SU(3) group is a special unitary group of 3×3 complex matrices with unit determinant. It consists of all 3×3 unitary matrices with determinant 1, $\det(\boldsymbol{U}) = 1$.

Matrices in SU(3) are unitary, which means they satisfy the condition:

$$\boldsymbol{U}\boldsymbol{U}^\dagger = \boldsymbol{U}^\dagger\boldsymbol{U} = \mathbb{I},$$

where $\boldsymbol{U}$ is a 3×3 matrix and $\boldsymbol{U}^\dagger$ denotes the conjugate transpose of $\boldsymbol{U}$.

The Gell-Mann matrices satisfy the orthogonality condition:

$$\mathrm{Tr}(\boldsymbol{\lambda}_i\boldsymbol{\lambda}_j) = 2\delta_{ij},$$

where δ_{ij} is the Kronecker delta.

The Gell-Mann matrices can be used as generators to construct elements of the SU(3) group. Any matrix $\boldsymbol{U}$ in SU(3) can be written as:

$$\boldsymbol{U} = e^{i\theta^a \boldsymbol{\lambda}_a},$$

where θ^a are real parameters and $\boldsymbol{\lambda}_a$ are the Gell-Mann matrices [17].

2.2 Quantum Mechanics

2.2.1 Dirac Notation

In quantum mechanics, Dirac notation, or bra-ket notation, is a mathematical notation developed by physicist Paul Dirac to describe quantum states and operators [18]. It provides a powerful and concise framework for representing and manipulating quantum states and operators, making it a fundamental tool, particularly in the field of quantum information.

Kets and Bras

A ket vector, denoted as $|\psi\rangle$, represents a quantum state in a Hilbert space. Kets are column vectors and can be expressed in a chosen basis. For example, in the position basis, the ket vector $|\psi\rangle$ corresponds to the state of a particle at a particular position.

The bra vector, denoted as $\langle\psi|$ is a row vector obtained by taking the complex conjugate transpose of the corresponding ket vector.

Inner Product

The inner product of two ket vectors $|\psi\rangle$ and $|\phi\rangle$ is denoted as $\langle\psi|\phi\rangle$. It represents the complex scalar obtained by taking the conjugate transpose of $|\psi\rangle$ and multiplying it with $|\phi\rangle$. The inner product is used to measure the similarity between quantum states.

Outer Product

The outer product of two ket vectors $|\psi\rangle$ and $|\phi\rangle$ is denoted as $|\psi\rangle\langle\phi|$. It represents the linear operator that maps the vector $|\phi\rangle$ to the vector $|\psi\rangle$, effectively transforming states.

Operators

Quantum operators, such as observables and transformations, are represented by linear operators. In Dirac notation, an operator is represented by a matrix, and its action on a ket vector $|\psi\rangle$ is given by $\boldsymbol{A}|\psi\rangle$, where $\boldsymbol{A}$ is the matrix representation of the operator.

Projection Operators

Projection operators are a special type of operator that project a quantum state onto a specific subspace. They are represented by outer products of a ket vector with itself, such as $|\psi\rangle\langle\psi|$.

Measuring a qubit in the computational basis can be represented by projection operators. For example, the projection operator onto the $|0\rangle$ state is given by:

$$\boldsymbol{P_0} = |\mathbf{0}\rangle\langle\mathbf{0}| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$$

Normalization

Ket vectors are normalized to have a unit length. A normalized ket vector $|\psi\rangle$ satisfies $\langle\psi|\psi\rangle = 1$. Normalization ensures that probabilities calculated using the inner product are valid [18].

2.2.2 Bloch Sphere

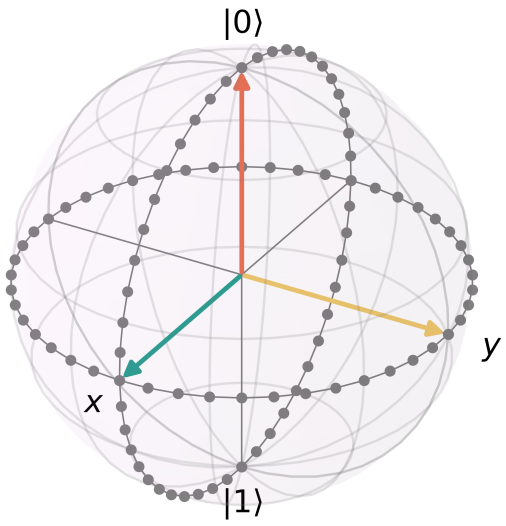


Figure 2.2: Bloch Sphere

The Bloch sphere is a geometric representation of a qubit state in three-dimensional space. It provides an intuitive visualization in quantum computing of the qubit's state and its transformations [19].

A qubit state can be represented as a linear combination of two basis states, typically denoted as $|0\rangle$ and $|1\rangle$. The general form of a qubit state is:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

where α and β are complex probability amplitudes satisfying the normalization condition $|\alpha|^2 + |\beta|^2 = 1$.

Each point on the surface of the sphere corresponds to a unique qubit state. The qubit state can be associated with a Bloch vector $\boldsymbol{r} = (x, y, z)$, where the coordinates (x, y, z) determine the point's position on the Bloch sphere.

The basis states $|0\rangle$ and $|1\rangle$ correspond to specific points on the Bloch sphere. The state $|0\rangle$ is represented by the north pole of the sphere, while the state $|1\rangle$ is represented by the south pole. The equator of the sphere represents a superposition of the two basis states.

2.2.3 Postulates of Quantum Mechanics

Quantum mechanics, a fundamental theory in physics, is described by a set of principles known as postulates. The number of postulates may vary depending on the formulation or

interpretation of quantum mechanics. However, the following four postulates outline the basic assumptions and rules that govern the behavior of quantum systems [21].

Postulate 1. *The state of a quantum system is completely described by a wave function or state vector in Hilbert space. The wave function contains all the information about the system and is represented by the symbol* $|\Psi\rangle$.

Postulate 2. *Physical quantities, such as position, momentum, and energy, can be measured by Hermitian operators called observables, which are linear operators that are equal to their adjoint. Hermitian operators have real eigenvalues, and their eigenvectors form a complete orthonormal set. This property ensures that the results of measurements are real numbers and that the wave function can be expressed as a linear combination of eigenvectors.*

Postulate 3. *The result of a measurement on a quantum system is one of the eigenvalues of the operator corresponding to the observed quantity. The probability of obtaining a particular eigenvalue is given by the square of the absolute value of the projection of the wave function onto the eigenvector corresponding to that eigenvalue, also known as the Born rule. After the measurement, the wave function collapses to the eigenvector corresponding to the observed eigenvalue.*

Postulate 4. *The evolution over time of the wave function is governed by the Schrödinger equation. The equation can be derived from the fact that the time-evolution operator must be unitary, and must therefore be generated by the exponential of a self-adjoint operator, which is the quantum Hamiltonian.*

$$i\overline{h}\frac{d}{dt}|\Psi(\boldsymbol{t})\rangle = \boldsymbol{H}|\Psi(\boldsymbol{t})\rangle \tag{2.1}$$

where t is time, $|\Psi(\boldsymbol{t})\rangle$ *is the state vector of the quantum system (*Ψ *being the Greek letter psi), and* $\boldsymbol{H}$ *is an observable, the Hamiltonian operator.*

These postulates provide the foundation for understanding and predicting the behavior of quantum systems. They have been extensively tested and verified through numerous experiments, making quantum mechanics one of the most successful and accurate theories of the physical properties of nature at the scale of atoms and subatomic particles.

2.2.4 Superposition

Superposition is a fundamental concept in quantum mechanics, where a quantum system can exist in multiple states simultaneously [22]. For example, a state $|\psi\rangle$ can be written as $|\psi\rangle = c_1|\boldsymbol{\phi_1}\rangle + c_2|\boldsymbol{\phi_2}\rangle$, where c_1 and c_2 are complex coefficients and $|\boldsymbol{\phi_1}\rangle$ and $|\boldsymbol{\phi_2}\rangle$ are orthogonal states.

This allows for the combination of different states with varying probabilities, resulting in a complex state that cannot be described by classical probabilities. Mathematically, superposition is represented by a linear combination of basis states:

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

where $|\alpha|^2$ and $|\beta|^2$ represent the probabilities of finding the system in the states $|\mathbf{0}\rangle$ and $|\mathbf{1}\rangle$, respectively. The coefficients α and β are complex numbers that satisfy the normalization condition $|\alpha|^2 + |\beta|^2 = 1$.

2.2.5 Qubits

While this thesis focuses on the exploration of qutrits, it is beneficial to begin with some foundational information on qubits, which are mainly used in existing quantum hardware.

Qubits are often represented using two-dimensional complex vectors, and the state of a qubit can be described as a superposition of basis states [20]. A qubit ket vector in the computational basis can be represented as:

$$|\mathbf{0}\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |\mathbf{1}\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

The corresponding bra vectors are obtained by taking the complex conjugate transpose:

$$\langle\mathbf{0}| = \begin{pmatrix} 1 & 0 \end{pmatrix}, \quad \langle\mathbf{1}| = \begin{pmatrix} 0 & 1 \end{pmatrix}$$

Inner Product

The inner product of two qubit ket vectors $|\psi\rangle$ and $|\phi\rangle$ can be calculated as:

$$\langle\psi|\phi\rangle = \psi_0^*\phi_0 + \psi_1^*\phi_1$$

Outer Product

The outer product of two qubit ket vectors $|\psi\rangle$ and $|\phi\rangle$ gives the corresponding matrix representation:

$$|\psi\rangle\langle\phi| = \begin{pmatrix} \psi_0\phi_0^* & \psi_0\phi_1^* \\ \psi_1\phi_0^* & \psi_1\phi_1^* \end{pmatrix}$$

2.2.6 Measurement

In quantum mechanics, measurements are represented by observables, which are represented by Hermitian operators. The outcome of a measurement is obtained by calculating the expectation value, given by $\langle\psi|\boldsymbol{A}|\psi\rangle$, where $\boldsymbol{A}$ is the observable and $|\psi\rangle$ is the quantum state.

To measure an observable, such as a Pauli operator, using the expectation value, the following steps are typically followed:

1. Consider a quantum system described by a quantum state $|\psi\rangle$.
2. Apply the Pauli operator to the quantum state, for example $\boldsymbol{\sigma_z}|\psi\rangle$.
3. Calculate the inner product of the resulting state with the original state $|\psi\rangle$, i.e., $\langle\psi|\boldsymbol{\sigma_z}|\psi\rangle$.
4. The expectation value is obtained as the absolute square of the inner product, i.e., $|\langle\psi|\boldsymbol{\sigma_z}|\psi\rangle|^2$.

The expectation value represents the average value of the measurement outcome when the measurement is repeated many times on identical systems prepared in the state $|\psi\rangle$. It helps to characterize the quantum state and understand its properties.

Expectation value is a statistical quantity, and individual measurements may not always yield the exact expected value. However, repeated measurements on identically prepared quantum systems will converge to the expectation value over time [20].

2.2.7 Entanglement

Entanglement, as a fundamental property of quantum mechanics, enables remarkable capabilities in quantum information processing. It allows for the encoding and manipulation of information in ways that go beyond classical systems. When two or more quantum systems become entangled, their states become inseparably linked, and the state of one system cannot be described independently of the others.

Mathematically, entanglement is represented using tensor products of ket vectors. For example, if we have two quantum systems represented by the ket vectors $|\phi_1\rangle$ and $|\phi_2\rangle$, their entangled state is given by $|\psi\rangle = |\phi_1\rangle \otimes |\phi_2\rangle$. This implies that the combined state of the two systems cannot be expressed as a simple product of their individual states.

Entanglement finds significant applications in quantum computing algorithms, where entangled qubits can be used to perform parallel computations and achieve exponential speedup in certain tasks. It also plays a crucial role in quantum teleportation, allowing the transmission of quantum states between distant locations by transferring the entanglement to another system. In quantum cryptography, entanglement enables secure communication protocols, such as quantum key distribution, that are intrinsically protected against eavesdropping.

A well-known example of entanglement involves a pair of qubits in the Bell state:

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|\mathbf{00}\rangle + |\mathbf{11}\rangle).$$

In this state, the two qubits are entangled in such a way that measuring the state of one qubit instantaneously determines the state of the other qubit, regardless of the physical distance between them. This phenomenon, known as quantum entanglement, defies classical intuition and plays a central role in understanding and harnessing the power of quantum information processing [20].

2.2.8 Tensors

In the context of quantum computing, tensors play a fundamental role for representing and manipulating multi-qubit systems as well as multi-qudit systems. Tensors are mathematical objects that generalize vectors and matrices to higher dimensions [23].

Tensor Products

The tensor product is a fundamental operation that combines two or more vectors or matrices to form a larger composite object. In quantum computing, the tensor product is used to construct the joint state of multiple qubits.

For two quantum states $|\psi\rangle$ and $|\phi\rangle$ represented by column vectors, their tensor product is given by:

$$|\psi\rangle \otimes |\phi\rangle = \begin{pmatrix} \psi_1|\phi\rangle \\ \psi_2|\phi\rangle \\ \vdots \\ \psi_n|\phi\rangle \end{pmatrix},$$

where ψ_i represents the ith component of $|\psi\rangle$.

Similarly, for two quantum operators $\mathbf{A}$ and $\mathbf{B}$ represented by matrices, their tensor product is given by:

$$\mathbf{A} \otimes \mathbf{B} = \begin{pmatrix} a_{11}\boldsymbol{B} & a_{12}\boldsymbol{B} & \cdots & a_{1n}\boldsymbol{B} \\ a_{21}\boldsymbol{B} & a_{22}\boldsymbol{B} & \cdots & a_{2n}\boldsymbol{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}\boldsymbol{B} & a_{n2}\boldsymbol{B} & \cdots & a_{nn}\boldsymbol{B} \end{pmatrix},$$

where a_{ij} represents the elements of matrix $\mathbf{A}$.

Tensor Product of Operators

The tensor product of operators is used to describe the joint action of operators on composite systems. For two operators $\mathbf{A}$ and $\mathbf{B}$, their tensor product $\mathbf{A} \otimes \mathbf{B}$ represents the combined action of $\mathbf{A}$ on one subsystem and $\mathbf{B}$ on another subsystem.

Mathematically, if $\mathbf{A}$ and $\mathbf{B}$ are operators acting on Hilbert spaces $\mathcal{H}_1$ and $\mathcal{H}_2$ respectively, then the tensor product $\mathbf{A} \otimes \mathbf{B}$ acts on the composite Hilbert space $\mathcal{H}_1 \otimes \mathcal{H}_2$. The action of the tensor product operator is defined as follows:

$$(\mathbf{A} \otimes \mathbf{B})\,|\psi_\mathbf{1}\rangle \otimes |\psi_\mathbf{2}\rangle = (\mathbf{A}|\psi_\mathbf{1}\rangle) \otimes (\mathbf{B}|\psi_\mathbf{2}\rangle)$$

for all vectors $|\psi_\mathbf{1}\rangle \in \mathcal{H}_1$ and $|\psi_\mathbf{2}\rangle \in \mathcal{H}_2$.

The tensor product of operators is important in quantum computing as it allows us to describe the combined dynamics of multiple subsystems or qubits. It forms the basis for constructing multi-qubit gates and representing entangled states.

Tensor Contractions

In the context of quantum computing, tensor contractions are often used to trace out subsystems. If we have a tensor $\boldsymbol{T}_{ijk}$ representing a composite system, tracing out a subsystem amounts to summing over the indices of that subsystem:

$$\boldsymbol{T}_{ij} = \sum_{k} \boldsymbol{T}_{ijk}$$

The resulting tensor $\boldsymbol{T}_{ij}$ represents the reduced system after tracing out the subsystem.

Furthermore, tensor contractions are used to calculate expectation values of operators. For example, given a state vector $|\boldsymbol{\psi}\rangle$ and an operator $\mathbf{A}$, the expectation value of $\mathbf{A}$ can be calculated by contracting the indices of the operator with the state vector:

$$\langle \mathbf{A} \rangle = \langle \boldsymbol{\psi} | \mathbf{A} | \boldsymbol{\psi} \rangle = \sum_{i} \langle \boldsymbol{\psi}_i | \mathbf{A} | \boldsymbol{\psi}_i \rangle$$

where $|\boldsymbol{\psi}\rangle_{\boldsymbol{i}}$ are the components of the state vector in a particular basis.

2.3 Qutrits

Qudits extend the concept of qubits, which have two distinguishable states, to a higher-dimensional space of d distinguishable states, where d is a positive integer. Qutrits, in particular, are qudits with three distinguishable states, often denoted as $|\mathbf{0}\rangle$, $|\mathbf{1}\rangle$, and $|\mathbf{2}\rangle$.

2.3.1 Qutrit States

A qutrit ket vector in the computational basis can be represented as:

$$|\mathbf{0}\rangle = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}, \quad |\mathbf{1}\rangle = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \quad |\mathbf{2}\rangle = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

The corresponding bra vectors are obtained by taking the complex conjugate transpose:

$$\langle \mathbf{0}| = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}, \quad \langle \mathbf{1}| = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}, \quad \langle \mathbf{2}| = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}$$

Mathematically, qutrit states are represented as vectors in a three-dimensional Hilbert space [24]. The state of a qutrit can be expressed as a linear combination of the basis states $|\mathbf{0}\rangle$, $|\mathbf{1}\rangle$, and $|\mathbf{2}\rangle$. A general qutrit state $|\boldsymbol{\psi}\rangle$ can be written as:

$$|\boldsymbol{\psi}\rangle = \alpha|\mathbf{0}\rangle + \beta|\mathbf{1}\rangle + \gamma|\mathbf{2}\rangle,$$

where α, β, and γ are complex probability amplitudes that satisfy the normalization condition $|\alpha|^2 + |\beta|^2 + |\gamma|^2 = 1$.

2.3.2 Inner Product

The inner product of two qutrit ket vectors $|\psi\rangle$ and $|\phi\rangle$ can be calculated as:

$$\langle\psi|\phi\rangle = \psi_0^*\phi_0 + \psi_1^*\phi_1 + \psi_2^*\phi_2$$

2.3.3 Outer Product

The outer product of two qutrit ket vectors $|\psi\rangle$ and $|\phi\rangle$ gives the corresponding matrix representation:

$$|\psi\rangle\langle\phi| = \begin{pmatrix} \psi_0\phi_0^* & \psi_0\phi_1^* & \psi_0\phi_2^* \\ \psi_1\phi_0^* & \psi_1\phi_1^* & \psi_1\phi_2^* \\ \psi_2\phi_0^* & \psi_2\phi_1^* & \psi_2\phi_2^* \end{pmatrix}$$

2.3.4 Trace

The trace of a qutrit operator represents the sum of its diagonal elements. Tracing out qutrit subsystems plays a crucial role in calculations and measurements. The trace of a qutrit operator $\mathbf{O}$ can be calculated as:

$$\text{Tr}(\mathbf{O}) = \mathbf{O}_{00} + \mathbf{O}_{11} + \mathbf{O}_{22}$$

The trace allows us to obtain reduced density matrices and calculate expectation values of observables.

2.3.5 Density Operator

In quantum mechanics, the density operator, denoted by $\boldsymbol{\rho}$, is a mathematical representation of a quantum state that accounts for both pure and mixed states. It provides a way to describe the statistical properties of a quantum system. For a qutrit system, the density operator is a 3×3 matrix given by:

$$\boldsymbol{\rho} = \sum_{i,j=0}^{2} \boldsymbol{\rho}_{ij} |\boldsymbol{i}\rangle\langle\boldsymbol{j}|$$

where $\boldsymbol{\rho}_{ij}$ represents the matrix elements of the density operator and $|\boldsymbol{i}\rangle\langle\boldsymbol{j}|$ represents the outer product of the qutrit ket vectors.

The density operator is Hermitian: $\boldsymbol{\rho}^\dagger = \boldsymbol{\rho}$, Positive Semidefinite: All eigenvalues of $\boldsymbol{\rho}$ are non-negative, and has Normalized Trace: $\text{Tr}(\boldsymbol{\rho}) = 1$.

The density operator allows us to describe mixed states, which are statistical ensembles of pure states. The density operator also provides a way to calculate expectation values of observables. The expectation value of an observable $\mathbf{A}$ with respect to the density operator $\boldsymbol{\rho}$ is given by:

$$\langle\mathbf{A}\rangle = \text{Tr}(\boldsymbol{\rho}\mathbf{A})$$

2.3.6 Entanglement and Density Operator

To investigate whether a quantum state is entangled, one can perform a partial trace operation on the density operator. Let's consider a composite system of two subsystems, labeled as A and B, with corresponding Hilbert spaces $\mathcal{H}_A$ and $\mathcal{H}_B$. To check for entanglement, performing a partial trace over one of the subsystems, say subsystem B, the resulting reduced density operator, denoted by $\boldsymbol{\rho}_A$, is obtained:

$$\boldsymbol{\rho}_A = \mathsf{Tr}_B(\boldsymbol{\rho}_{AB})$$

If the reduced density operator $\boldsymbol{\rho}_A$ is a pure state (i.e., rank-1 operator), then the original state of the composite system is entangled. On the other hand, if $\boldsymbol{\rho}_A$ is a mixed state (i.e., a statistical ensemble of states), then the original state is separable and not entangled.

The eigenvalues of the reduced density operator $\boldsymbol{\rho}_A$ can also provide information about the entanglement of the system. If at least one eigenvalue is zero, it indicates the presence of entanglement in the composite system.

2.3.7 Measurement

Similar to qubits, measurements on qutrits are represented by observables, which are Hermitian operators.

The outcome of a measurement is determined by calculating the expectation value $\langle\boldsymbol{\psi}|\mathbf{A}|\boldsymbol{\psi}\rangle$, where $\mathbf{A}$ is the observable and $|\boldsymbol{\psi}\rangle$ is the qutrit state:

$$\langle\boldsymbol{\psi}|\mathbf{A}|\boldsymbol{\psi}\rangle = \sum_{i,j=0}^{2} \boldsymbol{\psi}_i^* \mathbf{A}_{ij} \boldsymbol{\psi}_j$$

where $\boldsymbol{\psi}_i$ and $\boldsymbol{\psi}_j$ are the probability amplitudes of the qutrit state in the computational basis, and $\mathbf{A}_{ij}$ represents the elements of the matrix representation of the observable $\mathbf{A}$.

2.3.8 Operators

The quantum logic gates operating on single qutrits are represented as 3×3 unitary matrices [25]. Specifically, the rotation operator gates for $SU(3)$ are defined as:

$$\textbf{Rot}(\Theta_1, \Theta_2, \ldots, \Theta_8) = \mathsf{exp}\left(-i \sum_{a=1}^{8} \Theta_a \frac{\boldsymbol{\lambda}_a}{2}\right) \tag{2.2}$$

Here, $\boldsymbol{\lambda}_a$ refers to the ath Gell-Mann matrix, which was introduced in the previous section. The Θ_a values are real numbers with a periodicity of 4π.

These operators have significant importance in this thesis as they are utilized for both the encoding of the feature map and the rotations within the variational layer of the parameterized circuit. Detailed explanation on their use will be presented in subsequent chapters.

3. MACHINE LEARNING

3.1 Machine Learning Fields

Before delving into the exciting and rapidly changing field of quantum machine learning, a brief overview of machine learning is provided, which can be broadly classified into three subfields: supervised learning, unsupervised learning, and reinforcement learning.

In supervised learning, the goal is to learn a function that maps labeled data to new, unseen inputs. The labeled data typically consists of input-output pairs, where the input is a set of features or variables and the output is a known label or target value. The algorithm's objective is to find a function that generalizes to new, unseen data, allowing for accurate predictions or classifications to be made [26].

Unsupervised learning, on the other hand, aims to identify patterns and structure in unlabeled data. The algorithm is not provided with any explicit labels or target values. Instead, it must find structure and relationships within the data itself. Examples of unsupervised learning include clustering, dimensionality reduction, and anomaly detection.

Reinforcement learning is a type of machine learning that focuses on maximizing rewards in response to environmental stimuli. In this paradigm, an agent interacts with an environment and learns to take actions that maximize a cumulative reward signal. The objective is to find an optimal policy that maximizes the expected cumulative reward over time.

Supervised learning can be further classified into two major tasks: regression and classification. In regression, the goal is to predict a continuous output value, such as the yield of a crop based on factors like temperature, rainfall, and soil quality. On the other hand, in classification, the goal is to predict a discrete label or class, such as whether a tumor is malignant or benign based on its size, shape, and other characteristics.

Moreover, classification can be subdivided into binary classification and multi-label classification. In binary classification, the output label is either true or false, such as predicting whether an email is spam or not. In contrast, multi-label classification involves predicting multiple output labels, such as predicting which topics a news article might be relevant to.

3.2 Supervised Learning

Supervised learning involves training on a labeled dataset consisting of input data, or features, $\boldsymbol{x} \in \mathbb{R}^n$, and corresponding output labels, or targets, $\boldsymbol{y} \in \mathcal{Y}$. The goal is to learn a mapping between the input data and output labels, so that the algorithm can make predictions on new, unseen data.

This is achieved by minimizing a loss function $\mathcal{L}(y, f(\boldsymbol{x}))$ that measures the difference between the predicted output $\hat{y} = f(\boldsymbol{x})$ and the true output y. The mapping function $f(\cdot)$ is typically parameterized by a set of learnable parameters $\boldsymbol{\theta}$, and the optimization problem

can be formulated as follows:

$$\theta^* = \arg\min_\theta \frac{1}{N} \sum_{i=1}^{N} L\left(y_i, f_\theta(x_i)\right)$$

where N is the number of labeled examples in the dataset. The model is trained by finding the parameters θ that minimize the loss function $L(\theta)$ that measures the quality of the model with respect to the training data set [26].

3.2.1 Cost Function

The loss function can take various forms, depending on the type of problem being solved. For example, for a regression problem where the output y is a continuous variable, a common loss function is the mean squared error:

$$L(\hat{y}, y) = \frac{1}{2}(y - \hat{y})^2$$

For a classification problem where the output y is a categorical variable, a common loss function is the cross-entropy loss:

$$L(\hat{y}, y) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

where C is the number of classes, y_i is a binary indicator (0 or 1) of whether the true label is class i, and $\hat{y}_i$ is the predicted probability of class i.

The goal of training is to find the values of the parameters θ that minimize the average loss over the entire dataset:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} L(f(x^{(i)}; \theta), y^{(i)})$$

Once the algorithm has been trained on the labeled dataset and the model parameters $\boldsymbol{\theta}^*$ are obtained, it can be used to make predictions on new, unseen data [27].

3.2.2 Regularization

Regularization is a common technique used to prevent overfitting in machine learning models. L1 regularization, also known as Lasso regularization, adds a penalty term to the cost function proportional to the sum of the absolute values of the model's weights. In other words, it encourages the model to have sparse weights by shrinking some of them to zero. This can be helpful when the number of features in the dataset is large, and many of them are not relevant to the prediction task.

$$R1(\theta) = \lambda \sum_{i=1}^{n} |\theta_i| \tag{3.1}$$

L2 regularization, also known as Ridge regularization, adds a penalty term proportional to the sum of the squares of the model's weights. This encourages the model to have small weights by spreading the influence of each weight across all the features.

$$R2(\theta) = \lambda \sum_{i=1}^{n} {\theta_i}^2 \tag{3.2}$$

In both cases, θ represents the model parameters, n is the number of parameters, and λ is the regularization strength hyperparameter [27].

3.2.3 Optimization

In the context of machine learning, optimizers are algorithms used to iteratively update the parameters of a model in order to minimize the loss function. Different optimizers use different strategies for updating the parameters, with some methods being more efficient or effective than others depending on the problem at hand.

Gradient-based methods aim to identify an optimal solution by finding a point where the gradient is equal to zero. Some common optimizers include stochastic gradient descent (SGD), which updates the parameters in the direction of the negative gradient of the loss function, and variants such as momentum-based methods, which use a moving average of past gradients to smooth the update process and accelerate convergence. For the purposes of this thesis, optimizers from the SGD family were primarily used, specifically Adam and RMSProp proposed by Geoffrey Hinton.

RMSprop, short for Root Mean Square Propagation, is an optimization algorithm that adapts the learning rate for each parameter individually based on the magnitude of recent gradients. This adaptive learning rate helps to mitigate the issues of vanishing or exploding gradients, making it suitable for training deep neural networks. The algorithm maintains a moving average of the squared gradients for each parameter, which is then used to update the parameters. By dividing the learning rate by the square root of this average, RMSprop scales the learning rate based on the history of gradients. This helps to converge faster in regions with consistent gradients while being more cautious in fluctuating or noisy regions.

In RMSProp (Root Mean Square Propagation) the idea is to divide the learning rate for a weight by a running average of the magnitudes of recent gradients for that weight. The running average is calculated in terms of mean square as:

$$v(w,t) := \gamma v(w,t-1) + (1-\gamma)\left(\nabla Q_i(w)\right)^2$$

The concept of storing the gradients as a sum of squares is borrowed from Adagrad and ”forgetting”, with the forgetting factor γ, is introduced to solve Adagrad's diminishing learning rates in non-convex problems by gradually decreasing the influence of old data [28]. The parameters are updated as follows:

$$w := w - \frac{\eta}{\sqrt{v(w,t)}} \nabla Q_i(w)$$

RMSProp has shown good adaptation of learning rates and is capable of working with mini-batches. A simple outline of the algorithm can be seen below.

Algorithm 1 Root Mean Square Propagation

- Initialize parameters: θ
- Initialize first moment variable: $v = 0$
- Initialize learning rate: α
- Initialize decay rate: ρ

While stopping criterion not met

1. Compute gradient: $g \leftarrow \nabla L(\theta)$
2. Update first moment estimate: $v \leftarrow \rho v + (1 - \rho) g^2$
3. Update parameters: $\theta \leftarrow \theta - \frac{\alpha}{\sqrt{v+\epsilon}} g$ {ϵ is a small constant for numerical stability}

Another popular optimizer is Adam (short for Adaptive Moment Estimation) [29], a 2014 update to the RMSProp optimizer combining it with the main feature of the Momentum method. In this optimization algorithm, running averages with exponential forgetting of both the gradients and the second moments of the gradients are used. While Adam was trialed, it encountered difficulties in navigating the loss landscapes of these particular quantum models. Consequently, RMSProp emerged as the primary choice for the training process.

Now, gradient-based optimization often can be challenging as it may converge slowly and can fail to find the global optimum due to the presence of multiple local minima in the optimization problem. In these scenarios, gradient-free methods can be a useful alternative, as they can overcome the issue of local minima. However, they require higher computational capacity, especially for problems with high-dimensional search spaces.

It is important to note that finding an exact minimum of the loss function is not always necessary, as the ultimate goal is to create a model that can make accurate predictions. Regardless of the optimization method, determining the direction to search in a flat loss landscape can be challenging, a phenomenon known as a barren plateau [30]. In situa-

tions where gradient-based optimization is applicable, but the gradient is not available in closed form, Cobyla and L-BFGS-B can be valuable optimization methods to consider.

COBYLA is a derivative-free optimization algorithm that is designed for constrained optimization problems [31]. It approximates the objective function using a linear model and then minimizes the model within the constraints. It is particularly useful when gradient information is not available or when the objective function is noisy.

L-BFGS-B, on the other hand, is a quasi-Newton optimization algorithm that is designed for optimizing a function with bound constraints [32]. It approximates the inverse Hessian matrix using previous iterations and gradients, which makes it very efficient for high-dimensional problems. It is typically used for optimizing smooth, non-linear functions with bound constraints. These two alternatives were used for the optimization of the qubit-based circuits in this thesis for the purpose of comparison.

3.3 Principal Component Analysis

Principal Component Analysis (PCA) is a popular method for dimensionality reduction, which involves transforming higher-dimensional data into a smaller space, while preserving as much of the original information as possible. Dimensionality reduction is particularly useful when working with datasets with many features that may make the training of the models computationally expensive.

PCA works by finding the directions in the feature space that capture the most variance in the data [33]. These directions are called principal components, and they can be used to derive new, transformed features that retain most of the important information of the original features. The first principal component captures the direction of largest variance, the second principal component captures the direction of second largest variance, and so on.

To find the principal components, PCA applies a linear transformation on the original features such that the transformed features (the principal components) are uncorrelated and have decreasing variance. The first principal component is the linear combination of the original features that captures the most variance, and the second principal component is the linear combination of the original features that captures the most variance among all linear combinations that are uncorrelated with the first principal component, and so on.

In this thesis, Principal Component Analysis (PCA) is employed on real-world datasets characterized by highly correlated features. Retaining only the initial few components in these, reduces the dimensions and enhances computational efficiency, without compromising accuracy.

Quantum principal component analysis (qPCA) algorithms have also been proposed. In the paper by Lloyd, Mohseni and Rebentrost it is demonstrated how the quantum phase estimation algorithm can be used to estimate the eigenvalues of the dataset's covariance matrix, which are necessary for computing the principal components [34]. By using mul-

tiple copies of a quantum system with density matrix , qPCA can construct the unitary transformation e^{-it}, allowing to identify the eigenvectors that correspond to the large eigenvalues of an unknown low-rank density matrix much faster than any existing algorithm.

Although this topic is not within the scope of this thesis, it is worth mentioning that qPCA algorithms can provide exponential speedups compared to classical algorithms and have the potential to accelerate machine learning tasks such as clustering and pattern recognition. However, it is important to note that these algorithms require quantum random access memory (qRAM) [35], for which there is currently no feasible hardware options.

3.4 Support Vector Machines

Support Vector Machines (SVM) are a type of supervised learning algorithm used for classification and regression tasks in machine learning. They are particularly well-suited for handling high-dimensional data and are known for their ability to find the optimal separation between classes or to predict a continuous target variable.

The main idea behind SVM is to find the best hyperplane that separates different classes in the feature space [36]. This hyperplane is chosen in such a way that it maximizes the margin between the classes, which is the distance between the hyperplane and the nearest data points from each class. These nearest data points are called support vectors. Mathematically, SVM for binary classification can be formalized as follows:

Given a set of training data points $\boldsymbol{x_i}$ and their corresponding binary labels $y_i \in \{-1, 1\}$, the goal is to find the optimal hyperplane $\boldsymbol{w}^T\boldsymbol{x} + b = 0$ that maximizes the margin:

$$\boldsymbol{w}^T\boldsymbol{x} + b = 0$$

The margin is defined as the distance between the hyperplane and the support vectors, which can be calculated as $\frac{2}{||\boldsymbol{w}||}$:

$$\text{Margin} = \frac{2}{||\boldsymbol{w}||}$$

To maximize the margin, we need to minimize $||\boldsymbol{w}||$, subject to the constraint $y_i(\boldsymbol{w}^T\boldsymbol{x_i}+b) \geq 1$ for all data points i. This can be represented as an optimization problem:

$$\min_{\boldsymbol{w}} \frac{1}{2}||\boldsymbol{w}||^2, \quad \text{subject to } y_i(\boldsymbol{w}^T\boldsymbol{x_i} + b) \geq 1 \quad \forall i$$

SVMs can also be used for non-linear classification problems, where a technique called the kernel trick is applied. The kernel trick involves the use of feature maps to enable SVMs to perform non-linear classification, by transforming the input data into a new space

where they become separable. A feature map is a function that maps a data vector to a higher-dimensional feature space:

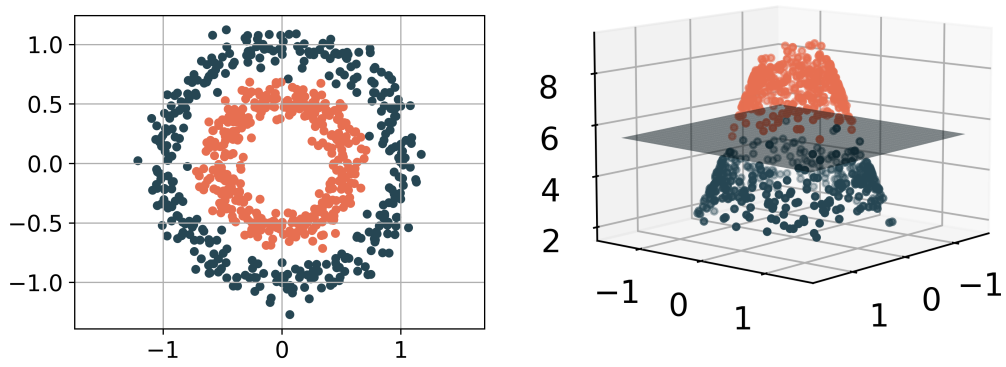
$$\varphi : \mathbb{R}^n \to \mathbb{R}^m$$

A kernel function is a function that corresponds to the dot product of two data points in this higher-dimensional feature space:

$$K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \varphi(\boldsymbol{x_i})^T \varphi(\boldsymbol{x_j})$$

Some common kernel functions include:

- Linear kernel: $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = \boldsymbol{x_i}^T \boldsymbol{x_j}$
- Polynomial kernel: $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = (\boldsymbol{x_i}^T \boldsymbol{x_j} + c)^d$
- Radial basis function (RBF) kernel: $K(\boldsymbol{x_i}, \boldsymbol{x_j}) = e^{-\gamma ||\boldsymbol{x_i} - \boldsymbol{x_j}||^2}$



(a) Non-linearly separable Circes dataset (b) Mapping to a 3-dimensional space

Figure 3.1: Illustration of kernel trick with a separating hyperplane

For the optimization problem, the dot product $\boldsymbol{x_i}^T \cdot \boldsymbol{x_j}$ between the data points in the original space is replaced with a kernel function $K(\boldsymbol{x_i}, \boldsymbol{x_j})$ that maps them in the higher-dimensional space. The decision function can then be written as:

$$f(\boldsymbol{x}) = \text{sign}\left(\sum_{i=1}^{n} \alpha_i y_i K(\boldsymbol{x_i}, \boldsymbol{x}) + b\right)$$

where α_i are the Lagrange multipliers obtained from solving the dual problem. The dual form can be solved efficiently, as the kernel function is typically more computationally efficient. This makes it suitable for handling complex decision boundaries and high-dimensional spaces, and enables the kernel trick. The mapping is done implicitly through the kernel function, which computes the dot product between data points in the higher-dimensional space without explicitly transforming them.

SVMs are employed in this thesis to facilitate a comparison between quantum and classical counterparts, but also to incorporate quantum feature maps. Quantum feature maps allow the mapping of classical data into quantum states, exploiting the enhanced representation capabilities of quantum systems with the similarity between the quantum states generated by the feature map and the desired target states.

3.5 Neural Networks

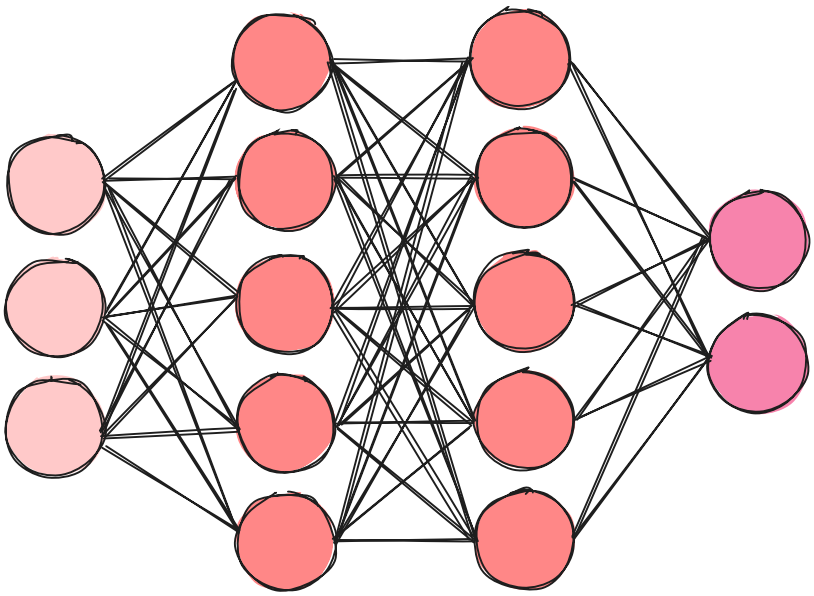


Figure 3.2: An artificial neural network

Neural networks are powerful models used in machine learning for tasks such as classification, regression, and pattern recognition. They are composed of interconnected layers of artificial neuron. The basic building blocks of neural networks are perceptrons, introduced by Frank Rosenblatt in 1957 as a supervised learning model for binary classifiers [37].

A perceptron is a simple binary classifier used in machine learning. It takes a set of input features $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ and assigns weights $\boldsymbol{w} = (w_1, w_2, \ldots, w_n)$ to each feature. The weighted sum of the inputs is then passed through an activation function to produce the output. Mathematically, the perceptron's output y can be represented as:

$$y = \begin{cases} 1 & \text{if } \boldsymbol{w} \cdot \boldsymbol{x} + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

where $\cdot$ denotes the dot product of vectors, b is the bias term, and the activation function is typically a step function. The perceptron learning algorithm is used to adjust the weights and bias to minimize classification errors. It updates the weights according to the formula:

$$w_i \leftarrow w_i + \eta \cdot (y - \hat{y}) \cdot x_i$$

where η is the learning rate, y is the predicted output, and $\hat{y}$ is the true output.

A perceptron can be used for linear binary classification tasks, but it has limitations when it comes to learning complex patterns and handling non-linearly separable data. However, when combined into multi-layer perceptrons or feed-forward neural networks, they can be used for more complex tasks and can be trained using powerful algorithms like backpropagation. The multilayer perceptron is a universal function approximator, as proven by the universal approximation theorem.

In a feedforward neural network, information flows from the input layer through one or more hidden layers to the output layer. Each node in the network applies a linear transformation to the inputs, followed by a non-linear activation function. Mathematically, the output y of a node can be represented as:

$$y = f(\boldsymbol{w} \cdot \boldsymbol{x} + b)$$

where $\cdot$ denotes the dot product of vectors, $\boldsymbol{w}$ represents the weights, $\boldsymbol{x}$ is the input, b is the bias term, and f is the activation function.

Training a neural network involves adjusting the weights and biases to minimize a loss function that measures the discrepancy between predicted outputs and true targets. This process is typically performed using optimization algorithms like stochastic gradient descent (SGD) and backpropagation, which iteratively update the weights based on the gradients of the loss function.

Neural networks have the ability to learn complex patterns and relationships from data, can handle high-dimensional inputs, capture non-linearities, and generalize well to unseen examples. Through proper design and training, neural networks can achieve impressive performance across a wide range of applications.

In this thesis, the aim is to explore the applicability of key concepts from classical neural networks, such as stacked layers, to quantum neural networks. The effectiveness of these architectures will be investigated specifically within the context of parametrized quantum circuits (PQCs). The fundamental idea underlying this investigation is that a PQC can be analogously regarded as a perceptron in the classical sense, thus suggesting that a Quantum Neural Network (QNN) can be conceptualized as stacked PQC layers.

3.6 Metrics

Performance metrics play a vital role in the evaluation of a model in machine learning and data science. While Accuracy is widely employed, the inclusion of Precision, Recall, and F1-Score provides a more thorough assessment, facilitating a comprehensive understanding of a model's effectiveness. By incorporating these metrics in this thesis, insights can be gained regarding the particular strengths and weaknesses of the models.

Accuracy is a commonly used performance metric that measures the overall correctness of a classification model. It calculates the proportion of correctly predicted instances out of the total number of instances. Accuracy is defined as:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$

While accuracy provides a general measure of model performance, it may not be sufficient when dealing with imbalanced datasets where the classes have significantly different sizes.

Precision focuses on the number of correctly predicted positive instances out of the total predicted positive instances. It quantifies the proportion of true positives out of all positive predictions. Precision is defined as:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Precision is useful in scenarios where minimizing false positives is important.

Recall measures the ability of a model to identify positive instances correctly. It quantifies the proportion of true positives (correctly predicted positive instances) out of all actual positive instances (both true positives and false negatives). Recall is defined as:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

A high recall indicates a low rate of false negatives.

F1-score is the harmonic mean of precision and recall, providing a single metric that balances both metrics. It takes into account both false positives and false negatives. The F1-score is defined as:

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The F1-score combines precision and recall into a single value. It is a useful metric when both false positives and false negatives need to be minimized, and a balance between precision and recall is desired [27].

4. PARAMETERIZED QUANTUM CIRCUITS

Quantum machine learning (QML), a field that combines quantum computing and machine learning, explores the use of quantum systems to enhance the performance of machine learning algorithms. There are four different approaches to QML, depending on whether the data and algorithms are classical or quantum [41].

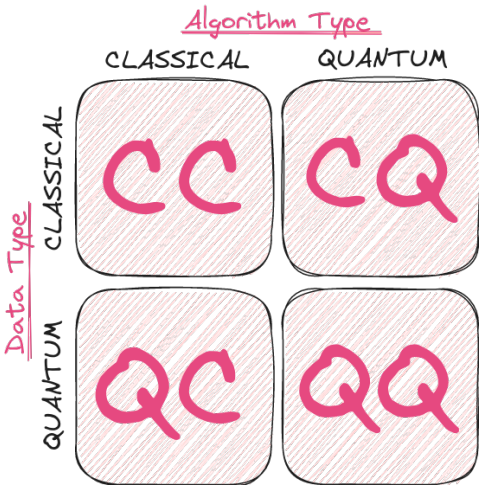


Figure 4.1: Quantum Machine Learning Approaches

1. Classical Data, Classical Algorithms: In this approach, both the dataset and the learning algorithms are classical. The dataset consists of observations from classical systems such as time series, text, or images, and the machine learning algorithms used are classical, but are inspired by quantum computing.

2. Quantum Data, Classical Algorithms: Here, the dataset contains observations obtained from natural or artificial quantum systems, while the learning algorithms remain classical. Quantum systems produce quantum data, which can be measurements of quantum states and classical machine learning algorithms are applied to analyze and extract insights from this quantum data.

3. Classical Data, Quantum Algorithms: In this case, the dataset comprises observations from classical systems, while the learning algorithms employed are quantum. By applying quantum algorithms to classical data, it is possible to achieve speedups or uncover patterns that might be challenging for classical algorithms.

4. Quantum Data, Quantum Algorithms: This approach involves both quantum datasets and quantum algorithms. The dataset consists of observations from quantum systems, and the learning algorithms used are specifically designed for quantum data analysis. This approach leverages the unique properties of quantum systems to process and extract meaningful information from quantum datasets.

Parameterized quantum circuits (PQC) belong to the category of processing classical data using quantum algorithms. They exhibit remarkable expressive power and serve as a key component for data-driven tasks, such as supervised learning, where they can be used for classification and regression tasks. They have also been used for generative modeling, allowing the generation of new samples that follow the same statistical patterns as the training data [42].

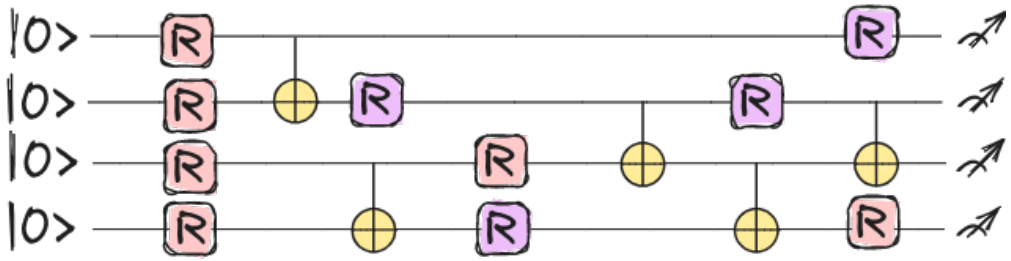


Figure 4.2: Parameterized Quantum Circuit

A PQC is simply a quantum circuit that contains adjustable parameters. These parameters can be optimized during the training process to adapt the circuit to specific tasks or datasets. They have been shown to have remarkable expressive power, allowing them to represent complex functions and correlations in data.

The training of a PQC involves finding the optimal values for its adjustable parameters. This is typically done using optimization algorithms, such as gradient-based methods, to minimize a chosen objective function or loss function. The training process can be performed using classical computers or on existing quantum hardware.

While PQCs offer significant potential, it is important to note that current hardware limitations pose some challenges to their practical implementation. Algorithms like QPCA, QSVM, and qClustering utilize qRAM to store and access large amounts of data efficiently, and offer exponential speedups compared to classical machine learning algorithms. However, there is no viable solution to realising qram. As a result, much of the focus in QML has been on developing near-term algorithms that can be executed on existing quantum devices.

While PQCs offer significant potential, it is important to note that current hardware limitations pose challenges to their practical implementation. Algorithms such as QPCA, QSVM, and qClustering utilize qRAM to efficiently store and access large amounts of data, providing exponential speedups compared to classical machine learning algorithms. However, the lack of a viable solution for realizing qRAM remains an obstacle. Consequently, much of the focus in QML has been on developing near-term algorithms that can be executed on existing quantum devices.

4.1 Quantum Gates

Quantum gates are the fundamental building blocks of quantum computing, playing a crucial role in the manipulation of qubits or qudits and their quantum states. These gates are represented as unitary transformations, ensuring that the total probability of finding a quantum system in any state remains constant.

The versatility of quantum gates lies in their ability to perform intricate operations on the quantum state of a system, allowing for complex computations and information processing. Unlike classical gates that operate on classical bits, quantum gates exploit the principles of superposition and entanglement to process information in a quantum parallel and exponentially scalable manner.

Various types of quantum gates exist, each tailored for specific purposes. Single-qubit gates act on individual qubits, altering their quantum states by rotating or transforming their probability amplitudes. Examples of single-qubit gates include the Hadamard gate, which creates superposition, and the Pauli gates (X, Y, Z), which perform rotations.

Multi-qubit gates, on the other hand, enable interactions between multiple qubits, leading to entanglement and entwined quantum states. The most well-known multi-qubit gate is the Controlled-NOT (CNOT) gate, which flips the target qubit if and only if the control qubit is in a specific state. This gate serves as a fundamental building block for constructing quantum circuits and executing quantum algorithms [43].

Gate Composition

Quantum gates can be composed to perform complex quantum operations. The composition of gates is achieved by matrix multiplication. In the context of qudits, where each qudit has d distinguishable states, the gates are represented by $d \times d$ unitary matrices. By multiplying these matrices, we can combine multiple gates to create new quantum operations.

Gate Universality

A universal set of gates typically consists of a small number of elementary gates that can be combined and applied in specific sequences to achieve arbitrary quantum computations. The most well-known universal set of gates is the set of one-qubit gates and the CNOT gate. By combining one-qubit gates and the CNOT gate, it is possible to construct any unitary transformation on the quantum state space.

4.1.1 Qubit Gates

In this subsection, an overview of the fundamental single-qubit gates that serve as the backbone of quantum computing will be provided. These gates play a pivotal role in manipulating individual qubits and are essential for performing a diverse range of quantum operations and computations on current quantum hardware [43].

- **Pauli-X Gate (NOT Gate):** The Pauli-X gate is a bit-flip gate that flips the state of a qubit from $|0\rangle$ to $|1\rangle$ and vice versa. It is represented by the following matrix:

$$\boldsymbol{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$$

- **Pauli-Y Gate:** The Pauli-Y gate is a combined bit-flip and phase-flip gate. It rotates the qubit state around the y-axis of the Bloch sphere.

$$\boldsymbol{Y} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$$

- **Pauli-Z Gate:** The Pauli-Z gate is a phase-flip gate that changes the sign of the $|1\rangle$ state. The corresponding matrix representation is:

$$\boldsymbol{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

- **Hadamard Gate:** The Hadamard gate creates superposition by transforming the $|0\rangle$ state to an equal superposition of $|0\rangle$ and $|1\rangle$ states.

$$\boldsymbol{H} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Multi-Qubit Gates:

Multi-qubit gates act on multiple qubits simultaneously. They are represented by unitary matrices that operate on the joint state of the qubits. Some commonly used multi-qubit gates include:

- **CNOT Gate (Controlled-X Gate):** The CNOT gate applies an X gate (bit-flip) operation on the target qubit if the control qubit is in the $|1\rangle$ state.

$$\textbf{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

- **Toffoli Gate (Controlled-Controlled-X Gate):** The Toffoli gate is a three-qubit gate that performs an X gate operation on the target qubit if both control qubits are in the $|1\rangle$ state. Otherwise, it leaves the target qubit unchanged.

$$\textbf{Toffoli} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

4.1.2 Qutrit Gates

The quantum logic gates operating on single qutrits are typically represented by 3×3 unitary matrices. While the development of physical hardware for qutrits is an area of ongoing research, the mathematical formalization of qutrit gates plays a vital role in advancing our understanding and laying the groundwork for future qutrit-based quantum technologies.

- **Rotation Gates:** The rotation operator gates for $SU(3)$ are given by the formula:

$$\textbf{Rot}(\Theta_1, \Theta_2, \ldots, \Theta_8) = \exp\left(-i \sum_{a=1}^{8} \Theta_a \frac{\boldsymbol{\lambda}_a}{2}\right) \tag{4.1}$$

  where $\boldsymbol{\lambda}_a$ represents the ath Gell-Mann matrix, defined in the quantum information chapter. The Θ_a is a real value, with period 4π. These rotation operator gates enable arbitrary unitary transformations on qutrits [25].

- **Global Phase Shift Gate:** The global phase shift gate for qutrits is represented by the matrix:

$$\boldsymbol{Ph}(\delta) = \begin{bmatrix} e^{i\delta} & 0 & 0 \\ 0 & e^{i\delta} & 0 \\ 0 & 0 & e^{i\delta} \end{bmatrix} = \exp(i\delta\mathbb{I}) = e^{i\delta}\mathbb{I} \tag{4.2}$$

  This gate introduces a global phase factor $e^{i\delta}$, performing the mapping $|\Psi\rangle \mapsto e^{i\delta}|\Psi\rangle$. The global phase shift gate, together with the rotation operator gates, allows the expression of any single-qutrit gate in $U(3)$.

- **X Gate (Shift Gate):** The qutrit X gate is defined as:

$$\boldsymbol{X} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \tag{4.3}$$

  It performs a cyclic permutation of the qutrit states $|\mathbf{0}\rangle$, $|\mathbf{1}\rangle$, and $|\mathbf{2}\rangle$.

- **Generalized Hadamard Gate (QFT Gate):** The generalized Hadamard gate, also known as the quantum Fourier transform (QFT) gate, produces the superposition of basis states. It is an extension of the Hadamard gate to higher-dimensional quantum systems. The general form for a qudit is given by the matrix:

$$\textbf{QFT}_\textbf{d} = \frac{1}{\sqrt{d}} \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & \omega & \omega^2 & \ldots & \omega^{d-1} \\ 1 & \omega^2 & \omega^4 & \ldots & \omega^{2(d-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{d-1} & \omega^{2(d-1)} & \ldots & \omega^{(d-1)(d-1)} \end{bmatrix} \tag{4.4}$$

  where $\omega = e^{(2\pi i/d)}$. For a qutrit, the Hadamard gate is given by:

$$\boldsymbol{H} = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 1 & 1 \\ 1 & \omega & \omega^2 \\ 1 & \omega^2 & \omega \end{bmatrix} \tag{4.5}$$

- **LZ Gate:** The $\boldsymbol{LZ}$ operator is defined as $\boldsymbol{\lambda_3 + \sqrt{3}\lambda_8}$, where $\boldsymbol{\lambda_3}$ and $\boldsymbol{\lambda_8}$ represent the third and eighth Gell-Mann matrices, respectively:

$$\boldsymbol{LZ = \lambda_3 + \sqrt{3}\lambda_8} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{pmatrix} + \sqrt{3}\frac{1}{\sqrt{3}} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -2 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -2 \end{pmatrix} \tag{4.6}$$

  This operator is particularly useful for measurement, because it is diagonal in the computational basis and possesses a uniform spectrum of eigenvalues $-2, 0, 2$.

These qutrit gates serve as the fundamental building blocks for manipulating qutrits and performing quantum operations in the encoding scheme, as well as the variational circuits presented in the subsequent chapter.

4.2 Data Encoding

Data encoding on a qudit involves representing classical information in the quantum state of a system with more than two levels, but it similar to qubits. The process of data encoding is fundamental to QML algorithms, as the choice of encoding scheme can significantly impact the performance and efficiency of the model. In this thesis, angle encoding is utilized, but a brief overview of commonly used encoding techniques, such as basis encoding and amplitude encoding, will be provided as well.

4.2.1 Basis Encoding

Basis encoding in quantum systems is a method used to represent classical information in the form of quantum states, specifically computational basis states. In this encoding technique, binary data is directly mapped to the corresponding quantum states. For example, a 4-bit binary string 1001 is represented by the 4-qubit quantum state $|1001\rangle$ [47].

This means that one bit of classical information is represented by one quantum subsystem. In basis encoding, the quantum state representing a dataset can be expressed as a superposition of computational basis states:

$$|\boldsymbol{\mathcal{D}}\rangle = \frac{1}{\sqrt{M}} \sum_{m=1}^{M} |\boldsymbol{x^{(m)}}\rangle.$$

Where M is the number of samples in the dataset, and $x^{(m)}$ represents the m-th sample.

For example, for a classical dataset containing $x^{(1)} = 01$ and $x^{(2)} = 11$, the corresponding basis encoding utilizes two qubits to represent $|\boldsymbol{x^{(1)}}\rangle = |\mathbf{01}\rangle$ and $|\boldsymbol{x^{(2)}}\rangle = |\mathbf{11}\rangle$, resulting in the state:

$$|\boldsymbol{D}\rangle = \frac{1}{\sqrt{2}}|\mathbf{01}\rangle + \frac{1}{\sqrt{2}}|\mathbf{11}\rangle.$$

In this encoding scheme, the classical dataset is mapped onto the quantum state $|\boldsymbol{D}\rangle$, which is a superposition of the basis states $|\mathbf{01}\rangle$ and $|\mathbf{11}\rangle$ with equal amplitudes. It is important to note that for N bits, there are 2^N possible basis states. Given that the dataset contains M samples where $M \ll 2^N$, the basis encoding will be sparse [48].

An example of basis encoding can be found in Quantum Key Distribution (QKD), where Alice sends a qubit in state $|\mathbf{0}\rangle$ for a classical bit in state 0 and a qubit in state $|\mathbf{1}\rangle$ for a classical bit in state 1.

4.2.2 Amplitude Encoding

In the amplitude encoding technique, data is encoded into the amplitudes of a quantum state. A normalized classical N-dimensional datapoint x is represented by the amplitudes of an n-qubit quantum state $|\boldsymbol{\psi_x}\rangle$ as

$$|\boldsymbol{\psi_x}\rangle = \sum_{i=1}^{N} x_i |\boldsymbol{i}\rangle,$$

where $N = 2^n$, x_i is the i-th element of x, and $|\boldsymbol{i}\rangle$ is the i-th computational basis state. In this case, the elements x_i can have different numeric data types, such as integers or floating-point numbers [47].

For example, let's consider the four-dimensional floating-point array $x = (1.0, 0.0, -5.5, 0.0)$ that we want to encode. The first step is to normalize it, i.e., $x_{\text{norm}} = \frac{1}{\sqrt{31.25}}(1.0, 0.0, -5.5, 0.0)$. The corresponding amplitude encoding uses two qubits to represent x_{norm} as

$$|\boldsymbol{\psi_{x_{\text{norm}}}}\rangle = \frac{1}{\sqrt{31.25}} \left(|\mathbf{00}\rangle - 5.5|\mathbf{10}\rangle\right).$$

The number of amplitudes to be encoded is $N \times M$ for M N-dimensional data points. As a system of n qubits provides 2^n amplitudes, the encoding requires $n \geq \log_2(NM)$ qubits.

4.2.3 Angle Encoding

Angle encoding utilizes rotation gates to encode classical information x. The classical information determines the angles of rotation gates:

$$|\boldsymbol{x}\rangle = \bigotimes_i \boldsymbol{R(x_i)}|\mathbf{0}\rangle^{\boldsymbol{n}},$$

where $\boldsymbol{R}$ in the case of qubits can be one of $\boldsymbol{R_x}$, $\boldsymbol{R_y}$, or $\boldsymbol{R_z}$, with x, y, and z being the axis of rotation in the Bloch sphere. Typically, the number of qubits used for encoding is equal to the dimension of the vector [47].

Unlike the previous techniques, angle encoding encodes one data point at a time instead of the entire dataset. However, it offers the advantage of requiring N qubits or less, for encoding a N-dimensional data point, and a constant-depth quantum circuit, making it well-suited for current quantum hardware.

For the purpose of this thesis, in the case of qutrits, the Gell-Mann matrices are utilized as rotation operator gates for $SU(3)$ in a similar manner. This encoding technique maps a single value $x \in \mathbb{R}$ into a quantum state using the expression

$$x \mapsto \boldsymbol{Rot}_a(x)|\mathbf{0}\rangle = e^{-ix\frac{\boldsymbol{\lambda_a}}{2}}|\mathbf{0}\rangle,$$

where $\boldsymbol{\lambda_a}$ is the ath Gell-Mann matrix. Further details on this encoding technique will be provided in the next chapter.

4.2.4 Quantum Feature Maps

In the machine learning chapter, an overview was provided on the feature maps in the classical sense, where data are transformed into a new space where it may be easier to classify. Similarly, in the context of quantum computing, quantum feature maps are employed to encode classical data into quantum states. These maps operate in a Hilbert space, with the feature vectors being quantum states.

A quantum feature map $\phi : \mathcal{X} \to \mathcal{F}$ transforms classical data $\vec{\boldsymbol{x}} \in \mathcal{X}$ into a quantum state $|\boldsymbol{\phi}(\vec{\boldsymbol{x}})\rangle \in \mathcal{F}$ by way of a unitary transformation $\boldsymbol{U}(\boldsymbol{\phi}(\vec{\boldsymbol{x}}))$. The unitary transformation is typically a variational circuit whose parameters depend on the input data. The goal of the quantum feature map is to transform the classical data into a quantum state that can be used as input to a quantum or classical algorithm for classification or regression.

There is ongoing reasearch on the automatic design of quantum feature maps. Notably, in the recent paper ”Automatic design of quantum feature maps” the authors propose a technique for the automatic generation of optimal ad-hoc ansätze for classification using quantum support vector machines (QSVM). The technique is based on NSGA-II multiobjective genetic algorithms, which allow for the maximization of accuracy and minimization of the ansatz size. The paper demonstrates the validity of the technique with a practical example on a non-linear dataset and shows other application fields of the technique that reinforce the validity of the method [49].

In this thesis, the feature map is employed in the quantum kernel method to encode the features onto the qutrits. Furthermore, the feature map is used as the encoding layer within the architecture of the quantum neural network (QNN). Positioned prior to the variational layer, the encoding of classical data into quantum states is facilitated by the feature map, resulting in a quantum-inspired representation of the input.

4.3 Variational Quantum Classifier

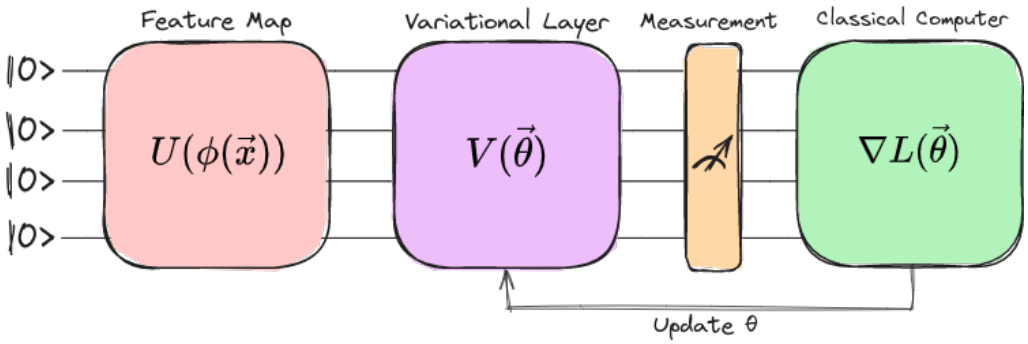


Figure 4.3: Variational Quantum Classifier

In the paper ”A Quantum Approximate Optimization Algorithm” Edward Farhi, Jeffrey Goldstone, and Sam Gutmann introduced a quantum algorithm that produces approximate solutions for combinatorial optimization problems, known as the Quantum Approximate Optimization Algorithm (QAOA) [38]. The algorithm consists of a series of quantum gates applied to a set of qubits, with the aim of finding the optimal solution that minimizes the cost function.

At the same time, the paper ”A variational eigenvalue solver on a quantum processor” by Alberto Peruzzo et al. explored a new approach to solving eigenvalue problems using a hybrid quantum-classical algorithm called the Variational Quantum Eigensolver (VQE) [39]. The authors used a photonic quantum processor to approximate the ground state of molecular hydrogen. The experiment was performed on a two-qubit system using linear-optical quantum gates, single-photon sources, and single-photon detectors, showcasing the potential of VQE for practical applications in quantum chemistry and optimization problems.

These two Variational Quantum Algorithms (VQAs) paved the way for variational training. The key idea is to parameterize a quantum circuit and optimize its parameters to minimize a cost function that measures the classification error. The input data is typically encoded into the quantum circuit using a feature map, which maps the classical data to a quantum state. Then the variational layer, is constructed using gates and operations that can be applied to qubits, and naturally extend to qudits. The measurement outcomes of the quantum circuit are then used to determine the class assignment.

A Variational Quantum Classifier (VQC) is a hybrid quantum-classical optimization algorithm in which an objective function is evaluated, and the parameters of this function are updated using classical optimization methods [40].

The VQC consists of three main components:

1. Quantum Feature Map: A quantum feature map can be represented as a unitary transformation $U(\phi(\vec{x}))$ that maps the classical input data to a quantum state, where

$\phi(\vec{x})$ is the feature map function.

2. Variational Circuit: The variational circuit can be represented as a unitary transformation $V(\vec{\theta})$ that depends on a set of trainable parameters $\vec{\theta}$. The output state of the variational layer can be then represented as: $|\psi_{out}\rangle = V(\vec{\theta})U(\phi(\vec{x}))|\psi_{in}\rangle$, where $|\psi_{in}\rangle$ is the initial state of the quantum system.

3. Measurement: The measured expectation value using an operator is interpreted as the output of a classifier. The measurement process can be represented as a function of the output state: label$(\vec{x}) = \langle\psi_{out}|O|\psi_{out}\rangle$, where O is an observable that represents the measurement operator.

In the training phase, the objective is to determine the optimal values for $\vec{\theta}$ that yield the most accurate predictions. To achieve this, the classifier employs classical optimization algorithms. It compares the predicted labels $\hat{y}$ with the actual labels y provided in the training data and computes the loss using a cost function. The classifier then iteratively adjusts the parameters of the variational circuit based on the computed loss. This iterative process continues until the cost function reaches a stable state [40].

4.3.1 Quantum Neural Networks

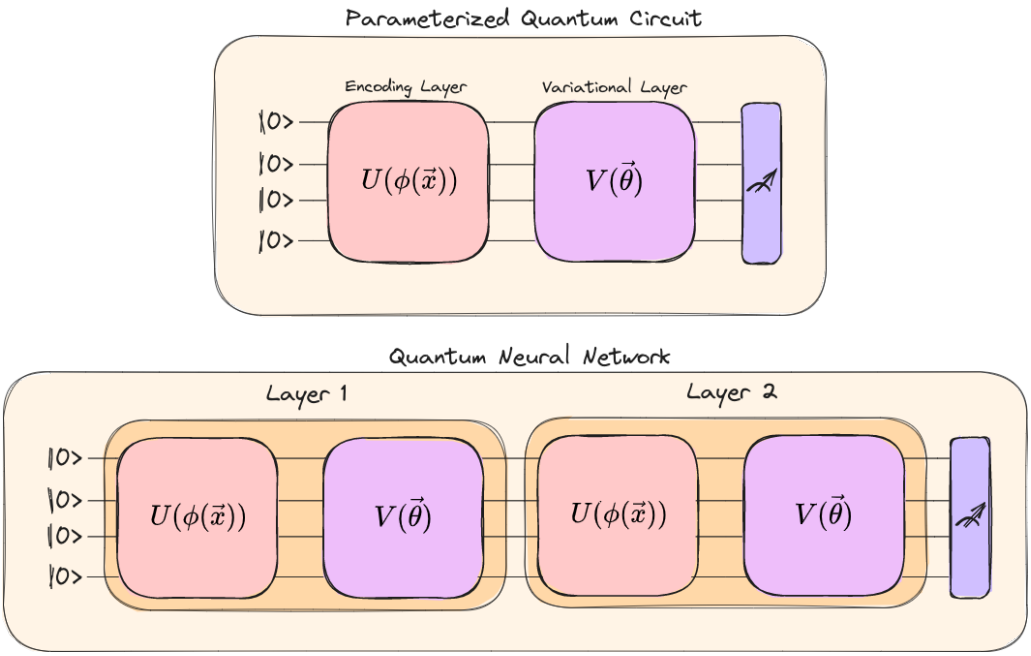


Figure 4.4: Quantum Neural Network

If a PQC can be thought of as a perceptron in the classical sense, then a Quantum Neural Network (QNN) can be viewed as a stack of two or more PQC layers. This analogy

allows for the creation of a modular framework that can be adjusted and easily modified based on the specific problem and its underlying geometry. While there are various architectures and variations of QNNs that have been investigated, this thesis will adopt the straightforward approach of using a stack of PQC layers.

The concept has gained significant attention, particularly with the paper ”Data re-uploading for a universal quantum classifier” by Adrián Pérez-Salinas, Alba Cervera-Lierta, Elies Gil-Fuster, and José I. Latorre. Their work showcases the construction of powerful quantum classifiers using multiple re-uploading layers, even with a single qubit [44].

The proposed approach organizes a quantum circuit as a sequence of data re-uploading and single-qubit processing units. This design enables the quantum classifier to handle complex data by accommodating multiple input dimensions and output categories. Additionally, the strategy's efficiency is enhanced when extended to multiple qubits, leveraging entanglement to expand the involved superpositions during the classification process.

In a QNN, each PQC layer performs a transformation on the input quantum state, resulting in a new state that serves as the input to the subsequent PQC layer. The parameters of the PQCs are trained to optimize a given objective function, typically through gradient-based optimization methods.

The depth of a QNN refers to the number of stacked PQC layers. As observed in practical implementations, more complex problems often require an increase in the depth of the QNN to capture intricate features and relationships within the data. Increasing the depth allows for a more expressive representation of the problem, enabling the QNN to learn and classify complex patterns.

The choice of QNN depth is a trade-off between computational resources and performance. While deeper QNNs may provide better representation power, they also require more qubits, longer gate sequences, and increased optimization complexity. Therefore, the depth of a QNN should be carefully considered based on the available quantum hardware and the problem's requirements.

By exploiting the modular nature of QNNs and adjusting the depth of the network, the architectures are flexible and can adapt to various problem domains, which is the reason this approach is employed in this thesis. While there are similarities between the quantum and classical neural networks in terms of representation, training, and universal approximation, it's important to note that they operate on fundamentally different principles, with the former leveraging quantum mechanics for computation.

4.3.2 Optimization

In the paper ”Quantum Circuit Learning” Kosuke Mitarai, Makoto Negoro, Masahiro Kitagawa, and Keisuke Fujii proposed a hybrid approach that combined classical optimization techniques with quantum circuits, allowing the circuits to learn and improve their performance on specific tasks [45].

The authors showcased a quantum circuit with adjustable parameters designed to represent a trainable model. The parameters of the circuit were optimized using classical optimization algorithms to minimize a given cost function and demonstrated the effectiveness of their approach on various tasks, including image classification and generative modeling.

The paper also introduced the concept of parameter-shift rule, which enables efficient computation of the gradient of a quantum circuit's output with respect to its parameters. This gradient information is crucial for the optimization process and allows for the application of gradient-based optimization algorithms to train the quantum circuit.

At the same time the paper titled "Evaluating Analytic Gradients on Quantum Hardware" by Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac, and Nathan Killoran addressed the challenge of efficiently computing analytic gradients for quantum circuits on quantum hardware. They introduced a method for evaluating analytic gradients on quantum hardware, which involves measuring a set of expectation values associated with different shift parameters [46].

These expectation values can be used to calculate the gradient of a quantum circuit's output with respect to its parameters. The authors also discussed practical challenges, such as noise and limited circuit depth, and propose strategies to mitigate these issues. The paper includes experimental results demonstrating the feasibility and effectiveness of computing analytic gradients on quantum hardware.

The findings of these papers had significant implications for QML and optimization algorithms. The ability to efficiently compute gradients on quantum hardware opened up new possibilities for training quantum models and optimizing their performance.

4.3.3 Parameter Shift Rule

Gradient-based methods aim to identify an optimal solution by finding a point where the gradient is equal to zero. The parameter shift rule is a technique used in QML to estimate gradients and allows for gradient calculations without the need for complex differentiation of quantum circuits.

Let $f(\theta)$ be a quantum circuit with parameterized gates θ. To estimate the gradient of an expectation value $E(\theta)$, the parameter shift rule is given by:

$$\frac{\partial E}{\partial \theta} = \frac{f(\theta + \frac{\pi}{2}) - f(\theta - \frac{\pi}{2})}{2}$$

where θ represents the parameter being differentiated, and $\frac{\pi}{2}$ is a shift applied to the parameter. This rule provides an approximation of the gradient using two evaluations of the circuit with shifted parameters.

4.4 Vanishing Gradients

The problem of vanishing gradients arises when gradients become extremely small or even zero during training of deep neural networks. This hinders the ability of gradient-based optimization methods to learn and make progress towards the optimal solution. As a result, the gradient tends to either explode or vanish in earlier layers, making it difficult to optimize the network.

The vanishing gradient problem is also present in QML, where it can make optimization of parameterized quantum circuits difficult. The instability of the gradient in QML is a result of the composition of unitary operators and the repeated multiplication of parameters in the optimization algorithm. This can lead to the gradient becoming extremely small or even zero, making it difficult to optimize the circuit and make progress towards the optimal solution.

The problem of barren plateaus is related to the idea of gradient concentration, where the gradient of a PQC with respect to its parameters concentrates around zero for certain circuits. Gradient concentration has been observed in circuits with a large number of qubits and/or layers, and with certain types of randomly generated parameterizations. The three phenomena, gradient concentration, exponential concentration of cost around the mean, and exponential narrowness of minima, occur together, meaning that if one is present, the other two are also present.

In the paper ”Barren plateaus in quantum neural network training” [50] the authors discovered and proved that for a wide class of reasonable parameterized quantum circuits, the probability that the gradient along any reasonable direction is non-zero to some fixed precision is exponentially small as a function of the number of qubits.

Analytically and numerically, it has been observed that for many random quantum circuits, observable expected values converge to their averages over Hilbert space and gradients converge to zero. This insight sheds light on the geometry of quantum circuits and their relevance to hybrid quantum-classical algorithms. It also suggests that randomly initialized circuits of sufficient depth may have limited utility in these algorithms.

There are several approaches to mitigating gradient concentration. One approach is to use a different parameterization for the circuit, such as a hardware-efficient ansatz or a circuit with more structure that does not suffer from gradient concentration. Another approach is to use a modified optimization algorithm that takes into account the structure of the cost landscape, such as the quantum natural gradient optimizer [51].

In this thesis quasi-Newton methods, like the Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) [32], are also trialed to combat such landscapes in the qubit circuits. Finally, when dealing with larger systems, strategies that can also be effective are the utilization of structured initial guesses [52], the adoption of layerwise learning [53] and the use of local instead of global observables [54].

4.4.1 Local Observables

In the paper titled ”Cost function dependent barren plateaus in shallow parameterized quantum circuits” by Cerezo et al. [54], the authors explore the relationship between locality, trainability, and the performance of VQAs. The concepts are illustrated through large-scale simulations of a quantum autoencoder implementation.

The authors consider a PQC, which is constructed using an alternating layered ansatz composed of blocks that form local 2-designs. A 2-design represents a set of quantum states that approximates the uniform distribution over the entire Hilbert space to the second order. A ”local 2-design” refers to a set of states that approximates the uniform distribution over a smaller, local Hilbert space.

Two theorems related to gradient scaling in VQAs are presented in this paper:

Theorem 1. *Let C be a cost function defined in terms of global observables, and let $V(\theta)$ be a shallow PQC. Then, for a fixed depth of $V(\theta)$, the expected gradient of C with respect to any of its parameters θ_i will vanish exponentially with the number of qubits n, i.e.,*

$$|\partial C/\partial \theta_i| \leq \exp(-n/\kappa),$$

where κ is a constant depending on the depth of $V(\theta)$.

Theorem 2. *Let C be a cost function defined in terms of local observables, and let $V(\theta)$ be a parameterized quantum circuit of depth $O(\log n)$ composed of blocks forming local 2-designs. Then, for any fixed depth of $V(\theta)$, the expected gradient of C with respect to any of its parameters θ_i will vanish at worst polynomially with the number of qubits n, i.e.,*

$$|\partial C/\partial \theta_i| \leq \text{poly}(n),$$

where the polynomial bound depends on the depth of $V(\theta)$.

The main takeaway from this research is that the choice of cost function significantly impacts the trainability and performance of VQAs. When the cost function is defined using global observables involving all qubits, barren plateaus may arise, impeding the optimization process. In contrast, employing local observables that focus on a subset of qubits can help avoid barren plateaus and enhance the trainability of the algorithm.

4.4.2 Layerwise Learning

Layerwise learning is a training strategy for QNNs that aims to address the problem of vanishing gradients and make better use of the resources provided by NISQ devices. This strategy was proposed in the paper ”Layerwise Learning for Quantum Neural Networks” and the main idea is to incrementally grow the circuit depth during optimization and update only subsets of parameters at each training step [53].

The paper demonstrates the effectiveness of the layerwise learning approach through an image-classification task on handwritten digits. The results show that layerwise learning attains an 8% lower generalization error on average compared to standard learning schemes for training quantum circuits of the same size. Moreover, the percentage of runs that reach lower test errors is up to 40% larger compared to training the full circuit, which is susceptible to creeping onto a plateau during training.

The structure of the layers to be stacked is defined first, with the same layout for all layers. One layer consists of random gates on each qubit initialized with zero, and two-qubit gates connect qubits to enable generation of entanglement. The number of start layers, the number of epochs to train each set of layers, the number of new layers added in each step, and the maximum number of layers trained at once are specified as hyperparameters.

The training process consists of two main phases. In the first phase, the circuit is successively grown by adding new layers and training subsets of layers for a specified number of epochs. This step avoids initializing on a plateau and provides a good starting point in the optimization landscape. In the second phase, the circuit is split into partitions, and the parameters of each partition are trained alternately while the parameters of the inactive partitions are frozen. This phase is repeated until the loss converges.

The authors argue that the properties of their algorithm, such as the ability to handle low-depth circuits, limited parameter updates per step, and the larger magnitude of gradients compared to training the full circuit, contribute to its efficacy on NISQ devices. By considering the impact of sampling noise and utilizing this training strategy, the algorithm aims to provide more reliable results in the presence of quantum hardware limitations.

4.4.3 Initialisation

In the paper ”An initialization strategy for addressing barren plateaus in parameterized quantum circuits” by Edward Grant, Leonard Wossnig, Mateusz Ostaszewski, and Marcello Benedetti, the authors propose an initialization strategy that involves randomly selecting some initial parameter values and then choosing the remaining values so that the circuit is a sequence of shallow unitary blocks, each evaluating to the identity. This approach limits the effective depth of the circuits used to calculate the first parameter update, preventing them from being stuck in a barren plateau at the start of training [52].

Empirical evidence is provided in the paper to demonstrate the effectiveness of this initialization strategy in training variational quantum eigensolvers (VQE) and quantum neural networks (QNN). The authors show that the gradient variance does not decrease exponentially during training, and the model does not get stuck in a barren plateau.

However, the authors also note that more work is needed to assess the impact of input states and data encoding methods on the initialization strategy. They mention that there is a problem-dependent trade-off to be analyzed, and other potential strategies for avoiding barren plateaus, such as layer-wise training, regularization, and imposing structural constraints on the ansatz.

It is important to note that the paper also mentions certain limitations of their initialization strategy. For instance, the full exploration of the impact of input states and data encoding methods on the strategy is not addressed. Moreover, the trade-off between circuit depth and the ability to avoid barren plateaus is dependent on the specific problem at hand. Other potential strategies, such as layer-wise training, regularization, and the imposition of structural constraints on the ansatz, may be more suitable for certain datasets.

In this thesis, a modified approach of this strategy was employed. The optimization process began with multiple iterations to select an initial favorable point, thereby preventing the model from getting trapped in a barren plateau during the initial stages of training. This technique, combined with the strategies of local observables and layerwise learning outlined before, ensured the avoidance of plateau initialization and facilitated a promising starting point within the optimization landscape. These methods proved sufficient for the classification tasks investigated within the scope of this thesis.

4.5 Information Capacity

Understanding the limits and capabilities of a model is a key aspect in the field of machine learning. In classical models, the capacity is often quantified by considering factors such as the number of parameters or employing concepts like VC dimension, defined as the maximum number of points that the hypothesis class can shatter. These measures provide insights into the model's capacity, to ensure its suitability for a given learning task and gain a deeper understanding of its potential limitations.

Computational learning theory, a subfield of artificial intelligence, focuses on elucidating the efficacy of learning various functions. It seeks to ascertain the minimum amount of data necessary to facilitate effective learning, and explores the balance between the complexity of a learned model and its capacity to generalize accurately to previously unseen data. Additionally, the field delves into devising mechanisms to guarantee the convergence of learning algorithms towards the desired solution.

A measure of a model's ability to fit data is the PAC-Bayes framework, which combines the concepts of Probably Approximately Correct (PAC) learning and Bayesian learning. A model endowed with higher capacity can fit a wider range of functions, but it may also overfit the data. The PAC-Bayes bound offers a way to strike a balance between the model's fit to the training data and its inherent complexity.

In the paper titled ”A PAC-Bayesian Approach to Spectrally-Normalized Margin Bounds for Neural Networks,” the authors extend the PAC-Bayesian framework to neural networks with the incorporation of a spectrally-normalized margin-based bound. This bound quantifies the margin between accurately classified examples and the decision boundary established by the neural network [56].

The VC dimension serves as another measure of capacity for a neural network, representing the capacity of a hypothesis class, which encompasses all possible functions that a learning algorithm can choose from. Formally, if there exists a set of points that can

be perfectly labeled by all possible binary assignments using the functions or classifiers in the hypothesis class, this maximum number of points is the VC dimension. Vapnik and Chervonenkis introduced the VC dimension in their seminal work ”On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities” [57].

The authors demonstrated that when the VC dimension of a hypothesis class is finite, the empirical risk converges uniformly to the true risk. The VC dimension provides a crucial foundation for understanding the trade-off between the complexity of a hypothesis class and its ability to generalize. A higher VC dimension implies a greater capacity to fit the training data, but it also raises the risk of overfitting and poor generalization.

These classical machine learning measures to assess a model’s capacity may not directly apply to quantum models. There is significant interest in exploring whether quantum computers can offer an advantage, but quantifying the information capacity of a QML model is not as straightforward. The unique properties of quantum systems require distinct approaches for evaluating their capacity.

In the paper ”Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms” Sukin Sim, Peter D. Johnson, and Alan Aspuru-Guzik introduce the concepts of expressibility and entangling capability as metrics to discern between different parameterized quantum circuits. Expressibility refers to the extent to which the circuit’s hypothesis space covers the Hilbert space, while entangling capability pertains to its capacity to produce entangled states [58].

A general unitary operation can access the entire Hilbert space, but it does not guarantee high information capacity. Variational circuits with high expressibility, can have flat optimization landscapes, making it challenging to identify the correct search directions for minimizing the loss function. Consequently, training such models becomes more intricate and prone to overfitting.

Finally, in the papers ”The Power of Quantum Neural Networks” [59] and ”Effective Dimension of Machine Learning Models” [60] Amira Abbas et al. introduce the concept of effective dimension as a capacity measure for QML models. They argue that traditional capacity measures are insufficient in explaining crucial observed characteristics in practice. The effective dimension captures both the model’s ability to fit different functions and provides bounds on generalization error.

The effective dimension measures the proportion of parameters actively used in a neural network. The authors demonstrate that certain QNNs exhibit faster training compared to classical models due to their optimization landscapes, which have a more evenly spread Fisher information spectrum. This property contributes to improved resilience and efficiency during training.

These findings underscore the potential advantages of QML, particularly in the context of QNNs. However, there are still open questions regarding the reasons behind the high effective dimensions of QNNs and whether these results extend to more general classes of quantum models.

5. IMPLEMENTATION AND BENCHMARKS

5.1 Datasets

In order to gain insights into the fitting process of the model, initially, binary classification problems in two dimensions, were chosen, namely Circles, XOR, and Moons. By choosing these specific datasets, it becomes possible to visually evaluate the model's performance and its ability to separate classes effectively. This assessment is facilitated by plotting the decision boundaries, enabling a clear visualization of the model's separation capacity.

Subsequently, the evaluation expanded to include multiclass classification problems with high-dimensional data, aiming to assess the model's capabilities in handling complex classification tasks. In the case of qutrits, multiclass classification becomes particularly relevant as each distinguishable state of the qutrit can be assigned to a separate class. For this purpose datasets with three labels were selected, namely Wine Cultivars, Iris, Seed, and Glass.

- The Wine Cultivars Dataset contains chemical analysis results of wines from three different cultivars in Italy.
- The Iris Dataset comprises measurements of sepal length, sepal width, petal length, and petal width for three species of iris flowers.
- The Seed Dataset provides geometric measurements of kernels belonging to three varieties of wheat.
- The Glass Dataset includes chemical attributes about three types of glass.

These datasets provide diverse and challenging scenarios for evaluating the performance of the models in multiclass classification tasks. Detailed information and links to their UCI repositories can be found in the appendix C.

By incorporating real-world datasets with varying numbers of features, the effectiveness and generalizability of the models can be comprehensively assessed. Additionally, the differences in feature sets among them offers an opportunity to test the adaptability of the models' feature maps.

5.2 Gell-Mann Feature Map

In the case of qubits, several feature maps can be used to encode classical data into quantum states. The choice of feature map depends on the specific problem at hand and the desired representation of the data. The Pauli Feature Map, for instance, uses the Pauli matrices (I, X, Y, Z) and is primarily employed in quantum kernel methods.

With the goal of enhancing encoding capabilities, the Gell-Mann feature map is introduced, which draws upon the mathematical framework provided by the special unitary group SU(3). The utilization of the proposed Gell-Mann feature map enables the encoding of information within an 8-dimensional space. This empowers the quantum system to capture and process significantly larger amounts of data even within a single qutrit.

The Gell-Mann feature map, as well as the variational layer of the QNN, use the rotation operator gates for $SU(3)$:

$$\textbf{Rot}(w_1, w_2, \ldots, w_8) = \exp\left(-i \sum_{a=1}^{8} w_a \frac{\boldsymbol{\lambda}_a}{2}\right) \tag{5.1}$$

Where, $\boldsymbol{\lambda}_a$ is the ath Gell-Mann matrix, and w_a is a real value, with period 4π.

It is important to clarify that in the context of this study, the Gell-Mann feature map does not pertain to a specific arrangement of gates, but rather to the utilization of Gell-Mann rotation operators, for the encoding process.

5.3 Encoding and Variational Layers

Empirical findings have identified a configuration that outperforms others for the encoding and variational layers of the PQCs investigated in this thesis. However, to optimize performance for certain problems, adjustments such as incorporating extra gates or increasing the number of qutrits were required. These modifications will be discussed in detail for each architecture, while the underlying core structure of the layers is presented below:

- **Encoding Layer:** This layer employs the first four Gell-Mann matrices to encode the input vector $\vec{\boldsymbol{x}}$ into a quantum state:

$$|\boldsymbol{\psi}_{\vec{\boldsymbol{x}}}\rangle = \exp\left[i \sum_{j=1}^{4} x_j \cdot \boldsymbol{\lambda}_j\right] |0\rangle \tag{5.2}$$

  The exponential term with the Gell-Mann matrices acts as a rotation operator, where x_j represents the components of the input vector $\vec{\boldsymbol{x}}$.

- **Variational Layer:** The last four Gell-Mann matrices are utilized in this layer to introduce variational parameters $\vec{\boldsymbol{w}}$ into the quantum state. The variational layer is defined as follows:

$$|\boldsymbol{\psi}_{\vec{\boldsymbol{x}},\vec{\boldsymbol{w}}}\rangle = \exp\left[i \sum_{j=5}^{8} w_{j-4} \cdot \boldsymbol{\lambda}_j\right] |\boldsymbol{\psi}_{\vec{\boldsymbol{x}}}\rangle \tag{5.3}$$

  The variational weights $\vec{\boldsymbol{w}} = (w_0, w_1, w_2, w_3)$ are the parameters to be optimized.

5.4 Quantum Kernel

5.4.1 Overview

In this method, a quantum device is employed to encode the features of two specific data points through the application of the Gell-Mann feature map. Subsequently, the estimation of their inner product and the kernel is used as input for a classical support vector machine. Kernel-based training bypasses the processing parts of common variational circuits and only depends on the data encoding.

This approach presents significant interest in situations where the evaluation of kernels using classical methods proves to be computationally infeasible due to exponential growth in runtime as a function of the input dimension. The introduction of this approach can be attributed to Maria Schuld et al., as outlined in their paper titled Quantum Machine Learning in Feature Hilbert Spaces [61].

They expressed the quantum feature map $\phi(\boldsymbol{x})$ as simply the process that encodes classical data $\boldsymbol{x}$ into a quantum state. If $\phi(\boldsymbol{x})$ is a feature map for data $\boldsymbol{x}$, $|0\rangle$ is the initial state of the system, and $\boldsymbol{U}(\boldsymbol{x})$ a unitary operation dependent on the data, then the quantum state after encoding is given by:

$$|\boldsymbol{\phi}(\boldsymbol{x})\rangle = \boldsymbol{U}(\boldsymbol{x})|0\rangle \tag{5.4}$$

The quantum kernel function $K(x, x')$ then measures the similarity between two quantum states corresponding to data x and x' and defined as their inner product:

$$K(x, x') = |\langle \boldsymbol{\phi}(\boldsymbol{x})|\boldsymbol{\phi}(\boldsymbol{x'})\rangle|^2 \tag{5.5}$$

This approach offers the capability to replace many near-term quantum variational models with a support vector machine. This kernel-based approach guarantees finding solutions that are equally good or even better in some cases than those trained using variational circuits [62].

5.4.2 Architecture

Within the context of this thesis, different combinations of rotation gates and encodings were explored. While some datasets required alternative gate arrangements, for the majority of datasets, the following two kernel architectures proved the most effective. The first and simplest kernel involved using a single qutrit.

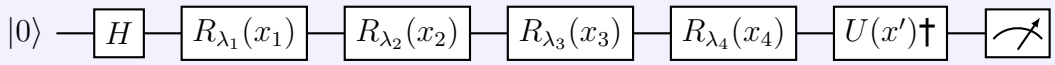


Figure 5.1: A Single Qutrit Kernel

To encode the features onto one qutrit, the qutrit is initially placed in a superposition state by applying the Hadamard operator, allowing it to exist as a combination of its potential states. Subsequently, the features are encoded onto the qutrit using the first four Gell-Mann matrices.

The second data point is encoded using the conjugate transpose of these gates, symbolized with $\boldsymbol{U(x')^\dagger}$ on this and subsequent circuits. The resulting states are then used to compute the quantum kernel by taking the inner product between them. This quantum kernel can then be passed to a classical SVM classifier.

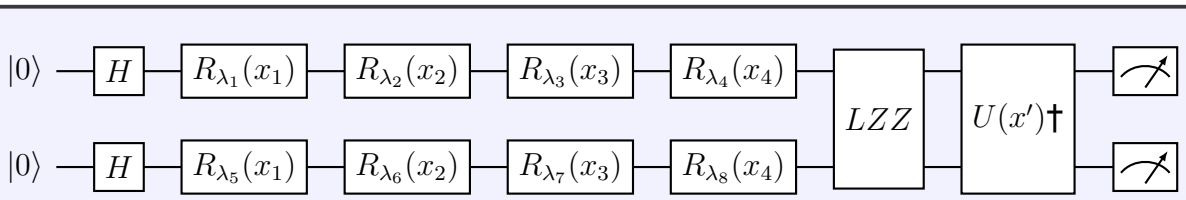


Figure 5.2: A Two Qutrit Kernel

The second kernel discovered during experimentation is particularly intriguing, as it involves the utilization of two qutrits placed in a superposition state using the Hadamard operator. The features are then encoded on the first qutrit using the first four Gell-Mann matrices, and on the second qutrit using the last four Gell-Mann matrices. Subsequently, an entangling operation is performed between them by applying the $\boldsymbol{LZZ}$ gate, defined as follows:

$$\boldsymbol{LZZ} = \exp\left[-i\boldsymbol{LZ} \otimes \boldsymbol{LZ}\right] \tag{5.6}$$

Where, the matrix $\boldsymbol{LZ}$ is defined as $\boldsymbol{LZ} = \boldsymbol{\lambda}_3 + \sqrt{3}\boldsymbol{\lambda}_8$, as previously discussed.

The two diagrams provided above illustrate the encoding process for four features of the Iris dataset. When datasets contained more than four features, stacked layers were employed in a similar arrangement of rotation gates to accommodate the additional features. For a more comprehensive understanding of the implementation, refer to the detailed Python code provided in Appendix B.1.

5.4.3 Results

A quantum kernel alone cannot be used to make predictions on a dataset, but only serves as a tool to measure the overlap between two data points. For the SVM, sci-kit learn's Support Vector Classifier (SVC) was employed, which requires a kernel function that takes two sets of data points. To validate each kernel, a check was performed to ensure that evaluating the kernel of a data point with itself returned 1, since the simulation is noiseless.

Different architectures were trialed, but in the majority of cases, utilizing two qutrits with the kernel method mentioned in the previous section yielded the best results, effectively capturing the complexity of the data. To evaluate the performance of the classifier, the percentage of correctly classified data points in the dataset was measured.

5.4.3.1 Binary Classification

For the binary classification task, problems in two dimensions were selected, specifically Circles, XOR, and Moons, for visualisation of its separation capacity. However, since the feature map described above requires at least four dimensions, the two features were uploaded once more in the subsequent Gell-Mann rotations.

Separation capacity is the ability of a model to distinguish between the different classes in a dataset. It can be seen as the complexity of decision boundaries that the model can learn. A model with a high separation capacity can learn complex decision boundaries that can effectively separate classes that are non-linearly separable, while a model with a low separation capacity might only be able to learn linear decision boundaries.

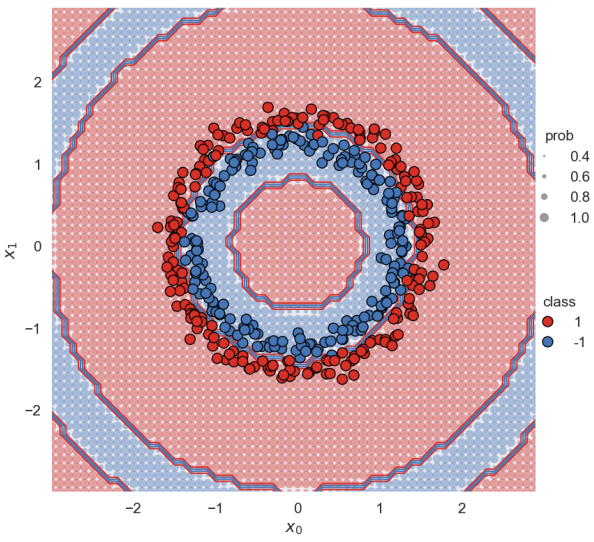


Figure 5.3: Circles Decision Boundaries

A decision boundary is a hypersurface that partitions the underlying vector space into one set for each class. To gain insight into the model's fitting process, the decision boundaries were plotted using the data visualization functions from Tim von Hahn's blog [63].

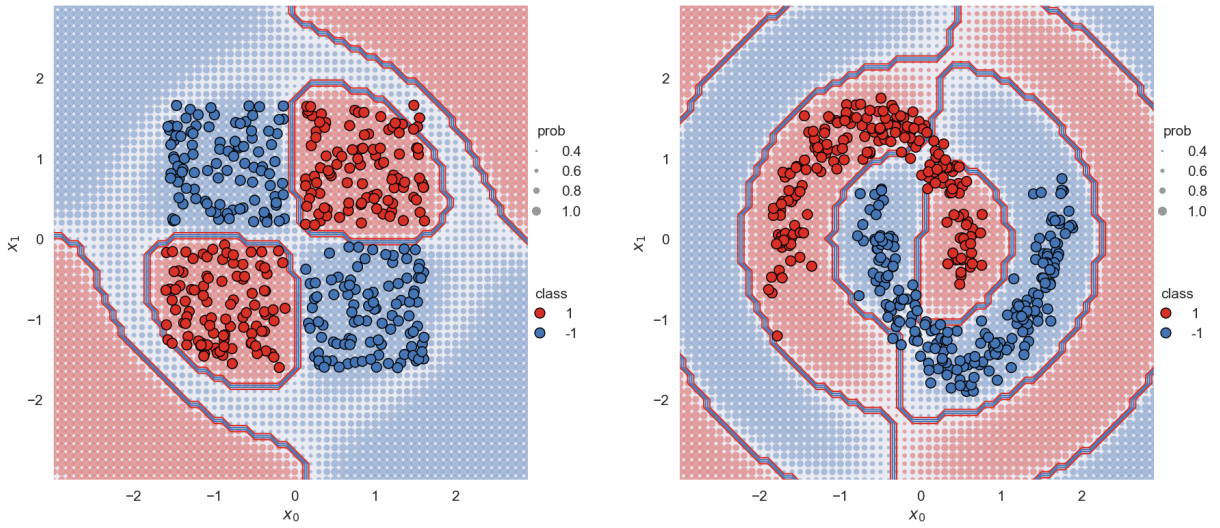
In addition to illustrating the decision boundaries between classes, these plots provide a visual representation of the probability assigned to a data point belonging to a specific class. This is achieved by adjusting the size of the dots, with larger dots indicating a higher likelihood of the data point being classified into that particular class.

This can be observed in Figure 5.3, where the quantum kernel combined with the SVM achieves an accuracy of 100% on the Cir-

cles dataset, showcasing its ability to accurately classify all instances in this dataset.

This probability of a prediction is often referred to as confidence, and it is distinct from accuracy. Confidence reflects the model's certainty in its prediction, while accuracy measures the overall correctness of the model's predictions across a dataset.

In classifiers like logistic regression, the confidence can be directly interpreted as a probability. For classifiers like SVM, the confidence might be derived from the distance of a data point to the decision boundary. It is important to note that a model can have low confidence in correct predictions and still maintain high accuracy.



(a) XOR Accuracy: 99.17% (b) MOONS Accuracy: 96.67%

Figure 5.4: SVM Decision Boundaries

The benchmark results, presented in Figure 5.4, demonstrate for the XOR dataset, the model achieves a high accuracy of 99.17%, and an accuracy of 96.67% on the Moons dataset. It is important to note that the accuracy was evaluated on the test set, while the decision boundaries were on the entire dataset.

The decision boundaries of all three binary problems indicate that the outer points in the datasets can be accurately classified. The majority of the data instances fall within the correct class for circles, and for the other two slightly more complex datasets, the performance remains nearly perfect. Furthermore, no strong artifacts that would raise concerns about the model's reliability are observed.

5.4.3.2 Multiclass Classification

For the multiclass classification tasks, the datasets consisted of three classes and four or more dimensions. A single layer of the feature map utilized four rotations for a qutrit, as illustrated in Figures 5.1 and 5.2. For datasets with higher than four dimensions, stacked

layers were utilized. This approach involved breaking down the datasets and using multiple layers of encoding for each set of features.

The results below were obtained using the two-qutrit kernel presented in the architecture section, except for the Wine dataset, where Hadamard gates were not utilized. Since the SVM method relies on scikit-learn's SVC, the decision boundaries and support vectors are determined by the input data. Therefore, conducting multiple benchmarks is unnecessary, since running it with identical inputs consistently yields the same outcomes.

Table 5.1: Metrics of the Quantum Kernel's performance on multiclass datasets

	IRIS	WINE	GLASS	SEED
Recall	90.00%	83.97%	90.48%	88.10%
Precision	92.31%	83.51%	90.43%	89.56%
Accuracy	90.00%	83.33%	90.24%	88.10%
F1-score	89.77%	83.42%	90.21%	87.93%

The model achieved high recall scores across all datasets, ranging from 83.97% to 90.48%. This indicates that the model successfully identifies a significant portion of true positive instances within each class. The precision scores vary, with values between 83.51% and 92.31%, which shows that the model's ability to accurately classify instances within each predicted positive class varies across the multiclass datasets.

The accuracy scores range from 83.33% to 90.24%, achieving an overall good level of correctness in its predictions across the majority of datasets. The F1-scores, which provides a balanced evaluation of the model's performance, range from 83.42% to 89.77% and reflect its ability to balance precision and recall for each class.

The obtained metrics demonstrate that the quantum kernel effectively maps the features in the qutrits, enabling the SVM to accurately predict the target class. These results hold promise for introducing a variational layer in place of the classical approach within the QNN framework. This transition would simply shift the focus towards optimization and the search for optimal parameters as the primary challenge.

5.5 Quantum Neural Network

5.5.1 Overview

For the quantum kernel a hybrid quantum-classical method was employed with the help of NumPy and scikit-learn, but in the case of the neural network, with the introduction of the variational layer and the increase in the number of operations involved, PyTorch was chosen. PyTorch facilitates the explicit definition of the model's architecture and is efficient in handling complex operations.

The QNN class is implemented as a module that encapsulates the functionality of the model and inherits from PyTorch's `Module` class. This design choice allows for smooth

integration with PyTorch's optimization algorithms and cost functions. After evaluating various optimizers, `RMSprop` was chosen for training. Additionally, considering that the datasets are multiclass, the `cross-entropy` loss function was selected, making the model well-equipped to handle binary and multiclass problems effectively.

To address the challenge of barren plateaus, a combination of strategies was employed, as described in the previous chapter. Primarily, local observables were use to combat vanishing gradients and improve the trainability of the algorithm, as discussed in the paper "Cost function dependent barren plateaus in shallow parameterized quantum circuits" [54].

Additionally, leveraging the concepts presented in the paper titled "An initialization strategy for addressing barren plateaus in parameterized quantum circuits" [52], if the accuracy of an initial guess fell below a specific threshold, an alternative point in the parameter space was selected. Incorporating this technique, provided a favorable starting point within the optimization landscape and made training feasible.

5.5.2 Architecture

In the QNN, each layer applies a transformation to the quantum state, generating a new state as input for the subsequent layer. The parameters of each layer are optimized through training. This approach based on the paper "Data re-uploading for a universal quantum classifier" [44], allows the quantum classifier to handle complex data with multiple input dimensions and output categories.

The QNN class, implemented as a PyTorch module, allows for the specification of the number of layers and features during initialization. This design choice enables a modular framework that can be readily adjusted and modified during hyperparameter tuning to align with the underlying geometry of the problem.

Selecting the depth of the QNN involves a trade-off between computational resources and performance. Deeper QNNs may provide enhanced representation power, but they also introduce increased optimization complexity due to the requirement of either more qutrits or longer gate sequences. An increase in depth carries, also, the potential risk of overfitting, necessitating a delicate balance to be struck in model design.

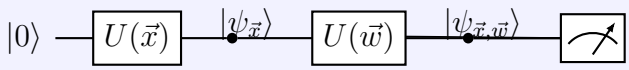


Figure 5.5: A Single QNN Layer composed of Encoding and Variational Layers

As shown in Figure 5.5, a single layer of the QNN comprises an encoding layer and a variational layer. The encoding layer, denoted as $\boldsymbol{U}(\vec{\boldsymbol{x}})$, follows a feature map similar to the kernel.

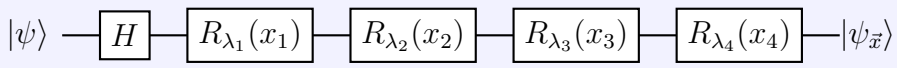


Figure 5.6: Encoding Layer

As illustrated in Figure 5.6, the initial step involves preparing the qutrit in a superposition state by applying the Hadamard operator. Subsequently, the features are encoded onto the qutrit using the first four Gell-Mann matrices. In the following variational layer, as depicted in Figure 5.7, the remaining four Gell-Mann matrices are utilized, and the weights of the QNN's parameters are optimized.

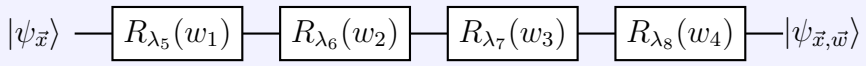


Figure 5.7: Variational Layer

Figure 5.5 represents a single layer of the QNN. Additional layers are stacked at the end, similar to how perceptrons are added in classical machine learning neural networks, as depicted in the circuit shown in Figure 5.8.

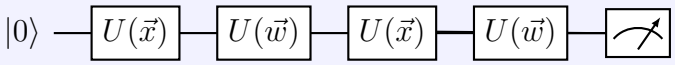


Figure 5.8: Two Stacked QNN Layers

The effectiveness of increasing the number of qutrits varied across datasets. In some cases, utilizing multiple qutrits led to improved performance, while in others, a single qutrit was sufficient. Furthermore, the impact of entangling operations depended on the underlying structure of the dataset, emphasizing its data-specific nature. However, it is important to note that using more than two qutrits was unnecessary, as this capacity effectively captured the information present in most datasets.

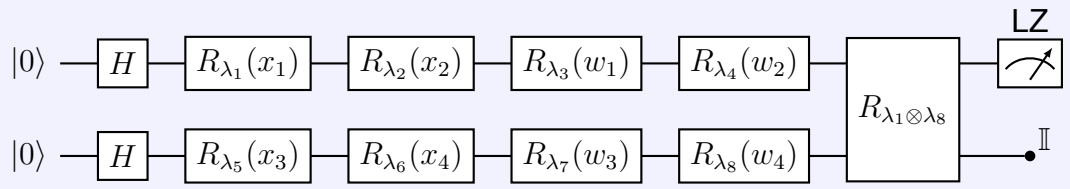


Figure 5.9: Two Qutrits With Encoding and Variational Layers

The architecture presented in Figure 5.9, resembling the feature map for two qutrits in the quantum kernel section, employs the Gell-Mann matrices in a similar manner. However, a notable difference arises in the application of an entangling operator:

$$\boldsymbol{R_{\lambda_1 \otimes \lambda_8}} = \exp[i \boldsymbol{\lambda_1} \otimes \boldsymbol{\lambda_8}] \tag{5.7}$$

Here, the tensor product of the first and eighth Gell-Mann matrices is utilized. Additionally, for the QNN a local measurement approach is used, employing the $\boldsymbol{LZ}$ operator.

For the sake of simplicity in the training process, the probabilities of the quantum state are utilized instead of running the circuit multiple times and obtaining the expectation values. This is made possible by accessing the quantum state directly through the Python simulation. For each architecture that was tested, a distinct class was implemented, and the corresponding code can be found in the GitHub repository. For reference, the implementation of the single qutrit QNN architecture is provided in Appendix B.2.

5.5.3 Results

During the training process, a thorough search was conducted to explore various combinations of learning rates, weight decay, and momentum factors in the optimizers that supported this. The objective was to fine-tune the model's hyperparameters and discover the optimal configuration that maximized convergence speed.

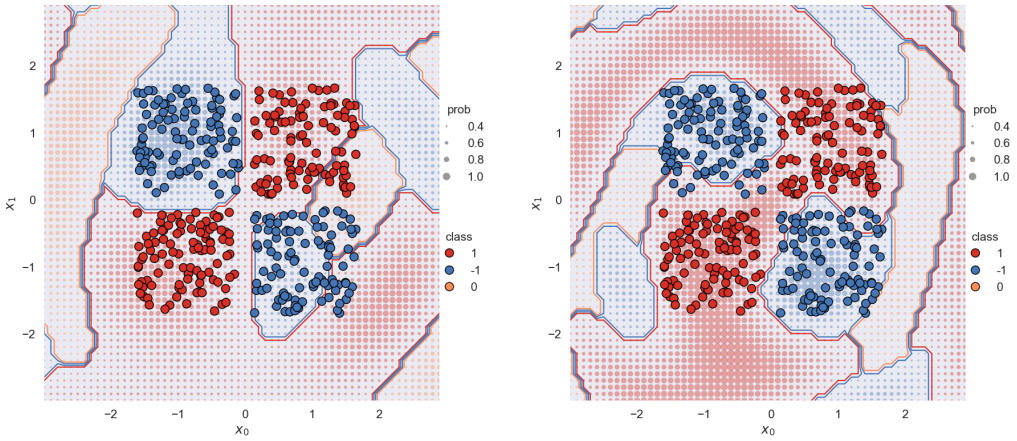
In the preliminary experiments, the parameter shift rule was employed, but it was later replaced with the `RMSProp` optimizer, which demonstrated superior performance. `RMSProp` was used to update the parameters of the QNN, with an initial `learning rate` set to 0.001, a decay factor `gamma` of 0.9, and an `epsilon` value of 1e-8 for numerical stability.

The input features were appropriately scaled, using scikit-learn's `StandardScaler` preprocessing class, which transforms the data by removing the mean and scaling to unit variance. The standard score of a sample x is calculated as: $z = \frac{x-u}{s}$, where u is the mean of the training sample, and s is the standard deviation of the training samples. Standardization is a necessary requirement for neural networks, as they might behave poorly if the individual features do not resemble standard normally distributed data.

The `Dataset` and `DataLoader` classes were utilized, enabling efficient preprocessing, seamless integration of data loading with the model training process, and leveraging the capabilities of the broader PyTorch ecosystem. Various batch sizes were tested to strike the optimal balance between training efficiency and model performance. Due to the small size of the datasets, a small `batch size` of 4 was chosen, since increasing the batch size beyond this point proved to be impractical.

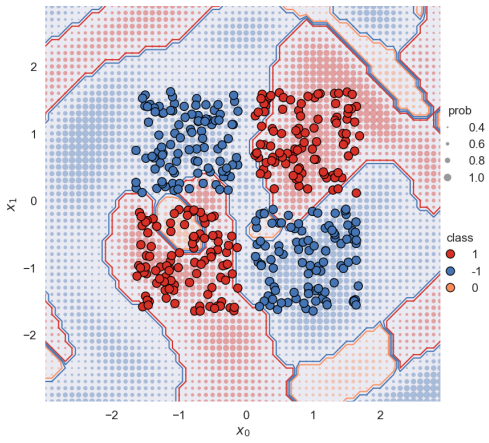
PyTorch's `cross entropy` loss function was utilized to compute the loss between the input logits and target, as it operates on the probabilities of the quantum state. The computed loss is then used to perform the backward pass, and update the model's parameters.

5.5.3.1 Binary Classification



(a) 1 Layer Accuracy: 85.3%

(b) 2 Layers Accuracy: 91.25%



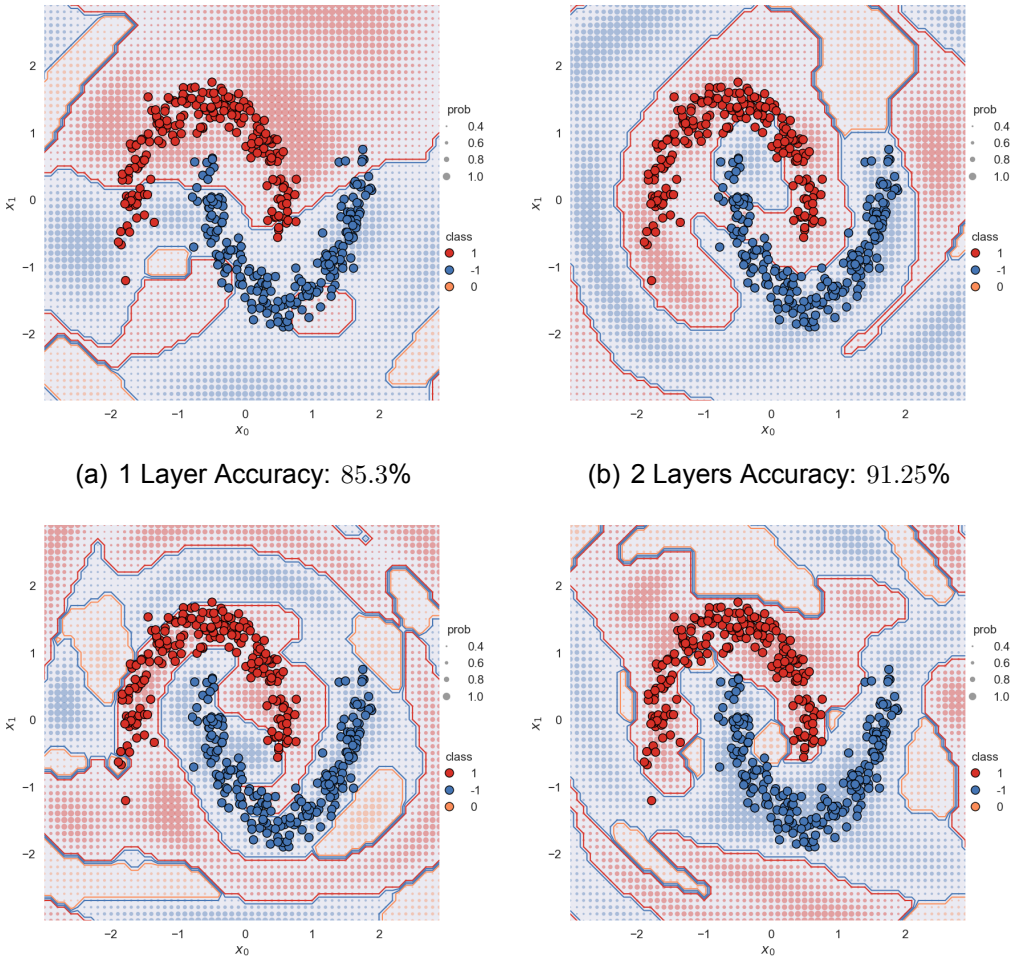
(c) 3 Layers Accuracy: 100%

Figure 5.10: QNN Decision Boundaries XOR Dataset

In the initial evaluation of the QNN's performance and to compare with the quantum kernel's separation capacity, the XOR and Moons datasets were employed. Notably, there exists a distinction between neural networks and support vector machines in terms of their approach to learning decision boundaries.

When considering artificial neural networks or perceptrons, the separation capacity is determined by the number of hidden layers present in the network. Specifically, the absence of hidden layers restricts the network to learning only linear problems. However, the inclusion of one hidden layer enables the network to learn any continuous function, thus facilitating the adoption of arbitrary decision boundaries.

This behavior was also detected in the case of the QNN, where an increase in depth resulted in the emergence of increasingly complex and intricate decision boundaries. This can be clearly observed in Figure 5.10, which demonstrates the progression from 1 to 3 layers. Additionally, as the QNN learns to effectively separate the classes in the XOR dataset, the accuracy naturally improves, progressing from 85.3% to a perfect 100%.



(a) 1 Layer Accuracy: 85.3%

(b) 2 Layers Accuracy: 91.25%

(c) 3 Layers Accuracy: 92.5%

(d) 4 Layers Accuracy: 100%

Figure 5.11: QNN Decision Boundaries Moons Dataset

The same phenomenon is even more evident in Figure 5.11, where the presence of one layer results in a simple linear separation, followed by increasing intricacy as additional layers are introduced. In this case, the decision boundaries progressively improve to better align with the interleaving shapes present in the Moons dataset.

The accuracy of the QNN on the Moons dataset exhibits a steady improvement, starting from 85.3% with one layer and reaching 100% with four layers. Similar to the quantum kernel's measurement in the previous section, the accuracy presented below the plots is determined using the test set, while the decision boundaries are plotted over the entire dataset, providing a comprehensive visualization.

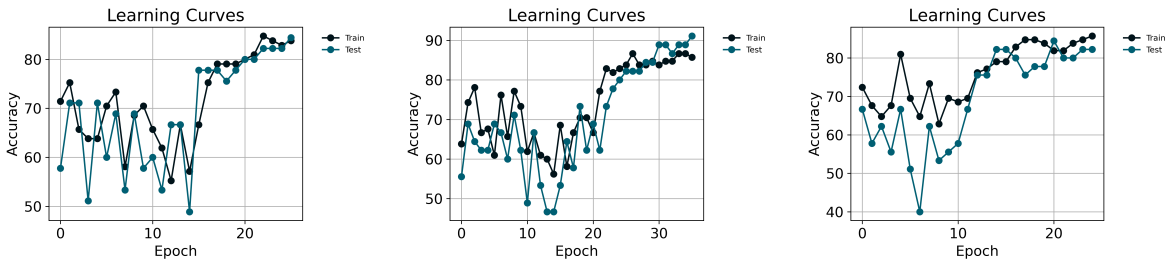
For both the XOR and Moons datasets, the QNN employed a single qutrit architecture as described in the previous section and in the code of Appendix B.2. These binary classification problems demonstrated that using more than one qutrit was unnecessary for effective learning. However, since each layer requires a minimum of four dimensions, the two-dimensional features were ”re-uploaded” in the subsequent Gell-Mann rotations.

5.5.3.2 Multiclass Classification

For the multiclass classification tasks, learning curves are plotted to visually represent the model's performance and learning progress over time. These curves reveal important trends such as convergence and potential overfitting or underfitting.

The Iris dataset consists of four features, while the remaining datasets have higher dimensions. To enhance computational efficiency while maintaining consistent encoding, PCA was employed to reduce the dimensions to four in all datasets, except for the Glass dataset. In the Glass dataset, the original dimensions were retained and encoded in the subsequent layer, since this yielded better accuracy.

The following results were obtained using the single qutrit architecture. Although the quantum kernel achieved better accuracy with two qutrits compared to the single qutrit in multiclass problems, the QNN achieved similar performance with both architectures. Considering the longer training time required by the two qutrit architecture, which is expected due to the increased number of operations, the decision was made to utilize the single qutrit architecture for all tasks.

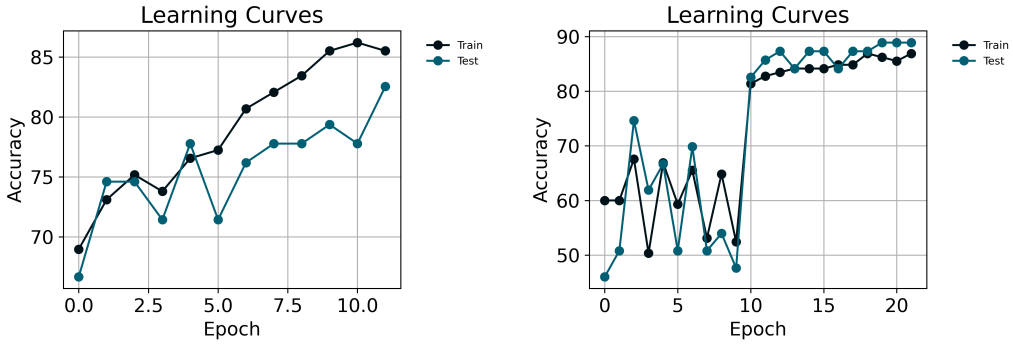


(a) QNN of 3 Layers (b) QNN of 4 Layers (c) QNN of 5 Layers

Figure 5.12: IRIS Dataset Learning Curves

In the Iris dataset, the model achieved its optimal performance with four layers, as demonstrated by the learning curves presented in Figure 5.12, having a 91.11% accuracy on the test set. The increased depth of the QNN allowed the model to effectively capture the intricate features and complex patterns within the data.

However, it is worth noting that beyond the point of four layers, the model encountered overfitting to the training set, indicated by the black line, and a divergence for the accuracy of the test set, indicated by the green line.

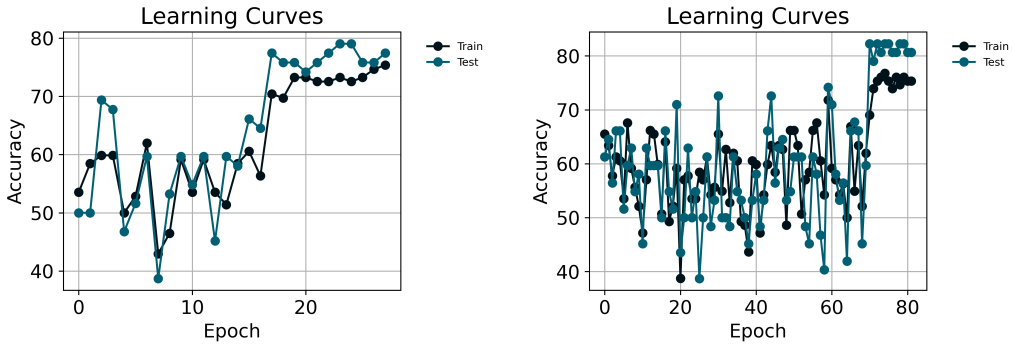


(a) QNN of 1 Layer (b) QNN of 3 Layers

Figure 5.13: SEED Dataset Learning Curves

For the Seed dataset, the QNN required fewer epochs to train effectively. As illustrated in Figure 5.13, the model achieved its peak performance with three layers, attaining an accuracy of 88.89%, a macro average F1-Score of 88.96%, and similar performance across Recall and Precision metrics.

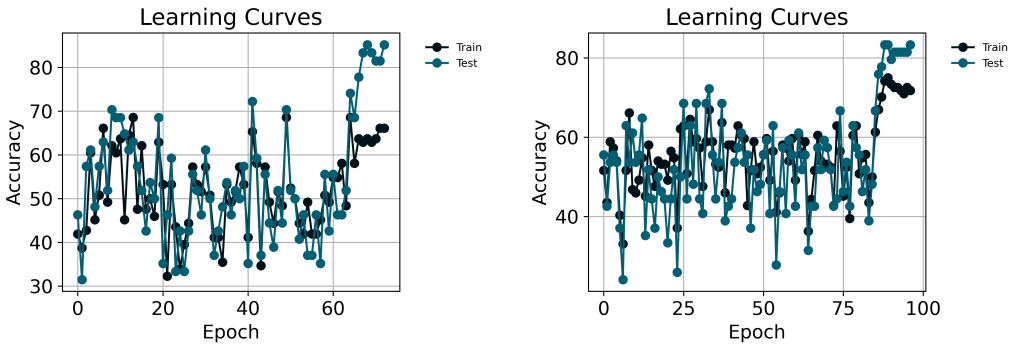
Additionally, the learning curves indicate a healthier training process compared to the one-layer model, as there was no divergence between the training and test sets, suggesting it can generalize to unseen data.



(a) QNN of 1 Layer (b) QNN of 2 Layers

Figure 5.14: GLASS Dataset Learning Curves

As mentioned earlier, the Glass dataset utilized all available features, necessitating an increase in the number of layers and parameters for the variational layer. Consequently, more epochs were required to achieve a good initial starting point and optimize the model. This prolonged convergence is particularly noticeable in the case of the two-layer architecture, where it took 70 epochs to find a favorable state.



(a) QNN of 2 Layers (b) QNN of 3 Layers

Figure 5.15: WINE Dataset Learning Curves

Lastly, for the Wine dataset, the model necessitates an extended number of epochs to identify an optimal initial starting point. Interestingly, the model demonstrates comparable accuracy with both two and three layers. Specifically, the accuracy achieved with two layers is recorded at 85.19%, and exhibiting similar performance across all metrics.

The observed initial instability in the learning curves can be attributed to the initialization method, whereby the model randomly selects a point in the hyperparameter space. While this approach has demonstrated usefulness, it is worth considering alternative strategies such as employing a quantum natural gradient or leveraging Fisher information. These methods have the potential to enhance the training landscape, mitigating the need for ad hoc measures, and provide more robust gradient techniques.

Table 5.2: Metrics of the QNN's performance on multiclass datasets

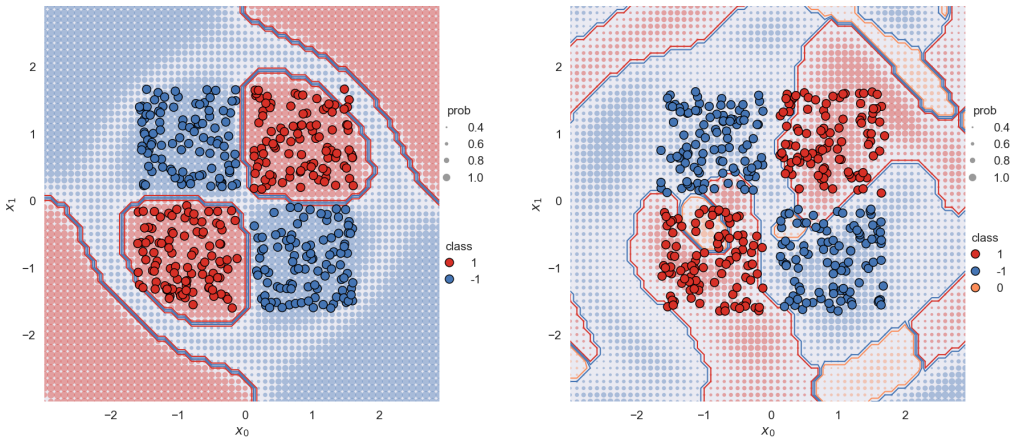
	IRIS	WINE	GLASS	SEED
Layers	4	2	2	3
Recall	91.11%	84.23%	82.54%	88.89%
Precision	91.23%	84.61%	82.96%	89.10%
Accuracy	91.11%	85.19%	82.54%	88.89%
F1-score	91.10%	84.37%	82.35%	88.96%

Table 5.2 presents an overview of the QNN's performance, showcasing the obtained metrics along with the corresponding number of layers for each dataset. Multiple benchmarks were conducted, resulting in comparable outcomes.

5.6 Comparative Analysis

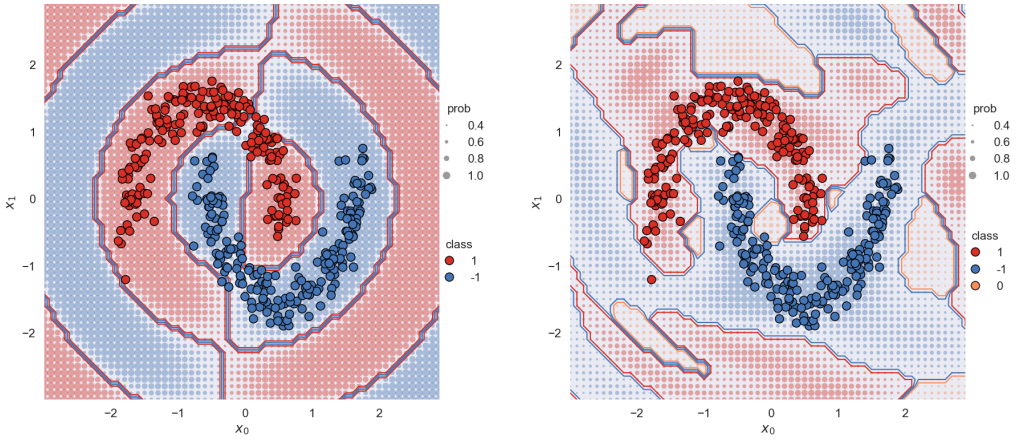
5.6.1 Quantum Kernel vs. Quantum Neural Network

The visual representation of decision boundaries through plotting proved to be insightful in understanding the geometric structure of binary classification problems when working with two-dimensional data. It shed light on the model's capacity and the discernible patterns present in the data, highlighting notable distinctions between the fitting processes of the quantum kernel and quantum neural network.



(a) Quantum Kernel SVM (b) Quantum Neural Network with 3 Layers

Figure 5.16: XOR Decision Boundaries Comparison



(a) Quantum Kernel SVM (b) Quantum Neural Network with 4 Layers

Figure 5.17: Moons Decision Boundaries Comparison

The decision boundaries derived from SVM exhibit a distinct and easily interpretable geo-

metric separation in both the Moons and XOR datasets. In contrast, the decision boundaries produced by the QNN are characterized by intricate and nuanced patterns that appear to closely align with the data points, particularly evident in the case of the Moons dataset. This behavior aligns with the inherent capability of neural networks to capture intricate and non-linear relationships, enabling them to model complex patterns within the data more effectively.

This distinguishing characteristic positions quantum neural networks as promising tools for addressing real-world problems where datasets lack explicit geometric structures. Their potential lies in their ability to capture and exploit the inherent complexity and non-linearity embedded within the data.

Table 5.3: Comparison of Metrics for Quantum Kernel and QNN on Multiclass Datasets

	IRIS	WINE	GLASS	SEED
Recall (Kernel)	90.00%	83.97%	90.48%	88.10%
Precision (Kernel)	92.31%	83.51%	90.43%	89.56%
Accuracy (Kernel)	90.00%	83.33%	90.24%	88.10%
F1-score (Kernel)	89.77%	83.42%	90.21%	87.93%
Recall (QNN)	91.11%	84.23%	82.54%	88.89%
Precision (QNN)	91.23%	84.61%	82.96%	89.10%
Accuracy (QNN)	91.11%	85.19%	82.54%	88.89%
F1-score (QNN)	91.10%	84.37%	82.35%	88.96%

The multiclass results provided further evidence of the QNN's prowess in uncovering and leveraging intricate patterns within real-world datasets. In all evaluated datasets, the QNN consistently outperformed the quantum kernel across all metrics, except for the Glass dataset. This outcome underscores the QNN's superior performance and its ability to effectively model complex relationships and patterns present in the data.

It is important to note that, in the case of quantum kernels, utilizing the original features of the dataset led to superior outcomes compared to applying PCA and using the reduced features obtained from it. However, interestingly, the opposite trend was observed in the context of the QNN.

This difference in performance can be attributed to several factors, which require further investigation and analysis. The quantum kernels might be more effective when the data distribution aligns well with the quantum feature space. Naturally, it might be the case that the increased depth of the network and number of parameters simply required a different optimization approach.

The size of the dataset can also play a role in the observed results. If the dataset is relatively small, which is the case with these datasets, the quantum kernels might be more resilient to overfitting and better able to capture the underlying patterns in the raw features, whereas the QNN could struggle due to the increased model complexity.

5.6.2 Qutrit Circuit vs. Qubit Circuit vs. Classical SVM

In order to evaluate the performance of the qutrit QNN, a comparison will be conducted using a qubit circuit and a classical SVM. To implement the qubit circuit, Qiskit's VQC will be utilized, which constructs a parameterized quantum circuit for data classification.

Scikit-learn's SVC was selected as a representative classical machine learning method due to its versatility and suitability as a baseline classifier for model evaluation. By conducting a comparative analysis of the performance between quantum algorithms and scikit-learn's SVC, the strengths and weaknesses of quantum approaches in tackling multiclass classification problems can be assessed.

The VQC in Qiskit offers support for various loss functions and optimizers, allowing flexibility in the training process. It also provides the option for warm start, which means that weights from a previous fit can be used to initialize the next fit. Furthermore, the VQC can be executed using either a QuantumInstance or a Backend to run the quantum circuits.

The VQC takes as parameters the number of qubits, feature map, ansatz, loss function, and optimizer. For this study, it was decided to use four qubits, matching the number of features in the Iris dataset. In the other datasets, PCA was employed to reduce the number of features to four. From optimizers the L_BFGS_B and COBYLA were used.

The ZZFeatureMap was selected as the feature map, a commonly used data encoding method. For the ansatz, the RealAmplitudes circuit was employed, which consists of alternating layers of rotations and entanglements.

Both the feature map and the ansatz can be repeated a certain number of times, which is adjustable and serves as an argument in the circuit's construction. This provides flexibility in designing the architecture of the circuit. Additional details and the implementation can be found in Appendix B.3.

Table 5.4: Comparison of the QNN's performance to Qubit VQC and SVM

	IRIS	WINE	GLASS	SEED
Classical SVM Accuracy	100%	91.7%	100%	90.48%
Qutrit Kernel Accuracy	90.00%	83.33%	90.24%	88.10%
Qutrit QNN Accuracy	91.11%	85.19%	82.54%	88.89%
Qubit VQC Accuracy	86.67%	80.78%	68.98%	73.81%
Qutrit QNN Layers	4	2	2	3
Qutrit QNN Parameters	16	8	8	12
Qubit VQC ZZ Repetitions	1	1	2	2
Qubit VQC Parameters	12	16	20	24

The classical SVM achieves perfect accuracy on the IRIS and GLASS datasets, while for the WINE and SEED datasets, 91.7% and 90.48% respectively. This showcases the strong performance in multiclass classification tasks. The qutrit QNN achieves a competitive accuracy, demonstrating its potential in multiclass classification tasks, but it is still

lower comparable to the classical SVM. It is worth noting that quantum methods, although promising, is expected to have lower outright accuracy compared to classical methods.

In comparison, the quantum kernel with SVM achieves slightly lower accuracy across all datasets and the qubit VQC's performance falls behind all of them. The circuit employing four qubits achieves 86.67% accuracy on the IRIS dataset, 80.78% on the WINE dataset, 68.98% on the GLASS dataset, and 73.81% on the SEED dataset. This suggests the need for further improvement, such as increasing the number of qubits or parameters, in order to enhance the performance.

In terms of model complexity, the qutrit QNN has 16 parameters for the IRIS dataset, 8 parameters for the WINE and GLASS datasets, and 12 parameters for the SEED dataset. On the other hand, the qubit VQC uses 1 repetition for the ZZ feature map on the IRIS and WINE datasets, and 2 repetitions on the GLASS and SEED datasets. The number of parameters in the ansatz of the VQC is 12 for the IRIS dataset, 16 for the WINE dataset, 20 for the GLASS dataset, and 24 for the SEED dataset.

These results highlight that even with fewer parameters, the single qutrit circuit outperforms the qubit VQC and can capture more information through its rotations. Additionally, the qubit circuit employs 4 qubits compared to the 1 qutrit of the QNN. Although increasing the number of qubits or parameters would likely improve the VQC to achieve comparable results, it would also increase the cost. The objective here was to demonstrate the performance of the VQC with a similar number of parameters.

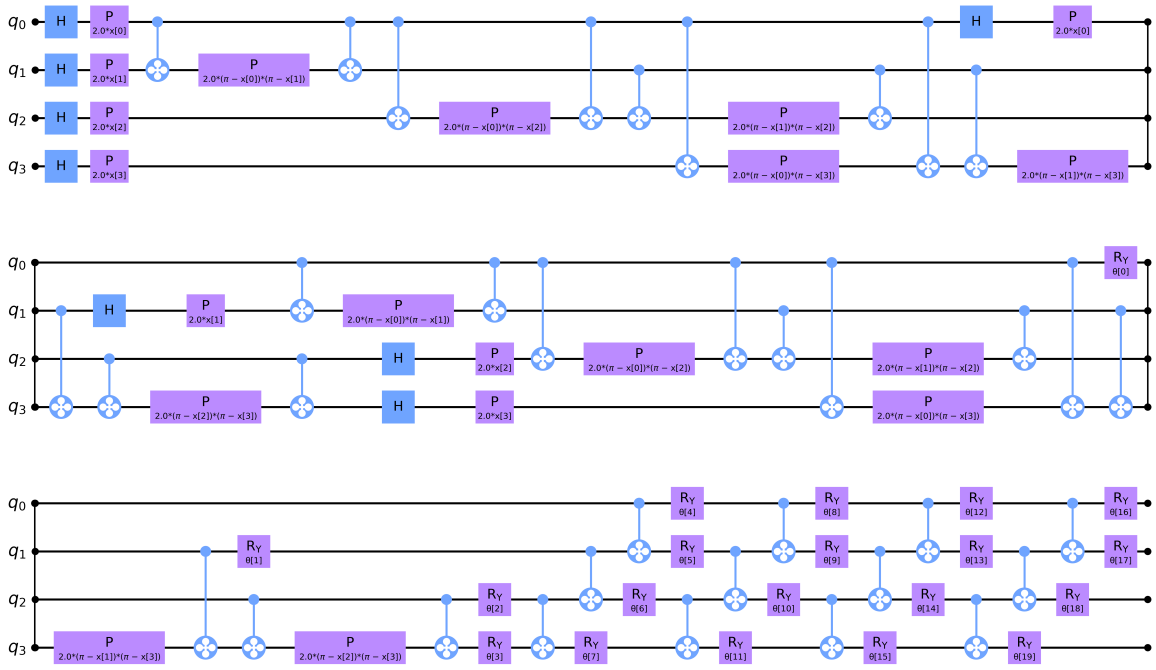


Figure 5.18: Qubit VQC utilizing the ZZ Feature Map and Real Amplitudes Ansatz

A clear observation of the complexity and number of gates required for the qubit VQC can

be seen in Figure 5.18, specifically in the case of the Glass dataset which presented the most challenging scenario. This circuit yielded the highest accuracy for the dataset by utilizing 2 repetitions of the ZZ feature map and 4 repetitions of the ansatz, resulting in a total of 20 trainable parameters. This circuit also took longer to train, and underperformed even with different optimization algorithms.

It is worth noting that potential improvements to the qubit VQC involve increasing the repetitions of the ZZ feature map and the number of rotations or entangling operations. However, such modifications would introduce an unfair comparison to the shallow depth of the single qutrit QNN, which could also achieve higher accuracy by increasing its depth.

The key point remains that despite having fewer parameters, the expanded computational space of the qutrit and the utilization of the Gell-Mann feature map are capable of capturing information that surpasses the capabilities of a circuit with four qubits.

These insights significantly contribute to advancing our understanding of the capabilities and limitations of quantum methods in the context of multiclass classification problems. Among the evaluated quantum approaches, the single qutrit QNN demonstrated competitive accuracy, indicating its potential effectiveness in tackling complex classification tasks. The quantum kernel approach showcased comparable performance to the qutrit QNN, but utilized two qutrits, which may provide an avenue for exploring higher-dimensional quantum feature spaces.

On the other hand, the qubit VQC fell behind, unable to achieve the same level of accuracy as its quantum counterparts. The classical SVM consistently outperformed all quantum methods, affirming the superiority of classical machine learning models in this particular scenario. These findings, although expected to some extent, underscore the necessity for continued efforts to challenge classical machine learning models.

6. CONCLUSIONS AND FUTURE WORK

Several key observations emerge from the evaluation of the Gell-Mann encoding employed as a quantum kernel for the SVM and as a feature map for the QNN. The obtained results demonstrate remarkable promise, showcasing the feasibility of training by incorporating this encoding. It effectively expands the capacity and potential of quantum circuits that utilize qutrits.

However, it is essential to interpret these results with caution as they are based on simulations rather than physical hardware implementation, where noise significantly affects performance. Additionally, although the feasibility of training is demonstrated using the Gell-Mann feature map, classical methods continue to outperform their quantum counterparts. This outcome is expected, though, in the early stages of QML, and cases where the opposite holds true should be approached with skepticism.

Furthermore, the effectiveness of the proposed methods heavily relies on the problem space and the underlying structure of the dataset. As heuristic methods, the selection of a different feature map or variations in the weights and layers of the variational model can entirely reshape the loss landscape.

Now, when comparing the QNN, a parameterized circuit with multiple layers, to the quantum kernel method, a notable observation emerged. It became evident that unless the variational circuit has a parameter count lower than the dimensions of the training data, kernel methods were more efficient. While the QNN exhibited slightly better accuracy in the examined problems, the improvement was marginal in most cases. Consequently, careful consideration should be given to this aspect.

Another intriguing distinction observed in the quantum kernel method was that encoding the complete features of the dataset into the feature map using stacked layers yielded superior results compared to utilizing PCA and encoding the resulting principal components. Interestingly, the opposite trend was observed for the QNN. Several factors may contribute to this phenomenon.

This discrepancy can potentially be attributed to the increase in complexity and operations. As observed in the multiclass classification examples, surpassing a certain threshold of network layers rendered the model excessively challenging to train effectively. A similar observation holds true when increasing the depth of the encoding layer in the quantum kernel method.

Furthermore, an important observation from the experimental findings was that utilizing a single layer in most cases posed a significant obstacle during the training process for the QNN. The model encountered difficulty in surpassing the 50% accuracy threshold, underscoring the inherent impracticability of capturing the requisite complexity of the data with just a single encoding and variational layer.

On the other hand, increasing the number of layers not only prolonged the training time for each epoch, but also extended the overall duration until the model converged. This out-

come is expected, as introducing additional layers entails a higher number of parameters, requiring increased computational resources and optimization time.

However, despite the increased training time, the inclusion of more layers in the QNN architecture presented an opportunity to capture richer and more intricate representations of the data, leading to improved model performance. This trade-off between training time and enhanced expressiveness necessitates careful consideration when designing the QNN architecture.

Moreover, to overcome the challenges encountered when increasing the number of layers, exploring alternative network architectures could yield fruitful results. Another avenue worth exploring is the utilization of transfer learning techniques to enhance the initial stages of training, by initializing the model's weights with values derived from a pre-trained model. Considering that the weights often resided within similar ranges throughout the experiments, this approach can be particularly valuable when working with limited data, as it allows the model to leverage the data used in pre-training.

Finally, addressing challenges such as barren plateaus necessitated exploring strategies like structured initial guesses, local cost functions, and integrating correlations between layers. These strategies demonstrated effectiveness in the examined cases, but their heuristic nature requires further investigation. One promising avenue for optimization, that wasn't explored, is the utilization of a QNG that leverages the Fubini-Study metric tensor to construct a quantum analog of natural gradient descent.

Looking ahead, future research in this area should prioritize the establishment of rigorous proofs, theoretical frameworks, and mathematical conditions for trainability. The demonstrated feasibility of qutrit quantum models using the Gell-Mann encoding underscores the promise of these methods. By developing a robust foundation that guides the optimization of these algorithms, we can fully harness the power of quantum computing to effectively address complex classification problems. Ultimately, this advancement will contribute to bridging the gap between classical and QML approaches.

ABBREVIATIONS - ACRONYMS

PQC	Parameterized Quantum Circuit
NISQ	Noisy Intermediate-Scale Quantum
QRAM	Quantum Random Access Memory
QML	Quantum Machine Learning
QNN	Quantum Neural Network
QFT	Quantum Fourier Transform
VQC	Variational Quantum Classifier
VQA	Variational Quantum Algorithm
SVM	Support Vector Machine
RBF	Radial Basis Function
CNN	Convolutional Neural Network
QFI	Quantum Fisher Information
QNG	Quantum Natural Gradient
QKD	Quantum Key Distribution
QSVM	Quantum Support Vector Machine
QAOA	Quantum Approximate Optimization Algorithm
PCA	Principal Component Analysis
QPCA	Quantum Principal Component Analysis
CNOT	Controlled-NOT

APPENDIX A. QUANTUM HARDWARE

A.1 NISQ

The Noisy Intermediate-Scale Quantum (NISQ) era represents the current state of quantum computing, featuring quantum processors with up to 1000 qubits. These processors are not yet advanced enough to achieve fault tolerance or reach quantum supremacy. Being sensitive to their environment, they are referred to as ”noisy” and are susceptible to quantum decoherence. Continuous quantum error correction remains beyond their current capabilities. The term NISQ was coined in 2018 by John Preskill [64].

NISQ algorithms are specifically designed for the processors of this era, such as the VQE and the QAOA. While these algorithms utilize NISQ devices, certain computations are offloaded to classical processors. They have demonstrated success in quantum chemistry and show potential for applications in physics, material science, data science, cryptography, biology, and finance. However, the inherent noise in NISQ devices introduces errors into quantum computations, which can quickly render results invalid for complex calculations. Thus, error mitigation techniques are often necessary to ensure accurate outcomes.

The ultimate goal of the field of quantum computing is to construct large-scale, error-corrected quantum computers. However, the path to achieving this goal lies in the development and improvement of NISQ devices. Researchers are actively working on enhancing the quality of qubits, extending coherence times, and devising error correction codes and techniques.

While NISQ devices are not yet suitable for fault-tolerant operations or capable of achieving quantum supremacy, they still serve as valuable tools for exploring phenomena in many-body quantum physics and other potential applications. The NISQ era should be recognized as a significant step towards more powerful quantum technologies in the future.

Quantum technologists continue to strive for more accurate quantum gates and, eventually, fully fault-tolerant quantum computing. The realization of beyond-NISQ devices would enable the implementation of algorithms like Shor’s algorithm for breaking RSA encryption on very large numbers.

A.2 Near-term Machines

The current state of quantum hardware is limited, making even the simplest textbook quantum algorithms impractical. Achieving large-scale quantum computations requires fault-tolerant quantum computers with a significantly higher number of qubits and gates.

Recent advancements in quantum hardware focus on error mitigation, noise control, and the development of modular quantum computers. IBM’s ”Eagle” Quantum Processing

Unit (QPU) incorporates error mitigation strategies like Zero Noise Extrapolation (ZNE) to control noise and improve qubit performance. IBM's Heron processor represents a move towards modular quantum computers, enabling direct connectivity between processors.

The chips will be connected with conventional electronics, which will not be able to maintain the ”quantumness” of information as it moves from processor to processor. The aim is to link these chips with quantum-friendly fiber-optic or microwave connections, which will open the path towards distributed, large-scale quantum computers with as many as a million connected qubits.

Besides IBM, companies like Baidu and Alibaba are also making strides in quantum computing. Baidu offers access to a 10-qubit processor and has designed a 36-qubit superconducting quantum chip. Fujitsu is working with Riken to provide access to Japan's first quantum computer with 64 superconducting qubits, while in 2020, the Indian government pledged to spend $1.12 billion on quantum technologies, and for innovative “qudit” photonics computing.

Xanadu, a Canadian quantum technology company, is developing fault-tolerant quantum computers using silicon photonics. They offer cloud access to their gate-based photonic quantum computers, with plans to expand their processor capabilities. Xanadu is also leading the development of PennyLane, an open-source software framework for quantum machine learning, chemistry, and computing.

Google's Quantum AI team aims to build a fully error-corrected quantum computer with millions of qubits. They focus on building an error-corrected quantum bit (qubit) prototype and have invested in the development of their quantum hardware. Google's quantum campus in Santa Barbara, California, serves as a dedicated facility for quantum data centers and quantum processor manufacturing.

These efforts promise significant improvements in quantum computing capabilities, overcoming the limitations imposed by noise and paving the way for practical quantum computations. Quantum computing has made significant progress in recent years and the focus is shifting from increasing the number of quantum bits or ”qubits” to developing practical hardware.

A.3 Qudit Hardware

Qudits offer a promising avenue for achieving significant improvements in quantum circuit decomposition and cost reductions for essential quantum algorithms, paving the way for scalable quantum computation. Notably, optical quantum states based on entangled photons lie at the core of quantum information science, serving as pivotal building blocks for solving questions in fundamental physics and are at the heart of quantum information science. Integrated photonics has become a leading platform for the compact, cost-efficient, and stable generation and processing of non-classical optical states.

The paper ”On-chip Generation of High-Dimensional Entangled Quantum States and Their

Coherent Control" by Michael Kues et al explores the generation, manipulation, and control of high-dimensional entangled quantum states on a chip [1]. The authors highlight the advantages of using high-dimensional quantum systems, known as qudits, over conventional qubit-based systems in terms of increased information capacity and improved resilience against noise and errors.

The paper describes the experimental implementation of high-dimensional quantum states using integrated photonics. The authors discuss the use of integrated waveguide circuits on a chip to generate and manipulate entangled photon pairs with high-dimensional quantum states. They also delve into the coherent control of high-dimensional entangled states on the chip, through the utilization of reconfigurable waveguide circuits for precise manipulation of photon states.

Additionally, the paper addresses the characterization and measurement of high-dimensional entangled states. The implementation of state tomography techniques is discussed, enabling accurate determination of the quantum states produced on the chip. The measurement results validate the successful generation of high-dimensional entanglement and demonstrate the functionality of the on-chip platform to perform deterministic high-dimensional gate operations.

Another notable approach is the paper "Asymptotic Improvements to Quantum Circuits via Qutrits" where the authors focus on the Generalized Toffoli gate, an important primitive in quantum algorithms. They present a construction that uses qutrits and doesn't require ancilla bits, which are extra bits used to implement irreversible logical operations in quantum computing. Instead, the qutrit's third state is used to store temporary information [2].

The Generalized Toffoli gate, has been studied before, but previous circuit constructions for this gate, such as the Gidney, He, and Barenco designs, rely on qubits and have different tradeoffs in terms of circuit depth and ancilla usage. In their paper the qutrit-based construction, eliminates the need for ancilla bits by directly storing temporary information in the qutrit controls. This eliminates the requirement for ancilla bits and improves the efficiency of the circuit. This new construction results in significant improvements in circuit depth and gate count for important quantum algorithms like Grover's search and Shor's factoring algorithm. It also offers a more favorable tradeoff between information compression and higher per-qudit errors, justifying the use of qutrits in quantum computing.

Quantum computing company Rigetti is exploring experimental new hardware configurations that could improve the performance of its quantum processors, by introducing a third energy state to its qubits, thus turning them into qutrits. Their superconducting quantum processors are based on the transmon design [3] that can address multiple energy states, enabling increased information encoding and decreased readout errors [4]. This is in part due to the much larger state space accessible using qutrits — single qutrit operations live in SU(3), while two-qutrit operations live in SU(9) — a more than twofold increase in dimensionality over the two-qubit case.

Quantum computing company Rigetti is exploring experimental new hardware configurations that could improve the performance of its quantum processors by introducing a third energy state to its qubits, thus turning them into qutrits. Their superconducting quantum

processors are based on the transmon design [3], which allows them to access multiple energy states and offers the advantages of increased information encoding and decreased readout errors [4]. One key factor contributing to these advantages is the significantly larger state space accessible using qutrits compared to qubits. Single qutrit operations reside in SU(3), while two-qutrit operations exist in SU(9), resulting in a more than twofold increase in dimensionality compared to the two-qubit case.

In the paper ”Quantum Information Scrambling on a Superconducting Qutrit Processor,” the authors explore the dynamics of quantum information in strongly interacting systems using qutrits (three-level quantum systems) instead of the conventional two-level qubits [5]. This work demonstrates the potential of higher-dimensional quantum systems like qutrits in achieving resource-efficient encoding of complex quantum circuits, serving as a proof of principle for using qutrit-based quantum processors and paves the way for building more advanced quantum information processors.

Finally, another notable paper ”Quantum Information Scrambling on a Superconducting Qutrit Processor” explores the dynamics of quantum information in strongly interacting systems, known as quantum information scrambling. This phenomenon has recently become a common thread in understanding black holes, transport in exotic non-Fermi liquids, and many-body analogs of quantum chaos. Previously, verified experimental implementations of scrambling focused on systems composed of two-level qubits. However, higher-dimensional quantum systems, such as qutrits (three-level quantum systems), may exhibit different scrambling modalities and are predicted to saturate conjectured speed limits on the rate of quantum information scrambling.

The authors take the first steps toward accessing such phenomena by realizing a quantum processor based on superconducting qutrits. They demonstrate the implementation of universal two-qutrit scrambling operations and embed them in a five-qutrit quantum teleportation protocol link.aps.org. The measured teleportation fidelities (Favg=0.568±0.001) confirm the presence of scrambling even in the presence of experimental imperfections and decoherence [5].

The teleportation protocol connects to recent proposals for studying traversable wormholes in the laboratory and demonstrates how quantum technology that encodes information in higher-dimensional systems can exploit a larger and more connected state space to achieve resource-efficient encoding of complex quantum circuits.

The development of a superconducting five-qutrit processor that runs a quantum teleportation algorithm serves as a proof of principle. The authors engineer two new ways of entangling superconducting qutrits, placing them in a nonclassical state required for quantum computing. While it is still too early for these studies to teach us something new about quantum information propagation, similar algorithms run on future quantum processors might shed light on such fundamental questions.

APPENDIX B. CODE IMPLEMENTATION

B.1 Qutrit Quantum Kernel

In this section the code presented was used to perform the quantum kernel-based classification using qutrits. The first step was defining the qutrit states as column vectors (`q0`, `q1`, `q2`) and the Gell-Mann matrices (`gm1` to `gm8`), the Hadamard operator for qutrits, as well as the `LZ` matrix constructed using the Gell-Mann matrices `gm3` and `gm8` and the `LZZ` matrix calculated as the Kronecker product of `LZ` with itself.

```
# Define the qutrit states as column vectors
q0 = np.array([[1], [0], [0]])
q1 = np.array([[0], [1], [0]])
q2 = np.array([[0], [0], [1]])

# Define the Gell-Mann matrices
gm1 = np.kron(q0, q1.T) + np.kron(q1, q0.T)
gm2 = -1j * (np.kron(q0, q1.T) - np.kron(q1, q0.T))
gm3 = np.kron(q0, q0.T) - np.kron(q1, q1.T)
gm4 = np.kron(q0, q2.T) + np.kron(q2, q0.T)
gm5 = -1j * (np.kron(q0, q2.T) - np.kron(q2, q0.T))
gm6 = np.kron(q1, q2.T) + np.kron(q2, q1.T)
gm7 = -1j * (np.kron(q1, q2.T) - np.kron(q2, q1.T))
gm8 = 1/np.sqrt(3) * (np.kron(q0, q0.T) + np.kron(q1, q1.T) - 2*np.kron(q2, q2
    .T))

# Define the Hadamard operator for qutrits
H = (1/np.sqrt(3)) * np.array([[1, 1, 1], [1, np.exp(2j*np.pi/3), np.exp(-2j*
    np.pi/3)], [1, np.exp(-2j*np.pi/3), np.exp(2j*np.pi/3)]])

# Define the LZ and LZZ operator
LZ = gm3 + np.sqrt(3) * gm8
LZZ = np.kron(LZ2, LZ2)
```

Listing B.1: Qutrit States and Operators

The encoding function is responsible for encoding four features onto a qutrit. To achieve this, it utilizes two important mathematical constructs: the Gell-Mann matrices and the Hadamard operator. The Hadamard operator is utilized to initially put the qutrit into a superposition state, allowing it to exist in a combination of its possible states. Then the initial four Gell-Mann matrices are used to encode the features.

```
def encoding(vector):
    """Encode four features on a qutrit."""
    generators = [gm1, gm2, gm3, gm4]
    sum_val = 0
    for i in range(4):
        sum_val += (1j * vector[i] * generators[i])
    return np.dot(expm(sum_val), np.dot(H, q0))
```

Listing B.2: Encoding Function

The `kernel` function computes the quantum kernel between two data points by encoding them on qutrits, applying an entanglement gate, and computing the inner product between the resulting states. The `kernel_matrix` function computes the kernel matrix between two sets of data points by evaluating the kernel function on pairwise data. Then, this can be passed to an SVM classifier instantiated with the `kernel_matrix` as the kernel.

```
1
2  def kernel(x1, x2):
3      """The quantum kernel."""
4      qutrit_1 = encoding(x1)
5      qutrit_2 = encoding(x2)
6
7      # Entanglement gate
8      entangle_gate = 1j * LZZ2
9
10     # Applying entanglement between the three qutrits
11     qutrit_1x2 = np.kron(qutrit_1, qutrit_2)
12     kron1 = np.dot(expm(entangle_gate), qutrit_1x2)
13
14     qutrit_1 = encoding(x2)
15     qutrit_2 = encoding(x2)
16
17     # Applying entanglement between the three qutrits
18     qutrit_1x2 = np.kron(qutrit_1, qutrit_2)
19     kron2 = np.dot(expm(entangle_gate), qutrit_1x2)
20
21     return np.real(np.dot(kron1.conj().T, kron2)**2)[0][0]
22
23 def kernel_matrix(A, B):
24     """Compute the matrix whose entries are the kernel
25        evaluated on pairwise data from sets A and B."""
26     return np.array([[kernel(a, b) for b in B] for a in A])
27
28 svm = SVC(kernel=kernel_matrix).fit(X_train, y_train)
```

Listing B.3: Kernel Functions

The provided code was utilized for quantum kernel-based classification tasks, where the data was represented using one or more qutrits, with two qutrits yielding optimal results for most datasets. The kernel matrix was computed using the specified quantum kernel function. For instance, in the case of the Iris dataset, the approach achieved an accuracy of 90%. Similarly, for other problems, variations of the code were employed, such as using more qutrits or adjusting the number of stacked layers.

B.2 Qutrit Quantum Neural Network

In the case of the neural network, PyTorch is chosen over NumPy due to its robustness and efficiency in handling complex operations and training custom models. This decision is influenced by the increase in complexity and the number of operations involved in

the network. Previously, for the quantum kernel a hybrid quantum-classical method was employed, but now, with the introduction of the variational layer, PyTorch facilitates the explicit definition of the model's architecture and training process.

```
class QNN(nn.Module):

    def __init__(self, num_layers, num_features):
        super(QNN, self).__init__()
        self.num_layers = num_layers
        self.num_features = num_features

        # Define the qutrit states as column vectors
        self.q0 = torch.tensor([[1], [0], [0]], dtype=torch.cfloat)
        self.q1 = torch.tensor([[0], [1], [0]], dtype=torch.cfloat)
        self.q2 = torch.tensor([[0], [0], [1]], dtype=torch.cfloat)

        # Define the Gell-Mann matrices
        self.gm1 = torch.kron(self.q0, self.q1.T) + torch.kron(self.q1, self.q0.T)
        self.gm2 = -1j * (torch.kron(self.q0, self.q1.T) - torch.kron(self.q1, self.q0.T))
        self.gm3 = torch.kron(self.q0, self.q0.T) - torch.kron(self.q1, self.q1.T)
        self.gm4 = torch.kron(self.q0, self.q2.T) + torch.kron(self.q2, self.q0.T)
        self.gm5 = -1j * (torch.kron(self.q0, self.q2.T) - torch.kron(self.q2, self.q0.T))
        self.gm6 = torch.kron(self.q1, self.q2.T) + torch.kron(self.q2, self.q1.T)
        self.gm7 = -1j * (torch.kron(self.q1, self.q2.T) - torch.kron(self.q2, self.q1.T))
        self.gm8 = 1/torch.sqrt(torch.tensor(3., dtype=torch.float)) * (torch.kron(self.q0, self.q0.T) + torch.kron(self.q1, self.q1.T) - 2*torch.kron(self.q2, self.q2.T))
        self.generators = [self.gm1, self.gm2, self.gm3, self.gm4, self.gm5, self.gm6, self.gm7, self.gm8]

        # Define the LZ and LZZ matrices
        self.lz = self.gm3 + torch.sqrt(torch.tensor(3., dtype=torch.float)) * self.gm8
        self.lz[1][1]=0
        self.hadamard = (1 / torch.sqrt(torch.tensor(3.0))) * torch.tensor([[1, 1, 1], [1, torch.exp(torch.tensor(2j) * torch.tensor(3.1416 / 3.0)), torch.exp(torch.tensor(-2j) * torch.tensor(3.1416 / 3.0))], [1, torch.exp(torch.tensor(-2j) * torch.tensor(3.1416 / 3.0)), torch.exp(torch.tensor(2j) * torch.tensor(3.1416 / 3.0))]], dtype=torch.cfloat)
        self.LZZ2 = torch.kron(self.lz, self.lz)

        # Create the parameters list
        self.weights = nn.ParameterList()
        for i in range(self.num_layers*(int(self.num_features/4))):

            # Weights for the Gell-Mann rotations
```

```
            self.weights.append(nn.Parameter(torch.randn(4, dtype=torch.float)
    ))

    def forward(self, batch):

        logits = torch.empty(batch.shape[0], 3, dtype=torch.double)
        for idx, x in enumerate(batch):
            qutrit_1 = torch.matmul(self.hadamard, self.q0)

            # Apply rotations using Gell-Mann matrices
            for i in range(self.num_layers*(int(self.num_features/4))):

                # Apply the encoding
                encoded = torch.zeros([3,3], dtype=torch.cfloat)
                for index in range(4):
                    encoded += (1j * x[(index+4*i)%self.num_features] * self.
    generators[index])
                qutrit_1 = torch.matmul(torch.matrix_exp(encoded), qutrit_1)

                # The variational layer
                gm_weights = self.weights[i]
                encoded = torch.zeros([3,3], dtype=torch.cfloat)
                for index in range(4):
                    encoded += (1j * gm_weights[index] * self.generators[index
    +4])

                qutrit_1 = torch.matmul(torch.matrix_exp(encoded), qutrit_1)

            # Get the probabilities
            probabilities = torch.abs(qutrit_1.flatten())**2
            probabilities /= torch.sum(probabilities)
            logits[idx] = probabilities

        return logits
```

Listing B.4: Quantum Neural Network class

The QNN class is implemented as a module to encapsulate the functionality of the model and inherit from PyTorch's `nn.Module`. Additionally, this design choice enables seamless integration with PyTorch's optimization algorithms, such as the `RMSprop` optimizer which is mainly used for training, and commonly used loss functions like cross-entropy loss. This combination of modular design, efficient computations, and built-in optimization tools makes PyTorch an ideal framework for developing and training the QNN model in a reliable and robust manner.

The full implementation can be found in the GitHub repository:
`https://github.com/Themiscodes/Quantum-Neural-Networks`

B.3 Qubit Variational Quantum Classifier

This code demonstrates the usage of the Qiskit and scikit-learn libraries to define and utilize the VQC. The specific details of data ingestion and preprocessing are omitted, but numpy and pandas were employed for data manipulation, and `MinMaxScaler` for preprocessing.

The `ZZFeatureMap` is employed to transform the features of the input data into quantum states, and the `RealAmplitudes` class is used to construct a variational form for the ansatz. The dimensions of the feature map and the number of qubits for the ansatz are automatically determined based on the shape of the input data.

To optimize the parameters of the variational form during the training process, the `L_BFGS_B` optimizer is selected. However, in some instances the `COBYLA` optimizer was used instead.

Subsequently, an instance of the `VQC` is created, which incorporates the previously defined `feature map`, `ansatz`, and `optimizer`. The VQC class is responsible for training the classifier using a quantum circuit. This is accomplished by invoking the fit method on the classifier instance.

Lastly, the trained classifier is evaluated using a test dataset by invoking the score method. This method returns a score that represents the performance of the classifier.

```
from qiskit.circuit.library import ZZFeatureMap, RealAmplitudes
from qiskit.algorithms.optimizers import COBYLA, L_BFGS_B
from qiskit_machine_learning.algorithms.classifiers import VQC
from sklearn.model_selection import train_test_split
from qiskit.utils import algorithm_globals

# Define the feature map and ansatz
encoding = ZZFeatureMap(feature_dimension=X.shape[1], reps=1)
ansatz = RealAmplitudes(num_qubits=X.shape[1], reps=2)

# Optimizer
optimizer = L_BFGS_B(maxfun=10, maxiter=12)

# Create a VQC instance
classifier = VQC(feature_map=encoding, ansatz=ansatz, optimizer=optimizer,)

# Training
classifier.fit(train_features, train_labels)

# Evaluate the VQC on the test dataset
test_accuracy = classifier.score(test_features, test_labels)* 100
```

Listing B.5: Qiskit Variational Quantum Classifier

APPENDIX C. DATASETS

The datasets chosen for multiclass classification were wine cultivars, iris, seed, and glass from the UCI Machine Learning Repository. In a classification context, these are well posed problems with ”well behaved” class structures.

Iris Dataset

- Description: This dataset is perhaps one of the most famous datasets in the field of machine learning. It contains measurements of sepal length, sepal width, petal length, and petal width for three different species of iris flowers.
- Number of Instances: 150
- Number of Attributes: 4
- Attribute Information: The attributes include sepal length, sepal width, petal length, and petal width, all measured in centimeters.
- Dataset Link: `https://archive.ics.uci.edu/ml/datasets/iris`

Wine Cultivars Dataset

- Description: This dataset contains the results of a chemical analysis of wines from three different cultivars in Italy. The analysis determined the quantities of various constituents present in the wines.
- Number of Instances: 178
- Number of Attributes: 13
- Attribute Information: The attributes include measurements of alcohol, malic acid, ash, alkalinity of ash, magnesium, total phenols, flavanoids, non-flavanoid phenols, proanthocyanins, color intensity, hue, OD280/OD315 of diluted wines, and proline.
- Dataset Link: `https://archive.ics.uci.edu/ml/datasets/wine`

Seed Dataset

- Description: This dataset includes measurements of geometric properties of kernels belonging to three different varieties of wheat. The properties were derived from digitized images of the kernels.
- Number of Instances: 210
- Number of Attributes: 7

- Attribute Information: The attributes include measurements such as area, perimeter, compactness, length of kernel, width of kernel, asymmetry coefficient, and length of kernel groove.
- Dataset Link: `https://archive.ics.uci.edu/ml/datasets/seeds`

Glass Dataset

- Description: This dataset contains information about various types of glass. The data includes features like the refractive index and the percentages of different chemical elements present in the glass, which can be used to predict the type of glass.
- Number of Instances: 214
- Number of Attributes: 10
- Attribute Information: The attributes include features like refractive index, sodium, magnesium, aluminum, silicon, potassium, calcium, barium, iron, and the type of glass.
- Dataset Link: `https://archive.ics.uci.edu/ml/datasets/glass+identification`

These datasets provide diverse and challenging scenarios for assessing the models performance in handling multiclass classification tasks. The inclusion of real-world datasets further contributes to a comprehensive evaluation of the models' effectiveness and generalizability. The difference in the number of features across these datasets also provides an opportunity to test the adaptability of the feature map.

REFERENCES

[1] Kues, M., Reimer, C., Roztocki, P. et al, *On-chip generation of high-dimensional entangled quantum states and their coherent control*, 2017. doi:10.1038/nature22986

[2] Pranav Gokhale, Jonathan M. Baker, et al, *Asymptotic Improvements to Quantum Circuits via Qutrits*, 2019. arxiv:1905.10481

[3] Koch, J., et al, *Charge-insensitive qubit design derived from the Cooper pair box*, 2007. doi:10.1103/PhysRevA.76.042319

[4] Mallet, F., Ong, F. R., Palacios-Laloy, A., Nguyen, F., Bertet, P., Vion, D., and Esteve, D., *Single-shot qubit readout in circuit quantum electrodynamics*, 2009. doi.org:10.1038/nphys1400

[5] M. S. Blok, V. V. Ramasesh, T. Schuster, K. O'Brien, J.M. Kreikebaum, D. Dahlen, A. Morvan, B. Yoshida, N. Y. Yao, I. Siddiqi, *Quantum Information Scrambling in a Superconducting Qutrit Processor*, 2020. arXiv:2003.03307

[6] Michael Artin, *Algebra*, 1991. ISBN 978-0-89871-510-1

[7] Helmut H. Schaefer and Manfred P. Wolff, *Topological Vector Spaces*, 1999. ISBN 978-1-4612-7155-0

[8] R. G. Lerner and L. G. Trigg, *Encyclopaedia of Physics*, 1991. ISBN 0-89573-752-3

[9] William C. Brown, *Matrices and vector spaces*, 1991. ISBN 978-0-8247-8419-5

[10] Richard L. Burden and J. Douglas Faires, *Numerical Analysis*, 1993. ISBN 0-534-93219-3

[11] Walter Ruding, *Real and Complex Analysis*, 1987. ISBN 978-0-07-100276-9

[12] Armand Borel, *Linear algebraic groups, Graduate Texts in Mathematics*, 1991. doi:10.1007/978-1-4612-0941-6, ISBN 978-0-387-97370-8

[13] Hans Kurzweil and Bernd Stellmacher, *Theorie der endlichen Gruppen*, 1998. doi:10.1007/978-3-642-58816-7

[14] Joel Merker, *Theory of Transformation Groups, by S. Lie and F. Engel (Vol. I, 1888)*, 2010. arXiv:1003.3202

[15] Nicolas Bourbaki, *Lie Groups and Lie Algebras*, 1989. ISBN 978-3-540-64242-8

[16] Brian C. Hall, *Lie Groups, Lie Algebras, and Representations: An Elementary Introduction*, 2015. ISBN 978-3319134666

[17] Murray Gell-Mann, *"Symmetries of Baryons and Mesons", Physical Review, American Physical Society*, 1962. doi:10.1103/physrev.125.1067 ISSN 0031-899X

[18] Paul A. M. Dirac, *A new notation for quantum mechanics, Mathematical Proceedings of the Cambridge Philosophical Society*, 1939. ISBN 978-0198520115

[19] Felix Bloch, *Nuclear induction*, 1946. doi:10.1103/physrev.70.460

[20] Michael A. Nielsen and Isaac L. Chuang, *Quantum Computation and Quantum Information*, 2004. ISBN 978-0-521-63503-5

[21] Donald A. McQuarrie, *Quantum Chemistry*, 1983. ISBN 978-0-935-70213-2

[22] Paul A. M. Dirac, *The Principles of Quantum Mechanics*, 1930/1958. ISBN 978-0-198-52011-5

[23] Donald A. Danielson, *Vectors and Tensors in Engineering and Physics*, 2003. ISBN 978-0-8133-4080-7

[24] Mark Byrd, *Differential geometry on SU(3) with applications to three state systems*, 1998. arXiv:math-ph/9807032 doi:10.1063/1.532618

[25] Colin P. Williams, *Explorations in Quantum Computing*, 2011. ISBN 978-1-84628-887-6

[26] Stuart J. Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach*, 2003. ISBN 0-13-790395-2

[27] Ian Goodfellow, Yoshua Bengio and Aaron Courville, *Deep Learning*, 2016. http://www.deeplearningbook.org

[28] Geoffrey Hinton, *Neural Networks for Machine Learning*, 2012. http://www.cs.toronto.edu/ tijmen/csc321/slides/lecture_slides_lec6.pdf

[29] Diederik P. Kingma and Jimmy Ba, *Adam: A Method for Stochastic Optimization*, 2014. arXiv:1412.6980

[30] Jarrod R. McClean, Sergio Boixo, Vadim N. Smelyanskiy, Ryan Babbush, and Hartmut Neven, *Barren plateaus in quantum neural network training landscapes*, 2018. doi:10.1038/s41467-018-07090-4 arXiv:1803.1117

[31] M. J. D. Powell, *A Direct Search Optimization Method That Models the Objective and Constraint Functions by Linear Interpolations*, 1994. doi:10.1007/978-94-015-8330-5_4

[32] Dong C. Liu and Jorge Nocedal, *On the limited memory BFGS method for large scale optimization*, 1989. doi.org:10.1007/BF01589116

[33] Pearson, K, *On Lines and Planes of Closest Fit to Systems of Points in Space*, 1901. doi:10.1080/14786440109462720

[34] Seth Lloyd, Masoud Mohseni and Patrick Rebentrost, *Quantum Principal Component Analysis*, 2013. arXiv:1307.0401

[35] Vittorio Giovannetti, Seth Lloyd and Lorenzo Maccone, *Quantum random access memory*, 2008. arXiv:0708.1879

[36] Boser, Bernhard E., Guyon, Isabelle M., and Vapnik, Vladimir N., *A training algorithm for optimal margin classifiers*, 1992. doi:10.1145/130385.130401

[37] Frank Rosenblatt, *The perceptron: A probabilistic model for information storage and organization in the brain*, 1958. https://doi.org/10.1037/h0042519

[38] Edward Farhi, Jeffrey Goldstone and Sam Gutmann, *A Quantum Approximate Optimization Algorithm* , 2014. arXiv:1411.4028

[39] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J. Love, Alán Aspuru-Guzik and Jeremy L. O'Brien, *A variational eigenvalue solver on a quantum processor*, 2014. arXiv:1304.3061

[40] Maria Schuld, VAlex Bocharov, Krysta Svore and Nathan Wiebe, *Circuit-centric quantum classifiers*, 2018. arXiv:1804.00633

[41] Vedran Dunjko, Jacob M. Taylor and Hans J. Briegel, *Quantum-enhanced machine learning*, 2016. arXiv:1610.08251

[42] Marcello Benedetti, Erika Lloyd, Stefan Sack and Mattia Fiorentini, *Parameterized quantum circuits as machine learning models*, 2019. arXiv:1906.07682

[43] Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. Divincenzo, Norman Margolus, Peter Shor, et al., *Elementary gates for quantum computation*, 1995. arXiv:quant-ph/9503016 doi:10.1103/physreva.52.3457

[44] Adrián Pérez-Salinas, Alba Cervera-Lierta, Elies Gil-Fuster and José I. Latorre, *Data re-uploading for a universal quantum classifier*, 2020. arXiv:1907.02085

[45] Kosuke Mitarai, Makoto Negoro, Masahiro Kitagawa and Keisuke Fujii, *Quantum Circuit Learning*, 2018. arXiv:1803.00745

[46] Maria Schuld, Ville Bergholm, Christian Gogolin, Josh Izaac and Nathan Killoran, *Evaluating analytic gradients on quantum hardware*, 2018. arXiv:vqcpaper

[47] Maria Schuld and Francesco Petruccione, *Supervised Learning with Quantum Computers*, 2018. doi:10.1007/978-3-319-96424-9

[48] Seth Lloyd, Maria Schuld, Aroosa Ijaz, Josh Izaac and Nathan Killoran, *Quantum embeddings for machine learning*, 2020. arXiv:2001.03622

[49] Sergio Altares-López, Angela Ribeiro and Juan José García-Ripoll, *Automatic design of quantum feature maps*, 2021. arXiv:2105.12626

[50] Jarrod R. McClean, et al, *Barren plateaus in quantum neural network training landscapes*, 2018. arXiv:1803.11173v1

[51] James Stokes, Josh Izaac, Nathan Killoran and Giuseppe Carleo, *Quantum Natural Gradient*, 2019. doi:10.22331/q-2020-05-25-269 arXiv:1909.02108

[52] Edward Grant, Leonard Wossnig, Mateusz Ostaszewski and Marcello Benedetti, *An initialization strategy for addressing barren plateaus in parameterized quantum circuits*, 2019. doi:10.22331/q-2019-12-09-214 arXiv:1903.05076

[53] Andrea Skolik, Jarrod R. McClean, Masoud Mohseni, Patrick van der Smagt and Martin Leib, *Layerwise learning for quantum neural networks*, 2020. doi:10.1007/s42484-020-00036-4 arXiv:2006.14904

[54] M. Cerezo, Akira Sone, Tyler Volkoff, Lukasz Cincio and Patrick J. Coles, *Cost Function Dependent Barren Plateaus in Shallow parameterized Quantum Circuits*, 2020. doi:10.1038/s41467-021-21728-w arXiv:2001.00550

[55] Beng Yee Gan, Daniel Leykam and Dimitris G. Angelakis, *Fock State-enhanced Expressivity of Quantum Machine Learning Models*, 2021. arXiv:2107.05224

[56] Behnam Neyshabur, Srinadh Bhojanapalli and Nathan Srebro, *A PAC-Bayesian Approach to Spectrally-Normalized Margin Bounds for Neural Networks*, 2018. arXiv:1707.09564

[57] Vladimir Vapnik and Alexey Chervonenkis, *On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities*, 1971. doi:10.1137/1116025

[58] Sukin Sim, Peter D. Johnson and Alan Aspuru-Guzik, *Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms*, 2019. arXiv:1905.10876

[59] Amira Abbas, David Sutter, Christa Zoufal, Aurélien Lucchi, Alessio Figalli and Stefan Woerner, *The power of quantum neural networks*, 2020. arXiv:2011.00027

[60] Amira Abbas, David Sutter, Alessio Figalli and Stefan Woerner, *Effective dimension of machine learning models*, 2021. arXiv:2112.04807

[61] Maria Schuld and Nathan Killoran, *Quantum machine learning in feature Hilbert spaces*, 2018. arXiv:1803.07128

[62] Maria Schuld, *Supervised quantum machine learning models are kernel methods*, 2021. arXiv:2101.11020

[63] Tim von Hahn, *Beautiful Plots: The Decision Boundary*, 2021. https://www.tvhahn.com/posts/beautiful-plots-decision-boundary/

[64] John Preskill, *Quantum Computing in the NISQ era and beyond*, 2018. arXiv:1801.00862 doi:10.48550