DEGREE PROJECT IN TECHNOLOGY,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2021

A Quantum Neural Network for Noisy Intermediate Scale Quantum Devices.

Altay Dikme

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF ENGINEERING SCIENCES

Author

Altay Dikme altay@kth.se
School of Engineering Sciences
KTH Royal Institute of Technology

Place for Project

Torshamnsgatan 23
164 83, Stockholm
Sweden
Ericsson

Examiner

Val Zwiller
KTH Royal Institute of Technology

Supervisor

Ahsan Jawed Awan

Ericsson

Abstract

Neural networks have helped the field of machine learning grow tremendously in the past decade, and can be used to solve a variety of real world problems such as classification problems. On another front, the field of quantum computing has advanced, with quantum devices publicly available via the cloud. The availability of such systems has led to the creation of a new field of study, Quantum Machine Learning, which attempts to create quantum analogues of classical machine learning techniques. One such method is the Quantum Neural Network (QNN) inspired by classical neural networks. In this thesis we design a QNN compatible with Noisy Intermediate Scale Quantum (NISQ) devices, which are characterised by a limited number of qubits and small decoherence times. Furthermore we provide an implementation of the QNN classifier using the open source quantum computing software development kit, Qiskit provided by IBM. We perform a binary classification experiment on a subset of the MNIST data set, and our results showed a classification accuracy of 80.6% for a QNN with circuit depth 20.

Keywords

Quantum Computing, Quantum Machine Learning, Classification, NISQ Devices, Variational Algorithm, Quantum Neural Networks

Sammanfattning

Neurala nätverk har varit en stor del av utvecklingen av maskininlärning som ett forskningsområde i det senaste årtiondet, och dessa nätverk har flera appliceringsområden, som till exempel klassificieringsproblemet. Parallelt med denna utveckling, har forskning kring kvantdatorer vuxit fram, med flera kvantsystem allmänt tillgängliga via molnet. Denna tillgänglighet har lett till skapandet av ett nytt forskningsområde; kvantmaskininlärning, som försöker skapa motsvarigheter till klassiska maskininlärningsmetoder på kvantdatorer. En sån metod är kvantneurala nätverk som inspireras av klassiska neurala nätverk. I denna avhandling designar vi ett kvantneuralt närverk som är kompatibel med nuvarande kvantsystem, som kännetecknas av ett begränsat antal qubits och korta dekoherenstider. Dessutom tillhandahåller vi en implementering av en klassificerare med ett kvantneuralt nätverk, med hjälp av IBMs programvaruutvecklingsmiljö Qiskit. Vi utför ett binärt klassificeringsexperiment på en delmängd av MNIST-datamängden, och våra resultat visar en klassificeringsnoggrannhet på $80,6\%$ för ett kvantneuralt nätverk med kretsdjup 20.

Nyckelord

Kvantberäkning, Kvantmaskininläring, Klassificiering, Variationsalgoritmer, Kvantneurala Nätverk

Acknowledgements

I would like to thank Ericsson for the opportunity to work on this thesis, which was both interesting and challenging for me. Special thanks to my supervisor Ahsan Javed Awan for the weekly discussions. Ahsan gave me valuable advice, information and guidance when I was struggling on understanding certain theoretical aspects of the thesis. I also want to highlight the great job done by Ericsson as a company regarding the current Covid-19 pandemic, and making it possible to work from home while supporting their employees.
Furthermore I would like to thank my peers at KTH who I've spent the last five years with. Special thanks to Nicolas Reichel whom I've spent countless hours debating difficult problems and concepts.
Last but not least, I would like to thank my family for their support and interest in the topic of my studies. My father for asking me how the qubits are doing, my mother for pushing me to work as hard as possible, and my sister for supporting me. I would also like to thank my girlfriend Ellinor Kalderén for always being by my side and motivating me.

Acronyms

NISQ	Noisy Intermediate Scale Quantum
PQC	Parametric Quantum Circuit
VQA	Variational Quantum Algorithm
QNN	Quantum Neural Network
QSVM	Quantum Support Vector Machine
QGAN	Quantum Generative Adversarial Network
NN	Neural Networks
SDK	Software Development Kit
QkNN	Quantum k-Nearest-Neighbors

Contents

Chapter 1

Introduction

1.1 Motivation

Four decades ago Richard Feynman proposed the idea of a quantum computer, which could be used to simulate quantum systems which were too complex to simulate for conventional computers [1]. Only recently have such quantum computers finally been constructed, and been made commercially available [2, 3]. The advent of actual quantum computers expanded the possibility of calculations beyond simulation on classical systems, however these systems are not fault-tolerant. This is due to factors such as decoherence of qubits, gate erros, and measurement erros within the quantum system. These devices are colloqually referred as NISQ devices, and we are said to be in the NISQ-era of quantum computing at present day [4]. Despite the faults of present day devices, the availability of quantum systems via the cloud has led to research on a large number of applications of quantum computing, such as optimization [5, 6], factorization [7], molecule simulations [8], finance [9], and machine learning [10–16].

Specifically the field of Quantum Machine Learning has grown at a fast pace, where attempts are being made to integrate machine learning algorithms with quantum subroutines. Several quantum equivalents such as the Quantum Support Vector Machine (QSVM), Quantum Generative Adversarial Network (QGAN) and Quantum k-Nearest-Neighbors (QkNN) have been implemented [10–13, 17]. Until recently however there was no consensus as to what constituted a QNN. Instead criteria for such a method was described in [18], and further work on quantum perceptron models were conducted [19].

While specifics regarding the definition of a QNN do not have consensus, there is a general and currently accepted model constituting of a Parametric Quantum Circuit (PQC) [20]. These PQC models have a structure that is directly inspired by the structure of the classical Neural Networks (NN) equivalent [14–16, 21, 22]. These models have shown promising results [14, 15, 23], however it is important to note that while such QNN models have been shown to work well, most experiments have been performed using ideal (i.e error free) simulations on classical computers and not on actual NISQ devices. It is therefore of interest to investigate the requirements for a QNN to run on a NISQ device, and determine if one can design a well performing QNN which is compatible with NISQ devices.

1.2 Purpose and Thesis Statement

This thesis aims to investigate the limitations that NISQ devices impose on the design of QNNs, and present a QNN framework that is compatible for use with such devices. Apart from providing the framework we also investigate the performance of the presented QNN and compare it to previous works, i.e other QNN models, both in terms of performance but also in terms of compatibility. The thesis purpose is thus as follows.

> **To design a Quantum Neural Network for Noisy Intermediate Scale Devices, which are characterised by limited qubits and short coherence times, and investigate the performance of such QNN in terms of classification accuracy.**

1.3 Outline

The remainder of the thesis will be structured as follows:

- **Chapter 2 - Theoretical Background**, introduces the required theoretical knowledge related to this work.
- **Chapter 3 - Classical & Quantum Neural Networks**, introduces the concept of a Quantum Neural Network and how it relates to it's classical equivalent.
- **Chapter 4 - Methodology**, describes the choices we make when designing the QNN model, and also the experimental setup for this thesis investigation.
- **Chapter 5 - Results and Discussion**, presents the results of our experiments

along with a discussion of the results, and comparisons with the results of previous works.

- **Chapter 6 - Conclusion**, presents potential future works to be conducted on the topic, and concludes this thesis.

Chapter 2

Theoretical Background

2.1 Quantum Computing

2.1.1 Qubits

Quantum computing systems make use of quantum bits, so called qubits which can be seen as an analogue to the classical bit used in classical computing. A bit has two possible states which it can be in: 0 and 1. A qubit can also be in these states, however the main difference between the two is that a qubit may also be in some linear superposition of these two states. Mathematically we can denote a qubit as a unit vector in the two-dimensional complex vector space $\mathbb{C}^2$. We can denote a qubit in the 0 state to be in the $|0\rangle$ state and if it is in the 1 state it is in the $|1\rangle$ state. These braket states are defined as

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \alpha, \beta \in \mathbb{C} \tag{2.1}$$

Mathematically then the distinction between a bit and a qubit is that while a bit may only be in the state $|0\rangle$ or $|1\rangle$, a single qubit can be in an arbitrary linear superposition state

$$|\Psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = \begin{bmatrix} \alpha \\ \beta \end{bmatrix} \tag{2.2}$$

under the constraint that we have

$$|\alpha|^2 + |\beta|^2 = 1. \tag{2.3}$$

The constraint is due to a key postulate of Quantum Mechanics known as the Born rule which states that the probability of measuring a quantum state in a given state λ is given by $|\langle\lambda|\Psi\rangle|^2$. Therefore in our case we have the probability of measuring the state in $|0\rangle$ as

$$|\langle 0|\Psi\rangle|^2 = |\alpha\langle 0|0\rangle + \beta\langle 0|1\rangle|^2 = |\alpha\langle 0|0\rangle|^2 = |\alpha|^2$$

and similarly the probability of measuring the state in $|1\rangle$ is $|\beta|^2$. Furthermore we know that the total probability must be equal to 1, and hence the constraint $|\alpha|^2 + |\beta|^2 = 1$.

Since both α and β are complex one may expect to have four degrees of freedom, however the constraint in Eq. 2.3 removes one degree of freedom. Furthermore if we introduce a coordinate change and use the Hopf coordinates. We can rewrite α and β as

$$\alpha = \cos\frac{\theta}{2}, \quad \beta = e^{i\phi}\sin\frac{\theta}{2} \quad \implies |\Psi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle \tag{2.4}$$

With this parametrization one can visualize a general single qubit state using a 2-Sphere S^2, known as the Bloch sphere. Any point on the sphere (and in turn any possible single qubit state) can be descriped using the two angles θ and ϕ.

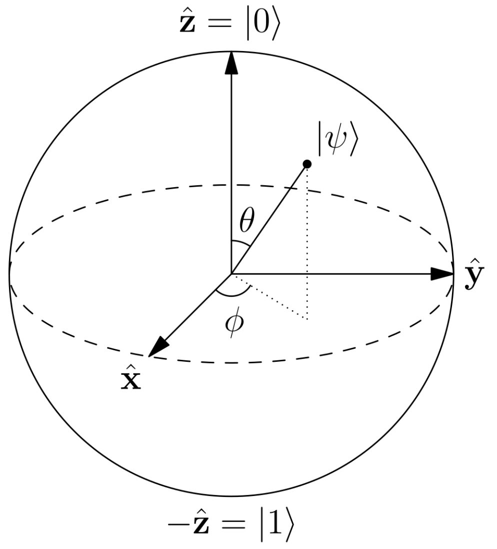

Figure 2.1.1: Bloch sphere representation.

2.1.2 Multiple Qubits and Entanglement

Multiple Qubits

Thus far we have only looked at a single qubit and how we can represent and visualize it. To understand and work with quantum computers however we need to describe how composite systems, i.e multiplie qubits work. Considering the case of two qubits, there are four possible states $|00\rangle, |01\rangle, |10\rangle$ and $|11\rangle$. Similar to the single qubit case, we can determine a mutually orthogonal basis to represent these states.

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ 0 \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.5}$$

where $\otimes$ is the Kronecker product. We can repeat this for the remaining states. In total we have the orthogonal basis given by:

$$|00\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \ |01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \ |10\rangle = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \ |11\rangle = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.6}$$

A general two qubit state can then be written as

$$|\Psi\rangle = \alpha |00\rangle + \beta |01\rangle + \gamma |10\rangle + \delta |11\rangle = \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ \delta \end{bmatrix} \tag{2.7}$$

The normalization constraint is then expanded to

$$|\alpha|^2 + |\beta|^2 + |\gamma|^2 + |\delta|^2 = 1. \tag{2.8}$$

The coefficients α,β,γ and δ are the amplitudes of the bases and just as in the one qubit case $|\alpha|^2$ represents the probability of measuring state $|00\rangle$, $|\beta|^2$ represents the probability of measuring state $|01\rangle$ and so on. Thus the expansion of the normalization constraint as the sum of probability for all states must be 1.

Furthermore it is possible to represent an n-qubit quantum state by a 2^n-dimensional complex vector in the Hilbert space. Denoting the state amplitudes as a_i, the vector can be written as

$$|\Psi\rangle = \sum_{i=0}^{2^n-1} a_i |i\rangle \tag{2.9}$$

with the same logic as earlier after measurement the probability of the state $|\Psi\rangle$ collapsing to the state $|i\rangle$ is given by $|a_i|^2$ and the normalization constraint is given by $\sum_{i=0}^{2^n-1} |a_i|^2 = 1$. This demonstrates a strength of quantum computing, as n qubits can be used for 2^n-dimensional vectors.

Entanglement

Another important concept in quantum computing and quantum mechanics overall is entanglement, which was first formulated in a joint paper by Albert Einstein, Boris Podolsky and Nathan Rosen by presenting the Einstein-Podolsky-Rosen (EPR) paradox. Ironically the paradox was presented as an argument against the completeness of quantum mechanics. The paradox aimed to show that the violation of *local realism* leads to, in Einstein's words, "*spukhafte Fernwirkung*" or "*spooky action at a distance*". Local realism is the collective assumption of localism and realism, meaning that the propagation of information is limited by the speed of light (locality) and that measurements only reveal elements of reality already present in the system being measured (realism). It was later proven by John Stewart Bell that the key assumption, the principle of locality, was mathematically inconsistent with the predictions of quantum theory and provided an upper limit known as Bell's inequality [24].

Bell's inequality introduced the opportunity to experimentally test the validity of local hidden-variable theories which the EPR trio championed, and a variety of experiments have validated Bell's inequality and therefore excluded the idea of local realism from quantum mechanics [25–29]. This lesson in history gives us an idea of the origins of entanglement. Entangled states violate the idea of local realism as is demonstrated by the maximally entangled Bell states. One of the Bell states is given by

$$|\Psi\rangle = \frac{1}{\sqrt{2}} \left(|00\rangle + |11\rangle\right). \tag{2.10}$$

The qubits are in two possible quantum states $|00\rangle$ and $|11\rangle$, and one can see that if the first qubit is in the zero state ($|0\rangle$) then the other must also be in the $|0\rangle$ state. When the first qubit is in the one state ($|1\rangle$) then the other is also in the $|1\rangle$ state. Thus we can see that by measuring the first qubit we a priori know what the result of a measurement on the second qubit will be, and this is due to the entanglement between the two qubits.

2.1.3 Quantum Circuits and Gates

Quantum circuits are built up by combinations of quantum logic gates which transform a given quantum state to another state. The question is then how to represent such quantum logic gates, and what sort of constraints are there for a given gate? One important thing to keep in mind is that the normalization constraint must still be fulfilled after the transformation, in order to ensure that the sum of probabilities are equal to 1. For this the transformation must preserve the inner product. By Wigner's theorem we have:

Theorem 1 *Wigner's Theorem: Any mapping of the vector space onto itself that preserves the value of $|\langle\phi|\psi\rangle|$ may be implemented by a unitary operator U:*

$$|\psi\rangle \rightarrow |\psi'\rangle = U|\psi\rangle$$
$$|\phi\rangle \rightarrow |\phi'\rangle = U|\phi\rangle$$

We can see that if we have a unitary operator U then that means that

$$UU^\dagger = U^\dagger U = I \tag{2.11}$$

where the dagger symbol, $\dagger$ denotes the Hermitian adjoint. If we now have a state ϕ and ψ, and apply a unitary transformation to both of them we see that the inner product is unchanged, and thus the normalization constraint will still by fulfilled after a transformation of the states.

$$\langle\phi| U^\dagger U |\psi\rangle = \langle\phi|\psi\rangle \tag{2.12}$$

Furthermore the number of input qubits and output qubits of any quantum gate should be the same. A unitary quantum gate acting on n qubits is represented by a $2^n \times 2^n$ unitary matrix, since the statevector of an n qubit system is 2^n dimensional.

Circuit Composition

We now have an understanding of what types of transformations can be applied to qubits, we also need to mathematically determine what happens when several quantum gates are applied in series to a qubit, and also how quantum gates applied on different qubits impact the overall state of the system. The two important concepts to keep in mind when composing quantum circuits are: **Serial Gates** and **Parallel Gates**.

Serial Gates

A serially applied circuit simply consists of a series of gates acting on the same qubit or qubits. The simplest example can be given with a two unitary gates applied on a single qubit. If we denote the gates A and B as the first and second gates respectively, the effect of the two gates can be described as a single gate C applied on the state $|\Psi\rangle$ where $C = B \cdot A$.

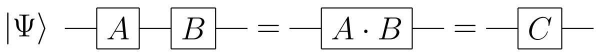

Figure 2.1.2: Circuit diagram of two unitary gates in series. The unitary gates can be multiplied together with matrix multiplication and be represented by the corresponding unitary C.

Since both A and B are unitary 2×2 matrices (for the single qubit case) the resulting gate C will also be represented by a unitary 2×2 matrix.

Parallel Gates

When we have a quantum circuit consisting of more than one qubit, the question is how we mathematically formulate the application of gates on different qubits. The tensor product of two quantum gates is the gate that is equal to the two gates in parallel. Thus if we for example perform an X rotation on the first qubit in state $|\Psi\rangle$ and a Y rotation on the second qubit in the state $|\Phi\rangle$, this is equivalent to performing an $X \otimes Y$ transformation on the $|\Psi\rangle \otimes |\Phi\rangle$ state.

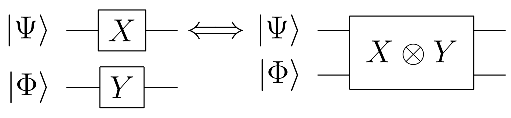

Figure 2.1.3: Circuit diagram of two unitary gates in parallel. In this case the Kronecker product of the two gates give the resulting unitary gate which transforms the combined state.

In the case where you perform a transformation on one qubit but not the other (or others) as in the case below, where we apply a Hadamard gate H on the first qubit and do nothing to the second qubit. This is equivalent to applying a $H \otimes I$ transformation on the $|\Psi\rangle \otimes |\Phi\rangle$ state. See Figure 2.1.4 for a visual representation.

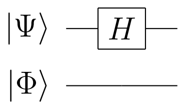

Figure 2.1.4: Example circuit where we apply a Hadamard (Could be any) gate on the first qubit and do nothing on the second qubit. This is equivalent to applying the $H \otimes I$ unitary to the $|\Psi\rangle \otimes |\Phi\rangle$ state.

If we extend this notion to more than two qubits then transformations applied in parallel to n qubits is equivalent to the transformation given by the tensor product of all the gates acting on the statevector. A simple example is if we apply the Hadamard gate to all n qubits then that is equivalent to applying a $H^{\otimes n}$ transformation to the combined statevector of all the qubits.

2.1.4 NISQ Devices and their challenges

NISQ devices refer to currently available quantum computers with anywhere from 5-100 qubits. These have the potential to solve problems which are not solvable in a feasible timeframe on classical systems, however NISQ devices are not yet fault-tolerant. The main issues of NISQ devices are; measurement, decoherence and gate infidelity which we will describe in short (See [30] for an in-depth explanation). Furthermore NISQ devices cannot perform quantum error correction due to the limited number of qubits. The most important error source to consider with current devices is *decoherence*. Due to the limited coherence time of qubits on NISQ devices only short depth quantum circuits can be reliably executed. Therefore quantum circuits must be kept to a minimum in terms of depth, in order to reduce execution time. With a shorter execution time there is a smaller probability of decoherence of qubits occuring and thus more accurate results.

Decoherence

In an ideal noise-free quantum measurement a measurement in the Z-basis on the determinate ground state $|0\rangle$ will always yield the same result, i.e there is a 100% probability of measuring the state to be $|0\rangle$. In reality however this is not the case because a physical qubit can never be completely isolated from it's environment, resulting in some quantum noise. Thus one can expect the result of a set of measurements on identically prepared states to be some distribution of $|0\rangle$ and $|1\rangle$. There are two primary sources of this quantum noise. The first is *relaxation* characterized by the relaxation time T_1, and *dephasing* characterized by the dephasing time T_2. Relaxation time is a property of all superconductive qubit implementations, as the qubit states are *nondegenerate*, $|0\rangle$ is the ground state and $|1\rangle$ is the first excited state. Thus an infinitesimal energy exchange with the environment can lead to the spontaneous relaxation of the $|1\rangle$ state to the $|0\rangle$ state. The expectation value of the time this takes is the relaxation time T_1.

Dephasing is the process wherin environmental coupling causes the relative phase between two eigenstates of the total qubit state to become randomized. In other words the qubit becomes entangled with the environment and the single qubit state is no longer a pure state. The dephasing time T_2 can be described as a combination of the relaxation time and a 'pure dephasing' term from the environmental coupling [31].

$$\frac{1}{T_2} = \frac{1}{T_1} + \frac{1}{\tau_\phi}$$

Gate Infidelity

The error caused by the physical implementation of quantum gates is referred to as gate infidelity. In short, a superconducting phase qubit uses electromagnetic pulses to induce Rabi oscillations in the qubit, thereby causing a state transition with a given probability dependent on the frequency and time of the pulse [31]. In practice, this operation does not perfectly correspond to the theoretical gate, but rather some perturbed (but still unitary) gate. The error caused by this perturbation is called the gate error.

2.1.5 IBM - Qiskit

Qiskit is IBM's Python based open-source Software Development Kit (SDK) for designing quantum circuits, which can be simulated locally or remotely on IBM's hardware [2, 32]. It is also possible to send jobs consisting on quantum circuits to any of IBM's available quantum chips. These quantum chips all fall into the realm of NISQ devices, and suffer from decoherence and gate infidelity issues. Therefore it is still important to keep these sources of error in mind when designing quantum circuits. Due to the simplicity with installing Qiskit and it's use, it will be the main framework used for computation and simulation in this thesis.

Chapter 3

Classical & Quantum Neural Networks

In this chapter we will give a brief overview of what classical neural networks are, what they are used for and furthermore we will describe the reasoning behind the quantum version of such a neural network.

3.1 Neural Networks

When talking about NN we refer to artificial neural networks used in the field of machine learning and not biological neural networks. Neural networks can be used in many applications such as function approximation, data processing or classification such as pattern detection [33]. Neural networks are a form of supervised learning which are trained by processing samples, and generally consist of layers of neurons. Each neuron is linked with other neurons by a link, similar to biological neurons, and the "strength" of the link is determined by a weight θ_i. These weights are then iteratively tuned using a method of gradient descent in order to reach an optimum point of some cost function $\mathcal{C}(\vec{\theta})$. A more complete overview of neural networks is given in [33].

The formal classification problem which neural networks attempt to solve can be formulated as follows.

Problem 1 *We let $\mathcal{X}$ be the set of inputs and $\mathcal{Y}$ be the set of outputs. The goal is that given a dataset $\mathcal{D} = \{(x^1, y^1), \dots, (x^M, y^M)\}$ of pairs which we call the training input and target output, to predict the output $y \in \mathcal{Y}$ of a new input $x \in \mathcal{X}$ which has not earlier been* *seen. We assume that $\mathcal{X} \in \mathbb{R}$ and $\mathcal{Y} \in \{-1, 1\}$ which is simply a binary classification task.*

3.2 Quantum Neural Networks

Since QNNs are meant to be the quantum equivalent of NN, it makes sense that this should also be a supervised learning task with the same problem definition described in the previous section (Problem 1). A QNN falls into a broader category of quantum algorithms known as Variational Quantum Algorithm (VQA). The one common factor of all VQAs is that they are algorithms which make use of some variational paraemeter or parameters $\vec{\theta}$. This is achieved by including gates which are parametrized by continuous variables such as the Pauli X rotation

$$R_x(\theta) = e^{-i\theta\frac{\sigma_x}{2}} = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}$$

where σ_x is the Pauli X matrix. A circuit which makes use of such parametric gates are called PQC. Thus one can describe a VQA as a class of PQC-based algorithms with the goal of properly tuning parameters in order to solve some target task. Quantum Neural Networks are in turn a type of VQA. Similar to classical NN, the target task of a QNN is to minimize some cost function during the training procedure. We will return to the cost function and training procedure, however now we will describe the general structure of a QNN.

From a circuit perspective, we need to start by encoding the classical data into a state of an n qubit quantum system. This is done using an encoding circuit which we denote $S(\vec{x})$, acting on a quantum register initially in the ground state $\left|\vec{0}\right\rangle = |0\rangle^{\otimes n}$. The resulting state is denoted as $S(\vec{x})\,|0\rangle^{\otimes n} = |\phi(x)\rangle$. Once the data has been mapped to a quantum system, we wish to apply the model circuit $U(\vec{\theta})$. This model circuit is also called the *variational ansatz*, which arguably is the most important part of the QNN circuit. The variational ansatz $U(\vec{\theta})$ can be described as a product of L sequentially applied unitary gates.

$$U(\vec{\theta}) = U_L(\theta_L)U_{L-1}(\theta_{L-1})\dots U_2(\theta_2)U_1(\theta_1) = \prod_{i=1}^{L} U_i(\theta_i) \tag{3.1}$$

Once the model circuit is applied to our state of embedded quantum data the resulting state is given by $U(\vec{\theta})\,|\phi(x)\rangle = \left|\Psi(x, \vec{\theta})\right\rangle$. The third step of the QNN is to read out the results of the circuit given some set of parameters. The general procedure is to measure the first of n qubits. Repeated measurements of the first qubit results in some distribution of the qubit being in the state $|0\rangle$ or $|1\rangle$. Using this distribution we can determine the expectation value of the circuit given a set of parameters. In other words by repeatedly measuring the first qubit (in the computational basis) what we are essentially calculating is the expectation value of the σ_z Pauli operator on the subspace of the first qubit. Mathematically we can formulate this as

$$\mathbb{E}(\sigma_z) = \left\langle \Psi(x, \vec{\theta}) \right| U^\dagger(\vec{\theta})(\sigma_z \otimes \cdots \otimes I)U(\vec{\theta}) \left| \Psi(x, \vec{\theta}) \right\rangle . \tag{3.2}$$

The expectation value takes on a value between -1 and 1, and furthermore we can threshold this value in order to yield the binary output, i.e the overall prediction of the model:

$$f(x; \vec{\theta}) = \begin{cases} 1 & \text{if } \mathbb{E}(\sigma_z) > 0 \\ -1 & \text{if } \mathbb{E}(\sigma_z) < 0 \end{cases} \tag{3.3}$$

To summarize a general QNN model is a quantum circuit consisting of four parts. The four parts in order are:

1. Encoding data point $\vec{x}_i$ into a quantum state $|\phi(\vec{x}_i)\rangle$ by applying a unitary gate $S(\vec{x})$ to the ground state $|0\rangle^{\otimes n}$.
2. Variational Ansatz. Apply a parametrized unitary $U(\vec{\theta})$ to the state $|\phi(\vec{x}_i)\rangle$ and generate an output state $\left|\Psi(\vec{x}_i, \vec{\theta})\right\rangle$.
3. Measure the expectation value of a chosen observable. This is done by performing repeated measurements of the circuit. We use the Pauli operator σ_z on the first qubit and 1024 shots, i.e measure each circuit 1024 times and calculate the expectation value from the results. The number of shots may be varied.
4. Post-Processing. We do not do any post-processing however it is important to note that it is possible, if one for example wishes to threshold the output between 0 and 1 instead of -1 and 1, or if one wishes to add a bias term, using the same justification as classical NN i.e if we wish to shift our prediction by some amount.

The choice of the first and second parts, i.e the choice of data encoding and variational ansatze are the most important part of the QNN. The goal of this paper is to determine some choice of encoding and variational ansatz which results in a short-depth QNN circuit

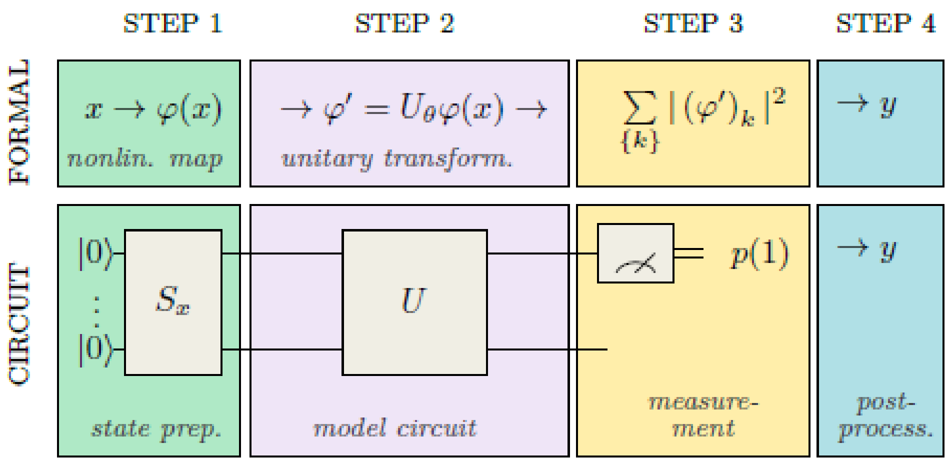

Figure 3.2.1: Visual representation of the four parts of the QNN model. Image taken from [14].

which is viable to run on NISQ devices. Another important aspect is that such a short depth circuit must of course have some degree of viability when it comes to classification accuracy. In the following sections we will discuss the issue with data encoding and our choice for this QNN, as well as looking at various variational ansatze which may be used.

3.3 Data Encoding

The first problem with QNNs is how to map classical data to some quantum mechanical state. Typically we have some vector $x \in \mathcal{X}$ where $\mathcal{X}$ is some input set, and we are looking for the mapping $x \to |\phi(x)\rangle$ where $|\phi(x)\rangle \in \mathcal{F}$ where $\mathcal{F}$ is a Hilbert space which we call the *feature space*. Thus a feature map is a mapping ϕ which takes data from the input space to the feature space, i.e $\phi : \mathcal{X} \to \mathcal{F}$. In a quantum system the feature mapping corresponds to a unitary circuit acting on the ground state, i.e some unitary $S(\vec{x})$ acting on $|0\dots0\rangle$, giving us $S(\vec{x})\,|0\dots0\rangle = |\phi(x)\rangle$. We will discuss some of the most common embeddings currently used and their pros and cons. An overview of the encoding schemes discussed below is given in [34].

3.3.1 Basis Encoding

This is the most straightforward method of embedding classical data to a quantum device. Given some binary input one simply prepares the quantum states accordingly mapping a $0 \rightarrow |0\rangle$ and $1 \rightarrow |1\rangle$. More formally we have

$$S(\vec{x}) : x \in \{0,1\}^n \rightarrow |i\rangle \tag{3.4}$$

where i is the corresponding bit string. An example is if we have input data $x = 0101$ then the quantum embedding is $|0101\rangle$.

Basis encoding is cheap in terms of circuit depth, as it simply requires a single qubit X gate to flip a 0 to a 1, however there is no advantage in terms of the number of qubits required. n-dimensional data requires n qubits to encode the data.

3.3.2 Angle Encoding

The second method is similar to basis encoding, and is called angle encoding. Formally we have:

$$|\phi(\vec{x}_n)\rangle = \bigotimes_{i=1}^{N} \cos(x_i)\,|0\rangle + \sin(x_i)\,|1\rangle \tag{3.5}$$

for some input $x = [x_1, \ldots, x_M]$ [34], and $\vec{x}_n$ denotes the nth data vector in our data set. The state preparation unitary $S(\vec{x})$ can be written as $S(\vec{x}) = \otimes_{j=1}^{N} S_k$ where:

$$S_k = \begin{bmatrix} \cos(x_k) & -\sin(x_k) \\ \sin(x_k) & \cos(x_k) \end{bmatrix} \tag{3.6}$$

Similarly to basis encoding this method requires N qubits to embed N-dimensional data, and also uses a constant depth circuit. One reason to use this method of embedding over basis encoding is that non-binary data can be encoded.

How would the circuit look for this? Simply explaining the one qubit case: We have an $R_Y(\theta)$ gate which can be written as:

$$R_Y(\theta) = \exp\left(-i\frac{\theta}{2}\sigma_Y\right) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix} \tag{3.7}$$

Thus if we apply an $R_Y(2x)$ gate to the ground state we have

$$R_Y(2x)\,|0\rangle = \begin{bmatrix} \cos(x) & -\sin(x) \\ \sin(x) & \cos(x) \end{bmatrix} |0\rangle = \cos(x)\,|0\rangle + \sin(x)\,|1\rangle \tag{3.8}$$

Which is the single qubit case of Eq. (3.5). Thus we can simply encode the data using a single gate. For the several qubit case we still require one gate per qubit, i.e that the gate depth is constant at 1.

3.3.3 Dense Angle Encoding

Dense angle encoding is a method which is similar to angle encoding but a slightly more generalized version where we can embed two features per qubit instead of one by making use of the relative phase degree of freedom. The encoding is defined as:

$$|\phi(\vec{x}_n)\rangle = \bigotimes_{i=1}^{N/2} \cos(\pi x_{2i-1})|0\rangle + e^{2\pi i x_{2i}} \sin(\pi x_{2i-1})|1\rangle \tag{3.9}$$

What gates are needed in order to implement dense angle encoding? The required gates are not clearly stated and thus one of the contributions of this thesis will be to present the gates required for dense angle encoding.

We begin by looking at the simple case with two features, i.e $\vec{x} \in \mathbb{R}^2$ meaning that we only require one qubit for the encoding. We know from earlier that:

$$R_Y(\theta)|0\rangle = \cos(\theta/2)|0\rangle + \sin(\theta/2)|1\rangle \tag{3.10}$$

Furthermore we can see that a single qubit rotation about the z-axis on the Bloch sphere is given by the Phase gate.

$$P(\phi) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix} \tag{3.11}$$

We can thus see that if we apply a Phase gate on the state above we get:

$$P(\phi)R_Y(\theta)|0\rangle = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{bmatrix} \begin{bmatrix} \cos(\theta/2) \\ \sin(\theta/2) \end{bmatrix} = \cos(\theta/2)|0\rangle + e^{i\phi}\sin(\theta/2)|1\rangle \tag{3.12}$$

We see that we if we set $\theta = 2\pi x_1$ and $\phi = 2\pi x_2$ we arrive at the expression for dense angle encoding:

$$|\phi(x)\rangle = P(2\pi x_2)R_Y(2\pi x_1)|0\rangle = \cos(\pi x_1)|0\rangle + e^{2\pi i x_2}\sin(\pi x_1)|1\rangle \tag{3.13}$$

If we wish to visualize our dense encoding scheme in terms of gates in a circuit it would look as follows in Fig 3.3.1 to encode two features to a single qubit.

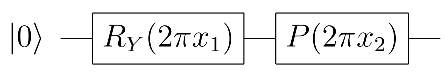

Figure 3.3.1: Dense Angle Encoding for the single qubit case where the input vector $\vec{x}$ only has two features.

For several qubits we would have the following state (which is just a re-writing of Eq. 3.9)

$$|\phi(\vec{x}_n)\rangle = \begin{pmatrix} \cos(\pi x_1) \\ e^{2\pi i x_2} \sin(\pi x_1) \end{pmatrix} \otimes \cdots \otimes \begin{pmatrix} \cos(\pi x_{2i-1}) \\ e^{2\pi i x_{2i}} \sin(\pi x_{2i-1}) \end{pmatrix}. \tag{3.14}$$

the circuit structure would just be a generalization of the same circuit above, shown in Fig 3.3.2.

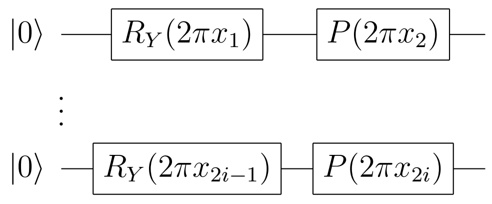

Figure 3.3.2: Dense Angle Encoding for multiple qubits. Can be expanded depending on the dimension of the input data $\vec{x}$.

3.3.4 Amplitude Encoding

In amplitude encoding, the data is encoded into the amplitudes of a quantum state. Given a normalized N-dimensional input vector $\vec{x}$ (Normalization constraint $||\vec{x}|| = 1$), we can represent it by the amplitudes of the $\log_2(N)$-qubit quantum state $|\phi(\vec{x})\rangle$ given by:

$$|\phi(\vec{x})\rangle = \sum_{i=1}^{N} x_i |i\rangle \tag{3.15}$$

where x_i is the ith element of the input vector $\vec{x}$, and $|i\rangle$ is the ith computational basis state. As stated earlier we only require $\log_2(N)$ qubits in order to encode N dimensional data, or another way of saying the same thing is that with n qubits we can encode 2^n dimensional data. If however the data is not 2^n dimensional but say for example $2^n - 1$ dimensional, one can pad the input data with *non-informative* constants, i.e zeros such that the input vector becomes exactly 2^n dimensional.

In theory the idea of amplitude encoding is a great one as we make use of the exponentially growing Hilbert space as we introduce more qubits, meaning that we can encode data with large dimensionality using few qubits. The downside however with this method is that it is not clear exactly how the state preparation routine $S(\vec{x})$ looks. At worst, the gate depth of the state preparation routing is 2^n where n is the number of qubits [14, 34, 35]. This creates issues when one wishes to apply ampltidue encoding strategies with NISQ devices as it is simple to see that there will be fidelity and decoherence issues even for a small number of qubits.

Despite the large advantage that amplitude encoding has over other data encoding methods in terms of the dimension of data which it is able to encode, it has a major disadvantage in terms of the number of gates required. This makes amplitude encoding impractical for use with NISQ devices.

3.4 Variational Ansatze

The variational ansatze is arguably the core of a QNN circuit. The question is now, how should such an architecture look? Should one take an *ad hoc* approach and change the structure of the circuit seemingly randomly until a favourable classification accuracy is reached or is there perhaps some systematic manner in which we can determine the ideal ansatz architecture? There remains a lack of understanding exactly what constitutes a "good" circuit architecture for different purposes.

One proposal is to look at two descriptors of the circuit architectures in order to determine how well they may perform for a certain problem [36]. These descriptors are; **Expressibility** and **Entanglement Capability**, which can be quantified by computing statistical properties based on sampling states from the given circuit architecture.

3.4.1 Expressibility

Expressiblity of a quantum circuit is defined to be "the circuit's ability to generate pure states that are well representative of the Hilbert space". This is simple to visualize in the case of a single qubit, it simply is the circuit's ability to reach any position on the Bloch sphere. One can generalize this mathematically to several qubit quantum circuits by comparing the distribution of states obtained from a given circuit architecture, to the maximally expressive uniform distribution of states known as the ensemble of Haar random states. The expressibility of the circuit is then given by $|A^{(t)}(U)|^2$ where

$$A^{(t)}(U) := \int U^{\otimes t} |0\rangle \langle 0| (U^{\dagger}_{Haar})^{\otimes t} dU_{Haar} - \int U^{\otimes t} |0\rangle \langle 0| (U^{\dagger}_{Haar})^{\otimes t} dU \tag{3.16}$$

A visual demonstration of expressibility is quite simple. If we consider the single

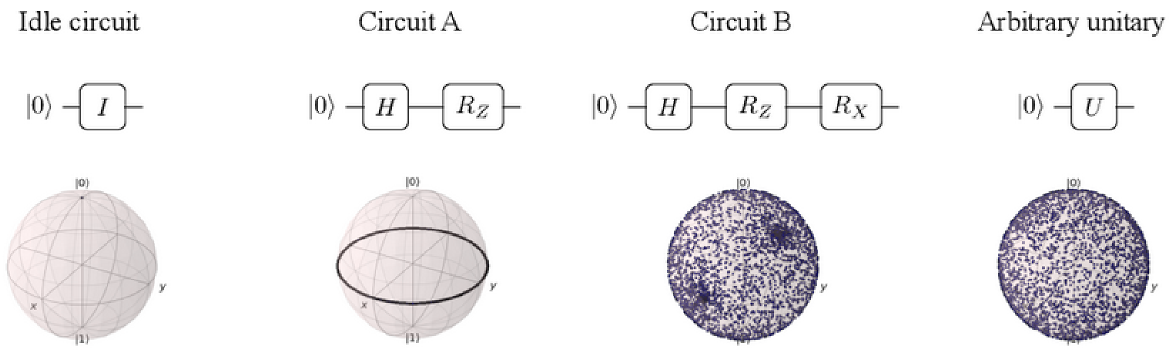

Figure 3.4.1: Visual demonstration of the concept of expressibility with four different quantum circuits in the single qubit case taken from [36].

qubit case, there are several gate combinations which will give us differing values of expressibility. In Fig. 3.4.1 we can see some examples of different circuits with increasing expressibility from left to right. The least expressive case is the circuit with fixed gates that always output the same state such as the I (Identity Gate which does nothing, as shown in the figure) or the X gate. Circuit A shows a circuit where a Hadamard gate is applied followed by a parametrized R_Z rotation gate. This is more expressive because sampling the circuit for different parameters results in a wider variety of states (The states on the equator of the Bloch sphere). Circuit B further improves the expressibility by introducing a parametrized R_X gate which allows us to rotate along another degree of freedom. Finally the most expressible single qubit circuit is simply the arbitrary unitary gate. This is more expressible than the case of Circuit B since if one uniformly samples both circuits, the states of the arbitrary unitary will also be uniform wheras the states of Circuit B will concentrate states at the $+x$ and $-x$ poles.

When expanding this concept for several qubits, it is not immediately clear as in the single qubit case what sort of gate will have a large expressibility. It is clearly important since we want to be able to represent as many quantum states as possible.

3.4.2 Entangling Capability

In the general context of VQA, a natural approach has been to consider circuit architectures which are capable of preparing strongly entangled quantum states, i.e maximally entangled states, using the argument that such circuits "reach wide corners of the Hilbert space". Furthermore there is also evidence that circuits with high entangling capability can capture non-trivial correlation in the data. In order to quantify the entangling capability of a certain circuit architecture the Meyer-Wallach measure is used. The Meyer-Wallach measure can be defined as follows:

$$Q(|\psi\rangle) \equiv \frac{4}{n}\sum_{j=1}^{n} D(\iota_j(0)\,|\psi\rangle\,, \iota_j(1)\,|\psi\rangle) \tag{3.17}$$

where $\iota_j(b)\,|b_1,\ldots,b_n\rangle = \delta_{bb_j}\left|b_1,\ldots,\hat{b}_j,\ldots,b_n\right\rangle$ and $b_j \in \{0,1\}$. The hat symbol denotes the absence of the j-th qubit. Furthermore the generalized distance D is given by:

$$D(|u\rangle\,, |v\rangle) = \frac{1}{2}\sum_{i,j} |u_i v_j - u_j v_i|^2 \tag{3.18}$$

and the states $|u\rangle$ and $|v\rangle$ are a linear superposition of some basis, i.e we have $|u\rangle = \sum u_i\,|i\rangle$ and $|v\rangle = \sum v_i\,|u\rangle$. To show an example of the entangling measure Q we can look at the cases where we have $|\psi\rangle = |01\rangle$ and $|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. In the first case we can see that $Q(|01\rangle) = 0$, and for the second case $Q(\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) = 1$. Thus we see that if our state is a product state the entanglement measure will be equal to zero, and if it is maximally entangled (As the Bell state in the example) then the measure will be equal to 1.

When looking at the entangling capability of a certain circuit architecture, it is defined the be the average Meyer-Wallach entanglement of states generated by the circuit. This is estimated by sampling circuit parameters and computing the Meyer-Wallach average of the output states. Thus a circuit architecture that can only give us product states will give us an entangling capability of 0 and one that always produces highly entangled states will produce a score closer to 1.

3.5 Circuit Training Procedure

Once the QNN circuit structure has been decided, we also need to discuss the training procedure of the QNN. As explained earlier, a classical NN can be trained using gradient descent, where the weights/parameters are updated according to a gradient descent update rule. This is done in order to reach a minima of the cost function. How can we extend this to the quantum domain, or in other words, how can we train the parameters of our QNN? In order to do this we must determine some notion of a derivative of the quantum circuit with respect to each parameter in our QNN. This is known as the Quantum Gradient [37]. Before describing Quantum Gradients we will explain the general training procedure.

Similar to the classical NN, we choose an objective function which we wish to calculate. The simplest choice is the least-squares objective cost function.

$$\mathcal{C}(\vec{\theta};\mathcal{D}) = \frac{1}{2}\sum_{i=1}^{N} |\mathbb{E}(\sigma_z) - y^m|^2 \tag{3.19}$$

where $\mathbb{E}(\sigma_z)$ is the prediction after a measurement (See Eq. (3.2)) and can only take values between -1 and 1, i.e that $-1 \leq \mathbb{E}(\sigma_z) \leq 1$. Due to the limitations of NISQ devices, however a normal gradient descent procedure will be problematic. The problem being that with a normal gradient descent procedure one must go through the entire training data set in order to calculate the cost (Eq. 3.19) and update the parameters $\vec{\theta}$, resulting in a time consuming training step.

We can therefore use *Stochastic Gradient Descent* instead, where we don't consider the entire training set $\mathcal{D}$ in each iteration, but instead look at a single data point per iteration. This is also known as single batch gradient descent where we use a batch $\mathcal{B}$ of size 1. The cost function of each iteration in this case is then given by

$$\mathcal{C}(\vec{\theta};\mathcal{B}) = \frac{1}{2}|\mathbb{E}(\sigma_z) - y^m|^2. \tag{3.20}$$

The cost is minimized by gradient descent which updates each parameter θ_i of the QNN, given by the update rule:

$$\vec{\theta}^{(t)} = \vec{\theta}^{(t-1)} - \eta\frac{\partial\mathcal{C}}{\partial\vec{\theta}} \tag{3.21}$$

where η is the learning rate hyperparameter chosen by the user. Furthermore the derivative of the cost function with respect to the parameters $\vec{\theta}$ for a single data sample is given by

$$\frac{\partial\mathcal{C}}{\partial\vec{\theta}} = (\mathbb{E}(\sigma_z) - y^m)(\partial_\theta\mathbb{E}(\sigma_z)) \tag{3.22}$$

Recalling that $\mathbb{E}(\sigma_z) = \left\langle\Psi(x,\vec{\theta})\middle| U^\dagger(\vec{\theta})(\sigma_z\otimes\cdots\otimes I)U(\vec{\theta}) \middle|\Psi(x,\vec{\theta})\right\rangle$, then the term $\partial_\theta\mathbb{E}(\sigma_z)$

is the derivative of the QNN circuit with respect to the gate parameters $\vec{\theta}$. This is where the notion of a quantum gradient joins the discussion, as we need to define what the derivative of a circuit or gate is, and more importantly we need to determine how it can be procured using NISQ devices.

3.5.1 Quantum Gradients

The concept of a quantum gradient was first proposed in [37], and further optimized [38]. Furthermore an alternative method of calculating the gradient was proposed in [14]. An overview of calculating quantum gradients for discrete-variable circuits and continuous-variable circuits is given in [39]. We will briefly cover the two methods of differentiation, and in the next chapter motivate our choice of method. The two methods are the **Parameter Shift Rule** and the **Linear Combination of Unitaries**.

Parameter Shift Rule

A PQC $U(\vec{\theta})$ consists of a sequence of single-parameter gates (See Eq. (3.1)) where each gate is of the form

$$\mathcal{G}(\mu) = e^{-i\mu G} \tag{3.23}$$

where G is a Hermitian operator. We see that it's derivative is given by

$$\partial_\mu \mathcal{G}(\mu) = -iGe^{-i\mu G} = (-iG)\mathcal{G}. \tag{3.24}$$

For simplicity one can assume that a single gate in the PQC has a parameter, i.e that the PQC unitary is of the form $U(\mu) = V\mathcal{G}(\mu)W$. The derivative w.r.t the parameter μ is then

$$\partial_\mu U(\mu) = \partial_\mu \langle\psi| \mathcal{G}^\dagger \hat{O} \mathcal{G} |\psi\rangle = \langle\psi| \mathcal{G}^\dagger \hat{O} (\partial_\mu \mathcal{G}) |\psi\rangle + \text{h.c.} \tag{3.25}$$

In this case we have absorbed the gate V into the observable such that $\hat{O} = V^\dagger \hat{B} V$, and also absorbed the gate W into the state $|\psi\rangle = W|0\rangle$. Furthermore we can make use of the following lemma

Lemma 1 *For any two operators B and C we have*

$$\langle\psi| B^\dagger \hat{Q} C |\psi\rangle + h.c. = \frac{1}{2}\Big(\langle\psi| (B+C)^\dagger \hat{Q} (B+C) |\psi\rangle - \langle\psi| (B-C)^\dagger \hat{Q} (B-C) |\psi\rangle \Big) \tag{3.26}$$

We can now insert Eq. (3.24) in Eq. (3.25) giving us

$$\partial_\mu U(\mu) = \langle\psi| \mathcal{G}^\dagger \hat{O}(-iG)\mathcal{G} |\psi\rangle = \langle\psi'| \hat{O}(-iG) |\psi'\rangle \tag{3.27}$$

where we absorb the operator $\mathcal{G}$ such that $|\psi'\rangle = \mathcal{G}|\psi\rangle$. The idea is now that if the Hermitian operator G has *two* distinct eigenvalues we can due to the unobservable global phase shift the eigenvalues to $\pm r$ without any loss of generality. We now make use of Lemma 1, setting $B = \mathbb{I}$ and $C = -ir^{-1}\mathcal{G}$ gives us

$$\partial_\mu U(\mu) = \frac{r}{2}\Big(\langle\psi'| (\mathbb{I} - ir^{-1}G)^\dagger \hat{O}(\mathbb{I} - ir^{-1}G) |\psi'\rangle - \langle\psi'| (\mathbb{I} + ir^{-1}G)^\dagger \hat{O}(\mathbb{I} + ir^{-1}G) |\psi'\rangle\Big) \tag{3.28}$$

Furthermore we can make use of the following theorem of which we omit the proof (See ref [39] for proof).

Theorem 2 *If a Hermitian operator G of the unitary operator $\mathcal{G}(\mu) = e^{-i\mu G}$ has at most two distinct eigenvalues $\pm r$ the following identity holds*

$$\mathcal{G}\left(\frac{\pi}{4r}\right) = \frac{1}{\sqrt{2}}(\mathbb{I} \pm ir^{-1}G) \tag{3.29}$$

Thus we can reach the form of the derivative in Eq. (3.28), by shifting the parameters of the quantum gate $\mathcal{G}$ by $\frac{\pi}{4r}$. Thus the deriviative of the quantum circuit w.r.t μ is given by

$$\partial_\mu U(\mu) = r\Big(\langle\psi| \mathcal{G}^\dagger(\mu+s)\hat{O}\mathcal{G}(\mu+s) |\psi\rangle - \langle\psi| \mathcal{G}^\dagger(\mu-s)\hat{O}\mathcal{G}(\mu-s) |\psi\rangle\Big) \tag{3.30}$$

where $s = \frac{\pi}{4r}$. Thus the derivative of a circuit w.r.t a certain gate parameter can be determined by executing two additional circuit measurements on a quantum device. The first measurement where you shift the parameter by $+s$ and the second measurement where you shift the parameter by $-s$, in order to finally determine the derivative for that parameter given by Eq. 3.30.

Furthermore as explained in [37] and [39] if G is the Pauli operators $\frac{1}{2}\{\sigma_x, \sigma_y, \sigma_z\}$ then we have $r = 1/2$ and $s = \pi/2$, meaning that if our PQC consists of rotation gates then we can easily determine the derivative of a given gate (and by doing so we can also determine the gradient) by two additional circuit executions with a macroscopic parameter shift. In circuit form we can show the circuits required in order to determine the derivative of a certain parameter θ of some rotation gate (in this case an R_X rotation). Fig. 3.5.1 shows the two macroscopic shifts required. The important thing to note about the parameter

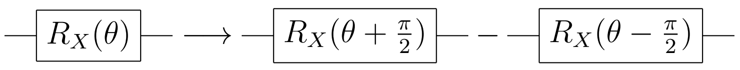

Figure 3.5.1: A parametrized quantum gate, and the two circuits required in order to determine the gradient of the given gate for some value of θ. Note the minus sign which comes from Eq. (3.30).

shift rule is that we can only use it if the single gate only has 2 *distinct* eigenvalues. This is not a problem if we use as described above the Pauli rotation gates, however if we wish to use more general gates parametrized by more than one parameter (which leads to more eigenvalues) we must use another method of determining the derivative of the given gate which we will describe in the coming section.

Linear Combination of Unitaries

As explained above, sometimes the parameter shift rule does not apply. We can circumvent this issue by introducing and ancilla qubit. For finite dimensional systems the derivative of a gate $\partial_\mu \mathcal{G}$ can be decomposesd into a linear combination of unitary matrices A_1 and A_2.

$$\partial_\mu \mathcal{G} = \frac{\alpha}{2}((A_1 + A_1^\dagger) + i(A_2 + A_2^\dagger)) \tag{3.31}$$

One can generalize this to such that

$$\partial_\mu \mathcal{G} = \sum_{k=1}^{K} \alpha_k A_k \tag{3.32}$$

where α_k are real coefficients and A_k are unitary matrices. Using this we can write the derivative of the circuit as

$$\partial_{mu} U(\mu) = \sum_{k=1}^{K} \alpha_k [\langle\psi| \mathcal{G}^\dagger \hat{O} A_k |\psi\rangle + \text{h.c.}] \tag{3.33}$$

The idea is then that one can use Lemma 1 to compute each term in the sum using a linear combination of unitaries $\mathcal{G}$ and $A_k = A$. To do this on a quantum circuit we need to introduce an ancilla qubit in the state $|0\rangle$. Next we apply a Hadamard gate to it in order to obtain a bipartite state

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \otimes |\psi\rangle \tag{3.34}$$

Next we apply the controlled gate $C(\mathcal{G})$ conditioning the ancilla on being in the 0 state, followed by a controlled gate $C(A)$ conditioned on the ancilla being in the 1 state. The resulting bipartite state is

$$\frac{1}{\sqrt{2}}\left[|0\rangle \mathcal{G}|\psi\rangle + |1\rangle A|\psi\rangle\right] \tag{3.35}$$

If we now apply a second Hadamard gate on the ancilla qubit we are left with

$$\frac{1}{2}\left[|0\rangle (\mathcal{G}+A)|\psi\rangle + |1\rangle (\mathcal{G}-A)|\psi\rangle\right] \tag{3.36}$$

Measuring the ancilla qubit will then give us one of two states. Either the result is

$$|\psi_0\rangle = \frac{1}{2\sqrt{p_0}}(\mathcal{G}+A)|\psi\rangle \tag{3.37}$$

$$p_0 = \frac{1}{4}\langle\psi|(\mathcal{G}+A)^\dagger(\mathcal{G}+A)|\psi\rangle \tag{3.38}$$

or the state

$$|\psi_1\rangle = \frac{1}{2\sqrt{p_1}}(\mathcal{G}-A)|\psi\rangle \tag{3.39}$$

$$p_1 = \frac{1}{4}\langle\psi|(\mathcal{G}-A)^\dagger(\mathcal{G}-A)|\psi\rangle \tag{3.40}$$

Next one can measure the observable $\hat{O}$ for the final state reapetedly in order to obtain the expectation values of $\hat{O}$ conditioned on the ancilla qubit. These are

$$\mathcal{E}_0 = \langle\psi_0|\hat{O}|\psi_0\rangle = \frac{1}{4p_0}\langle\psi|(\mathcal{G}+A)^\dagger\hat{O}(\mathcal{G}+A)|\psi\rangle \tag{3.41}$$

$$\mathcal{E}_1 = \langle\psi_1|\hat{O}|\psi_1\rangle = \frac{1}{4p_1}\langle\psi|(\mathcal{G}-A)^\dagger\hat{O}(\mathcal{G}-A)|\psi\rangle \tag{3.42}$$

We can see that from here we can use Lemma 1 in order to calculate $\langle\psi|\mathcal{G}^\dagger\hat{O}A|\psi\rangle + \text{h.c.}$. We see that it is given by

$$\langle\psi|\mathcal{G}^\dagger\hat{O}A|\psi\rangle + \text{h.c.} = 2(p_0\mathcal{E}_0 - p_1\mathcal{E}_1) \tag{3.43}$$

Knowing this we can determine the derivative of the circuit with respect to some parameter. The most time consuming step is to determing the decomposition of $\mathcal{G}$ into a linear combination of unitaries A_k, however once this is found the remainder is much faster than the parameter shift method, as we only need to perform repeated measurements of one set of parameter values for each derivative whereas in the parameter shift method one had to evaluate the circuit for two different parameter shifts. Fortunately the Qiskit SDK has a gradient framework[1] which automatically determines the linear combination of unitaries A_k. Another advantage of this method is that one can determine the derivative of more general gates and gates with more parameters.

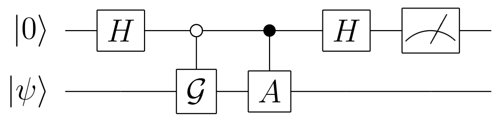

Figure 3.5.2: Circuit diagram detailing the general structure of the Linear Combination of Unitaries method of determining the gradient of a given gate $\mathcal{G}$. The gate A consists of some linear combination A_k which can be determined using the Qiskit Gradient framework.

3.5.2 Optimizer

In classical machine learning there have been several improvements to the stochastic gradient descent optimization step [40]. For our experiments we use one such improvement called momentum gradient descent. This method adds a momentum term to the standard stochastic gradient optimization step, with hyperparameter m. This leads to a faster convergence to the cost minima, but adds one hyperparameter which requires tuning. We choose to use this method as it is a good trade off between classification accuracy versus the number of hyperparameters. The optimization step is shown below.

$$\begin{aligned} \theta^{(t)} &= \theta^{(t-1)} - a^{(t)} \\ a^{(t)} &= m a^{(t-1)} + \eta \frac{\partial \mathcal{C}(\vec{\theta}, \mathcal{B})}{\partial \theta} \end{aligned} \tag{3.44}$$

Furthermore one can choose a variety of optimizers which have had good result on classification problems with classical NN. One optimizer which has shown good results for classical problems is the Adam optimizer, which has an adaptive learning rate and stores exponentially decaying averages of past squared gradients [41]. This optimizer has

[1]See https://qiskit.org/documentation/locale/de_DE/tutorials/operators/02_gradients_framework.html

three hyperparameters which require tuning, and the optimization step is given by

$$\theta^{(t)} = \theta^{(t-1)} - \eta^{(t)} \frac{a^{(t)}}{\sqrt{b^{(t)}} + \epsilon} \tag{3.45}$$

$$a^{(t)} = \frac{\beta_1 a^{(t-1)} + (1-\beta_1)\nabla \mathcal{C}(\vec{\theta}, \mathcal{B})}{(1-\beta_1^t)} \tag{3.46}$$

$$b^{(t)} = \frac{\beta_2 b^{(t-1)} + (1-\beta_2)(\nabla \mathcal{C}(\vec{\theta}, \mathcal{B}))^{\odot 2}}{(1-\beta_2^t)} \tag{3.47}$$

$$\eta^{(t)} = \eta^{(t-1)} \frac{\sqrt{(1-\beta_1^t)}}{(1-\beta_2^t)} \tag{3.48}$$

The hyperparameters being β_1,β_2 and ϵ and $(\nabla \mathcal{C}(\vec{\theta}, \mathcal{B}))^{\odot 2}$ denotes the elementwise square of the gradient.

Chapter 4

Methodology

The goal of this degree project is to propose a QNN circuit structure and method can be run on NISQ devices. In Section 2.1.4 we discussed the challenges which current NISQ devices face. The most acute problem being decoherence of qubits. Thus in order to reduce decoherence errors of qubits the circuit depth must be kept to a minimum.

The contents of this chapter will cover the reasoning behind the choices we make regarding the QNN structure, the experimental setup and the choice of NISQ device.

4.1 Circuit Construction Choices

In Section 3.3 we introduced different methods of encoding classical data to quantum systems using an encoding circuit $S(\vec{x})$. Each method has it's advantages and disadvantages, such as the numbeer of features which may be encoded using a given number of qubits, or the gate depth of the encoding circuit $S(\vec{x})$. Furthermore in Section 3.4 we introduced two descriptors of a given variational ansatz architecture, the expressibility and entangling capability. While these descriptors impact how *good* a QNN may perform, we must also keep in mind the limitations of NISQ devices. As stated above the chief issue is that we need to minimize the circuit depth and find some optimal short-depth circuit, i.e the QNN circuit which gives us the best classification accuracy on the test set. In the following section we will describe the choice of encoding circuit $S(\vec{x})$ and variational ansatz architecture $U(\vec{\theta})$ keeping these ideas in mind.

4.1.1 Choice of Encoding

In the scope of this thesis we have chosen to use a single encoding scheme, the Dense Angle Encoding (See Section 3.3.1) because it offers the best trade off between the number of features encoded and the circuit depth of the encoding circuit $S(\vec{x})$. The encoding circuit has constant depth 2, and also each qubit encodes 2 features. This then means if we have an N-dimensional input vector $x = (x_1, \ldots, x_N)$, then $N/2$ qubits are required. With current NISQ devices this means that the dimension of our feature vectors must be limited, as the number of qubits are limited.

Basis, and angle encoding are not good options because we encode single features per qubit. Amplitude encoding may seem to be a better option at first glance due to the fact that we can encode 2^N features using N qubits, however the crux of using this scheme is that the circuit depth of the encoding circuit $S(\vec{x})$ in this case is of the order 2^N[34] as explained in Section 3.3.4. This immediately disqualifies the use of Amplitude encoding for our QNN as the depth of our circuit will be too large for use with NISQ devices. Thus the best compromise is Dense Angle Encoding.

4.1.2 Choice of Variational Ansatze

The choice of the variational ansatz is interesting as we wish to find some short-depth circuits which exhibit high expressibility and entanglement capacity. There have been investigations of common ansatze which look at precisely the expressibility and entanglement capability [36]. From this investigation we can look at which circuit architectures are rank highest in these two descriptors while also paying attention to the circuit depth.

Ansatz #1

One potential circuit structure is shown in Fig. 4.1.1 corresponding to Circuit 14 from [36]. This circuit is called the *shifted-circular alternating* ansatz, and can be generalized to be constructed by blocks where we have a layer of single qubit unitaries R_Y followed by a layer of controlled unitary gates CR_X. It consists of circular entanglement where the entanglement connecting the first with the last qubit is shifted by one to the right in each block. Furthermore the role of control and target qubits are swapped every block (therefore alternating). This is a good candidate for the QNN structure as we can

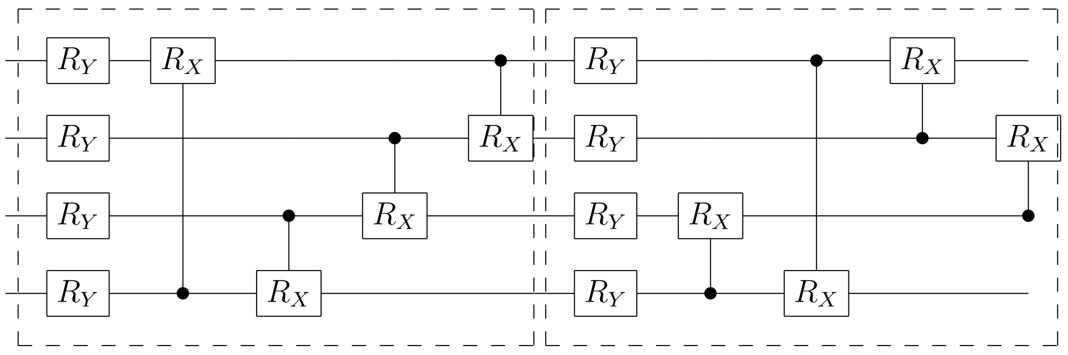

Figure 4.1.1: Circuit 14 from [36]. May be broken down into two code blocks and further generalized if one wishes to add more blocks. We denote the first block as block 1A and the second bloch as block 1B

expand or contract the depth of our circuit. We can also expand the ansatz to include more qubits than the 4 qubits depicted in the image.

It is also a good choice based on the expressibility and entangling capability constraints. Of all the circuits investigated in [36], this one has the second highest expressibility (The circuit with the best expressibility has a much larger circuit depth) and sixth best entangling capability when looked at for the 4 qubit case.

Furthermore, as it is simple to generalize the circuit structure in Fig. 4.1.1, we can also use a block of this circuit and combine it with another circuit architecture. For clarity in later parts of this thesis we denote the entire ansatz as shown in Fig 4.1.1 as **Block 1**, and we can further break it down and denote the first block of the ansatz (Dashed lines on the left) as **Block 1a** and the other block as **Block 1b**.

Ansatz #2

The second ansatz circuit architecture which is interesting is given by Circuit 19 depicted in Fig. 4.1.2. This architecture is very similar to the architecture shown in Ansatz #1, with the difference being that there is an extra layer of single qubit unitaries. In this ansatz we have a layer of R_X rotation gates followed by another layer of R_Z rotation gates, before completing the circuit with the controlled two qubit rotation gates. This circuit ansatz does not rank as high as the previous one in terms of the expressiblity and entangling capability descriptors, however it is a short-depth circuit which we can either extend by adding another block at the end of the controlled two qubit rotation gates, or we can combine this gate with another ansatz architecture such as a block from Ansatz

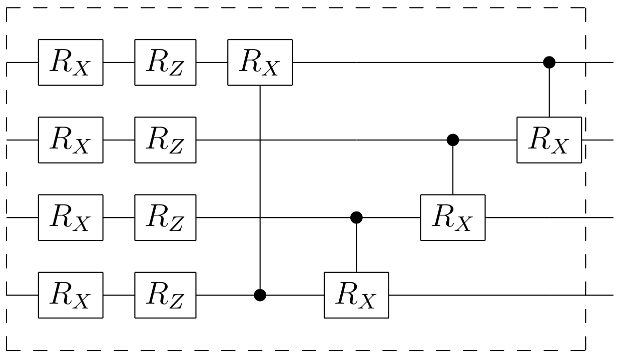

Figure 4.1.2: Circuit 19 from [36]. This ansatz is less expressible than block 1.

#1. For the purposes of this thesis we denote this ansatz simply as **Block 2**.

Ansatz #3

The third and final ansatz circuit architecture is Circuit 4 which is depicted in Fig. 4.1.3. Once again we have a similar architecture as in ansatz #1 and #2, with a layer of single qubit unitaries followed by a layer of controlled two qubit unitaries. This circuit architecture is, like ansatz #2, short depth. Therefore we can use repeated layers of this architecture or interweave it with the other ansatze. This ansatz is denoted as **Block 3** which we refer to later when deciding ansatzes to test.

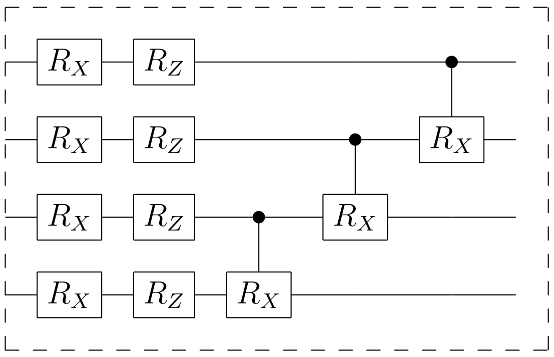

Figure 4.1.3: Circuit 4 from [36]. This is the least expressible ansatz of the three which we are exploring.

4.2 Quantum Gradient & Optimizer Choices

In this section we will discuss the choice of quantum gradient and optimizer. In Sections 3.5.1 and 3.5.2 the quantum gradient and optimizers were introduced respectively. To summarize, there are two methods of determining the quantum gradient (analytically), which depend on what sort of gates our PQC constists of. With our choices of ansatze we have no limitation as to which quantum gradient calculation method we must choose. This is due to the fact that all of our gates consist of single or controlled Pauli rotation gates. Thus we can use either the Parameter Shift Rule or the Linear Combination of Unitaries method. Since in theory both methods should result in the same gradient result, we can ask ourselves which method is most efficient in terms of execution time?

While the Parameter Shift Rule requires two additional executions of the same circuit with a macroscopic shift of a parameter, the Linear Combination of Unitaries method requires only one additional execution of a similar circuit with an ancilla qubit. Thus if one can spare an extra qubit as an ancilla, using the Linear Combination of Unitaries to determine the quantum gradient should be the fastest method. Using this reasoning we chose to use the Linear Combination of Unitaries method for our experiments.

For this thesis we chose to use standard gradient descent and also the momentum optimizer (Eq. (3.44)), in order to determine if there is some improvement in the quantum setting, as there is in the classical one. We decided against using other optimizers such as the Adam optimizer also introduced in Section 3.5.2, due to the fact that training and testing is a lengthy process and that these other optimizers have more hyperparameters. This means that finding ”good” hyperparameters becomes time comsuming.

4.3 Experimental Setup

Once the QNN structure, gradient calculation and optimizer has been chosen we wish to demonstrate that the QNN classifier works. We also wish to determine if there is some advantage compared to classical NNs. For the purpose of this degree project we aim to classify the MNIST data set. The problem definition is the same as described in Section 3.1 (Problem 1), where we want to train the QNN such that it is able to predict the output y of a new input x which it has not seen earlier. In this case that means that we wish to train the QNN on MNIST images such that if we introduce a new image x that it can correctly classify (y) the image.

The first issue which we need to resolve with the MNIST dataset is to classically pre-process our data such that we can actually use it on a NISQ device. As stated in Section 4.1.1 when using Dense Angle Encoding we can only encode two features per qubit. The MNIST dataset consists of 28×28 vectors, which when flattens results in a 784 dimensional input vector. In order to encode this we would require 392 qubits, which is not feasible with NISQ devices, which generally have between $5-50$ qubits. Clearly some pre-processing in the form of dimensionality reduction needs to be performed before the data is compatible with NISQ devices.

4.3.1 Step 1 - Pre-Processing data

First we must convert the multiple class classification problem of the MNIST data set to a binary classification problem. This is easily achieved by choosing two different numbers which we wish to classify. We choose the digits 0 and 1, and filter out the remaining numbers from the data set. Next we randomly choose a subset of 2000 images for our training set and 500 for our test set (No overlap between the sets).

As described above, we now wish to reduce the dimension of the images from 28×28 to something more manageable. We aim to reduce the images to 4×4 because this means that the flattened input vector will be 16 dimensional requiring 8 qubits to encode the data. We follow the method proposed in [42] by first removing 6 pixels from each edge of the image resulting in a dimensional reduction from 28×28 to 16×16. The idea behind this is that if one looks at the MNIST images, most of the white pixels are in the center of the image and thus the values close to the edges will tend to be zeros. Next we remove every other row and column of our 16×16 image. This results in an 8×8 image. Finally we use Bilinear Interpolation using the Tensorflow command `tensorflow.image.resize` in order to further reduce the dimension to 4×4. We can see in Fig. 4.3.1 that when

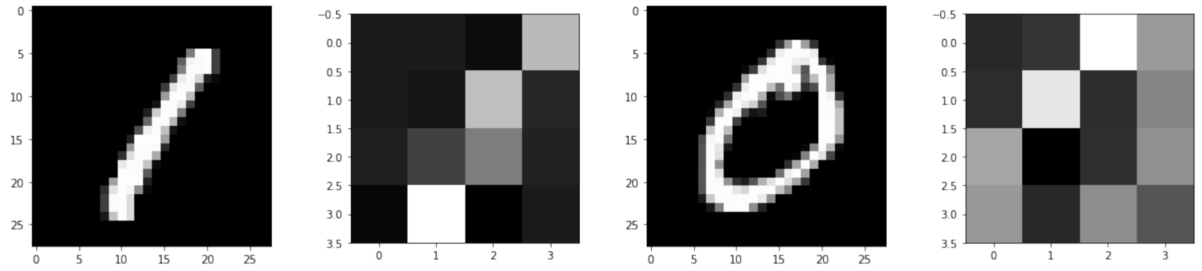

Figure 4.3.1: Examples of figures from the MNIST data set before and after we reduce the dimension from 28×28 to 4×4. There is an example of a 1 (left) and 0 (right) before and after dimension reduction.

using this scheme to reduce the dimension, we are left with 4×4 images that for the human eye, one can clearly still see that the one is a one and the zero is a zero. The reason for using this scheme to reduce the dimension instead of simply using Bilinear Interpolation using `tensorflow.image.resize` on the original 28×28 images is that we retain more information using this method. See Fig. 4.3.2 where we resize the same images. It is clear that only using bilinear interpolation results in more information loss as we decrease the dimension to 4×4.

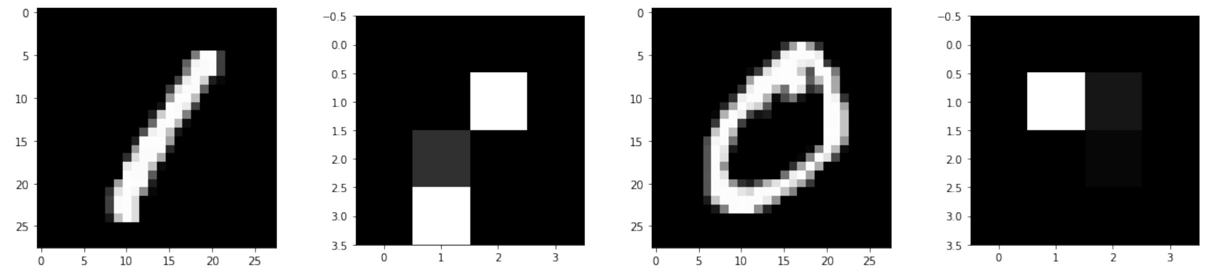

Figure 4.3.2: Examples of figures from the MNIST data set before and after we reduce the dimension from 28×28 to 4×4 using only bilinear interpolation.

4.3.2 Step 2 - Circuit Preparation

Once images have undergone pre-processing to reduce their dimension we are left with a training data set with $n_{\text{train}} = 2000$ samples, and a test set with $n_{\text{test}} = 500$ samples where each sample is a flattened 16 dimensional vector $\vec{x}$. The next step is to then create the QNN circuits which we will be training. As mentioned in Section 4.1.1 we encode each sample using Dense Angle encoding, and append the variational ansatz to each circuit. We are left with 2000 training circuits and 500 test circuits. Figure 4.3.3 shows the encoding circuit of the first sample in the training data set. One circuit is prepared per sample, each with differing values of the parameters depending on the values of the input vector $\vec{x}$.

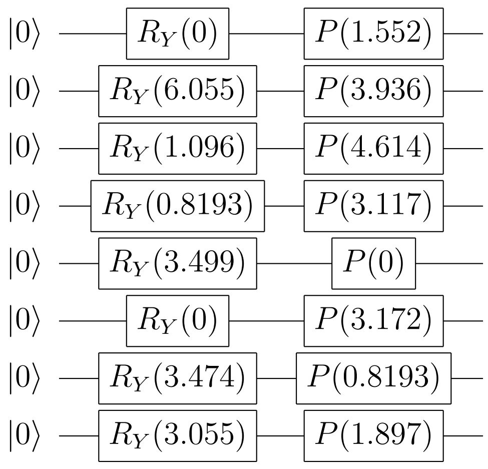
$|0\rangle$ $R_Y(0)$ $P(1.552)$
$|0\rangle$ $R_Y(6.055)$ $P(3.936)$
$|0\rangle$ $R_Y(1.096)$ $P(4.614)$
$|0\rangle$ $R_Y(0.8193)$ $P(3.117)$
$|0\rangle$ $R_Y(3.499)$ $P(0)$
$|0\rangle$ $R_Y(0)$ $P(3.172)$
$|0\rangle$ $R_Y(3.474)$ $P(0.8193)$
$|0\rangle$ $R_Y(3.055)$ $P(1.897)$

Figure 4.3.3: Encoding circuit of the first sample in the training data set.

4.3.3 Step 3 - Training

IBM's NISQ devices are available for use via the Qiskit SDK, however a queue based system is implemented meaning that for each job we wish to send to a NISQ device we must wait in a queue behind jobs sent by other researchers. Long job queues are not uncommon, therefore implementing an iterative stochastic gradient descent algorithm on a real NISQ device becomes a challenge. The issue being that depending on the choice of quantum gradient calculation we must perform between $1-3$ jobs per sample. If one has a large training set then the time required for the training process will not be feasible. Due to this constraint, we decide to locally simulate the training/testing process using Qiskit's `QasmSimulator`. Locally simulating quantum systems is becomes slow as the number of qubits which we wish to simulate increases, however in our case with $n = 8$ qubits the simulation will be comparatively faster than if we were to queue jobs with a real NISQ device.

Similar to training classical NN, the procedure for training the QNN is an iterative process. The training process can be summarized as:

1. Evaluate expectation value of the quantum circuit of sample m with a set of parameters $\vec{\theta}$.
2. Calculate the cost $\mathcal{C}(\vec{\theta}, \mathcal{B})$ using the prediction and label y^m of the sample.
3. Evaluate quantum gradient of the given quantum circuit.
4. Update the parameters $\vec{\theta}$ using the chosen optimizer.
5. Repeat from step 1. with the next sample $m + 1$ until no more samples.

4.3.4 Step 4 - Testing

After the training procedure we are left with a set of parameters $\vec{\theta}_{\text{opt}}$ that in theory should minimize the cost function $\mathcal{C}(\vec{\theta}, \mathcal{B})$, or in other words for us it should be able to generalize and predict if an MNIST image is a zero or a one. We test the model by showing it a test set of new images which it must attempt to classify. The testing procedure can be summarized as:

1. Evaluate expectation value of the quantum circuit of sample m with the ideal parameters $\vec{\theta}_{\text{opt}}$.

2. Threshold the expectation value in order to determine the prediction $f(x; \vec{\theta})$ (See Eq. (3.3)).

3. Compare prediction $f(x; \vec{\theta})$ with sample label y^m. If same then the QNN has correctly classified the sample.

4. Repeat from step 1. with the next sample $m + 1$ until no more samples.

5. Calculate classification accuracy. This is simply given by

$$\text{Classification Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}} \times 100.$$

We can repeat Steps $1 - 4$ (Sections 4.3.1-4.3.4) with the same training and testing data but with different variational ansatz architectures, in order to determine what type of ansatz gives the best performance (in terms of classification accuracy). We choose a combination of ansatze with the goal of not allowing the total gate depth of each circuit ansatz combined with the encoding gates overreach 20.

The different ansatze and ansatze combinations used in our tests are depicted in Table 4.3.1, along with the number of parameters of each ansatz and the circuit depth for the 8 qubit case. The 8 qubit case is chosen because in the circuit preparation step we require 8 qubits to encode our pre-processed MNIST data.

Variational Ansatz	Total Circuit Depth incl. encoding	Number of Parameters
1	20	32
1a + 2	21	40
1a + 3	20	39
1a	11	16
1b	11	16
2	12	24
3	11	23
2+3	21	47
3+3	20	46

Table 4.3.1: The different ansatze used in the potential QNN structure, corresponding with the total circuit depth of each ansatze including the encoding routine of data (Dense angle encoding), and the total number of parameters which are trained.

Chapter 5

Results

In this chapter we will present the results of our experiments which were explained in Chapter 4. We begin by presenting the results of the QNNs with differing variational ansatze using standard stochastic gradient descent, followed by the results of the same ansatze, but with the momentum optimizer instead. We will also present results using a variety of hyperparameters for each method.

The results are given in terms of the classification accuracy of the test set consisting of 500 samples as earlier mentioned. The different variational ansatze which we investigate are shown in Table 4.3.1 in the previous section. These choices of ansatze are by no means exhaustive, and there may be other better suited ansatzes, however we chose these ansatze based on their expressibility, entangling capabilities and circuit depths.

5.1 Comparison of Optimizers

The main purpose for performing experiments using the standard stochastic gradient descent optimization step (See EQ. (3.21)) is for us to be able to compare the test accuracy when we later train using the momentum optimizer. In other words we wish to have a baseline to compare with, and see if choosing another optimizer can yield more favourable results as is the case in classical machine learning. In theory we would expect more favourable results, as other optimizers (such as momentum) converge to minima in the cost landscape faster.

In Table 5.1.1 we show the classification accuracy on the test set using the Block 1 (Ansatz #1) as the variational ansatz, and the standard stochastic gradient descent optimization step, the momentum optimizer, and also the Adam optimizer. The point is that while in theory the Adam optimizer may work better than the momentum optimizer, one must first determine a ”good” set of hyperparameters which is time consuming from a training point of view. This reasoning is why we decide the use the momentum optimizer moving forward, as it may be easier to determine ”good” hyperparameters, since there are fewer of them for the momentum optimizer.

Variational Ansatz	Total Circuit Depth incl. encoding	Number of Parameters	Test Accuracy Standard	Test Accuracy Momentum	Test Accuracy Adam
1	20	32	71.53%	80.18%	73.14%

Table 5.1.1: Performance results for the same variational ansatz QNN using three different optimizers. Also included are the total circuit depth and number of parameters in the variational ansatz. The hyperparameters were as follows. For standard SGD learning rate $\eta = 0.05$. For momentum optimizer, $\eta = 0.05$, $m = 0.9$. For Adam optimizer, $\beta_1 = 0.9$, $\beta_2 = 0.99$, $\epsilon = 10^{-8}$.

5.2 Results using Momentum Optimizer

As stated in the previous section we perform the remainder of the experiments using the momentum optimizer. We performed the experiments according to the experimental setup explained in Section 4.3. The various ansatzes tested were shown in Table 4.3.1. We present the test results in Table 5.2.1.

Variational Ansatz	Total Circuit Depth	Number of Parameters	Test Accuracy
1	20	32	80.18%
1a + 2	21	40	78.8%
1a + 3	20	39	77%
1a	11	16	49%
2	12	24	62.2%
3	11	23	59.8%
2+3	21	47	61.8%
3+3	20	46	61%

Table 5.2.1: Experimental results of the various QNN variational ansatze. Each QNN was trained and tested on the same data using the same hyperparameters, $\eta = 0.05$ and $m = 0.9$.

Some observations that can be made of the results is that the variational ansatz given by block #1 gives the best test accuracy of 80.18%, and also that this was the Block with the highest expressibility and entangling capability of the three blocks/ansatze which we decided to investigate. We also note that when combining part of block #1 (The first sub-block #1a) and one of the other ansatze blocks (such as block #3) we achieve a slightly lower accuracy, despite having an increased number of parameters. Note also the results in which we use two layers of block #3 (The ansatz with lowest expressibility and entangling capability of the three), we have a QNN with the same depth as the previous two discussed, but with more parameters (46 versus 32 and 39 respectively). Despite having the same circuit depth, and more parameters the test accuracy is much lower than the previous two at 61%. This may indicate that increasing the number of parameters does not necessarily lead to better performance, and it may increase training time as the gradient becomes larger requiring more derivative calculations to take place. It is also worth noting that two layers of block #3 have a lower expressibility than a full block #1 (See Figure 3 in [36]), which may affect the performance.

Furthermore we can note that when using only the smaller blocks such as block #1a, 2 and 3, we have a much smaller circuit depth, however the performance of such models is also not ideal (Test accuracy in the $50 - 60\%$ range). This may be due to the fact that the smaller blocks have a much lower expressibility than the larger ones. We also see in this case that of the small blocks, the block with highest expressibility performs the best. In this case block #2 has a higher expressibility than block #3 and #1a, and block #3 has a higher expressibility than #1a. Therefore from our experiments we can note that a variational ansatz with higher expressibility results in a better classification accuracy.

To conclude we can state that the combination of highly expressive ansatze combined with dense angle encoding yields a QNN with a circuit depth such that it is compatible for use with NISQ devices. Furthermore as noted, increasing the number of parameters does not necessarily result in a better classification accuracy. Thus when designing QNNs one can use a variational ansatz with high expressibility and fewer parameters, and apply the linear combination of unitaries gradient calculation method in order to fast compute the gradients, in turn speeding up the training process.

5.3 Varied Hyperparameters

We took the best performing ansatz from the previous section (Block 1) and varied the hyperparameters in order to determine if we could tune them, and in turn improve classification accuracy. The two hyperparameters which can be tuned are, the learning rate η and the momentum term m (See Eq. (3.44)). We decide against a grid search due to time constraints, and instead use values which may intuitively make sense, such as using a smaller learning rate because we suspect that we are overshooting the optima of the cost function. Table 5.3.1 shows the different hyperparameters we used together with the Block 1 ansatz, along with the classification accuracy on the test data. We note

Learning Rate , η	Momentum, m	Test Accuracy
0.05	0.9	80.18%
0.01	0.9	80.6%
0.01	0.99	63.4%
0.01	0.8	79%
0.01	0.7	66.8%
0.005	0.8	63.8%

Table 5.3.1: Experimental results of the Block 1 variational ansatz QNN, with varying hyperparameters.

a slight increase in test accuracy when lowering the learning rate η from 0.05 to 0.01. We also notice that while it is harder to find good hyperparameters, it is fairly easy to find bad ones, as for example a lower momentum term can results in an approximately 14 percentage point reduction in the test accuracy.

5.4 Discussion

Our short-depth QNN structure shows some promising results, as we have managed to achieve satisfactory accuracy on the test data in some cases. While the results are promising achieving an accuracy of 80.6% on the test set, when compared to classical NN, our results fall short as classical NN models and their evolutions (Convolutional Neural Networks for example) can reach a classification accuracy of 99% [43]. It is important to note that while classical NN can have better results, the experimental setups in other literature may differ from our setup. For example it may not be fair to compare the classification accuracy of a classical NN method that does not have the same number of tuneable parameters as our model. Another important distinction to make is that the training and test data set used in this thesis are comparatively small, and results may be improved by training on a larger training set. Therefore in order to accurately compare our results with the classical counterpart one should create a NN with an equal number of parameters, and traing/test it on the same data set.

5.4.1 Comparison with previous works

Comparing our QNN to classical NN is of course important, as we wish to determine if the quantum equivalent of a NN actually has or can have an advantage over classical methods. However it is equally important to compare our QNN variation with other QNN classifiers presented in previous works. By doing so we may hopefully highlight some of the strengths and weaknesses of our QNN. Just as in classical machine learning the MNIST data set is also popular for use within quantum machine learning, and therefore there are previous works which have attempted to create QNNs to classify this data set [14, 15, 42].

As discussed in the previous section, the experimental set ups will differ from paper to paper, and therefore the comparison of results may not be completely proper, however we may still be able to draw some general conclusions.

In [15] the authors downsample the MNIST data set to 4×4 images (as we do), and identify the digits 3 and 6, whereas we chose to identify the 0 and 1. The encoding scheme used is not stated in the paper, however it is stated that 16 qubits are used for the experiments. Furthermore the choice of variational ansatz consists of three layers of parametrized ZX and three layers of parametrized XX gates. In total the circuit depth of the variational ansatz of this method is at least 96 (also the total number of parameters). Furthermore these are all two qubit gates which are not natively implemented in most quantum circuits, but rather consist of some decomposition of quantum gates and therefore it may be logical to assume that the actual circuit depth if implemented on a real NISQ device would be much larger. By inspection of the variational ansatz architecture it becomes clear that such a QNN would not perform well on NISQ devices, however in the authors simulations the QNN achieves "two percent categorical error", i.e a classification accuracy of 98%.

In [14] the authors explicitly state that the goal is to create a low-depth circuit for classification. Instead of converting the MNIST data set to a strictly binary classification problem, the authors take a "one-versus-all" approach. The encoding method of choice used by the authors is Amplitude Encoding (Section 3.3.4), and as they mention in the paper encoding 1000 features into 10 qubits requires at least 2048 gates. The authors first downsample the MNIST images from 28×28 to 16×16, next they encode these images into 8 qubits using amplitude encoding. The variational ansatz used is very similar to Ansatz #1 used in this thesis, with the only difference being that instead of single parameter rotation gates, the authors use general unitary gates parametrized by three angles. Thus a circuit with the same depth as ours used in Ansatz #1 can have three times as many parameters. One important aspect however is, similarly to the previous paper, that these general unitary gates are not natively implemented in most NISQ devices meaning that a general unitary consists of some decomposition of gates which will affect the circuit depth. For example in Qiskit a general unitary gate is implemented using five seperate rotation gates [1]. The authors use three different versions of the same ansatz, where they have one, two and three blocks, and present the most favourable results. The authors best results achieved a 3.3% test error, i.e a 96.7% test accuracy. Just as with the previous paper however it is clear that by using amplitude encoding and general unitary gates in the variational ansatz that the circuit depth would be too large for any practical use on NISQ devices.

The final example of a previous work which uses the MNIST data set for classification purposes is [42]. We have used the same pre-processing method as proposed in this paper, however the authors of this paper do not use bilinear interpolation to reduce the image size from 8×8 to 4×4 as we do. The authors use amplitude encoding as the encoding method, meaning that for $8 \times 8 = 64$ dimensional feature vectors they require 6 qubits (Since $2^6 = 64$)[2]. The variational ansatz differed from the previous papers, and the authors decided to create an ansatz consisting of layers of Pauli rotations interleaved with CNOT (Controlled X gates) ladders. The ansatz is shown in Fig. 5.4.1.

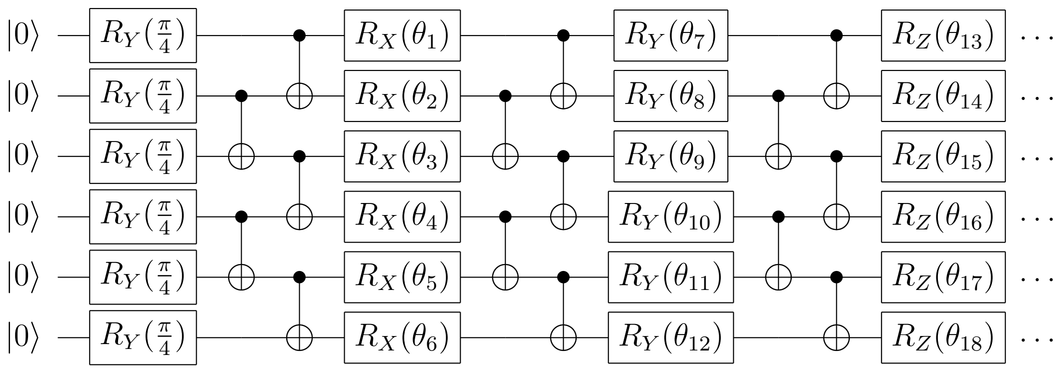

Figure 5.4.1: Variational ansatz used in [42]. The interleaved pattern continues until the ansatz has a total of 400 parameters θ_i.

As with the other two papers which we have already discussed, the authors of this paper also achieve a high test accuracy of approximately 96%, however we can see both from

[1]See https://qiskit.org/documentation/stubs/qiskit.circuit.library.U3Gate.html.
[2]Note that there is an error in the paper which states that 8 qubits are used.

the choice of encoding and variational ansatz that the circuit depth is too large to be compatible with NISQ devices.

In comparison to the three papers described above, our QNN does not achieve the same level of test accuracy. This may be due to a wide variety of factors, such as differing experimental setups (Reducing dimension of images or the number of training/test samples) or due to other choices such as the choice of encoding, variational ansatz or optimizer. One reason may be the fact that for larger variational ansatze one has more trainable parameters which may result in better performance. Despite not reaching the same level of test accuracy, our QNN architecture achieves a good accuracy on a limited number of trainable parameters and most importantly, using a short depth circuit. The major strength of our QNN compared to the other models described above is that ours is compatible for use with NISQ devices.

Chapter 6

Conclusion

6.1 Future Work

While our method does not surpass classical NN or other QNN models in terms of performance, it does present a QNN structure compatible with NISQ devices. Due to limitations in the queue based system of IBM's quantum chips we were not able to test the QNN architecture on an actual device, so thus far it is only compatible in theory due to it's short circuit depth. Further investigations may be conducted on an actual NISQ device if they are made avaiable for lengthier bookings in the future. Otherwise another option for testing the QNN model in more realistic settings may be to perform simulations using noise models belonging to certain IBM quantum chips.

Furthermore both the choice of encoding, and variational ansatz warrant further research. If alternative ways of encoding classical data to qubits, become available then perhaps one can determine a method of encoding more than two features per qubit as we use in Dense Angle encoding. The preferrable method would be to determine some short depth encoding scheme for Amplitude encoding as one would then be able to make use of the exponentially increasing Hilbert space of several qubits. This would eliminate the need to reduce the dimensionality of our input classical data, and could lead to favourable results. Furthermore the choice of variational ansatz can be investigated further. For this thesis we chose our ansatze under the premise that a high expressibility and entangling capability should result in a well performing QNN. This relationship between classification accuracy and expressibility/entangling capability should be investigated further especially since there has been recent evidence that a higher expressibility may lead to barren plateaus (Flat cost landscape meaning that the QNN may converge in “bad” local optima) [44].

Another potential area of further investigation is to repeat the simulations done in this thesis with a more advanced choice of optimizer, for example the Adam optimizer described in Section 3.5.2, and performing a grid search of hyperparameters. Thus one can determine the optimal hyperparameters for this classification problem with the given QNN architecture.

6.2 Final Words

To conclude, we can state that in this thesis we have presented a QNN architecture methodology which is compatible with NISQ devices by taking into account the circuit depth. The constraint of short circuit depths has permeated into every choice made for the QNN. The result is a few short depth QNN structures (same encoding routing, with different variational ansatze), which have mixed performances on the test accuracy. By comparing our work with the previous works we can see that there is a trade off between the classification accuracy and circuit depth (80.6% accuracy at depth 20, versus 96% at an approximate depth of 200 for example). The QNN structure can and should be investigated further as our investigation shows that this type of short depth QNN can have promising classification results. Another interesting direction would be a Scalable QNN architecture with performance that scales with circuit depth and number of qubits. Furthermore our results show that higher expressibility can lead to better performance, and thus the relationship between expressibility and trainability is an interesting topic which should be studied further.

Bibliography

[1] Feynman, Richard P. "Simulating physics with computers". en. In: *International Journal of Theoretical Physics* 21.6-7 (June 1982), pp. 467–488. ISSN: 0020-7748, 1572-9575. DOI: 10.1007/BF02650179. URL: http://link.springer.com/10.1007/BF02650179 (visited on 06/22/2021).

[2] *IBM Quantum*. 2021. URL: https://quantum-computing.ibm.com/.

[3] *Rigetti Systems*. 2021. URL: https://qcs.rigetti.com/qpus/.

[4] Preskill, John. "Quantum Computing in the NISQ era and beyond". en. In: *Quantum* 2 (Aug. 2018), p. 79. ISSN: 2521-327X. DOI: 10.22331/q-2018-08-06-79. URL: https://quantum-journal.org/papers/q-2018-08-06-79/ (visited on 06/22/2021).

[5] Farhi, Edward, Goldstone, Jeffrey, and Gutmann, Sam. "A Quantum Approximate Optimization Algorithm". In: *arXiv:1411.4028 [quant-ph]* (Nov. 2014). arXiv: 1411.4028. URL: http://arxiv.org/abs/1411.4028 (visited on 06/22/2021).

[6] Grover, Lov K. "A fast quantum mechanical algorithm for database search". en. In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC '96*. Philadelphia, Pennsylvania, United States: ACM Press, 1996, pp. 212–219. ISBN: 978-0-89791-785-8. DOI: 10.1145/237814.237866. URL: http://portal.acm.org/citation.cfm?doid=237814.237866 (visited on 06/22/2021).

[7] Shor, Peter W. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". en. In: *SIAM Journal on Computing* 26.5 (Oct. 1997), pp. 1484–1509. ISSN: 0097-5397, 1095-7111. DOI: 10.1137/S0097539795293172. URL: http://epubs.siam.org/doi/10.1137/S0097539795293172 (visited on 06/22/2021).

[8] Lanyon, B. P., Whitfield, J. D., Gillett, G. G., Goggin, M. E., Almeida, M. P., Kassal, I., Biamonte, J. D., Mohseni, M., Powell, B. J., Barbieri, M., Aspuru-Guzik, A., and White, A. G. "Towards quantum chemistry on a quantum computer". en. In: *Nature Chemistry* 2.2 (Feb. 2010), pp. 106–111. ISSN: 1755-4330, 1755-4349. DOI: 10.1038/nchem.483. URL: http://www.nature.com/articles/nchem.483 (visited on 06/22/2021).

[9] Egger, Daniel J., Gambella, Claudio, Marecek, Jakub, McFaddin, Scott, Mevissen, Martin, Raymond, Rudy, Simonetto, Andrea, Woerner, Stefan, and Yndurain, Elena. "Quantum Computing for Finance: State of the Art and Future Prospects". In: *IEEE Transactions on Quantum Engineering* 1 (2020). arXiv: 2006.14510, pp. 1–24. ISSN: 2689-1808. DOI: 10.1109/TQE.2020.3030314. URL: http://arxiv.org/abs/2006.14510 (visited on 06/22/2021).

[10] Yang, Jiaying, Awan, Ahsan Javed, and Vall-Llosera, Gemma. "Support vector machines on noisy intermediate scale quantum computers". In: *arXiv preprint arXiv:1909.11988* (2019).

[11] Schuld, Maria and Killoran, Nathan. "Quantum Machine Learning in Feature Hilbert Spaces". In: *Physical Review Letters* 122.4 (Feb. 2019). ISSN: 1079-7114. DOI: 10.1103/physrevlett.122.040504. URL: http://dx.doi.org/10.1103/PhysRevLett.122.040504.

[12] Havlíček, Vojtěch, Córcoles, Antonio D., Temme, Kristan, Harrow, Aram W., Kandala, Abhinav, Chow, Jerry M., and Gambetta, Jay M. "Supervised learning with quantum-enhanced feature spaces". In: *Nature* 567.7747 (Mar. 2019), pp. 209–212. ISSN: 1476-4687. DOI: 10.1038/s41586-019-0980-2. URL: http://dx.doi.org/10.1038/s41586-019-0980-2.

[13] Zoufal, Christa, Lucchi, Aurélien, and Woerner, Stefan. "Quantum Generative Adversarial Networks for learning and loading random distributions". In: *npj Quantum Information* 5.1 (Nov. 2019). ISSN: 2056-6387. DOI: 10.1038/s41534-019-0223-2. URL: http://dx.doi.org/10.1038/s41534-019-0223-2.

[14] Schuld, Maria, Bocharov, Alex, Svore, Krysta, and Wiebe, Nathan. "Circuit-centric quantum classifiers". en. In: *Physical Review A* 101.3 (Mar. 2020). arXiv: 1804.00633, p. 032308. ISSN: 2469-9926, 2469-9934. DOI: 10.1103/PhysRevA.101.032308. URL: http://arxiv.org/abs/1804.00633 (visited on 02/12/2021).

[15] Farhi, Edward and Neven, Hartmut. "Classification with Quantum Neural Networks on Near Term Processors". en. In: *arXiv:1802.06002 [quant-ph]* (Aug. 2018). arXiv: 1802.06002. URL: http://arxiv.org/abs/1802.06002 (visited on 02/12/2021).

[16] Cerezo, M., Arrasmith, Andrew, Babbush, Ryan, Benjamin, Simon C., Endo, Suguru, Fujii, Keisuke, McClean, Jarrod R., Mitarai, Kosuke, Yuan, Xiao, Cincio, Lukasz, and Coles, Patrick J. "Variational Quantum Algorithms". en. In: *arXiv:2012.09265 [quant-ph, stat]* (Dec. 2020). arXiv: 2012.09265. URL: http://arxiv.org/abs/2012.09265 (visited on 02/12/2021).

[17] Khan, Sumsam Ullah, Awan, Ahsan Javed, and Vall-Llosera, Gemma. "K-means clustering on noisy intermediate scale quantum computers". In: *arXiv preprint arXiv:1909.12183* (2019).

[18] Schuld, Maria, Sinayskiy, Ilya, and Petruccione, Francesco. "The quest for a Quantum Neural Network". en. In: *Quantum Information Processing* 13.11 (Nov. 2014), pp. 2567–2586. ISSN: 1570-0755, 1573-1332. DOI: 10.1007/s11128-014-0809-8. URL: http://link.springer.com/10.1007/s11128-014-0809-8 (visited on 02/12/2021).

[19] Tacchino, Francesco, Macchiavello, Chiara, Gerace, Dario, and Bajoni, Daniele. "An artificial neuron implemented on an actual quantum processor". In: *npj Quantum Information* 5.1 (Mar. 2019). ISSN: 2056-6387. DOI: 10.1038/s41534-019-0140-4. URL: http://dx.doi.org/10.1038/s41534-019-0140-4.

[20] Benedetti, Marcello, Lloyd, Erika, Sack, Stefan, and Fiorentini, Mattia. "Parameterized quantum circuits as machine learning models". In: *Quantum Science and Technology* 4.4 (Nov. 2019), p. 043001. ISSN: 2058-9565. DOI: 10.1088/2058-9565/ab4eb5. URL: http://dx.doi.org/10.1088/2058-9565/ab4eb5.

[21] Benedetti, Marcello, Lloyd, Erika, Sack, Stefan, and Fiorentini, Mattia. "Parameterized quantum circuits as machine learning models". In: *Quantum Science and Technology* 4.4 (Nov. 2019), p. 043001. ISSN: 2058-9565. DOI: 10.1088/2058-9565/ab4eb5. URL: http://dx.doi.org/10.1088/2058-9565/ab4eb5.

[22] Mangini, Stefano, Tacchino, Francesco, Gerace, Dario, Bajoni, Daniele, and Macchiavello, Chiara. "Quantum computing models for artificial neural networks". In: *EPL (Europhysics Letters)* 134.1 (Apr. 2021), p. 10002. ISSN: 1286-4854. DOI:

10.1209/0295-5075/134/10002. URL: http://dx.doi.org/10.1209/0295-5075/134/10002.

[23] Abbas, Amira, Sutter, David, Zoufal, Christa, Lucchi, Aurelien, Figalli, Alessio, and Woerner, Stefan. “The power of quantum neural networks”. In: *Nature Computational Science* 1.6 (June 2021), pp. 403–409. ISSN: 2662-8457. DOI: 10.1038/s43588-021-00084-1. URL: http://dx.doi.org/10.1038/s43588-021-00084-1.

[24] Bell, John S. “On the Einstein Podolsky Rosen Paradox”. en. In: *Physics* 1.3 (Apr. 1964), pp. 195–200. URL: https://cds.cern.ch/record/111654/files/vol1p195-200_001.pdf (visited on 04/05/2019).

[25] Hensen, B., Bernien, H., Dréau, A. E., Reiserer, A., Kalb, N., Blok, M. S., et al. “Loophole-free Bell inequality violation using electron spins separated by 1.3 kilometres”. In: *Nature* 526 (Oct. 2015), p. 682. URL: https://doi.org/10.1038/nature15759.

[26] Hensen, B., Kalb, N., Blok, M. S., Dréau, A., Reiserer, A., Vermeulen, R. F. L., et al. “Loophole-free Bell test using electron spins in diamond: second experiment and additional analysis”. In: *Scientific Reports* 6.1 (Sept. 2016). arXiv: 1603.05705. ISSN: 2045-2322. DOI: 10.1038/srep30289. URL: http://arxiv.org/abs/1603.05705 (visited on 04/05/2019).

[27] Shalm, Lynden K., Meyer-Scott, Evan, Christensen, Bradley G., Bierhorst, Peter, Wayne, Michael A., Stevens, Martin J., et al. “Strong Loophole-Free Test of Local Realism”. en. In: *Physical Review Letters* 115.25 (Dec. 2015). ISSN: 0031-9007, 1079-7114. DOI: 10.1103/PhysRevLett.115.250402. URL: https://link.aps.org/doi/10.1103/PhysRevLett.115.250402 (visited on 04/05/2019).

[28] Weihs, Gregor, Jennewein, Thomas, Simon, Christoph, Weinfurter, Harald, and Zeilinger, Anton. “Violation of Bell’s Inequality under Strict Einstein Locality Conditions”. In: *Phys. Rev. Lett.* 81 (23 Dec. 1998), pp. 5039–5043. DOI: 10.1103/PhysRevLett.81.5039. URL: https://link.aps.org/doi/10.1103/PhysRevLett.81.5039.

[29] Dikme, A., Reichel, N., Laghaout, A., and Björk, G. *Measuring the Mermin-Peres magic square using an online quantum computer*. 2020. arXiv: 2009.10751 [quant-ph].

[30] Nielsen, Michael A. and Chuang, Isaac L. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. 10th. USA: Cambridge University Press, 2011. ISBN: 1107002176.

[31] Clarke, John and Wilhelm, Frank K. "Superconducting quantum bits". en. In: *Nature* 453.7198 (June 2008), pp. 1031–1042. ISSN: 0028-0836, 1476-4687. DOI: 10.1038/nature07128. URL: http://www.nature.com/articles/nature07128 (visited on 05/05/2019).

[32] Aleksandrowicz, Gadi, Alexander, Thomas, and Panagiotis Barkoutsos, et. al. *Qiskit: An Open-source Framework for Quantum Computing*. Version 0.7.2. Jan. 2019. DOI: 10.5281/zenodo.2562111. URL: https://doi.org/10.5281/zenodo.2562111.

[33] Mehlig, B. *Machine learning with neural networks*. 2021. arXiv: 1901.05639 [cs.LG].

[34] LaRose, Ryan and Coyle, Brian. "Robust data encodings for quantum classifiers". In: *Physical Review A* 102.3 (Sept. 2020). arXiv: 2003.01695, p. 032420. ISSN: 2469-9926, 2469-9934. DOI: 10.1103/PhysRevA.102.032420. URL: http://arxiv.org/abs/2003.01695 (visited on 05/23/2021).

[35] Plesch, Martin and Brukner, Časlav. "Quantum-state preparation with universal gate decompositions". In: *Physical Review A* 83.3 (Mar. 2011). ISSN: 1094-1622. DOI: 10.1103/physreva.83.032302. URL: http://dx.doi.org/10.1103/PhysRevA.83.032302.

[36] Sim, Sukin, Johnson, Peter D., and Aspuru-Guzik, Alan. "Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms". In: *Advanced Quantum Technologies* 2.12 (Dec. 2019). arXiv: 1905.10876, p. 1900070. ISSN: 2511-9044, 2511-9044. DOI: 10.1002/qute.201900070. URL: http://arxiv.org/abs/1905.10876 (visited on 05/23/2021).

[37] Mitarai, K., Negoro, M., Kitagawa, M., and Fujii, K. "Quantum circuit learning". In: *Physical Review A* 98.3 (Sept. 2018). ISSN: 2469-9934. DOI: 10.1103/physreva.98.032309. URL: http://dx.doi.org/10.1103/PhysRevA.98.032309.

[38] Crooks, Gavin E. *Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition*. 2019. arXiv: 1905.13311 [quant-ph].

[39] Schuld, Maria, Bergholm, Ville, Gogolin, Christian, Izaac, Josh, and Killoran, Nathan. "Evaluating analytic gradients on quantum hardware". In: *Physical Review A* 99.3 (Mar. 2019). ISSN: 2469-9934. DOI: 10.1103/physreva.99.032331. URL: http://dx.doi.org/10.1103/PhysRevA.99.032331.

[40] Ruder, Sebastian. *An overview of gradient descent optimization algorithms*. 2017. arXiv: 1609.04747 [cs.LG].

[41] Kingma, Diederik P. and Ba, Jimmy. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[42] Sweke, Ryan, Wilde, Frederik, Meyer, Johannes, Schuld, Maria, Faehrmann, Paul K., Meynard-Piganeau, Barthélémy, and Eisert, Jens. "Stochastic gradient descent for hybrid quantum-classical optimization". en. In: *Quantum* 4 (Aug. 2020). arXiv: 1910.01155, p. 314. ISSN: 2521-327X. DOI: 10.22331/q-2020-08-31-314. URL: http://arxiv.org/abs/1910.01155 (visited on 05/31/2021).

[43] An, Sanghyeon, Lee, Minjun, Park, Sanglee, Yang, Heerin, and So, Jungmin. *An Ensemble of Simple Convolutional Neural Network Models for MNIST Digit Recognition*. 2020. arXiv: 2008.10400 [cs.CV].

[44] Holmes, Zoë, Sharma, Kunal, Cerezo, M., and Coles, Patrick J. *Connecting ansatz expressibility to gradient magnitudes and barren plateaus*. 2021. arXiv: 2101.02138 [quant-ph].