WROCŁAW UNIVERSITY OF SCIENCE AND TECHNOLOGY
Faculty of Electronics, Photonics and Microsystems

FIELD: Electronics (EKA)

BACHELOR'S THESIS

Quantum-Classical Hybrid Neural Networks and Their Applications

Kwantowo-klasyczne hybrydowe sieci neuronowe oraz ich zastosowania

AUTHOR:
Patryk Dolaciński

SUPERVISOR:
dr hab. inż. Przemysław Śliwiński

GRADE:

WROCŁAW 2022

To my parents for providing conditions for my growth and to my supervisor for expanding my horizons and interests

Contents

1 Introduction **5**
1.1 Abstract [English] . . . 5
1.2 Streszczenie [Polish] . . . 6

2 Machine Learning **9**
2.1 Artificial Neural Networks (ANN's) . . . 9
2.2 Supervised Learning . . . 10
2.3 Input data structures . . . 10
2.4 Data augmentation . . . 11
2.5 Backpropagation . . . 11
2.6 Hyperparameters and Model Optimization . . . 11
2.7 Deep learning . . . 11
2.8 Exploration of ML methods . . . 12
2.8.1 Convolutional Neural Networks (CNN's) . . . 12
2.8.2 Transfer Learning . . . 13
2.8.3 Generative Adversal networks (GAN's) . . . 15

3 Quantum Computing **17**
3.1 The case for quantum . . . 17
3.2 Qubit . . . 17
3.3 Visualizing quantum states . . . 19
3.4 Quantum Computing 101 . . . 20
3.5 1 qubit gates . . . 20
3.5.1 U-gate: . . . 20
3.5.2 X-gate: . . . 20
3.5.3 Y-gate: . . . 20
3.5.4 Z-gate: . . . 21
3.5.5 Hadamard gate: . . . 22
3.5.6 Phase shift gate: . . . 22
3.5.7 S-gate: . . . 23
3.5.8 T-gate: . . . 23
3.5.9 Dagger variations . . . 23
3.6 Quantum circuits . . . 24
3.6.1 Measurements . . . 24
3.7 2 qubit gates . . . 25
3.7.1 Controlled X | CX | CNOT . . . 25
3.7.2 Swap gate . . . 26
3.8 3 qubit gates . . . 26

3.8.1 Toffoli's gate . . . 26
3.8.2 Controlled SWAP gate . . . 27
3.9 Reaching Universality . . . 27
3.10 Current state of Quantum Devices . . . 28
3.10.1 Physical devices . . . 28
3.10.2 The road to a million quantum states . . . 28
3.10.3 Simulations . . . 29
3.11 Quantum Programming . . . 29

4 Implementation **31**
4.1 Datasets . . . 31
4.1.1 MNIST dataset . . . 31
4.1.2 Stanford Dogs Dataset . . . 31
4.1.3 NIST dataset . . . 33
4.2 Used technology . . . 34
4.3 Description of used hybrid algorithms . . . 34
4.3.1 Quantum-Classical Convolutional Neural Networks (QCCNN) . . . 34
4.3.2 Classical-Quantum (CQ) transfer learning . . . 35
4.3.3 Quantum Generative Adversal Networks . . . 36

5 Results **39**
5.1 QCCNN with MNIST . . . 39
5.1.1 The 10 layer question . . . 39
5.1.2 Results for other depth QCCNN's . . . 40
5.1.3 QCCNN's - conclusions . . . 42
5.2 CQ Transfer learning approach . . . 42
5.2.1 Raw results . . . 42
5.2.2 Results with augmented dataset . . . 42
5.2.3 Circuit depth influence on network performance . . . 45
5.2.4 Learning rate influence on networks performance . . . 47
5.2.5 QTL networks - conclusions . . . 49
5.3 QGAN's . . . 49
5.3.1 Generating letter D . . . 50
5.3.2 Generating letter J . . . 51
5.3.3 Generating letter I . . . 52
5.3.4 QGAN's - conclusions . . . 53

6 Conclusions **55**
6.1 The case for Hybrid Quantum Classical Neural Networks . . . 55
6.1.1 Higher accuracy . . . 55
6.1.2 Faster convergence to minima . . . 55
6.1.3 Quantum insights . . . 55
6.2 Shortfalls . . . 56
6.2.1 Instability and Noise . . . 56
6.2.2 Infrastructure . . . 56
6.3 Final words . . . 57

Bibliography **59**

*A*bbreviations list

- qubit - Quantum Bit
- QC - Quantum Computing
- ML - Machine Learning
- TL - Transfer Learning
- GAN - Generative Adversal Network
- QGAN - Quantum Generative Adversal Network
- QML - Quantum Machine Learning
- NN - Neural Network
- ANN - Artificial Neural Network
- HNN - Hybrid Neural Network
- CNN - Convolutional Neural Network
- QNN - Quantum Convolutional Neural Network
- QCCNN - Quantum Classical Convolutional Neural Network
- GPU - Graphics Processing Unit
- NIST - Institute of Standards and Technology dataset
- MNIST - Modified Institute of Standards and Technology dataset
- CQTL - Classical-To-Quantum Transfer Learning
- NISQ - Noisy Intermediate-scale Quantum era

Chapter 1

Introduction

1.1 Abstract [English]

Ever since the first description of an algorithm that would be faster on a quantum computer than on a classical one by *David Deutsch, 1985* [18] many times research has proven that quantum computers can aid us in surpassing the barrier set by fastest and most efficient classical algorithms in many tasks which may prove crucial in much of modern-day computer science problems, especially within the fields of optimization [44], cybersecurity [10] and communications [12][38]. Grover [24] showed us an example of a search algorithm that can be faster than a classical one while Shor [42] allowed us to factor integers within polynomial time-frame - something that wasn't possible on classical computers. These are only few examples of proven superiority of quantum algorithms over the classical ones.

Recent developments in the field of Quantum Computing have led to several developments in the forms of Quantum Machine Learning (QML) algorithms which have shown to have promising results [11], especially in the case when the data is inherently quantum [9]. This only proves that the development of quantum computation at it's core if not for the quantum advantage over the classical algorithms also proves as a useful research tool for understanding quantum mechanics itself.

While the prospect of having better algorithms for the quantum world is lucrative, a search began for a quantum advantage within the field of machine learning - a quantum ML algorithm which would be more effective at a certain task than a classical ML algorithm - in machine learning terms this could mean the trained model achieving better accuracy, less overfitting to the training data or simply being faster to train. Most recent developments have led to experimentation with hybrid neural networks (HNN) utilizing both classical and quantum machine learning methods within their architecture which shows even more promising results, even when the data is not inherently quantum. Examples include hybrid convolutional neural networks (QCCNN) [33][26]. Both with convolutional layers swapped for quantum convolutional layers and with classical convolutional layers connected to deep quantum layers. Other examples of HNN include regression models [37], hybrid transfer learning [8], quantum implementations of Hopfield networks [40] as well as Quantum Generative Adversal Networks (QGAN's) [16][29] which are only few of quantum machine learning algorithms proposed by research.

The aim of this paper is exploration of few chosen hybrid neural networks architectures in depth in order to highlight their potential strengths and shortcomings as well as to show

that they might have possible uses in real life problems even now, in the early stages of Quantum Computing that we are currently in.

Chapter 2 contains review of used technology and literature exploring basics of ML. Chapter 3 reviews quantum information theory that is essential to understanding implemented algorithms. Chapter 4 contains review of datasets used during the experimentation as well as in-depth description of implemented hybrid ML algorithms and how they work. Chapter 5 contains all the results produced during experimentation and their analysis. Finally, in chapter 6 I draw conclusions on the results of experimentation and future of QML field in general.

1.2 Streszczenie [Polish]

Od roku 1985, gdy David Deutch zaproponował algorytm, który byłby szybszy na komputerze kwantowym niż na klasycznym[18], pojawiło się wiele innych algorytmów dowodzących, że komputery kwantowe mogą nam pomóc w przekroczeniu barier wyznaczonych przez złożoność obliczeniową najlepszych klasycznych algorytmów. Opracowane algorytmy kwantowe mogą okazać się kluczowe w rozwiązaywaniu współczesnych problemów informatyki, szczególnie w dziedzinach optymalizacji [44], cyberbezpieczeństwie [10] i komunikacji [12][38]. Grover [24] pokazał nam przykład algorytmu wyszukiwania, który może być szybszy niż klasyczny, podczas gdy Shor [42] pozwolił nam na rozkład liczb całkowitych na czynniki pierwsze w czasie wielomianowym - coś, co nie było możliwe na klasycznych komputerach. Istnieje wiele wiecej przykładów udowodnionej teoretycznie przewagi algorytmów kwantowych nad klasycznymi w wielu innych problemach.

Ostatnie osiągnięcia w dziedzinie Kwantowej Informatyki doprowadziły do rozwinięcia algorytmów kwantowego uczenia maszynowego (QML), które okazały się mieć obiecujące wyniki [11], szczególnie w przypadku, gdy dane wejściowe są z same w sobie kwantowe [9]. Przykłady te tylko dowodzą, że rozwój informatyki kwantowej i komputerów kwantowych w swej istocie, jeśli nie dla kwantowej przewagi nad klasycznymi algorytmami, okazuje się również przydatnym narzędziem badawczym dla zrozumienia samej mechaniki kwantowej.

Podczas gdy perspektywa posiadania lepszych algorytmów dla świata kwantowego jest atrakcyjna, rozpoczęto poszukiwania kwantowej przewagi w dziedzinie uczenia maszynowego - kwantowego algorytmu uczenia maszynowego, który byłby bardziej skuteczny w pewnych aspektach niż klasyczny algorytm ML - w kategoriach uczenia maszynowego może to oznaczać, że wyszkolony model osiąga lepszą dokładność, wykazuje mniejsze dopasowanie do danych szkoleniowych lub po prostu jest szybszy w szkoleniu. Najnowsze osiągnięcia doprowadziły do eksperymentów z hybrydowymi sieciami neuronowymi (HNN) wykorzystującymi w swojej architekturze zarówno klasyczne, jak i kwantowe metody uczenia maszynowego, co daje jeszcze lepsze rezultaty nawet w przypadkach, gdy dane wejściowe nie są kwantowe. Przykładem mogą być hybrydowe konwencjonalne sieci neuronowe (QCCNN) [33] zarówno z warstwami konwolucyjnymi zamienionymi na kwantowe warstwy konwolucyjne, jak i z klasycznymi warstwami konwolucyjnymi połączonymi z głębokimi warstwami kwantowymi. Innymi przykładami HNN są modele regresji [37], hybrydowe uczenie transferowe [8], kwantowe implementacje sieci Hopfielda [40], jak również kwantowy odpowiednik Generative Adversal Networks (QGAN's) - sieci do generowania obrazów [16][29]. Są to tylko jedne z wielu algorytmów QML proponowane przez badaczy w ostatnich latach.

Celem niniejszej pracy jest dogłębna eksploracja kilku wybranych architektur hybrydowych sieci neuronowych, aby ukazać ich potencjalne mocne i słabe strony, a także zademonstrować, że mogą one mieć możliwe zastosowanie w rzeczywistych problemach nawet teraz, we wczesnym stadium rozwoju Komputerów Kwantowych.

Rozdział 2 zawiera przegląd wykorzystywanych technologii oraz literatury zgłębiającej podstawy Uczenia Maszynowego. Rozdział 3 zawiera przegląd teorii i literatury związanej z informatyką kwantową, która jest niezbędna do zrozumienia zaimplementowanych algorytmów. Rozdział 4 zawiera przegląd zbiorów danych użytych podczas eksperymentów, jak również dogłębny opis zaimplementowanych hybrydowych algorytmów ML i sposobu ich działania. Rozdział 5 zawiera wszystkie wyniki uzyskane podczas eksperymentów i ich analizę. Na koniec w rozdziale 6 wyciągane są wnioski dotyczące bardziej ogólnych wyników eksperymentów i przyszłości dziedziny Informatyki Kwantowej i Kwantowego Uczenia Maszynowego.

Chapter 2

Machine Learning

Machine learning as a field is concerned with constructing computer programs that improve and adapt using the data they are provided. It seeks to construct programs that can distinguish between certain given questions in entirely new situations based on the experience from already presented data.[34]

This approach may be beneficial when the problem at hand is too complicated for normal programming - it allows to deploy such application to learn from experience and based on that deploy its newfound knowledge to the problem at hand. It is also beneficial in exposing certain connections and correlations that appear between provided data, which if not serving completely as means of processing the data can serve as a simple preinvestigation method saving time and resources. Sometimes the structure and size of presented data may simply be too much for a human to find correlations in and using such techniques may be a necessity.

2.1 Artificial Neural Networks (ANN's)

Artificial neural networks (ANN's) are computational processes designed in such a way that resembles real life biological systems such as human brain. The network consists of few layers of computational units called neurons. The first layer called input layer reads input data and forwards them into next layer on the basis of weight assigned to each neuron. The last layer is the output layer while the layers in-between are often referred to as hidden layers - as they are not interacting with the outside world. An example architecture of an ANN is shown in figure 2.1.

Each consecutive layer trains by the means of forming connection between neurons - in purely mathematical terms this means adjusting weight between each neuron so that neurons that have a lot of common in them fire together. The last layers contains as many neurons as there are answers in the question - if it is a simple yes/no answer then the output layer will only have 1 neuron, if the question is whether there is a plane, dog or a car on the image then the output layer will consist of 3 neurons and so on. these networks usually learn by the means of supervised learning described in next section (2.2).

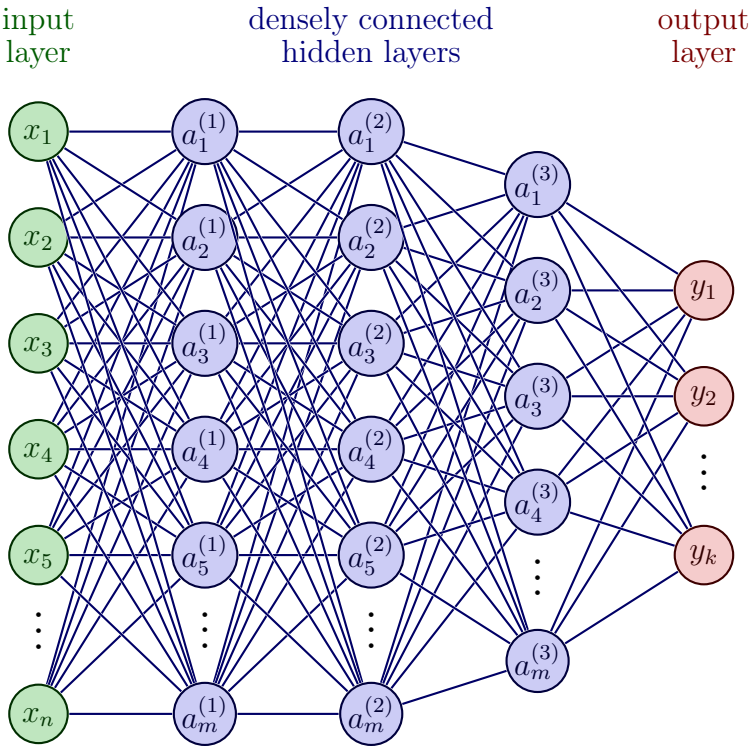


Figure 2.1 Example architecture of an ANN

2.2 Supervised Learning

One of the forms which is within the scopes of this paper is supervised learning - it is one of the methods in which a neural network may learn. It is often described as learning with a teacher - someone who oversees the learning process and corrects the mistakes. In machine learning terms it means that the training process consists of iterating over labeled data, preforming recognition task and then checking whether the task was performed correctly or not. Based on that feedback the weight of the neurons within the NN are adjusted accordingly via a chosen method. This process will usually go on until a desirable accuracy is achieved at the task performed by the NN.

2.3 Input data structures

It is important to mention how the data that the NN will train on is structured. Since dataset can range from house prices to pictures of cancerous tissue the data can have many different structures and formats, nonetheless some things stay in commons.

During the training of the NN the data will usually be split into training and validation data or training, testing and validation data. The training data is shown to the network in order to train it - the network tries to classify the given object correctly and then it is given an answer if it performed good or bad.

The validation set of data is used to check whether the network performs correctly even on data it has never seen during the training - since it has only seen the training set. This allows to check whether the network can generalize well, and to ensure that it won't be only good at guessing the data it has only seen during training.

If one decides to use test subset of database it will be used at the end of the training to determine the final quality of the trained model.

The split ratio of the subsets can differ quite a lot between different datasets but usually we will want to use as much of the data for training the model and leave enough for validation/testing subsets. Example ratios of *50% training 25% testing 25% validation* and *75% training 25% validation* can both be valid depending on the nature of used dataset.

2.4 Data augmentation

Data augmentation is a method often used to extend the dataset - the technique consists of randomly transforming the training images within the dataset by cropping, resizing, blurring etc.

It allows to effectively make the data that is presented to the network more diverse and at times harder which often can result in achieving higher accuracy on validation data and thus can result in higher accuracy when the model is deployed in real life implementation. It also allows to avoid *overfitting* - the NN model learning the training data so well that it cannot generalize on real/validation data. [43]

2.5 Backpropagation

One of the more renowned algorithms of adjusting the weight of neural network is the backpropagation method introduced by *Rumelhart et alia in 1986*[41]

The algorithm calculates the gradient of the cost functions in goal of minimizing the cost function by computing the gradient one layer at a time beginning from the last one.

Since its introduction this algorithm has been used even more widely in ML applications and is perhaps the most commonly used method of adjusting the weight as of writing this paper. Because of that, many variations of backpropagation algorithm have been developed[34].

2.6 Hyperparameters and Model Optimization

Once a ML model achieves acceptable initial results it will usually need to be optimized if a more favourable results are expected. This is done by the use of so called hyperparameters - parameters that control the way the mode learns, such as learning rate.

An optimized model can achieve way better results than an not optimised one and same will be true for the hybrid models explored within the scope of this work.

2.7 Deep learning

Deep structured learning or - as is is more widely known - deep learning, is a method within ML that wishes to harness power of many layered NN to extract layered information from the input data in a goal of better modeling complex relationships that occur among the presented data. [17]

Artificial neural networks that use many hidden layers to progressively extract features from given data are often called deep neural networks. These types of networks are far more widely popular nowadays since many problems that require the use of ML often are complex and simpler architectures simply can't achieve desirable accuracy at such tasks.

All of the three explored NN architectures within this paper are in fact deep neural networks, so it is important that this definition was brought up.

2.8 Exploration of ML methods

This section goes over classical methods and algorithms that are foundations for the explored hybrid quantum classical neural networks within this paper.

2.8.1 Convolutional Neural Networks (CNN's)

While proving extremely effective, normal artificial neural networks fall short when the data increases in dimensionality - a good and simple example are 2 dimensional images - a 64 by 64 RGB image will have 12288 different values which would mean that our NN requires that many neurons in the input layer - otherwise we would start losing raw input data which may impact the accuracy of the model.

History

As is the case with most most of machine learning methods the CNN's were inspired by nature. The first CNN architecture was introduced by *Fukushima, 1982* [22] which was inspired by the work of *Hubel and Wiesel, 1962* [30] which identified two basic visual cell types in the brain. Some more research was made into CNN's over the following years, however they have failed to gain the traction up until 2004, when *Kyoung-Su Oh and Keechul Jung* in their paper *GPU implementation of neural networks* [36] showed a 20 times speedup with CNN's when implementing them on GPU's. This breakthrough has put NN back into the picture of modern science and allowed for development of more sophisticated architectures, which has paved the way to very deep neural network architectures and their power in recognition tasks that we often see manifesting today in many modern day implementations.

Inner workings of CNN's

Convolutional neural networks (CNN's) are a specific subclass of ANN's - they still work based on the same principles as normal artificial neural networks do. The primary difference is that CNN's contain convolution layers and pooling layers which are specifically designed to improve efficiency with 2-dimensional data. The convolution layer applies a kernel over all the areas of the image and outputs activation maps that represent features found by the convolution layer in that specific part of an image. Pooling layer simply acts as a down-sampling layer taking max or mean values from the area, which allows the NN to improve efficiency by reducing dimensinality while still retaining reasonable accuracy. At the end of the CNN we find densely connected layers - something which works analogous to the normal ANN. This combination allows neural network to efficiently work with 2-dimensional data.

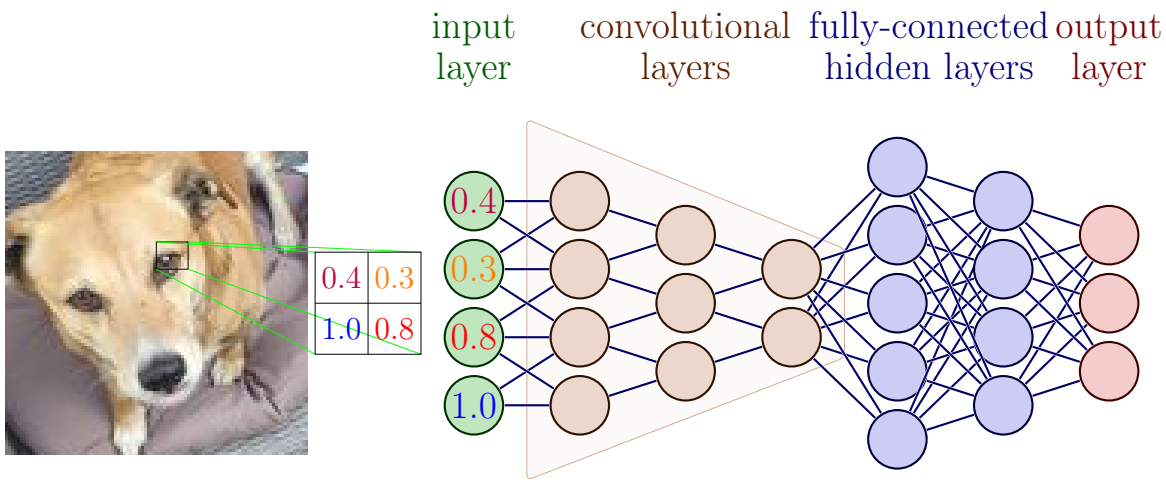


Figure 2.2 Visualization of working convolution matrix with a simple CNN architecture

2.8.2 Transfer Learning

Transfer learning is a machine learning method, in which a network that is already trained is used as part of the new neural network architecture in the goal of transferring part of the knowledge learned by the trained network onto the new network and dataset it faces. This approach - as most of the machine learning field - is also inspired by the human mind. The theoretical basics for transfer learning were first introduced by *Bozinovski and Fulgosi in 1976*[13]. More recently transfer learning has been applied with combination of deep neural networks, an example being a work by *Tan et alia*[45]. These types of network usually implement quite complex architectures, which means that applying them with learned models to new problems via the method of transfer learning can bring a huge boost in accuracy of the model, time it takes to train the model as well as the general robustness of the model with less overfitting of the training data and better recognition between classes that are very similar to each other. [45][46]

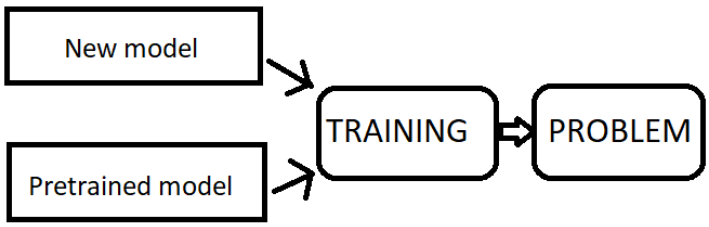


Figure 2.3 Visualization of the transfer learning method

Over the years much work was put into creating and training different transfer learning networks with many examples being openly available to public use such as deep residual networks (ResNet for short) introduced by *He et alia*[25], AlexNet for image classification introduced by *Krizhevsky et alia*[32] and densely connected convolutional networks (often known as DenseNet) introduced by *Huang et alia*. Their open source implementations with different development frameworks means that they can be effectively utilized by anyone with access to the internet erasing the requirement for heavy computational power within many implementation since the trainable part of the network needs to be less complex

thus making real life implementations of machine learning and AI in general less costly and more accessable to everyone.

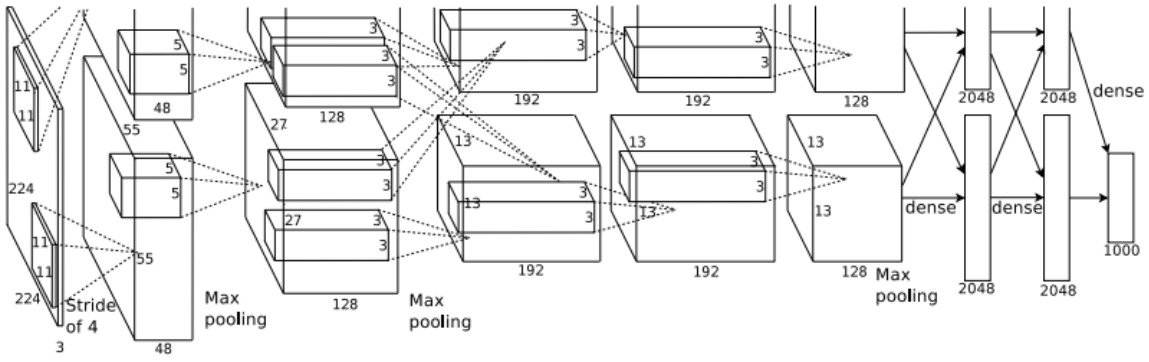


Figure 2.4 Sketch from original alexnet paper [32] showing the complex architecture of this networks architecture

Data

Important part of training the networks for transfer learning purposes is using a complex enough dataset, since the end goal is robustness. With image recognition tasks the networks are usually trained on ImageNet [2] dataset which contains 1281167 Training images and 50000 validation images of 1000 different classes ranging from dogs to cars as seen in figure 2.5.

(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 2.5 Example of 8 images taken from imagenet [2] dataset

The power of transfer learning

Transfer learning has proven to be an effective method in many machine learning tasks especially when the number of data is limited or the classes themselves are very complex. One of the key tasks is helping in diagnosing deceases such as Alzheimer's decease - *Nicholas et alia* [35], COVID-19 - *Ezzet et alia*[21] and breast cancer - *khan et alia*[31].

There are also simpler, more mundane tasks that can be accelerated with the help of transfer learning and pretrained models which are incredibly powerful by themselves - almost anything we can think of can be implemented into an end user application ranging from image classification to natural language processing (NLP)[48] making this method incredibly versatile and useful in many use cases and real life applications of ML and AI.

2.8.3 Generative Adversal networks (GAN's)

Generative Adversal Networks is a type of NN used for generation of artificial data. They achieve it via use of 2 NNs - one which is called the **generator** and the other the **discriminator**. The generator NN tries to generate the data for example an image in such a way that it cannot be distinguished from the used dataset. The task of the discriminator NN is to decide whether the image presented by the generator is a forgery or a real sample from the dataset.[15].

Both NN are trained synchronously - the generator generates images and receives feedback from the discriminator which receives both real images and forgeries alike - this way it trains to distinguish them better while the generator constantly improves based on feedback from the discriminator as seen in figure 2.6.

The generator has no access to the real data - it's only way of interacting with the dataset is through the discriminator which provides feedback whether the generated image passes as a real one from the dataset or not.

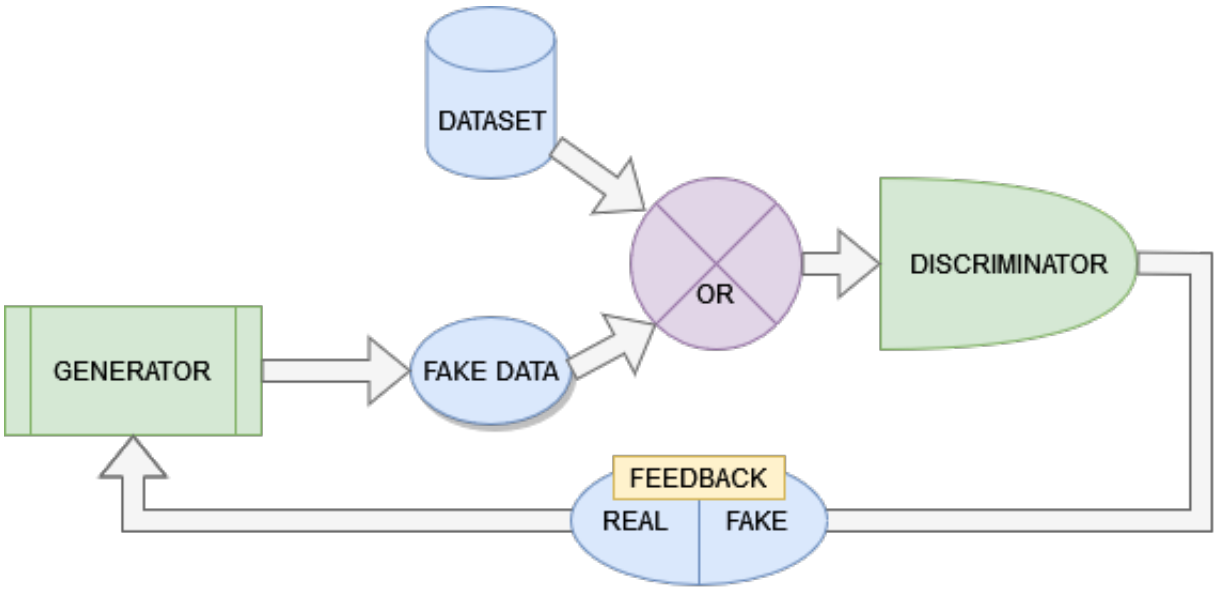


Figure 2.6 Chart visualizing the whole architecture of a GAN

Chapter 3

Quantum Computing

3.1 The case for quantum

Before delving into theory behind quantum information theory and quantum computation this section delves into reasons for why quantum computation should be explored and developed in the first place.

The first ever quantum algorithm demonstrating quantum advantage over a classical problem was shown by David Deutch in 1985 [18] who later with His colleague *Richard Jozsa* generalized the problem in 1992 [19]. The Deutch-Jozsa algorithm became known as the first ever quantum algorithm to outperform any possible deterministic classical algorithm at the task at hand - determining if a given Boolean function is balanced or not [19].

While quantum advantage was found the problem proposed by *David Deutch* was not particularly useful in most real life implementations and thus quantum computing was unable to gain much traction from this discovery. Over the years, however there have been many more publications showing quantum advantage in more common and widely used tasks - possibly the best example could be the Grover search algorithm [24] which allows to find the input within a n-sized list in just $\mathcal{O}(\sqrt{n})$ time giving us a quadratic speedup over best classical search algorithm - algorithms that are exponentially more widely used in many modern day applications.

There exist many more quantum algorithms that have proven quantum advantage over the classical ones in certain tasks such as Shor's algorithm [42], however they won't be described here since they are out of the scope of this paper.

There are even proven examples of QML algorithms outperforming classical ML algorithms in machine learning tasks when data is inherently quantum such as quantum state entanglement classifications shown by *Basheer er alia* [9] which notably does not require classical encoding to of the data and could theoretically be done on a entirely quantum system.

3.2 Qubit

In purely mathematical terms a quantum bit (or qubit for short) is an two dimensional complex vector space $\mathbb{C}^2$ representing a quantum state which in practice means that we are talking about a 4-dimensional real space $\mathbb{R}^4$, since we need 2 real dimensions per

each complex dimension. To write down qubits we can use matrices but usually we use a simpler braket notation introduced by Paul Dirac [20], which is a standard for writing down states in quantum physics and allows us to simply represent the many possible states a single qubit can have. It consist of an bra $\langle 0|$ and a ket $|0\rangle$ as the name braket would suggest. So a state of the qubit can be written as:

$$|\psi\rangle = \alpha \begin{pmatrix} 0 \\ 1 \end{pmatrix} + \beta \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \alpha\,|1\rangle + \beta\,|0\rangle \tag{3.1}$$

where α and β are complex numbers representing the amplitudes of the real and imaginary part of our quantum state. The two simplest quantum states a qubit can have are $|0\rangle$ which corresponds to 0 in classical terms and $|1\rangle$ corresponding to classical state 1. These two state are called computational basis state and are the orthonormal basis of our quantum state.

A single bit can hold either 1 or 0 as an information while single qubit technically can have an infinite amount of states - the only limitations is our ability to distinguish between them - this other states are possible due to superposition between computational basis states of $|1\rangle$ and $|0\rangle$. for example we could have the most boring kind of superposition such as:

$$|\psi\rangle = \frac{1}{\sqrt{2}}\,|1\rangle + \frac{1}{\sqrt{2}}\,|0\rangle \tag{3.2}$$

This in practice means that we would measure this qubit there is a 50% chance of measuring state of 1 and 50% chance of measuring 0. We can easily calculate the probabilities by squaring the amplitudes of the specific qubit, in our case that would be $(\frac{1}{\sqrt{2}})^2 = \frac{1}{2}$ in both cases. We call this principle **Born's rule** [27]

This very fact when combined with errors introduced due to physical nature and current capabilities of quantum hardware means that usually we need to run specific task n-times and then, from the distribution of all the measurement results we can approximate the state of the qubit at that moment. In the earlier example if we would run measurement of our qubit 1000 times we would approximately get 500 measurements of state 1 and 500 measurements of state $|0\rangle$.

It is also important to mention that by the power of quantum mechanics measuring a state of the qubit sets it to a definite state making our qubit either $|1\rangle$ or $|0\rangle$ - whichever we measured on our qubit - this feature of QC makes it both advantageous and disadvantageous in many aspects. In the earlier example we would always need to set our qubit back to the state to a superposition $\frac{1}{\sqrt{2}}\,|1\rangle + \frac{1}{\sqrt{2}}\,|0\rangle$ by transformations before we could measure it another time in state of superposition.

Since squaring the amplitudes gives us the probabilities of specific state being measured it means that this squared sum must be equal to 1 since we must measure either 1 or 0 - there exists no other possibilities. We can use that fact to rewrite our equation thanks to pythagorean trigonometric identity:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle = e^{i\gamma}(cos\frac{\theta}{2} + e^{i\gamma}sin\frac{\theta}{2}) \tag{3.3}$$

The first part of the equation 3.3 $e^{i\gamma}$ corresponds to a global phase within our qubit - something that is not measurable which means we can ignore it for simplicity's sake, if we do so we end up with an even simpler equation:

$$|\psi\rangle = cos\frac{\theta}{2} + e^{i\gamma}sin\frac{\theta}{2} \tag{3.4}$$

3.3 Visualizing quantum states

If we use the resulting equation 3.4 we can neatly plot it on a 3-dimensional sphere called a Bloch sphere which is often used for visualizing single qubit states.

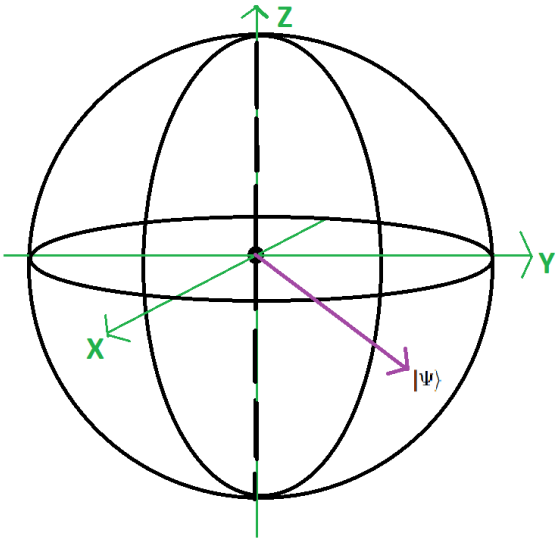


Figure 3.1 sketch of a Bloch sphere representing $|\psi\rangle$ state

Figure 3.2 illustrates representations of earlier mentioned quantum states graphed with help of IBM's qiskit [1] library for python.

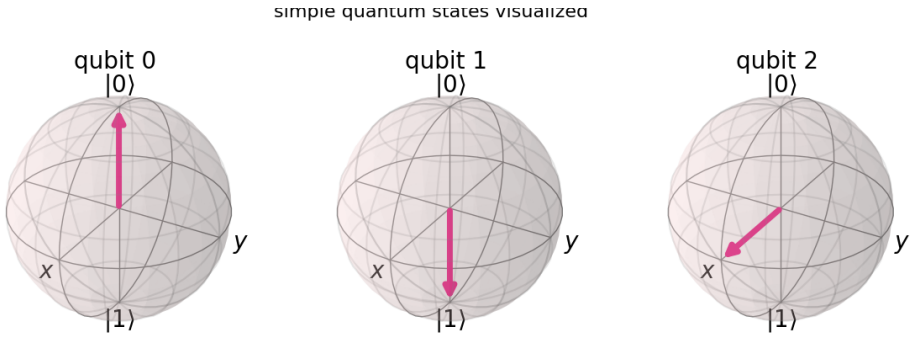


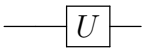
Figure 3.2 Example showing state $|0\rangle$, $|1\rangle$ and $(\frac{1}{\sqrt{2}}|1\rangle + \frac{1}{\sqrt{2}}|0\rangle)$ on Bloch spheres

3.4 Quantum Computing 101

The core calculations on quantum computers exist by the basis of linear transformations on the specific qubits, we use matrices to represent transformation on qubits. While in classical computers we are limited to only one operation - simply flipping the bit from 0 to 1 and vice versa via Boolean NOT gate in quantum computers we use certain 1-qubit quantum gates which transforms the state of our desired qubit in more sophisticated ways.

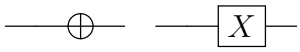
3.5 1 qubit gates

3.5.1 U-gate:



All of the single qubit gates as stated earlier are certain unitary transformation on our quantum states represented by matrices. The most universal way to represent any such transformations is to apply a unitary gate - or a U-gate for short. This gate is only theoretical as it is simply impossible to create without first using some other more basic gates transformation gate.

3.5.2 X-gate:



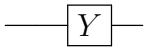
The X-gate allows us to flip the state of the qubit by the basis of axis X on the Bloch sphere, it can be described as matrix:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{3.5}$$

when our qubit is not in superposition applying the X gate flips the state to the other one - similarly to a NOT gate in classical computers:

$$X\left|1\right\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1+0 \\ 0+0 \end{pmatrix} = \left|0\right\rangle \tag{3.6}$$

3.5.3 Y-gate:



The Y gate flips the state by the Y axis on the Bloch sphere, in non superposition states it acts the same as X-gate. We describe it as:

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \tag{3.7}$$

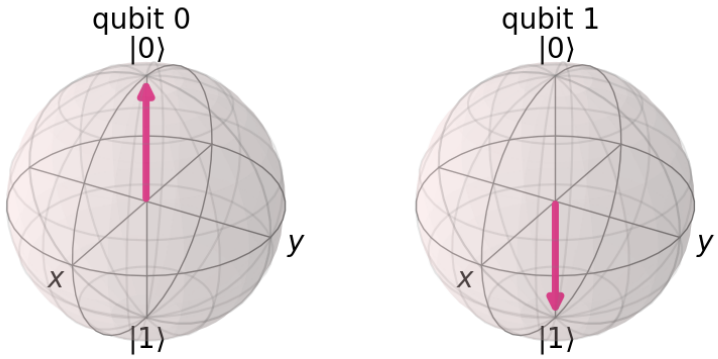


Figure 3.3 Qubit before (*qubit 0*) and after (*qubit 1*) applying the X gate.

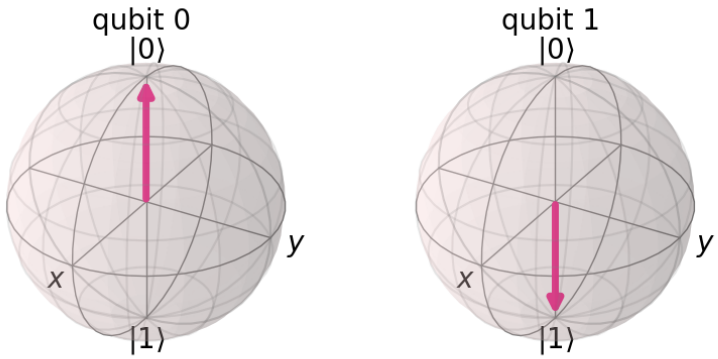
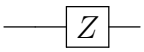


Figure 3.4 Qubit before (*qubit 0*) and after (*qubit 1*) applying the Y gate.

3.5.4 Z-gate:



Finally the Z-gate flips the state of a qubit by the basis of z axis on our Bloch sphere, it can be written as a matrix:

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{3.8}$$

Notably if our qubit is in state $|1\rangle$ or $|0\rangle$ it will have no effect on the state of this qubit.

$$Z\,|0\rangle = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 1+0 \\ 0+0 \end{pmatrix} = |0\rangle \tag{3.9}$$

These first three gates are often called Pauli's gates named after Pauli Wolfgang. They along with Hadamard gate defined below create the core of single qubit operations and quantum computing itself.

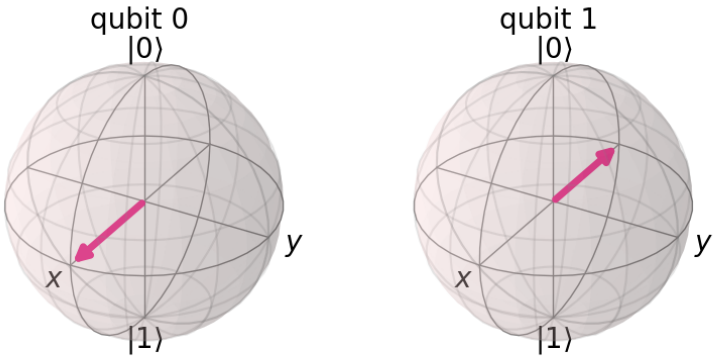
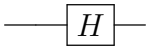


Figure 3.5 Qubit before (*qubit 0*) and after (*qubit 1*) applying the Z gate.

3.5.5 Hadamard gate:



The Hadamard gate is perhaps one of the most essential components of quantum computation as a whole. This is the gate that allows us to put our qubit into a superposition state of $\frac{1}{\sqrt{2}}|1\rangle + \frac{1}{\sqrt{2}}|0\rangle$ from state $|0\rangle$. It is described by the following matrix:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{3.10}$$

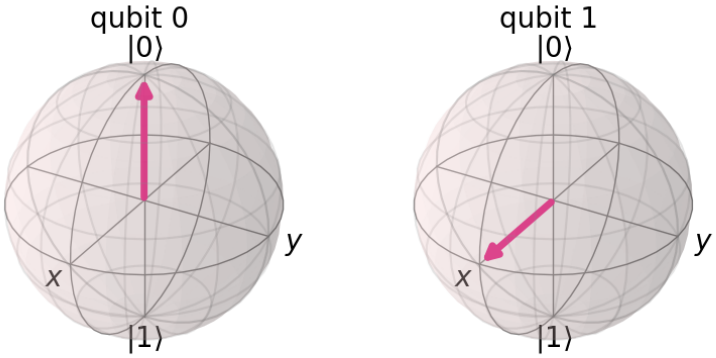
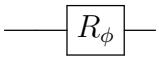


Figure 3.6 Qubit before (*qubit 0*) and after (*qubit 1*) applying the H gate.

3.5.6 Phase shift gate:



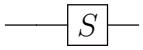
A phase gate changes the phase of a qubit - after applying a P-gate the probability of measuring state $|1\rangle$ and $|0\rangle$ stays the same while the phase changes. In the Bloch sphere

visualization it will be seen as a rotation around the Z-axis. The gate can be defines as:

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

As we can see the Pauli's Z-gate is a special case of phase shift gate - one which rotates the phase by π. There are also other commonly used special cases of the P-gate described below.

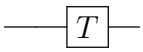
3.5.7 S-gate:



An S gate is a special case of a P-gate, which rotates the phase by $\frac{\pi}{2}$.

$$S = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{2}} \end{bmatrix} \tag{3.12}$$

3.5.8 T-gate:



Lastly, we can often meet the T gate, which similarly to the S gate is a special case of a P-gate that rotates the phase by $\frac{\pi}{4}$.

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{\frac{i\pi}{4}} \end{bmatrix} \tag{3.13}$$

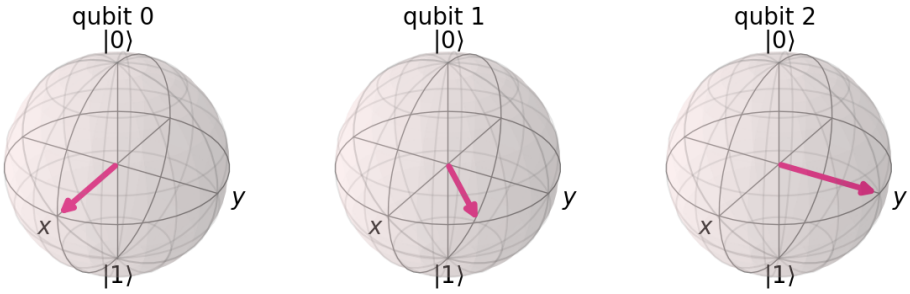
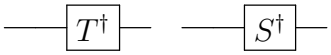


Figure 3.7 Qubit before (*qubit 0*) and after applying the T gate (*qubit 1*) and S gate (*qubit 2*).

3.5.9 Dagger variations



It is also important to mention dagger variations of phase shift gates, which simply perform phase rotation in the other direction.

$$S^\dagger = \begin{bmatrix} 1 & 0 \\ 0 & -e^{\frac{i\pi}{2}} \end{bmatrix} \quad T^\dagger = \begin{bmatrix} 1 & 0 \\ 0 & -e^{\frac{i\pi}{4}} \end{bmatrix} \tag{3.14}$$

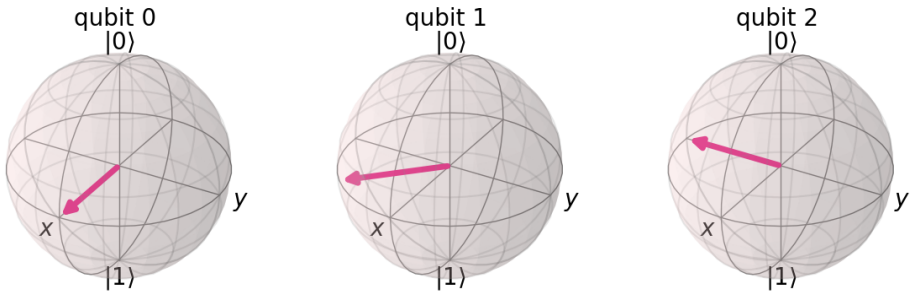


Figure 3.8 Qubit before (*qubit 0*) and after applying the dagger T gate (*qubit 1*) and dagger S gate (*qubit 2*).

3.6 Quantum circuits

Before introducing multi qubit gates it is important to define how a quantum algorithm is written. We call them quantum circuits and they are a diagram describing transformation or measurement of each qubit. Below are two examples of quantum circuits at work drown with the help of IBM's qiskit [1] library for python:

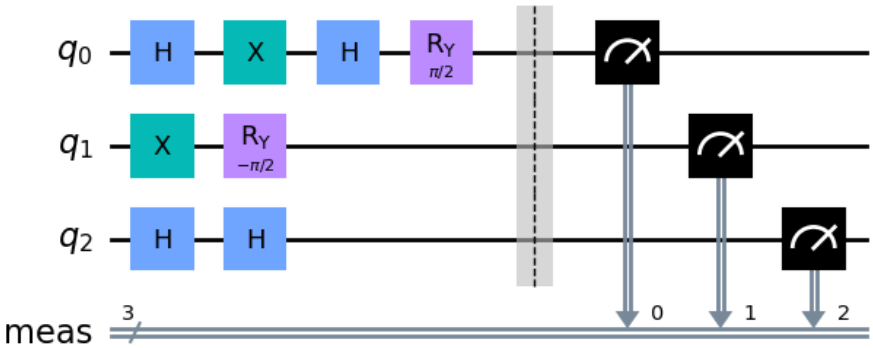
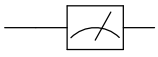


Figure 3.9 Example of a simple 3 qubit circuit using only 1bit gates (this circuit isn't particularly useful)

3.6.1 Measurements



The symbols at the end of the circuit 3.9 tells us that this specific qubit is being measured - in the example on figure 3.9 all qubits are measured.

Measurements can also be performed in different basis - for example we could apply Hadamard gates before measuring the state and thus 'shift' the state of a qubit - measuring $|1\rangle$ with probability of 1 when the state of the qubit was $\frac{1}{\sqrt{2}}|1\rangle + \frac{1}{\sqrt{2}}|0\rangle$ before applying

the Hadamard gate. This can be done in any basis we want, however the most common other basis is the mentioned Hadamard basis, where state $|1\rangle$ is also often denoted as state $|+\rangle$ and state $|0\rangle$ as $|-\rangle$.

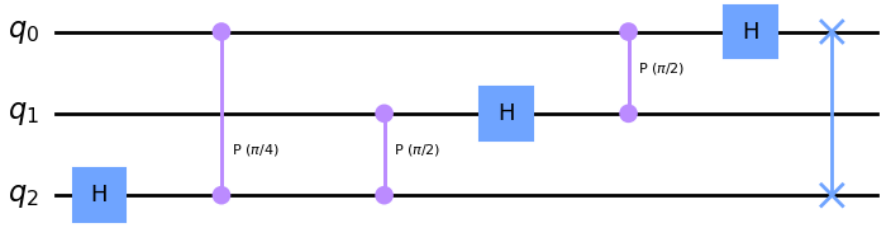


Figure 3.10 more sophisticated 3 qubit circuit, this circuit performs a 3 qubit quantum Fourier transform [47]

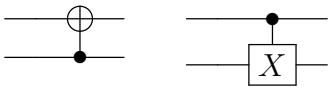
3.7 2 qubit gates

When we begin describing multi qubit circuits we will use the following notation:

$$|01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{3.15}$$

Equation 3.15 represents a 2 qubit circuit with first qubit in state $|0\rangle$ and the second in state $|1\rangle$

3.7.1 Controlled X | CX | CNOT



The controlled X (Controlled NOT) allows for 2 qubits to interact with each other, specifically the gate affects the state of the so called target qubit depending on the state of control qubit. The gate can be written as:

$$CX = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.16}$$

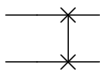
The target qubit will not be affected if the control qubit is in state $|0\rangle$ but if it is in state $|1\rangle$ then the control qubit will change its state to $|1\rangle$ if it was in state $|0\rangle$ or to $|0\rangle$ if it was in state $|1\rangle$. The gate often is compared as an analogue of classical XOR (exclusive OR) gate as seen in table 3.1.

there also exist controlled variants of Y an Z Pauli's gates, but since they work on the same principle as CNOT they will not be described here.

CNOT			XOR		
x	y	y'	x	y	x+y
$\|0\rangle$	$\|0\rangle$	$\|0\rangle$	0	0	0
$\|0\rangle$	$\|1\rangle$	$\|1\rangle$	0	1	1
$\|1\rangle$	$\|0\rangle$	$\|1\rangle$	1	0	1
$\|1\rangle$	$\|1\rangle$	$\|0\rangle$	1	1	1

Table 3.1 Table showing the difference in basis states with classical binary states represented by truth tables. y' shows a state of the y qubit after applying the gate. the state of the control qubit x does not change at all hence it is not described here

3.7.2 Swap gate



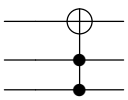
The swap gate is also essential part of quantum computing, as the name would suggest it allows us to swap states of 2 qubits. It is represented by the following matrix:

$$SWAP = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.17}$$

If we have a circuit with qubits of states $|01\rangle$ applying a swap gate on them will result in these qubit having states $|10\rangle$.

3.8 3 qubit gates

3.8.1 Toffoli's gate



Lastly there also exist 3 qubit gates, one of them being Toffoli gate which would be a controlled-controlled-NOT, it is described by the following matrix:

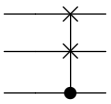
$$CCX = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.18}$$

This gate flips the target qubit whenever two control qubits are in state $|1\rangle$. Table 3.2 shows toffoli's gate truth table - on basis states the gate only flips target qubit when both control qubits are in state $|1\rangle$ - we can think of this gate as an analogue to classical AND gate.

control qubit 1	control qubit 2	target qubit	target qubit'
$\|0\rangle$	$\|0\rangle$	$\|0\rangle$	$\|0\rangle$
$\|0\rangle$	$\|0\rangle$	$\|1\rangle$	$\|1\rangle$
$\|1\rangle$	$\|1\rangle$	$\|0\rangle$	$\|0\rangle$
$\|0\rangle$	$\|1\rangle$	$\|1\rangle$	$\|1\rangle$
$\|1\rangle$	$\|0\rangle$	$\|0\rangle$	$\|0\rangle$
$\|1\rangle$	$\|0\rangle$	$\|1\rangle$	$\|1\rangle$
$\|1\rangle$	$\|1\rangle$	$\|0\rangle$	$\|1\rangle$
$\|1\rangle$	$\|1\rangle$	$\|0\rangle$	$\|0\rangle$

Table 3.2 Toffoli gate truth table showing target qubits before and after applying the Toffoli gate - both control qubits states

3.8.2 Controlled SWAP gate



Another three qubit gate is controlled-SWAP gate, it can be described by the matrix:

$$CSWAP = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.19}$$

As all the controlled gates this gate only performs its action when the control qubit is in state $|1\rangle$ - at which point the gate performs SWAP operation of the two target qubits.

3.9 Reaching Universality

Before we can construct any quantum hardware it is also important to explore gate identities, which allow us to perform certain transformations by using other more trivial transformations. This allows us to create universal quantum hardware with minimal number of physical types of quantum gates since we can perform more advanced operations using the simpler gates.

In classical computers we can realize all other logical gates with NAND and NOR gates. This approach does not only make mass production easier, it also decreases the number of logical gates that we must design and calibrate hence making an overall development of hardware an easier and faster process. This in quantum hardware will be incredibly useful since their realization is already hard.

Perhaps the most simple of identities includes phase shift gate notably:

$$TT = S \quad \text{—}\boxed{T}\text{—}\boxed{T}\text{—} = \text{—}\boxed{S}\text{—} \tag{3.20}$$

Other gate identities include:

$$HH = H^2 = I \quad \text{—}[H]\text{—}[H]\text{—} = \text{—}[I]\text{—} \tag{3.21}$$

$$X^2 = I \quad \text{—}[X]\text{—}[X]\text{—} = \text{—}[I]\text{—} = \text{—} \tag{3.22}$$

$$Y^2 = I \quad \text{—}[Y]\text{—}[Y]\text{—} = \text{—}[I]\text{—} = \text{—} \tag{3.23}$$

$$Z^2 = I \quad \text{—}[Z]\text{—}[Z]\text{—} = \text{—}[I]\text{—} = \text{—} \tag{3.24}$$

$$SWAP_{12} = CNOT_{12}CNOT_{21}CNOT_{12} \quad \text{—×—} = \text{—●—⊕—●—} \tag{3.25}$$
—×— —⊕—●—⊕—

When it comes to reaching universality with quantum hardware we can either use a combination of Toffoli gate with any unary operator with real coefficients α and β such as Hadamard gate. We can also reach universality with a combination of CNOT, T gate and once again Hadamard gate. [27]

3.10 Current state of Quantum Devices

3.10.1 Physical devices

The physical realization is probably the hardest challenge when it comes to quantum computing as as stated before, at least that is the case for now. There are several proposed theoretical solutions that could be used to physically realize a qubit [27] however they wont be described here as they are out of the scope of this work.

As of writing this paper quantum computing is in its very first stages when it comes to hardware, edge quantum devices reach around 10^2 qubits and are quite expensive to create and maintain. Recent examples of quantum hardware would include IBM's Quantum Eagle - 127 qubit quantum device or the Chinese Zuchongzhi (祖冲之) - 62 qubit Quantum device. By 2023 IBM want's to introduce a quantum machine with 1121 qubits which would be a milestone pushing us into 10^3 ranges. Not only the low number of available qubits limits the potential and scale of quantum applications that can be created it also is prone to high error from the environment caused by physical nature of quantum devices. Naturally this calls for simplicity within each quantum algorithm and application, resulting in hybrid solutions where quantum hardware is only responsible for the part of the application where a quantum advantage can be achieved, while the rest of the solution is delivered by classical hardware which for now is far more stable and reliable medium. This approach is even more reasonable when we realize that classical computers are already used to control inner workings of quantum computers.

3.10.2 The road to a million quantum states

Speaking of reliability, it is estimated that in order for any sort of proposed error correction algorithm similar to the ones that are used today in classical computers to be effective the quantum device should be at least in the ranges of 10^6 [39] qubits and the error

rate itself should not be bigger than 10^{-6}. While this number may seen quite big when compared to current capabilities of quantum devices many people are pointing out the similarities between current advancement of quantum technology to the fifties/sixties era of classical computers [7]. Reaching this milestone should give humanity - or whoever controls this device - quantum supremacy allowing to reap all the benefits of already proven quantum advantage algorithms and likely the many more that will be developed in the future. Until quantum hardware reaches this state we cannot reasonably think of quantum solutions which are fully independent from classical computers within the scope of algorithms not to mention fully independent quantum devices not using any classical computers as control units. We can't also think of implementing some of the quantum algorithms with proven quantum advantage reliably because of these limitations as they may be sensitive to errors.

3.10.3 Simulations

There exist several methods of simulating quantum computations on classical hardware, but this approach fails very quickly when the number of q-bits increases. To store an n qubit system on classical hardware we need to store all complex numbers describing the system's wave function - in practice this means 2^n complex numbers to store - an exponential increase, which should be a red flag for everyone, not only complexity theorists. This very fact limits possibilities of quantum development on classical computers and allows only for small applications to be simulated such as these presented in this work.

3.11 Quantum Programming

The art of quantum programming - that is programming a quantum device to perform desired tasks is realized with higher level programming languages, over the years quantum simulators were developed for many languages such as: C++, Java, Python, Rust and F# [3]. Several language were made specifically for the task of quantum programming notably IBM's openQASM and Microsoft's Q#, which more importantly can be both used to program real quantum devices which is not necessarily the case for QC libraries from previously mentioned languages.

The code itself usually will consist of a set of instructions that tell the simulator or the real device how our quantum circuit is composed - what kind of transformations (or gates) will be used, what are the inputs, which qubits to measure etc meaning it is often only composed of high level instructions making the process of learning such library quite easy provided that we know the theory of QC.

```
import qiskit as q
from qiskit.visualization import plot_bloch_multivector
from matplotlib.pyplot import close
from matplotlib.pyplot import show
# close all figures
close('all')
#%% Build circuit
qcSize = 3  # size of our quantum circuit
qc = q.QuantumCircuit(qcSize)

qc.x(1) # apply CNOT gate on qubit 2
qc.h(2) # apply hadamard gate on qubit 3

```

```
qc.draw(output='mpl'))
#%% Simulate!
qSim = q.Aer.get_backend('aer_simulator') # get backend for simulating
    quantum device
qc.save_statevector() # save the statevector
qobj = q.assemble(qc)                    # compile quantum circuit
result = qSim.run(qobj).result()         # run our simulation
exitStates = result.get_statevector()    # save quantum states

plot_bloch_multivector(exitStates,'simple quantum states visualized') #
    plot all 3 qubits states
show()
```

Listing 3.1 Example code from IBM's qiskit [1] python library which displays quantum states on 3 bloch spheres shown earlier in figure 3.2

Chapter 4

Implementation

4.1 Datasets

4.1.1 MNIST dataset

One of the more commonly used datasets in machine learning research is the MNIST dataset[4] (Modified Institute of Standards and Technology dataset) containing 70000 images of Arabic digits. I have also decided to use it as a first dataset, as its images are only 32 by 32 pixels in size and thus, despite the big size of the dataset it can be still processed and trained upon relatively fast. This is especially important when we consider that parts of the neural networks used in training are run on simulated quantum devices which as described in subsection 3.10.3 require a lot of computer resources to run, hence this dataset allowed for experimentation within reliable time-frame.

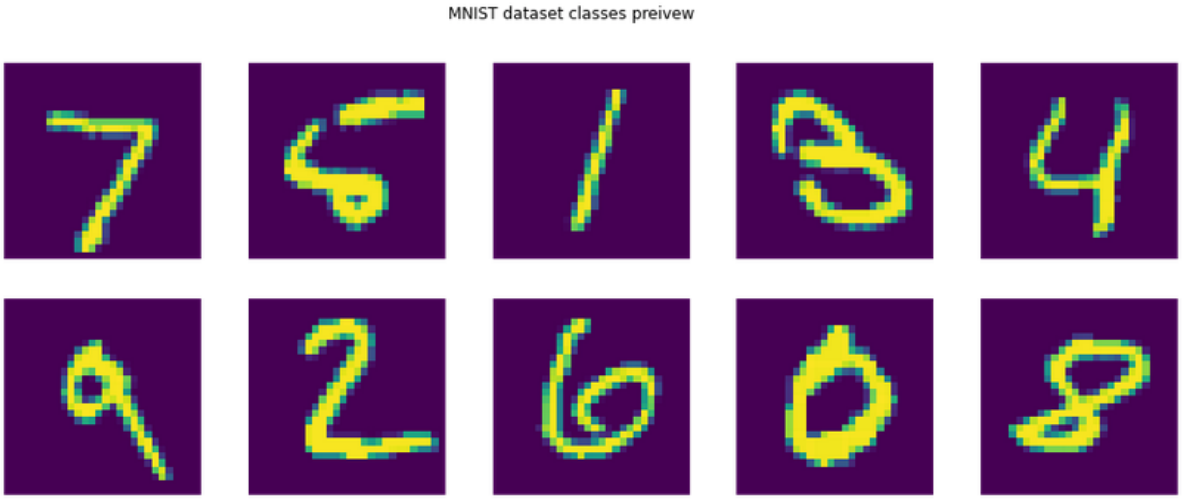


Figure 4.1 All ten classses contained within the MNIST dataset

4.1.2 Stanford Dogs Dataset

The Stanford Dogs dataset [6] is quite a challenging dataset - it contains 120 classes of different dog breeds each with around 100-200 (20580 images in total) images, which were collected from the imagenet [2] dataset mentioned in chapter 2. Small number of images

per class, huge number of classes, similarity between some dog breeds and many dogs with other animals or items appearing in one photo make achieving a high accuracy on this dataset quite a hard challenge.

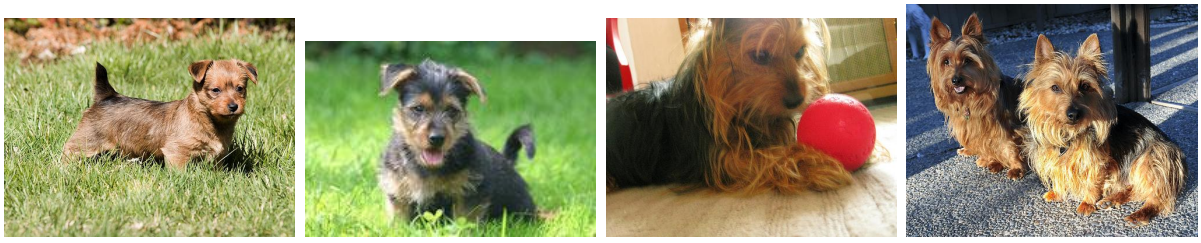
(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j) (k) (l)

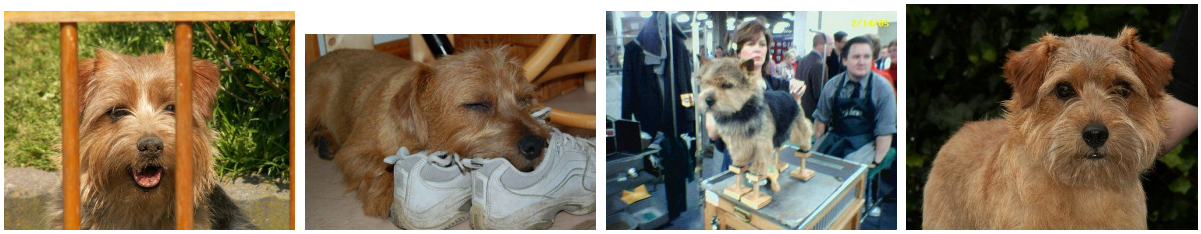
Figure 4.2 Preview of some images from the Stanford Dog dataset

This dataset was used for experimentation with quantum transfer learning - since some classes are very similar to each other and QTL was only experimented upon within a binary classification variant - more classes would require more qubits and way more time to simulate this architecture in more sophisticated variant. The exact dog breeds chosen were *Australian Terrier* and *Norfolk Terrier* which from now on shall be referred to as *quantum terriers*. Those 2 breeds are as their name would in fact suggest to us very similar to each other as presented on figures 4.3 and 4.4.



(a) (b) (c) (d)

Figure 4.3 Preview of quantum Australian terriers



(a) (b) (c) (d)

Figure 4.4 Preview of quantum Norfolk terriers

4.1.3 NIST dataset

For experimentation with Quantum generative adversal neural networks the architecture proposed by *Huang et alia* [29] which was discussed in subsection 4.3.3.

The exact letters chosen were **J**, **D** and **I** - these letters were chosen to see how well the network can recreate the images depending on the complexity of the letter. Handwritten D is usually very simple while handwritten J and I not necessarily so.

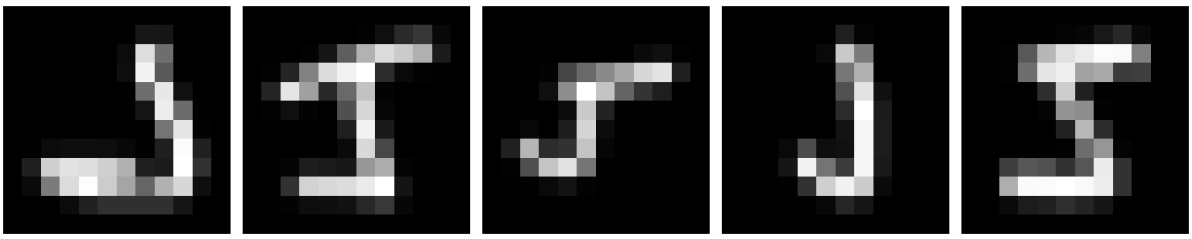


Figure 4.5 Some of the images of letter J contained within the dataset

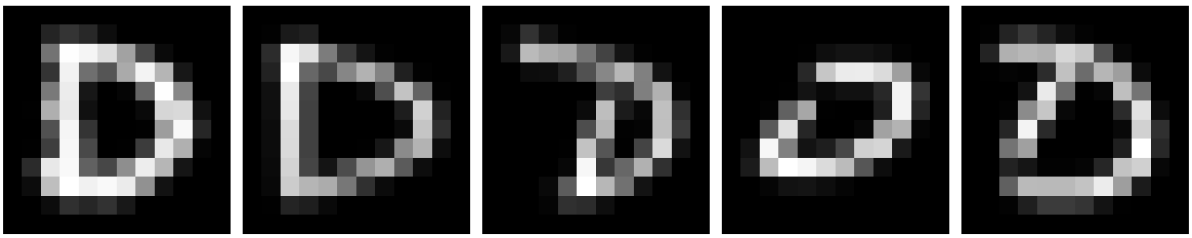


Figure 4.6 Some of the images of letter D contained within the dataset

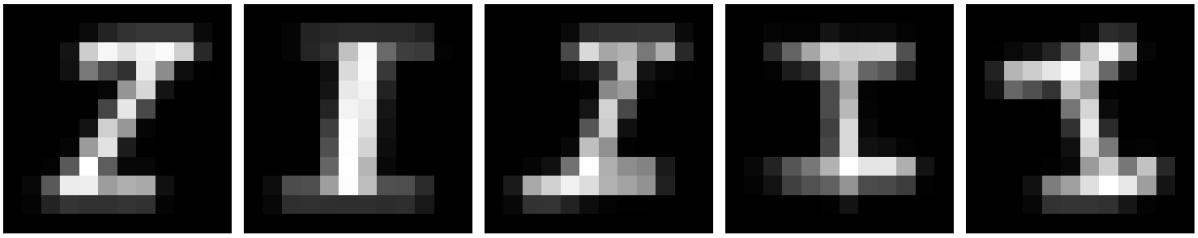


Figure 4.7 Some of the images of letter I contained within the dataset

4.2 Used technology

Python programming language was the primary tool used for this work. TensorFlow and PyTorch libraries were used for the purposes of developing neural networks while Pennylane was used to ensure easy integration of QC simulations with neural networks. IBM's Qiskit was used for learning quantum computing, as well as figures used in chapter 3 - review of quantum information theory.

4.3 Description of used hybrid algorithms

4.3.1 Quantum-Classical Convolutional Neural Networks (QC-CNN)

The first explored HNN was hybrid convolutional neural network - one where convolution layers are replaced with quantum circuits, which act in their place. While the number of qubits with this approach grows quadratically we can still easily simulate implementations where the convolution matrix is of sizes 2x2 and 3x3. This does not only prove useful for research purposes but means that this algorithm can even be implemented and run on small quantum devices, since it is common for convolutional networks to use convolution matrices of this sizes.

The specific explored architecture was the "quanvolutional" neural network proposed by *Hunderson et alia, 2020* [26]

One method of initialising the weights of quantum convolution circuits explored in Hunderson's work was random weight initialisation - this is the architecture I have decided to explore. Random weight initialization of convolution layers at first point may seem counter-intuitive, yet it is a proven method[23][14] in classical convolution networks, which allows to speed up training process since some weights don't need to be trained but just initialised from some chosen random distribution $\mathcal{D}$.

The quantum convolution circuit (QCC) was explored in three variants, each one with different amount of layers with randomly initialised weights - experimentation was done with 1 layer QCC, 3 layer QCC and 10 layer QCC. Figure 4.8 shows a diagram of how the used quantum layer works.

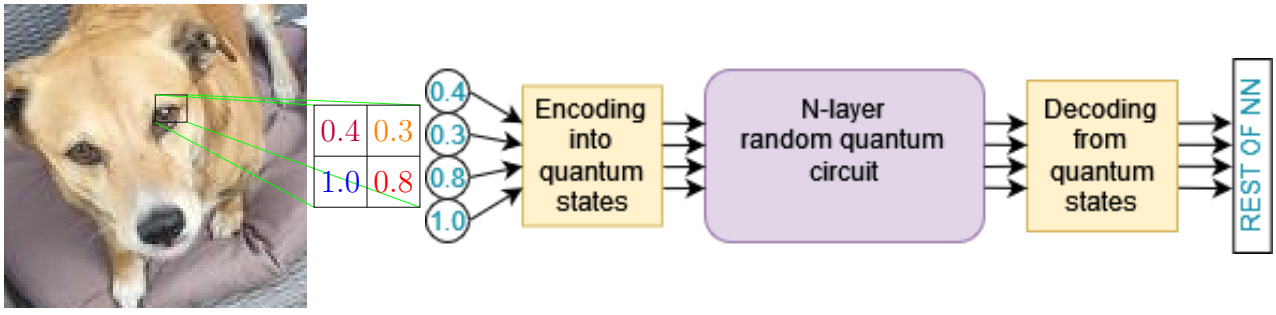


Figure 4.8 A diagram showing of how quantum convolution layer works

In the classical analogue of convolutional neural network I have used normal convolutional layers with same parameters as the quanvalutional one - convolution matrix size is 2x2. The rest of the architecture of the neural networks remains the same within both classical and hybrid analogues - densely connected layer after convolution layers.

Experimentation with HNN was performed with 4 qubit 2x2 convolution matrix on full MNIST dataset described in subsection 4.1. As a point of reference the prepossessing of the whole dataset with quantum convolution layer took about 4 and a half days in the most time consuming variant - 10 layer quantum convolution circuit. The results of all variants will be analyzed in section 5. They were all saved and are available in the resources section of this paper.

4.3.2 Classical-Quantum (CQ) transfer learning

Another scope of experimentation was quantum transfer learning - specifically the approach when we apply quantum layer to pretrained classical network - or simply classical-to-quantum transfer learning. The pretrained weight are loaded from a ResNet16[25] network trained on imagenet[2] dataset. The trainable part consists of so called dressed quantum circuit introduced by *Andrea et alia* [8]

The dressed quantum circuit is a trainable quantum layer with classical layer before it and after it, which act as encoders and decoders of quantum data respectively. 2n qubits - where n is the number of classes - are used for the quantum part of the dressed circuit. At the end there is one last layer using softmax activation function with n neurons corresponding to n classes.

The quantum circuit itself consists of entangling layer which uses Hadamard gates to entangle the state of each qubit. Then RY gates are used to encode the data into the qubits after which follows an n-depth layer of RY gates, which are trainable parameters within the whole network. All of this is followed by Z gates since we are measuring the data in Z basis. Figure 4.9 visualizes this design.

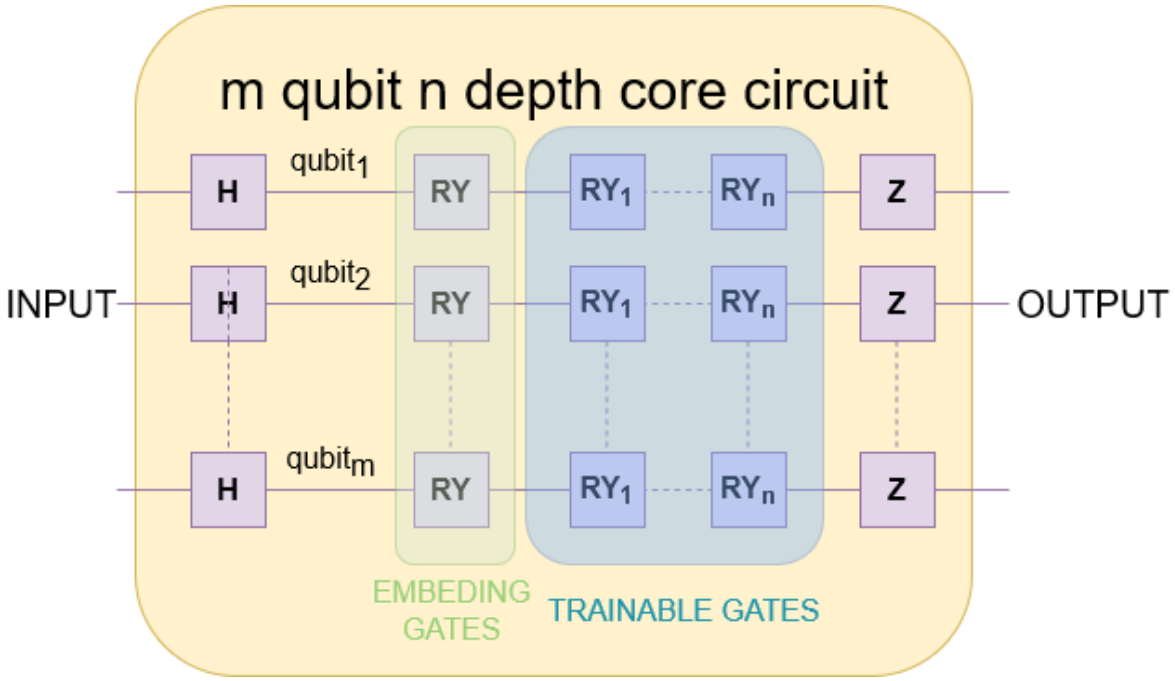


Figure 4.9 Circuit diagram of the quantum circuit sitting at the core of the dressed quantum circuit

Since this particular implementation requires $2\dot{n}$ qubits (n being the number of classes) experimentation was only performed on binary dataset - one containing only 2 classes -

to avoid long experimentation times caused by requirement to use more qubits within the simulations.

Figure 4.10 shows the specific implementation used in the experimentation done with this QML method. ResNet16 outputs 512 features meaning that our encoding layer of the dressed quantum circuit needs to process it down into 4 features - since 4 qubits were used. The output layer consists of 2 neurons since experimentation was performed on a binary dataset.

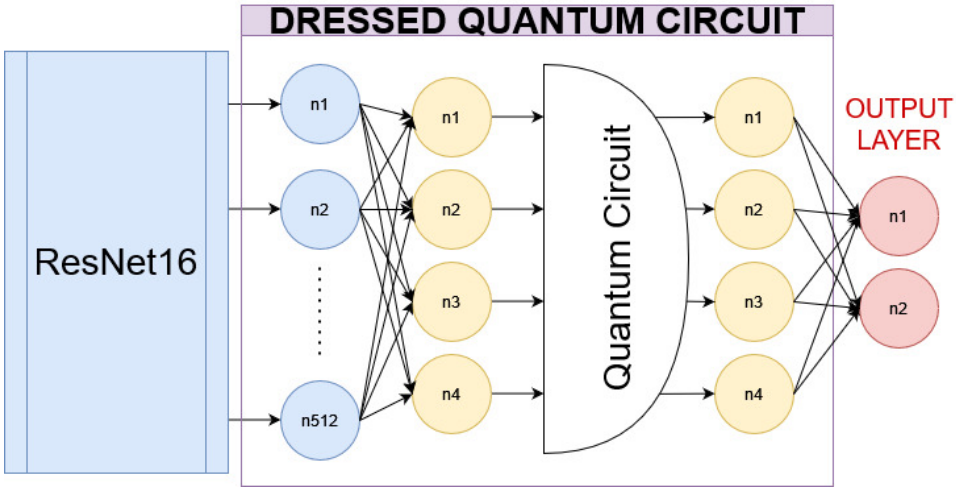


Figure 4.10 Diagram showing in detail the implementation of the dressed quantum circuit used in the transfer learning experimentation

4.3.3 Quantum Generative Adversal Networks

The Quantum Generative Adversal Networks (QGAN) was the last architecture explored within this paper. It is based upon paper by *Huang et alia* [29], which proposes both implementations of quantum discriminator and generator alike.

Exploration was performed with quantum generator and classical discriminator - thus it still remains a hybrid implementation which uses one wholly quantum network and one purely classical network in its implementation. The quantum generator works by implementing sub-generator that generates patches of the image, which then are added together to form the final image as seen in diagram on figure 4.11. This approach allows to implement this architecture for bigger pictures even when the number of qubits is limited, since we can always use more sub-generators. This was in fact used within this particular implementation to successfully learn on images of size 12 by 12 pixels despite using the same numbers of qubits (4) as the authors of the architecture who experimented with 8 by 8 pixel images.

The Quantum Generator Network Circuit consists of firstly RY gates which set the qubits into random states - that represent the noise. Then an n layer parameterized RY gates are used, which are the trainable part of the generator. At the end a layer of Z gates is placed since the measurement is taken in Z-basis.

Experimentation with this architecture was performed on letter dataset described in subsection 4.1 which was resized to 12x12 pixels to limit simulation time. The architecture

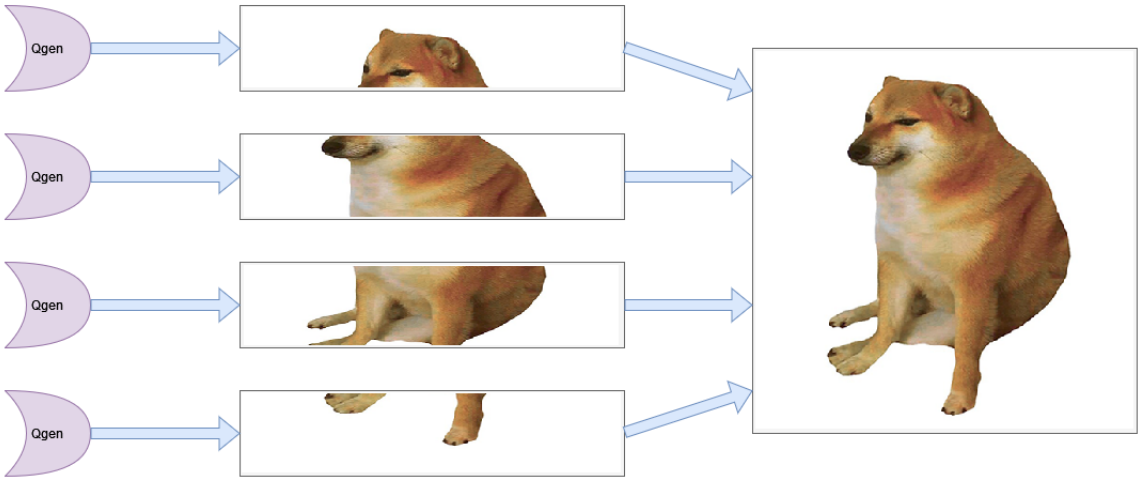


Figure 4.11 Diagram showing the idea behind the explored 'patchy' architecture of a QGAN

of the quantum generator network was implemented by using 5 qubits per sub-generator with 9 sub generators.

Chapter 5

Results

This chapter showcases the results of experimentation done within scope of this bachelor's thesis based on earlier described theory and HNN architectures.

5.1 QCCNN with MNIST

The Quantum Classical Convolutional Neural Network was, as described in the previous sections, analyzed with MNIST dataset [4]. The results for all three different depth quantum convolution layers as well as classical analogue CNN and ANN are showcased on figure 5.1 and figure 5.2. Additionally figures 5.4 and 5.3 show the difference between accuracy of all models excluding ANN in comparison to the QCCNN with 1 quantum layer.

All tested hybrid implementations of QCCNN **outperformed** classical analogue CNN. The classical ANN despite performing the worst still performed reasonably well most likely due to small dimensions of input data.

5.1.1 The 10 layer question

The 10 layer quanvolution circuit surprisingly performed the worst out of all three QCCNN's - this would suggest that adding too much of random quanvolution layers is harming the performance of the NN in the long run. It can also be noticed that the 10 layer network had the worst loss and accuracy out of all tested architectures and only barely managed to overcome the classical analogue in the later epochs of training. Additionally figure 5.4 clearly shows that the 10 layer QCCNN had the accuracy way lower than the other networks at the start and it only managed to surpass the classical CNN by the end of the training - even then it did barely surpass it in performance.

In the quanvolution paper [26] the results of increasing layers was increase in accuracy when it came to non random layers however there was no such comparison showcased for the random layers - the authors only compared random layers performance to non random one which seemed to indicate similar performance. The results performed here would indicate that more experimentation is required when it comes to randomly initialised weight in QCCNN's.

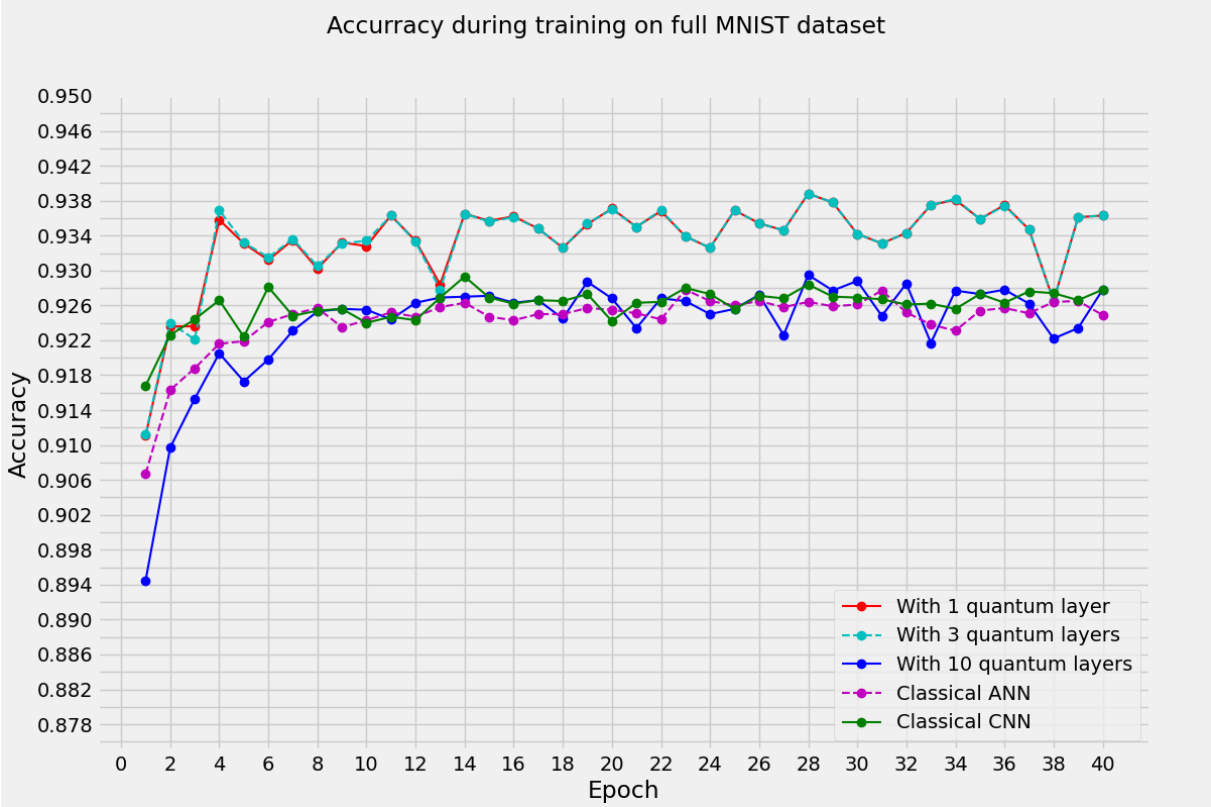


Figure 5.1 Accuracy on test dataset while training QCCNN model on MNIST dataset

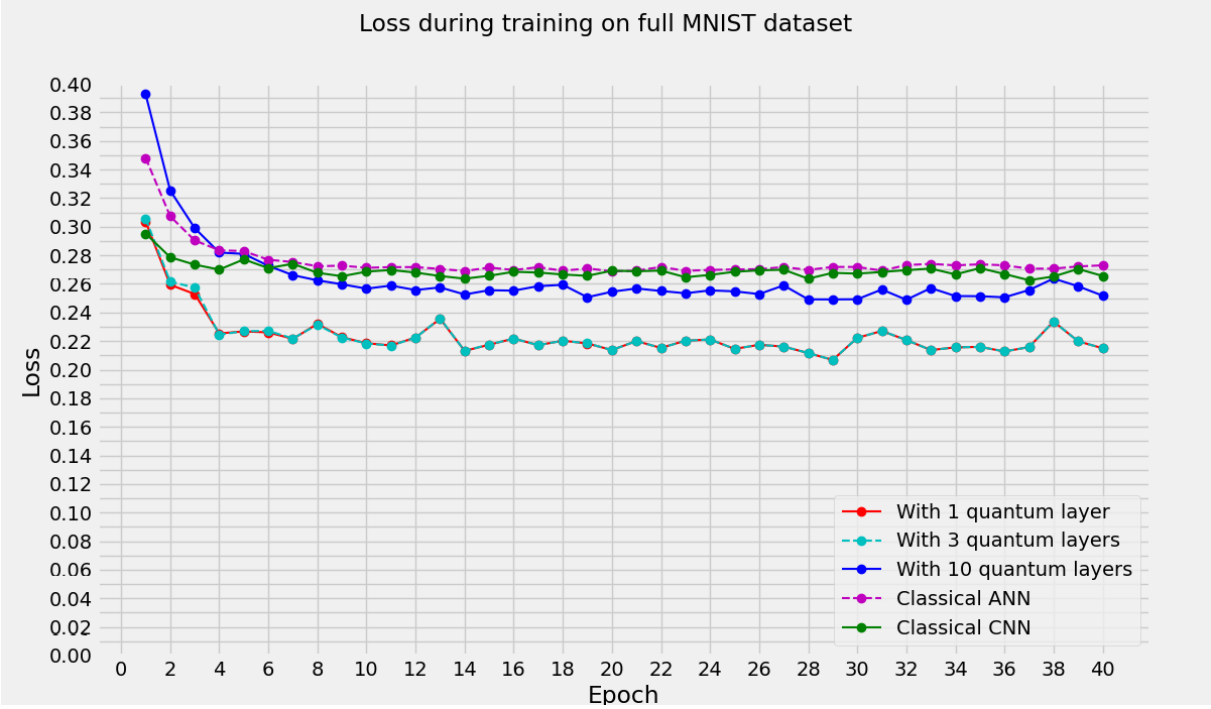


Figure 5.2 Loss on test dataset of QCCNN models on MNIST dataset

5.1.2 Results for other depth QCCNN's

The 1 layer and 3 layer QCCNN's both performed very similarly, the only difference was visible in the first few epochs - no design was clearly outperforming the other during that

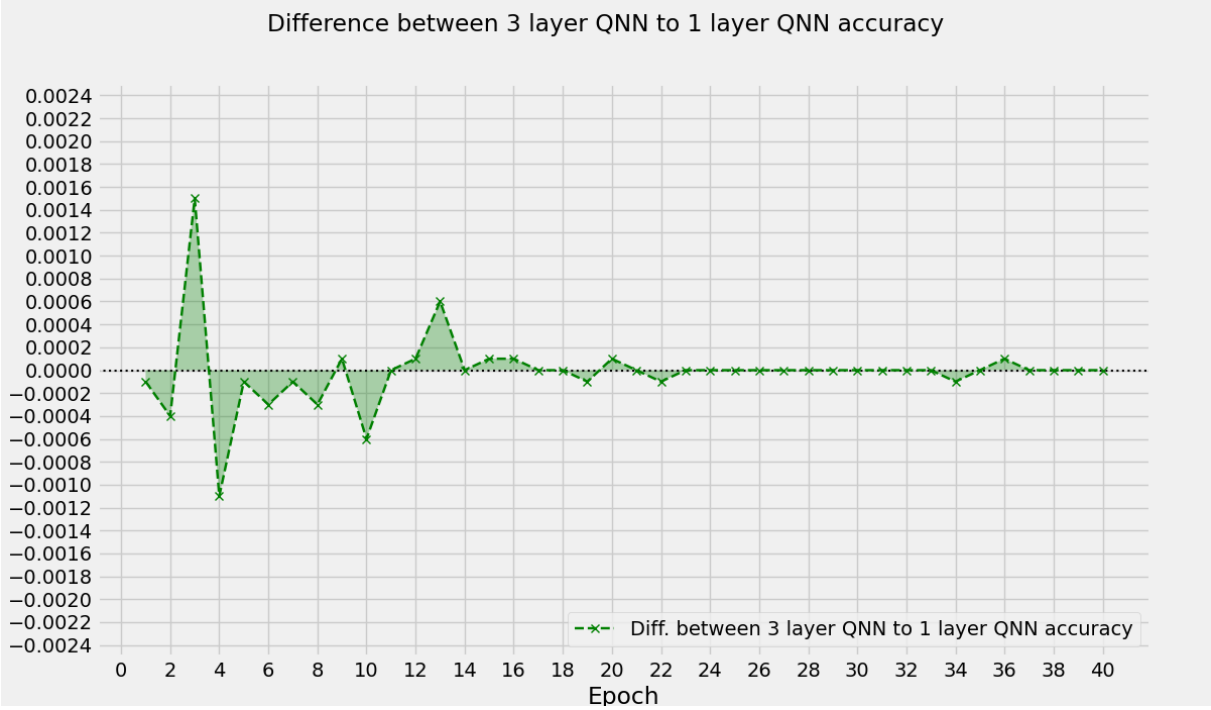


Figure 5.3 Loss on test dataset of QCCNN models on MNIST dataset

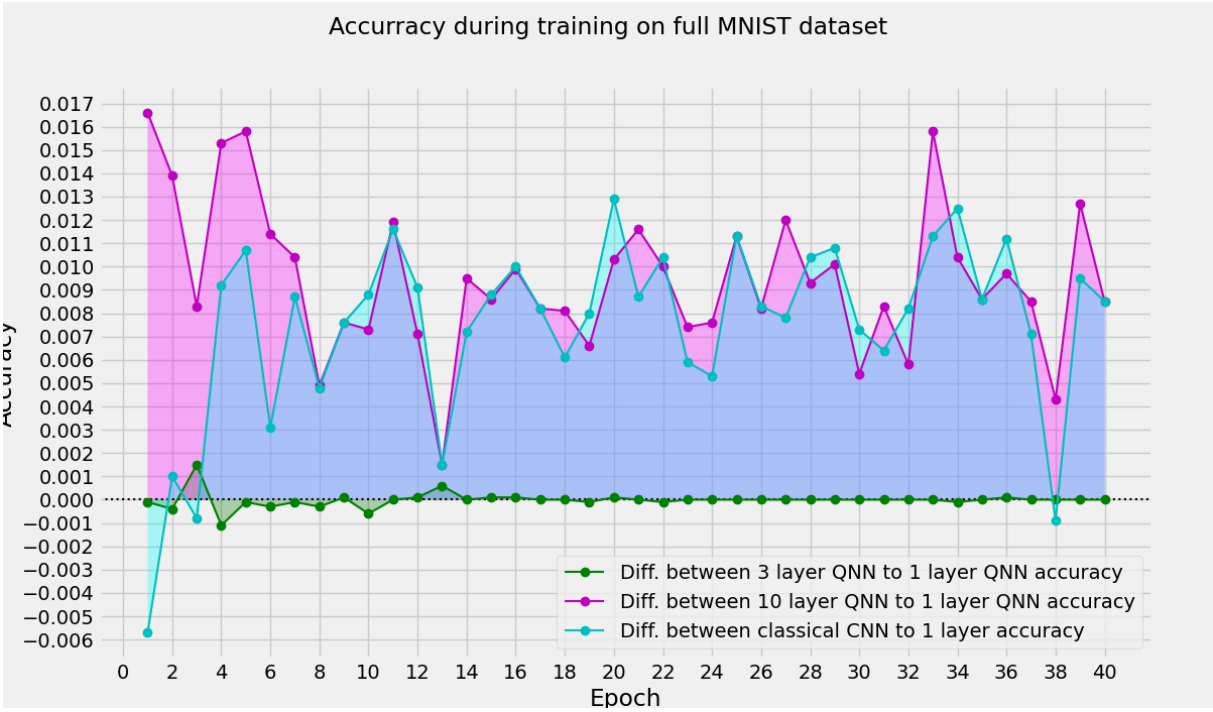


Figure 5.4 Loss on test dataset of QCCNN models on MNIST dataset

epoch range. During the later epochs the results for both architectures clearly converged into the same accuracy and loss range and both architectures performed almost the same as seen in all figures in this subsection, especially figure 5.3.

5.1.3 QCCNN's - conclusions

While the described examples clearly show HNN outperforming classical analogue NN, it in no means shows a quantum advantage over the classical NN, which is yet to be seen in research. The accuracy in CNN would still be increased if a more complex architecture would be used without requiring a quantum computer or exponential time increase of QC simulations done on classical computer. When something like that is proven through research only then will we be able to talk about the viable case for quantum convolutional neural networks.

It is also important to state that all HNN implementations had way more variation between accuracy in each consecutive epoch when compared to entirely classical approaches. This phenomena can be seen on figure 5.1 and especially figure 5.2.

5.2 CQ Transfer learning approach

This subsection explored the results of experimentation done with hybrid classical to quantum transfer learning. The Quantum Transfer Learning algorithm used dressed quantum circuit as proposed by *Andrea et alia* [8] which was discussed in subchapter 4.3.2, while the classical analogue simply used one hidden layer after ResNet16 with pretrained weights.

5.2.1 Raw results

Initial experimentation was performed without using data augmentation. The quantum transfer network used depth of 5 within the circuit since the original paper experimented with similar depths - 4, 5 and 6 [8]. The learning rate parameter was set at 0.0004, the results for accuracy and loss function during training can be seen on figures 5.5 and 5.6.

5.2.2 Results with augmented dataset

Next the same parameters networks were trained with augmented dataset. The augmentation used was random horizontal flipping, cropping the image, slight change in colors as well as addition of Gaussian noise to the picture. The results of accuracy and loss during training are shown in figures 5.7 and 5.8 respectively.

Even though the hybrid architecture appear to fare better when it comes to accuracy as seen on figure 5.7 the loss function on figure 5.8 for the hybrid architecture looks worse than the classical one, the difference seems to be bigger than during the training on not augmented dataset which is shown on figure 5.6.

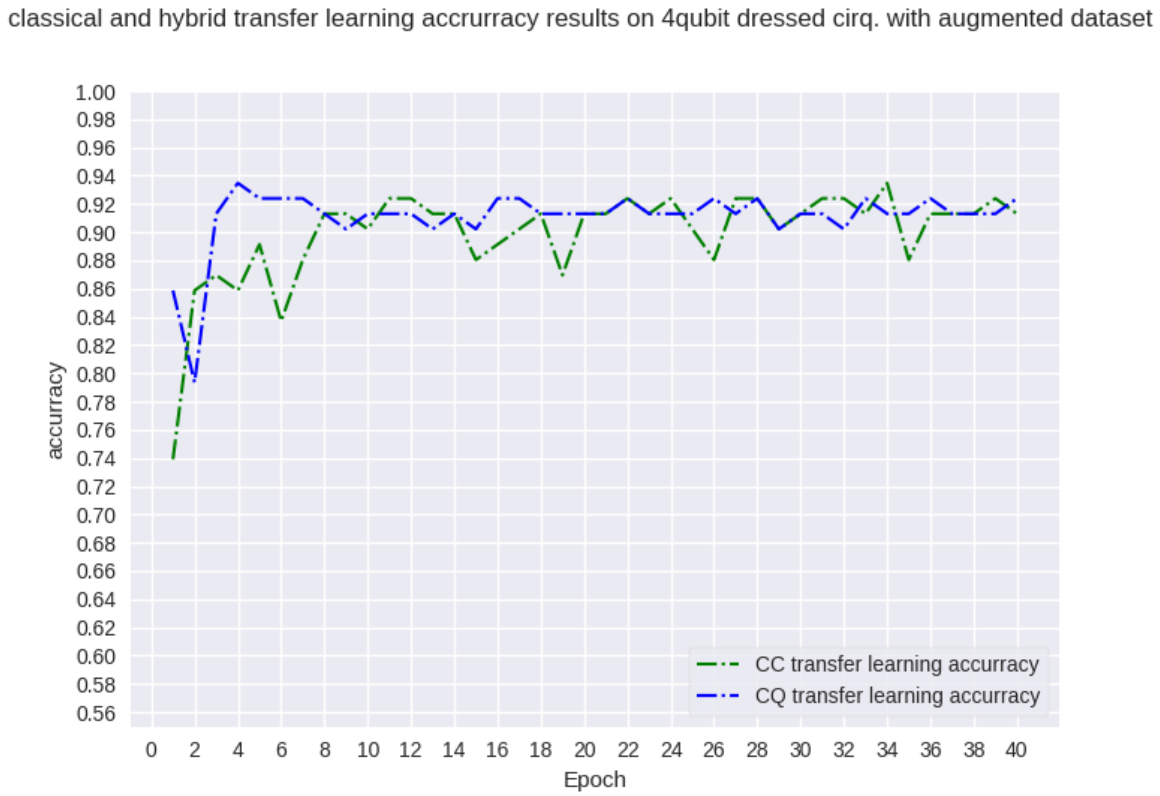


Figure 5.5 Accuracy during training on not augmented dataset

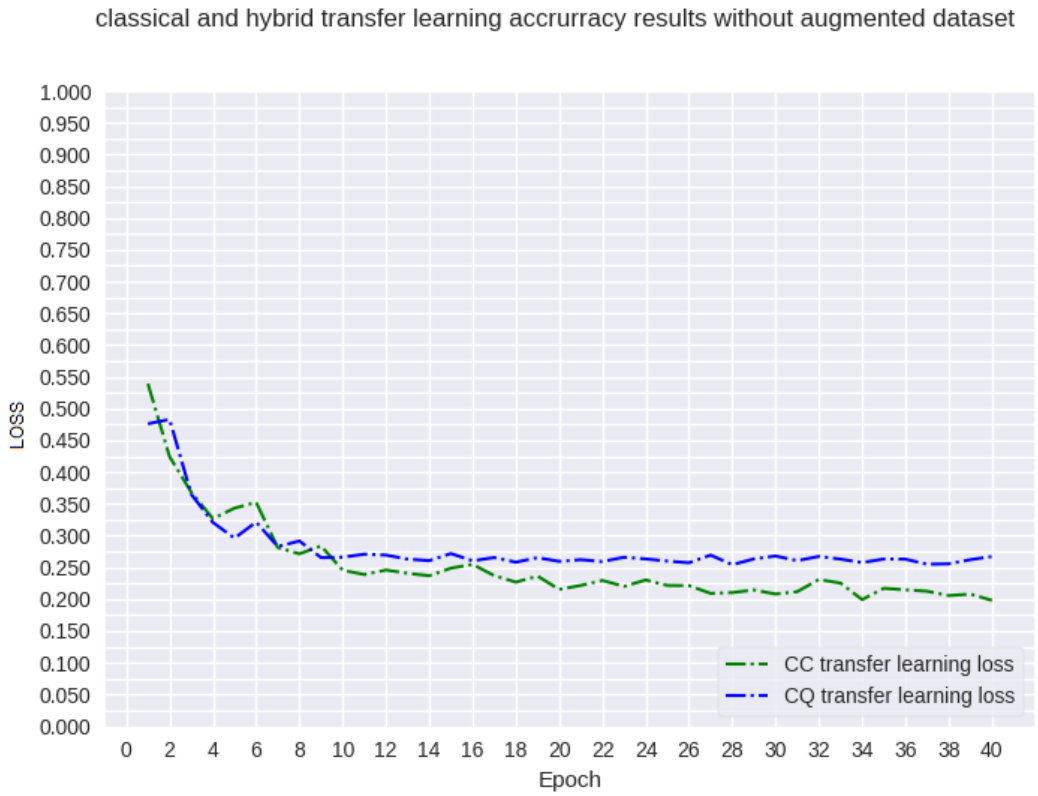


Figure 5.6 Loss during training on not augmented dataset

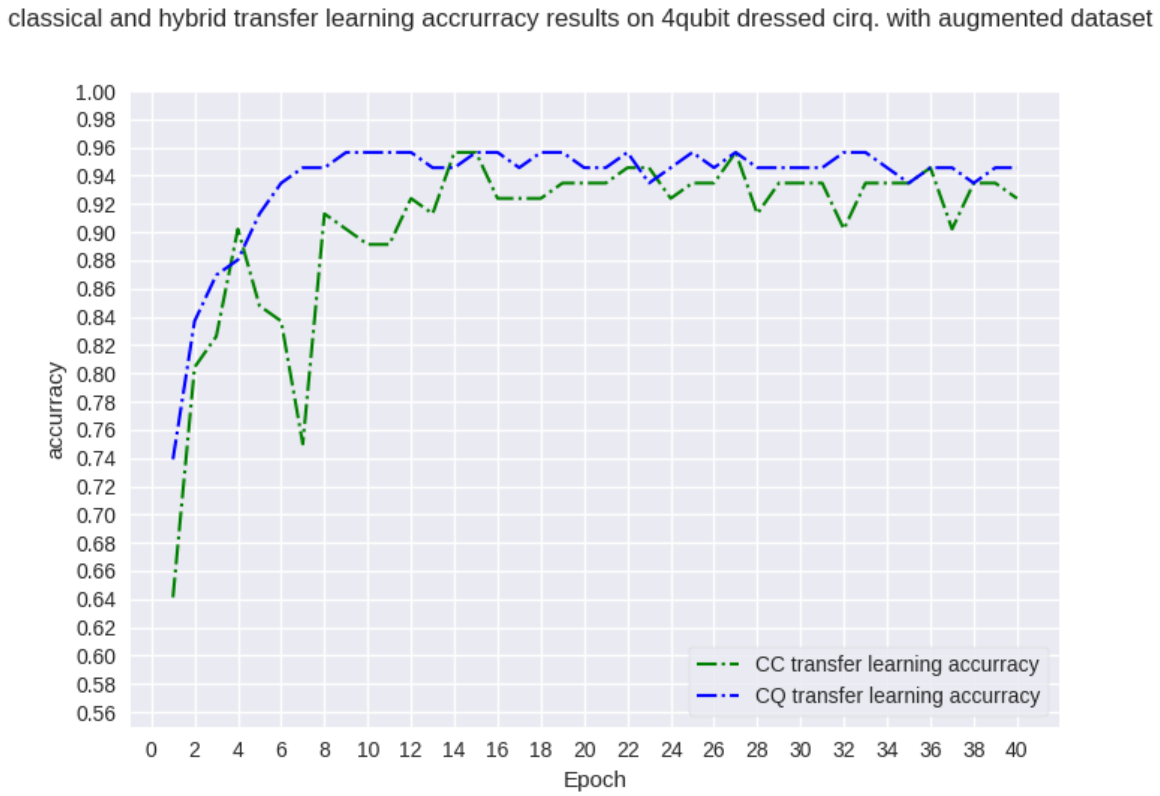


Figure 5.7 Accuracy during training on augmented dataset

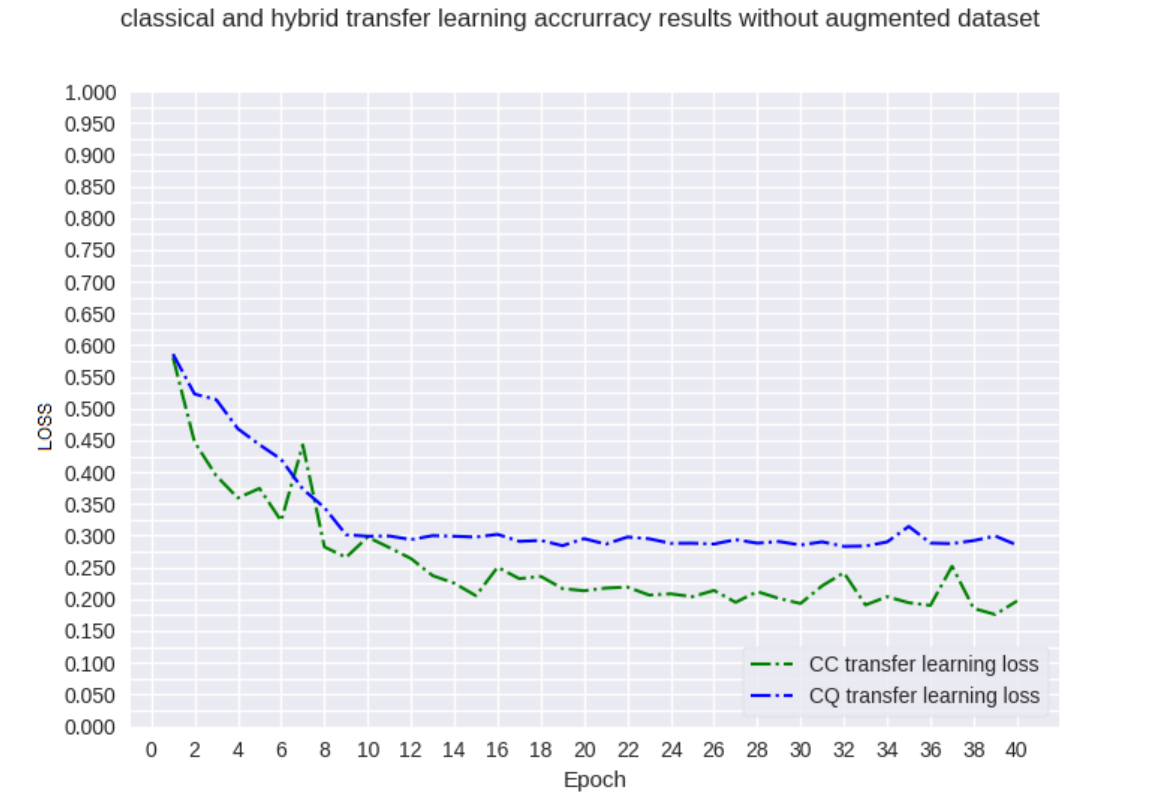


Figure 5.8 Loss during training on augmented dataset

5.2.3 Circuit depth influence on network performance

Next the hybrid transfer architecture performance was tested depending on the depth of the quantum circuit. The experimentation was performed with depths ranging from 2 to 8 layers, the performance was judged based upon best accuracy the model would achieve as well as the average accuracy of the model between epoch 11 and 40.

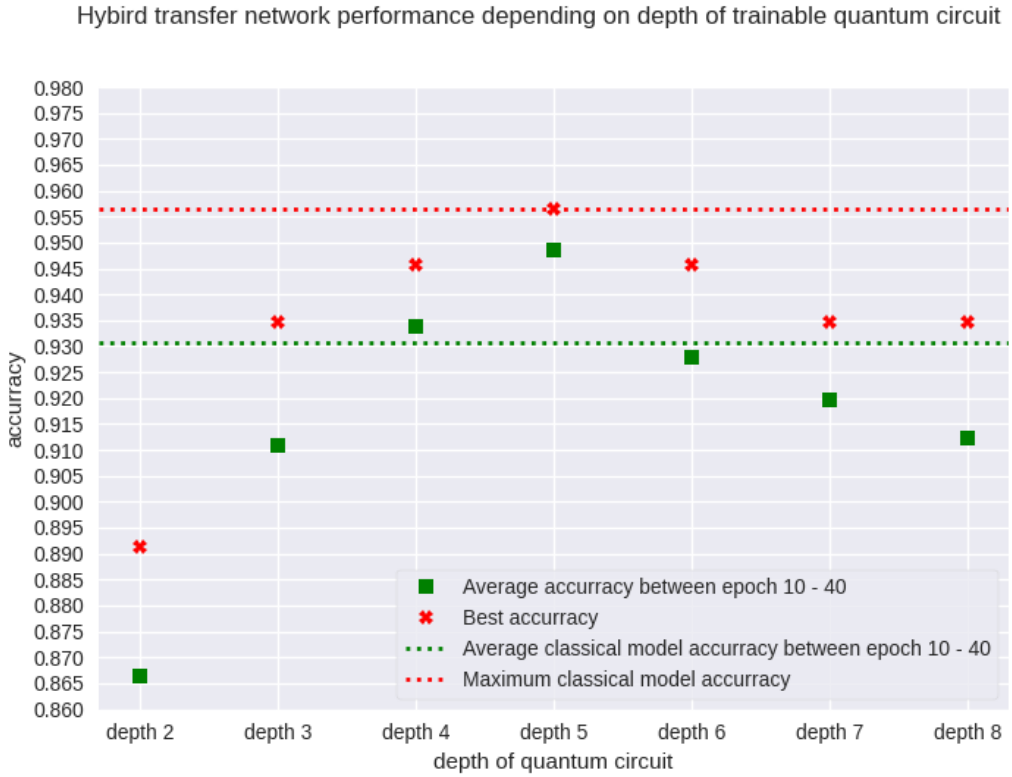


Figure 5.9 Influence of depth of core circuit on accuracy

The results visible on figure 5.9 indicate that a circuit with 5 trainable layers of RY gates would be best suited for this particular dataset - it is important to point out that this can very depending on the dataset as shown in Andrea's work[8]. With this architecture the depth becomes itself an additional hyperparameter we can tune.

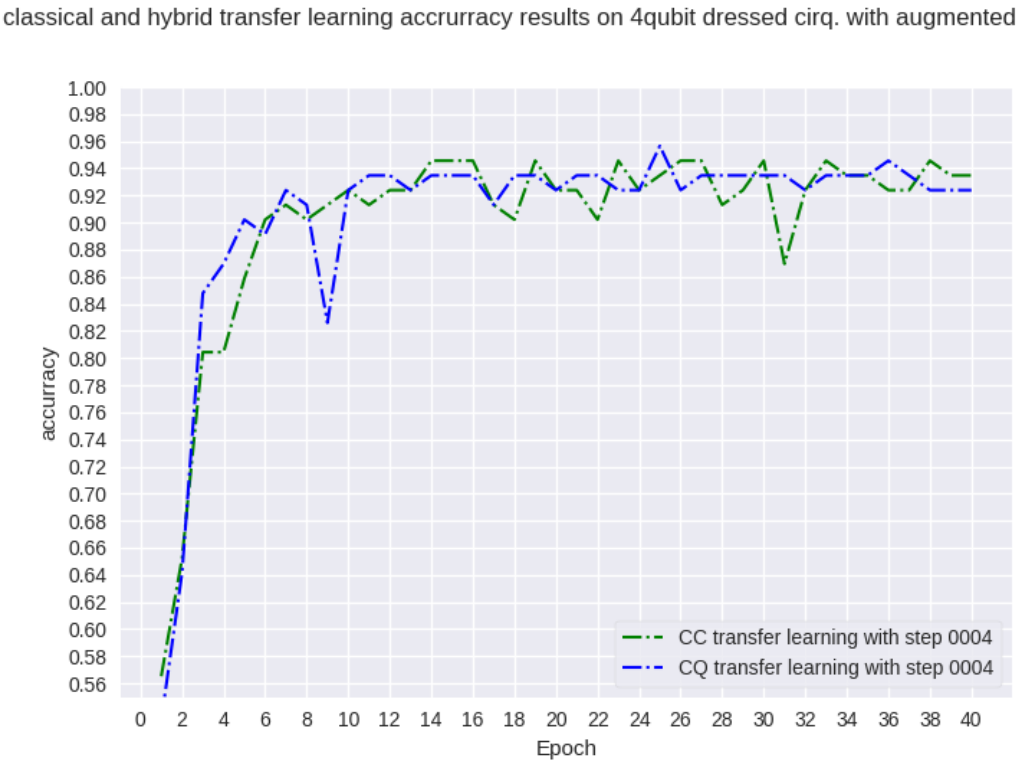


Figure 5.10 Accuracy during training of transfer models with learning step 0.0004 and quantum circuit depth of 5

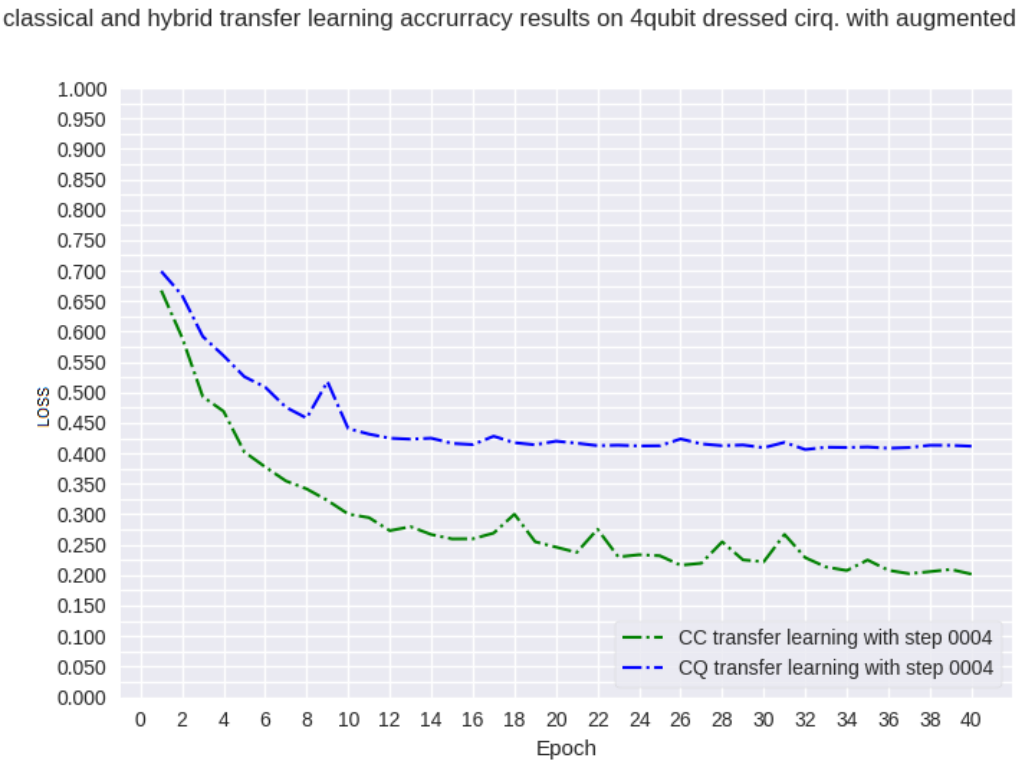


Figure 5.11 Loss during training of transfer models with learning step 0.0004 and quantum circuit depth of 5

Figures 5.10 and 5.11 show the example of worse performance achieved with dressed circuit of 6 layers. The network Does not achieve higher accuracy than it's classical counterpart. Interestingly the loss function from training the hybrid architecture vastly underperforms when compared to the classical analogue as seen on figure 5.11 and earlier hybrid examples from figure 5.5 and 5.7.

5.2.4 Learning rate influence on networks performance

Next, few different values of learning rate were analyzed to determine their impact on accuracy from training the neural network. To determine the best learning step the same approach was used as with determining optimal circuit depth. The results can be seen on figure 5.12. As we can see the hybrid neural network was much more sensitive to changes of this hyperparameter, the optimal learning rate from the few that were explored turned out to be 0.00045 or 0.0004 - we can see that with this value of learning rate the hybrid network managed to achieve higher maximum accuracy that the classical network, however on average the network achieved higher accuracy with learning step of 0.00045. The results of training both classical and hybrid networks with value of 0.00045 learning rate are shown on figure 5.13 and 5.14. We can see on figure 5.14 that the difference between loss of CQ and CC transfer learning architectures was even bigger than before with architectures of these depth (figures 5.5 and 5.7).

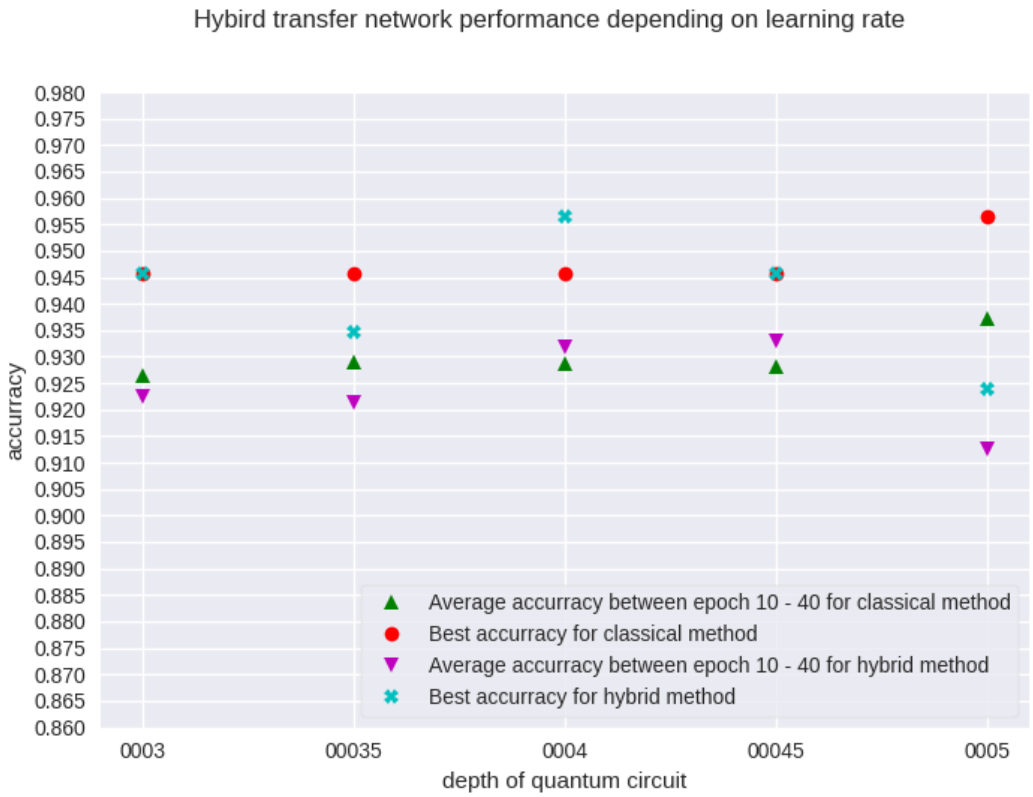


Figure 5.12 Influence of learning rate parameter on network accuracy

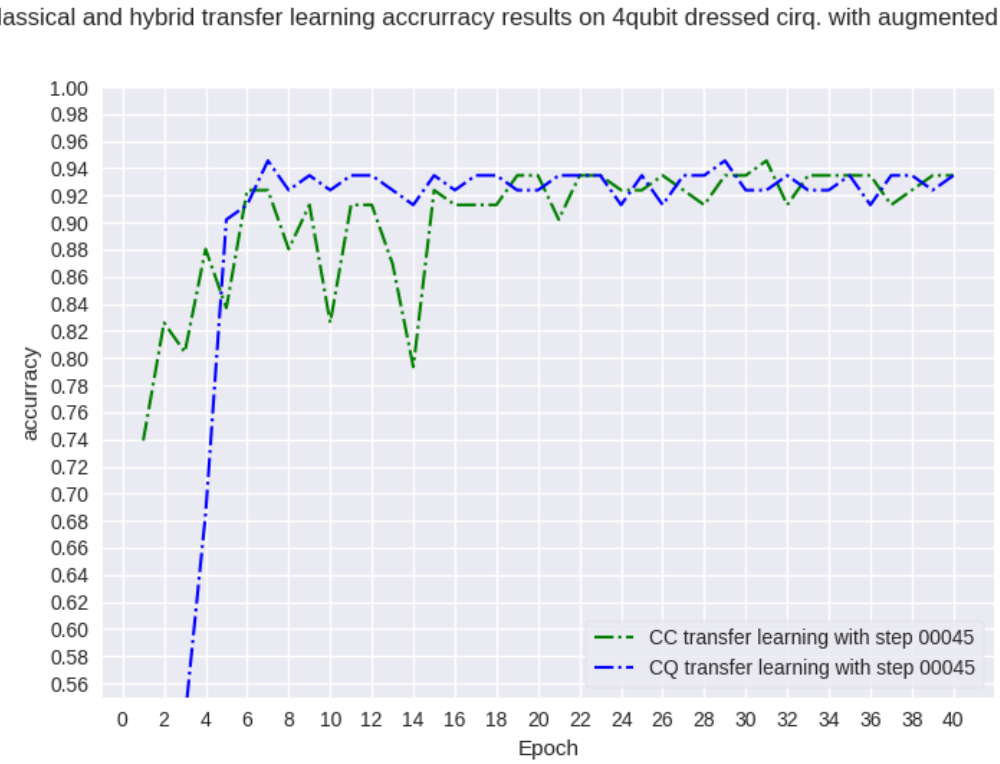


Figure 5.13 Accuracy during training of transfer models with learning step 0.00045 and quantum circuit depth of 5

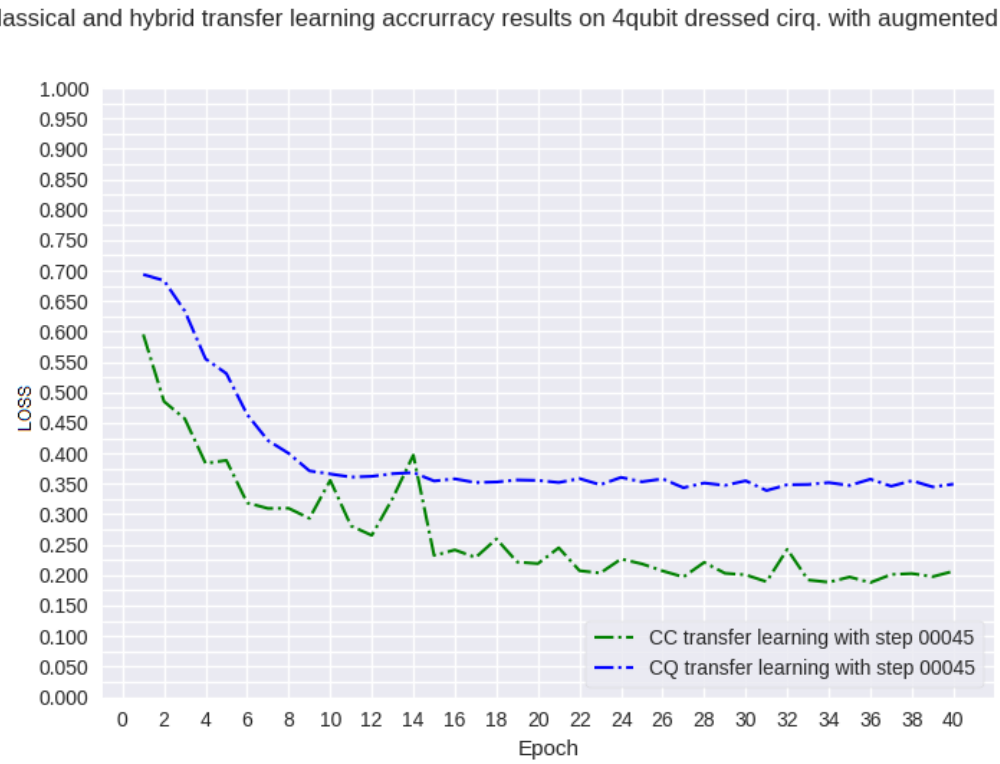


Figure 5.14 Loss during training of transfer models with learning step 0.00045 and quantum circuit depth of 5

5.2.5 QTL networks - conclusions

The presented architecture outlines a method of implementing classical-to-quantum implementation of transfer learning, which could allow to use vast power of pretrained very deep classical networks in conjunction with quantum trainable networks, such as dressed circuit which potentially could bring a boost to transfer learning methods, provided a potential quantum advantage within the field of machine learning is found and the training process of network on particular used dataset would benefit from it.

Figures 5.5, 5.7, 5.10 and 5.13 seem to indicate that the hybrid networks was able to achieve faster favourable accuracy when compared to classical analogue. This could be specific to this or only some datasets and more research is required however this may point in a direction of a potential quantum advantage when using quantum layers in conjunction with classical NN architectures.

Apart from achieving worse loss function results it is also important to point out that the hybrid network seemed to be more sensitive to changes of learning rate and circuit depth - in practice that means that hyperparameter optimization with such a network would become a way more important aspect than in classical network as failing to find optimal hyperparameters seems to impact network accuracy much more than in classical analogue as seen on figures 5.12 and 5.9

Even if such hybrid networks could not be effectively applied to gain an advantage the Andrea's paper also outlines quantum-to-quantum transfer learning[8] which potentially could bring a boost when quantum machine learning is used for inherently quantum data since normal QML methods have already been proven to be superior when the data is quantum[9] - quantum computers work on the same principles as the data and also the data would potentially not need to be encoded. This could be even more amplified if for example pretrained quantum networks could be applied on QC which have less qubits than the number of neurons of such networks by designing a similar method as the patchy quantum generators from Huang's work[29].

5.3 QGAN's

The last explored architecture was a quantum generative adversal network which was tested with 3 classes of 12 pixel by 12 pixel images of letters J, D and I as mentioned in subsection 4.3.3.

5.3.1 Generating letter D

First letter D was explored for it's simple structure, with which the network should not have much trouble. Figure 5.15 presents the resulting generated images after n iterations of the QGAN training circle.

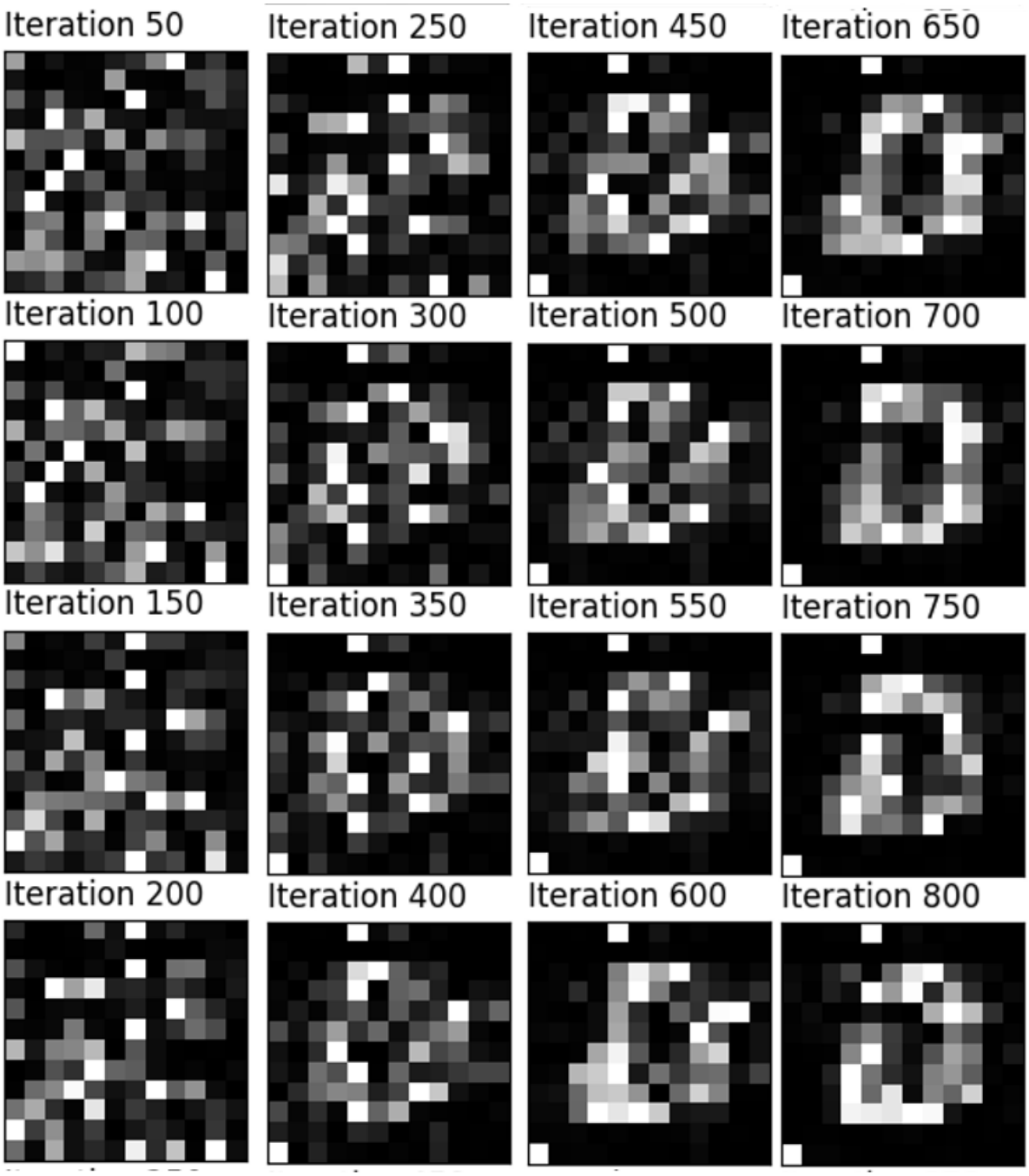


Figure 5.15 results of QGAN taught on letter D

Around iteration six hundredth the network begins to recreate a somewhat good quality image of letter D.

5.3.2 Generating letter J

Letter J, as seen in figure 4.5 has more variation when it comes to handwritten version of it and thus it may be a harder challenge for the network to recreate it correctly. As it can be seen on figure 5.16 this was exactly the case:

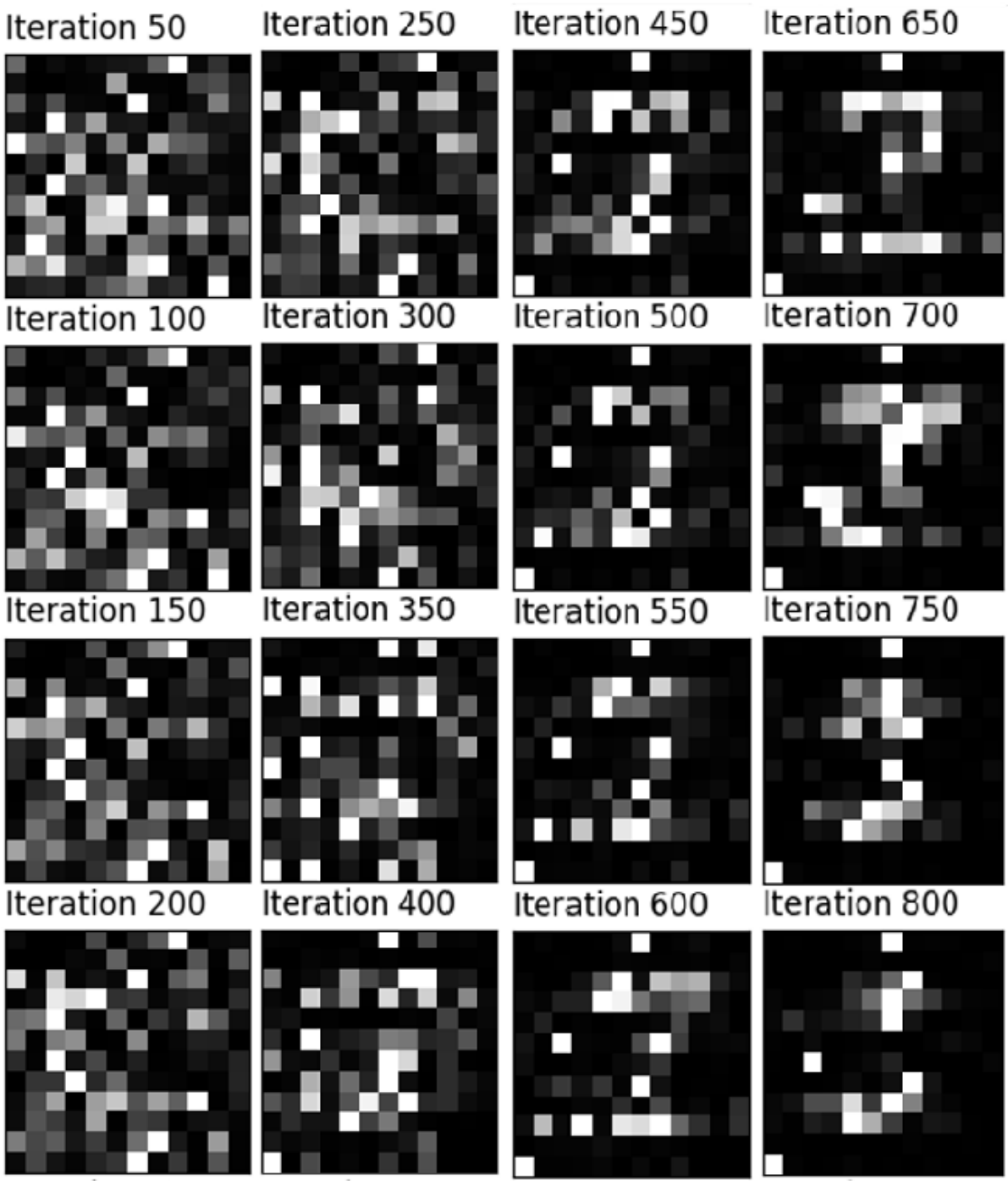


Figure 5.16 results of QGAN taught on letter J

During iteration seven hundredth the letter J is somewhat recognizable, however the images generated later are missing the middle part of the letter - most likely due to it often being drawn at an angle or in different place as seen on figure 4.5 which was reflected in the used dataset.

5.3.3 Generating letter I

The handwritten letter I was a hardest challenge since simply it varies so much between each handwritten image as shown on figure 4.7. After iteration 500 hundredth the network did manage to recreate an image that can be somewhat thought of as the letter I, although a very distorted one.

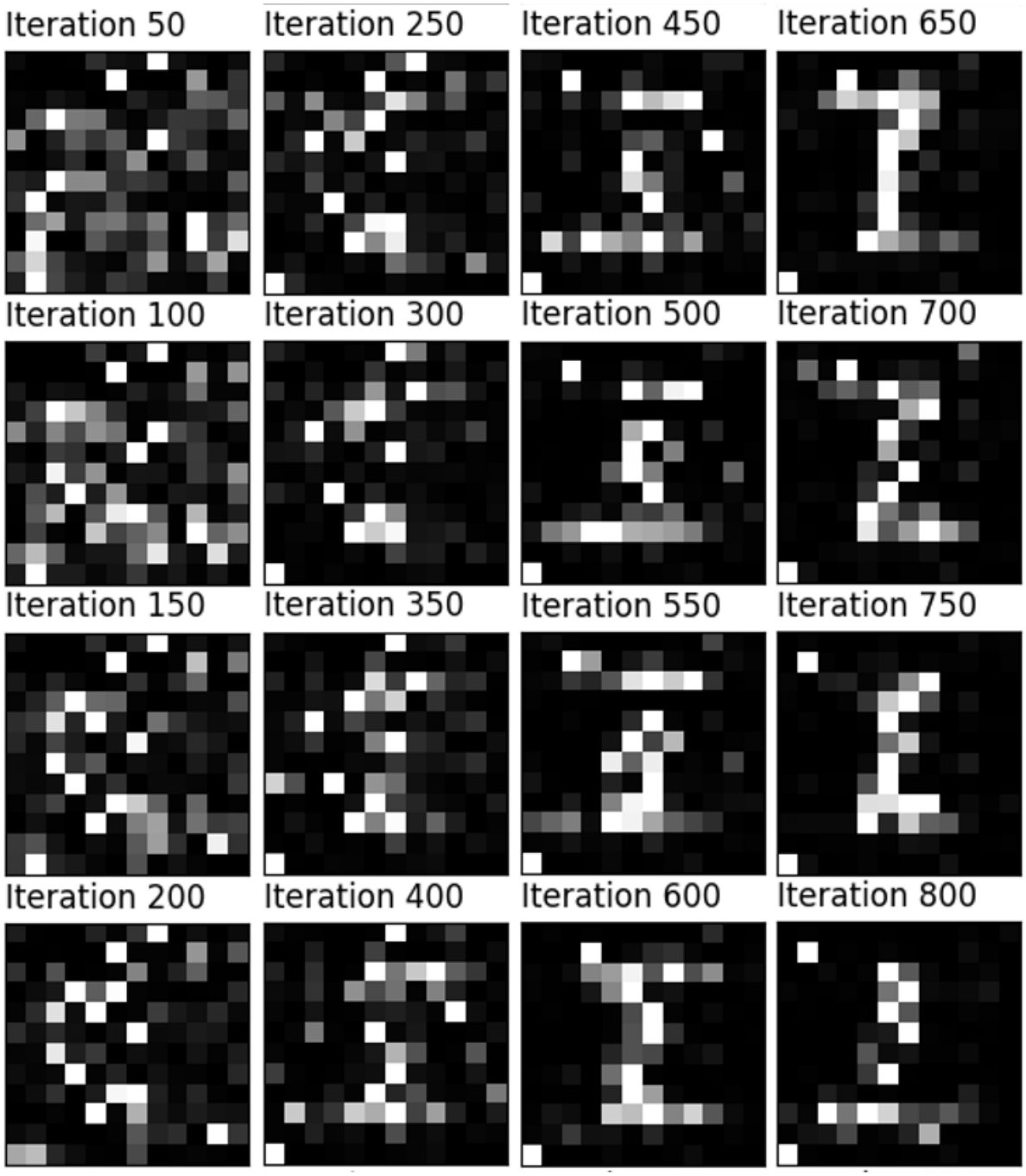


Figure 5.17 results of QGAN taught on letter *I*

The poor result can once again be attributed to lower quality of input data, which simply varies too much between each image - we must remember than in reality the network simply learns the distribution of the input dataset. The handwritten I is simply to unique for many individuals as seen for example on the first image shown on figure 4.7 which looks more like a letter Z than letter I.

5.3.4 QGAN's - conclusions

The network did manage to recreate all letters in some way, however noise was still visible. Interestingly all letters recreated in later stage had a bright pixel in lower left part of the image.

The poor results are mostly attributed to the distribution of the training dataset used for this network. The dataset is constructed from handwritten letters and they are often drawn at an angle or simply with slightly different style which is even reflected in the low resolution of 12 by 12 pixels that they were rescaled to as seen in figures 4.6, 4.7 and 4.5. As mentioned in subsection 4.3.3 the GAN's are basically networks that try to learn the distribution of a particular dataset - hence we may conclude that poor quality images are mostly due to this reason - the distribution was very distorted in the end for some handwritten letters.

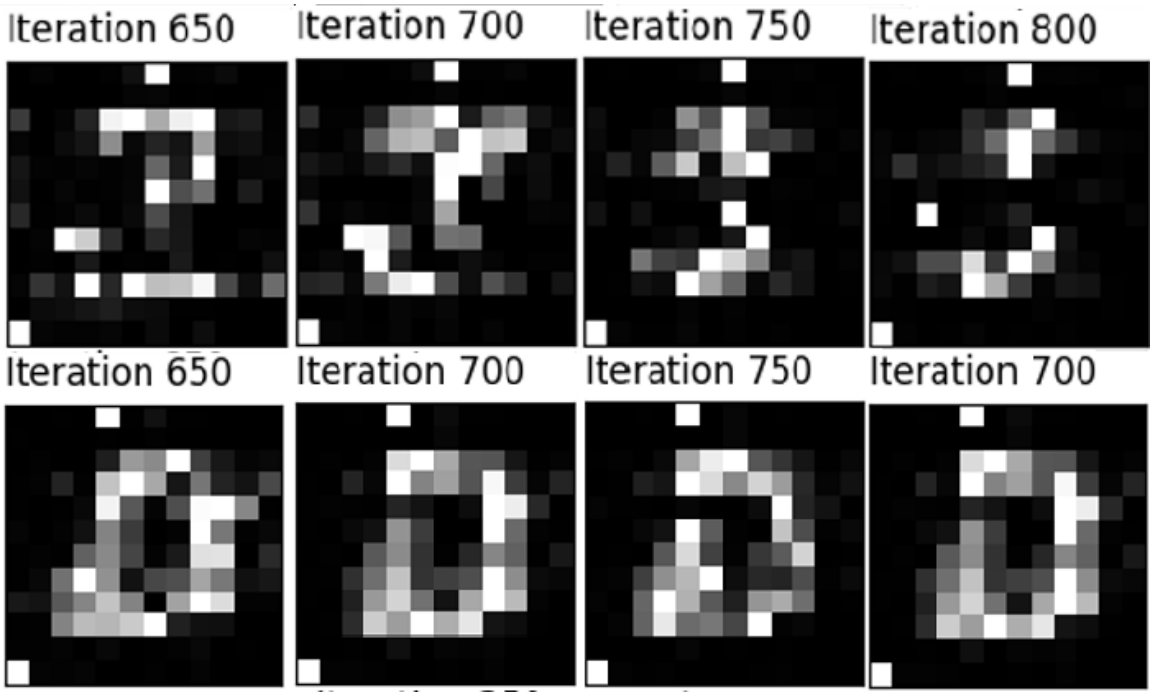


Figure 5.18 Generated letters J and D compared with each other

The 'patchy' architecture allows for implementation of more generators when bigger pictures are submitted - the specific architecture implemented used 9 subgenerators with 4 qubits while the the authors of the method used 4 4-qubit subgenerators for smaller images [16] - this shows that the proposed method may find implementation even when the quantum resources are limited, a situation that is the current status quo of quantum computing.

This example also shows how important it is to choose a well defined dataset if we wish to achieve good quality generated images. The dataset chosen was quite a hard challenge for this network however example of letter D on figure 5.15 shows well what the network can achieve even if the data is not necessarily of very good quality.

Chapter 6

Conclusions

6.1 The case for Hybrid Quantum Classical Neural Networks

There are some potential advantages of using quantum circuits within classical neural networks as shown in previous chapter.

6.1.1 Higher accuracy

Results of QCCNN's analysis show that the network was able to reach higher accuracy than the classical analogue as seen on figure 5.1. While this certainly seems lucrative we can still achieve higher accuracy with classical ML methods by simply adding more layers to the network or using different approaches, such as transfer learning.

While a quantum advantage on classical data still remains to be proven, we can think that perhaps quantum computers within NN architectures would allow us to expose some linear features which are more easily distinguishable when using QC. This very point calls for further investigation but if proven would mean that using such hybrid architectures would allow us to get higher accuracy of models which wouldn't be possible without any classical ML algorithms and thus would potentially offset the costs of quantum hardware.

6.1.2 Faster convergence to minima

The explored architectures show a potential when it comes to achieving faster training results. QTL in particular has shown that it converges to the near minima levels faster when compared to their classical counterpart. This is visible on all accuracy results of QML subsection although most prominent on figure 5.5.

This may be due to - again - quantum part of the network being more inclined to distinguish nonlinear features within the dataset which means that the network overall has an easier time distinguishing the two classes, which in turn results in this phenomena.

6.1.3 Quantum insights

Development of quantum algorithms may improve our knowledge about quantum physics since it allows us to interact with the quantum data directly without first quantitizing it.

We do not know whether large scale quantum computers which can work independently from classical computers (apart from operating the quantum hardware) will exist but if they do, the developed algorithms will be ready for implementation and use. The developed quantum algorithms can also potentially bring insights back to the classical world and improve existing classical algorithms which are basis for developing the quantum algorithms.

6.2 Shortfalls

The experiments performed with QC simulators showed also some potential weaknesses of hybrid architectures.

6.2.1 Instability and Noise

A first notable weakness would be that these networks in general seem to vary more in the results that can be achieved with them from epoch to epoch basis when compared with classical analogue as presented in subchapter 4.3.1 - exploration of QCCNN's which was specifically visible with 1 layer and 3 layer quanvolution layers as seen on figure 5.1. While the general accuracy achieved was higher, it also seemed to be more unstable between consecutive epochs, on epoch 11 and 28 specifically we can see it went down considerably - to the similar levels of accuracy achieved by the rest of architectures.

It is also important to point out that current quantum devices implement a lot of noise and thus we may conclude that this instability would be even more visible on real quantum devices when compared to simulations results showcased in this work.

The similar can be drawn for QTL methods as seen with their sensitivity to changing hyperparameters. Exploration of this architecture has shown that hyperparameter tuning is an even more important aspect in QTL that in classical TL and we may conclude that this importance would be even more amplified if we were to use a real quantum computer for this architecture. While these networks may potentially bring benefits in their current state it would seem like tuning them will also be a harder challenge.

When it comes to QGAN's, they would also be most likely affected by the noise implemented by NISQ era QC, however the authors point out that they could be more resistant to it[29].

6.2.2 Infrastructure

An obvious point is that implementation of such hybrid architecture will first of all require a quantum computer, even if such architecture could be realised by using cloud architectures via connecting to publicly available computer or a one that we bought calculation time on - this still means the algorithms need to be realized in some part differently to allow for such model to learn.

This may be in part mitigated by architectures such as presented QCCNN which allows to firs use QC to process and save the images and then they can be processed as a normal NN, however The very early advancement - and price - of quantum architecture means that for now it still remains a challenge and means that they are potentially not viable to implement if no quantum advantage will be harnessed from them.

6.3 Final words

Quantum computing is an emerging field that could bring us potential advantages in many modern day problems[42][38][12][10] however much remains to be done in this field for Quantum Computing to be a viable approach in solving them. We do not know whether QC will be adapted to a wide scale in the future or will remain only as a small subfield of Information Theory developed only for research purposes or very specific problems, in which it can be easily harnessed for an advantage such as cryptography [10] or factorization of numbers [42]. The thing that remains sure however is that for now it is actively being researched and if not bringing advantages by itself, it can bring us insights about quantum world around us and the classical algorithms that we try to implement into quantum world, which might bring improvements to them as well.

Bibliography

[1] Ibm's qiskit library homepage for quantum development. `https://qiskit.org/`. Accessed: 2022-05-14.

[2] Imagenet homepage. `https://www.image-net.org/`. Accessed: 2022-11-07.

[3] list of qc simulators. `https://quantiki.org/wiki/list-qc-simulators`. Accessed: 2022-09-23.

[4] Mnist dataset website. `http://yann.lecun.com/exdb/mnist/`. Accessed: 2022-05-14.

[5] Pennylane website. `https://pennylane.ai/`. Accessed: 2022-07-11.

[6] Stanford dogs dataset webpage. `http://vision.stanford.edu/aditya86/ImageNetDogs/`. Accessed: 2022-11-09.

[7] S. Aaronson. *Quantum Computing Since Democritus*. Cambridge University Press, 2013.

[8] M. Andrea, B. Thomas, I. Josh, S. Maria, and K. Nathan. Transfer learning in hybrid classical quantum neural networks, 2021.

[9] A. Basheer, A. Afham, and S. K. Goyal. Quantum k-nearest neighbors algorithm. *arXiv preprint arXiv:2003.09187*, 2020.

[10] C. H. Bennett, F. Bessette, G. Brassard, L. Salvail, and J. Smolin. Experimental quantum cryptography. *Journal of cryptology*, 5(1):3–28, 1992.

[11] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

[12] D. Boschi, S. Branca, F. De Martini, L. Hardy, and S. Popescu. Experimental realization of teleporting an unknown pure quantum state via dual classical and einstein-podolsky-rosen channels. *Physical Review Letters*, 80(6):1121, 1998.

[13] S. Bozinovski and A. Fulgosi. The influence of pattern similarity and transfer learning upon training of a base perceptron b2. In *Proceedings of Symposium Informatica*, volume 3, pages 121–126, 1976.

[14] W. Cao, X. Wang, Z. Ming, and J. Gao. A review on neural networks with random weights. *Neurocomputing*, 275:278–287, 2018.

[15] A. Creswell, T. White, V. Dumoulin, K. Arulkumaran, B. Sengupta, and A. A. Bharath. Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1):53–65, 2018.

[16] P.-L. Dallaire-Demers and N. Killoran. Quantum generative adversarial networks. *Physical Review A*, 98(1):012324, 2018.

[17] L. Deng, D. Yu, et al. Deep learning: methods and applications. *Foundations and trends® in signal processing*, 7(3–4):197–387, 2014.

[18] D. Deutsch. Quantum theory, the church–turing principle and the universal quantum computer. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 400(1818):97–117, 1985.

[19] D. Deutsch and R. Jozsa. Rapid solution of problems by quantum computation. *Royal Society*, 1992.

[20] P. A. M. Dirac. A new notation for quantum mechanics. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 35, pages 416–418. Cambridge University Press, 1939.

[21] D. Ezzat, H. A. Ella, et al. Gsa-densenet121-covid-19: a hybrid deep learning architecture for the diagnosis of covid-19 disease based on gravitational search optimization algorithm. *arXiv preprint arXiv:2004.05084*, 2020.

[22] K. Fukushima and S. Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.

[23] R. Giryes, G. Sapiro, and A. M. Bronstein. Deep neural networks with random gaussian weights: A universal classification strategy? *IEEE Transactions on Signal Processing*, 64(13):3444–3457, 2016.

[24] L. K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[25] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[26] M. Henderson, S. Shakya, S. Pradhan, and T. Cook. Quanvolutional neural networks: powering image recognition with quantum circuits. *Quantum Machine Intelligence*, 2(1):1–9, 2020.

[27] J. D. Hidary. *Quantum Computing: An Applied Approach*. Springer, 2019.

[28] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[29] H.-L. Huang, Y. Du, M. Gong, Y. Zhao, Y. Wu, C. Wang, S. Li, F. Liang, J. Lin, Y. Xu, et al. Experimental quantum generative adversarial networks for image generation. *Physical Review Applied*, 16(2):024051, 2021.

[30] D. H. Hubel and T. N. Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1):215–243, 1968.

[31] S. Khan, N. Islam, Z. Jan, I. U. Din, and J. J. C. Rodrigues. A novel deep learning based framework for the detection and classification of breast cancer using transfer learning. *Pattern Recognition Letters*, 125:1–6, 2019.

[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[33] J. Liu, K. H. Lim, K. L. Wood, W. Huang, C. Guo, and H.-L. Huang. Hybrid quantum-classical convolutional neural networks. *Science China Physics, Mechanics & Astronomy*, 64(9):1–8, 2021.

[34] T. M. Mitchell and T. M. Mitchell. *Machine learning*, volume 1. McGraw-hill New York, 1997.

[35] P. J. Nicholas, A. To, O. Tanglay, I. M. Young, M. E. Sughrue, and S. Doyen. Using a resnet-18 network to detect features of alzheimer's disease on functional magnetic resonance imaging: A failed replication. comment on odusami et al. analysis of features of alzheimer's disease: Detection of early stage from functional brain changes in magnetic resonance images using a finetuned resnet18 network. diagnostics 2021, 11, 1071. *Diagnostics*, 12(5):1094, 2022.

[36] K.-S. Oh and K. Jung. Gpu implementation of neural networks. *Pattern Recognition*, 37(6):1311–1314, 2004.

[37] E. Paquet and F. Soleymani. Quantumleap: Hybrid quantum neural network for financial predictions. *Expert Systems with Applications*, 195:116583, 2022.

[38] S. Pirandola, J. Eisert, C. Weedbrook, A. Furusawa, and S. L. Braunstein. Advances in quantum teleportation. *Nature photonics*, 9(10):641–652, 2015.

[39] J. Preskill. Fault-tolerant quantum computers. 1998.

[40] P. Rebentrost, T. R. Bromley, C. Weedbrook, and S. Lloyd. Quantum hopfield neural network. *Physical Review A*, 98(4):042308, 2018.

[41] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[42] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[43] C. Shorten and T. M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[44] K. Srinivasan, S. Satyajit, B. K. Behera, and P. K. Panigrahi. Efficient quantum algorithm for solving travelling salesman problem: An ibm quantum experience. *arXiv preprint arXiv:1805.10928*, 2018.

[45] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu. A survey on deep transfer learning. In *International conference on artificial neural networks*, pages 270–279. Springer, 2018.

[46] L. Torrey and J. Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global, 2010.

[47] Y. S. Weinstein, M. Pravia, E. Fortunato, S. Lloyd, and D. G. Cory. Implementation of the quantum fourier transform. *Physical review letters*, 86(9):1889, 2001.

[48] K. Weiss, T. M. Khoshgoftaar, and D. Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016.

List of Figures

2.1 Example architecture of an ANN 10
2.2 Visualization of working convolution matrix with a simple CNN architecture 13
2.3 Visualization of the transfer learning method 13
2.4 Sketch from original alexnet paper [32] showing the complex architecture of this networks architecture 14
2.5 Example of 8 images taken from imagenet [2] dataset 14
2.6 Chart visualizing the whole architecture of a GAN 15

3.1 sketch of a Bloch sphere representing $|\psi\rangle$ state 19
3.2 Example showing state $|0\rangle$, $|1\rangle$ and $(\frac{1}{\sqrt{2}}|1\rangle + \frac{1}{\sqrt{2}}|0\rangle)$ on Bloch spheres . . . 19
3.3 Qubit before (*qubit 0*) and after (*qubit 1*) applying the X gate. 21
3.4 Qubit before (*qubit 0*) and after (*qubit 1*) applying the Y gate. 21
3.5 Qubit before (*qubit 0*) and after (*qubit 1*) applying the Z gate. 22
3.6 Qubit before (*qubit 0*) and after (*qubit 1*) applying the H gate. 22
3.7 Qubit before (*qubit 0*) and after applying the T gate (*qubit 1*) and S gate (*qubit 2*). 23
3.8 Qubit before (*qubit 0*) and after applying the dagger T gate (*qubit 1*) and dagger S gate (*qubit 2*). 24
3.9 Example of a simple 3 qubit circuit using only 1bit gates (this circuit isn't particularly useful) 24
3.10 more sophisticated 3 qubit circuit, this circuit performs a 3 qubit quantum Fourier transform [47] 25

4.1 All ten classses contained within the MNIST dataset 31
4.2 Preview of some images from the Stanford Dog dataset 32
4.3 Preview of quantum Australian terriers 32
4.4 Preview of quantum Norfolk terriers 33
4.5 Some of the images of letter J contained within the dataset 33
4.6 Some of the images of letter D contained within the dataset 33
4.7 Some of the images of letter I contained within the dataset 33
4.8 A diagram showing of how quantum convolution layer works 34
4.9 Circuit diagram of the quantum circuit sitting at the core of the dressed quantum circuit 35
4.10 Diagram showing in detail the implementation of the dressed quantum circuit used in the transfer learning experimentation 36
4.11 Diagram showing the idea behind the explored 'patchy' architecture of a QGAN 37

5.1 Accuracy on test dataset while training QCCNN model on MNIST dataset 40

5.2 Loss on test dataset of QCCNN models on MNIST dataset 40
5.3 Loss on test dataset of QCCNN models on MNIST dataset 41
5.4 Loss on test dataset of QCCNN models on MNIST dataset 41
5.5 Accuracy during training on not augmented dataset 43
5.6 Loss during training on not augmented dataset 43
5.7 Accuracy during training on augmented dataset 44
5.8 Loss during training on augmented dataset 44
5.9 Influence of depth of core circuit on accuracy 45
5.10 Accuracy during training of transfer models with learning step 0.0004 and quantum circuit depth of 5 46
5.11 Loss during training of transfer models with learning step 0.0004 and quantum circuit depth of 5 46
5.12 Influence of learning rate parameter on network accuracy 47
5.13 Accuracy during training of transfer models with learning step 0.00045 and quantum circuit depth of 5 48
5.14 Loss during training of transfer models with learning step 0.00045 and quantum circuit depth of 5 48
5.15 results of QGAN taught on letter D 50
5.16 results of QGAN taught on letter J 51
5.17 results of QGAN taught on letter I 52
5.18 Generated letters J and D compared with each other 53

List of Tables

3.1 Table showing the difference in basis states with classical binary states represented by truth tables. y' shows a state of the y qubit after applying the gate. the state of the control qubit x does not change at all hence it is not described here 26
3.2 Toffoli gate truth table showing target qubits before and after applying the Toffoli gate - both control qubits states 27