University of Stavanger

Faculty of Science and Technology

MASTER'S THESIS

Study program/ Specialization: Master of Science in Data Science	Spring semester, 2023. Open / Restricted access
Writer: Emil Haavardtun	……Emil Haavardtun…… (Writer's signature)
Faculty supervisor: Ferhat Özgur Catak External supervisor(s): Shaukat Ali	
Thesis title: Novel Data Encodings For Quantum Machine Learning: Enhancing the Quantum Feed-forward Neural Network for Improved Image Recognition	
Credits (ECTS): 30	
Key words: QML, QNN, Image recognition, PQC, QFFNN, CIFAR-10, MNIST, Tetris	Pages: …79…………… + enclosure: …95…… Stavanger, …2023.06.15…. Date/year

University of Stavanger

Faculty of Science and Technology
Department of Electrical Engineering and Computer Science

Novel Data Encodings For Quantum Machine Learning

Enhancing the Quantum Feed-forward Neural Network for Improved Image Recognition

Master's Thesis in Data Science
by

Emil Haavardtun

Internal Supervisors

Ferhat Özgur Catak

External Supervisors

Shaukat Ali

June 15, 2023

Preface

This thesis is the culmination of my research and exploration. It has been submitted in partial fulfilment of a degree in Master of Science in Data Science at the University of Stavanger. The work herein is entirely my own.

[illegible]

Abstract

Quantum machine learning combines the realms of quantum computing with artificial intelligence, providing novel approaches to problem-solving. Quantum image recognition is one such problem that has attracted significant attention. However, many existing algorithms face a common challenge – current quantum hardware has limited available qubits. Based on the quantum machine learning framework anointed quantum feed-forward neural network, this thesis proposes novel data encodings for image recognition. By fully leveraging the capabilities of quantum computing, these new encodings represent more of each pixel's information without requiring additional qubits. Specifically, the encodings are devised to characterise continuous pixel values, three-channel binary pixel values and three-channel continuous pixel values. In particular, the three-channel encodings enable the model to consider colours in images for increased performance. Through extensive evaluation of various datasets, the proposed encodings are demonstrated to enhance the performance of the quantum feed-forward neural network. Furthermore, the model is extended to the multi-class problem, further showcasing the improvement which the encodings provide. The work done herein is exploratory by nature, tested through classical simulations of quantum computers. As many quantum image recognition models utilise many of the same principles as the quantum feed-forward neural network, it is reasonable to imagine that the ideas and insight present in this thesis can be applied to such other models.

Acknowledgements

I would like to express my thanks to my friends and family for the help and support writing this thesis, both through constructive conversations and proofreading. Furthermore, I would like to thank my supervisor Shakut Ali and his colleagues for providing me with the idea and opportunity to work on this cool project.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Machine learning is a booming field in the contemporary world. In 2015, Google used machine learning to beat a professional Go player in an equal match [1], the first time a computer had ever managed such a feat. Since then, the usage of machine learning has only increased. Perhaps the greatest strength of machine learning is its ability to consider vast amounts of data. This, however, also gives rise to some of its biggest challenges, increased complexity and prolonged computation times. Enter the world of quantum machine learning, where machine learning meets the power of quantum computing.

1.1 Introducing quantum computing

Based on quantum mechanics, quantum computing originated in the 1980s with lectures by physicist Richard Feynman [2]. Unlike classical computers that use classical bits, quantum computers employ the principles of quantum mechanics to store information in *qubits* (quantum bits). These qubits allow algorithms to utilise properties of quantum mechanics to solve specific problems in disparate ways compared to classical computers. For instance, such algorithms can theoretically have a lower time complexity than the fastest possible classical algorithm. The first considerable invention in quantum computing was the development of Shor's algorithm. Shor's algorithm is an algorithm for factoring integers into primes, a task which it completes in polynomial time. This firmly puts it in the time complexity class BQP (bound-error quantum polynomial time) where no known classical algorithm has a comparable time complexity [3].

While at first a theoretical field, quantum computing has, over the years, taken a step into physical existence. In 2018, the term noisy intermediate-scale quantum (NISQ) computing was coined for the era that quantum computing now exists in [4]. These NISQ computers can exceed 100 qubits [4, 5], passing the limits for what can be brute forced with modern digital super-computers [4]. With these recent advancements, the possible fields for quantum computing have increased manifold. Some areas are optimisation algorithms, Fourier transforms, partial differential equations, function approximations and search algorithms, with all the applications that follow.

1.2 Quantum computing in machine learning

With such great promises of improvement, it is clear that quantum machine learning (QML) is an exciting field in data science. Of particular interest to this work is the possibility of quantum computing in image recognition. While the outlooks are positive, it is also essential to understand where quantum computing should be employed. Given that classical computers are already exceptional at classical computation, it follows that quantum computing should be utilised in symbiosis with classical computers. For example, a part of the previously mentioned Shor's algorithm is the quantum Fourier transform (QFT), a quantum implementation of the fast Fourier transform (FFT). A classical machine learning model using FFT could then instead query a quantum computer for the results of the faster QFT. The remainder of the training process is left to the classical computer.

1.3 Quantum image recognition

Image recognition is a field showcasing machine learning's remarkable potential to outperform traditional programming approaches. In image recognition, the improvement comes from the algorithm's capability to consider large amounts of data and its ability to learn complex and intricate patterns. The best image recognition models have for many years been convolutional neural networks (CNN) [6], an idea developed in the 1990s [7]. Furthermore, one of the most significant improvements since then comes from increasingly powerful computers, allowing for faster training and increased parameter count. This is where quantum computing enters the picture. Quantum computing provides not only reduced time complexity for certain algorithms but also novel architecture designs that leverage unique properties of quantum mechanics. For example, quantum computing can implement true randomness or new feature maps.

In quantum image recognition, there are various approaches utilising these properties. Some models strives to implement quantum alternatives of components in traditional models, such as quantum kernels from CNNs. Others take a more novel approach. One of the first approaches in this second category was the quantum feed-forward neural network developed in 2018 [8], the model on which this thesis focuses. While there are various advancements in quantum image recognition, there is one factor that heavily limits modern experiments. As earlier stated, modern quantum computers have upwards of a hundred qubits. On a classical computer, a hundred bits would be enough to represent a ten times ten binary image, far from the quality of images used in the digital world. For problems like these, quantum computing presents various solutions, such as dense quantum image encoding.

1.4 Research objectives

The quantum feed-forward neural network (QFFNN) is one of the most exciting current models for image recognition, but little work has been done to extend the model beyond the initial implementation. While effective at simple tasks, the QFFNN model uses a classical-like encoding strategy for pixel values. The more data a model has access to, the higher the potential for making better predictions. This thesis proposes that the QFFNN model will perform better if it has access to more information about lumination. Consequently, the main aim of this work is to devise and implement data encodings which encode more data for the model to use. The base encoding can be defined as a binary encoding of a single-channel image. Given this base encoding as the binary encoding, the three research questions of this thesis can be stated as follows:

- Can the base encoding be extended to greyscale values in a way the QFFNN model understands? And does such an extension provide improved performance?

  The aim is to develop a new encoding for the QFFNN model, which encodes greyscale values in the input qubits. Then verify the performance of such an encoding.

- Can the base encoding be extended to multiple channels in a way the QFFNN model understands? And does such an extension provide improved performance?

  The aim is to develop a new encoding for the QFFNN model, which incorporates all three channels in a colour image. Then verify the performance of such an encoding.

- Given that both the previous encoding methods are possible to devise. Can the two encoding methods be combined into a new encoding which encodes more than two values in multiple channels?

  The aim is to combine the two previous encodings, then verify if this new encoding further increases the performance of such an encoding.

1.5 Contributions

There are various other papers which aim to enhance the QFFNN model for image recognition. All of these have, to the knowledge of this thesis' author, working on improving the network model, not enhancing the initial encoding of the data. Even so, it seems reasonable to assume that others have considered using superpositions to encode more data. For instance [8] showed that superpositions in the input qubits could be used to represent multiple labelled data at the same time. Despite this, no other work which focuses on the input encoding has been found, especially not for colour images like those in the CIFAR-10 dataset.

In this work, three new encodings are developed for the quantum feed-forward neural network model. The various main contributions are as follows:

- A new encoding is devised, implementing greyscale values as free rotations around a single axis to represent more than two values.

- A new encoding is devised, implementing multi-channel pixel values where each channel is encoded as a rotation around a unique axis.

- The two encodings are combined by allowing free rotations for each of the unique axes, representing various channel values for each axis.

- To test the encodings for coloured images, two new datasets are devised, allowing for specific testing.

- All three models are tested on realistic images from the two datasets, MNIST and CIFAR-10.

- To further test the performance of the new encodings, the model is extended to the multi-class classifier.

1.6 Structure

The remaining content of this thesis consists of the following chapters.

- Chapter 2 provides a comprehensive explanation of the fundamental concepts and theories related to quantum machine learning. In this chapter, the fundamental principles of quantum computing, quantum machine learning and image presentation are explained, providing a theoretical foundation for the subsequent chapters. Moreover, the chapter delves into the current state of research in the field, surveying various approaches and developments.

- Chapter 3 begins by explaining the QFFNN model utilised in this thesis. It provides an in-depth explanation of the various components of the model. The chapter then devises the novel data encodings and the approaches to test them, including conceiving strategic new datasets.

- Chapter 4 presents the experimental results. First, the single-axis encoding is tested on the binary-class datasets. Secondly, the multi-axis encodings are tested on the binary-class datasets. The encodings are then tested on the multi-class problem. Finally, various additional results are presented. For each subsection, interesting or unexpected results are discussed.

- Chapter 5 concludes the thesis by summarising the results. It also stakes out areas that can be further explored.

Chapter 2

Background

This chapter gives a thorough explanation of the theory necessary to follow the remainder of the thesis. First, image representation on digital computers is explained. Second, the three public datasets, tetrominoes, MNIST and CIFAR-10, used in this thesis are explained. Followingly, a comprehensive introduction to quantum computing is presented, including general concepts which are used subsequently, as well as an introduction to quantum machine learning. In addition, some relevant topics from classical machine learning are detailed. Throughout the explanation, various related works are explored. Note, in the following sections and the remainder of the work, the word 'classical' refers to distinguishing classical systems from their quantum counterpart, e.g., classical computing versus quantum computing.

2.1 Image representation

Before delving into the specifics of these datasets, some clarity to the different types of image representations is presented. Images can be split into four representations, binary, greyscale, colour and multispectral [9]. This research solely concerns itself with the first three. Mutual for all the representations, images are stored as an array of pixel positions, and each pixel's value corresponds to the pixel's lumination value.

Colour images

In addition to the relative position, a colour image encapsulates information about different colours, typically referred to as channels in digital image representation. A typical standard for storing a colour image is in the RGB space, where the lumination of the three monochrome colours red, green and blue is stored within a three-dimensional array. Typically in digital images, colours are represented by an 8-bit value, allowing for a total of 256 distinct values per colour. The 8-bit encoding balances the need to keep dimensionality low while aiming for accurate representation.

Greyscale images

A greyscale image is a two-dimensional image, a monochrome image. It can be derived from a colour image by reducing the dimensionality. Various methods exist for this conversion, such as employing linear or gamma functions to combine the colour channels. However, it is also possible to discard two of the colours keeping only the final colour as a monochrome image. It is crucial to note that the dimensionality of the image is reduced to a third, and the remaining information is lost. Like colour images, the intensity, or inverse intensity, of grey is often stored as an 8-bit value.

Binary images

Unlike greyscale and colour images, binary images store pixel values in a single bit. For example, representing a foreground (black) pixel as true and a background (white) pixel as false. Characterised by this simple representation, binary images only store spatial information, making them the most elementary form of image representation. A binary image can be derived from a greyscale image, usually by grouping pixel values as above or below some threshold value. This process reduces the dimensionality by $256/2 = 128$. Even so, it is commonly used in many computer vision applications as the reduced dimensionality allows for faster training through faster processing and less memory usage.

2.2 Datasets

In this study, the datasets utilised are collections of labelled images, and the subsequent section elaborates on the various datasets, detailing their origin and properties.

2.2.1 Tetrominoes

Tetrominoes are a subset of the geometric figures known as polyominoes. In particular, it is the subset of one-sided polyominoes composed of four (tetra) squares [10]. The dataset utilised in this thesis comprises images of the previously mentioned figures, depicting both their form and chromatic properties. Consult section 3.5 for the implementation details. Tetrominoes are widely recognised due to their inclusion in the popular video game Tetris [11], wherein players control these pieces. Figure 2.1 shows a game of Tetris, with the various noticeable shapes of coloured tetrominoes in the middle of the board.

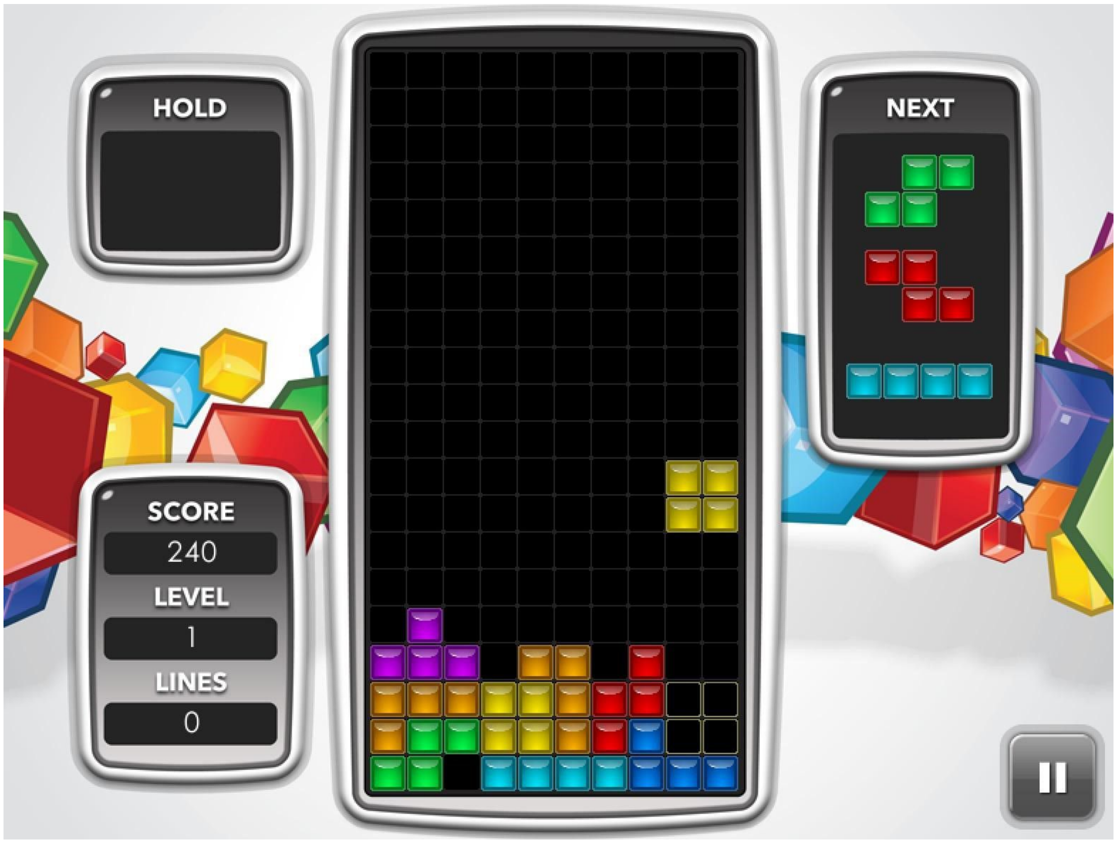


Figure 2.1: A game of Tetris with the noticeable different pieces, distinguishable through shape and colour [1].

This paper is primarily concerned with the application of tetrominoes as a problem for image classification with machine learning. Such classification can be accomplished through various basic algorithms, such as contour detection, which do not require any form of machine learning. In classical machine learning, the main interest in Tetris is, therefore, as a reinforcement learning problem, training an agent to play the game, for an example consult [12]. Furthermore, with the application of classical machine learning techniques, classifying images remains relatively uncomplicated for any reasonable classical model. However, the dataset still poses an intriguing challenge in image classification when employed on quantum computers [13].

2.2.2 MNIST

MNIST (Modified National Institute of Standards and Technology) is a dataset of digitalised images of handwritten digits with the corresponding label. The dataset consists of ten exclusive classes, one for each digit. In total, there are 60 000 training images and 10 000 test images. Each image is 28 by 28 pixels, and the digits have been

[1]Retrieved from [14].

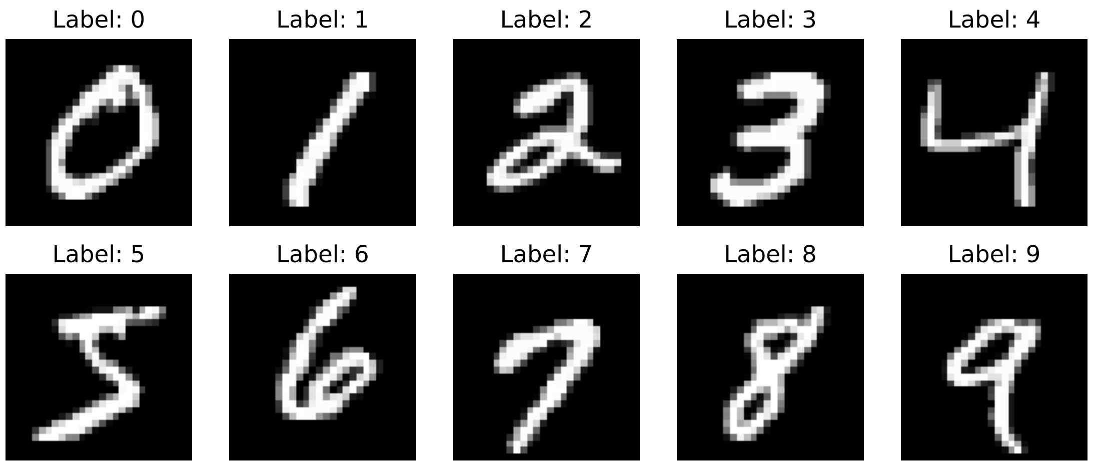


Figure 2.2: Depicting the MNIST dataset with all ten classes and their corresponding label.

centred. The images are greyscale, where each pixel value ranges from 0 (white) to 255 (black). An example of an image from each class can be observed in Figure 2.2.

2.2.3 CIFAR-10

CIFAR-10 (Canadian Institute For Advanced Research) is a popular dataset used as an image classification problem. It consists of 60 000 labelled colour images. The images are of size 32 by 32 pixels and are divided into 10 mutually exclusive classes. The colours are represented as RGB values from 0 to 255, corresponding to their intensity. Figure 2.3 shows a sample of how each class looks.

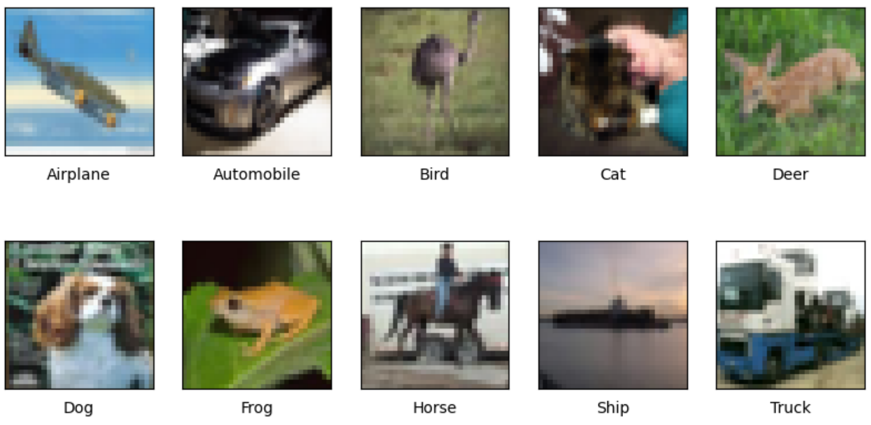


Figure 2.3: Depicting the CIFAR-10 dataset with all ten classes and their corresponding label.

2.3 Quantum computing

Rooted in the principles of quantum mechanics, quantum computing represents an entirely new computing method. A basic introduction to the principles of quantum computing which is necessary to follow the subsequent thesis is presented in this section. Like Boolean algebra is the language of logic in classical computers, linear algebra is the language of quantum computing. It is presumed the reader has some prior knowledge of linear algebra. Nonetheless, the comprehensive theories presented herein remain accessible without it. The following section, and thesis as a whole, focus on the theoretical aspect of quantum computers.

A fundamental characteristic to consider about quantum computing is the relationship between quantum and classical computing. Quantum computing does not operate exclusively to classical computing. Rather, almost all current quantum computing is controlled through a classical computer. A classical computer configures the quantum computer, deciding which logical gates are applied where and so forth. Furthermore, the classical computer treats both the input and the output of the quantum computation. Thus, from the perspective of the classical computer, the quantum computer is just another function – a part of a system. This quantum function takes an input, performs computations within the quantum domain, and produces an output which the classical computer can further treat, for example, by displaying it to the user.

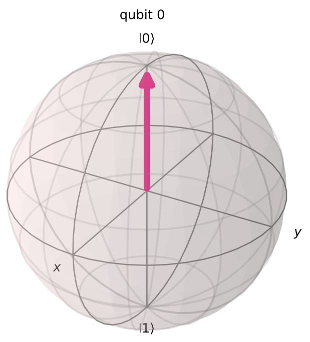
2.3.1 Qubits

Qubits, a portmanteau of ‘quantum’ and ‘bit’, are the building blocks of quantum computers, the basic unit of information. They are similar to classical bits in that they have two distinct states corresponding to the classical 0 and 1.

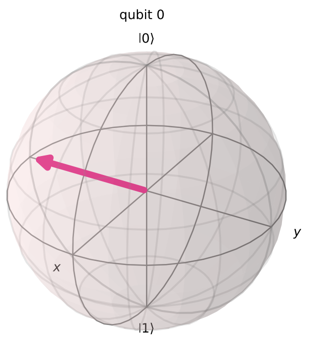
In quantum computing, these values are labelled as $|0\rangle$ and $|1\rangle$. $|\rangle$ is the *Dirac notation* for vectors in a vector space. In particular, quantum mechanics is part of the Hilbert space, a complete inner product space. In quantum computing, it is used to represent state vectors, vectors which give the state of a quantum system. For example, the two vectors explained earlier given in vector notation are

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{and} \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{2.1}$$

The two vectors $|0\rangle$ and $|1\rangle$ constitute the computational basis of quantum computing.



(a) Qubit in the base state $|0\rangle$.



(b) Qubit in a superposition.

Figure 2.4: Two Bloch spheres depicting two qubits in two different states.

However, unlike classical bits, a qubit can also exist in a *superposition.* In a superposition, the qubit exists as a complex linear combination of the two base states, allowing the qubit to store additional information beyond the capabilities of a classical bit.

To visualise this superposition, the Bloch sphere is a useful tool. First presented in [15], the Bloch sphere is a geometric representation of the state of a qubit. Figure 2.4 shows two Bloch spheres. The leftmost sphere depicts a qubit in the base state $|0\rangle$, while the rightmost sphere depicts a qubit in a state halfway between the two base states. Note, the $|0\rangle$ in the Bloch sphere corresponds to a positive direction along the z-axis. By measuring on a different basis, for example, by measuring along the y-axis instead, the second qubit would correspond to one of the base states, while the first qubit would be in a superposition of the base states. Which direction is considered upwards is irrelevant for theoretical quantum computing and depends on the implementation details of the physical quantum computer. To follow the common practice, the z-axis will be considered the standard measurement basis in the following explanations, even though the y-axis will be used in a later chapter, upon which it will be thoroughly noted.

The superposition of a qubit can be written as,

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle\,, \tag{2.2}$$

where $\alpha, \beta \in \mathbb{C}$. Moreover, the coefficients must adhere to the normalisation condition:

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

A typical implementation of a qubit in a physical quantum computer is as an electron or photon, where the superposition is the spin of the particle [16].

As α and β are complex numbers, it is trivial to see that the linear combination in a qubit can take infinitely many possible states. As a qubit can take infinitely many states, it can store infinite information. However, quantum mechanics states that upon being measured, the qubit collapses into one of the two base states. Specifically, upon observing a system in a superposition, only one of the two base states can be measured. The superposition corresponds to the probability of measuring either state. A qubit in a superposition close to $|1\rangle$ is more likely, but not guaranteed, to be observed in this state. Thus, it becomes clear that while a qubit can be encoded to contain any amount of information, only the two base states can be realised through a single measurement. Rather, an approximation of a superposition can be measured by repeatedly preparing a qubit in some state and then observing the qubit. By averaging the results, the original state can be estimated to arbitrary precision, given sufficient repeats.

Similarly, the probabilities associated with the qubit states can be derived from the absolute square of the coefficients. Specifically, the square of each coefficient yields the probability of the qubit being in a particular state.

For example, a qubit in the following state:

$$|\psi\rangle = \frac{1}{\sqrt{2}} \left(|0\rangle + |1\rangle\right), \tag{2.4}$$

will, upon measuring, collapse into $|0\rangle$ or $|1\rangle$ with a probability given as,

$$P(|0\rangle) = \frac{1}{2} \text{ and } P(|1\rangle) = \frac{1}{2}. \tag{2.5}$$

In fact, this state, known as the plus, $|+\rangle$, state, is one of the states where upon being observed, the qubit has an equal (50%) probability of being in either base state.

2.3.2 Entanglement

Despite the intriguing nature of superpositions in a qubit, a single qubit does not offer any significant advantages over a classical bit. Nonetheless, when two or more qubits are introduced to a system, the power of quantum entanglement provides new opportunities. Quantum entanglement is a puzzling but incredible property of quantum mechanics which allows two or more particles to become linked.

The concept of quantum entanglement was first introduced by Einstein-Podolsky-Rosen [17] in a thought experiment, then expanded upon by Schrödinger [18]. The phenomenon occurs in a system of two or more particles in superpositions. The entanglement signifies that when a state in a superposition in one of the particles is observed, information about the other particles can be determined before any measurements are done on those other particles. Alternatively, the state of one particle can not be described without information about the other particles. For example, by measuring the spin property of one particle in an entangled particle pair, the spin of the corresponding particle can also be predicted. The second particle may no longer be in a superposition, even before being observed. Incredibly, this property violates the idea of local realism in physics; the entanglement exists even when the particles are separated by a large distance.

The implications of quantum entanglement are profound, as it is this property which distinguishes quantum computing so greatly from classical computing. Unlike superpositions alone, the use of entanglement allows quantum algorithms to perform computations in ways which are not possible with classical algorithms. For example, the famous Shor's algorithm is a prime-factorisation algorithm which utilises the quantum Fourier Transform algorithm to validate many possible inputs simultaneously. It is important to note that the algorithm does not test all properties simultaneously, but rather, it discovers the global properties of the problem at hand by testing some well-chosen possibilities synchronously. As a result, the algorithm has a polynomial time complexity, while the best-known classical algorithm has an exponential time complexity.

2.3.3 Quantum Logic Gates

Just as qubits are the quantum equivalent of classical bits, quantum logic gates are the logical gates of quantum computers. A sequence of quantum gates is referred to as a quantum circuit. The quantum gates modify the states of the qubits in a quantum computer. Unlike classical logic gates, which operate on binary states, quantum gates can create superpositions in qubits, which results in many new gates. Quantum gates can be divided into two classifications, gates which operate on a single qubit and gates which operate on multiple qubits. Mathematically, quantum gates can be described as multiplying the state vectors of the qubit with a matrix representing the gate. For example, the following is the identity gate:

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.6}$$

Given a qubit in the plus state, multiplying the qubit by the identity gate, it is clear that the state remains:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \end{bmatrix} \tag{2.7}$$

or in Dirac notation:

$$I \left|+\right\rangle = \left|+\right\rangle \tag{2.8}$$

Single qubit gates

Single qubit gates are gates which act on a single qubit. The gates of concern for this thesis are the Pauli gates, the rotation operator gates and the Hadamard gate.

The Pauli gates are the x, y and z gates, which rotate the qubit around the x, y and z axes, respectively.

The X-gate for instance, would rotate a qubit from $|0\rangle$ to $|1\rangle$ and vice versa. The X-gate is given by the matrix:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{2.9}$$

The rotation operator gates, R_X, R_Y and R_Z, are the parameterised versions of the Pauli gates, allowing for arbitrary rotation around their respective axis. The R_X gate is given by the matrix:

$$R_X(\theta) = \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{pmatrix}, \tag{2.10}$$

where θ is the rotation angle. The R_Y and R_Z gates are defined similarly. As the X-gate is a full rotation around the x-axis, it is given by $R_X(\pi)$. An observant reader might notice that this does not give the same matrix as the x-gate defined above. This is due to global phase differences, which are irrelevant in quantum computing.

A very important gate in quantum computing is the Hadamard gate, as it is the simplest gate which creates superposition, converting a qubit from the base state to the plus state. It is given by the matrix:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{2.11}$$

Multi-qubit gates

As the name suggests, multi-qubit gates operate on multiple qubits together. There are several types of multi-qubit gates, but for this thesis, only the Ising coupling gates and the multi-qubit rotation operator gates are of interest. For these particular multi-qubit gates, it is important to distinguish between the controlling qubit(s) and the target qubit(s).

The Ising coupling gates are renowned for creating entanglement between qubits, a crucial aspect of the algorithms. Specifically, the three types of Ising coupling gates that are of concern are the R_{XX}, R_{YY} and R_{ZZ} gates. The Ising coupling gates rotate a qubit just like the single-qubit rotation does. But unlike the single-qubit rotation gates, the Ising coupling gates apply the rotation conditionally based on the value of the controlling qubit. For instance, the gate may only be applied if the controlling qubit is the $|1\rangle$ state, while nothing happens if the qubit is in the $|0\rangle$ state. Thus, given the arbitrary qubits A and B, if $A = |1\rangle$ rotate B by θ radians. Interestingly this creates a quantum entanglement between the qubits. Recall that qubits collapse into a base position upon being measured, thus, the controlling qubit is not in a relative superposition when the gate measures it.

Mathematically, these rotation operators are the tensor product between their respective single-qubit rotations. Like single-qubit rotation operators, these gates are parameterised by a rotation angle θ, which additionally determines the strength of the entanglement effect.

The R_{XX} gate is a parameterised version of the XX-gate, the tensor product of two X-gates. The R_{XX} gate is given by the matrix:

$$R_{XX}(\theta) = \exp\left(-i\frac{\theta}{2}X \otimes X\right), \tag{2.12}$$

where θ is the rotation angle. The matrix for the gate is written as:

$$R_{XX}(\theta) = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & 0 & 0 & -i\sin\left(\frac{\theta}{2}\right) \\ 0 & \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) & 0 \\ 0 & -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) & 0 \\ -i\sin\left(\frac{\theta}{2}\right) & 0 & 0 & \cos\left(\frac{\theta}{2}\right) \end{bmatrix}. \tag{2.13}$$

In addition to the R_{XX}-gate, the R_{YY}-gate and R_{ZZ}-gate is defined in similar fashion.

Quantum circuits

A set of qubits with gates applied to them are called a quantum circuit. A quantum computer is a computer which implements such a circuit in practice. Quantum computers today are categorised as noisy intermediate-scale quantum computers. The 'noisy' in NISQ refers to the fact that the system is unstable leading to unwanted errors [4]. These errors occur both naturally over time and when gates are applied. For example, an R_X gate can rotate the qubit more than intended. On current quantum computers, these errors can range from around 1e$-$3% to above 10%. Furthermore, multi-qubit gates have higher error rates than single-qubit gates [19, 20]. The contemporary NISQ computers which IBM has developed had, at the time of writing, error rates for CNOT-gates of about 1e$-$2% to 1e$-$1%, as shown in Table 2.1. CNOT is a common quantum gate, a quantum version of the classical NOT gate. Note that more complex multi-qubit gates are often built up of simpler gates, such as CNOT-gates, and are thus combinations of errors.

Table 2.1: Quantum computing error rates [2].

Computer Name	Qubits	Measurement Error Rate	CNOT-Error Rate
ibm_cairo	27	1.210e-2	8.665e-3
ibm_lagos	7	1.610e-2	7.618e-3
ibm_washington	127	1.370e-2	1.224e-2
ibmq_lima	5	3.530e-2	1.073e-2

Constant research is being done on how these errors can be mitigated through better physical implementations of quantum computers and through error correction code [21]. Nonetheless, algorithms developed for a modern NISQ computer should keep in mind these errors, aiming to minimise both the qubits in use and the circuit depth.

Furthermore, current quantum computers are severely limited in size. The largest publicly available quantum computer by IBM at the time of writing had 127 qubits [5]. Comparably, the Windows 10 operating system takes up about 20GB of data [22], that is, $1.6e11$ classical bits. This also limits the size of the quantum circuits and, thus, also the algorithms.

2.3.4 Quantum image representation

A key part of this thesis, quantum image representation is a popular field of research, combining the principles of quantum mechanics and digital image processing. As hereinabove explained, limiting quantum circuits is of utmost importance in quantum computing. Therefore, various dense data encodings have been created to represent digital images on quantum computers. Following are some of the basic methods for representation.

- *Qubit Lattice Encoding* [23] - In qubit lattice encoding, the image is represented in a one-to-one pixel-to-qubit encoding, similar to the classical encoding of images. In its simplest form, it can be used to encode binary images where, for instance, $|0\rangle$ corresponds to white while $|1\rangle$ corresponds to black. In a proposed version of qubit lattice encoding, greyscale pixel values can be represented as quantum states of the form:

$$|I\rangle_{i,j} = \cos\frac{\theta_{i,j}}{2}|0\rangle + \sin\frac{\theta_{i,j}}{2}|1\rangle \tag{2.14}$$

  where θ_{ij} is the pixel-value of $pixel_{i,j}$.

[2]Retrieved from [5] a cold day in April of 2023.

- *Flexible Representation of Quantum Images* [24] (*FRQI*) - In the FRQI encoding, both the pixel-value and pixel-position are encoded in the amplitude of the qubits. As FRQI uses separate qubits to calculate the pixel value, a colour image can be stored by using three qubits for pixel values [25]. The image representation can be described as:

$$|I(\theta)\rangle = \frac{1}{2^n} \sum_{i=0}^{2^{2n}-1} (\cos\theta_i |0\rangle + \sin\theta_i |1\rangle) \otimes |i\rangle \,. \tag{2.15}$$

- *Novel Enhanced Quantum Representation* (*NEQR*) - NEQR is an alternative version of FRQI aiming to improve shortcomings at the cost of more qubits. The main difference lies in pixel-values being encoded in a register of qubits using only the base states. The expression for NEQR representation is given as:

$$|I\rangle = \frac{1}{2^n} \sum_{i=0}^{2^{2n}-1} \sum j = 0^{2^{2n}-1} |f(i,j)\rangle |i,j\rangle \,, \tag{2.16}$$

  where $f(i,j)$ is the pixel-value at position (i,j) [26].

While especially the last two encodings are popular, there exists numerous other encodings, such as NAQSS, GNEQR and FRQCI [27].

2.4 Quantum machine learning

In recent years the combination of quantum computing and machine learning has created the field of quantum machine learning (QML). By combining classical data, quantum data, classical computers and quantum computers, [28], using the notation, $L_{goal}^{context}$, splits QML into the four following subcategories:

- L_c^c - Classical data on classical machines. Machine learning in the classical sense. An example is linear regression.

- L_c^q - Quantum data on classical machines. Applying machine learning to quantum phenomena. An example is using classical machine learning to predict quantum states.

- L_q^c - Classical data on quantum machines. The category of interest for this thesis. An example is predicting images using quantum phenomena.

- L_q^q - Quantum data on quantum machines. Similar to L_c^q, but using a quantum computer instead. An example is using quantum machines to predict quantum states.

There are many various algorithms in the QML realm, such as; quantum k-means [29], quantum support vector machines [30] and recurrent quantum neural network [31].

Furthermore, there exist hybrid algorithms which combine classical algorithms with quantum algorithms. An example of such is the QCCNN model, refer to section 2.5, where a normal CNN network uses one or more quantum filters [13].

2.4.1 Parameterised quantum circuits

One major cornerstone in QML is the parameterised quantum circuit (PQC), also known as variational quantum circuits. PQCs are a type of variational quantum algorithm which serves as an adaptable approach for computing objective functions within the framework of supervised learning. A variational algorithm is an algorithm where the parameters are adjusted to solve problems. By changing the parameters of the PQCs, the algorithms can estimate an objective function which hopefully fits the data.

The fundamental usage of PQCs is in hybrid neural networks. A quantum computer implements some objective function using the PQC. A classical computer then repeatedly calls the quantum circuit for the output value of the objective function, adjusting the parameters of the circuit until the output estimates the desired target. If the objective function is classically difficult to simulate, the algorithm provides an exponential speedup in time. An example of a variational algorithm is the quantum approximation optimisation algorithm for solving combinatorial problems [32].

There are many reasons why PQCs are an advantageous tool for NISQ computers. One great advantage comes from the model's resistance to systematic error and noise. The model can adapt to the errors through the parameters. For example, the objective function is correct when an R_{XX}-gate rotates the target qubit π radians, that is, the correct parameter value is π. Consider, however, that the physical implementation of the gate overrotates by some unknown x radians with a probability p. During training, the model could then find that the parameter, and thus the applied rotation, $\pi - p \cdot x$ gives a better answer on average. Comparably, an algorithm which bases its result on a hard-coded rotation π radians would get the wrong result. Another precedence in variational algorithms is that they are easily adapted into a classical system, splitting the workload where it is sensible. Furthermore, the circuit depth is relatively short, often applying just a few layers of gates per qubit while still corresponding to a vast vector space which can represent relatively many values.

2.5 Image classification in QML

As images are classical data, it clearly puts themselves in the L_c^c and L_q^c categories. In the L_q^c category, there exist various models for image classification. Below, some of the various implementations are detailed.

One notable model is the quantum convolutional neural network (QCNN) model [33]. The QCNN model is, as the name suggests, a quantum version of the classical CNN networks. As CNNs consist of convolutional layers and pooling layers, the QCNN model implements quantum versions of these features as smaller quantum circuits through PQCs. Each quantum convolutional filter is implemented as a separate PQC to keep the parameters separate, weights in the classical sense, quickly increasing the numbers of PQCs and, thus, qubits required as the model adds quantum kernels. For an example and a tutorial on a simple QCNN, refer to [34]. [35] applied the QCNN model to the MNIST dataset, achieving comparable results to classical models, showing that the quantum components perform equally but not better than their classical counterparts.

Another model is the so-called hybrid quantum-classic convolutional neural network (QCCNN) [13]. It takes a similar approach to the QCNN model, implementing quantum kernels in a hybrid network. Like the QCNN model, the quantum filters are implemented on PQCs. Yet, the QCCNN model does not define quantum pooling layers. In the article, the model is applied to a greyscale version of the tetrominoes dataset, successfully distinguishing the classes.

Quantum feed-forward neural network (QFFNN) [8] is the model used in this thesis. The model is a multi-purpose quantum machine learning model, but in their work [8] showed that the model could be used for image recognition, employing it on the MNIST dataset. Note that in [8], the model is referred to simply as a quantum neural network (QNN), but to avoid confusion with other models, the name QFFNN is used throughout this thesis, following practice from [36]. The term feed-forward refers to the data traversing the network unidirectionally. QFFNN was the first fully quantum feed-forward model [36]. Like the QCNN and the QCCNN model, the QFFNN model uses a PQC as the main component of the network. Nonetheless, unlike the others, QFFNN does not follow convolutional neural networks. Instead, the complete data is examined by a single PQC. As NISQ computers are limited in qubits, the size of the input data is also limited.

Various research has been done to improve or test the QFFNN model. The work in [37] tested the model on a dataset of medical CT-scan images. The article [38], researched how the circuit could be made hardware efficient, that is, reducing the circuit depth. Furthermore, the articles [39] and [40] focused on taking from deep neural networks. The first, and perhaps most interesting approach, splitting the QFFNN model into an encoder, a transformer and an output layer by using three different circuits. The latter article employs the QFFNN as a final layer in a classical deep neural network.

An approach which does not involve neural networks was carried out in [27]. The authors implemented a quantum version of the Sobel operator for edge detection. The approach was tested on both the MNIST dataset and the CIFAR-10 dataset achieving some success. The authors also investigated the results of noise on their model, which drastically reduced the classification accuracy.

2.6 Gradient descent optimisers

A common part of all the networks employed in this thesis is the optimisers they utilise as the learning algorithm. The algorithm is responsible for adjusting the learning parameters. One common approach to optimisers is based on gradient descent. Gradient descent is an algorithm for finding some local minimum in a function. In the case of learning the function of interest is typically the loss function, hence, finding a local minimum corresponds to finding a value where the loss is low. This approach of finding a minimum is done by going in descending direction of the gradient. As suggested above, the algorithm may become 'stuck' in a local minimum of the loss function, without finding the global minimum. Throughout this thesis two particular gradient descent algorithms are named, the stochastic gradient descent algorithm and the Adam optimiser.

The stochastic gradient descent algorithm is an algorithm where only a randomly selected part of the gradient is evaluated each time. In neural networks this is often done through smaller batch sizes, optimising based only on a portion of the training data at once. The result is typically quicker convergence than pure gradient descent [41].

The Adam optimiser is also a gradient descent algorithm, but unlike the stochastic algorithm, Adam updates each parameter independently. The result has positive and negative consequences, but in particular, it typically yields an even quicker convergence rate than the stochastic gradient descent [42].

Chapter 3

Methods

The following section presents the methodology employed to tackle the research objectives. First, it gives a thorough explanation of the QFFNN model. Based on this explanation, novel data encodings are explicated. Insight is then given to the various considerations that are made to the preprocessing of the images. Founded on the data encodings in combination with the preprocessing, two new datasets are introduced. The datasets aim to allow for the new encodings to be verified.

3.1 Quantum Feed-Forward Neural Network

The model which this thesis aims to improve is the quantum feed-forward neural network (QFFNN) model, proposed in [8]. The overall architecture for the QFFNN model is shown in Figure 3.1. First, the model takes an input image of n by m pixels. In the figure, an example image of size 4 by 4 is depicted. Each of the pixel values is encoded in respective qubits. The model thus requires $n \cdot m$ qubits to encode a n by m image. A parameterised quantum circuit is applied to the input qubits. The PQC changes the value of some readout qubit based on the parameters. The readout qubit is measured, and a prediction is produced. A four by four image would therefore require $4 \cdot 4 + 1 = 17$ qubits in total.

In addition to the predictive components, the model requires a way of learning. The model learns by adjusting the parameters of the PQC until the output matches the targets. Deciding how the parameters should be changed is the task of an optimiser algorithm. [8] chose a gradient descent optimiser. To use an optimiser algorithm there

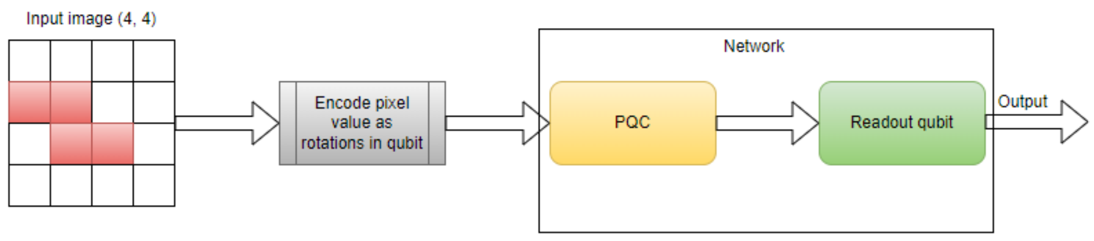


Figure 3.1: Architecture diagram of the QFFNN model. Takes an input image and returns a prediction.

must also be a loss function, evaluating the outcome relative to the target. Optimisers and loss functions are well-known from classical machine learning. In fact, they are implemented classically. Nevertheless, careful consideration must be given when deciding upon the specifics of the algorithms, as elaborated in the sections below.

3.1.1 The input encoding

To encode the incoming images, [8] uses a straightforward approach. Effectively they implement the Qubit Lattice encoding with a one-to-one pixel-to-qubit relation. Furthermore, the pixel values are encoded in the base states of the qubit. The foreground colour is encoded as $|1\rangle$ while the background colour is encoded as $|0\rangle$. A pixel is defined as a foreground colour if it is above some threshold value t within the range of values. As the qubits are in $|0\rangle$ by default, a simple procedure where the qubit is rotated to $|1\rangle$, if it is above the threshold value, is as described in the following algorithm.

Algorithm 3.1 Fixed rotation encoding

```
procedure FIXED_ROTATION_ENCODING(qubit, $p$)
    if $p > t$ then
        $qubit \leftarrow X$
    end if
end procedure
```

3.1.2 Applying the parameterised quantum circuit

A parameterised quantum circuit consists of parameterised gates. In the QFFNN model, two-qubit gates are used. Recall that a two-qubit gate acts upon a target qubit, based on a controlling qubit. For the QFFNN model, these gates are applied between their respective qubit, representing a pixel value and the readout qubit. The input qubit with some pixel value is the controlling qubit, while the readout qubit corresponds to the target qubit. Originally introduced by [43], a parameterised gate is defined as

$$G(z, v) = \begin{pmatrix} z & v \\ -v^* & z^* \end{pmatrix}, \tag{3.1}$$

where $z, v \in \mathbb{C}$, * is the complex conjugate and $|z|^2+|v|^2 = 1$. By changing the parameters the effect of the gate is changed.

The PQC is made up of sequences of these parameterised gates, also known as unitaries. The model allows the composition of gates to be changed and the various possible sequences make up what [8] refers to as a 'toolbox of unitaries'. One such set of gates, or a unitary, can in classical terms be considered a feature map. Given L layers of gates, a sequence of gates is then given by,

$$U(\vec{\theta}) = U_L(\theta_L)U_{L-1}(\theta_{L-1})\dots U_1(\theta_1), \tag{3.2}$$

where each U_i is a parameterised gate. In classical neural network terms, one can think of the parameters as neuron weights. Likewise, each layer of gates corresponds to a layer of neurons. In particular, [8] decided to use the Ising coupling gates, turn to section 2.3.3. A figure depicting some arbitrary sequence of gates is shown in Figure 3.1. In the figure, the state of the input qubits are the lines coming out of the numbers on the left. The boxes are the gates, and each column represents a layer. Note that in the figure there are unitaries operating between the input qubits and also with the readout qubit as the controlling qubit. [8] eventually settled on acting only upon the readout qubit. Nevertheless, these gates can be changed should the developer desire different interactions. In fact, the gates could prepare every unitary in the Hilbert space [36], with enough qubits and circuit depth. Regrettably, this conflicts with NISQ computers. Therefore, selecting a suitable combination of gates which is effective at considering the input data is a challenging problem for algorithms employing PQCs. Following [44], the two Ising coupling gates R_{XX} and R_{ZZ} were selected as they were confirmed to be effective. In correspondence with the aims of the thesis, which focuses on adjusting the input encoding, the gates were not varied. Furthermore, the readout qubit was kept as the only target qubit.

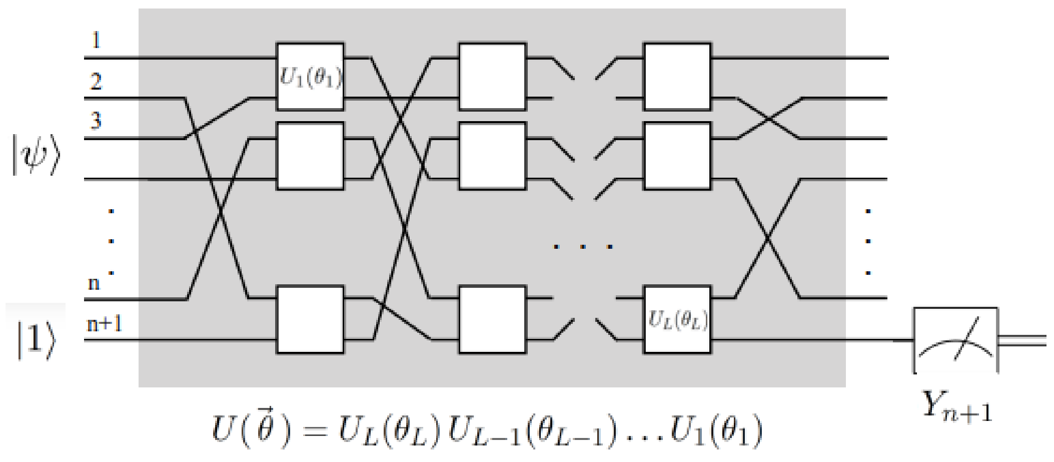

1
2
3
$|\psi\rangle$
n
n+1
$|1\rangle$
$U_1(\theta_1)$
$U_L(\theta_L)$
Y_{n+1}
$U(\vec{\theta}) = U_L(\theta_L)\,U_{L-1}(\theta_{L-1})\dots U_1(\theta_1)$

Figure 3.2: The PQC component. The input qubits 1 to n in some state $|\psi\rangle$ and a readout qubit. Each layer applies parameterised gates $U(\theta)$ between the qubits. The final state is measured in Y_{n+1} [1].

3.1.3 Making a prediction

After the parameterised gates in the QFFNN model have transformed the state of the readout qubit based on the input, it is measured using the Pauli operator σ_y. The Pauli operators, corresponding to the Pauli gates, observe a qubit on the related basis. In particular, the σ_y operator measures in the y-basis. After being transformed by the PQC the readout qubit is likely to be in a superposition, but upon being observed the qubit collapses and σ_y maps the output, denoted Y_{n+1}, to -1 or 1. The output is the prediction of the QFFNN model. To change the prediction, the model changes the parameters of the PQC, which results in a different state for the readout qubit and, thus, a different prediction. By adjusting the parameters, the model aims for the prediction and the target label to correspond.

To give an example, consider a single input qubit, x, which takes one of the two quantum states $|0\rangle$ and $|1\rangle$. The target labels are -1 and 1, respectively. Additionally, there is one output qubit in the $|0\rangle$ state. The PQC consists of one $R_{XX}(\theta)$-gate. To start, set $\theta = 0$, which leaves the target qubit untouched, independent of the controlling qubit. Observing the readout gives $x = |0\rangle\,, y = -1 \vee x = |1\rangle\,, y = 1 \rightarrow -1$. The prediction is correct half the time. The strategy is to change the value of θ such that the readout qubit is measured as 1 when the controlling qubit makes the gate act. The controlling qubit invokes the gate when it is in the state $|1\rangle$, thus, the target qubit can be rotated to the state $|1\rangle$ to match the target output. For this simple example, a trivial solution is $\theta = \pi$, rotating the target qubit a half rotation around the x-axis. Setting $\theta = pi$ in the R_{XX} gate is a special case known as the 'bit flip' XX-gate.

[1]Image taken from [36].

3.1.4 Confidence in predictions and the loss function

By adjusting the parameters, the model aims for the prediction Y_{n+1} and the target label z to correspond. The loss function sets a value for the deviation of the prediction to the target. As detailed subsection 3.1.3, the QFFNN model provides outputs as -1 or 1, which is impractical for a loss function. Either the prediction is correct, or it is unconditionally wrong. Therefore, the model would like an estimate of the confidence in the prediction. [8] solves this by measuring the readout qubit multiple times and averaging the results. The new result y, with reservation to variation, directly corresponds to the probabilities of measuring the superposition of the qubit. The value of y is in the range $[-1, 1]$. Defining the unitary acting upon the input state as

$$U(\vec{\theta})\,|z, 1\rangle\,, \tag{3.3}$$

the average prediction y can then be written as,

$$y = \langle z, 1|\, U^{\dagger}(\vec{\theta}) Y_{n+1} U(\vec{\theta})\, |z, 1\rangle\,. \tag{3.4}$$

With a method of estimating the confidence of the model, [8] defines the sample loss function as,

$$\text{loss}(\vec{\theta}, z) = 1 - z \cdot y. \tag{3.5}$$

A graph of the two possibilities, $z = -1$ and $z = 1$ is shown in Figure 3.3.

One useful feature of the sample loss function is that it gives loss values for predictions that are correct, but not confident. This motivates the network to continue to update the parameters. As quantum computers are highly exposed to noise, confidence in the classifications can be very important. The loss function defined above is a simplified version of the hinge loss function, which is commonly used in machine learning. Hinge loss can be defined as

$$\text{loss}(\vec{\theta}, z) = \max(0, 1 - z \cdot y), \tag{3.6}$$

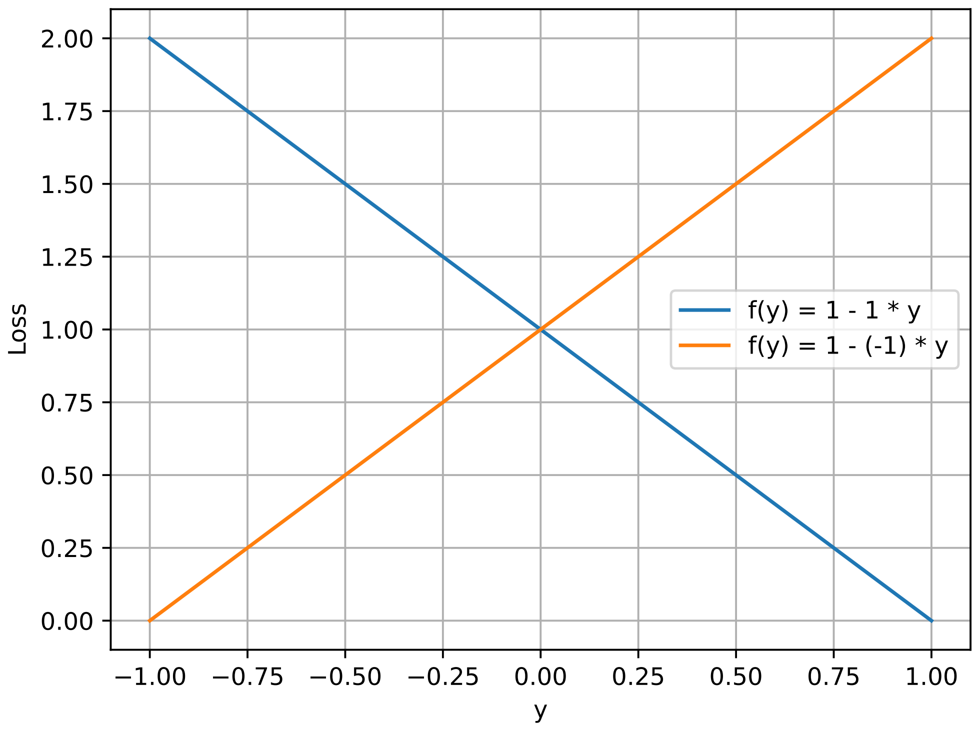


Figure 3.3: The sample loss function for $z = -1$ and $z = 1$.

where the max function selects the highest of the two input values. Consequently, the max function ensures that the loss is never negative. Nonetheless, looking at Figure 3.3 or evaluating the original loss function, it is clear that since the output is in the range $[-1, 1]$ the sample loss function can never be negative. Thus, the maximum function has no effect and the two loss functions are equal within the range. This thesis implements the hinge loss function as the loss function, as it has good support in machine learning libraries.

During the experimental phase, it was discovered that the squared hinge loss function, defined as,

$$loss(\vec{\theta}, z) = \max(0, 1 - z \cdot y)^2, \tag{3.7}$$

provided better results than the original hinge loss function. As this was discovered late in the project, the linear hinge loss function was the one used, except for the results verifying the squared hinge loss function, refer to subsection 4.5.3. Figure 3.4 showcases

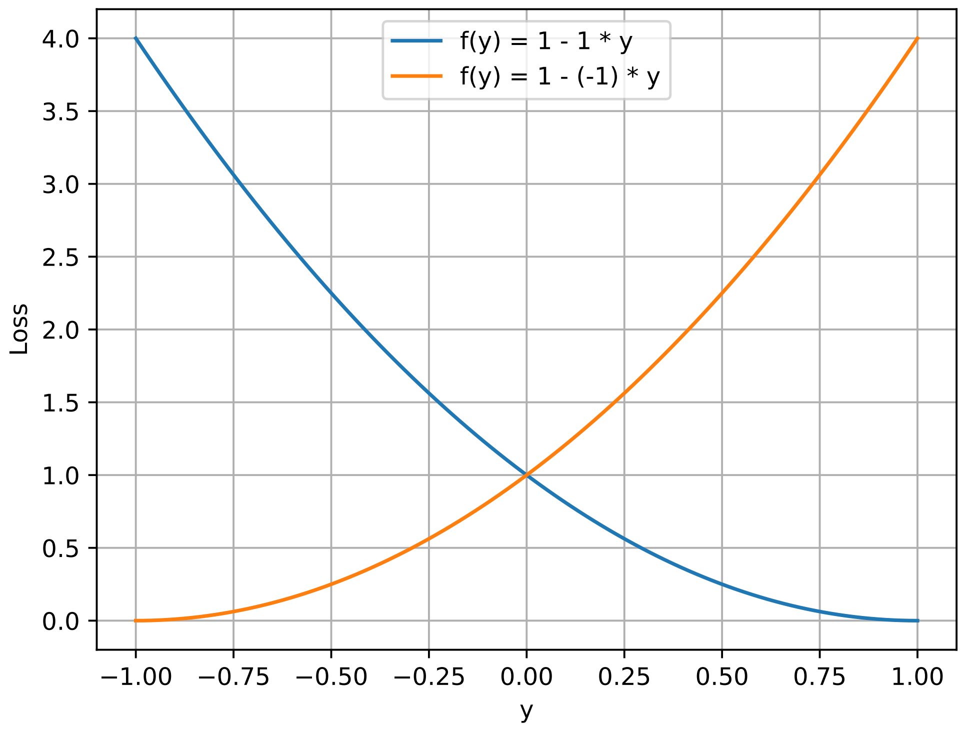


Figure 3.4: The squared hinge loss function for $z = -1$ and $z = 1$.

the squared hinge loss function. Noticeably it does not have a linear rate of change. Compared to the linear hinge loss function, it evaluates to a higher loss for wrong predictions and a lower loss for correct predictions.

3.1.5 Training the network

The missing piece to a complete neural network is the most important part, the learning algorithm. In the QFFNN model, the learning algorithm is responsible for changing and adapting the parameters of the PQC. The algorithm decides the rotation θ to which each gate applies. For the QFFNN model, [8] describes the stochastic gradient descent algorithm. They note that for the QFFNN model the gradient can not 'blow up' as the gradient is bounded by the number of parameters, and the model employs only one or two parameters per qubit, resulting in few total parameters. Exploding gradients is a well-known problem in deep learning where the gradient becomes too large for the network to achieve meaningful learning. Only in 2015, with the introduction of residual networks, a version of convolutional neural networks was a solution to the exploding gradient problem introduced [45]. While the stochastic gradient descent is a well-established algorithm in deep learning, the article [44], which showcases the QFFNN model, uses the Adam optimiser. It is assumed that the reasoning behind the change is grounded in sound logic or empirical evidence, and this thesis sees no reason to deviate from this choice, especially as the quick convergence in the Adam optimiser is likely to result in less training time, an important factor for simulation-based testing.

3.2 Data encoding

The QFFNN model was developed for the MNIST dataset, which consists of relatively small images. While the model can theoretically be scaled to fit any image size, the limits of NISQ computers severely restrict the maximum image size. The subsequent subsections propose novel data encodings, aiming to utilise more information from the input data without increasing the number of qubits encoding the image.

The original implementation of the QFFNN model employs a binary encoding strategy. Each pixel value is encoded through half rotation around the x-axis. Qubits representing foreground pixels are set to the $|1\rangle$ state, while background pixels are kept in the $|0\rangle$ state. In the case of the MNIST dataset, the pixel values range from 0 to 255. Thus, the binary encoding reduces the dimensionality by a factor of 128. For colour images, such as those in the CIFAR-10 dataset, the dimensionality is first reduced by the conversion from the RGB space to the greyscale space.

Section 2.3.4 details various encoding techniques which already exist for quantum image representation. Except for the Qubit Lattice encoding, which the original encoding for the QFFNN model is a variation of, the other techniques are focused on denser encodings that use fewer qubits than pixels. Howbeit, the PQC model applies parameterised gates between the various qubits. Therefore, it is not clear how the PQC model can implement these other techniques while keeping the functionality. Hence, this thesis focuses on enhancing the original encoding.

In the following sections, two new encodings are developed. One considers the full range of monochrome values, and the other utilises the information in colours. In addition, a section is detailed to combine the two encodings.

3.2.1 Free rotation encoding.

The first modification comes through introducing free rotations. The original encoding is the binary mapping

$$x \leq t \mapsto |0\rangle \, , x > t \mapsto |1\rangle \, , \tag{3.8}$$

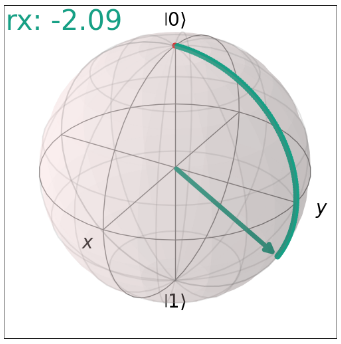


Figure 3.5: Free rotation around the x-axis, rotated $\frac{-2\pi}{3}$ radians.

where x is the pixel value and t is some threshold value. The new encoding aims instead to utilise the continuous positions along one axis between the two base states. This can be achieved by mapping the pixel value to a rotation θ between 0 and π radians, which is then applied to the qubit. Refer to Figure 3.5 for an example rotation.

Applying this rotation, the qubit is now in a superposition where the probabilities correspond to the pixel value. High pixel values are encoded in superpositions more likely to be measured as $|1\rangle$ while low pixel values are encoded closer to $|0\rangle$. Importantly, while the data points are encoded as more specific values than previously, the superpositions imply that the values are not directly measurable. Therefore, it is the task of the PQC, and the feature map which it employs, to understand and characterise the encoding in an improved manner.

Practically, the pixel value is normalised to a value in the range $[0, 1]$, then multiplied by π to find the angle θ. The normalisation maps the minimum possible value to zero and the maximum possible value to one while the remaining values are scaled linearly between them.

The rotation is then encoded into the target qubit using an $R_X(\theta)$-gate where θ is the angle. While the RX-gate allows for arbitrary precision in theory, the error and accuracy of the quantum computer put physical limitations on the precision. The procedure is shown in the following algorithm.

Algorithm 3.2 Free single-axis rotation encoding

```
procedure FREE_ROTATION_ENCODING(qubit, $p$)
    $p \leftarrow$ normalise($p$)
    $qubit \leftarrow RX(p \cdot \pi)$,
end procedure
```

The approach of encoding pixel values as free rotations is similar to the superposition encoding presented in the Qubit Lattice encoding, consult subsection 2.3.4. Even so, there is one notable difference. The Qubit Lattice encoding suggests encoding the pixel value as any superposition. This approach could have been implemented for this work, but it was decided against for a few reasons. Firstly, it is not known if the PQC model understands the superpositions in the qubits. Limiting the encoding to a single axis makes verification easier. Furthermore, a later aim of the thesis is to implement colour encoding in the same qubit. By limiting the superposition that the qubit can take options are left for future investigation. Furthermore, the x-axis was chosen simply as the original paper used half rotations around the x-axis. Nonetheless, the model could have employed the y-axis or z-axis instead. For the z-axis, the base state of the qubit would have to be changed as a rotation around the z-axis to a qubit in the state $|0\rangle$ is irrelevant.

3.2.2 Multi-axis encoding.

The second modification encompasses the introduction of rotations around the y and z axes. As explained earlier, an image loses data when it is converted from colour to grey. The original QFFNN model only concerns itself with greyscale images, but if the QFFNN model can encode and understand colour values, it can learn features that the greyscale model can not. To encode this additional information, the three colour values in the RGB colour space are split into separate rotations. For example, the red value is encoded by a rotation around the x-axis, green by a rotation around the y-axis and blue by a rotation around the z-axis. Similar to the original encoding with half rotations around the x-axis, the basic encoding is to apply X-, Y- and Z-gates if the values are above some threshold value. The following function is a mapping of the colour values to the gates

Algorithm 3.3 Multi-axis encoding

```
procedure MULTI-AXIS_ENCODING(qubit, $R, G, B$)
    $R \leftarrow$ normalise($R$)
    $G \leftarrow$ normalise($G$)
    $B \leftarrow$ normalise($B$)
    if $R > t$ then
        $qubit \leftarrow X$
    end if
    if $G > t$ then
        $qubit \leftarrow Y$
    end if
    if $B > t$ then
        $qubit \leftarrow Z$
    end if
end procedure
```

where t is the threshold value.

Nevertheless, there are some drawbacks to this map function. From the base state $|0\rangle$, the qubit can only reach $|1\rangle$. Hence, the mapping is not bijective. For example, the pixel values

$$a = 0.7, 0, 0, b = 0, 1, 0, \tag{3.9}$$

with a threshold $t = 0.5$, both map to the state, $|1\rangle$. Thus the model can not distinguish between the two values. Similarly, consider for instance the two pixel values

$$c = 0, 0, 0, d = 1, 1, 0. \tag{3.10}$$

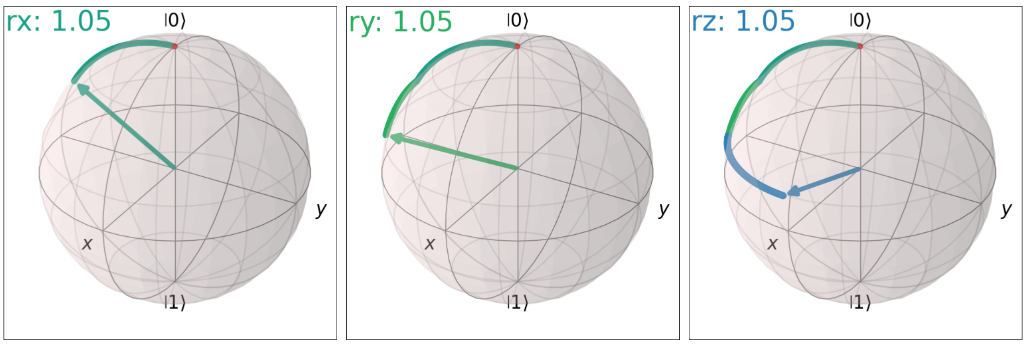
The first pixel, c, naturally applies no rotations and stays in $|0\rangle$. However, the second pixel, d, also rotates a half rotation around the x-axis, then a half rotation around the y-axis, bringing it back to the $|0\rangle$ state. A pixel with intense colours ends up in the same state as a background pixel. However, the perhaps most important limitation is that the Z-gate does not affect qubits in the two base states.

As a result, the two colours mapped by x and y rotations are conflicting, and the third colour is ignored. Clearly, this is not optimal. Therefore, by combining the multi-axis encoding with the free rotations encoding, an improved mapping function is proposed. The new mapping function is

Algorithm 3.4 Free rotation multi-axis encoding.

```
procedure FREE_ROTATION_MULTI-AXIS_ENCODING(qubit, $R, G, B$)
    $R \leftarrow$ normalise($R$)
    $G \leftarrow$ normalise($G$)
    $B \leftarrow$ normalise($B$)
    $qubit \leftarrow RX(R \cdot \pi)$
    $qubit \leftarrow RY(G \cdot \pi)$
    $qubit \leftarrow RZ(B \cdot \pi)$
end procedure
```

Both the drawbacks of the previous mapping are ameliorated by the free rotations. Firstly, pixels can now map to any state. Almost each pixel value has a unique superposition.



(a) Free rotation around the x-axis.

(b) Free rotation around the y-axis.

(c) Free rotation around the z-axis.

Figure 3.6: Showcasing the steps of the free rotation multi-axis encoding, all rotations are $\pi/3$ radians.

Nonetheless, some pixel values still end up in the same form. For instance, the pixels c and d both map to the base state $|0\rangle$.

In addition, given that the qubit has been set in a superposition by the X- and Y-gates, the Z-gate now changes the state of the qubit. It is, however, important to note that the problems have not been removed altogether. For instance, the pixels c and d still map to the same state, and the Z-gate still has no effect on the base states. It is, however, reasonable to assume that a realistic dataset such as CIFAR-10 consists of mostly unique mappings for each pixel. Furthermore, like the free rotation solution, it is the task of the feature map and PQC circuit to distinguish these values.

3.3 Image representations

A novel data encoding is suggested to utilise the information otherwise lost when converting from a coloured image to a binary image. Baseline binary models are implemented, along with the new models featuring the proposed data encoding. Whenever applicable, classical models are implemented as baselines. The models are trained following established best practices.

3.3.1 Colour space transformation

Classically, images are represented as matrices, where each element corresponds to a pixel in the image. Moreover, all preprocessing of images is done classically, as it falls outside the scope of the quantum models.

The conversion from coloured images to greyscale is done through a linear transformation. Although the transformation is an approximation of the advanced gamma correction method [46], it was chosen over gamma correction for a variety of reasons. Gamma correction is a far more expensive algorithm, and it is developed for contriving accuracy to the human eye, not for a machine [46]. As such, a computer is unlikely to understand a gamma-corrected image any better. Furthermore, this linear transformation is a common approach in image processing libraries used for image transformation for machine learning, such as Mathworks and Pillow [47, 48].

The linear transformation is the formula

$$\text{Greyscale} = 0.2989 \cdot \text{red} + 0.5870 \cdot \text{green} + 0.1140 \cdot \text{blue}, \tag{3.11}$$

following the ITU-R Recommendation BT. 601-2 to best represent the lumination in the picture [49, 50].

3.3.2 Downsampling

In digital image processing, downsampling refers to the process of reducing the number of pixels in an image. It is a crucial procedure in the NISQ era, especially for the QFFNN algorithm, which requires a qubit corresponding to each pixel. Without techniques such as downsampling, it would be impossible to represent larger images for the algorithm.

In image processing, two main downsampling techniques are cropping and resizing [51]. Cropping involves selecting a specific region of pixels while discarding the remaining ones to create a new smaller image. Numerous strategies for cropping exist, and in the present study, the centre crop technique is used, which rests on the assumption that objects of interest are positioned towards the centre. This assumption is particularly relevant in datasets such as CIFAR-10, where the images have been handpicked for object recognition [52]. Conversely, resizing creates a smaller or larger image by calculating pixel values from the original image. In this thesis, the resize method by `TensorFlow` was used, employing their default technique of bilinear interpolation [53]. This approach calculates the new pixel value by interpolating between the four nearest neighbours in the input image. To investigate which method is most effective on the CIFAR-10 dataset, the main dataset of interest, the classic model defined in section 3.7 was tested, employing the two downsampling methods. The results are presented in subsection 4.5.1. It was

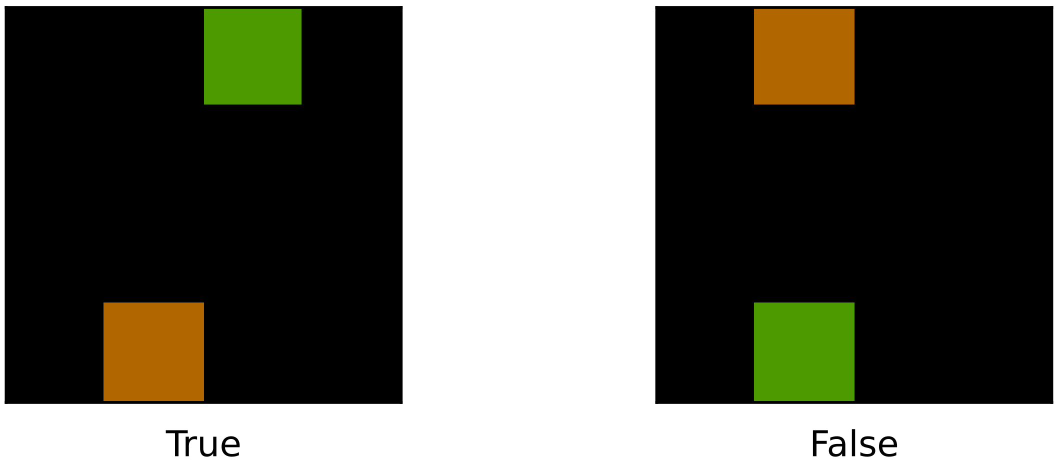


Figure 3.7: An example from each of the two classes in the pixel dataset. The position of the squares in each row is random.

concluded that resizing produced the best results, and accordingly, this method is used when necessary to downsample images.

3.4 Pixel dataset

The pixel dataset shown in Figure 3.7, is a dataset constructed specifically to show shortcomings in the greyscale model. The image size is four by four and a random pixel in the top row and a random pixel in the bottom row are coloured. The remaining pixels are null-value pixels. The dataset is a binary dataset where one class has a green-coloured cell in the top row and an orange-coloured cell in the bottom row. The other class consists of the consists of reversed images, one cell in the top row is an orange-coloured cell while one cell in the bottom row is green-coloured. The dataset aims to represent values that are different in a colour space while being equal when transformed into a greyscale space. By arranging the pixels at random positions along their respective rows, the model must understand not only the colours but also their respective position. The complete dataset has 16 distinct, yet similar, images for each class.

In subsection 3.3.1, a linear transformation formula is presented, whereas, for this dataset, a simpler transformation is given as

$$\text{Greyscale} = 0.3 \cdot \text{red} + 0.6 \cdot \text{green} + 0 \cdot \text{blue}. \tag{3.12}$$

This transformation was chosen for various reasons. Noticeably, the blue value is multiplied by zero. As such, the intensity of the blue colour is not represented in the greyscale space. Encoding only two of the base colours allows for experimenting with the effect of various gate combinations on the feature space. Additionally, fewer rotations are less error-prone.

For simplicity, the first colour was chosen as the base where red and green are multiplied by 1, giving the RGB value,

$$p_1 = 0.3\mathrm{R} + 0.6\mathrm{G} + 0\mathrm{B} \tag{3.13}$$

which corresponds to a green colour. As the luminosity of a pixel varies from 0 to 1, the inverse pixel becomes

$$\begin{aligned} p_2 &= 1 - p_1, \\ p_2 &= (1-0.3)\mathrm{R} + (1-0.6)\mathrm{G} + 0\mathrm{B}, \\ p_2 &= 0.7\mathrm{R} + 0.4\mathrm{G} + 0\mathrm{B}, \end{aligned} \tag{3.14}$$

which corresponds to an orange colour.

3.5 Tetrominoes

The tetrominoes dataset is similar, but more complex than the pixel dataset. The shapes are more intricate, made up of multiple coloured pixels. Like the pixel dataset, it is also four by four pixels and in the RGB space. But unlike the pixel dataset, the tetrominoes dataset is a multi-class dataset. Additionally, each class has a unique shape and a unique colour. A good greyscale model has no problem distinguishing these unique shapes, without considering the colours of the tetrominoes. Nonetheless, as explained earlier, NISQ computers are limited by how many qubits they can use. Consequently, downscaling images is of utmost interest. For example, a five-qubit computer is only capable of classifying two by two images. Four qubits for encoding and one readout qubit. When an image is downscaled, valuable shape information becomes less clear. Therefore, this dataset was selected to explore if a colour model can outperform a greyscale model

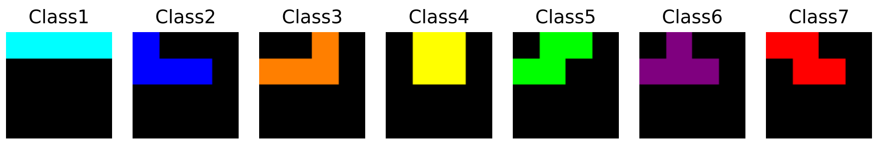


Figure 3.8: All seven tetromino pieces.

Table 3.1: Tetrominoes and their colours, represented in RGB and hexadecimal.

Class	Shape	Colour	RGB	Hex
1	**I**	Cyan	(0, 255, 255)	#00FFFF
2	**J**	Blue	(0, 0, 255)	#0000FF
3	**L**	Orange	(255, 165, 0)	#FFA500
4	**O**	Yellow	(255, 255, 0)	#FFFF00
5	**S**	Green	(0, 255, 0)	#00FF00
6	**T**	Purple	(128, 0, 128)	#800080
7	**Z**	Red	(255, 0, 0)	#FF0000

on downscaled images. It was also implemented as a simple dataset for testing the multi-class classifiers, where, given the proposed encodings work, the classifiers have an easier task of distinguishing the classes than in an even more realistic dataset such as the CIFAR-10 dataset.

The full dataset consists of the seven standard Tetris pieces, as shown in Figure 3.8. The RGB colours are listed in Table 3.1. Unlike the pixel dataset, where the colours were handpicked, these are the original colours for the tetrominoes creating a more realistic dataset. Likewise, to create a realistic scenario, the thorough linear transformation given in subsection 3.3.1 is used. As mentioned, the size is four by four, and all pieces are positioned in the top of the picture. For pieces that are three pixels wide, they are positioned to the left in the image, that is, at least one pixel appears in the left-most column. Furthermore, two examples of a re-scaled classes in both colour and greyscale are shown in Figure 3.9. Looking at this figure the inherent problem of the fixed rotation encoding becomes clear. As the tetrominoes are all positioned in the upper left corner of the images, the images all downscales to similar two by two images. Only the two top pixels are non-black, furthermore. Thus, with two input qubits each taking two distinct states, only four possible variation can be distinguished by the fixed rotation model. Furthermore, as all figures are positioned to the left, the leftmost pixel will never be black, reducing the options to three distinct images. Hence, the fixed rotation single-axis encoding will struggle to distinguish the classes.

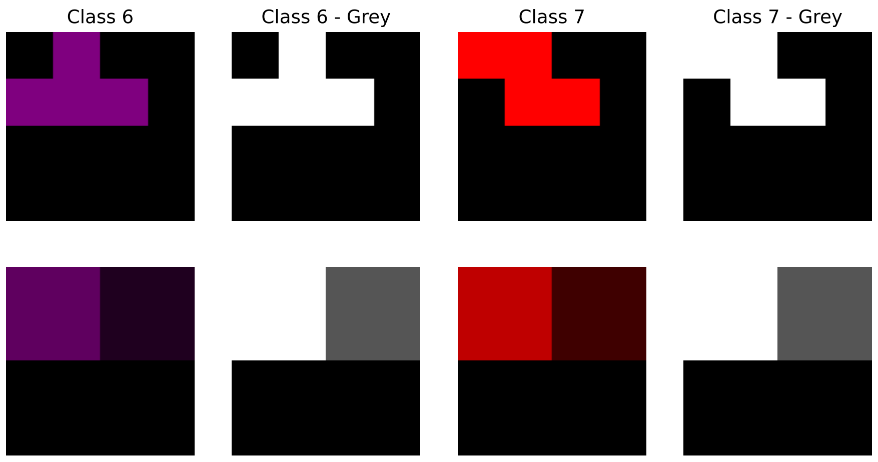


Figure 3.9: Class 6 and Class 7 tetrominoes and their greyscale counterparts. The bottom row is the corresponding images downscaled to 2 by 2 pixels.

Various subsets, as well as the full dataset, are tested. The dataset is split into a binary dataset, which purpose is to evaluate the performance of the colour model in a more realistic scenario than the pixel dataset. The dataset is also split into a trinary-class dataset. Both the trinary-class and the seven-class datasets are used to test the multi-class performance. Sampling only three classes give the fewest classes, allowing the multi-class classifier to be tested in a simple environment. The dataset is also split into a trinary dataset where the goal is to check a basic version of the multi-class problem. Finally, the full dataset is tested.

3.6 Multi-class classification

While binary classification is a useful tool, it is often the case that classifying multiple classes is of interest. For example, the MNIST and CIFAR-10 datasets are both multi-class datasets. One aim of this thesis is to evaluate the performance of the free rotation and the multi-axis encodings for such multi-class problems.

One approach to the multi-class problem is to use a multi-class classifier, a model that can output more than two values. Some thought was given to how the model could be extended, and the problems that would arise. As the readout qubit is already used to encode a class, and the qubit is measured multiple times to give a confidence value of the model prediction, it seems out of the question to use the qubit further. Instead, the model could be extended by adding more readout qubits, the unitary gates then act on all the readout qubits, predicting a class using binary numbers. Two qubits could encode four classes, three qubits eight classes and so on. While adding one or two qubits to a NISQ device is not a problem, there are some other issues with this approach. As the gates have to interact with two readout qubits they would become more complex which increases the error rate. Furthermore, the unitary gates would have to interact differently with each qubit to encode different classes. Likely increasing the circuit depth and thus the error rate. In addition, it is not clear that the parameters for each gate would be able to classify multiple classes. With all this in mind, and as the multi-class problem in itself is not the focus of this thesis, such an approach was not implemented.

Limited to binary classifiers, there are two approaches for a multi-class problem, one-vs-all and one-vs-one. In the one-vs-all approach, a binary classifier is trained with one class as the positive class and all other classes as the negative class. The classifier then only has to learn to recognise the positive class. For each class, a model is trained, and the models are combined into an ensemble model. This ensemble model makes predictions by choosing the model with the highest output value, that is, the model that is most confident in predicting the positive class. Comparatively, the one-vs-one approach trains a binary classifier for each pair of classes. The models are then ensembled and the class with the most votes is chosen. For a one-vs-all approach, the total number of models is n classes, while the one-vs-one approach requires $n(n-1)/2$ models. As the one-vs-all approach requires fewer models, thus fewer simulations which take less time or fewer NISQ devices which introduce less error, it was chosen for this thesis.

3.7 The classical model

The classical model implemented comes from [44] and is based on the LeNet model, a simple convolutional network [7]. The simple structure is showcased in Figure 3.10. In the example presented the input is a four by four image. The loss function is binary cross-entropy and like the QFFNN model, the optimiser is the Adam optimiser. This network was chosen as it resembles the QFFNN model by having comparably few trainable parameters. For the example provided below the model has 37 parameters, in contrast to the QFFNN model's 32 parameters for the same input. When the input is colour images the input size multiplies by three, thus also multiplying the parameters by three. However, this is still many orders of magnitude lower compared to a more advanced network which can reach millions of trainable parameters.

3.8 Simulating noise

To test the robustness of the model quantum noise is simulated, while the QFFNN model is tested on the CIFAR-10 dataset. TensorFlow provides support for simulating noise while training, in particular, for PQC experimentation they provide a noisy version of the PQC component.

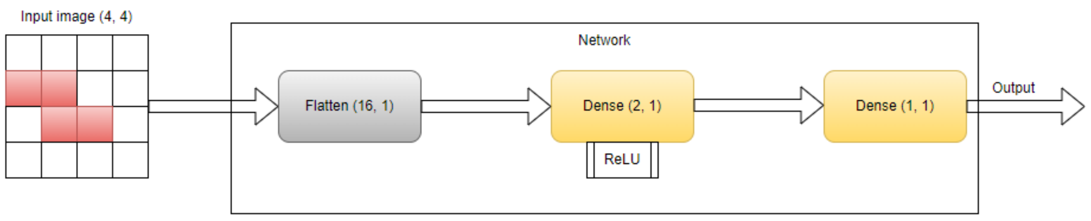


Figure 3.10: Classical LeNet architecture.

To imitate noise, TensorFlow [54] uses trajectory simulation, a memory-efficient approach which sacrifices some accuracy for great computational speedup. When using the trajectory simulation the simulations can be repeated multiple times to give some average noise in a circuit. Furthermore, TensorFlow provides the option to sample for noise during each repetition. This further increases the randomness in the noise and was thus chosen.

Chapter 4

Experimental Results

The following chapter details the experimental results of the simulations. First, the experimental setup is explained. In the following sections, the various results are presented. For each of the sections, the results are analysed. The metrics by which the models are evaluated are the quantitative measurements of accuracy and loss. Accuracy is the percentage of images which are correctly labelled, a higher accuracy suggests a better performance. Loss is the quantitative value with which the model fails to fully predict a class. A lower loss means the model is both guessing correctly and doing it confidently. A lower loss thus suggests better performance. Furthermore, the accuracy and loss can be applied to the validation or test datasets. Validation accuracy and loss are the values with which the model reports for the validation dataset while training. The test results are the results that the model reports on the test set in separate tests on unseen data, importantly the model is not training during testing. As the datasets are balanced, these metrics provide an accurate measurement of the models' performance. Additionally, validation loss and accuracy training is provided. These training evaluations allow for additional analysis to be performed, such as checking for over-fitting.

For the validation loss and accuracies, five independent simulations were completed. The line depicted in the graphs is the mean value and the shaded area is the empirical standard deviation. Unless otherwise stated, the quantum models are trained with 10 epochs in total, while the classical models are trained with 50 epochs.

Specifically, the results are split into the following four sections. First, the free rotation encoding is tested on the binary-class tetrominoes, the MNIST dataset and the CIFAR-10 dataset, the results are compared to the fixed rotation model and the classical model. Second, the multi-axis encoding for colours is evaluated on the pixel dataset, the tetrominoes dataset and the CIFAR-10 dataset. Both the fixed rotation and free rotation implementations are tested and compared to their single-axis counterparts. Third, the model is extended to the multi-class problems, first verified on a three-class tetromino dataset, then on the full tetromino dataset as well as the MNIST and CIFAR-10 datasets. Finally, various additional problems are tested, such as simulating noise and errors.

4.1 Experimental Setup

The tests are conducted on a consumer-grade computer running Python code. Machine learning was implemented through the TensorFlow [54] and Keras [55] libraries, while quantum computing was provided by the TensorFlow Quantum [56] library. All tests are simulations. For testing, 20% of the dataset is used. CIFAR-10 and MNIST are downloaded through the API provided by TensorFlow. For these datasets, the datasets are already split into train and test subsets, for the tetromino and pixel value this is done randomly. For all the datasets, the training set is further split with 80% randomly selected for training and 20% for validation. The tetromino and pixel value dataset are implemented through Python code.

4.2 Free rotations results

A proposed improvement was encoding the data as free rotations around each axis. The following section tests the free rotations on the binary tetrominoes dataset, the MNIST dataset and the CIFAR-10 dataset. The colour images were converted to greyscale images using the standard linear transformation as detailed in the background.

4.2.1 Binary-class tetrominoes

To verify that the model is capable of handling free rotations it is first tested on a downsampled version of the tetrominoes dataset. The images are rescaled to 2 by 2 pixels and class-1 and class-2 from the dataset are selected. For comparison and verification, the fixed rotation model is also tested. The results in Figure 4.1 show that the free rotation model was able to learn the classes while the fixed rotation was not.

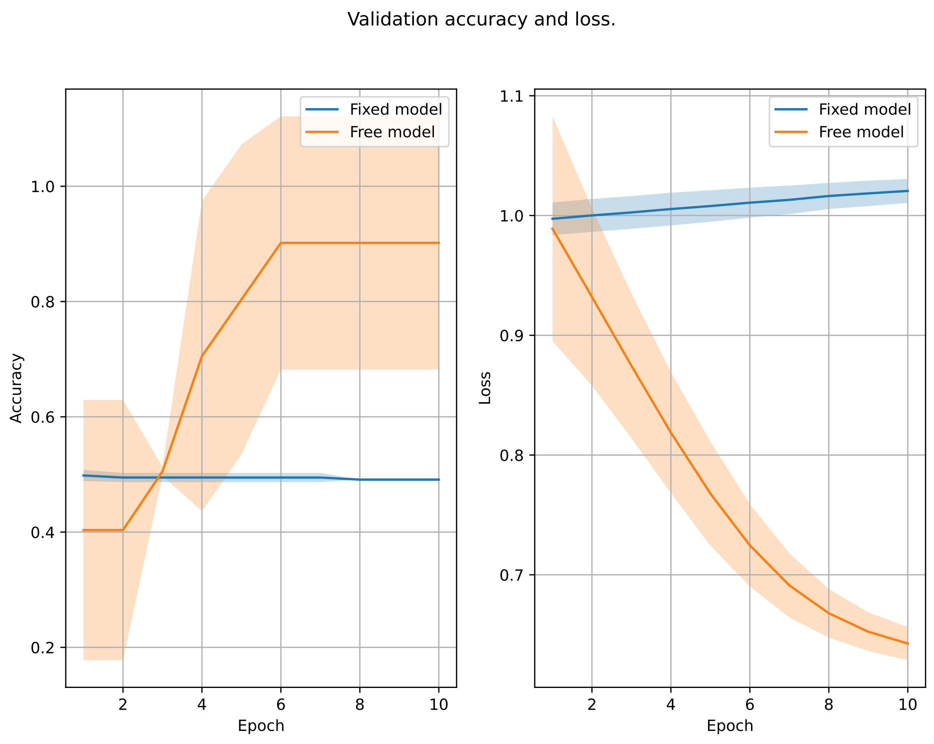


Figure 4.1: The validation accuracy and loss for the QFFNN greyscale model with fixed and free encoding on class-1 and class-2 from the tetrominoes dataset.

4.2.2 MNIST and CIFAR-10

For MNIST, the free rotation was tested on class-3 versus class-6. For the CIFAR-10 dataset, the classes were 6 versus 8. For both datasets the images are downscaled to a 4 by 4 pixel size. In Table 4.1, the accuracy and loss for each model are given.

Table 4.1: The test accuracy and loss of the QFFNN model with fixed and free single-axis encoding on the MNIST and CIFAR-10 datasets.

Model	Accuracy	Loss
MNIST fixed	0.844	0.340
MNIST free	0.891	0.329
CIFAR-10 fixed grey	0.691	0.672
CIFAR-10 free grey	0.722	0.661

The validation accuracy and loss for the free rotation and the fixed rotation for MNIST are presented in Figure 4.2. Looking at this more detailed information, it becomes clear that the free rotation consistently outperforms the fixed model. In addition, the variation between the five simulations is greater for the fixed model.

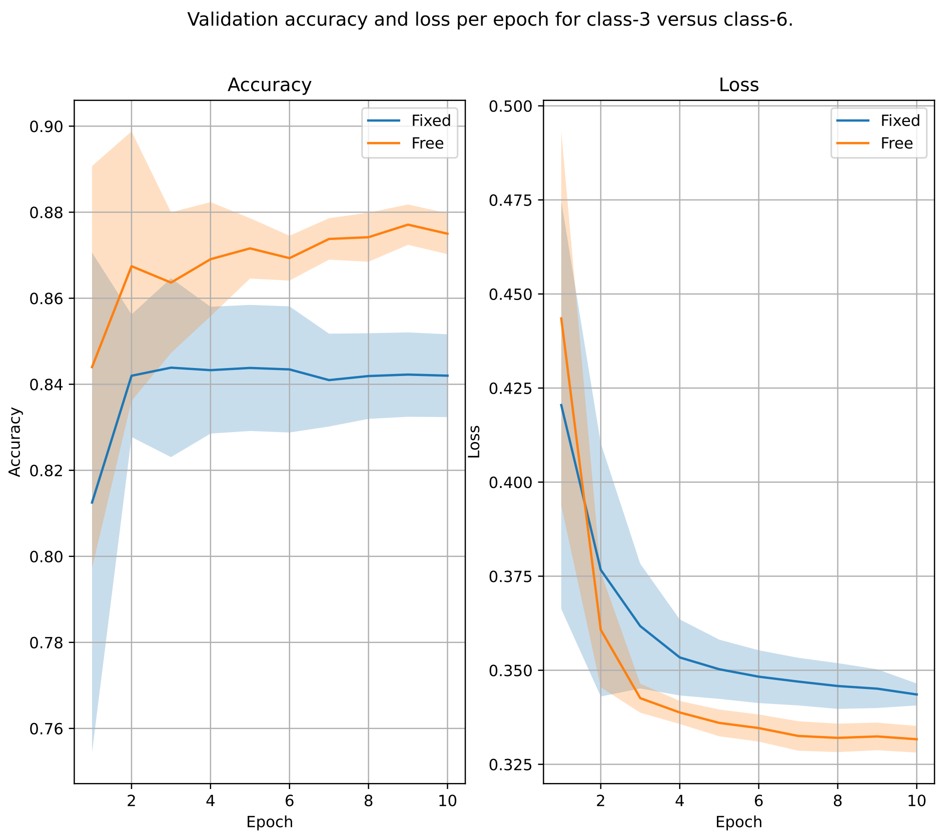


Figure 4.2: The validation accuracy and loss for the QFFNN model with free and fixed single-axis encoding on class-3 and class-6 from the MNIST dataset.

The same graphs are plotted for the CIFAR-10 dataset in Figure 4.3. Again the results are in favour of the free rotation, achieving higher accuracy and lower loss. Noticeably the variation for the fixed model is greater.

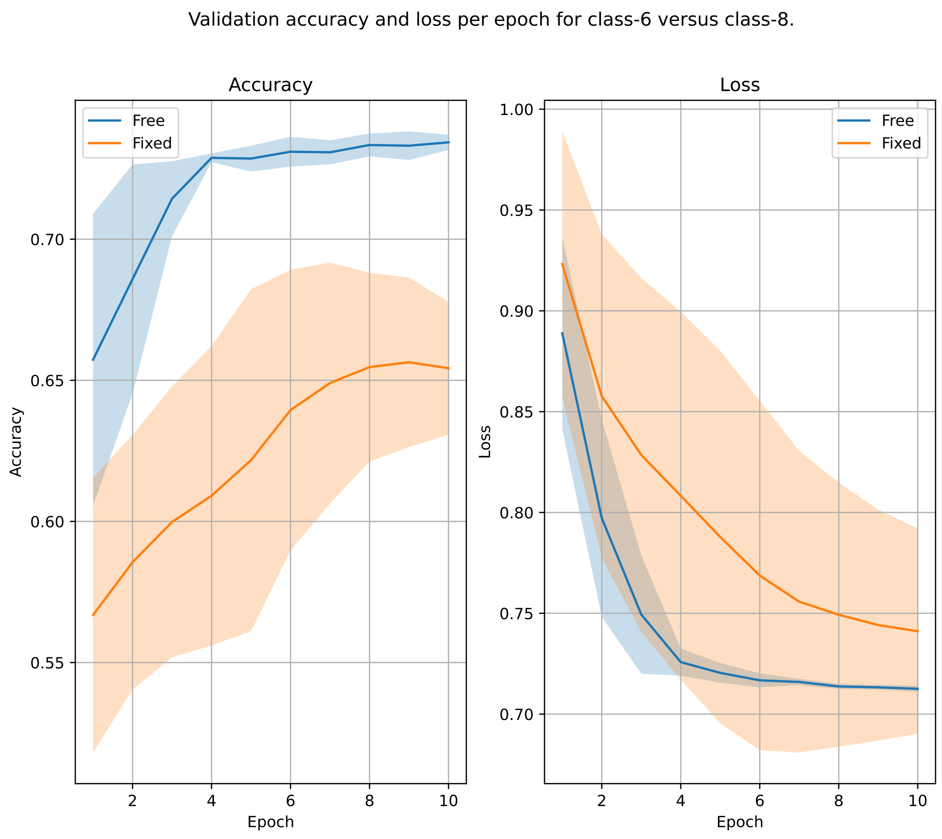


Figure 4.3: The validation accuracy and loss for the QFFNN model with free and fixed single-axis encoding on class-6 and class-8 from the CIFAR-10 dataset.

4.2.3 Discussion of results

It is important to recall that, while the greyscale model fails to classify the dataset, this is the expected result, as the pixel values map to equal states in the input qubits for both classes. By changing the threshold, such that one of the classes received a rotation which the other did not, it is possible that the fixed model could predict the classes. This also highlights a strength of the free rotations encoding which does not rely on the correct thresholding value being selected.

Another interesting result is the reduced variation from the free encoding for the advanced datasets. While a decreased variation is in itself not a positive, this could be further proof that the encoding has a positive effect. While free rotations introduce superpositions, which by nature are random – a potential source of confusion, if the model on average is able to estimate and utilise the free rotation it can better group values together. Thus the change in parameters gives more gradual changes to accuracy.

4.3 Multi-axis encoding results

First, the fixed multi-axis encoding is tested, that is, the encoding which applies set rotation around each axis based on some threshold value. For the tests, a threshold value of $t > 0.5$ was chosen as this is the middle of the possible values in all the datasets. The encoding is tested on the pixel dataset, the binary tetrominoes dataset and the CIFAR-10 dataset.

4.3.1 Pixel dataset

The results in Table 4.2 show that, as expected, the grey model has an effective 50% accuracy. That is, the model is unable to distinguish the models. Notice that also the classical greyscale model was unable to learn.

Table 4.2: Test accuracy of the base models on the pixel dataset.

Model	Accuracy	Loss
Quantum grey	0.517	0.998
Classical colour	1.000	0.237
Classical grey	0.484	1.002

For the next section, the colour encoding is tested. Recall that for the pixel dataset only the red and green values have values above 0. Hence, the red value is mapped to the X-gate and the green value is mapped to the Y-gate or Z-gate. These results, Table 4.3, show that the X-gate, mapping only the red values, is enough to learn the dataset.

Table 4.3: Test accuracies of the quantum fixed multi-axis encoding on the pixel dataset.

Rotations	Accuracy
Red $\rightarrow$ X	1.000
Red $\rightarrow$ X, green $\rightarrow$ Y	0.545
Red $\rightarrow$ X, green $\rightarrow$ Z	1.000

A deeper insight into the training process is shown in Figure 4.4. The encoding used here is the X-gate plus Z-gate combination. As the results show, the greyscale model has a consistent loss, indicating no learning. Contrary, the colour model has a decreasing loss and stabilises at 100% accuracy at the tenth epoch for all five simulations.

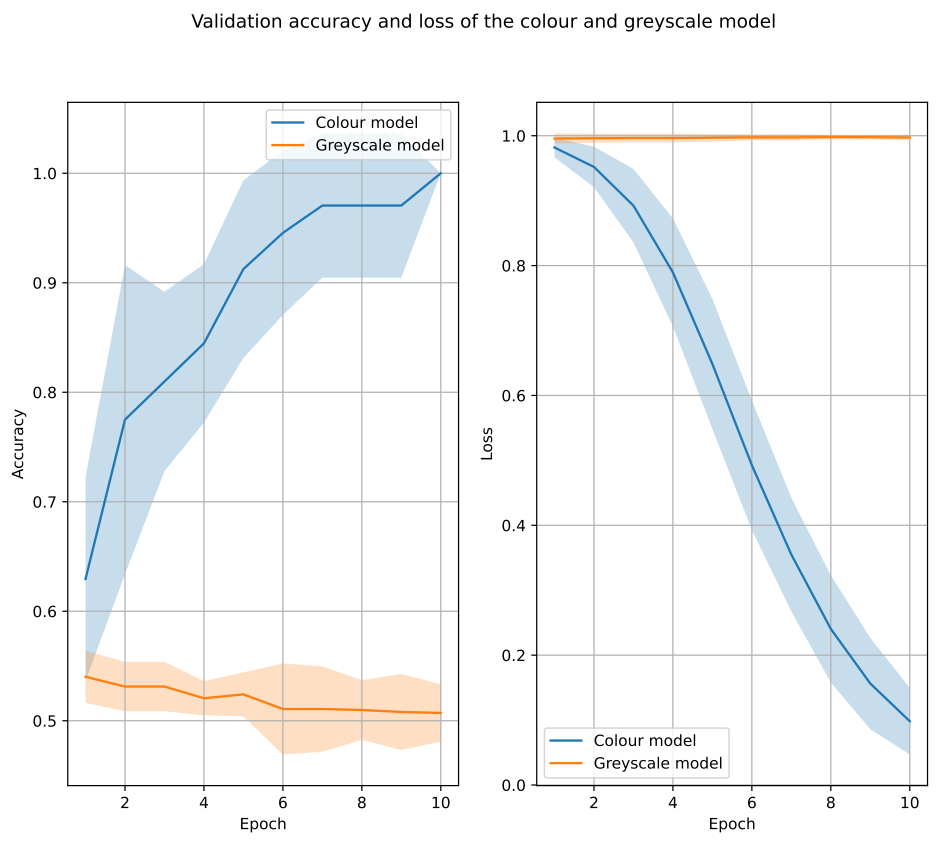


Figure 4.4: The validation loss for the QFFNN model with fixed multi-axis encoding versus fixed single-axis encoding on the pixel dataset.

Figure 4.5 shows the corresponding training data for the classical model. As can be observed, the graphs are very similar, but note that the classical colour model only starts learning after about 5 epochs and does not finish until around 25 epochs, noticeably more than the quantum model, which starts learning from the first epoch and finishes by the tenth.

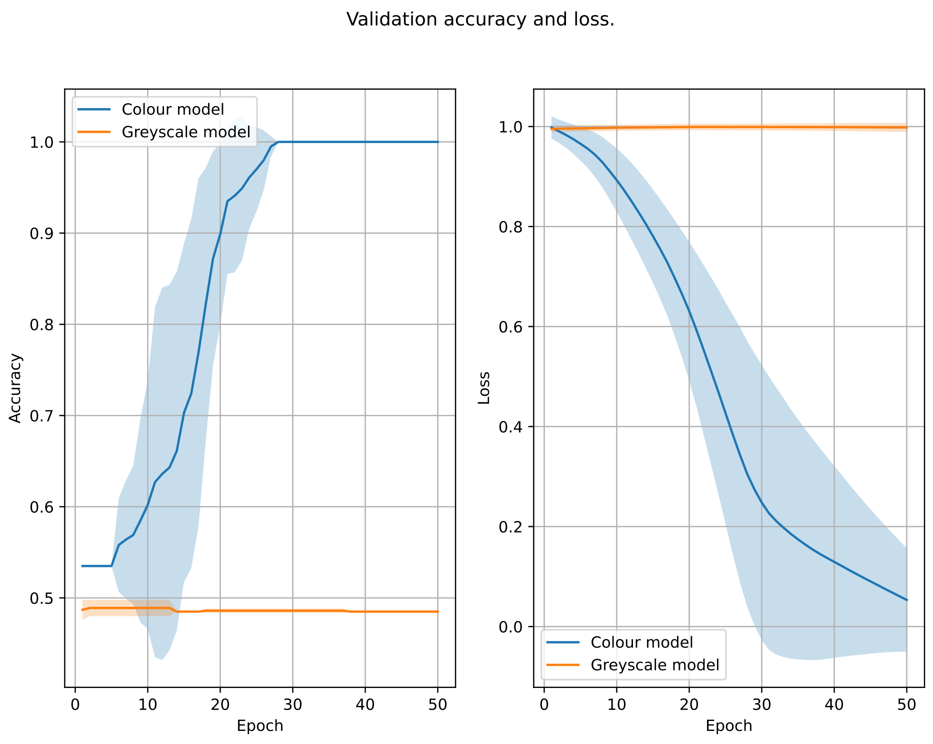


Figure 4.5: The validation accuracy and loss for the classical model with colour images versus greyscale images on the pixel dataset.

4.3.2 Binary-class tetrominoes

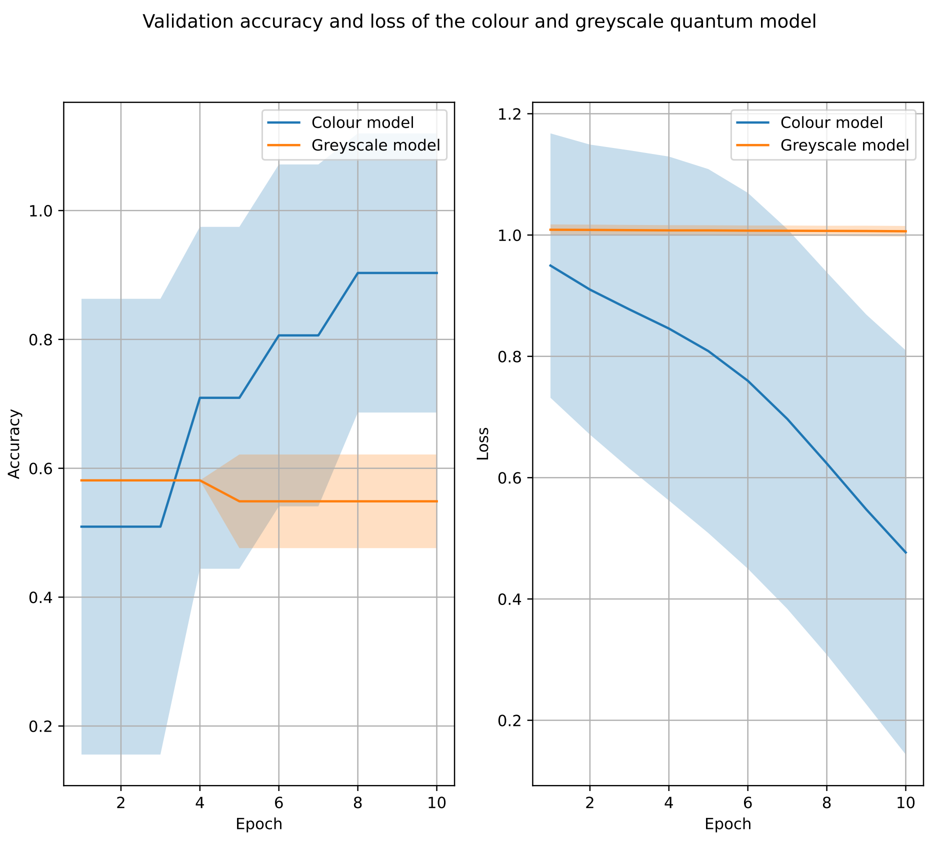
Following are the model accuracies for the various models on the binary class tetrominoes. The two classes were class-1 and class-2, consult section 3.5. The data was resized to a two by two image for all models. Furthermore, recall that for the tetrominoes, the greyscale conversion no longer makes the classes indistinguishable. With this in mind, it is not a surprise that the classical model on the grey data learnt to classify. The simulations also show that the multi-axis encoding provides an improvement over the free-axis encoding.

Table 4.4: The test accuracy of the models on class-1 and class-2 from the tetrominoes dataset.

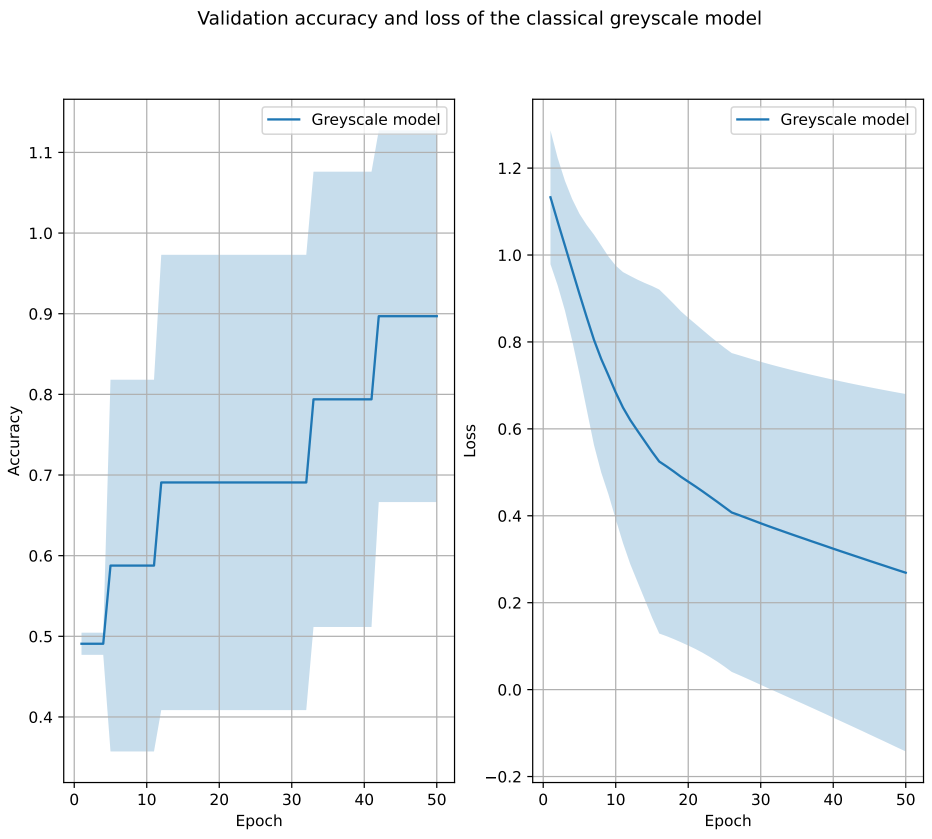
Model	Accuracy
Quantum colour	1.000
Quantum grey	0.482
Classical grey	1.000

Figure 4.6a and Figure 4.6b displays the validation accuracy and loss during the training. Similarly to the pixel dataset, the greyscale QFFNN model has a stable loss, indicating no learning. The colour QFFNN model, however, has greater variance in both loss and

accuracy than when trained on the pixel dataset. This variation is matched by the classical model, indicating that the dataset is more difficult to learn. Furthermore, the QFFNN model learns the data in fewer epochs than the classical model.



(a) The validation accuracy and loss for the QFFNN model with fixed multi-axis encoding versus single-axis encoding on class-1 and class-2 from the tetrominoes dataset.



(b) The validation accuracy and loss for the classic greyscale model on class-1 and class-2 from the tetrominoes dataset.

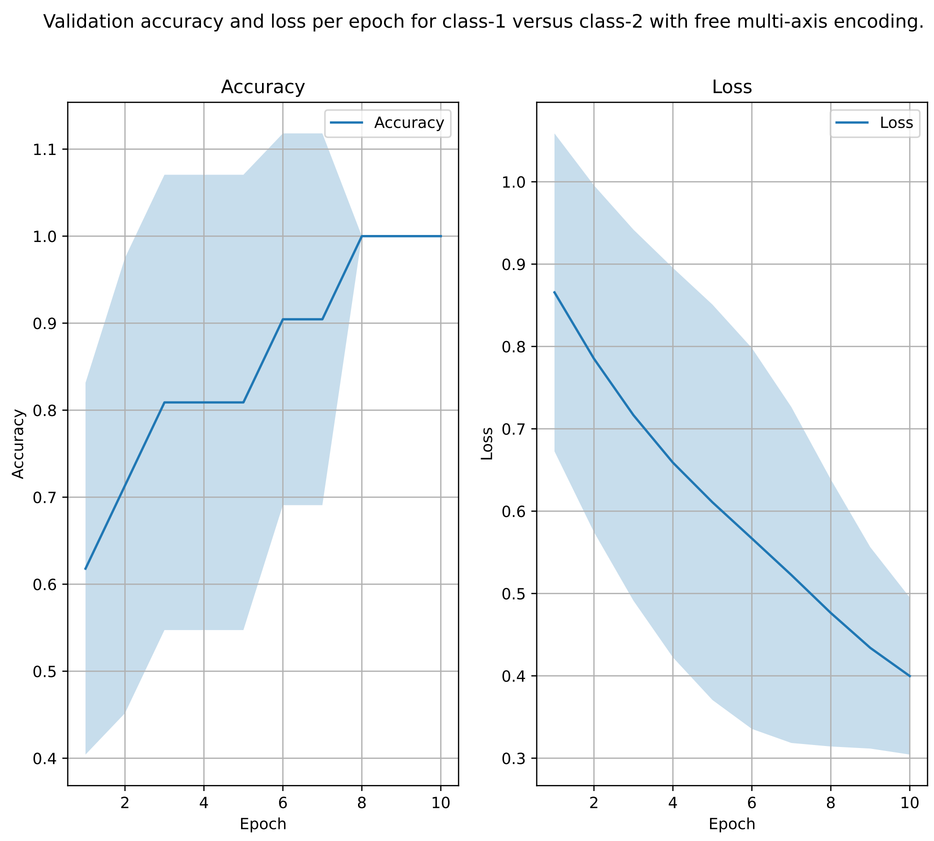


Figure 4.7: The validation accuracy and loss for the QFFNN model with free multi-axis encoding on class-1 and class-2 from the tetrominoes dataset.

In addition to the fixed multi-axis encoding, also the free multi-axis encoding was tested. Comparing the results in Figure 4.7 with the fixed encoding, it is clear that the free encoding gives faster convergence towards 100% accuracy. The model does not, however, show less variance during training than the fixed encoding.

4.3.3 CIFAR-10

Figure 4.8 depicts the results of both the types of multi-axis encoding and the single-axis encoding. The images were rescaled to a 4 by 4 image size. Both approaches outperform fixed colour encoding. Furthermore, the free multi-axis encoding achieves a higher accuracy than the fixed multi-axis encoding. Nonetheless, the loss values are about equal. The free encoding does have a lower variance.

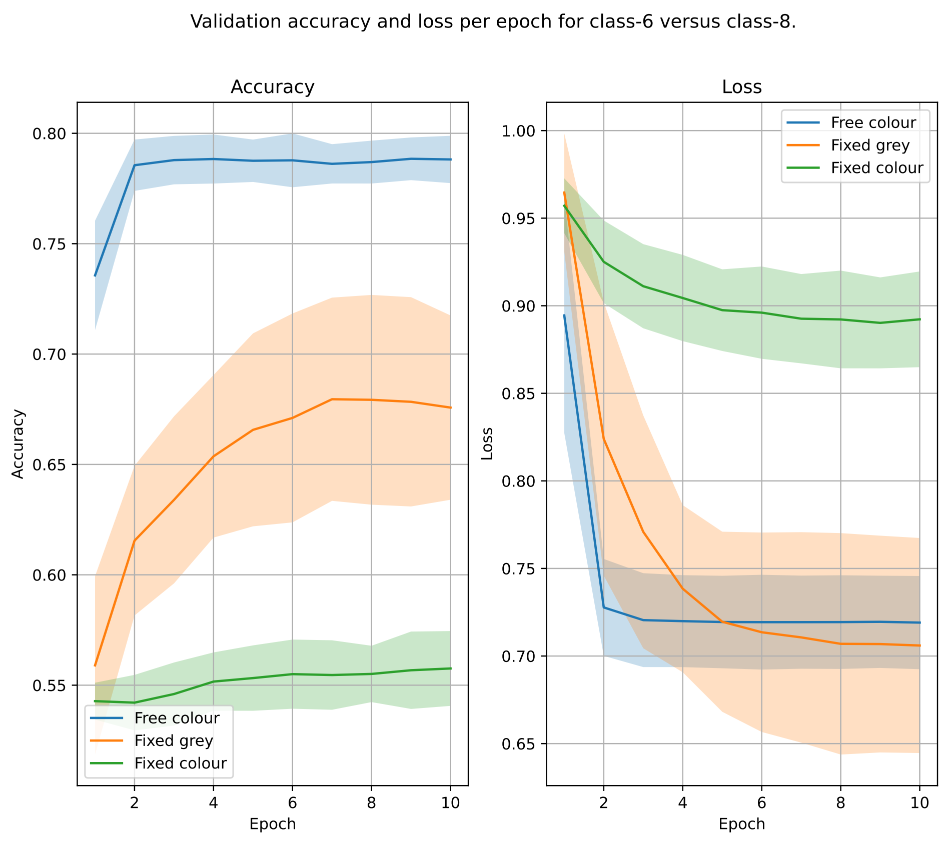


Figure 4.8: The validation accuracy and loss on class-6 and class-8 from the CIFAR-10 dataset. Comparing free multi-axis encoding, fixed multi-axis encoding and fixed single-axis encoding.

4.3.4 Discussion of results

Noticeable for the pixel dataset is the success of the $Red \rightarrow X$, a single-axis encoding, while the greyscale model failed. However, this is predictably the result, as only one of the two pixels has red values above the threshold value. Consequently, when considering the red values only through this mapping, the question that the model is asked to classify is if the reddish pixel is positioned in the uppermost or lowermost row, ignoring all other data. Furthermore, the X, Y encoding achieving approximately 50% verifies that the two gates nullify each other. In addition, the Z-gate is shown to not have a negative effect on the model.

Another interesting point is that the quantum model approaches 100% accuracy faster than the classical model. One possible rationale for this observation is that the algorithms use different loss functions. Unlike the cross entropy loss function, the hinge punishes not only based on wrong predictions but also based on the confidence of the prediction. This might motivate the model to quicker approach a 100% accuracy, especially when the dataset is so simple. Another possibility, which could be considered in parallel, is that the parameters in the PQC adapt more at the same time, compared to the weights in the classical model.

4.4 Multi-class results

For the multi-class section, the models are trained as one correct class versus the other wrong classes. The dataset is balanced to make each class contain equally many images. For the multi-class predictions, the models are combined into one ensemble model.

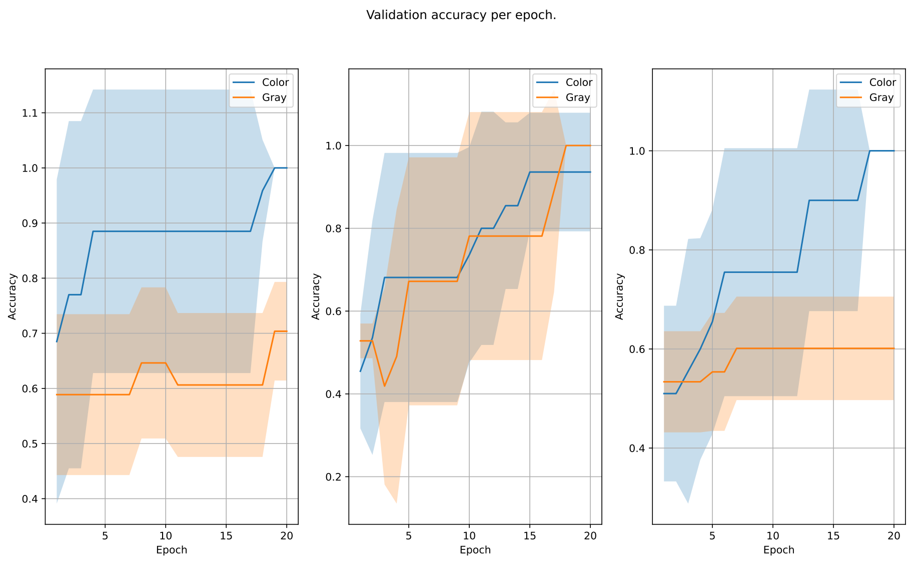
4.4.1 Trinary-class tetrominoes

The trinary dataset is the dataset composed of the first three tetromino classes, downscaled to a 2 by 2 image size. Both the single-axis and the multi-axis results were done with free rotations, and the peak test accuracy over five simulations was as follows.

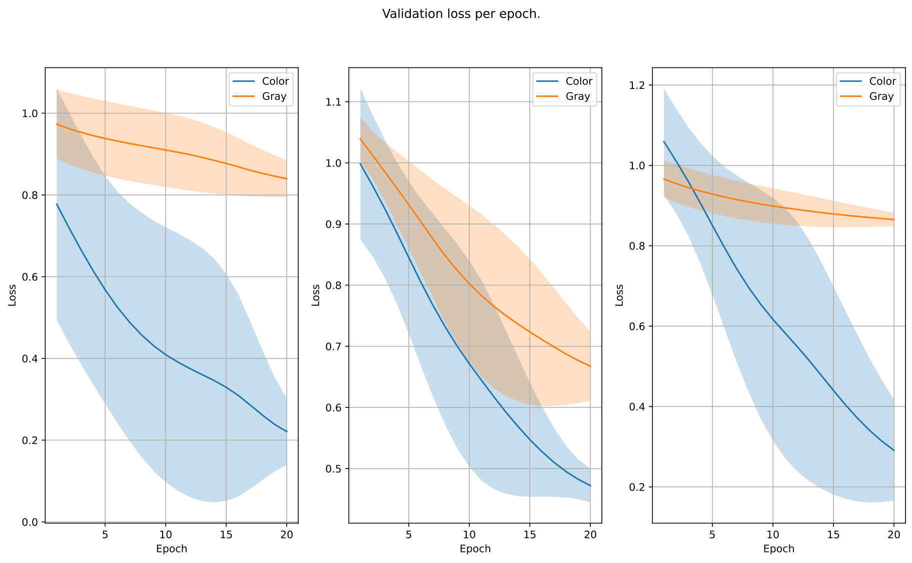
Table 4.5: Test accuracy of the models for each true class on the tetrominoes 3-class dataset.

Model	Class 1	Class 2	Class 3
Multi-axis	1.000	1.000	1.000
Single-axis	0.724	1.000	0.734

Figure 4.9a and fig. 4.9b provides a deeper dive into the training process. The result shows that for class-1 and class-3, while the reported highest value was slightly above 70% for both classes, the mean accuracy is about 66%, corresponding to a random guess baseline. The loss for these two classes goes down as the models reach 66% but then seems to level out, suggesting no further learning. The multi-axis model for class-1 and class-3 shows large variance, but all attempts do learn to recognise the images. For class-2, it is clear that the dataset is easier to recognise as also the greyscale model achieves 100% accuracy. Noticeably, when looking at the loss the multi-axis model consistently outperforms the single-axis model.



(a) The validation accuracy for the QFFNN model with free multi-axis and free single-axis encoding on the tetrominoes 3-class dataset. In terms of the true class, the dataset is ordered class-1, class-2 and class-3.



(b) The validation loss for the QFFNN model with multi-axis and single-axis encoding on the tetrominoes 3-class dataset. In terms of the true class, the dataset is ordered class-1, class-2 and class-3.

Following are the accuracies for the ensemble model showing that the multi-axis model was completely successful, while the single-axis model failed to classify the class-1 images.

4.4.2 Seven-class tetrominoes

The seven-class tetrominoes dataset is the full tetrominoes dataset. All images were downscaled to 2 by 2 pixels. Both the single-axis and the multi-axis results were done with free rotations, and the peak test accuracy over five simulations was as shown in Figure 4.10.

Table 4.6: Ensemble model prediction accuracy for each class in the tetrominoes 3-class dataset.

Model	Class 1	Class 2	Class 3	Total accuracy
Multi-axis	100	100	100	100%
Single-axis	0	100	100	66.667%

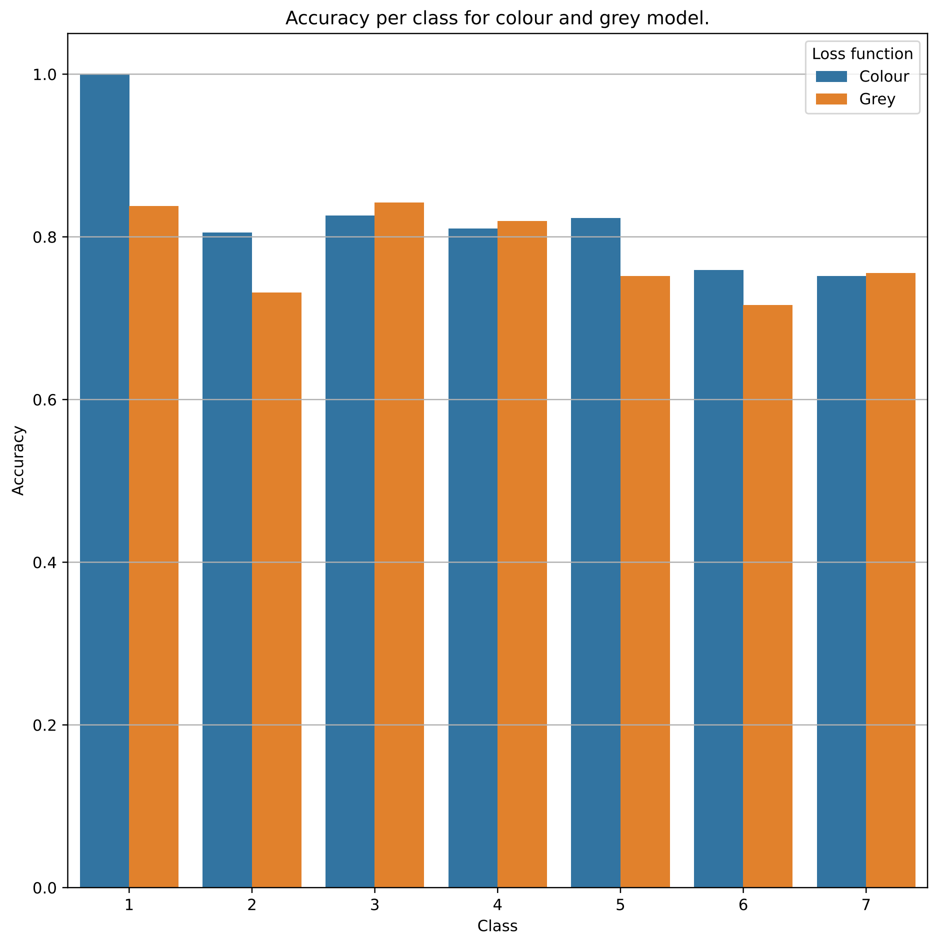


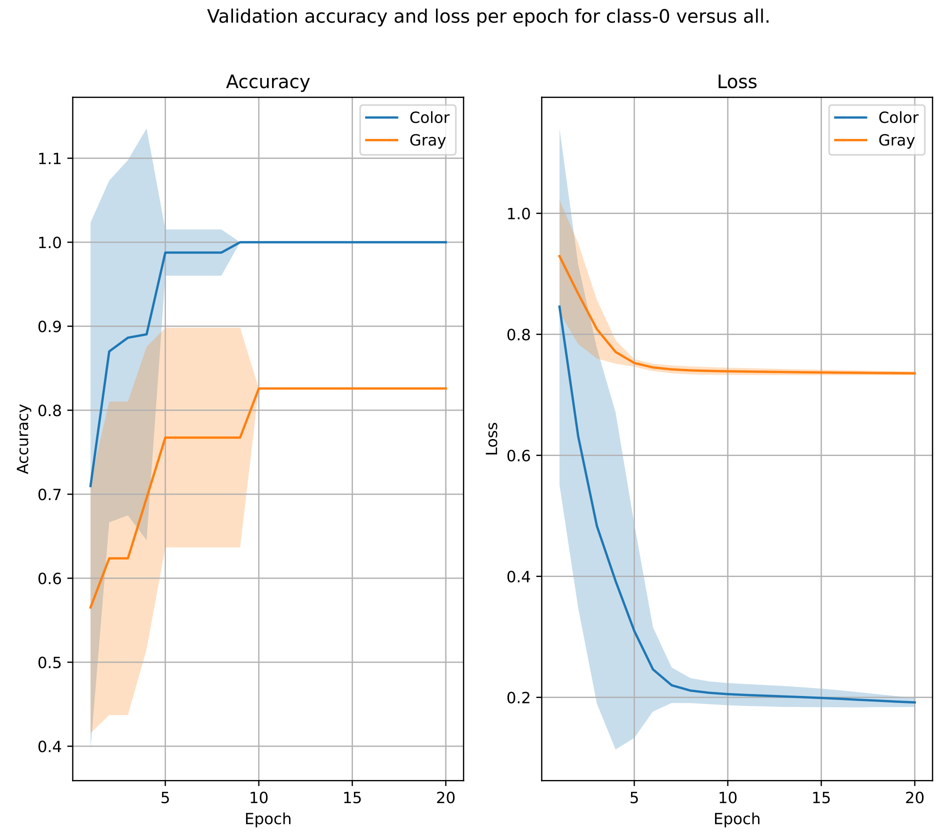
Figure 4.10: Test accuracies for free multi-axis rotations versus free single-axis rotations for the full tetrominoes dataset.

The ensemble model accuracies are shown in Table 4.7 and show that the multi-axis model achieves much higher accuracy than the single-axis model. Nonetheless, it also shows that the multi-axis model is not perfect.

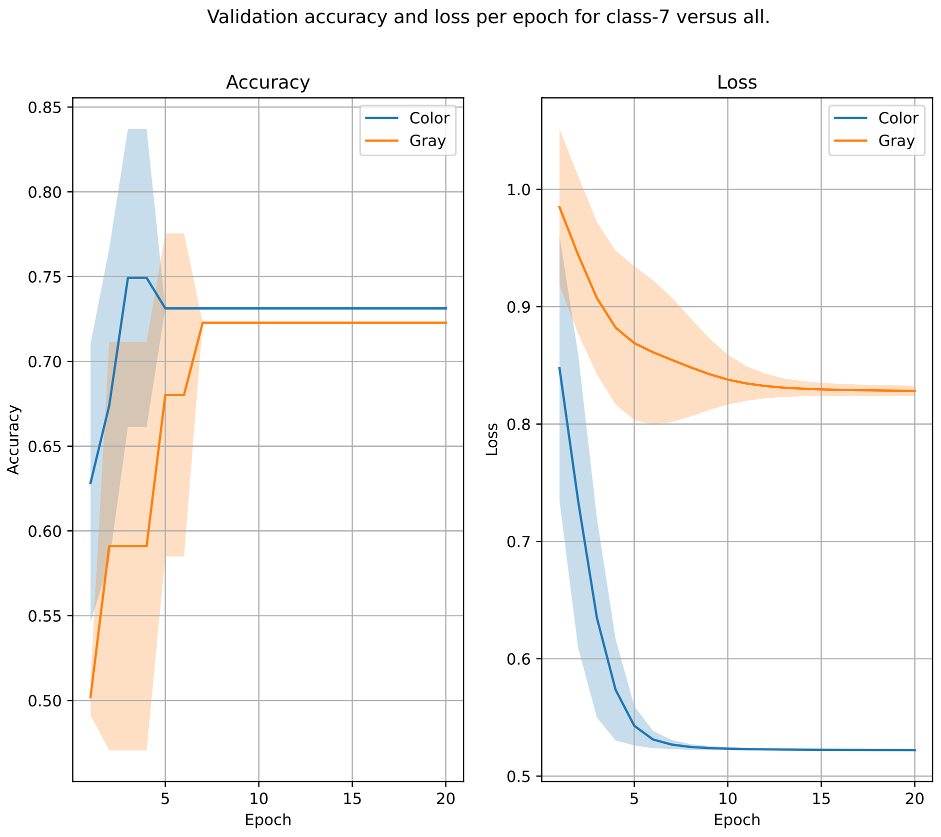
Table 4.7: Ensemble model prediction accuracy for each class in the Tetrominoes Dataset.

Model	1	2	3	4	5	6	7	Total accuracy
Multi-axis	100	100	0	100	100	100	100	85.7%
Single-axis	100	0	0	0	0	100	0	28.6%

The following two tables, Figure 4.11a and Figure 4.11b depict the validation accuracy and loss during training for the class-0 and class-7 models. These two models were the best and worst-performing models. Since the training is done in a one versus-all setting, the true and false class is not balanced, thus, the base accuracy is 6 out of 7, or approximately 86%. As the graphs depict, the grey model for the class-0 problem performed at about this accuracy, suggesting that it did not learn the class at all, which is further confirmed by the loss. Similarly, for the class-7 problem did both models end up at about 5/7 which suggests that the models did not only fail to learn the class-7, but also confused mistook another class for class-7. Noticeably, the loss for the class-7 problem is much lower for the colour model, despite similar accuracy performance.



(a) Validation accuracy and loss for free multi-axis rotations versus free single-axis rotations for class-0 versus all on the tetrominoes dataset.



(b) Validation accuracy and loss for free multi-axis rotations versus free single-axis rotations for class-7 versus all on the tetrominoes dataset.

4.4.3 CIFAR-10

The following section presents the results of the multi-class problem for the CIFAR-10 dataset on images downscaled to 4 by 4 pixels. The free and fixed multi-axis encoding is shown in Figure 4.12 and and the free and fixed single-axis encoding is shown in Figure 4.13. In both cases the multi-axis rotations report improvements. For the colour encoding, three of the classes exhibit marginally worse results for the multi-axis rotation, the other seven classes report various degrees of improvements. Noticeably, the fixed rotation encoding performs at the level of a random classifier, suggesting that the fixed rotations cancels each other out, ruining the training. For the single-axis encoding, the free encoding either performed on par with the fixed encoding or slightly above. Possibly, too much information is lost in the downscaling process. Table 4.8 lists the ensemble model of the free multi-axis versus the fixed single-axis encoding, showing an improvement.

Table 4.8: Ensemble model prediction accuracy for each class versus the rest.

Model	0	1	2	3	4	5	6	7	8	9	Total accuracy
Multi-axis	23	44	1	18	50	25	12	11	24	16	22.4%
Single-axis	43	8	29	8	9	4	12	17	22	22	17.4%

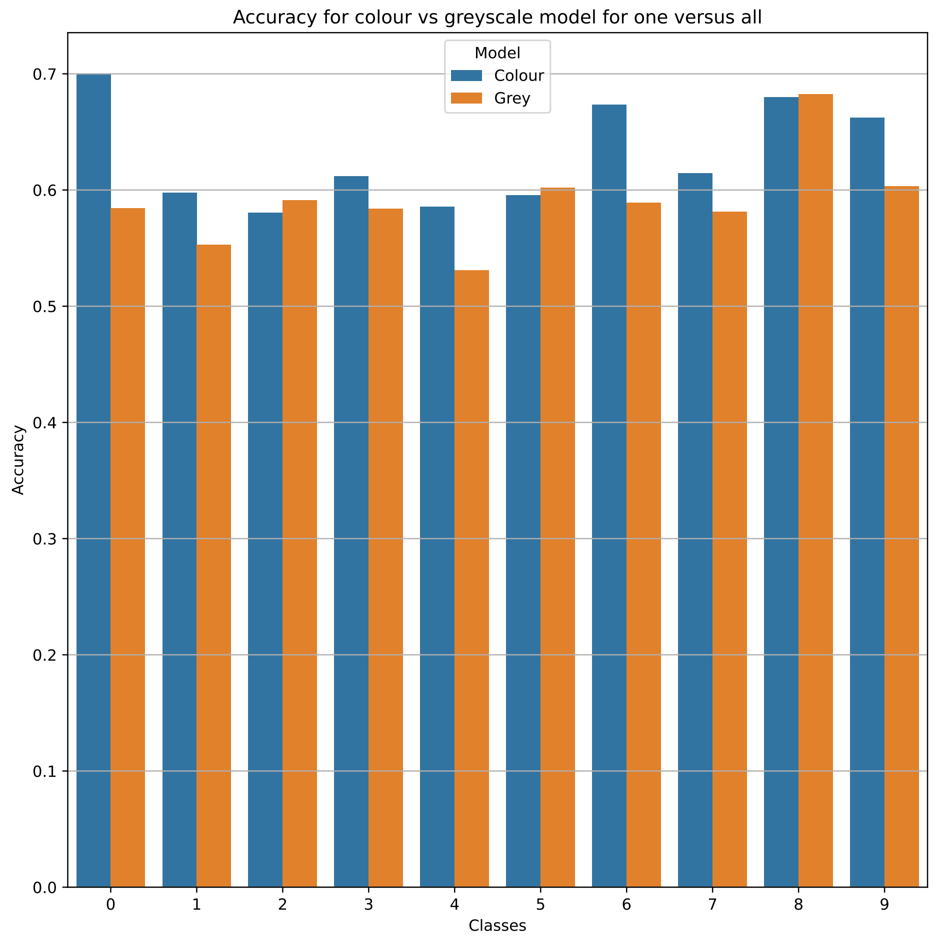


Figure 4.12: Test accuracies for free multi-axis rotations versus free single-axis rotations for the CIFAR-10 dataset.

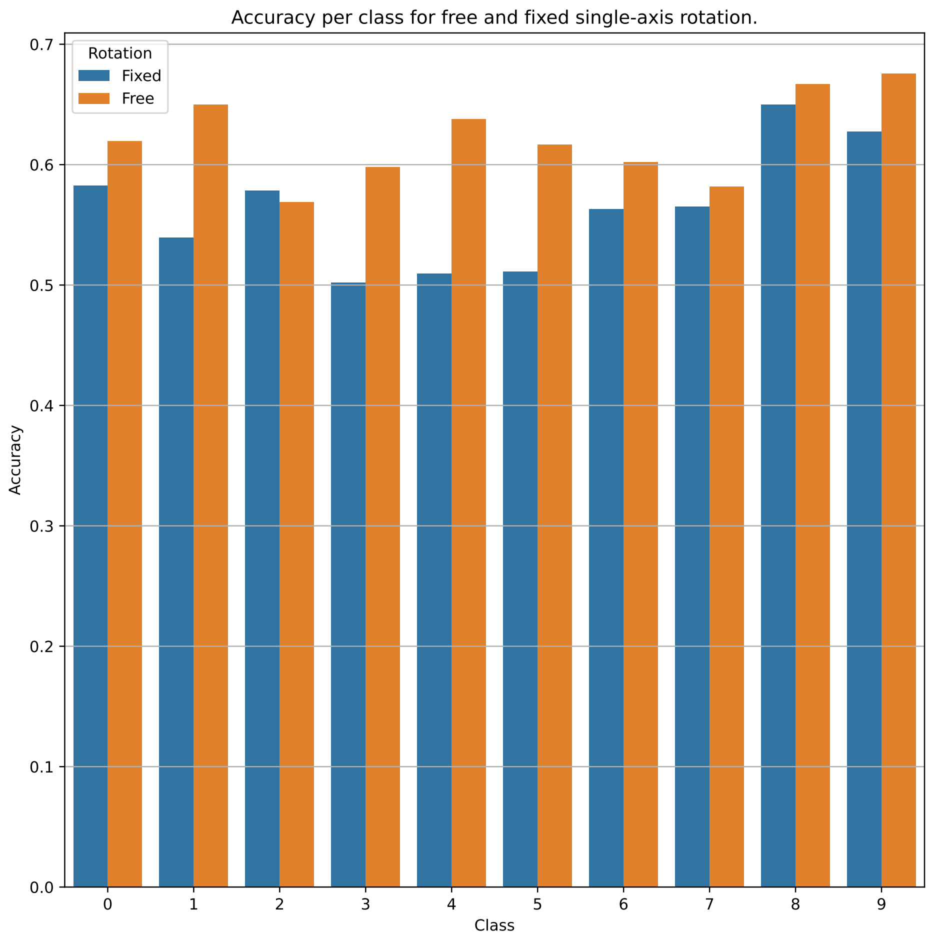


Figure 4.13: Test accuracies for fixed versus free single-axis rotations for the CIFAR-10 dataset.

4.4.4 MNIST

Figure 4.14 and table 4.9 present the results of the free encoding versus the fixed encoding on the multi-class MNIST problem, downscaled to a 4 by 4 image size. In general, the free rotations either performed on par with the fixed rotations or better. Nevertheless, the improvements were not great, only about 5% for the best improvement. Furthermore, the ensemble models achieved equal accuracy for both encodings. In particular, some classes stood out as being easier to predict, in particular class-2 which achieved both the highest accuracies and the most correct predictions for the ensemble model.

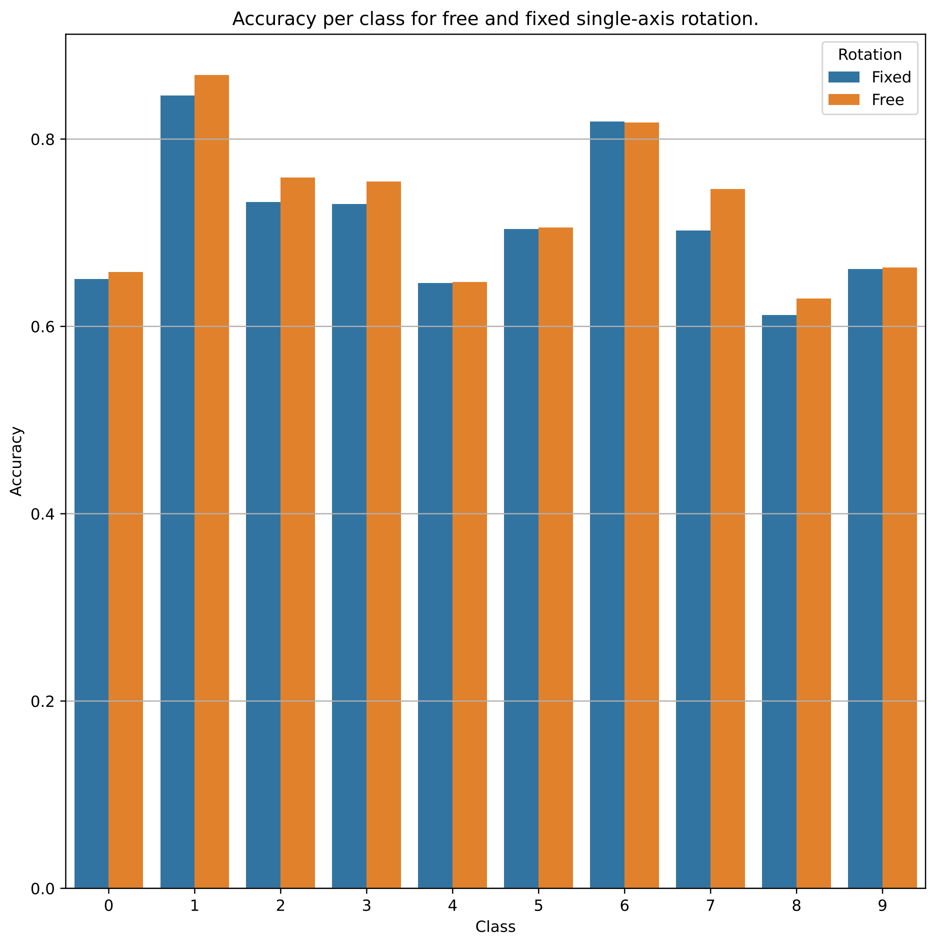


Figure 4.14: Test accuracies for fixed single-axis rotations versus free single-axis rotations for the MNIST dataset.

Table 4.9: Ensemble model prediction accuracy for each class versus the rest.

Model	0	1	2	3	4	5	6	7	8	9	Total accuracy
Fixed rotation	14	82	25	40	6	41	10	13	44	28	30.3%
Free rotation	42	88	21	42	6	29	8	19	42	7	30.4%

4.4.5 Discussion of results

For the 7-class tetrominoes problem, the perhaps most interesting result is the loss performance of the colour model on the class-7 problem. Despite showing about equal accuracy to the greyscale model, the colour model has a significantly lower loss. This suggests that while the colour model does not learn to recognise class-7, it has great confidence in recognising images in the false class. This is further illustrated by the small variance and steep decline in loss, taking only 5 epochs to stabilise.

For the multi-axis results on CIFAR-10, two classes stand out as exhibiting the greatest gain: class-0 (aeroplanes) and class-6 (frogs). For the aeroplane class, one plausible explanation is that the colours are easy to generalise and distinguish. The pictures are of commercial aeroplanes, that are white [57], or of military aeroplanes, that are grey [58]. In addition, many of the pictures are taken of an aeroplane against a blue or white sky. Such a generalisation of colours could be part of what the model learns to outperform the greyscale model. A similar case can be made for the frog class. Consisting of mostly green frogs in a green forest setting, if most pixel values are greenish the image is likely to be a frog. Furthermore, classes that did not observe significant improvements, such as class-1 (cars) and class-5 (dogs), are more varied in the colour of both the object of interest and the background. Noticeably, class-8 (boats) are primarily composed of white boats against blue water or blue sky backgrounds. Despite this potential for generalisation, the model for class-8 did not achieve any improvement over the single-axis counterpart. However, the model did achieve the second-highest accuracy overall and the highest for the single-axis encoding. Therefore, any such generalisation might be irrelevant to the spacial information.

While the free encoding model for the MNIST dataset outperformed the fixed encoding in the binary class, the results suggest at best a mediocre improvement for the free encoding for the multi-class problem. There are various possible reasons for this. One reason could be that the figures are relatively thin drawings and centred by weight, almost all non-black pixels are in the centre of the image. Thus, when the pixels are downsampled so greatly, it is possible that the centre pixels become very similar to highly illuminating values, with the opposite case for the outer pixels. Hence, the PQC can struggle to distinguish the pixels from the various classes, effectively turning it into binary encoding. A possible solution would be to use a different downsampling technique.

4.5 Additional results

In the section below some additional results of interest are posted. First, the investigation finds the best downsampling technique for the CIFAR-10 dataset. Then noisy simulations are tested, also on the CIFAR-10 dataset. Finally, the squared hinge loss function is tested on the seven-class tetrominoes dataset.

4.5.1 The best downsampling for CIFAR-10

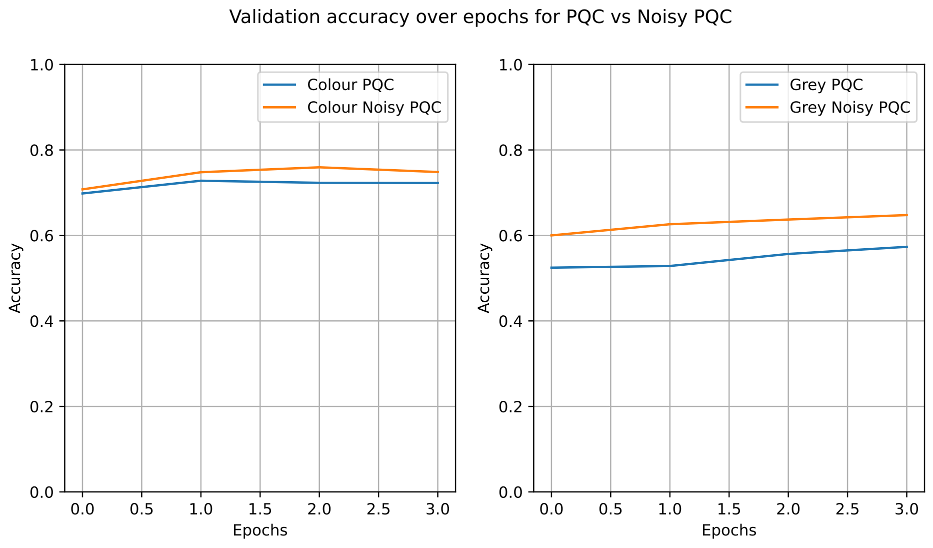
An important question when downscaling large images to a measly 16 pixels, is which downsampling technique to use. The results in Table 4.10 show the test loss for the classical model between randomly selected classes. For all classes, the resize method performed best. This is to be expected as there is no guarantee that the object of interest, which it is assumed contributes the most during classification, is in the centre of the image.

Table 4.10: Test loss for the classical model for the resize and crop downscaling methods for each class versus the rest.

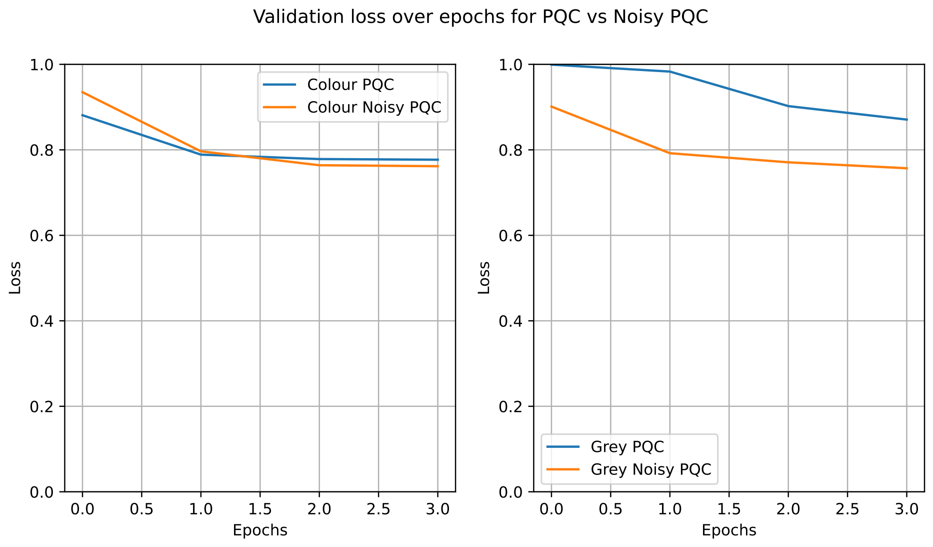
Classes - Loss	Resize	Crop
6 versus 8	0.231	0.343
3 versus 4	0.533	0.649
7 versus 9	0.425	0.586
2 versus 7	0.515	0.648

4.5.2 Testing the model with noisy simulations

To validate the performance of the model in a more realistic scenario, the model was tested with noise simulations. The images were downscaled to 4 by 4 and the repetitions in the noise sampling was set to 100. The performance is displayed Figure 4.15a and Figure 4.15b against a noiseless simulation. As the figures show, the model is adapting to the noise, even outperforming the noiseless model. While the higher accuracy is within experimental variation, it is possible that the multiple repetitions not only give the model time to adjust to the noise. It also gives the model time to learn the dataset beyond what the model learnt from a single noiseless repetition.



(a) Validation accuracy for the colour and greyscale model on the CIFAR-10 dataset with and without noise simulation for the PQC. The models are tested on class-6 versus class-8. The noise is sample based, the repetitions is set to 100.



(b) Validation loss for the colour and greyscale model on the CIFAR-10 dataset with and without noise simulation for the PQC.

4.5.3 Squared hinge loss function

During experimentation, some results suggested that the squared hinge loss function outperformed the normal hinge loss function. This was consequently tested in depth. The tests were done on the tetrominoes seven-class problem, downscaled to a 4 by 4 image size. As Figure 4.16 displays, the squared hinge loss function greatly improved the results for the multi-class problem.

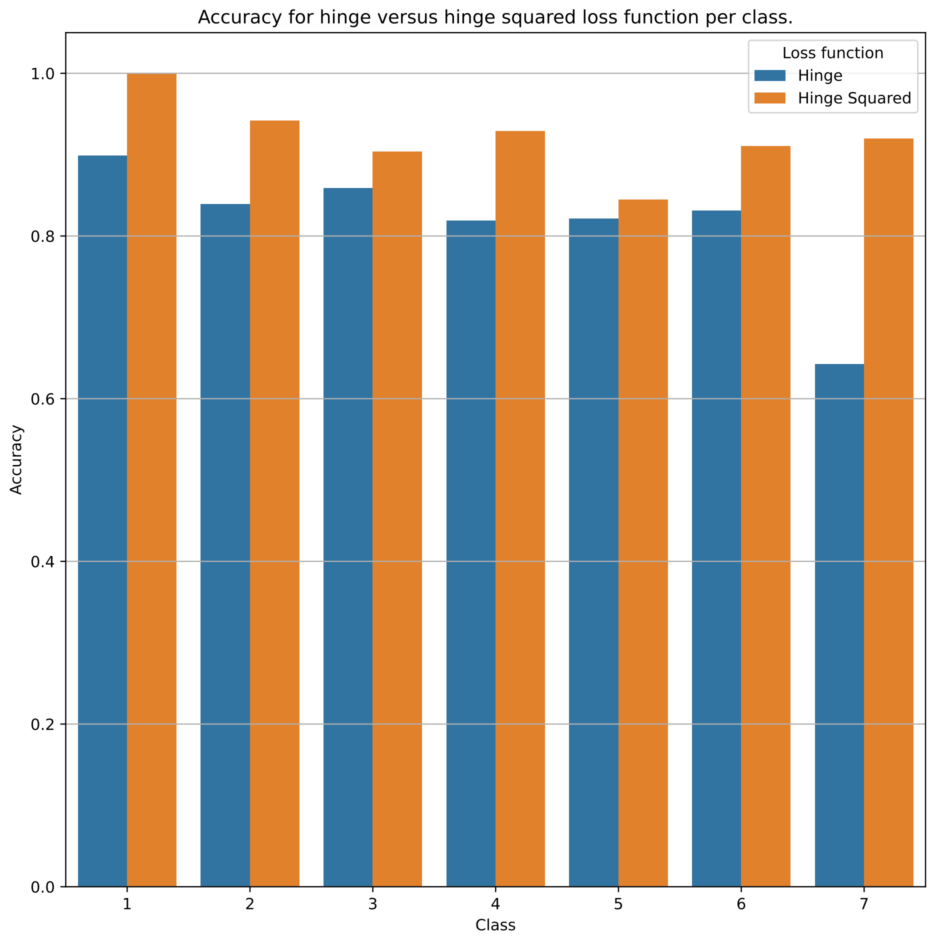


Figure 4.16: Test accuracies for hinge loss function versus hinge squared loss function for the seven-class tetrominoes dataset.

Chapter 5

Final Remarks

With automated learning and vast amounts of data, machine learning addresses problems that traditional algorithms struggle to overcome. One of the areas where machine learning has stood out as being most prominent is image recognition. Another field that has witnessed pronounced improvements is quantum computing. By harnessing the dazzling properties of quantum mechanics, a new method of computing has arisen. Introducing both unique new solutions and increased computational power. With this incredible combination at hand, quantum computing is opening up new possibilities.

Quantum computing has the potential to revolutionize image recognition. Quantum machine learning models fuse quantum computing and image recognition into powerful models. One such model is the quantum feed-forward neural network (QFFNN) introduced by [8], using superposition and entanglement to provide a novel method of understanding image features. Nevertheless, contemporary quantum computers are limited in circuit depth, restricted in qubit capacity and exposed to noise and errors.

5.1 Conclusion

While addressing these limitations of NISQ computers, this thesis aimed to enhance the quantum feed-forward neural network model. New encoding methods were developed allowing the model to consider more data at once. Furthermore, new datasets were developed to evaluate the effectiveness of these encodings.

In particular, three specific encodings were developed aiming to answer the three research questions. The first question asked if the model could be improved by encoding greyscale values instead of binary values. The model was successfully enriched by encoding greyscale values as free rotations around a single axis, improving the performance of the model. In particular, it showed that the free rotations ignore the need for good thresholding values in binary images. The second research question was concerned with multi-channel pixel values. By allowing the model to map each channel to a separate axis, it successfully encoded colour information from the RGB space. Again the results were positive, as the model learnt to classify images that the greyscale model could not. Furthermore, it showed that the new encoding did not tamper with the model's ability to learn multiple shapes for each class. However, it also showed some limitations of the fixed multi-axis encoding as the rotations might counter each other. As both these results proved successful, the free multi-axis encoding was devised, combining the two previous ones to answer the third research question. The results showed that the free multi-channel encoding provided a further increase to the model's performance. Also on the advanced datasets CIFAR-10 and MNIST datasets.

With these new improvements equipped, the model was extended to the multi-class problem, where the enhancements again showed advancement. The best results were perceived for the downscaled seven-class tetrominoes dataset, where the improvements allowed the model to achieve almost 100% ensemble accuracy. For CIFAR-10 and MNIST the results were positive, but the ensemble models performed poorly, suggesting that the downscaling is drastic. However, for the CIFAR-10 dataset, the fixed multi-axis encoding performed on par with a random algorithm for many of the classes, also performing worse than the binary encoding, again suggesting that the fixed rotations hinders each other. Nevertheless, the best performing encoding was the free multi-axis encoding. This shows that the encoding is able to provide valuable information, even on such heavily downscaled advanced images.

Additionally, the resize method from TensorFlow was found to outperform the centre crop method for the CIFAR-10 dataset, therefore, this method was used throughout the project. The performance of the squared hinge loss function was tested on the multi-class problem, outperforming the linear hinge loss function. While not used for this project, future work should consider to the squared hinge loss function during training. Finally, the network was verified in a noisy simulation environment, where the model adapted effectively, performing on par with the noiseless model.

To conclude, the new encodings were effective at enhancing the model, and the model was successfully extended to the multi-class problem. It was verified that the quantum feed-forward neural network model performs better with more information about the lumination of the pixels. Confined to 16 input + 1 readout qubits, the model was able to achieve almost 80% accuracy on binary-class images of 32 times 32 pixels that were downscaled by a total factor of 64. Meanwhile, the best publicly available quantum computers currently boast 127 qubits, allowing for 126 input qubits which could reduce the downscaling factor to improve the accuracy or downscale images of greater size.

5.2 Future work

The following are two major directions for future work based on this thesis. The first proposed direction is to implement other encodings. While the encoding developed here proved to be effective in its own right, there is certainly potential for taking it one step further. For instance, there might be an encoding that better incorporates the z-axis for the third colour, as currently, the rotations around the z-axis rely on there being some rotation around the other axes first. Another possibility is to use some of the other encoding techniques entirely. By using an encoding like Flexible Representations of Quantum Images, it could be possible to keep the colour encoding while also reducing the qubits required to encode an image or put another way, some amount of qubits could encode a larger image. With a larger image, more detail is available from which the model can learn. There remains a great question as to how the model would be able to interact with such an encoding.

The second direction is to consider the strategies suggested here in other models. For instance, the QCNN model also uses PQC as the main component and encodes each pixel value on a per-qubit basis. Therefore, it is likely that the encoding methods suggested here could be used to extend the input layer in the model, allowing also the QCNN model to consider more of the incoming data. Furthermore, the QFFNN model is not only an image classification model. Therefore, is it possible that the improvements found can be employed for other use-cases of the QFFNN model.

As an additional note, while the following experiments have been thoroughly tested in simulations, it remains to implement them on physical hardware.

Bibliography

[1] D. Silver and D. Hassabis. *AlphaGo: Mastering the ancient game of Go with Machine Learning.* https://ai.googleblog.com/2016/01/alphago-mastering-ancient-game-of-go.html, 2016.

[2] Y. Wang. Quantum computation and quantum information. http://www.jstor.org/stable/41714771, 2012.

[3] Microsoft. Quantum computing history and background. https://learn.microsoft.com/en-us/azure/quantum/concepts-overview, 2022.

[4] J. Preskill. *Quantum Computing in the NISQ era and beyond.* https://www.arxiv-vanity.com/papers/1801.00862/, 2018.

[5] IBM. Compute resources. https://quantum-computing.ibm.com/services/resources?tab=systems, 2023.

[6] Papers with code. *Image Classification on MNIST.* https://paperswithcode.com/sota/image-classification-on-mnist, 2023. Leaderboard of models for the MNIST dataset.

[7] Y. LeCun and others. Gradient-based learning applied to document recognition. http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf, 1998.

[8] E. Farhi and H. Neven. *Classification with Quantum Neural Networks on Near Term Processors.* https://arxiv.org/pdf/1802.06002.pdf, 2018.

[9] E. Umbaugh. *Digital image processing and analysis : human and computer vision applications with CVIPtools.* https://www.arxiv-vanity.com/papers/1801.00862/, 2011.

[10] S. W. Golomb. *Polyominoes: Puzzles, Patterns, Problems, and Packings.* Princeton University Press. (Second Edition), doi=https://doi.org/10.2307/j.ctv10vm1sc, 1994.

[11] Tetris Holding. *Tetris* [Electronika 60]. Russia: Pajitnov, A., 1985.

[12] M. Stevens and S. Pradhan. *Playing Tetris with Deep Reinforcement Learning.* http://cs231n.stanford.edu/reports/2016/pdfs/121_Report.pdf, n.d. Retrieved 2023.

[13] Liu et al. *Hybrid Quantum-Classical Convolutional Neural Networks.* https://arxiv.org/pdf/1911.02998.pdf, 2021.

[14] Tetris Holding. *Tetris gameplay.* Retrieved from https://tetris.com/article/33/tetris-tips-for-beginners, 2017.

[15] F. Bloch. *Nuclear Induction.* https://journals.aps.org/pr/abstract/10.1103/PhysRev.70.460, 1946.

[16] M. H. Devoret and J. M. Martinis. *Implementing Qubits with Superconducting Integrated Circuits.* https://link.springer.com/chapter/10.1007/0-387-27732-3_12, DOI: https://doi.org/10.1007/0-387-27732-3_12, 2005.

[17] A. Einstein, B. Podolsky, and N. Rosen. *Can Quantum-Mechanical Description of Physical Reality Be Considered Complete?* https://journals.aps.org/pr/pdf/10.1103/PhysRev.47.777, 1935.

[18] E. Schrödinger. *Discussion of Probability Relations Between Separated Systems.* Proceedings of the Cambridge Philosophical Society, 31: 555–563, 32 (1936): 446-451, 1935, 1936.

[19] B. Foxen et al. Demonstrating a continuous set of two-qubit gates for near-term quantum algorithms. https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.125.120504, 2020.

[20] W. Huang, C.H. Yang, K.W. Chan, et al. Fidelity benchmarks for two-qubit gates in silicon. https://doi.org/10.1038/s41586-019-1197-0, 2019.

[21] Google Quantum AI. *Suppressing quantum errors by scaling a surface code logical qubit.* https://www.nature.com/articles/s41586-022-05434-1, 2023.

[22] Microsoft. *Windows 10 system requirements.* https://support.microsoft.com/en-us/windows/windows-10-system-requirements-6d4e9a79-66bf-7950-467c-795cf0386715, Retrieved 2023.

[23] S. E. Venegas-Andraca and S. Bose. Storing, processing, and retrieving an image using quantum mechanics. https://www.spiedigitallibrary.org/conference-proceedings-of-spie/5105/1/Storing-processing-and-retrieving-an-image-using-quantum-mechanics/10.1117/12.485960.short?SSO=1, 2003.

[24] P. Q. Le, F. Dong, and K. Hirota. *A flexible representation of quantum images for polynomial preparation, image compression, and processing operations.* `https://link.springer.com/article/10.1007/s11128-010-0177-y`, 2011.

[25] B. Sun et al. *An RGB Multi-Channel Representation for Images on Quantum Computers.* `https://www.fujipress.jp/jaciii/jc/jacii001700030404/`, 2013.

[26] Y. Zhang et al. Neqr: a novel enhanced quantum representation of digital images. `https://link.springer.com/article/10.1007/s11128-013-0567-z`, 2013.

[27] A. Anand et al. Quantum image processing. `https://arxiv.org/pdf/2203.01831.pdf`, 2022.

[28] E. Aïmeur, G. Brassard, and S. Gambs. Machine learning in a quantum world. `https://doi.org/10.1007/11766247_37`, 2006.

[29] S. Lloyd, M. Mohseni, and P. Rebentrost. *Quantum algorithms for supervised and unsupervised machine learning.* `https://arxiv.org/abs/1307.0411`, https://doi.org/10.48550/arXiv.1307.0411, 2013.

[30] V. Havlíček et al. *Supervised learning with quantum-enhanced feature spaces.* `https://www.nature.com/articles/s41586-019-0980-2`, https://doi.org/10.1038/s41586-019-0980-2, 2019.

[31] J. Bausch. *Recurrent Quantum Neural Networks.* `https://arxiv.org/abs/2006.14619`, 2020.

[32] E. Farhi, J. Goldstone, and S. Gutmann. A quantum approximate optimization algorithm. `https://arxiv.org/abs/1411.4028`, 2014.

[33] I. Cong, S. Choi, and M. D. Lukin. *Quantum convolutional neural networks.* `https://www.nature.com/articles/s41567-019-0648-8`, 2019.

[34] TensorFlow. *Quantum Convolutional Neural Network.* `https://www.tensorflow.org/quantum/tutorials/qcnn`, Retrieved 2023.

[35] S. Oh, J. Choi, and J. Kim. *A Tutorial on Quantum Convolutional Neural Networks (QCNN).* `https://arxiv.org/pdf/2009.09423.pdf`, 2020.

[36] L. Alchieri et al. *An introduction to quantum machine learning: from quantum logic to quantum deep learning.* `https://link.springer.com/article/10.1007/s42484-021-00056-8#Sec26`, 2021.

[37] K. Sengupta and P.R. Srivastava. *Quantum algorithm for quicker clinical prognostic analysis: an application and experimental study using CT scan images of COVID-19 patients.* `https://doi.org/10.1186/s12911-021-01588-6`, 2020.

[38] A. Kandala et al. *Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets.* `https://www.nature.com/articles/nature23879`, 2017.

[39] C. Zhao and X. Gao. *QDNN: deep neural networks with quantum layers.* `https://doi.org/10.1007/s42484-021-00046-w`, 2021.

[40] S. Abdel-Khalek et al. *Quantum neural network-based multilabel image classification in high-resolution unmanned aerial vehicle imagery.* `https://doi.org/10.1007/s00500-021-06460-3`, 2021.

[41] H. Robbins and S. Monro. *A Stochastic Approximation Method.* `https://www.jstor.org/stable/2236626`, 1951.

[42] D. P Kingma and J Ba. *Adam: A Method for Stochastic Optimization.* `https://arxiv.org/abs/1412.6980`, 2017.

[43] M. Schuld and F. Petruccione. *Supervised Learning with Quantum Computers.* Springer Cham, ISSN 2364-9054, DOI: https://doi.org/10.1007/978-3-319-96424-9, 2018.

[44] TensorFlow. *MNIST classification.* `https://www.tensorflow.org/quantum/tutorials/mnist`, Retrieved 2023.

[45] G. Philipp, D. Song, and J. G. Carbonell. *The exploding gradient problem demystified - definition, prevalence, impact, origin, tradeoffs, and solutions.* `https://arxiv.org/abs/1712.05577`, 2017.

[46] B. Rohrer. *How to Convert an RGB Image to Grayscale.* `https://e2eml.school/convert_rgb_to_grayscale.html`, 2019.

[47] Inc The MathWorks. rgb2gray. `https://www.mathworks.com/help/matlab/ref/rgb2gray.html`, 2023.

[48] J. A. Clark and contributors. Image module. `https://pillow.readthedocs.io/en/stable/reference/Image.html`, 2023.

[49] C. Poynton. Poynton's color faq. `https://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/POYNTON1/ColorFAQ.html`, 1997.

[50] International Telecommunicatino Union. Recommendation bt.601-7 (03/2011). `https://www.itu.int/rec/R-REC-BT.601-7-201103-I/en`, 2017.

[51] J. Chen et al. Automatic image cropping: A computational complexity study. `https://ieeexplore.ieee.org/document/7780430`, 2016.

[52] A. Krizhevsky. Learning multiple layers of features from tiny images. `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`, 2009.

[53] TensorFlow. tf.image.resize. `https://www.tensorflow.org/api_docs/python/tf/image/resize`, Retrieved 2023.

[54] M. Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. `https://www.tensorflow.org/`, `https://arxiv.org/pdf/2003.02989.pdf`, 2015. Software available from tensorflow.org.

[55] F. Chollet et al. *Keras*. `https://keras.io`, 2015. Software available from keras.io.

[56] M. Broughton et al. *TensorFlow Quantum: A Software Framework for Quantum Machine Learning*. `https://arxiv.org/abs/2003.02989`, 2020. Software available from https://www.tensorflow.org/quantum, part of the TensorFlow framework.

[57] Menkor Aviation. *Why are Airplanes White in Color?* `https://www.menkoraviation.com/en/questions/why-airplanes-are-painted-white/`, Retrieved 2023.

[58] US Army Air Defense Artillery School. *Visual Aircraft Recognition*. `https://irp.fas.org/doddir/army/fm3-01-80.pdf`, 2006.