Filip Igic

Quantum Autoencoders for Efficient Image Compression

B.Sc. (Hons) Computer Science

19th March 2021

Declaration

I, Filip Igic, certify that the material contained in this dissertation is my own work and does not contain unreferenced or unacknowledged material. I also warrant that the above statement applies to the implementation of the project and all associated documentation. Regarding the electronically submitted work, I consent to this being stored electronically and copied for assessment purposes, including the School's use of plagiarism detection systems in order to check the integrity of assessed work.

I agree to my dissertation being placed in the public domain, with my name explicitly included as the author of the work.

Signed:

Date:

Abstract

In this paper, we explore and test the limits of encoding classical images into quantum states using quantum autoencoders on different quantum image representation methods. The aim of the paper was to compress an image dataset that could be encoded using a near-perfect quantum simulator. Two different models are developed and run, and their results analysed and compared to one another.

Acknowledgement

I would like to take this opportunity to thank my supervisor, Dr Richard Jiang, for supervising me during my final-year project. His guidance and positivity have helped me with my work.

Filip Igic

March 2021

Contents

Chapter 1. Introduction

Both machine learning and quantum computing are increasingly popular fields of research and it would only be natural to try and fuse the two together. While it is still unclear how much of an advantage machine learning will have with quantum computing due to its limited computational resources, it still gives us a new way to approach machine learning.

1.1 Aims

Machine learning on classical computers has proven to be incredibly useful with its wide variety of algorithms, from being able to classify data to being able to generate new appropriate data based on previous training rounds.

However, these algorithms are computationally expensive and to aid that, quantum computers show a lot of promise due to their exponentially better performance in some algorithms such as Grover's Algorithm [1], which can find an item in $\sqrt{N}$ steps rather than the average $\frac{N}{2}$ steps it takes a classical computer.

Although quantum computers show a lot of promise, they are very limited in resources especially in the scalability of qubits. This is where a quantum autoencoder would be ideal as it could compress several qubit states into one, which would in return save a lot of resources.

The main aim of this paper was to discuss the current implementations of quantum autoencoders and apply them to different quantum image representations, exploring and testing the limits of fitting larger images within a realistic number of qubits.

An additional aim was to run the quantum autoencoder on an actual quantum computer and analyse the results.

1.2 Report Overview

Background Research: This chapter will discuss definitions within quantum computing as well as contain a literature review of all the relevant papers within this project.

Design: This chapter will explain all the design decisions that went into the quantum models when planning out the project.

Implementation: This will be an overview of the autoencoder implementation.

Testing and Evaluation: In this chapter I will analyse and discuss the results.

Conclusions: I revisit my aims and determine if they were achieved and talk about my personal reflection throughout this project.

Chapter 2. Background Research

2.1 Quantum Computing and Quantum Information

In this subchapter, I will be reviewing the definitions in Quantum Computing and Quantum Information [2] that will be used in this paper:

- Dirac (bra-ket) notation is the standard notation used for states in quantum mechanics [2]. A ket is used to denote a vector and looks like "$|v>$". A bra denotes the conjugate transpose of a ket and looks like "$<v|$"
  e.g.

$$|1> = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$<1| \; = [0 \; 1]$$

- The quantum bit or qubit is much like a classical bit as it can be in two possible states $|0>$ and $|1>$ (also known as the computational bases). However, the qubit can also be in a superposition, which is when a state is formed using linear combinations: $|\psi> = \alpha|0> + \beta|1>$ where $\alpha, \beta \in \mathbb{C}$.

- The state vector is a vector in a complex vector space which we use to describe the state of our system. The state vector is also normalised, this is because we want the probabilities that are related to the amplitudes to add up to 1. More specifically, the magnitude of the vector needs to be 1 i.e., $\langle\psi|\psi\rangle = 1$.
  So, if a state is formed as $|\psi> = \alpha|0> + \beta|1>$ then $\sqrt{|\alpha|^2 + |\beta|^2} = 1$.

- The Bloch sphere is a useful way of visualising a single qubit's state. We can map any single qubit state on the Bloch sphere using $|q> = \cos\frac{\theta}{2}|0> + e^{i\phi}\sin\frac{\theta}{2}|1>$ where $\theta, \phi \in \mathbb{R}$.

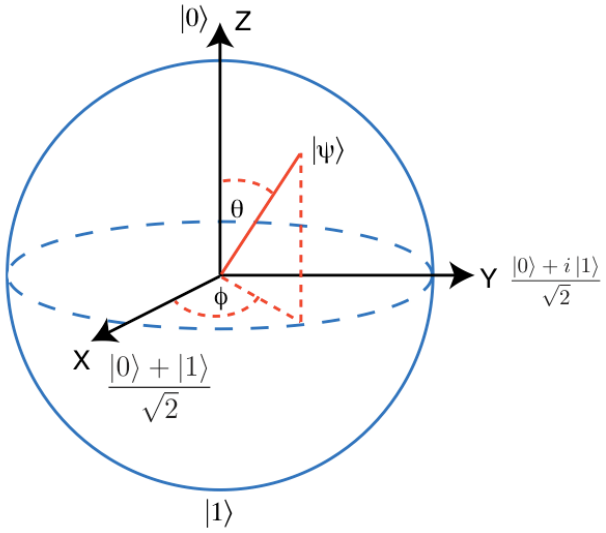


Figure 2.1 Bloch Spere Ref [8]

- Single qubit gates are unitary transforms applied to single qubit states. The following are the most frequently used:

  Hadamard $\frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$

  Pauli-X $\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$

  Pauli-Y $\begin{bmatrix}0 & -i\\ i & 0\end{bmatrix}$

  Pauli-Z $\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$

  Identity $\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$

  Rϕ-gate $\begin{bmatrix}1 & 0\\ 0 & e^{i\phi}\end{bmatrix}$

- Multiple qubit gates are unitary transforms that are applied to the collective state of multiple qubits. The following are the most frequently come across:

  Controlled-NOT $\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{bmatrix}$

  SWAP $\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$

  Controlled-Z $\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & -1\end{bmatrix}$

  Controlled phase $\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & i\end{bmatrix}$

  Toffoli $\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1\\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0\end{bmatrix}$

- A measurement is a projection of a quantum state onto $|0>$ and $|1>$.

2.2 Quantum Autoencoders for Efficient Compression of Quantum Data

In the work of Romero et al. they demonstrate how their quantum autoencoder can compress ground states of the Hubbard model and molecular Hamiltonians, on a quantum simulator.

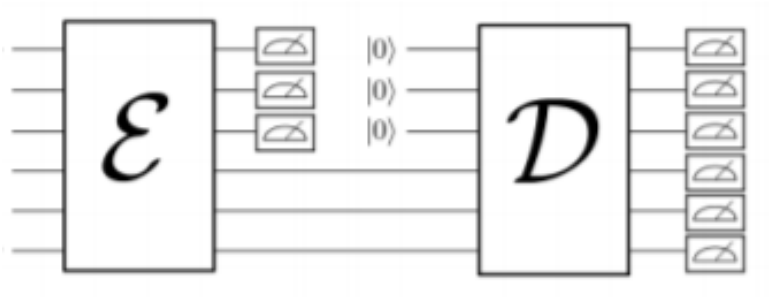


Figure 2.2 Circuit implementation of a 6-3-6 quantum autoencoder (cited from [3]) where the unitary quantum circuit ε encodes a 6-qubit input into a 3-qubit state which the unitary quantum circuit D then decodes by attempting to reconstruct the 6-qubit input onto the 6-qubit output.

To make a true autoencoder they must first discard some of the input qubits after the encoding ε and reset them to a state such as $|0>$ which are then used in the decoding part of the autoencoder D where $D = \varepsilon^{\dagger}$. E.g., the first three qubits in figure 2.1.

Next, is to ensure the autoencoder's unitaries are preserving the maximum amount of information in the latent space from the input qubits $|\psi_i>$. The authors propose measuring the fidelity of the initial input qubits $|\psi_i>$ and the output ${\rho_i}^{out}$ where the fidelity is defined as $F(|\psi_i>, {\rho_i}^{out}) = <\psi_i|{\rho_i}^{out}|\psi_i>$. A successful autoencoding would then produce the following $F(|\psi_i>, {\rho_i}^{out}) \approx 1$ for all input states.

Using the average fidelity acquired from the autoencoders training period, a cost function can be defined using classical learning methods to find the optimum unitary which maximises the average fidelity. In other words, which unitary can best encode, and decode qubit states.

The authors then go on to explain two different approaches for the cost function which is best explained along with the following quantum circuit.

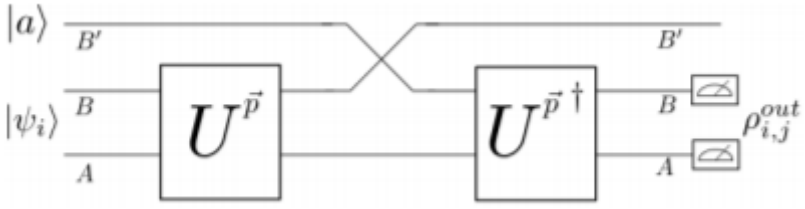


Figure 2.3 A quantum autoencoder circuit where its goal is to find some set of parameters $\vec{\rho}$ such that the averaged fidelity is maximised (cited from [3]). A and B are subsystems where B qubits are reset.

The first cost function is defined as

$$C_1(\vec{p}) \;=\; \sum_i p_i \cdot F(|\psi_i>, \rho_{i,\vec{p}}^{out})$$

where

$$\rho_{i,\vec{p}}^{out} \;=\; (U^{\vec{p}})_{AB'}^{\dagger} Tr_B[U_{AB}^{\vec{p}}[\psi_{iAB} \otimes a_{B'}](U_{AB}^{\vec{p}})^{\dagger}](U^{\vec{p}})_{AB'}$$

which measures the fidelity between the input qubits and output qubits.

However, the paper goes on to describe an alternative cost function

$$C_2(\vec{p}) \;=\; \sum_i p_i \cdot F(Tr_A[U^{\vec{p}}|\psi_i><\psi_i|_{AB}(U^{\vec{p}})^{\dagger}], |a>_B)$$

which trains only on the trash state (the measurements taken on the qubits that are reset) by measuring the trash state fidelity. One benefit of this approach is that it would be easier to prepare copies of a fixed reference state to do a SWAP test on the trash state, rather than having to do a SWAP test on the entire output state.

For the encoder and decoder circuits, two different models were tested. The first model was a network of all possible two-qubit gates and the second model was comprised of single qubit rotations at the beginning and at the end of the circuit as well as all possible controlled single qubit rotations in between the single qubit rotations.

2.3 Quantum Image Representations

2.3.1 FRQI

In this paper, a Flexible Representation of Quantum Images (FRQI) [4] is proposed, which encodes information about colours and their position in the form of a normalised state in a quantum computer.

The authors also simulate experiments on FRQI by storing and retrieving data, as well as apply quantum Fourier transform (QFT) as a processing operation to detect a line in binary images.

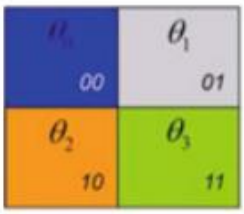
To represent the image information as quantum state, the following is used

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

where

$$\theta_i \;\in\; [0,\frac{\pi}{2}], i \;=\; 0,1,\ldots,2^{2n}-1$$

In this expression, $|i>$ are the $2^{2n} - D$ quantum states such as |00>, |01> etc. and θ is the vector of angles which are used to encode the colours as $(cos\,\theta_i|0> + sin\theta_i|1>)$.



$$|I\rangle = \frac{1}{2}\left[(\cos\theta_0|0\rangle + \sin\theta_0|1\rangle)\otimes|00\rangle + (\cos\theta_1|0\rangle + \sin\theta_1|1\rangle)\otimes|01\rangle + \right.$$
$$\left. + (\cos\theta_2|0\rangle + \sin\theta_2|1\rangle)\otimes|10\rangle + (\cos\theta_3|0\rangle + \sin\theta_3|1\rangle)\otimes|11\rangle\right]$$

Figure 2.4 This figure shows a simple 2x2 image and its FRQI state (cited from [4]).

More information on FRQI is available in chapter 3.1.

2.3.2 NEQR

The paper NEQR, a novel enhanced quantum representation for digital images [5] was proposed as an improvement over the previously reviewed FRQI [4].

The following advantages have been identified by the authors by comparing their solution to the FRQI model:

- A quadratic decrease in time complexity when preparing the image. When preparing a $2^n x 2^n$ image with FRQI the process costs $O(2^{4n})$.
- 1.5x higher compression ratio when compressing the image by minimisation of Boolean expressions.
- Images are retrieved accurately rather than probabilistically. In other words, FRQI stores pixel information as the probability amplitude of a single qubit meaning accurate image retrieval is impossible through finite quantum measurements.
- Larger amount of image operations can be performed conveniently when based on NEQR, particularly certain complex colour operations such as partial colour operations.

As opposed to FRQI which uses a single qubit to store the grey-scale information of a pixel, NEQR stores grey-scale information and position information into two entangled qubit sequences.

The following expression can be used to represent a $2^n x 2^n$ image

$$|I> = \frac{1}{2^n}\sum_{Y=0}^{2^n-1}\sum_{X=0}^{2^n-1}|f(Y,X)>|YX> = \frac{1}{2^n}\sum_{Y=0}^{2^n-1}\sum_{X=0}^{2^n-1}|\otimes_{i=0}^{q-1}>|C_{YX}^{i}>|YX>$$

where C is a binary representation of the grey-scale value

$$f(Y,X) = C_{YX}^{0}, C_{YX}^{1}, \ldots, C_{YX}^{q-1} \in [0,1], f(Y,X) \in [0, 2^{q-1}]$$

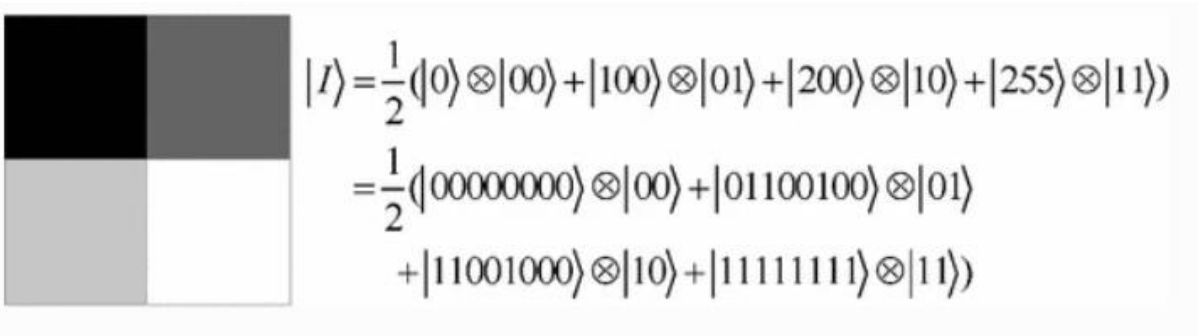


Figure 2.5 This figure shows a simple 2x2 image and its NEQR state (cited from [5]).

More information on NEQR is available in chapter 3.2.

Chapter 3. Design

3.1 FRQI Model

The FRQI Model will be a quantum circuit implementation based on the work Romero et al [3] and Le et al [4]. This subchapter will be broken down into a section describing how the FRQI will be made based on the FRQI paper, and a section on how the autoencoder will be designed based on the autoencoder paper.

3.1.1 Creating the FRQI State

To create the FRQI State $|I(\theta)>$ from the initialised state $|0>^{\otimes 2n+1}$, two steps are required.

The first step is to put all qubits into superposition except for one which will be used to encode the colour information onto a pixel giving us the state

$$|H> = \frac{1}{2^n}|0> \otimes \sum_{i=0}^{2^{2n}-1} |i>$$

For the second step, controlled rotation transforms are applied to change the state from $|H>$ to $|I(\theta)>$. As shown in the expression below demonstrated by the paper [4].

$$\mathcal{R}\,|H\rangle = \left(\prod_{i=0}^{2^{2n}-1} R_i\right)|H\rangle = |I(\theta)\rangle$$

$$R_i = \left(I \otimes \sum_{j=0, j\neq i}^{2^{2n}-1} |j\rangle\langle j|\right) + R_y(2\theta_i) \otimes |i\rangle\langle i|$$

$$R_y(2\theta_i) = \begin{pmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{pmatrix}$$

However, this approach requires an increasingly large number of simple gates in proportion to the number of controlled-rotation gates which means that implementing some sort of image compression before the autoencoder is very important.

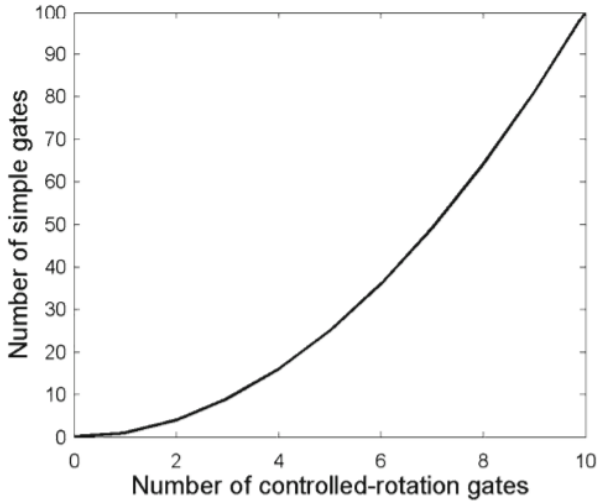
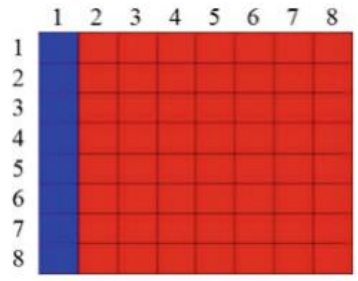


Figure 3.1 This figure shows the relation between the number of simple gates and the number of controlled-rotation gates (cited from [4]).

To compress the image the authors propose grouping the pixels with the same intensity (angle) together under one controlled-rotation gate by minimising the Boolean expression made by transforming the binary strings of the pixel positions to Boolean minterms such that $|0> = \bar{x}$ and $|1> = x$.

Consider the following 8x8 image where the first column is blue, and the rest are red. Therefore, the positions $|0>, |8>, |16>, |24>, |32>, |40>, |48>, |56>$ are all blue and can be encoded using a single conditional rotation.



a)

Position	Binary String	Boolean Expression
$\|0>$	$\|000000>$	$\overline{x_5}\overline{x_4}\overline{x_3}\overline{x_2}\overline{x_1}\overline{x_0}$
$\|8>$	$\|001000>$	$\overline{x_5}\overline{x_4}\overline{x_3}x_2\overline{x_1}\overline{x_0}$
$\|16>$	$\|010000>$	$\overline{x_5}x_4\overline{x_3}\overline{x_2}\overline{x_1}\overline{x_0}$
$\|24>$	$\|011000>$	$\overline{x_5}x_4x_3\overline{x_2}\overline{x_1}\overline{x_0}$
$\|32>$	$\|100000>$	$x_5\overline{x_4}\overline{x_3}\overline{x_2}\overline{x_1}\overline{x_0}$
$\|40>$	$\|101000>$	$x_5\overline{x_4}x_3\overline{x_2}\overline{x_1}\overline{x_0}$
$\|48>$	$\|110000>$	$x_5x_4\overline{x_3}\overline{x_2}\overline{x_1}\overline{x_0}$
$\|56>$	$\|111000>$	$x_5x_4x_3\overline{x_2}\overline{x_1}\overline{x_0}$

b)

$$e = \overline{x_5}\overline{x_4}\overline{x_3}\overline{x_2}\overline{x_1}\overline{x_0} + \overline{x_5}\overline{x_4}x_3\overline{x_2}\overline{x_1}\overline{x_0} + \cdots + x_5x_4x_3\overline{x_2}\overline{x_1}\overline{x_0}$$

$$e = \overline{x_2}\overline{x_1}\overline{x_0}$$

c)

Figure 3.2 Quantum Image Compression a) 8x8 red/blue image (cited from [4]) b) table of binary strings and Boolean minterms c) Boolean expression and the minimised Boolean expression.

3.1.2 Quantum Autoencoder

Once the FRQI state is created, the next logical step would be to encode the data with an autoencoder. To do that, I will be using the proposal by Romero et al [3] as reviewed in the background research chapter but instead of implementing a SWAP test to measure the fidelity between the reference and trash states for the cost function, I will be applying an inverse FRQI state preparation at the end of the autoencoder and measuring the probability of measuring $|0>^{\otimes q}$. I would then average over the losses in training, negate the values to cast as a minimisation problem as proposed by Sim et al [6].

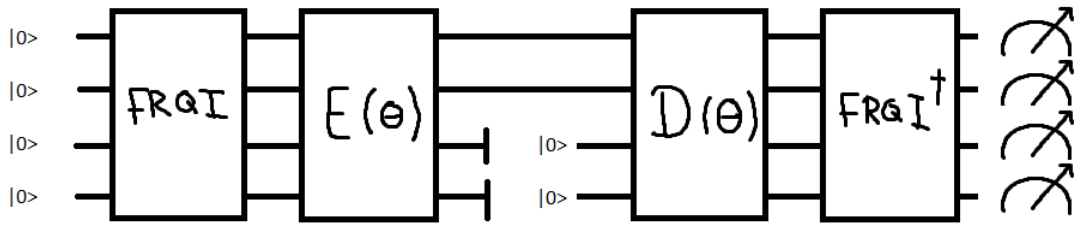


Figure 3.3 4-2-4 diagram showing how the FRQI state is encoded and sent through a 2-qubit latent space and then decoded and measured.

3.2 NEQR Model

3.1.1 Creating the NEQR State

The first step in creating the NEQR state is applying a unitary which applies an identity gate to all qubits and a Hadamard gate to the qubits representing the pixel position. In the paper, the unitary is represented as

$$U_1 = I^{\otimes q} \otimes H^{\otimes 2n}$$

Applying $U_1$to $|0>^{\otimes 2n+q}$ gives me the intermediary state which allows me to then encode the grey scale values to the pixel positions.

$$U_1(|0>^{\otimes 2n+q}) = |\psi>_1$$

For step 2, in which I prepare the quantum image, I must first divide the step into 2^{2n} sub-operations as all the pixels are in a superposition of a qubit sequence. For every pixel, the sub-operation is represented as

$$U_{YX} = (I \otimes \sum_{j=0}^{2^n-1} \sum_{i=0, ji \neq YX}^{2^n-1} |ji><ji|) + \Omega_{YX} \otimes |YX><YX|$$

where Ω_{YX} which is consisted of q quantum oracles $\Omega_{YX}^i: |0> \rightarrow |0 \otimes C_{YX}^i>$, is the operation that sets the pixel value

$$\Omega_{YX} = \bigotimes_{i=0}^{q-1} \Omega_{YX}^i$$

therefore the quantum transformation $\Omega_{YX}$to set the grey-scale for the pixel is

$$\Omega_{YX}|0>^{\otimes q} = \bigotimes_{i=0}^{q-1} (\Omega_{YX}^{i}|0>) = \bigotimes_{i=0}^{q-1} |0 \otimes C_{YX}^{i}>) = |f(Y,X)>$$

As U_{YX} is a sub-operation of setting the greyscale for a single pixel, step 2 is represented as unitary U_2which is the sum of all sub-operations

$$U_2 = \prod_{Y=0}^{2^n-1} \prod_{X=0}^{2^n-1} U_{YX}$$

After applying the unitary $U_2$to the state $|\psi>_1$, the entire preparation is done, and I have the quantum image in the state $|\psi>_2$ as described in the paper [5].

3.1.2 Quantum Autoencoder

For the autoencoder I will be using the same one discussed in the FRQI model. The only difference will be that an NEQR state will be created and inversed as opposed to the FRQI one.

To read more on the design of the quantum autoencoder, refer to chapter 2.2.

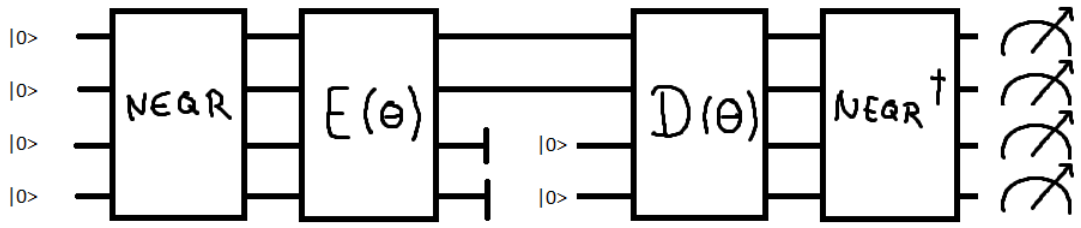


Figure 3.4 4-2-4 diagram showing how the FRQI state is encoded and sent through a 2-qubit latent space and then decoded and measured.

Chapter 4. Implementation

Tools

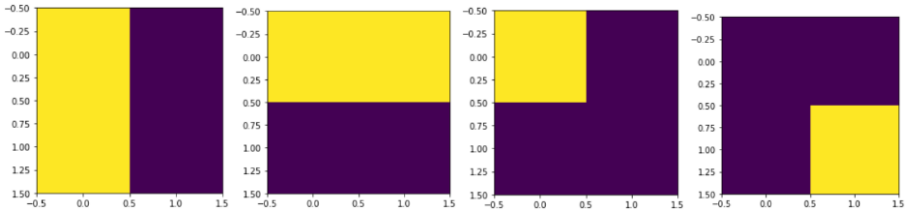
The tools I have used to implement my research on the application of quantum encoders on quantum images are the following:

- Python [7]: This was the programming language of choice as it has all the necessary frameworks I will be using.
- Qiskit [8]: Qiskit is an open-source python framework used to simulate quantum computers as well as work with actual quantum computers. This is what I used to implement all my circuits, manipulate qubits and run tests.
- SciPy [9]: is python based open-source software which contains multiple packages that will be used in the development: NumPy, for handling the arrays; SciPy, contains the optimisation code for the cost function; Matplotlib, which contains the code to visualise data and quantum circuits.
- JupyterLab [10]: JupyterLab was my IDE of choice. It is ideal for this project due to its ability of carrying out tests on quantum circuits and documenting my progress if needed.

Dataset

For the dataset I used a set of 2x2 images with some having minimal value and some more complex images having a varying value in every pixel. There are a couple reasons for choosing small images:

- I would have liked to run both the FRQI and NEQR model on an actual quantum computer and given the limited capabilities of a quantum computer and the amount of noise it produces this will be easier to analyse.
- I wanted to keep the complexity of the project quite low. Increasing the number of pixels on a classical image increases the complexity of the circuit which would be quite hard to debug given the time constraint I had on this project. More on increasing the size and analysis is discussed in chapter 6.



a) b) c) d)

Figure 4.1 This figure shows a portion of the dataset as an example a) Left vertical line. b) Top horizontal line c) Top left pixel on d) Bottom right pixel on

3.1 FRQI Model

The first step was to create the circuit that can prepare the FRQI state. To do this, I initialised three qubits and put the first two in superposition using the Hadamard gate. In the figure below you will see vertical dotted lines or barriers, these are there purely for visual purposes to help distinguish between the pixels. I then created a 2x2 template to encode data into the

FRQI state using a series of conditional-rotation gates and NOT gates to set the pixel position. Although the FRQI is capable of any grey-scale value range given a corresponding angle, I decided to use two angles to encode when a pixel was on or off, with the angles being pi/2 and 0, respectively. Lastly, I measured all three qubits and mapped them to three classical bits.

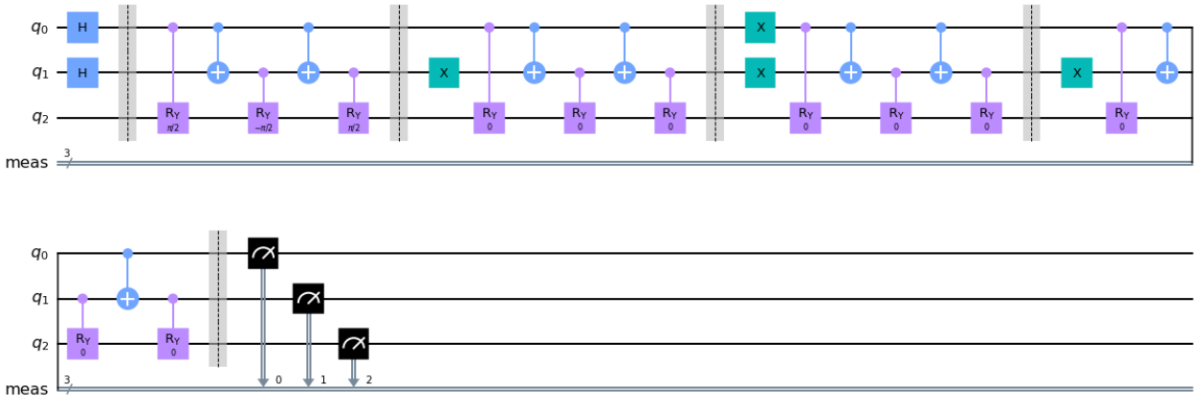
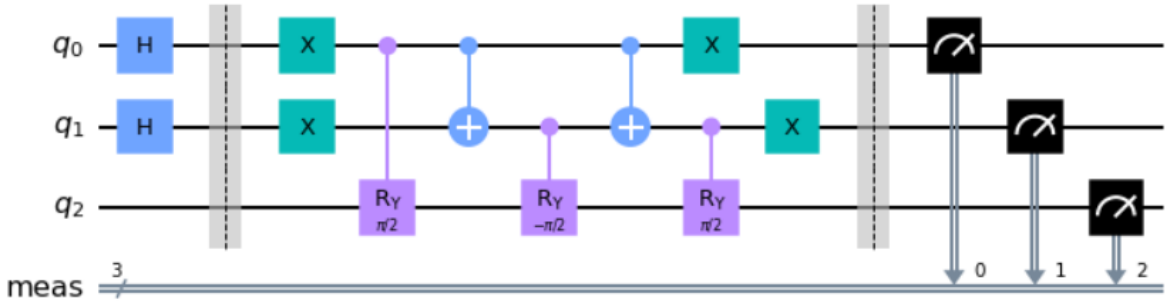
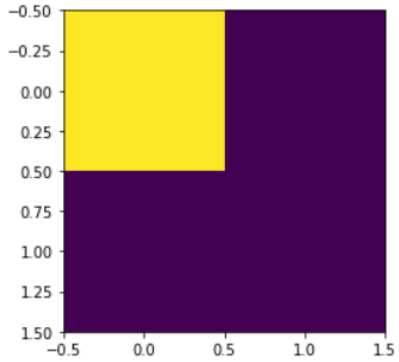


Figure 4.1 FRQI state preparation circuit for a 2x2 image before compression.

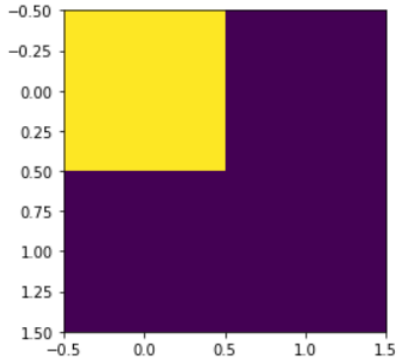
The next step was to compress the state preparation circuit. As the datasets were very small images, the Boolean minimisation method presented in the FRQI paper [4] was not needed such that I take all the Boolean minterms and minimise them. Instead, I just encoded the data with the pixel on, and when retrieving the image, I assumed all the data that was not found by measuring was just a pixel turned off.



a)



b)



c)

Figure 4.2 a) Shows the compressed version of the circuit in figure 4.1. b) is the image retrieved from the initial circuit c) the image retrieved from the compressed circuit

Once I had the image compressed by its width, I needed to implement the autoencoder to compress it by the number of qubits. I implemented a 3-1-3 autoencoder for this FRQI model.

In the paper by Romero et al [3], they demonstrate two different models. The first consisting of all the possible two-qubit gates among the qubits, and the second model, consisting of all possible controlled single-qubit rotations as well as a single qubit rotation on all qubits at the start of the encoder and at the end of the encoder.

There were many ways I could have implemented the unitary "encoder" but to keep things simple and relatively computationally inexpensive I made a circuit that consisted of two rotational gates and two control-nots, inspired by the approach from Sim et al [6].

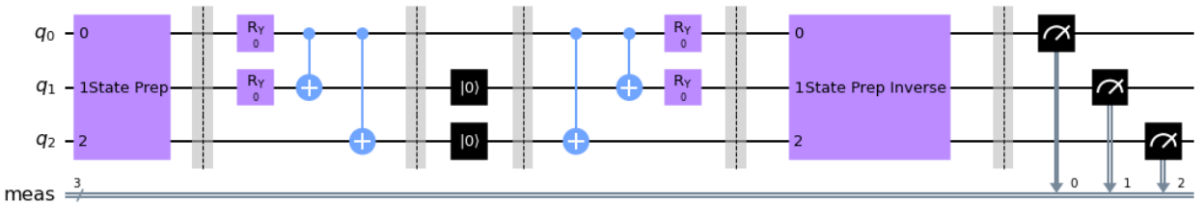


Figure 4.3 The full quantum circuit for training FRQI states.

As shown in Figure 4.3, the model starts by preparing the state. In this case, it was an image prepared as an FRQI as shown earlier in the chapter.

The next section encoded all the data into q_0. The figure shows rotations by the angle 0 but this is purely as a demonstration. When running the model, the zeros are replaced with the parameters calculated by the cost function.

Qubits q_1 and q_2 are then reset and the data travels along the latent space q_0.

In the next step, the inverse of the encoder was applied to reconstruct the FRQI from q_0. When testing, I measured here and retrieved the image. However, when training, I would apply the inverse of the state preparation unitary and then measure.

Once I have applied the inverse of the state preparation unitary and measured, I would expect the results to be "000" if the reconstructed state were the same as state created at the start of the circuit.

Using the probability of measuring "000" I negated that value and treated it as a minimisation problem. When implementing the cost function, I used the Constrained Optimization by Linear Approximation (COBYLA) method within the SciPy package. Using this method, I implemented the quantum circuit as a callable function which takes the parameters given by the COBYLA method. Apart from passing the function to the COBYLA method, I also passed in my initial guess which was typically the value of π .I also passed a constraint of 2π as an argument to the COBYLA method.

When testing the autoencoder using arbitrary states made by single-qubit rotational gates, I was able to achieve results of >95% preservation of information. All the code can be found on my Github page [12] and the results can be replicated.

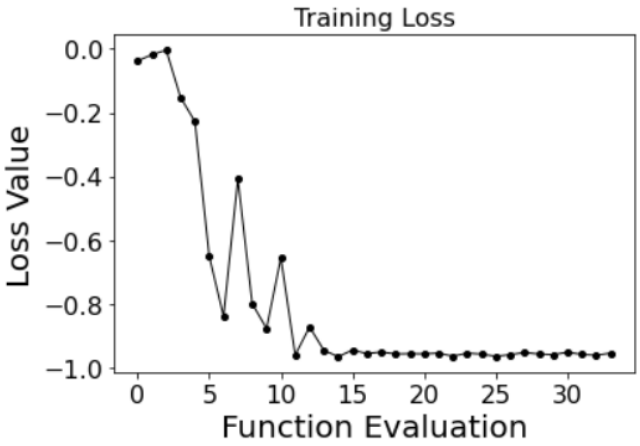


Figure 4.4 This figure shows the results of the cost function. Optimal parameters are when loss value is at -1.

When testing the model with the optimal parameters I retrieved the image by using the probability of a pixel having a value of 1. This is because the values are encoded in the amplitude as explained previously.

3.2 NEQR Model

When implementing the NEQR model, I started by preparing the NEQR state as demonstrated in the paper [5]. Similarly like with the FEQR model, I initialised 3 qubits to $|0>$ and put the first two qubits for determining the pixel position, into superposition. I used a series of NOT gates to set the pixel position and used a Toffoli gate to set the pixel value to on.

The qubits can be expanded to feature more pixel positions or to encode more information into the pixel values already provided which will be demonstrated in chapter 5.

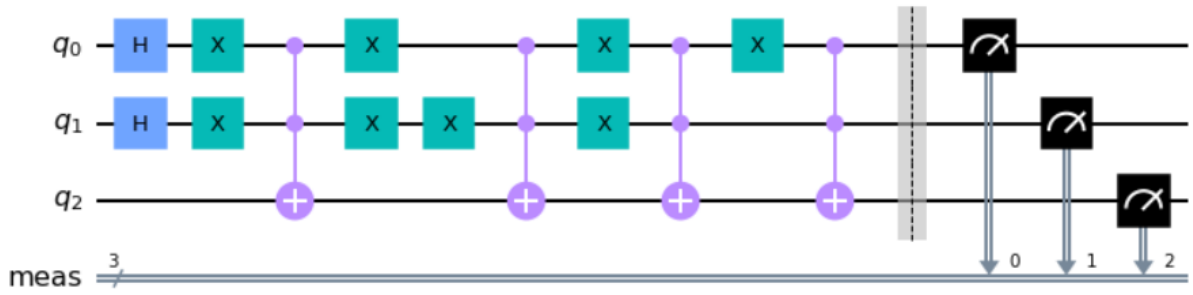
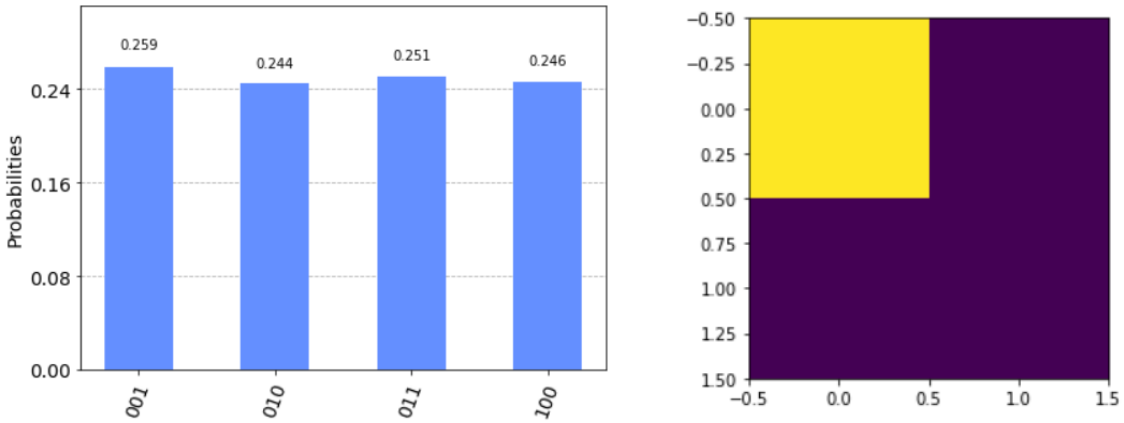


Figure 4.5 NEQR state preparation circuit encoding a 2x2 image with all pixels set to on.

Much like the FRQI implementation, when creating a 2x2 image there is no benefit to compression as proposed in the paper [5]. In this case, either the pixel is on, in which you must add the appropriate NOT and Toffoli gate or the pixel is off in which you don't have to add any gates.

The autoencoder works in the exact same way as the FRQI except it will be expanded on to more qubits in chapter 5 when testing a larger range of pixel values on the NEQR model.



a) b)

Figure 4.6 a) the probability count of each pixel after 4096 shots. b) the image retrieved from the counts

When measuring, Qiskit [8] returns the measurements as a dictionary.

To retrieve the image, I first looped through the dictionary and grouped the keys with the same last two digits (pixel position). I then chose the key with the highest value from each group and used that as data to build the image.

Although it might not be apparent from the figures between the FRQI and the NEQR models, as I expand into a larger range of pixel values in chapter 5, it will be apparent how fundamentally different these two image representations are in practice.

Chapter 5. Testing and Evaluation

Images with A Single Pixel Value

The first test was to auto-encode a 2x2 image with pixels containing a value of either 1 or 0.

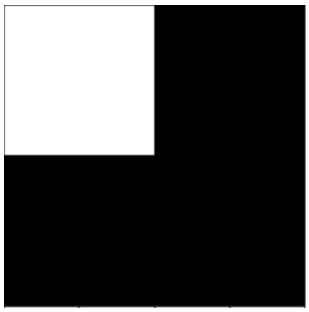
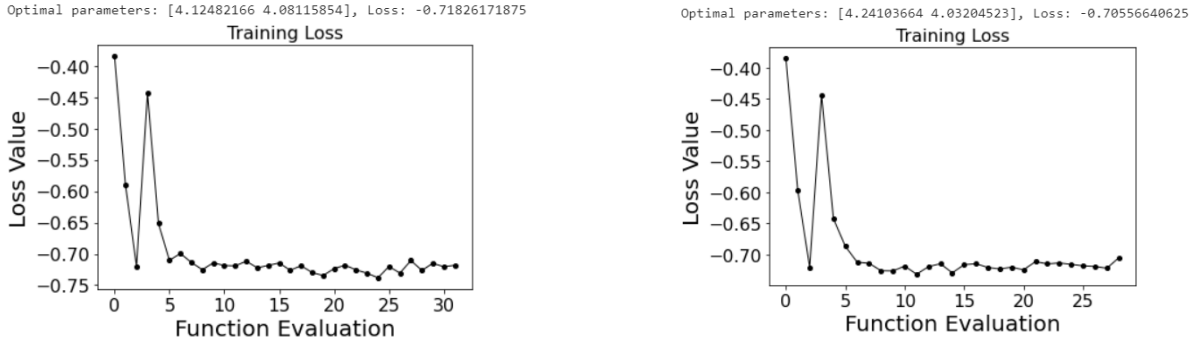


Figure 5.1 The first image in the dataset.

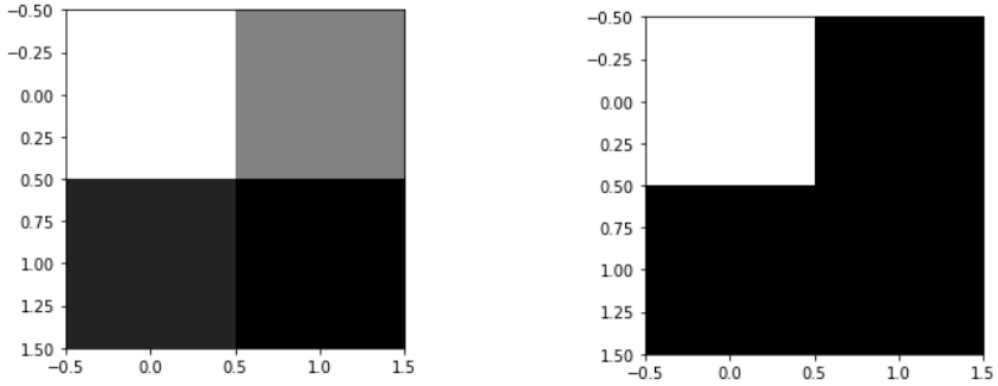
The first image contained a single pixel in the top left with a value of 1 which was tested on both models.



a) b)

Figure 5.2 a) Training loss data for the FRQI model b) Training loss data for the NEQR model

Although the training data from both models show similar results, with the FRQI model performing slightly better. The way that the FRQI image is encoded in the amplitude puts it at a disadvantage when retrieving the image as show in the figure below.



a) b)

Figure 5.3 a) Image retrieved from the FRQI model b) Image retrieved from the NEQR model.

By adding an additional rotational gate to the autoencoder in the FRQI model and passing three parameters instead of two, I was able to decrease the loss value by roughly 10%.

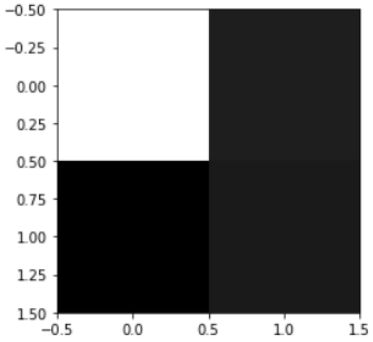


Figure 5.4 Image retrieved from the FRQI model after an increase in rotational gates.

The second image with a single value that was tested was an image with a vertical line on the right-hand side as shown below.

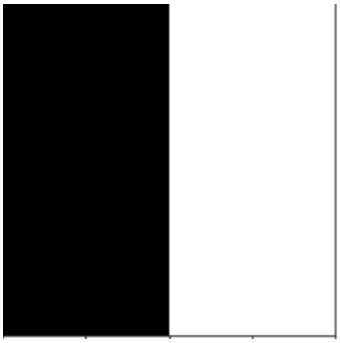
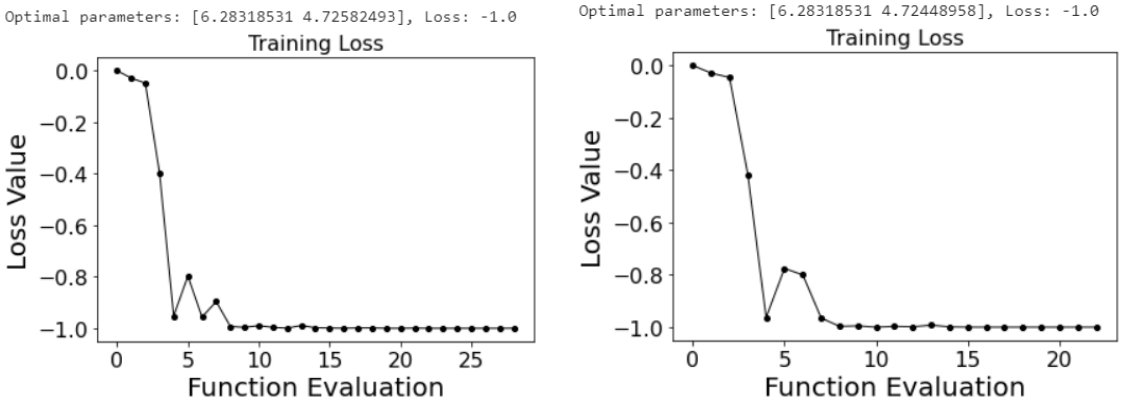


Figure 5.5 The second image in the dataset.

In this case, both models found an optimal solution which resulted in a loss value of -1 meaning there was a 100% recovery of data from the compression.



a) b)

Figure 5.6 a) Training loss data for the FRQI model b) Training loss data for the NEQR model

Not only did both models find a solution but both the optimal parameters are nearly exactly the same as one another. Testing other varitations of having a one pixel turned on and having

half of the pixels on give the same results. All results can be replicated using the code found on Github [12] by implementing the appropriate state prep circuit into the function “nerq” or “frqi”.

Images with Multiple Pixel Values

In this particular set of tests, I increased the value range a pixel can have from 0 to 1 to a range of 0 to 4. This gave the image more depth and exploit the benefits one quantum image representation can have over the other.

Before I started the testing, I needed to make adjustments to the NEQR model. To introduce it to a higher range of pixel values I had to increase the number of qubits. As I only increased the value range to four, I only had to initialise one more qubit than I had before as this gave me two qubits for encoding the data, giving me four different possible binary outcomes: “00”, “01”, “10” and “11”.

The autoencoder also had to be adjusted because of the extra qubit, so it could encode the data into the latent space. This made the autoencoder a 4-1-4 autoencoder.

This already shows the potential advantage that FRQI could have over NEQR if qubits were not so fragile and could preserve their state perfectly. For example, if I wanted to create an image with a pixel grey-scale range of 255 I would need 8 qubits for encoding the data in addition to the qubits holding the pixel position with the NEQR model as opposed to only the single qubit for encoding data with the FRQI model. However, the FRQI model isn’t perfect, preserving smaller angle adjustments is almost impossible and could lead to poor results as shown in the following tests.

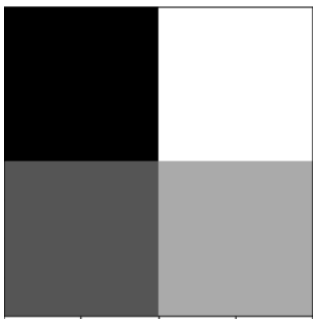
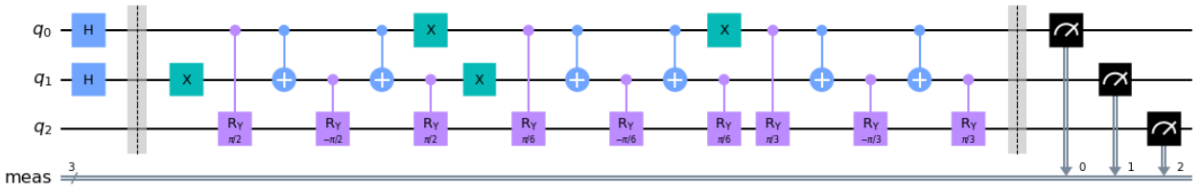
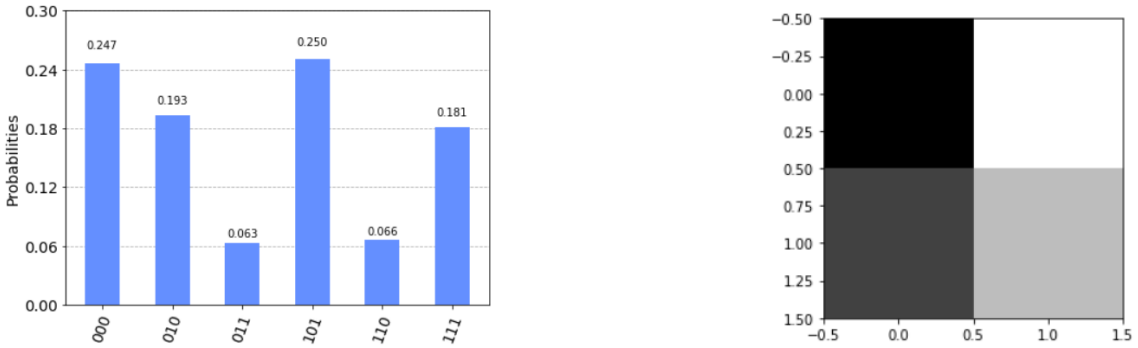


Figure 5.7 The image being tested. The pixel values are 0, 4, 1, 2, respectively.

To make sure that the image representations are encoding the data properly, I ran the circuits without the autoencoder. Figure 5.8 are the results.

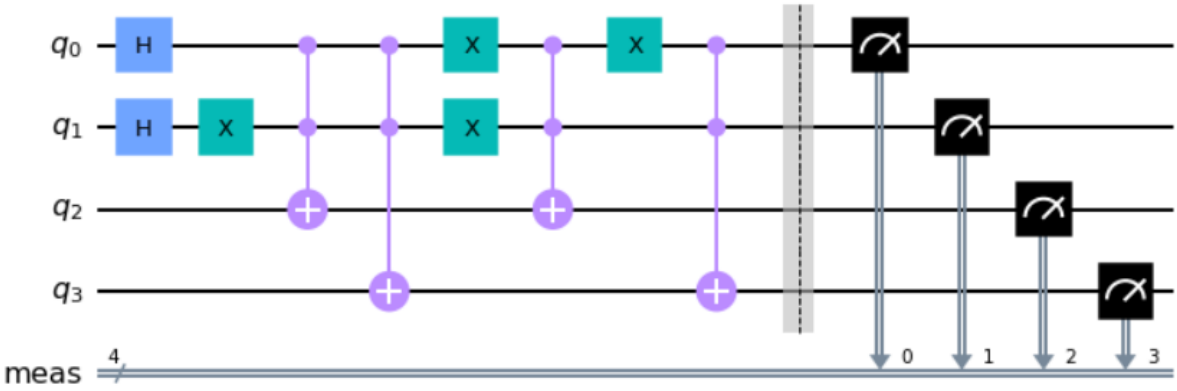


a)

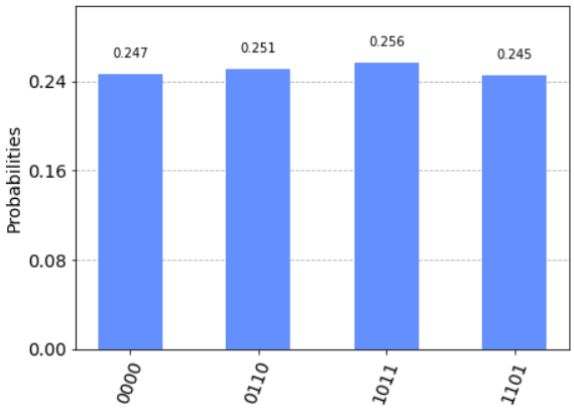


b)

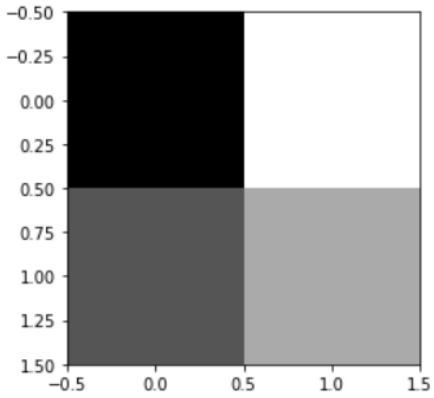
c)



d)



e)



f)

Figure 5.8 a) FRQI state preparation circuit b) Measurement counts c) Image retrieval d) NEQR state preparation circuit e) Measurement counts f) Image retrieval.

Once I ensured the results were valid, the autoencoder was introduced to the circuit.

After running the FRQI model multiple times with different initial guesses I was unable to retrieve an image at all.

For the NEQR model, the best outcome I was able to achieve with the simple autoencoder was a loss value of -0.24560546875. This resulted in an image with the correct top two pixel.

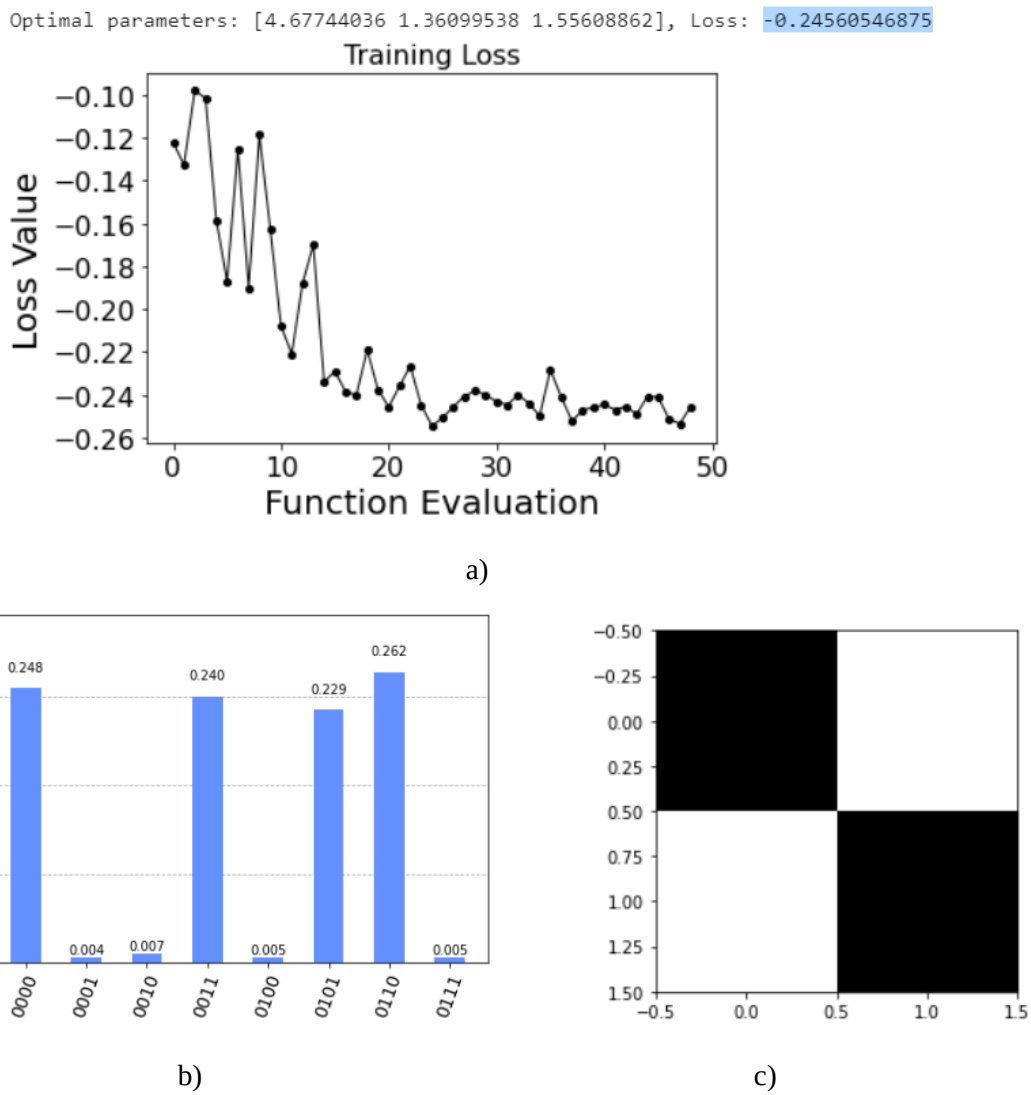


Figure 5.9 a) Training loss data b) Qubit measurement results c) NEQR image retrieved.

The testing shows that increasing the number of rotational gates, even by one, can cause a great improvement in encoding data. Both the NEQR model and the FRQI model both benefit differently from an increase in rotational gates.

When applying additional rotational gates to the FRQI model, the measurements show an increase in probability of retrieving the correct image data, and when applying additional rotational gates to the NEQR model, it helps encode the information when an increased number of qubits are required for storing larger pixel values.

Chapter 6. Conclusions

6.2 Review of Aims

The main aim of this project was to apply quantum autoencoder models to different quantum image representations and analyse the results.

In this project, I have applied a quantum autoencoder capable of compressing two different quantum image representations. Both models can compress a classical 2x2 image pixel where in some cases there is no loss of information.

Given the amount of reiterations the models had to go through for computing the cost function, I was unable to use one of IBM's computers to execute the jobs. Had I more time, I would have used the optimal parameters I got from my simulation and ran a test with those parameters as a proof of concept.

6.1 Personal Reflection

I was expecting to use my final year project as an opportunity to use all the knowledge I have acquired during my time as an undergraduate and apply it to my project. However, this was not the case.

My first hurdle was linear algebra. Linear algebra is essentially the language of quantum computing. In order for me to understand how quantum computer's function, it was crucial for me to develop a good understanding of vectors, matrices and everything else that comes along in quantum computing. This took a large amount of time for me to get a grasp of and took out a large amount of time on the project.

The next issue was developing an intuition for the laws of quantum mechanics. Unlike a standard bit that can either be on or off, a qubit has several properties that a classical bit does not have, such as superposition, entanglement, decoherence. It took me a long time to imagine a qubit in a superposition and how a measurement affects it. More importantly, using these properties to my advantage was something that I needed to also learn. I was so comfortable with classical computers after studying them for so long, that it was hard to break out of that mindset.

Lastly, I had to learn how to use frameworks to simulate quantum computers and create functioning quantum circuits that would achieve the results I was after. I learned how to create my own quantum circuits and custom gates that can then be compiled to quantum assembly and run on an actual quantum computer.

Throughout this project I have learned a lot. Not only on quantum computing itself, but also how to read research papers and apply the most current innovative ideas into my own projects. I have also learned that given enough determination and time I am still able to explore new fields and use a computer in such a way which can benefit that field.

6.3 Future Work

For future work, I would like to explore more image representations such as quantum image representations that can store rectangular images as opposed to just square images as well as build more complex models of quantum autoencoders that can preserve the data better within

a latent space, expanding into larger image datasets that contain enough detail for facial recognition.

I would also like to explore using the compressed state from the models, as a mean of efficient encryption. Where the autoencoders encoder and decoder unitaries are treated as an encryption/decryption function and the parameters $\vec{\theta}$ are treated as a key.

6.2 Overall Conclusions

Quantum computing is a growing field which is becoming more advanced every day. Soon it could be used a lot in many sectors such as finance, medicine, and AI as we approach the commercial use of quantum computers [11].

As the number of qubits grow and the time under which qubits can maintain their state grows, I personally think AI could do very well in quantum computing. As the noise becomes smaller and quantum computers become more scalable, could easily outperform classical AI in many aspects, especially within chemistry due to the fact that “chemical reactions are inherently quantum” as stated by Alán Aspuru-Guzik [13].

I am incredibly excited to see where quantum computing is headed, and I hope to develop ground-breaking work of my own in the near future with my newfound knowledge of quantum computing.

References

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

[2] Nielsen, M. A. & Chuang, I. (2002) Quantum computation and quantum information. American Association of Physics Teachers.

[3] Romero, J., Olson, J. P. & Aspuru-Guzik, A. (2017) Quantum autoencoders for efficient compression of quantum data. *Quantum Science and Technology,* 2(4)**,** 045001.

[4] Le, P. Q., Dong, F. & Hirota, K. (2011) A flexible representation of quantum images for polynomial preparation, image compression, and processing operations. *Quantum Information Processing,* 10(1)**,** 63-84.

[5] Zhang, Y., Lu, K., Gao, Y. & Wang, M. (2013) NEQR: a novel enhanced quantum representation of digital images. *Quantum information processing,* 12(8)**,** 2833-2860.

[6] Sim, S., Cao, Y., Romero, J., Johnson, P. D. & Aspuru-Guzik, A. (2018) A framework for algorithm deployment on cloud-based quantum computers. *arXiv preprint arXiv:1810.10576*.

[7] Anon. Python [Online]. Available at: https://www.python.org/.

[8] Anon. Qiskit [Online]. Available at: https://qiskit.org/.

[9] Anon. SciPy [Online]. Available at: https://www.scipy.org/.

[10] Anon. JupyterLab [Online]. Available at: https://jupyter.org/.

[11] Anon. Five strategies to prepare for paradigm-shifting quantum technology [Online]. Available at: https://www.ibm.com/thought-leadership/institute-business-value/report/quantumstrategy

[12] Igic, Filip. Github Code [Online]. Available at: https://github.com/filipigic/FYP/.

[13] Savage, Neil. Google's Quantum Computer Achieves Chemistry Milestone [Online]. Available at: https://www.scientificamerican.com/article/googles-quantum-computer-achieves-chemistry-milestone/

Appendices

Project Proposal

Quantum Neural Networks

Project Proposal

1. Introduction

Artificial intelligence (AI) is becoming increasingly popular within many sectors and has already given us the likes of self-driving cars, smart assistants and much more.

However, such technology requires a large amount of computational power and with Moore's Law coming to an end, the rate of advancement within AI will be slower.

To overcome classical computation's technological and thermodynamical limitations, quantum computing devices that compute based on the laws of quantum mechanics are becoming more of a likely solution.

In my proposed project, I plan on creating my own QNN capable of facial recognition which classifies quantum data and gives an appropriate output with very good accuracy.

2. The Proposed Project

2.1 Aim and Objectives

The aim of the project will be to develop a fully quantum neural network (QNN) capable of facial recognition which will run on a simulation of a quantum computer, and possibly on an actual quantum computer depending on the hardware requirements.

The objectives will be the following:

- Analyse and design the QNN:
  - Research most suitable frameworks and simulators for development.
  - Look at different methods of quantum perceptron and architecture from current proposals and consider their suitability for my own QNN
- Develop the QNN using a quantum framework such as IBM's Qiskit [1].
- Convert classical data to quantum data i.e. take a low-resolution image and map it to a quantum state which will then be used for training and testing.
- Analyse the outputs of the training and test data and review its accuracy and efficiency.

2.2 Related Work

Quantum machine learning is a growing research field that has recently been making incredible advancements both theoretically and practically, with an increasing amount of research papers.

2.2.1 Training deep quantum neural networks [2]

In this paper, the writers propose a natural quantum perceptron capable of doing universal quantum computation when implemented into a QNN, that would aid the problems of creating a neural network architecture, calculating the cost and using an optimisation algorithm.

In their proposal a quantum perceptron is a general unitary that acts on input and output qubits whose parameters are set with the weights and biases.

They used their QNN to learn an unknown unitary both with and without error, suggesting that their method would be appropriate for quantum devices subject to noise such as a noisy intermediate scale quantum device (NISQ).

Their training algorithm is efficient due to the dependability only requiring the width of an individual layer rather than the depth of the network although it does result in the con of having to evaluate the network multiple times in order to get the derivative of the cost function.

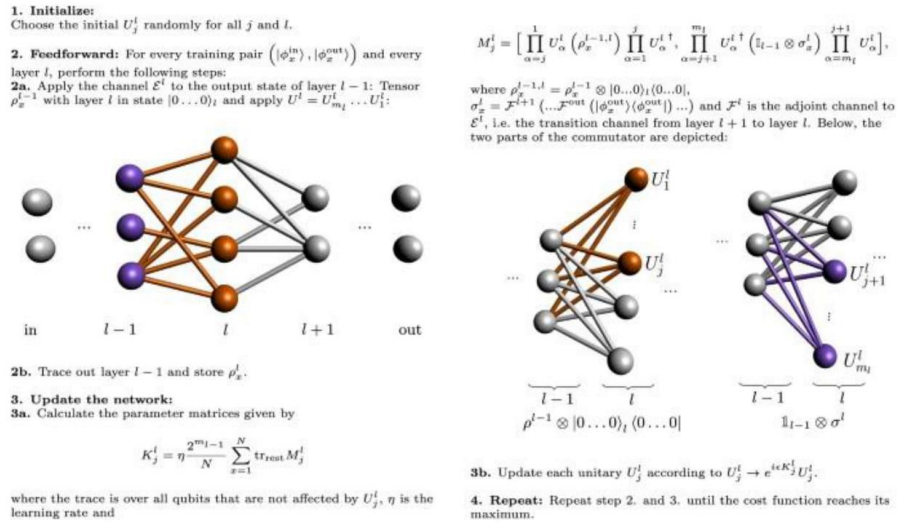


Figure 1. Box 1 Training Algorithm from their article

Overall, their paper shows much promise and with the development and improvements in NISQ devices and other quantum computers this will be a great method for quantum machine learning (QML).

2.2.3 Quantum Neuron: an elementary building block for machine learning on quantum computers [3]

In this paper, the writers propose a workaround to the problem of implementing a nonlinear activation function in a quantum computers linear nature by creating a natural neuron with threshold activation.

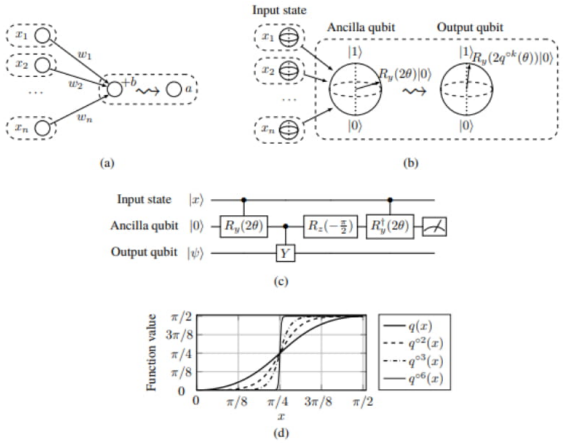


Figure 2. Figure from the Quantum Neuron article showing the basic setup of a quantum neuron model. a) The classical neuron setup b) The quantum neuron setup, showing the output qubit before and after the RUS circuit on a block sphere. c) Repeat-until-success (RUS) circuit. d) Nonlinear function and its self-composition.

This method allows for constructing a wide range of classical neural network constructions while keeping the ability of processing inputs in superposition, as well as keeping quantum coherence and entanglement.

In their paper, they show how they can rebuild a 3x3 structure with a corrupt top layer using their quantum neuron approach which uses a newly developed Repeat-until-success circuit technique for quantum gate synthesis.

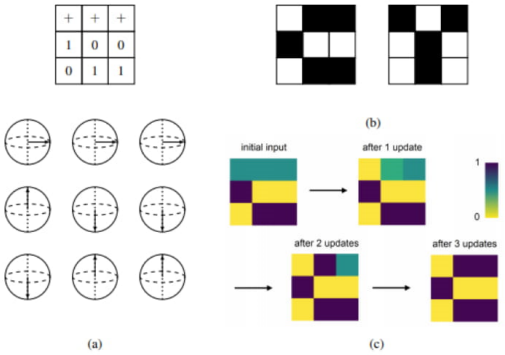


Figure 3. Example of a Hopfield Network built up of quantum neurons. a) Initial state of the network where the top three qubits are corrupted by being set to |+> state. b) The patterns the network is trained with both 'C' and 'Y'. c) The network through each update stage. The colours represent the marginal probability of each qubit; yellow being the state |0> and purple being |1>.

2.2 Overall Workplan

The work will be split up into the following stages:

- Research – Quantum computing is an emerging research field with great advancements happening everyday which is why it's extremely important for me to do constant, ongoing research and look out for new ground-breaking projects that could help me in developing an innovative machine learning model. This stage will run alongside other stages throughout the project.
- Analysis and Design – This stage is where I will look through different quantum simulators/frameworks such as Qiskit, Q# [4], Cirq [5], QuTip [6], Strawberry Fields [7, 8] etc. and decide which path I will take. I will also consistently be going through my research stage notes and looking for new methods that could give me ideas in developing my QNN. This stage will take approx. 7 days.
- QNN Development – In this stage I will be developing a fully quantum neural network that will take in quantum data for training and testing. This stage will require lots of time and preparation. I will have to work around many difficulties that present themselves with quantum computing such as the no-cloning theorem that states it is impossible to have an identical copy of an unknown qubit state. This will require me to adopt other researchers' proposals into my work or create a completely new type of neural network architecture for quantum computing. I estimate this stage will take approx. 5 weeks.
- Generating Quantum Data for training and testing – As quantum computers currently do not have the capacity to store data like in a classical computer, certain approaches must take place for us to manipulate the classical data and that is to embed the data into a qubit state. A good approach would be 'amplitude encoding' where I would take a vector 'x' and map its entries into different amplitudes of a quantum state [9]. This will take around 2 weeks.
- Training and Improvements – In this stage I will train the network and test it with the data sets I have made in the past stage. The training will be run through a couple of different multilayer networks such as a 1x1 and 1x2x1. This will take around 2 weeks depending on the how well the network responds to the data set.
- Analysis of QNN – In the final stage I will analyse the QNN I have built and compare it to other solutions, such as a basic feedforward classical neural network and a hybrid (Quantum-enhanced) neural network such as IBM's Qiskit and PyTorch [10] solution to classifying the MNIST dataset. This will roughly take around 2 weeks.
- Finally, I will spend the rest of my time writing up the report, following my notes I will have written down throughout my research. This stage will last up until submission day.

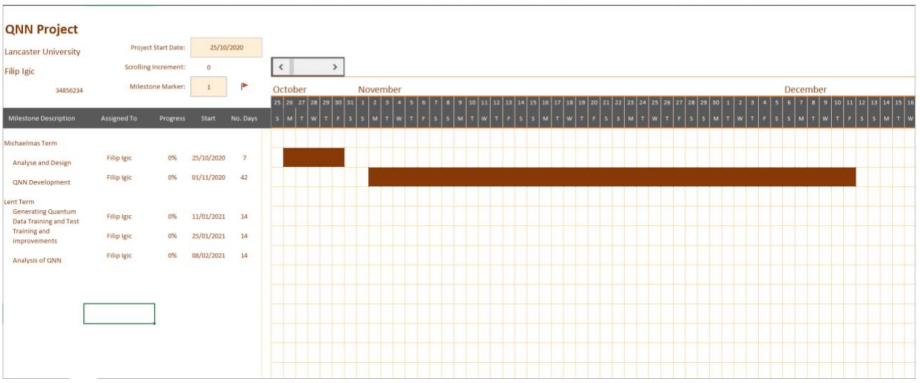


Figure 4. Gantt Chart showing my plan for Michaelmas term

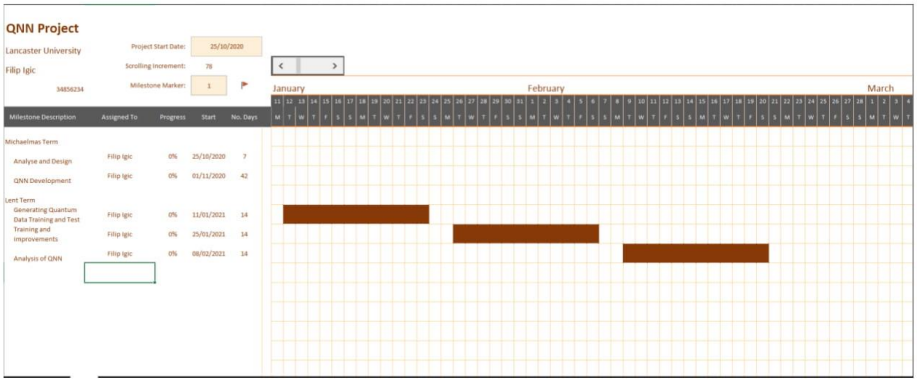


Figure 5. Gantt chart showing my plan for Lent term

3. References

1. https://qiskit.org/documentation/

2. Beer, K., Bondarenko, D., Farrelly, T. et al. Training deep quantum neural networks. Nat Commun 11, 808 (2020). https://doi.org/10.1038/s41467-020-14454-2

3. Cao, Y., Guerreschi, G. G. & Aspuru-Guzik, A. Quantum neuron: an elementary building block for machine learning on quantum computers (2017). https://arxiv.org/abs/1711.11240

4. Microsoft Quantum Development Kit. https://www.microsoft.com/en-us/quantum/development-kit

5. Cirq documentation. https://cirq.readthedocs.io/en/stable/

6. J. R. Johansson, P. D. Nation, and F. Nori: "QuTiP 2: A Python framework for the dynamics of open quantum systems.", Comp. Phys. Comm. 184, 1234 (2013) DOI: 10.1016/j.cpc.2012.11.019.

7. Nathan Killoran, Josh Izaac, Nicolás Quesada, Ville Bergholm, Matthew Amy, and Christian Weedbrook. “Strawberry Fields: A Software Platform for Photonic Quantum Computing”, Quantum, 3, 129 (2019).

8. Thomas R. Bromley, Juan Miguel Arrazola, Soran Jahangiri, Josh Izaac, Nicolás Quesada, Alain Delgado Gran, Maria Schuld, Jeremy Swinarton, Zeid Zabaneh, and Nathan Killoran. “Applications of Near-Term Photonic Quantum Computers: Software and Algorithms”, arxiv:1912.07634 (2019).

9. Quantum machine learning in feature Hilbert spaces. Maria Schuld∗ and Nathan Killoran arXiv:1803.07128v1 [quant-ph] 19 Mar 2018

10. https://qiskit.org/textbook/ch-machine-learning/machine-learning-qiskit-pytorch.html