



Trinity College Dublin

Coláiste na Tríonóide, Baile Átha Cliath

The University of Dublin

Quantum Image Processing

Paritosh Chauhan



Supervised by: Professor Rozenn Dahyot

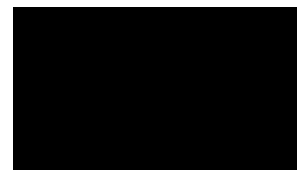
September 2020

A DISSERTATION SUBMITTED TO THE UNIVERSITY OF DUBLIN,
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE IN COMPUTER SCIENCE (INTELLIGENT SYSTEMS),
SCHOOL OF COMPUTER SCIENCE & STATISTICS
TRINITY COLLEGE DUBLIN, IRELAND

Declaration

I declare that this thesis has not been submitted as an exercise for a degree at this or any other university and it is entirely my own work.

I agree to deposit this thesis in the Universitys open access institutional repository or allow the library to do so on my behalf, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.



Paritosh Chauhan

September 7, 2020

Acknowledgments

I would like to express my deepest gratitude to my supervisor, Professor Rozenn Dahyot, who has not only provided immense support, guidance and feedback through each phase of this dissertation, but has also been a constant inspiration as a researcher.

I am grateful to Trinity College Dublin and the School of Computer Science and Statistics for helping me build a solid foundation of computer science principles as well as research methods, and providing me with a platform to excel.

The moral support and motivation extended to me by my family and friends have been indispensable factors in the completion and success of this dissertation, and for that I am truly thankful.

Any omission of acknowledgement does not reflect my lack of regard or appreciation.

Paritosh Chauhan
University of Dublin, Trinity College Dublin
September 2020

Abstract

The world is moving towards the Fourth Industrial Revolution which is driven by the advancement in technology like Quantum Computing, AI, Internet of Things (IOT), and other technologies. These technologies have been developed to integrate with each other and with other applications/services that use them to make our life easier. One such technology is Quantum Computing which is based on the laws of Quantum Mechanics and promises to solve the computation limitations that we find in any modern-day computers. Classical computers have always been limited in terms of processing power for fields like Computer Vision. Image processing plays an important role in algorithms in machine learning and AI that use it to classify images, enhance the image, or extract useful information out of it. Classical systems require a tremendous amount of memory to store and reconstruct the image which in turn increases the computation costs. These systems use finite states (0 and 1) to represent the information, while quantum computers consist of quantum bits (Qubits) that can be in superpositions of states 0 and 1. This property of qubits along with other laws like inference, entanglement, and coherence allows building a system that can outperform a classical system in terms of computation. This study provides a primer of quantum computing basics that is essential for understanding the underlying quantum computers. The study also juxtaposes current quantum technologies that are available in the market. The study then investigates the implementation of the FRQI model of image representation in the quantum simulator for storing and retrieval of pixel information to check on the efficiency of quantum computers for image processing. The results from the experiment illustrate that we could only recover up to 88% of the image from the current quantum circuit. This result when compared to the classical algorithms, requires more research on building efficient algorithms with the limitations of the current quantum processors and simulators. The study also reflects that the current quantum computer provides no advantage over their classical counterparts for image processing with FRQI image model representation.

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 10 |
| 1.1 | Motivation | 11 |
| 1.2 | Research Objective | 12 |
| 1.3 | Dissertation Structure | 12 |
| 2 | Background Research | 13 |
| 2.1 | Quantum Bit | 14 |
| 2.1.1 | A Single Qubit | 14 |
| 2.1.2 | Multiple Qubits | 15 |
| 2.1.2.1 | Correlation between qubits | 15 |
| 2.2 | Qubit Gates | 16 |
| 2.2.1 | Single Qubit Gates | 16 |
| 2.2.2 | Multi Qubit Gate | 18 |
| 2.3 | Noises in Quantum System | 19 |
| 2.4 | Quantum Circuit | 20 |
| 2.5 | Algorithm's Backbone | 20 |
| 2.6 | Quantum Computing and its Applications | 21 |
| 2.7 | Quantum Image Processing (QIP) | 22 |
| 2.8 | Image Representation in Quantum states | 23 |
| 2.8.1 | Quantum Lattice | 23 |
| 2.8.2 | Real Ket | 24 |
| 2.8.3 | FRQI - Flexible Representation of Quantum Images | 24 |
| 2.8.4 | Quantum Edge Detection | 25 |
| 2.8.5 | Quantum Processors | 26 |
| 2.8.5.1 | Quantum Annealers | 26 |
| 2.8.5.2 | Quantum Simulations | 27 |
| 2.8.5.3 | Universal Quantum | 27 |

| | | |
|----------|--|-----------|
| 2.8.5.4 | Noisy Intermediate-Scale Quantum systems | 28 |
| 2.8.5.5 | Fully error-corrected Quantum Computers | 28 |
| 2.8.6 | Quantum Computer Providers | 28 |
| 2.8.6.1 | DWave | 28 |
| 2.8.6.2 | ION Q | 29 |
| 2.8.6.3 | Rigetti | 29 |
| 2.8.6.4 | IBM | 30 |
| 2.8.6.5 | Microsoft Quantum Computer | 33 |
| 2.8.7 | Performance of Quantum Hardware | 36 |
| 3 | Design | 37 |
| 3.0.1 | Workflow | 37 |
| 3.0.2 | Image pre-processing | 37 |
| 3.0.3 | Quantum computing | 38 |
| 3.0.4 | Image Retrieval | 38 |
| 3.0.5 | Design Choices | 38 |
| 3.0.5.1 | Rigetti | 38 |
| 3.0.6 | Microsoft vs IBM | 39 |
| 3.0.6.1 | Hardware | 39 |
| 3.0.6.2 | Quantum Simulators | 39 |
| 3.0.6.3 | Qubits | 39 |
| 3.0.6.4 | Visualization | 39 |
| 3.0.6.5 | Circuit | 39 |
| 3.0.6.6 | Programming language | 40 |
| 3.0.6.7 | Noise Model | 40 |
| 4 | Implementation | 41 |
| 4.1 | Technical Setup | 41 |
| 4.2 | Hardware Setup | 41 |
| 4.3 | Images | 41 |
| 4.4 | Classical - Quantum Interface | 43 |
| 4.5 | Quantum interface | 44 |
| 4.6 | Quantum-Classical Interface | 46 |
| 4.7 | FRQI on Larger Pixel Image | 46 |
| 4.8 | Adding Noise Model | 46 |

| | |
|---|-----------|
| 5 Results and Discussion | 47 |
| 5.1 Image retrieval | 47 |
| 5.2 Geometric Transformation | 50 |
| 5.3 Analysis and Discussion | 51 |
| 6 Conclusion | 53 |
| 6.1 Limitations and suggested Future Work | 53 |
| 6.2 Final Words | 55 |
| Bibliography | 55 |
| Appendix A | 59 |
| A.1 Code and other files | 59 |
| A.2 Microsoft | 59 |
| A.2.1 Ways to trigger Q# programm | 59 |
| A.3 IBM | 59 |
| A.3.1 IBM installing libraries | 59 |
| A.3.2 IBM API Key Generation | 59 |
| A.3.3 Commands for python-resize-image | 60 |

List of Figures

| | | |
|------|---|----|
| 2.1 | Qiskit's representation of a single qubit | 15 |
| 2.2 | Qiskit's representation of a X-Gate applied to a single qubit | 17 |
| 2.3 | CNOT Circuit Representation from Qiskit | 18 |
| 2.4 | Toffoli gate using Qiskit | 19 |
| 2.5 | Quantum circuit representation of Quantum teleportation algorithm | 20 |
| 2.6 | Qubit Lattice Model | 23 |
| 2.7 | Real Ket Model | 24 |
| 2.8 | FRQI Pixel information in the form of equation | 25 |
| 2.9 | IBM's qubit placement for machine in Melbourne (15 qubits) | 30 |
| 2.10 | Biggest Quantum computer offering by IBM | 30 |
| 2.11 | Qsphere representation of 2 qubits | 31 |
| 2.12 | Measurement Probabilities | 31 |
| 2.13 | State Vector | 31 |
| 2.14 | Overview of Microsoft's Quantum Platform | 34 |
| 2.15 | Electrons in a Topological Qubit | 34 |
| 2.16 | Host program general execution model[1] | 36 |
| 3.1 | Image processing workflow | 37 |
| 4.1 | Image processing workflow | 42 |
| 4.2 | Code generated Image | 43 |
| 4.3 | Cat Image | 43 |
| 4.4 | Image processing workflow | 43 |
| 4.5 | Cat Image | 44 |
| 4.6 | Resized cat image | 44 |
| 4.7 | Quantum Computing Interface | 44 |
| 4.8 | Cat Non-Normalized Image | 45 |

| | | |
|------|-------------------------------------|----|
| 4.9 | Rotated cat image | 45 |
| 5.1 | Original Cat Image | 48 |
| 5.2 | Reconstructed cat image 1 | 48 |
| 5.3 | Reconstructed cat image 2 | 48 |
| 5.4 | Reconstructed cat image 3 | 48 |
| 5.5 | Original image | 49 |
| 5.6 | Greyscale Original image | 49 |
| 5.7 | Reconstructed image | 49 |
| 5.8 | Original image | 50 |
| 5.9 | Reconstructed image | 50 |
| 5.10 | Rotated image | 50 |
| 5.11 | Reconstructed image | 50 |
| 5.12 | Reconstructed Large Image | 51 |
| 5.13 | Failed Digit Rotation | 52 |

List of Tables

| | | |
|-----|---|----|
| 2.1 | Truth table | 19 |
| 2.2 | General category of Quantum Computer | 27 |
| 2.3 | IBM Qiskit Modules | 32 |
| 2.4 | Noise Model Parameters | 33 |
| 2.5 | Comparison between Rigetti, IBM, Qiskit | 36 |

Chapter 1

Introduction

The 20th century embarks with the shakeup in the computer paradigm by the development of so-called Quantum Computers. The world is moving towards a digital era where data processing will be an essential part of everyday life applications. With the increasing demand for faster and error-free processing, classical computers have been evolving continuously over time to an extent where they can be deemed to be not committing any computation error. Any circuit used in classical systems tends to cause an error in every 10^7 operations. As the workload is increasing on the data computation side, classical computers have been pushed to their limits in search of providing scalable computation power.

Many raise the question about the need for quantum computers when classical systems can currently fulfill most of the needs if the time to computation is not considered. The notion of developing these systems can be compared to the invention of Lasers. The research on lasers began to find alternative means of coherent light in comparison to light bulbs which are incoherent. The quantum mechanics lend a helping hand in explaining the reason for the random generation of electromagnetic waves from the light sources and engineering around it allowed to generate these waves in phases for a coherent source of light. Development around lasers evolved to an extent such that now, they have a variety of applications ranging from laser surgeries in medical fields to controlling remote control cars. These even are being used on some quantum computers where the atoms are controlled via the lasers. Similarly, quantum computers aren't meant to replace the classical systems but will be used in special cases where classical systems are lacking. But as was the case with lasers, the future of quantum computers seems to excite many researchers and industries with their capabilities. The relationship of quantum physics and computing is at a stage where progress in one field of science and technology acts as a reactant to another field that allows humans to create new technologies, is one of many ways to see how computers are going to be built in the future.

The combination of quantum mechanics with computer science has led to the development of systems

called quantum computers. These computers have a special property of providing faster computing using quantum particles like atoms and governed by the laws of quantum mechanics. These computers have been in demand mainly for research-oriented tasks like in the field of chemistry for simulating new chemical bonds that can be used for generating new vaccines or drugs. Material science can use these machines to find new elements in faster times that could have taken years with classical systems. Since the quest for quantum computing began with the development of Shor's algorithm[2] in 1994 that provides speedup in the factorization of a number, Quantum computers are expected to play a vital role in the cryptography.

Further, they could be very efficient in applications related to machine learning and AI where the current classical counterpart lack in, for example, training of data to generate models that can be used for classifications or prediction tasks. Image processing plays an integral part in fields like medical imaging with applications for better body image scans, faster diagnostic, and so on. They also form the backbone of many AI and computer vision applications. Most of the researchers and industries today concentrate not only on increasing the size of their datasets but also focuses on the quality of data. This is being done primarily to obtain more accuracy on the task that these datasets are being used for.

Digital image processing can be defined as a process in which an image is manipulated or useful information is extracted by running some algorithm. This is one of the fields that has picked up the pace of its research due to expansion in its applicability. They can be applied in learning about the image using machine learning algorithms, image enhancements to improve the quality, and many more. Classical systems limitations arise in this area as the amount of data required to process for an image increases with the quality of the image. With the advantages that quantum computer claims to provide, this field seems to be a perfect fit for the researchers to look into.

1.1 Motivation

Gordon E. Moore, who co-founded Intel had a perception that the number of transistors in a processing chip doubles every 18 months while its price reduces to half. This perception is known as Moore's Law. Designers of the transistors have always warned the world about the shrinking size of the transistors in the microchip which allows fitting more of these transistors on to the chip. On the other hand, this has also inspired them to innovate. If the transistor's size is reduced too small (less than $10nm$ per transistor), they might show some quantum effect and would not be able to stop the current from flowing through them. Researchers in this field have predicted to reach this limit in the next 5 years. Moore's law failing in the near future has fuelled the development of quantum computers at a much faster pace than before.

Another motivation to study quantum computers is the near-term devices that can be accessed via the cloud. Most of the leading Technology companies and some startups have shown advantages of using these computers over their traditional counterparts and provide help and support using videos & documentations to increase the awareness that might bring more people into research and developed of algorithms for these futuristic computers.

With the notion of understanding the future of computing, I wanted to drive my research by understanding the way in which quantum systems are been build and grasp the theory behind claims of it being faster than classical computers. There has been significant progress in the development of these systems in terms of accessibility. In theory, the limitations of classical systems in the field of image processing have been expected to be solved by using quantum systems, and implementing these solutions to find the reality of these systems adds to the motivation.

1.2 Research Objective

This study is done to understand the quantum computer, the need for it, and its applications in various fields. It also provides a comparison of current quantum computer technologies and the quantum processor they use. Finally, the study assesses the problems with image representation using the FRQI model in quantum systems.

1.3 Dissertation Structure

This dissertation is divided into 6 different chapters. Starting with Chapter 1 which contains the introduction giving an overview of quantum technology alongside the motivation and research objective for this topic of study. Chapter 2 provides a primer on the quantum basics and technologies that are currently available. Additionally, it also includes background research on image processing. Chapter 3 provides a brief description of the design implementation of the project which helps in understanding the next chapters. Further, Chapter 4 provides intricate details of the implementation that was done for the experiments during the study. The results obtained are analyzed and discussed in Chapter 5. Finally, Chapter 6 provides the conclusions on the findings of this study with any future works that can be done to improve on the results obtained in this study.

Chapter 2

Background Research

The development of transistor by John Bardeen, Walter Brattain, and Will Shockley in the year 1947 marked the beginning of a revolution that provided the fuel to increase the power of computer hardware and set the pace that allowed the growth to be codified by Gordon Moore. In 1965, Moore stated his observation that the number of transistors in an integrated circuit will double every two years while the cost of the computers is halved, which is known as Moore's Law. In the current scenario it is also believed that with the decreasing size and complexity of the transistors, the limitations could be reached sooner as quantum effect interferes with the functioning of the devices due to the smaller size of the circuit.

The failure of Moore's law could be solved by using a different computer paradigm that employs quantum mechanics for computations. In the year 1982, Nobel prize winner Richard Feynman proposed the first Quantum Computer for simulating experiments in physics [3]. In his paper, he stresses the point about the need for a quantum system that provides the platform for computation which is not possible to simulate in a classical computer. This allowed the base for the creation of various new algorithms that solve the problem seen on a classical computer.

In 1994, Shor's algorithm [2] was introduced by Peter Shor that proved the intractability of a classical computer to factorize a large number and produced a quantum algorithm that followed the principals of quantum mechanics. Similarly, Grover's algorithm devised by Lov Grover is a quantum algorithm that provides a search over unstructured data much faster than any other classical search algorithm. The following section discusses the basics of Quantum computing and some of the applications where quantum computers theoretically are proven to be better than classical counterparts.

2.1 Quantum Bit

2.1.1 A Single Qubit

In the classical computer fundamental unit of information is a bit that can attain a state of 0 or 1. Similarly, all the quantum computers rely on qubits for computational purposes which can have states $|0\rangle$, $|1\rangle$, or superposition of $|0\rangle$ and $|1\rangle$. The qubit state is denoted using Dirac notation " $|\rangle$ " from quantum mechanics. A qubit is said to be in superposition when it can be in any state other than $|0\rangle$ and $|1\rangle$ and combined in a linear combination of states as in Eq 2.1 .

$$|\psi\rangle = \alpha |0\rangle + \beta |1\rangle \quad (2.1)$$

The $|0\rangle$ and $|1\rangle$ are vectors that are in a two-dimensional complex vector space. They also form the orthonormal basis of that vector space. The α and β are considered to be complex numbers which also represents the probabilities of the state that the qubit acquires. If the value measured by the qubit is 0 then the probability is given by $|\alpha|^2$, similarly its $|\beta|^2$ for state equal to 1 and summation of these probability will be 1 , i.e. $|\alpha|^2 + |\beta|^2 = 1$. Since the states that a qubit can attain has infinite possibilities, two special states correspond to the classical state in the quantum world represented by -

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (2.2)$$

The Equation 2.2 describes the special state that can be combined to form and other state, For example - $\begin{bmatrix} a \\ b \end{bmatrix}$ can be represented as Eq 2.3 where a and b are complex numbers.

$$|\psi\rangle = a \begin{bmatrix} 1 \\ 0 \end{bmatrix} + b \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (2.3)$$

To represent the state using real numbers, we can use Eq 2.1 and sum of probabilities to derive equation 2.4 where $a = \cos\theta/2$ and $b = \sin\theta/2$. The equation ignores a term, $e^{i\varphi}$ as its has not observational effect on the state of the vector [4].

$$|\psi\rangle = \cos \frac{\theta}{2} |0\rangle + e^{i\varphi} \sin \frac{\theta}{2} |1\rangle \quad (2.4)$$

To visualize a single qubit we make use of a three- dimensional sphere called Bloch Sphere. Figure 2.1 represents a Bloch Sphere build using IBM's Qiskit libraries and python [5].

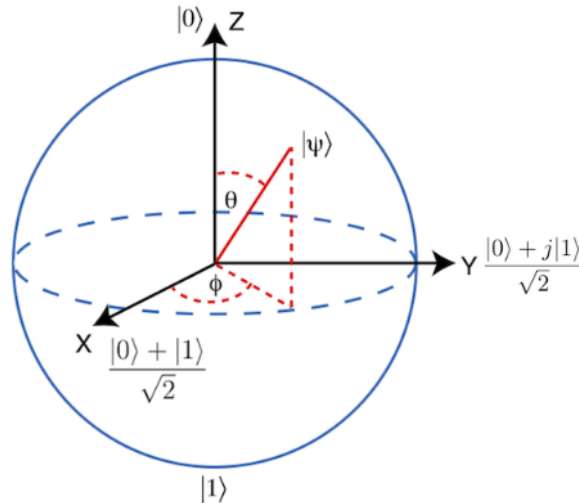


Figure 2.1: Qiskit's representation of a single qubit

2.1.2 Multiple Qubits

A quantum computer using a single bit is not fast even when compared to a simple calculator. Hence to realize and use the true computational power of quantum computers we employ multiple qubits. Similar to the case of a single bit, a two-qubit system consists of computational basis states represented by $|00\rangle, |01\rangle, |10\rangle, |11\rangle$. A point to note is that the vector represented by the two-qubit system is four-dimensional due to the tensor product of the states. This allows us to state that in an n -qubit system, the vector is 2^n dimensional. In a two qubit system, vector is represented using the vector, $|\psi'\rangle = a_{00}|00\rangle + a_{01}|01\rangle + a_{10}|10\rangle + a_{11}|11\rangle$ and the sum of the squares gives a result equivalent to 1, i.e. $|a_{00}|^2 + |a_{01}|^2 + |a_{10}|^2 + |a_{11}|^2 = 1$

Eq 2.5 vector representation of the tensor product between two qubits representing $|00\rangle$

$$|00\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \quad (2.5)$$

2.1.2.1 Correlation between qubits

Bell state or EPR pair is a special state in a two-qubit system where the probability of attaining 0, as a result, is equally possible as the probability to attain 1. This equal probabilities can be represented as $\frac{|00\rangle + |11\rangle}{\sqrt{2}}$. The qubit is said to be correlated when the measurement of state from two qubits are the same. In other words, the measurement of one qubit tells us the state about the other qubit in the same system. Another way to understand the entanglement of the qubits is that the tensor product of

the two-qubit cannot be represented as a single qubit state. [6]. This property is the backbone for a lot of quantum algorithms, for example, quantum teleportation.

2.2 Qubit Gates

To generate an expected output from the given input, we need to perform some logical operations on the circuit. In classical computing, the gate is considered universal if any transformation on the input takes a finite length circuit [6]. Similarly, in quantum computing, we are allowed to perform unitary operation and measurements. IBM uses microwave pulses to perform the rotations on the qubits and thus transforming the state.

2.2.1 Single Qubit Gates

NOT Gate In classical environment, the information (bits) is passed on to the gates for transformation using wires. In terms of classical bits, the NOT gate transforms $1 \rightarrow 0$ and $0 \rightarrow 1$ which is flipping the input. On the other hand in quantum world, the qubits are in superposition and not gate acts linearly to flip the associated probabilities of the state using pauli gates. The derivation of pauli gates can be derived from the outer product of $\langle 0|1\rangle$ (ket of $|0\rangle$ and Bra of $|1\rangle$) and $\langle 1|0\rangle$

$$\langle 0|1\rangle + \langle 1|0\rangle = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = X \quad (2.6)$$

Equation 2.6 represents the information relates to pauliX gate. If a qubit is in the state $|\Psi\rangle = \beta|1\rangle + \alpha|0\rangle$ and vector represented as $\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$, When we apply the pauliX gate (eq 2.6) to it the result is showed in eq 2.7 as $|\psi'\rangle = |\psi\rangle X$.

$$|\psi'\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \beta \\ \alpha \end{bmatrix} \quad (2.7)$$

From the above equation we can deduce that $|\Psi'\rangle = \alpha|1\rangle + \beta|0\rangle$. The X-Gate provides a rotation of π around the x axis of the rotation. Figure 2.2 shows the rotation of a vector along x axis in Bloch Sphere after applying X Gate. All the gates that perform transformation on a single qubit can be represented as a two by two matrices and has a unitary constraint. This unitary constraint is known as Born rule ($|\alpha|^2 + |\beta|^2 = 1$) which must hold true even after the gates perform the transformation. The unitary metrics U represents a metrics that

$$UU^\dagger = I \quad (2.8)$$

where I is an identity matrices and the U^\dagger represents the adjoint of U (obtained by transposing and then complex conjugating U).

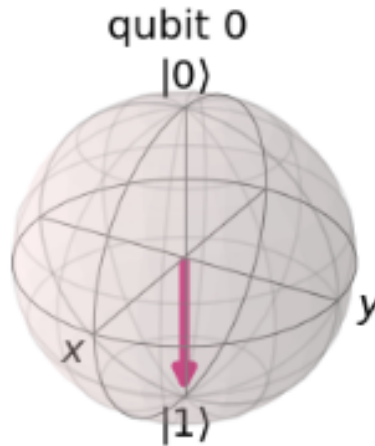


Figure 2.2: Qiskit's representation of a X-Gate applied to a single qubit

Hadamard Gate The Hadamard Gate is another important gate that takes a qubit and puts it in the superposition between $|0\rangle$ and $|1\rangle$. By applying H Gate to the $|0\rangle$ we get a qubit state in the position $\frac{|0\rangle + |1\rangle}{\sqrt{2}}$ while the transformation on $|1\rangle$ equals $\frac{|0\rangle - |1\rangle}{\sqrt{2}}$ and the gate is represented as

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (2.9)$$

Other Gates Some other important gates that perform rotations on a qubit along the axes of Bloch sphere are Pauli Y Gate and Pauli Z Gate. The Z Gate keeps the $|0\rangle$ unchanged and flips the sign of $|1\rangle$ to $-|1\rangle$. It is represented in the form of Pauli matrices as in eq 2.10

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad (2.10)$$

The Y Gate rotates the state vector along the y-axis and changes $|0\rangle$ to $-i|1\rangle$ and $|1\rangle$ to $i|0\rangle$. The Pauli matrices are represented in eq 2.11

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \quad (2.11)$$

Some other Gates which are used in the quantum world is the S Gate and $\pi/8$ Gate (T Gate).

Some of the correlation of these gates are $H = \frac{X+Z}{\sqrt{2}}$ and $S = T^2$.

$$S = \begin{bmatrix} 1 & 0 \\ 0 & -i \end{bmatrix} T = \begin{bmatrix} 1 & 0 \\ 0 & \exp(i\pi/4) \end{bmatrix} \quad (2.12)$$

2.2.2 Multi Qubit Gate

CNOT Gate In the classical computing environment, the NAND gate is considered as a universal gate as all the other gates like Not, Xor, And, Or and Nor can be derived from some combination of Nand Gate. Similarly in the quantum computing, CNOT or Controlled NOT gate provides the universality related to gates that can be used with other single-qubit gates. The Circuit representation of the CNOT gate is generated using Qiskit. [7]. The figure represents two qubits q_0 and q_1 where q_0 is the control bit and q_1 is a target bit. In CNOT, the change in the target qubit is determined by the control bit, i.e. if the control bit is 1 then the q_1 's state is flipped. The states can be represented as in Eq 2.11 and the matrix representation is similar to Eq 2.12.

$$|00\rangle \rightarrow |00\rangle; |01\rangle \rightarrow |01\rangle; |10\rangle \rightarrow |11\rangle; |11\rangle \rightarrow |10\rangle \quad (2.13)$$

$$U_{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (2.14)$$

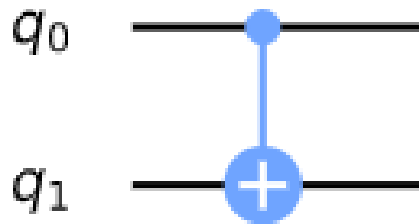


Figure 2.3: CNOT Circuit Representation from Qiskit

Toffoli Gate As we step into the new generation of computers, we want to ensure that the present classical logical circuits should be able to be incorporated in the quantum circuits. Straightforward thinking might be to make use of the qubit gates discussed before, but this is not the case. The qubit

gates are reversible, i.e. it can inverse itself Eq 2.8 while most of the classical logical gates like NAND gates are inherently irreversible.

Toffoli gates provide a reversible gate that can be used to replace the classical gates from the circuit. FIGURE [8] shows a Toffoli gate and Table 2.1 shows its truth table. The Toffoli gate consists of a single target qubit (q_2) which is controlled by two control qubits (q_0, q_1). The target bit is flipped only when both the control bits are in the state $|1\rangle$. The target qubit's state decides on the end resultant gate of the Toffoli gate. It can be an And gate or a NAND gate depending on the target qubit set to $|1\rangle$ or $|0\rangle$.

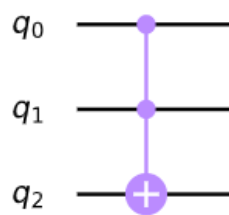


Figure 2.4: Toffoli gate using Qiskit

| Inputs | | | Outputs | | |
|--------|---|---|---------|----|----|
| a | b | c | a' | b' | c' |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 |

Table 2.1: Truth table

2.3 Noises in Quantum System

Quantum Coherence is related to the idea that each particle can be considered as a waveform. If this waveform-like structure is split into two waveforms, then the resultant two states form a single system such that they are in a superposition of each other. In a quantum system, this coherence is the heart of computing where the qubits can represent both 1 and 0 at the same time. These particles need to be isolated to maintain this "coherency" which makes it impossible to read or manipulate states. Any of the actions that force the exposure of particles to the environment like measurements of states result in "dissolution" or decoherence of the particle. Quantum computers need to maintain the coherence of the states to perform any meaningful computations. The quantum gates noises are divided into three main categories - bit flip, phase flip, and bit-phase flip. These noises are part of the quantum error correction code and are directly proportional to the noise generated in the circuit. In other words,

more gates that are used, there is a tendency to generate greater noise on the Quantum computer.

The existence of external interference is another problem that quantum computers have to deal with. The qubits are very fragile to interference like magnetic fields, temperature, and many more. Using excessive energy to perform manipulations on the qubits to hardware faults, all result in noises that affect the outcome of the system. Fidelity is used for checking the performance of the quantum system. It is a measure that ranges between 0 and 1. In simple terms, it tells us about how close the two states are to each other. A fidelity of 1 illustrates that the quantum states are the same while a value 0 illustrates the opposite. The equation for finding the fidelity is expressed in Eq 2.15 [9]. A metric can be defined based on this equation that provides distance between the two states called Bures Distance.

$$F(\rho, \sigma) = (T_r \sqrt{\sqrt{\rho} \sigma \sqrt{\rho}})^2 \quad (2.15)$$

2.4 Quantum Circuit

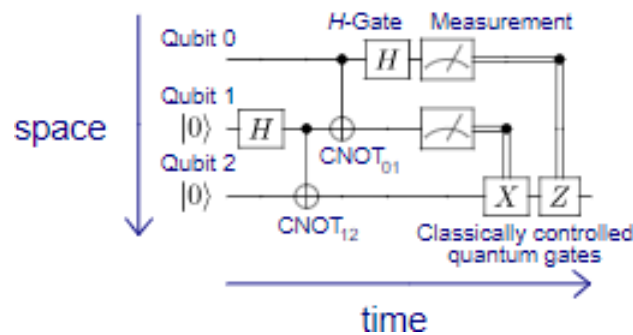


Figure 2.5: Quantum circuit representation of Quantum teleportation algorithm

A quantum algorithm consists of series of gate instructions to perform the action. These sequence of gates are placed in order of there call to form a quantum circuit. Each line in the circuit represents a register/qubit while the gates are represented based on their type. Figure 2.5 shows a Quantum teleportation circuit [10] which employs 3 qubits with two qubits initialized to $|0\rangle$ state. The circuit is read from left to right and H gate is used in qubits q0 and q1 & CNOT gate between q2,q1, and q1,q0 qubits. The measurement is then stored in the classical controlled bits.

2.5 Algorithm's Backbone

This section throws some light on the important elements that most of the quantum algorithms use in some way or another that provides them the arsenal for speeding up the computation.

Quantum Fourier Transform The best way to solve a problem in mathematics or computer science is to divide a bigger problem into smaller problems which has a known solution. Some of the famous Quantum algorithms like Shor's algorithm has Fourier Transform as one of the key elements used for factorizing a large number. In Quantum Fourier transform, the states are provided are orthogonal and perform linear operation on the basic states defined by the Eq 2.15 where $N = 2^n$ and the number of gates required is n^2 .

$$|j\rangle \longrightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{\frac{2\pi i j k}{N}} |K\rangle \quad (2.16)$$

Eq 2.5 can also be written in the form of y_k which is a discrete Fourier transform of the amplitudes x_j . These transformation are unitary and the proof of this is beyond the scope of the current research.

$$\sum_{j=0}^{N-1} x_j |j\rangle \longrightarrow \sum_{k=0}^{N-1} y_k |k\rangle \quad (2.17)$$

One of the main usages of these transformations is to estimate the phases manipulation which in turn is used for factoring or order-finding problems. The prime factor of an n-bit number can be found in $O(n^3)$ operations by transforming it into another problem of finding the order of a number x with co-prime N. This was the basis of Peter Shor which he builds upon to find the factorization of a number in $O(\log(N)^3)$ times.

2.6 Quantum Computing and its Applications

The invention of quantum computers are not meant to replace their classical counterparts but have the advantage of solving problems faster (some exponentially faster) than what a classical system would be able to. Tech giants around the world like Google, Microsoft, IBM, Rigetti Computing (Barkeley-based startup), D-Wave, and Intel are working extensively towards building a quantum computer. Their research has allowed other companies from other fields like finance and automotive to partner with quantum computer providers to take advantage that it has to offer.

For example, Delta Airlines uses IBM's quantum computer for routing/rescheduling flights during a crisis [11]. Banking giant JP Morgan & Chase uses a quantum computer from IBM Q to improve pricing on options and portfolio options using quantum computers [12]. Another important application of the quantum computer is in the field of chemistry. The Quantum computers provide simulations that provide better insights on the molecular bonds. Major players like IBM and Microsoft have libraries that are specifically designed for usage in the research of chemistry. One of the first adapters of a quantum computer for chemical reaction study is Mitsubishi chemicals that partnered with IBM to

study the reaction of lithium with oxygen in Li-air Batteries. [13].

Quantum computing also paves the path for improvement in the field of Artificial Intelligence in particular Machine Learning. One of the ways quantum computation can be proved advantageous over classical computing is the speed of results that can be achieved. We can also learn more from a lesser amount of data and provide better results when the data is noisy.

With the advancement in the quantum hardware over the years, researchers are opening up to find solutions to the shortcomings of classical systems in the field of Image Processing.

2.7 Quantum Image Processing (QIP)

Image classification using neural networks remains one of the most researched areas in the machine learning domain. The basic idea is to imitate the human image recognition using techniques in computing that imitates intelligence. Convolutional Neural Network is one such technique that requires training from a known set of images to predict an unknown set. The training of the network requires to study(see) many samples to accurately predict an unknown sample. This requires a large chunk of time and resources in the traditional computing environment.

In addition, the images have improved drastically over the period which adds new problems in the storage and retrieval as more information exchange is required. Classical computers are lacking due to the memory management techniques that are leveraged for storing image data. The approach for storing an image in the classical system requires the information about light intensity, pixel's correlation, and pixel's spatial disposition are stored in cells that are independent of each other and hence require more computation to reconstruct an image. Due to this, the reconstruction and retrieval of the image requires individual reads of bits and doesn't allow parallelism which in turn makes the process slower.[14]

Image Processing is a process of extractions of features from an image. In the case of neural networks, the feature vectors are fed to the network which is calculated by reading the data and its properties from the bits that are stored independently. The current systems provide a way that is not efficient in terms of computation and lacks parallelism due to the limited flexibility of the storage in memory. There is also a wastage of computations in terms of iterations that are required to finding a correlation between pixels.

Quantum Image Processing deals with the image representation and storing of image data in quantum states and transform those states to achieve the goal. The first step involved in the processing of images in quantum computers is translating pixels of an image in the quantum states. The next

section provides a summary of various techniques that have been published towards this translation of pixel to quantum states.

2.8 Image Representation in Quantum states

2.8.1 Quantum Lattice

A group of pixels comes together to form an image. These pixels contain properties like position and color intensities, that allows us to understand the image. The first method ever that allowed to convert pixel values into a quantum state was the 'Quantum Lattice' model. The research provided an analog conversion of the pixel into quantum states. For an image of size $N \times N$ image, $N \times N$ qubits are used that are arranged in the matrix format where each qubit is equivalent to each pixel. The research focused on the representation of the pixel in the form of frequencies rather than the linear combination of the RGB color model.[15] The frequencies represented by the pixel are calculated by a hypothetical machine that converts electromagnetic waves of the light into initialized qubits. The whole system can be represented as

$$A: F \longrightarrow \psi \quad (2.18)$$

where ψ represents the qubit that is initialized with the frequency F read by the machine A . ϕ is represented in the form of Eq. 2.4 where

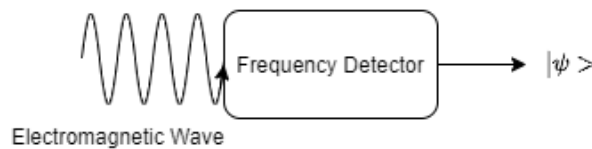


Figure 2.6: Qubit Lattice Model

The 'qubit lattice' structure is formed by setting up identical $K-1$ qubit behind each qubit that represents the frequency of the pixel. These frequencies are encoded in θ . The main motive to store the frequency in an identical qubit was for measuring the frequency back to pixel values to reconstruct the image from the quantum state. The amount of time it requires to retrieve the data is based on the number of pixels/qubits involved in the process. The number of qubits is directly proportional to the accuracy in the retrieval of images from its quantum state. Let a qubit have state α and β . We can denote the number of measurements for these states as M_α and M_β . Then by the Eq. 2.04 we can deduce $\cos^2 \frac{\theta}{2} = M_\alpha / (M_\alpha + M_\beta)$ when K is measured for infinite times. The frequency can be then retrieved when solved for theta.[16]

2.8.2 Real Ket

With the introduction of qubit lattice, Latorre in 2005 introduced an entangled model of image representation where each image was divided into 4 quadrants where each quadrant was numbered starting from left to right from the top row. these quadrants were then again divided into another 4 quadrants with numbers in the same manner. This division was continued until we get a single pixel. The model divides the image into smaller images (by dividing the image into 4 parts) and create a Quadtree structure with coefficients as the greyscale image Fig 2.7.[image compression and entanglement]

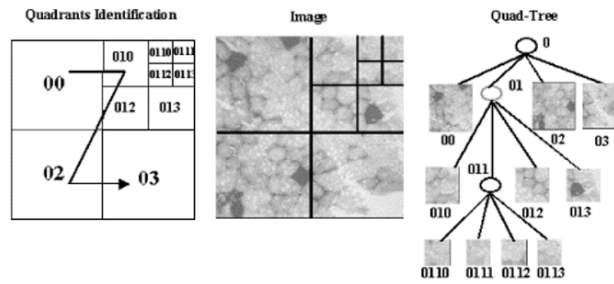


Figure 2.7: Real Ket Model

Both Real Ket and Quantum lattice provided ways to represent the image but they had their limitations. The lattice model required more qubits to represent an image that is practically impossible to attain. The real ket model provided better image compression and proved to be faster than the quantum lattice model but was limited with the randomness that was required in the pixels to work efficiently which was not in the case when we consider a real-world image as the pixels are related along with the required numbers of qubits to represent each pixel of an image.

2.8.3 FRQI - Flexible Representation of Quantum Images

In 2010, FRQI was introduced that provided an advantage over other models discussed as they require a lesser number of qubits to represent an image. In FRQI, the number of qubits required to represent an image of size $2^N \times 2^N$ requires only $2N+1$ qubits. This representation takes advantage of superposition between the qubits and allows encoding of color intensity and position of the pixels in the normalized state of qubit [17]. The image is represented in terms of θ in Eq. 2.18 and 2.19. The θ allows us to encode the color intensity while the position of the pixel is represented via $|i\rangle$.

$$|I(\theta)\rangle = \frac{1}{2^n} \sum_{i=0}^{2^{2n}-1} (\cos \theta |0\rangle + e^{i\varphi} \sin \theta |1\rangle) \otimes |i\rangle \quad (2.19)$$

$$\theta \in [0, \frac{\pi}{2}], \quad i = 0, 1, 2, 3, \dots, 2^n - 1 \quad (2.20)$$

The normalized state evaluates to $||I(\theta)\rangle||$ which when looked with respect to eq 2.18

$$||I(\theta)\rangle||^2 = \frac{1}{2^n} \sqrt{\sum_{i=0}^{2^{2n}-1} (\cos^2\theta + \sin^2\theta)} = 1 \quad (2.21)$$

| | |
|------------------|------------------|
| θ_0 00 | θ_1 01 |
| θ_2 10 | θ_3 11 |

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

Figure 2.8: FRQI Pixel information in the form of equation

The images are transformed from the initial state $(|0\rangle^{\otimes 2n})$ of the qubit to FRQI state using unitary transformation (denoted by P) that involves Hadmard Gate (denoted by H) to create superpositions of the initial qubit state followed by controlled rotations (denoted by R) to create the FRQI. Using the H gate (eq 2.9) on the controlled rotations around X and Y axis the resultant is an FRQI state represented as $P = HR$.

In terms of advantages of FRQI, the superposition of the qubit sequence allows us to transform all the pixels by just changing a single qubit. The drawback of this method of image representation is the number of gates that are required to prepare the image in the FRQI quantum image. A straightforward implementation of this method requires 2^{2n} controlled rotation and $2n$ Hadmard Gates. The overall computational complexity increases to quadratic (2^{4n}). Another drawback that comes with the FRQI state is that it can only be applied to square images.

Over the years many of the researchers have advanced in using FRQI as the base to improve on the various aspects in and around FRQI fundamentals. NEQR builds on the advantages that FRQI provides and takes 2 entangled qubits to store the pixel and color properties of a pixel. The method reduces the overall computational complexity from (2^{4n}) to (2^{2n}) [18]. It concentrates on the drawbacks of the FRQI model and store the information based on the qubit sequence that allows results in halves the computational complexity and improves the compression ratio by 1.5X. There are many more methods that researchers have published to represent an image in quantum states. but FRQI and NEQR are the most common ones from which FRQI having higher applicability in the current scenario.

2.8.4 Quantum Edge Detection

Image processing involves edge detection which is one of the basic transformation processes with applications like pattern matching and many more. A classical edge detection algorithms involves processing the image through varieties of masks to compute the gradient. This increases the com-

putational complexity to $O(2^n)$ as each pixel needs to be visited and the gradient for it needs to be calculated. The QHED (Quantum Hadmard Edge Detection) uses Hadmard gates to detect the boundaries in an image (Eq 2.22). The pixels are paired with their neighboring pixels and Hadmard gate is applied on it. The pixels with same identical density are cancelled out while the ones with the different values remain. In this way the boundaries for even pixels of the image are detected. To measure all the pixels of the image, quantum amplitude permutation is applied that performs the action in $O[\text{poly}(n)]$ [19].

$$|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} \quad |1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} \quad (2.22)$$

2.8.5 Quantum Processors

The advancement in technology in recent years has allowed us to transform quantum computers from their theoretical conception to physical and ready to use machines. These machines make use of particles that exhibit quantum properties and form the primary element in quantum processing units. These computers can be characterized into 3 different types based on the level of their implementation and the complexity of the processing that they can handle. Table 2.2 provides an overview of the categories.

2.8.5.1 Quantum Annealers

These types of Quantum computers are the simplest to build and are very restrictive. They are mainly used to solve the optimization problem and are built to achieve one type of task. The notion of having annealers in the scientific community is to find the right fit between the processor and algorithm to solve the problem faster. D-Wave is a company that specializes in annealers and has tested an algorithm that showed some positive results for optimizing traffic flow in the Beijing area. The company partnered with Google to experiment with VW.[20]. The optimization task when performed in classical computers can take years to provide a good result while in the case of Quantum annealers if the qubits are increased it can provide the result for the same task in hours. The current generation of quantum annealers, some of the scientists argue that annealers' performance can be matched by some of the supercomputers and be less error-prone than the annealer.

| Type | Application | Computational Power |
|-------------------|---|---|
| Quantum Annealer | Optimization Problems | Little or same speed as Classical Computers |
| Analog Quantum | Quantum Chemistry, Material Science, Machine Learning | High compared to Quantum Annealer |
| Universal Quantum | Secure Computing, Cryptography, Machine Learning, Quantum Dynamics | Very High - Reaching Quantum Supremacy |

Table 2.2: General category of Quantum Computer

2.8.5.2 Quantum Simulations

Quantum systems in its true essence are uncontrollable and unpredictable. The quantum simulators are built to mimic these dynamic or static properties of the quantum system. These simulations are characterized as Analog simulations, Digital Simulations, and a hybrid approach, i.e Analog-Digital simulations. The analog simulations were discussed by Richard Feynman [3], with the idea that the analog systems will help in stimulating the use of actual particles and provide robustness and scalability. digital simulations on the other hand, use logical quantum gates to provide the simulation with more flexibility and universality. The hybrid approaches allow to reduce the number of qubits required to simulate which reduces the errors and provides robustness along with scalability. These simulators have applications across various fields like Quantum Chemistry, where simulating chemical bonds could provide a faster generation of vaccines or drugs that can save lives. Similarly, computationally intensive tasks like machine learning can be improved with the help of these types of simulators. These quantum systems can perform simulations for about 20 to 50 qubits.

2.8.5.3 Universal Quantum

Universal Quantum computers are the most difficult machines to implement due to its technical challenges. When these challenges are overcome they will provide the fast outcomes to any complex calculations with the least time possible. These machines are expected to have processor units having more than 100,000 qubits. Researchers around the world have designed algorithms that can perform best on these types of machines. Some of those have been mentioned above like Shor's Algorithm and Grover's algorithm. The performance that is expected from these machines will revolutionize fields like quantum AI, Quantum Chemistry, Material Design, and many more. The highest qubits that are

achieved currently are by Honeywell in partnership with Microsoft that has a Quantum Volume of 64 Qubits.

2.8.5.4 Noisy Intermediate-Scale Quantum systems

NISQ machines are being inspired by the annealers and being build to be less prone to noises that generate errors in the results of quantum machines. They generally have qubits in the range of 50 to 100 qubits and require lesser qubits than Quantum annealers. These machines are still prone to some noises and hence cannot be deemed as replacement systems for quantum annealers.

2.8.5.5 Fully error-corrected Quantum Computers

In any quantum computer, there are limitations due to errors that occur in the process such as decoherence or any other quantum noise. These systems will try to resolve those drawbacks. In theory, these computers will be capable of keeping the system on the check and correct any errors if found. The corrections can be performed by employing special algorithms and additional qubits to keep track of the states. The implementation of these machines are still a talking point as the overhead to designing algorithms and added complexity of the qubits to correct errors will add to the overall performance of the system.

2.8.6 Quantum Computer Providers

There are many players in the race of achieving "Quantum Supremacy". Quantum Supremacy is a term that suggests that quantum computers are capable of handling any type of complex calculations and can be applied universally to any field of study. Supremacy also indicates that Quantum computers will be able to solve problems that classical computers cannot in a reasonable amount of time. Many companies are running the race to create and implement quantum technologies for fields where classical computers' processing power is being consumed to their limits. Companies like IBM, Google, Microsoft, Regetti, and Dwave have build quantum computers and collaborating with other companies to implement techniques to take advantage that quantum computing has to provide.

2.8.6.1 DWave

Dwave developed a system that uses quantum dynamics to solve problems in areas like AI, Machine Learning, optimization problems, and material science. The company works towards the goal to provide services and software for quantum computing. The company leverages on finding optimal solutions of corresponding mathematical function's global minima using quantum annealing to solve

problems. Dwave Leap provides an ecosystem of products that helps developers to work with their computers. It includes cloud access to the quantum computer along with IDE (Integrated Development Environment) with Dwave's Ocean SDK, a hybrid solver that allows checking the best possible way to solve in a combination of both classical and quantum computers and a problem solver that allows the developers to visualize the way their command is being transformed in company's QPU. In the heart of the Quantum Processing unit, metal loops form the qubit and tend to exhibit quantum-mechanical effect and become superconductors when used at close to absolute-zero temperatures. Other qubits are connected using similar metal loops that allow for communication between the qubits. [21]

2.8.6.2 ION Q

ION Q takes a different approach than other companies and uses actual atoms in the form of qubits. The qubits are captured in a 3-dimensional space using precision optical and mechanical engineering. Ionization is the process of converting Neutral Ytterbium Atoms (Y_b) into a ytterbium ion (Y_{b+}) by removing one electron by the means of lasers. This results in the atom being kept as positively charged and have only one valence electron. The ion is then fixed at a position in a linear ion chip using oscillating voltage to maintain its position. This varied oscillation of electromagnetic voltage also restricts the interference of noises and decoherence.

The ions can be chained linearly with each other to make it a multi-qubit system. This also allows reducing the overhead of maintaining communication between the ions as they aren't connected by a physical element like a wire which adds to the noise. The transformation using logical gates and reading/collapsing of ions to the final state of trapped ions is done using lasers. These machines can be accessed using Microsoft Quantum Azure (currently access only to private members) or Amazon Web Services(AWS) Bracket. [22]

2.8.6.3 Rigetti

Quantum computers by Rigetti are designed to keep superconducting qubits based processors and the chips are made of aluminum, indium, and niobium that provides its ultra-low-loss property. The architecture of the processor allows them to make the system ready for NISQ type of computers that also offer fast gate, low latency, and faster execution of programs. The qubit is made of Josephson junction aligned parallel to each other. These qubits are then coupled with capacitors and wires that allow them to create circuitry for manipulating, encoding, and measuring (reading out) quantum information. For the transformation of qubit states, the system employs microwaves or DC pulses to the array of qubits unlike the usage of laser in the case of IONQ. The company provides many open

source software for programmers and researchers to work with there system. PyQuil is the python library that allows the developers to design quantum programs(Quil Programs) which are compiled using Quil Compiler and can be run on a simulator known as QVM or on Rigetti's Quantum computer via cloud [23]. QVM is equipped with simulating the quantum program with or without noise and has optimizations like just-in-time compilations that allows us to compile and simulate large programs.

2.8.6.4 IBM

IBM is the first company among the lot that provides a complete ecosystem for its customers to accommodate quantum usage. Its allows cloud-based access to the public to its quantum computers with qubits ranging from 5 to 15 qubits. The company also provides premium access with more advanced tools and bigger capacity qubit systems to its customers via IBM Q Systems. IBM extends in quantum research by measuring a quantum system based on the 'volume'. A system's 'Quantum Volume' allows the company to set the benchmark on the number of complex circuits the system can handle without considering the performance of the individual qubits. The latest offering from the company has 53 qubits with a quantum volume of 8, located in Rochester, and another system with 27 qubits that has a quantum volume of 32. Fig 2.9 and 2.10 provide the qubit placement on the QPU [24].



Figure 2.9: IBM's qubit placement for machine in Melbourne (15 qubits)

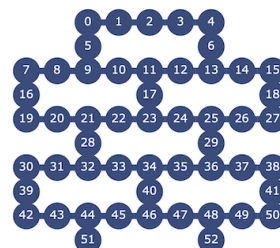


Figure 2.10: Biggest Quantum computer offering by IBM

IBM provides many ways to interact with their quantum technology. IBM Quantum Experience [25] allows the user to build scalable quantum programs with or without coding that can be executed in a simulator or physical Quantum machine.

Circuit composer from IBM is a new graphical tool that allows the users to create circuits and visualize them using Q-sphere. The circuit can be run on the simulator or real Quantum computers. The Q-sphere is not the same as the Bloch sphere and its a new way of visualizing any multiqubit (limited to 5 qubits) on a computational basis. The color of the node signifies the phase angle while the radii of the node signify the magnitude of the state. Fig 2.11 shows a 2 qubits system which is in the

superposition after applying Hadamard Gate. It also provides the state vector and the measurement probabilities for a better understanding of how the qubits are changing based on the universal gates. Additionally, the constructed circuit is automatically converted to the QASM compiler ready code.

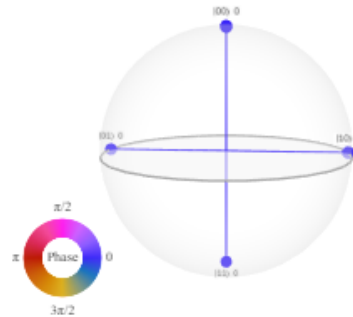


Figure 2.11: Qsphere representation of 2 qubits

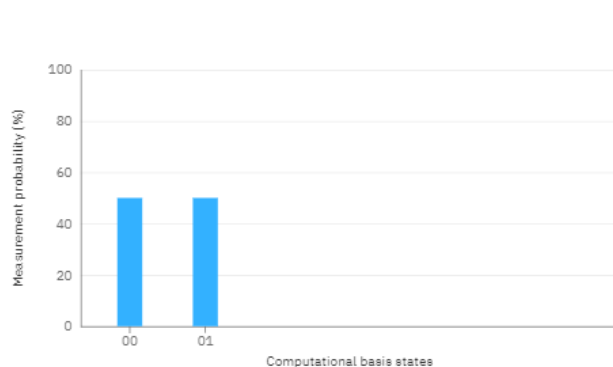


Figure 2.12: Measurement Probabilities

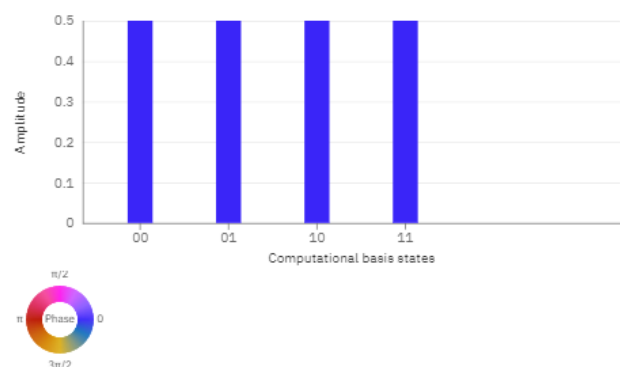


Figure 2.13: State Vector

In terms of Hardware access, IBM currently supports 20 Quantum Computers in total with some restricted to premium access for industrial and research partners. All the access to these machines can be done on the cloud using IBM Quantum Experience. The backends of IBM are the interfaces that are mainly of two types - classical-quantum simulator or quantum system. Each of these backends provides the details of the quantum system like error rate, properties related to the qubit, and so on.

They provide Qiskit, an open-source Software Development Kit that allows the programmers to run the programs on simulators or real quantum computers that can be accessed via the cloud. IBM also provides a range of computers to choose from based on the qubit requirements of the program. The developers are also exposed to the circuit based GUI that allows them to test and build quantum circuits for usage in quantum programming. IBM also provides access to the underlying hardware architecture selection where the developers are allowed to choose between superconducting qubits or trapped ion. QASM is a quantum assembly level language where all the programs written in qiskit

are internally converted. The qiskit libraries can be used in languages like python, javascript, or swift.

The OpenQASM simulator is designed to simulate up to 32 qubits without any noise. They also provide noise models that can be introduced by the programmers in their circuits. This will help them study and correct their circuits. They can run circuit depths up to 300 circuits. The simulator can also be run on local machines using Qiskit. Qiskit provides 4 main elements- Terra, Aer, Ignis, and Aqua. Each of these elements/libraries consists of functionalities that provide the complete quantum experience that an end-user expects.

With Terra consists of functions that bridge Qiskit with external Quantum Hardware. Aer provides the high-performance simulators created using C++ that simulate Terra complied circuits. Ignis provides the users with circuits that will best fit their experiments adding minimal error. They are mainly used for generating error correction codes for the circuits. Aqua consists of algorithms that can be useful in fields like AI, finance, Chemistry, and so on. These algorithms can be executed in Terra or Aer and provide the building blocks on the advantages of using quantum computers in those fields. Table 3.1 sums up some of the libraries.

| | |
|-------|---|
| Teraa | qiskit.circuit - Implementing circuits that consist of gates and qubits qiski.pulse - Implementing low-level circuits for reducing errors qiski.transpiler - Consists of pass manager that allows to user to optimize and get better circuits qiski.providers - Used for executing the circuit in Provider, Backend or Job qiski.visualization - Provides users with tools to visualize quantum circuits or states. |
| Aer | QasmSimulator - Provides Multi-Shot execution of circuits StatevectorSimulator - Provides final state vector after single-shot execution of circuits UnitarySimulator - Provides final unitary matrix after single-shot execution of circuits |
| Ignis | Circuits - Generated code for a particular ignis experiment Fitters - Provides modules that analyze the results from the experiment and fit as per real systems Filters - They allow to mitigate errors after finding them from executed Ignis experiments |
| Aqua | qiskit.chemistry - Algorithms that can be used in the field of Chemistry qiskit.finance - Algorithms that can be used in the field of Finance qiskit.ml - Algorithms that can be used in the field of Machine Learning qiskit.optimization - Algorithms that can be used in the field of optimization problems |

Table 2.3: IBM Qiskit Modules

IBM includes *noise* module in their Aer package that allows us to study the effect of noise on the

outcomes of a circuit. The users can create their custom noise models or use a basic one provided as part of Qiskit. These noise models can emulate gate errors, readout errors on each qubit along with T1 and T2 relaxation time of them. These simulations of noise are still an active research field and provide simulations of near to expected/seen noises on the quantum system. The basic noise model can be added as an extra parameter while executing the circuit as per the Table 2.4.[26]

| Parameters for Noise | Description |
|----------------------|---|
| noise_model | Tells the QasmSimulator's run method about the type of noise model to simulate |
| basis_gates | Allows to access basis gates for the noise model that is required for the simulator |
| coupling_map | Creates a Qobj from the real device so that compiler can act as the same |

Table 2.4: Noise Model Parameters

Documentation and support They provide an open-source software development kit called Qiskit (Quantum Information Software Kit) which provides the programmers to interact with the quantum system using the OpenQASM language [27]. They have a well established and supported quantum community that provides help via Slack. The documentation along with tutorials are hosted on GitHub [28].

2.8.6.5 Microsoft Quantum Computer

Microsoft is one of the youngest of all the major companies that are in the quantum computer arena. Though they are rapidly developing and innovating which other companies are finding it hard to compete with. Microsoft provides a full-stack approach for developing quantum solutions. There are three stages to it - Quantum Algorithm, Quantum Machine Code, and Quantum Hardware. Fig 2.14 shows the basic architecture of the complete system that provides a scalable quantum platform.

The platform consists of Q# which is the dedicated quantum language by Microsoft to run and execute quantum programs. Quantum Algorithm Libraries provide users with optimized and extensible algorithms for their use. Classical Host program handles the classical processing of information required for execution. Classical runtime can be on the local machine or executed on the cloud. The cloud-classical runtime when ran alongside Tracer systems, allows the user to execute large quantum algorithms and returns an analysis of the algorithm. The analysis includes the number of operation gates, noise effects, amount of parallelism, and so on. This complete system sits on top of Cryo-Control that uses lab equipment to control the Quantum computer. Qcodes is one of the open-source Cryo-control systems.

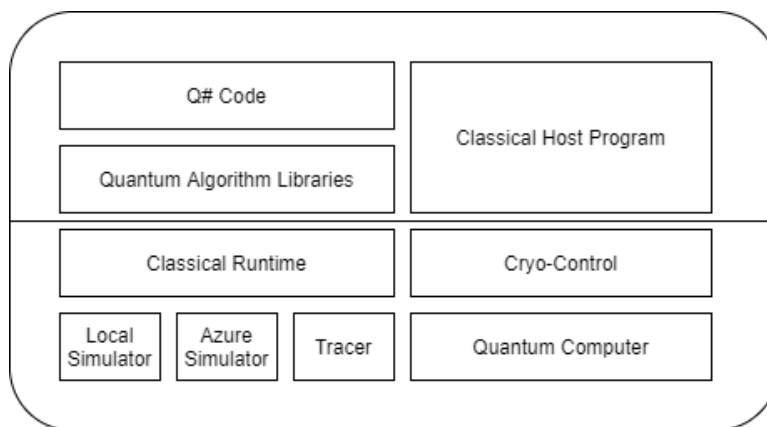


Figure 2.14: Overview of Microsoft's Quantum Platform

On the Quantum Hardware perspective, Microsoft has tried a different approach for developing their quantum systems by using topological qubits based on Majorana Fermions. By using this topological architecture, Microsoft claims to have made stable and faster qubits for its scalable quantum computer. Fig 2.15 helps us visualize what happens inside a qubit. The image is of a nanowire such that the electrons line up together. Each rectangle is an electron with a real and imaginary part placed in pairs such that one has a spin up while the other has the opposite spin. When these electrons pairs are placed in extremely cold temperatures they become superconducting and act like photons. These pairs are also known as Cooper Pair[29]. Microsoft uses similar to a material that uses high-spin orbit coupling that pairs the electrons in a 45-degree angle (as seen in the last segment of Fig 2.15). The real and imaginary electrons left without a pair at the two ends of the wire form a single electron. The qubit build in this way is extremely resilient to noise. If an electron is corrupted at one end, there is still a part of the information that is present on the other. The system has this type of qubits with some redundancy will result in a system that has longer coherence qubits[30].

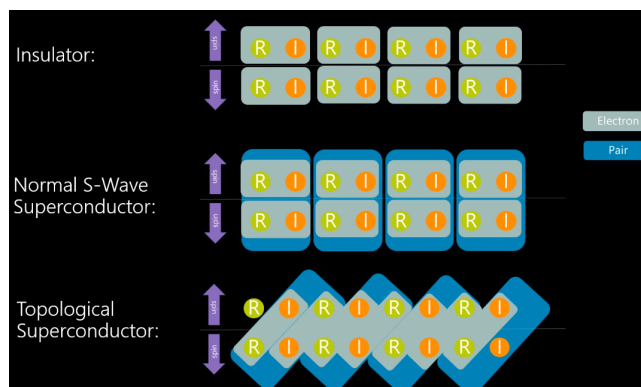


Figure 2.15: Electrons in a Topological Qubit

The seriousness of the company to provide solutions using quantum techniques can be conferred by the development of QDK. Quantum Development Kit is an open-source full-stack software that is built for developers to write, test, and debug their quantum program. Microsoft's quantum dedicated language, Q#, is a high-level programming language and its libraries are available in Python or .NET for inter-portability and also have full integration in the Visual Studio umbrella of software along with Jupyter Notebooks. The company currently provides a simulator that can be run locally using the QDK. A quantum community event hosted by Microsoft showed the possibility of accessing the quantum computer on the cloud via Microsoft Quantum Azure. The unique element about the Quantum Azure is that it also allows its users to access to other hardware and software from companies like Honeywell and IONQ (Quantum systems with trapped ion hardware); 1QBit (Quantum software solutions). Currently, access to these computers is not available. [31]

The QDK is designed to simulate 4 types of simulators based on the requirements, namely - Full State simulator, Simple resource estimator, Trace-based resource estimator, and Toffoli simulator. The Full State Simulator is based on the LIQUi|> [32] architecture which can simulate up to 30 qubits on a classical computer with 32 GB RAM and is used for executing along with debugging of quantum programs. The Simple resource estimator is built upon the Quantum trace simulator that provides the resources needed for running the algorithm written in Q#. Additionally, the Quantum trace simulator (Trace-based resource estimator) provides debugging of the classical part of the algorithm. The resources are estimated by executing the algorithm without any simulations on the quantum system. Microsoft also provides a simulator which only supports gates like X, CNOT, and multi-controlled X gate. This special simulator is more restrictive than the full-state simulator but allows the programmers to use up to a million of qubits.

The quantum programs once are written need to be structured in a particular format for the host machines to be able to execute the programs. Fig 2.16 provides an overview of the whole process. Q# operation consists of user-defined types, operations, and methods that the algorithm needs to perform and are encapsulated in a Statement block. These blocks are enclosed in Q# namespaces. Every quantum program can contain any number of namespaces, provided they are not nested inside another namespace. These namespaces can also be overloaded with the same name provided they don't have any function or operation in common. These namespaces are combined to form a file with extension *.qs file which is executed.

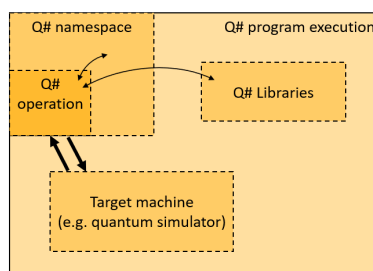


Figure 2.16: Host program general execution model[1]

Documentation and support The company provides the community for contributing to the QDK. There is extensive support on learning of quantum computing via Microsoft Katas. The open-source project can be found on GitHub [33]

2.8.7 Performance of Quantum Hardware

| Company | Rigetti | IBM | Microsoft |
|----------------------------------|---|--|---|
| Programming Language | pyQuil | Qiskit | QDK |
| Quantum Language | Quil | Qasm | Q# / LIQUi> |
| Quantum Hardware Size | ASPEN - 7 with 28 qubits AGAVE with 8 qubits | 5qubit 11 qubit\ | expected 8 qubit |
| QPU types | Superconducting Transmon Qubits | Superconducting Qubits | Topological Qubits |
| Host Classical Languages | Python | Python Javascript Ruby | Python .NET |
| Simulators for classical systems | PYQUIL - python based | QISKIT - python based | QDK - Q# or .net |
| Features | 1. Noise simulators 2. Slack support 3. Compiler specific to topology | 1. Qiskit libraries 2. Circuit Visualizer 3. Slack support | 1. QDK libraries 2. Dedicated Quantum language 3. Extensible algorithm implementation |
| Execution Platform | Cloud Based simulators | Local machine simulators Cloud based Simulators and Quantum Computers | Local Machine Simulator |

Table 2.5: Comparison between Rigetti, IBM, Qiskit

Chapter 3

Design

This section provides us with a high-level overview of the architectural design of the quantum computer and the technique it uses to transform the image into quantum states along with manipulating these states to perform image processing. This section also provides the reasoning for choosing IBM over its competitors for the execution environment and provides a complete workflow that helps in understanding the implementations discussed in Chapter 4.

3.0.1 Workflow

Image processing in a quantum computer involves various steps in order to achieve the goal. Fig 3.1 illustrates the basic flow of the same.

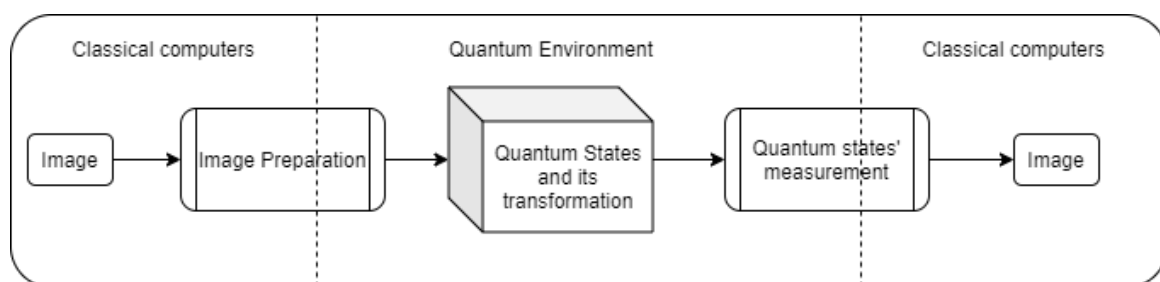


Figure 3.1: Image processing workflow

3.0.2 Image pre-processing

The first step in the image transformation is to process the image present in the classical computer to the required attributes which are based on the model selected for image representation in a quantum computers. This step is known as Classical to Quantum interface and involves pre-processing for translating pixels into quantum states. Chapter 3 describes the FRQI image model can be implemented for square images hence the conversion of images which are not square are converted by

cropping or adding padding around the sides, is one of a pre-processing step. Additionally, limitation related to the number of available qubits as per the environment selected results in additional complex programming in the Classical to Quantum interface.

3.0.3 Quantum computing

Once the pixels are converted from their classical version (pixel) to quantum states (qubits), the states can be manipulated by using a single or combination of universal gates. The group of gates form a circuit and transforms the properties of the qubits when passed through them. There are a range of algorithms that researchers have found that utilizes the quantum computer to perform image processing like image segmentation, geometric transformation, encryption, and watermark technique. Most of the image processing in the FRQI image representation model involves changing the θ which in changing the rotations using gates like NOT, CNOT, Toffoli Gates, and so on.

3.0.4 Image Retrieval

Once the transformation of the image is done we need to be able to retrieve these quantum states back into the format that can be understood by a classical computer. The Quantum to Classical interface involves the process of measurement where the qubits are collapsed from their superposition for their finite value. These values are then interpreted by the classical computer in-order to reconstruct the image after being transformed.

3.0.5 Design Choices

This section discusses the implementation approach taken for the study that comprises simulator comparison, ease of coding, compiler implementations, hardware implementations, and access to actual quantum computers. We compare IBM and Microsoft quantum implementations as those are the most developed of the lot and have different types of implementations for their hardware.

3.0.5.1 Rigetti

Rigetti is being considered for this study due to its limitations on the simulator front. QVM implementation is not build for the local machine and hence can only be executed on the cloud via an API access key. Additionally, there are limitations on the execution of larger circuits on the QVM has the maximum time allocated for the execution of a task is 1 minute with the current API key.

3.0.6 Microsoft vs IBM

Microsoft and IBM both provide a great full-stack solution for developing quantum solutions.

3.0.6.1 Hardware

IBM uses Superconducting Transmon Qubits for its QPU while Microsoft chooses to select topological qubits made of Majorana Fermions. The biggest advantage of IBM over Microsoft is access to a range of quantum computers. Microsoft has not provided any access to their machines hence the performance and advantages of using the topological qubits can not be realized.

3.0.6.2 Quantum Simulators

IBM provides both local and cloud-based simulators for executing quantum computers via their back-ends belonging to Terra and Aer modules. On the other hand, Microsoft currently provides only local machine simulators for execution.

3.0.6.3 Qubits

The number of qubits allowed to simulated is directly proportional to the design as well as the system that is being used for execution. IBM provides various machines that can be accessed via the cloud or ran locally. IBM Simulators are designed to simulate up to 32 qubits with max circuit depth of 300 and limitations of 8092 repetitions. Microsoft simulators are designed to support up to 30 qubits. One advantage of Microsoft is that it allows to debug the algorithm and also provides Resource estimators along with Tracers for the algorithm that needs to be executed in the environment.

3.0.6.4 Visualization

IBM provides an extensive set of libraries for visualizing qubits like plotting histograms, visualizing qubits using Bloch Sphere and Q-sphere. While Microsoft has no such library for visualization purposes yet.

3.0.6.5 Circuit

In chapter 2 we looked at the usage of Circuit builder that allows a GUI way of building circuits. These implementations are useful for checking the circuit without the overhead of coding. QDK doesn't provide any support related to building circuits.

3.0.6.6 Programming language

IBM uses Qiskit to create circuits that get converted into OpenQASM language for getting executed in the Quantum system. Microsoft provides its language for creating programs that are very similar to C# programming. Both of them have libraries in python language. As described in Chapter-2, the file structure required for executing quantum computers is simpler in IBM when compared to the QDK structure.

3.0.6.7 Noise Model

IBM also provides the option to add custom or ready-to-use noise models to the simulation while Microsoft doesn't have any support for this yet as the Topological qubits that are the hear of Microsoft's QPU is expected to mitigate the noise using redundancy of the qubits or engineering of the hardware. There is also a possibility to simulate the noise in other hardware that is part of Microsoft Quantum Azure. Currently, these systems have limited access to their partners which limits the knowledge.

Chapter 4

Implementation

This chapter provides in-depth implementation details of the FRQI model using IBM Qiskit and IBM's backend simulators. We will also incorporate the design decisions that are outlined in Chapter 3.

4.1 Technical Setup

The implementation of the Quantum circuit was done using Qiskit with python 3.5+ in the Visual Studio Code. Additionally, libraries like NumPy, Keras, MatPlot, resizeimage, and PIL were used for the classical processing of an image. For more details about the installation of qiskit library and API access see Appendix A.

4.2 Hardware Setup

This study was implemented on the local as well as cloud-based systems. The local setup had the configuration of - Intel Core i7-3632QM, 64-Bit with 2.20 GHz, and 8 GB RAM. The cloud-based implementation was done on the Jupyter notebook provided by IBM in its cloud.

4.3 Images

MNIST image The MNIST dataset consists of 70,000 grey-scale images of handwritten digits. These images have been size-normalized and centered based on the mass of pixel values with 28X28 size. The dataset is mainly used for benchmarking various machine learning algorithms. [34]. Fig 4.1 shows a subset of images that are present in the MNIST dataset.

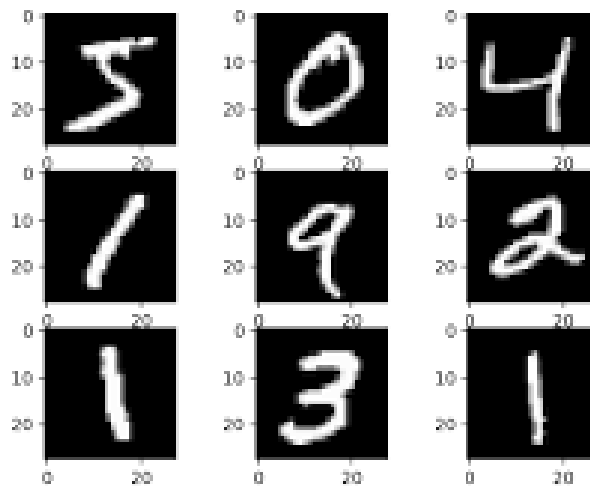


Figure 4.1: Image processing workflow

Image generated via python In order to test out the color variations , I created an image of required size by assigning RGB values to each pixel. Below mentions the function that returns an Image with the pixels as the RGB values and Fig 4.2 illustrates the image generated.

```
def generateImage(width, height):
    pixels = numpy.zeros((width,height,3), dtype=np.uint8)
    width_box=int(width/2)
    height_box=int(height/2)
    # red box on left top corner
    pixels[0:width_box, 0:height_box] = [255, 0, 0]
    # green box on right top corner
    pixels[0:width_box, height_box+1:h] = [0, 255, 0]
    # blue box on left bottom corner
    pixels[w_bound+1:w, 0:height_box] = [0, 0, 255]
    # grey box on right bottom corner
    pixels[w_bound+1:w, height_box+1:h] = [128, 128, 128]
    image = Image.fromarray(pixels,'RGB')
    return image
```

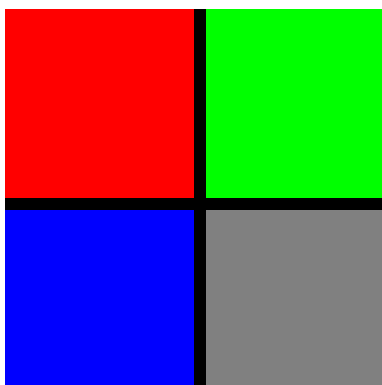


Figure 4.2: Code generated Image

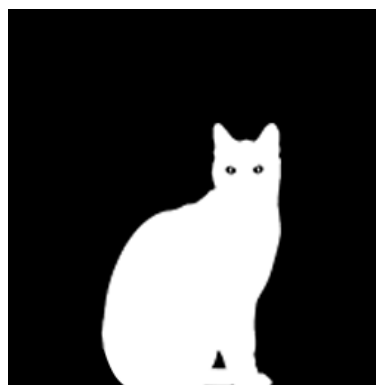


Figure 4.3: Cat Image

External Image The image of different pixel quantity is also used for testing the performance of the circuit by reducing the pixel quality of an image. Fig 4.3 is the image of size 120X120 that is used in this study.

4.4 Classical - Quantum Interface

The first step in image processing as outlined in Chapter 3 is classical to a quantum interface. This step involves any pre-processing of the image that is required to be done on the image before converting it into quantum states. Fig 4.4 illustrates these steps. The image is taken and normalized such that the pixel value lies between the range of 0 and 1. The qubits are initialized along with classical registers for measuring those qubits. These initialized qubits are then transformed into circuits that allow the encoding of the qubit.

The normalization steps involve restricting the image size to 32X32. The main reason for this size is to keep a check on the number of qubits and quantum gates required to encode the image. In Chapter-2 we discussed FRQI, where it requires $\log N$ qubits for representing a $N \times N$ size image. Hence for 32X32 image has 1024 total pixels and $\log_2(1024) = 10$. Further, an extra bit is required to store the color intensity which brings the total number of qubits used for this study to 11.

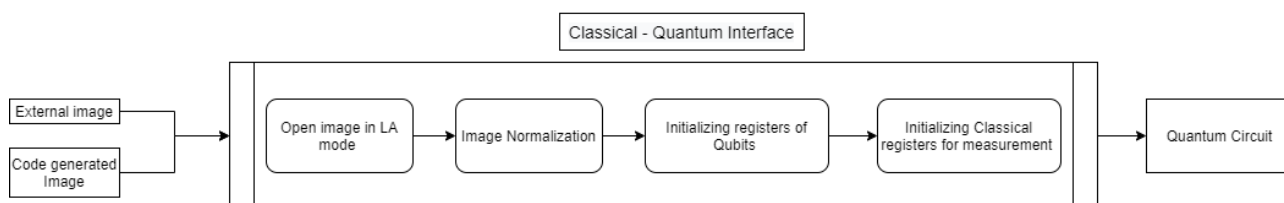


Figure 4.4: Image processing workflow

Python provides libraries that allow the programmer to resize any image either by cropping, resizing

cover, and many more. Refer to Appendix A for the library details. Fig 4.5 is the original image (120X120) and 4.6 is the resize-cover image (32X32) which illustrates the difference the image quality due to decrease in pixels.

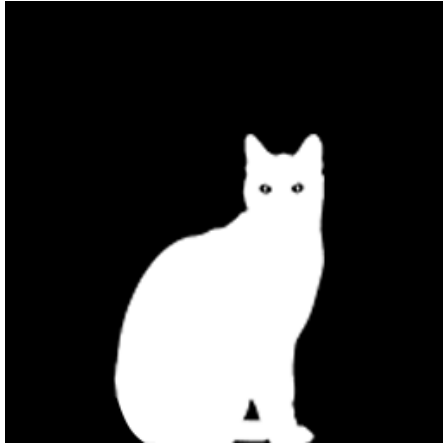


Figure 4.5: Cat Image

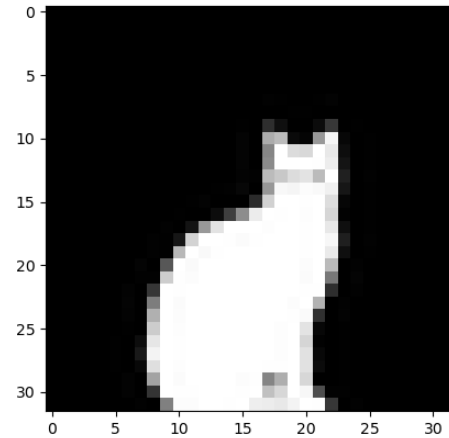


Figure 4.6: Resized cat image

4.5 Quantum interface

This step in the workflow consists of the creation of circuits using quantum gates and passing the qubits through it to transform their states. The transformation is reflected in the state vectors in the qubits. Since all the qubits are in a superposition, the measurement of these qubits will eventually result in collapsing in one of the two states. These measurements are then fed back to the classical registers kept from the previous stage. Fig 4.7 shows the basic working of this module. The inspiration for the FRQI circuit used for this study has been implemented as a Hackathon Challenge conducted by IBM which is one of IBM's ways to engage more developers and attract them towards quantum technology.[\[35\]](#)

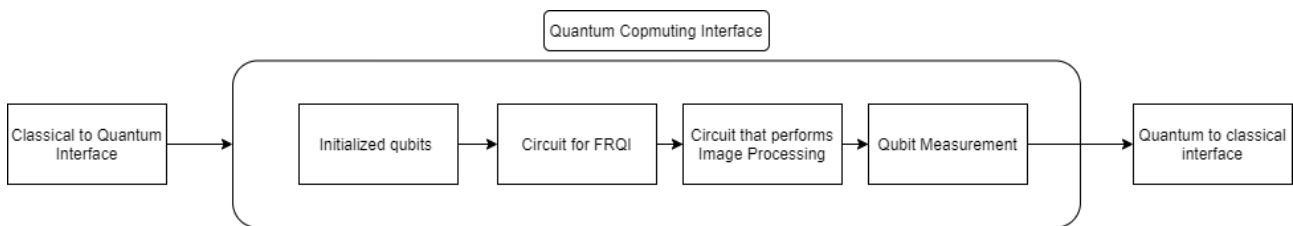


Figure 4.7: Quantum Computing Interface

The FRQI circuit is constructed using universal gates that rotate the state vectors to encode the information of the pixel and color intensities. Further, image processing involves changing qubit states to achieve the desired result. For instance, to rotate the image around its angle, we need to flip the qubit states from their current FRQI modeled state which can be done by adding another Pauli-X gate

on the circuit before measurement. Fig 4.8 is the original image and Fig 4.9 is the rotated image along that is generated after the classical post-processing of inverting colors using python PIL library's `invert` function. The circuit can also be modified to implement the edge detection on the encoded image. by adding the QHED algorithm circuit to the FRQI model.

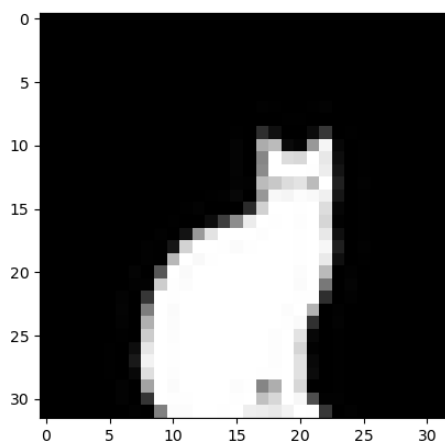


Figure 4.8: Cat Non-Normalized Image

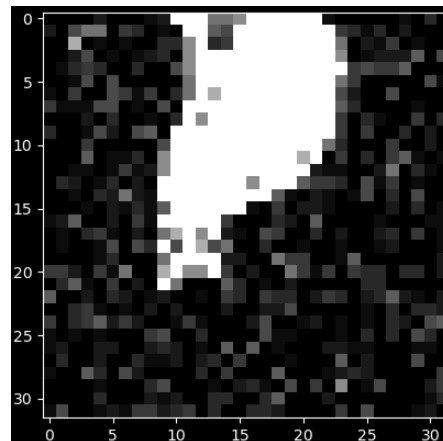


Figure 4.9: Rotated cat image

Once the circuit is constructed, we need to select a backend based on the requirements. The execute function of Qiskit requires a circuit and a backend with other parameters that the user can set or are provided by default. One of the parameters which allow executing the circuit repeatedly for sampling is *shots*. This parameter is set by default to 1024 and can be increased to a higher number depending upon the machine being used. For other parameters that can be added to execute see Appendix A.

Measurement of the qubits states collapses their superposition and provides a finite state. This is also part of the circuit and can be done using *measure* function of the QuantumCircuit object. These measurements are read out to the classical registers that are part of the circuit.

The execute command returns a *Job* object that encapsulates the details of the circuits as well as other relevant information for the executed task. The most important of the out of all the job functions are the `result()` and `status()` functions. The `status()` provides the details on the status of the job while the `result` function provides the results. The tasks in Job objects are asynchronous but they are changed to synchronous when the `result` method is called on them. By switching to synchronous we can be assured that the task will finish before moving to the next one.

The `result()` method of job object returns a *result* object which includes a function `get_counts(circuit)` which can be used to get an aggregated score of the circuit that was executed by the job. There are other parameters like `get_memory()`, `get_statevector()` and `get_unitary()`.

```
job = execute(quantumCircuit, Aer.get_backend('qasm_simulator'), shots=numOfShots
              )
```

4.6 Quantum-Classical Interface

After the circuit is executed, the information is stored in the classical registers. To reconstruct the image, we need to do post-processing on this data. The steps in reconstructing the image are reading the values from the `result_count` and averaging based on the number of shots used for running the circuit. Further, the steps are taken to de-normalize the image. Finally, using image libraries in python to display and store the image.

4.7 FRQI on Larger Pixel Image

This study also tried to implement FRQI for images that have a higher pixel density without python-recover-resizing. For this implementation, a picture of size 320*320 was taken and divided into 100 equal boxes of size 32X32. These boxes were then individually processed through the FRQI system and results were combined to form the image. The additional overhead of making the input image a square and frame-wise modeling added more computational complexity to the code.

4.8 Adding Noise Model

This experiment was also done by adding a noise model to the FRQI model using Qiskit's noise simulator to understand the effect of noise on the image illustrated in Fig 4.3. Below code shows the extra parameters that are required to add the noise that is generated by IBM's 16 qubit quantum system and simulate it in the local machine.

```
backend = provider.get_backend('ibmq_16_melbourne')
#Get noise model for the specified backend
noise_model = NoiseModel.from_backend(backend)

# Get coupling map from backend
coupling_map = backend.configuration().coupling_map
# Get basis gates from noise model
basis_gates = noise_model.basis_gates

result = execute(qc, Aer.get_backend('qasm_simulator'), shots=numOfShots,
                 coupling_map=coupling_map,
                 basis_gates=basis_gates,
                 noise_model=noise_model).result()
```

Chapter 5

Results and Discussion

This chapter presents the results obtained from the study and contrasts the findings from the classical approach that helps in deriving insights on the research question being addressed. FRQI is a model that uses probabilistic distribution for retrieval of qubit state. Hence the image retrieval from the quantum state involves the repetitive executions of the circuit for which the quantum system provides aggregated probabilities at the end. These repetitions can be set as a parameter named "shots" which is specified during the execution of the jobs.

5.1 Image retrieval

The FRQI model allows us to build a quantum state that represents the pixel properties in the form of qubits. During the study, different images were used for encoding from their classical pixel format to qubits and retrieving those quantum states back to a pixel.

Cat Image The cat image used for the study was a 120X120 pixel image which was reduced to 32X32 for FRQI model experimental setup. The experiment was done on the same image multiple times based on the number of correct pixels we could retrieve from the quantum state. Fig 5.1 is the image before the image is normalized. Fig 5.2, 5.3 and 5.4 are the images that were retrieved after 1 million, Ten thousand and 4500 shots of circuit for the image respectively.

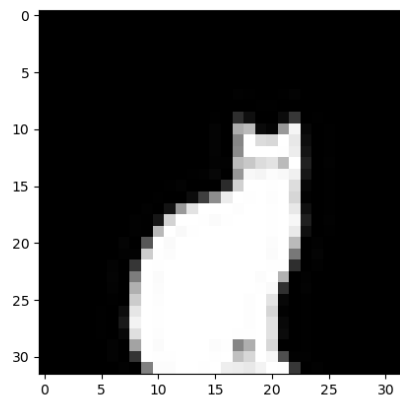


Figure 5.1: Original Cat Image

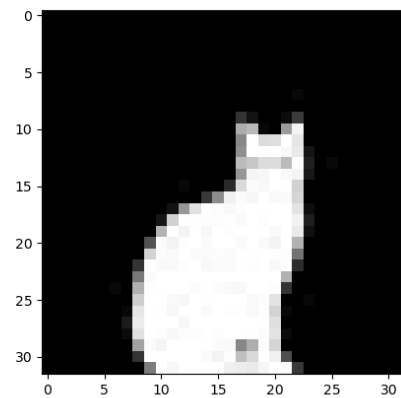


Figure 5.2: Reconstructed cat image 1

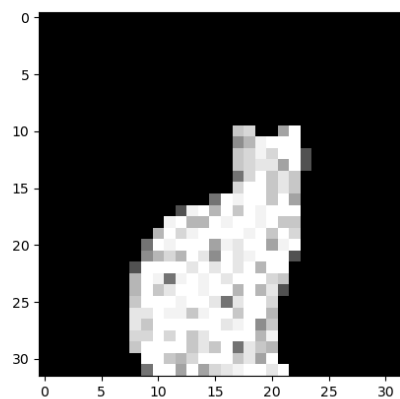


Figure 5.3: Reconstructed cat image 2

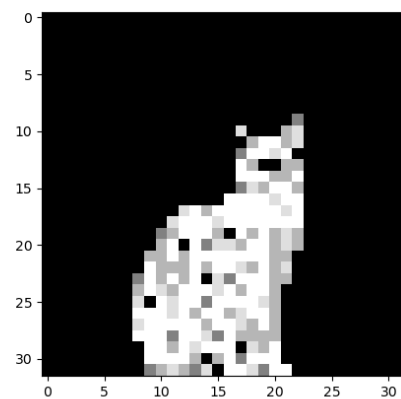


Figure 5.4: Reconstructed cat image 3

Code Generated Image The python generated image was used to see the effect of the presence of colors in the images and how the image model performs on the retrieval of this information stored in the qubits. Fig 4.2 is the image generated while Fig5.7 is the retrieved image from the original image illustrated by Fig 5.6. When compared to a black and white image, like MNIST or Cat Image, the recovery of pixels containing colors is not as accurate. The experiment conducted by me could only recover 11.6% of the original image after running the circuit for 1million times. There is a possibility of improving the recovery percentage after running for more iterations, but due to the "shots" limited to 1 million on the simulator, this was the best result that could be achieved in local simulation.

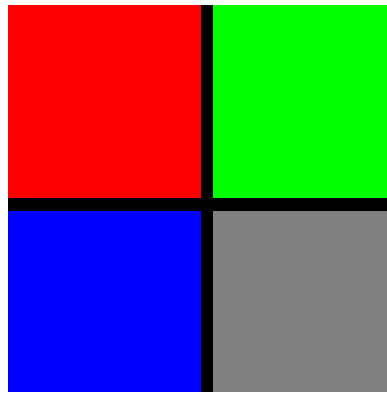


Figure 5.5: Original image

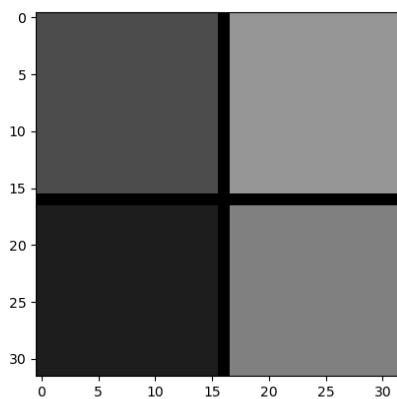


Figure 5.6: Greyscale Original image

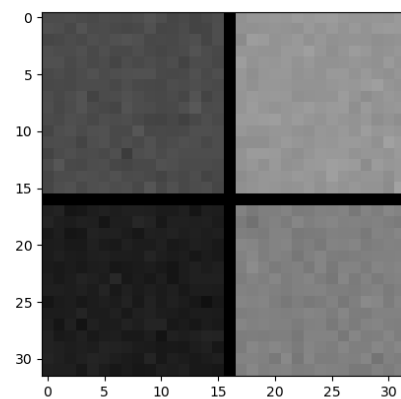


Figure 5.7: Reconstructed image

MNIST A different set of digit images from the MNIST were used for this study of FRQI image representation. The images being of a smaller size requires lesser qubits and performed better with circuit depth counting to 15604. The depth provides the critical path length of the circuit that consists of gates. A circuit is optimized if the gates used are lesser which in turn reduces the circuit depth. The MNIST digit image's data was almost half when the depth is compared to CAT image.

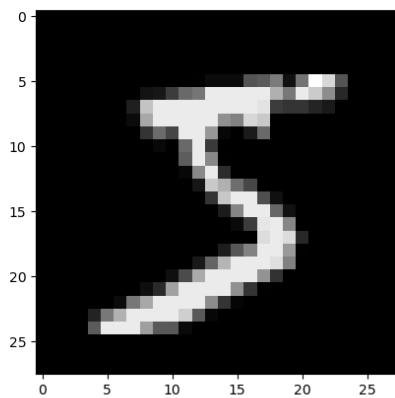


Figure 5.8: Original image

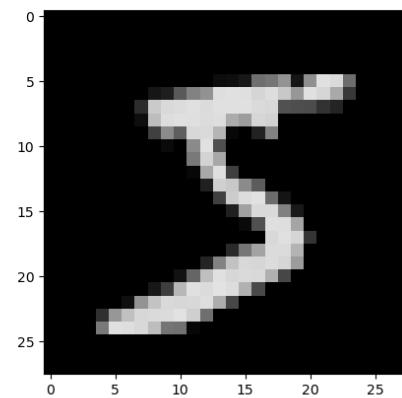


Figure 5.9: Reconstructed image

5.2 Geometric Transformation

Image Rotation The circuit was modified to rotate the image of Fig 5.1 which is shown in Fig5.10. The image was also processed using PIL in python to invert the color of the pixels as the rotation circuit caused the inversion of pixel colors in the original image. This is illustrated in Fig 5.11. The rotation of the image using this model was successful for only some images while the amount of retrieved image was very low. The information for some pixels is lost due to the rotation of qubits by the quantum gates.

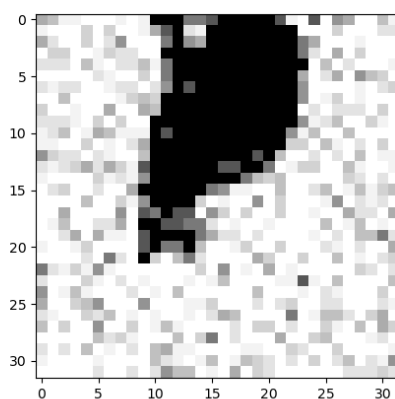


Figure 5.10: Rotated image

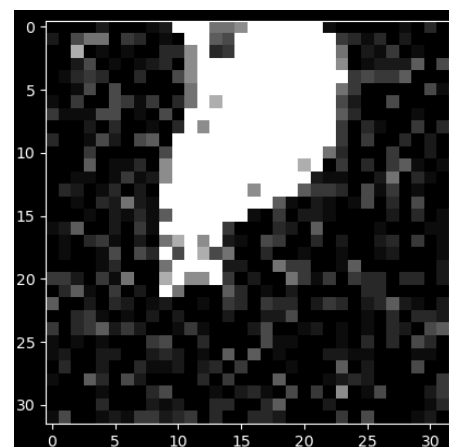


Figure 5.11: Reconstructed image

Sliding Frame The study was also attempted to perform the FRQI on a larger original image. The solution was to perform the encoding for the FRQI model on the larger image with a window size of 32X32 and concatenating these windows into a new image of the same size as that of the original. Fig 5.12 shows the image retrieved of size 320 X 320 using the sliding window method.

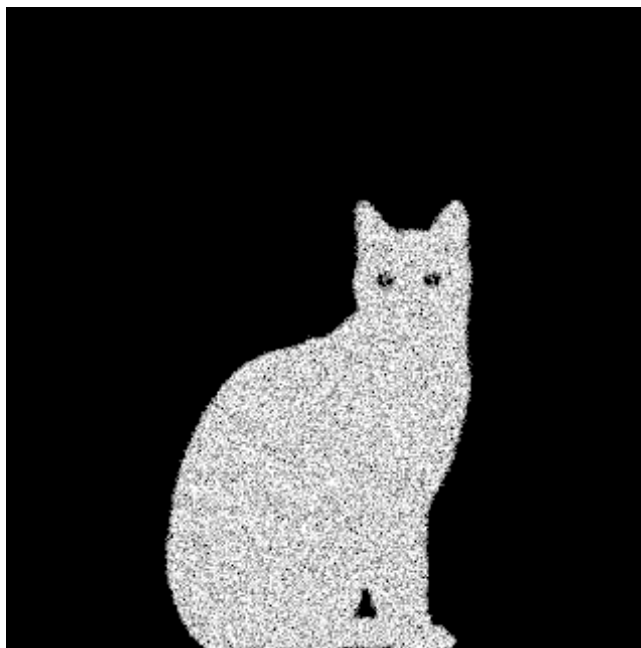


Figure 5.12: Reconstructed Large Image

5.3 Analysis and Discussion

This section provides an analysis of the results obtained during the study. The main motive of this study is to encode and retrieve the image from the FRQI model. The best performance was seen for Fig 5.1 and Fig 5.8 where the color density was limited to 1-2 colors. The model failed to retrieve images in terms of their pixel properties if there were more colors in the pixel. One of the reasons that can be inferred is related to the encoding of colors in θ which ranges from 0 to $\pi/2$. Hence having a smaller range to save so many colors requires the quantum computers and simulators to provide extreme precision on the rotations which currently none of the systems provide.

Rotation of the image was successful for the images except for the Fig 5.8 which is illustrated in Fig 5.13. The image rotation circuit acts on each qubit with an Xgate to rotate the image around the center. For the MNIST data, this operation resulted in the loss of the pixel's position. Due to limitations in the debugging and visualization tools that are available for quantum programming, it was difficult to find the reason for this behavior of the circuit.

IBM's Qiskit allows the implementation of a noise model but doesn't provide any details on the approximation of the time required to simulate the circuit with noises. The experiment was tried to add noise generated by 'id', 'cx' and 'u3' gates ran for more than 78 hours but ended with Memory issues.

This is something that can be executed in a much powerful machine than the one used currently to analyze the result.

The implementation of the sliding window to apply FRQI model on a larger image was a very good example in the way to show the amount of noise that the simulator tends to make. If this size of image is required to be encoded in one go, the circuit depth could be close to 10X times the current depth which is not possible to be used for any system that is available in the market.

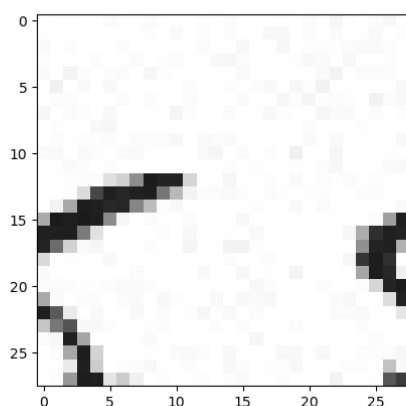


Figure 5.13: Failed Digit Rotation

The study was also attempted to run on actual quantum computer using the IBM Quantum Experience which was unsuccessful due to the limitations imposed on the circuit depth that comes with each quantum system. The 'ibmq_16_melbourne' is highest available qubit system (15 qubits) can only handle upto 300 circuit depth. The FRQI circuit which was used for the study was having a circuit depth of 376 for the smallest image of size 2X2. Hence the performance for this model could not be found out on a real Quantum System.

In the course of this study, there was also an attempt to do a Quantum Edge detection discussed in Chapter-2. The circuit was implemented but the edge detection could not perform as per requirement and the image from the circuit could not be retrieved. The visualization of this circuit in addition to the FRQI model was beyond the tools provided for error correction in the Qiskit library. In addition to this, there were also limitations on the performance of the circuit as the circuit depth increased to 40,000. In [36], the authors propose an algorithm that uses SVM to detect edges on an image. As currently most of the quantum computers are being optimized for machine learning algorithms, the results from this approach can be analyzed in the future.

Chapter 6

Conclusion

Quantum Computing has been around for many years but with the failing of Moore's law in the unforeseen future, and the need for faster computation has fueled the inspiration to develop these types of computers. One thing that this study has made clear is that quantum computers are not meant to replace classical computers. Currently, all the hardware that has been build for this set of computers are not portable and require lab equipment to control the quantum system. Classical systems have been evolving at a pace such that it has become error-free in computing. The advantages that current quantum systems provide can be emulated in some classical supercomputers with the same or little more time.

This study provides a primer on what Quantum Computing is along with various architectures that constitute the heart of Quantum computers. The study discusses the QImP (Quantum Image Processing) with a focus on the FRQI model. With the availability of simulators and physical machines that have been made available to the practitioners, practical implementations on them have been a challenge. The next section discusses the limitations that were found out during this study and suggested future works.

6.1 Limitations and suggested Future Work

In this study, we looked at the implementation of image representation in the FRQI model. The model is built on universal quantum gates and encodes the pixel position and color intensities on the state vector. A probabilistic approach is then chosen for retrieval of the quantum states. The presence of noises as discussed in Chapter -2 play a vital role in the outcomes. In a real Quantum Computer, coherence time is the total time that the qubit can be controlled before it gets destroyed. When there are more gates involved, there is a tendency for the algorithm to not execute correctly if the noises of

the gates add up more than the coherence time. Additionally, we loose on the tractability of the pixels once it's encoded in the circuit (using θ), which doesn't allow us to check on the states if required. The researchers are constantly trying to find a better way of representing an image. One of the recent studies came up with a model called Quantum Boolean Image Processing. In this image representation model, the image is divided into three color components and uses 8 bit-planes to get MSB (Most significant bit). This allows them to have a quantum boolean version of the image in the quantum system.[37]

There is a need to set standards for building algorithms related to quantum image processing as currently most of them describe an overall idea with no implementation on any physical devices. In other words, the algorithms are described in the theory have huge gaps in the implementation of the model. This doesn't provide any advantage of using quantum systems over traditional counterparts. Additionally, there is always a comparison between a classical computer and quantum computers, the research papers do not provide comparisons between a classical image processing algorithm vs quantum algorithm. This should be done to realize what quantum systems can do for the image processing field. The study also reflects on the need for building better quantum image transformation algorithms that can not only perform what current classical algorithms can but also make better use of the quantum properties for faster computation tasks.

The issue with any of the image representation model is the lack of visualization of qubits in different states. There are some tools like IBM's circuit visualization tools but they are limited to only a certain number of qubits (5 qubits). As an algorithm developer, the resources available are very limited in terms of understanding of the internal working of qubit when there are more than two. Further, there is no way to debug the code in qiskit and hence face a major issue if the circuit is not built correctly.

In recent times, many companies have claimed to build quantum systems that are better in solving major issues related to the quantum system. These issues are classified as errors, faults due to gate noises, and qubit decoherence. But the claims have been made on specific experiments not extending to the general usage, for which these systems are being utilized. The quantum program designers and developers have been constantly working on making quantum systems error-free. There have been some attempts that theoretically allow error corrections by the means of algorithms, but implementation and actual performance of these have been yet to be analyzed [38]. Simulators are designed on top of classical computer systems with claims that the performance is "close enough" to the actual quantum computer. But these claims have been made on running the systems on a lower number of qubits. In reality, most of the quantum algorithms which are begin implemented require a large number of qubits that have unexpected behaviors.

6.2 Final Words

There has been constant development in technology that shows the relevancy of applying concepts of Quantum Mechanics to develop quantum processing units. Currently, available quantum systems are not adequate to compete with the computation power required for processing data like images to get meaningful outcomes. What many clienteles of quantum solutions providing companies have spoken about is the advantage that quantum computers provide over classical computers are in solving optimization problems. This is a good step towards the future and provides a ray of hope for building better and stable quantum systems.

Bibliography

- [1] Microsoft, “Ways to run a q# program - microsoft quantum,” url = <https://docs.microsoft.com/en-us/quantum/user-guide/host-programs?tabs=tabid-python>, docs.microsoft.com, 05 2020.
- [2] P. W. Shor, “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer,” 1995.
- [3] R. P. Feynman, “Simulating physics with computers,” *International Journal of Theoretical Physics*, vol. 21, pp. 467–488, 06 1982.
- [4] M. A. Nielsen and I. L. Chuang, *Quantum computation and quantum information*. Cambridge Cambridge University Press, 2019.
- [5] H. Abraham, AduOfiei, I. Y. Akhalwaya, G. Aleksandrowicz, and e. a. Mantas epulkovskis, “Qiskit: An open-source framework for quantum computing,” 2019.
- [6] Microsoft, “The qubit in quantum computing - microsoft quantum,” docs.microsoft.com. [Online]. Available: <https://docs.microsoft.com/en-us/quantum/concepts/the-qubit>
- [7] —, “Multiple qubits and entangled states,” community.qiskit.org. [Online]. Available: <https://qiskit.org/textbook/ch-gates/multiple-qubits-entangled-states.html>
- [8] “More circuit identities,” community.qiskit.org. [Online]. Available: <https://qiskit.org/textbook/ch-gates/more-circuit-identities.html#ccx>
- [9] A. Uhlmann, “The transition probability in the state space of a \ast -algebra,” *Reports on Mathematical Physics*, vol. 9, p. 273279, 04 1976. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/0034487776900604>
- [10] Microsoft, “Quantum circuits - microsoft quantum,” docs.microsoft.com. [Online]. Available: <https://docs.microsoft.com/en-us/quantum/concepts/circuits>

- [11] “Delta partners with ibm to explore quantum computing an airline industry first,” Delta News Hub. [Online]. Available: <https://news.delta.com/delta-partners-ibm-explore-quantum-computing-airline-industry-first>
- [12] N. Stamatopoulos, D. J. Egger, Y. Sun, C. Zoufal, R. Iten, N. Shen, and S. Woerner, “Option pricing using quantum computers,” *Quantum*, vol. 4, p. 291, 07 2020. [Online]. Available: <https://arxiv.org/pdf/1905.02666.pdf>
- [13] “Inside quantum technology,” Inside Quantum Technology. [Online]. Available: <https://www.insidequantumtechnology.com>
- [14] S. E. Venegas-Andraca and J. L. Ball, “Processing images in entangled quantum systems,” *Quantum Information Processing*, vol. 9, pp. 1–11, 07 2009.
- [15] S. E. Venegas-Andraca and S. Bose, “Storing, processing, and retrieving an image using quantum mechanics,” *Quantum Information and Computation*, 08 2003.
- [16] M. Harding and A. Geetey, “Representation of quantum images,” 2018.
- [17] P. Q. Le, F. Dong, and K. Hirota, “A flexible representation of quantum images for polynomial preparation, image compression, and processing operations,” *Quantum Information Processing*, vol. 10, pp. 63–84, 04 2010.
- [18] Y. Zhang, K. Lu, Y. Gao, and M. Wang, “Neqr: a novel enhanced quantum representation of digital images,” *Quantum Information Processing*, vol. 12, pp. 2833–2860, 03 2013.
- [19] X.-W. Yao, H. Wang, Z. Liao, M.-C. Chen, J. Pan, J. Li, K. Zhang, X. Lin, Z. Wang, Z. Luo, W. Zheng, J. Li, M. Zhao, X. Peng, and D. Suter, “Quantum image processing and its application to edge detection: Theory and experiment,” *Physical Review X*, vol. 7, 09 2017.
- [20] “D-wave - media coverage,” www.dwavesys.com, 10 2019. [Online]. Available: <https://www.dwavesys.com/media-coverage/volkswagen-optimizes-traffic-flow-quantum-computers>
- [21] “Technology overview practical quantum computing d-wave technology overview,” D Wave, 2020. [Online]. Available: https://www.dwavesys.com/sites/default/files/Dwave_Tech%20Overview2_F.pdf
- [22] “Ionq,” IonQ, 2020. [Online]. Available: <https://ionq.com/technology>
- [23] “Rigetti computing,” Rigetti, 2020. [Online]. Available: <https://www.rigetti.com/what>
- [24] I. , “Docs and resources - backends,” IBM Quantum Experience, 2020. [Online]. Available: <https://quantum-computing.ibm.com/docs/manage/backends/>

- [25] —, “Docs and resources,” IBM Quantum Experience, 2020. [Online]. Available: <https://quantum-computing.ibm.com/docs/>
- [26] —, “Noisemodel qiskit 0.20.0 documentation,” qiskit.org, 2020. [Online]. Available: <https://qiskit.org/documentation/stubs/qiskit.providers.aer.noise.NoiseModel.html>
- [27] A. W. Cross, L. S. Bishop, J. A. Smolin, and J. M. Gambetta, “Open quantum assembly language,” *arXiv:1707.03429 [quant-ph]*, 07 2017. [Online]. Available: <https://arxiv.org/abs/1707.03429>
- [28] I. , “Qiskit,” GitHub, 04 2019. [Online]. Available: <https://github.com/Qiskit>
- [29] “Cooper pair,” Wikipedia, 04 2020. [Online]. Available: https://en.wikipedia.org/wiki/Cooper_pair
- [30] M. , “Achieving practical quantum computing,” Microsoft Quantum, 06 2018. [Online]. Available: <https://cloudblogs.microsoft.com/quantum/2018/06/01/achieving-practical-quantum-computing/>
- [31] Microsoft, “Quantum simulators and q# programs - microsoft quantum,” docs.microsoft.com. [Online]. Available: <https://docs.microsoft.com/en-us/quantum/user-guide/machines/>
- [32] M. , “Language-integrated quantum operations: $\text{Liqui}|>$,” Microsoft Research, 03 2016. [Online]. Available: <https://www.microsoft.com/en-us/research/project/language-integrated-quantum-operations-liqui/>
- [33] —, “microsoft/quantum,” GitHub, 08 2020. [Online]. Available: <https://github.com/microsoft/Quantum/blob/master/README.md>
- [34] Y. LeCun, C. Cortes, and C. Burges, “Mnist handwritten digit database,” *ATT Labs [Online]*. Available: <http://yann.lecun.com/exdb/mnist>, vol. 2, 2010.
- [35] L. García, N. Tanetani, J. P. Arias Zapata, H. Co, and E. P. Tapia, “Shedka/citiesatnight,” GitHub, 07 2020. [Online]. Available: <https://github.com/Shedka/citiesatnight>
- [36] H. Gómez-Moreno, . Maldonado-Bascón, . López-Ferreras, . Acevedo-Rodríguez, and . Martín-Martín, “Edge detection by using the support vector machines.” [Online]. Available: <https://pdfs.semanticscholar.org/b1d1/e8a9d6173458687bdfdc5a654423444f15b0.pdf>
- [37] M. Mastriani, “Quantum boolean image denoising,” *Quantum Information Processing*, vol. 14, pp. 1647–1673, 11 2014. [Online]. Available: <https://arxiv.org/ftp/arxiv/papers/1408/1408.2427.pdf>
- [38] A. Kandala, K. Temme, A. D. Córcoles, A. Mezzacapo, J. M. Chow, and J. M. Gambetta, “Error mitigation extends the computational reach of a noisy quantum processor,” *Nature*, vol. 567, pp. 491–495, 03 2019.

Appendix A

A.1 Code and other files

All the images and the code implementation is available on GitHub -

<https://github.com/paritoshkc/Quantum-Computing>

A.2 Microsoft

A.2.1 Ways to trigger Q# programm

There are mainly 2 ways in which Q# code can be executed - standalone application or host programs. In the case of a Standalone application, it can be executed using a command-line or inside Q# jupyter notebook. The host programs can be based either on python or .NET programming language. [31]

A.3 IBM

A.3.1 IBM installing libraries

The library can be cloned from GitHub [github link for qiskit core] or installed using the pip command below.

```
pip install qiskit[visualization]
```

A.3.2 IBM API Key Generation

To get access to the cloud-based Quantum systems we need to generate an API key by signing in the IBM Quantum Experience website <https://quantum-computing.ibm.com/> . Signing in and other details are provided in Appendix A. Once the API key is acquired we can save and load the account using the following command -

```
from qiskit import IBMQ

IBMQ.saveaccount('API_KEY')
IBMQ.loadaccount()
```

1. Create a new account and sign in to <https://quantum-computing.ibm.com/>. 2. Navigate to My Account by clicking on the profile button located at top right corner. 3. Locate the column 'Qiskit in local environment' and select Copy Token.

A.3.3 Commands for python-resize-image

```
\$ pip install python-resize-image

#resize an input image by defining the width and height of the image
image = resizeimage.resize_cover(image, [width, height])
```