

PROJECT REPORT ON

**QUANTUM MACHINE LEARNING ALGORITHM FOR
OPTIMIZING IMAGE DATA USING ENHANCED QUANTUM
CLASSIFICATION TECHNIQUES**

Submitted by

Utsav Pathak (19030141CSE060)

Siddhartha Das (19030141CSE061)

Saurabh Kumar (18030141CSE068)

Towards the partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Under the Supervision of

Dr. P. Mano Paul

Associate Professor, Department of CSE



ALLIANCE
UNIVERSITY

Department of Computer Science and Engineering
Alliance College of Engineering and Design
Alliance University - Bangalore-562106

June- 2023

Department of Computer Science and Engineering
Alliance College of Engineering and Design
Alliance University
Chikkahagade Cross, Chandapura-Anekal Main Road,
Bangalore-562106



CERTIFICATE

This is to certify that the project work entitled “**QUANTUM MACHINE LEARNING ALGORITHM FOR OPTIMIZING IMAGE DATA USING ENHANCED QUANTUM CLASSIFICATION TECHNIQUES**” is the bonafide work done by **Mr. Utsav Pathak** (19030141CSE060), **Mr. Siddhartha Das** (19030141CSE061), **Mr. Saurabh Kumar** (19030141CSE068), submitted in partial fulfillment of the requirements for the award of the degree **Bachelor of Technology in Computer Science and Engineering** during the year **2019-2023**.

Mano Paul
Dr. P. Mano Paul
SUPERVISOR

Dr. Abraham George
Dr. Abraham George
HOD

External Examiners: 1. Name: Dr. Abhinav G.

Signature: *Dr. Abhinav G.*

2. Name: Dr. Anoop Kumar

Signature: *Dr. Anoop Kumar*



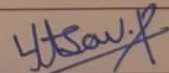

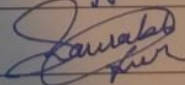
**ALLIANCE
UNIVERSITY**

Private University established in Karnataka State by Act No. 14 of year 2010
Recognized by the University Grants Commission (UGC), New Delhi

Alliance College of Engineering and Design

DECLARATION

This is to declare that the report titled “**QUANTUM MACHINE LEARNING ALGORITHM FOR OPTIMIZING IMAGE DATA USING ENHANCED QUANTUM CLASSIFICATION TECHNIQUES**” has been made for the partial fulfilment of the Course of **Bachelor of Technology in Computer Science and Engineering**, under the Supervision of **Dr. P. Mano Paul**. We confirm that this report truly represents our work undertaken as a part of our project work. This work is not a replication of work done previously by any other person. We also confirm that the contents of the report and the views contained therein have been discussed and deliberated with the faculty guide.

NAME	REG. NO.	SIGNATURE
UTSAV PATHAK	19030141CSE060	
SIDDHARTHA DAS	19030141CSE061	
SAURABH KUMAR	19030141CSE068	

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of the task would be put incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success.

First and foremost, we would like to thank The Almighty God, the creator of this cosmos and the ultimate in scientific factors, a proof of which is this work from quantum.

We are very thankful to our guide and Project Coordinator **Dr. P. Mano Paul**, Department of Computer Science and Engineering for his sustained inspiring guidance and cooperation throughout the process of this project. His wise counsel and valuable suggestions are invaluable.

We would like to thank **Dr. Abraham George**, Head of Department, Computer Science and Engineering and **Dr. Reeba Korah**, Dean, Alliance College of Engineering and Design for their encouragement and cooperation at various levels of Project.

We avail this opportunity to express my deep sense of gratitude and hearty thanks to the Management of Alliance University, for providing world class infrastructure, congenial atmosphere, and encouragement.

We express my deep sense of gratitude and thanks to the teaching and non-teaching staff at our department who stood with me during the project and helped me to make it a successful venture.

We place highest regards to my parents, my friends and well-wishers who helped a lot in making the report of this project and have been the pivot of motivation throughout our journey here at Alliance University.

UTSAV PATHAK

SIDDHARTHA DAS

SAURABH KUMAR

TABLE OF CONTENTS

Chapter No.	Chapter Name	Page No.
	Abstract	vii
	List of Figures	viii
	List of Tables	ix
1	Introduction	1
	1.1 Introduction to Quantum Theory	1
	1.2 Introduction to Quantum Computing	2
	1.3 Introduction to IBM Qiskit	3
	1.4 Introduction to Google Cirq	5
2	Literature Survey	8
	2.1 A Review of Methods for The Image Automatic Annotation	8
	2.2 Quantum Image Processing: Opportunities and Challenges	9
	2.3 Quantum Representation of Indexed Images and its Applications	10
	2.4. Improved FRQI on superconducting processors and its restrictions in the NISQ era	11
3	System Design	13
	3.1 System Architecture	13
	3.1.1 Classical CNN Architecture	14
	3.1.2 Quantum Circuit	18
	3.2.3 Combined Model	29
	3.2 Use Case Diagram	33
4	Proposed Design	35
	4.1 Estimator Quantum Circuit	35
	4.2 Initiator Quantum Circuit	37
	4.3 Deutsch Jozsa Algorithm	39
	4.4. Quantum Optimized CNN	40

5	Results	44
	5.1 Grayscale MNIST on IBM QISKIT	44
	5.2 Colour MNIST on Google CIRQ	45
	5.3 Circuit Simulation on Real Quantum Computer	46
	5.4 Difference caused by Quantum Error and Circuit Failure	46
6	Conclusion and Scope of Future Work	49
7	Appendix	50
	7.1 Source Code	50
	7.2 Formulas Used	52
8	References	53

ABSTRACT

With the massive collection of data and the usage of various predictive modeling with the techniques for images, the need for machine learning came into existence. Dealing with extremely large scales of data, and the need to solve complex problems which trace back to the origin, the need for implementation of quantum techniques became essential. Quantum theory accesses the power of computing at atomic levels and as a current trend, combines with machine learning which helps in speeding up the process and getting very high levels of accuracy. Quantum image processing helps in analyzing the images better, reducing noise and distortion factors, this gives a multi-point view of the image and generates better classification results. Image processing usually deals with all the features on an image, and utilizes image matrices to extract and compare information, this usually results in a loss/lossless, when images are compressed, transformed, and when noise reduction techniques are applied to the images. A typical image lattice gives far fewer details about the image, while analyzing it with a quantum system, helps achieve high accuracy for classification and regression tasks. The Representation of Quantum Images done in our project to classify images using an Enhanced Quantum Representation help in handling details better and obtaining a multidimensional view of the data. The goal is to analyze digital images and make a less intensive model that requires minimal training and gives out extremely good results for classification, segmentation, and other highly intensive tasks for large images like annotation.

List of Figures

Figure No.	Figure Name	Page No.
1	CNN Architecture	16
2	Quantum Logic Gates	21
3	Bloch Sphere	25
4	System Architecture	30
5	Use Case Diagram	33
6	Estimator Quantum Circuit	36
7	Initiator Quantum Circuit	38
8	Quantum CNN	41
9	Results on Grayscale MNIST	44
10	Results on Colour MNIST	45
11	Measurements in Tensor Space (Formula)	46
12	Annotation Recall and Validation (Formula)	46
13	Best Position of Particle (Formula)	46

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION TO QUANTUM THEORY

Quantum theory describes behaviors at the smallest scales. Quantum theory departs from classical physics by introducing the concept of quantization. According to this principle, certain physical quantities, such as energy, angular momentum, and position, can only take on discrete values called "quantum states" or "eigenstates." This contrasts with classical physics, where these quantities are assumed to be continuous. The mathematical description of quantum theory is based on wave functions.

A wave function is a mathematical function that associates each possible state of a particle with a complex number. It contains all the information about the particle's properties, including its position, momentum, and energy. Evolving systems are determined by using Schrödinger equation, which describes how the wave function changes over time. According to superposition, particles can exist in multiple states simultaneously. For example, an electron can be in a superposition of being in two different locations at the same time. This is in stark contrast to classical physics, where an object can only be in one state at a given time. Superposition forms the basis for many quantum phenomena and is a key resource in quantum computing. Measurement plays a crucial role in quantum theory. During quantum system measurements, the wave function "collapses" into one of its possible states, giving a specific result. The outcome of a measurement is probabilistic, and the probabilities of different outcomes are determined by the squared magnitudes of the wave function amplitudes. This probabilistic nature is deeply ingrained in quantum theory and is a departure from the deterministic nature of classical physics. Entanglement is another remarkable feature of quantum theory. When two or more particles become entangled, their quantum states become correlated in such a way that the behavior of one particle is intimately connected to the behavior of the others, regardless of the distance between them. This phenomenon, famously characterized by Einstein as "spooky action at a distance," has

been experimentally confirmed and is the basis for applications such as quantum teleportation and quantum cryptography.

Heisenberg's uncertainty principle, which captures the idea of uncertainty, is another notion introduced by quantum theory. According to this theory, there is a fundamental limit to the accuracy with which some complementary pairs of attributes, like location and momentum, may be measured at the same time. The other feature can be known less accurately the more exactly the first is understood. Because quantum things have wave-particle duality, they have this innate uncertainty. Although they go against our traditional intuitions, ideas like superposition, entanglement, and uncertainty have been supported by countless tests. In addition to revolutionizing physics, quantum theory has the potential to do the same for other industries in the future, including computers, communication, and sensing.

1.2 INTRODUCTION TO QUANTUM COMPUTING

A rapidly developing discipline is quantum computing, which uses the ideas behind quantum mechanics to do out intricate computations. Quantum computers use quantum bits, or qubits, which may exist in several states concurrently, as opposed to ordinary computers, which store and process information using bits that either represent a 0 or a 1. Due to this basic distinction, quantum computers can answer some problems far more quickly than conventional computers. Here is a brief description of how quantum computing functions. The idea of superposition is at the foundation of quantum computing. Particles in quantum physics can exist in several states concurrently up until they are measured. Like this, superposition allows qubits to simultaneously represent a mixture of 0 and 1. This property allows quantum computers to process a vast number of possibilities simultaneously.

Entanglement is yet another important idea. No matter how far apart qubits are from one another, when they become entangled, their states are inextricably connected. This enables quantum computers to create intricate relationships between qubits and perform computations on the entire system, rather than on individual qubits.

Quantum algorithms are created to take advantage of superposition and entanglement to more effectively address particular computing issues. Shor's algorithm is one such method that factors huge numbers exponentially more quickly than conventional techniques. This poses a serious threat to existing encryption techniques that rely on how difficult it is to factor huge numbers. It is difficult to put quantum computing into practice. Qubits may be created using a variety of physical processes, including topological qubits, trapped ions, superconducting circuits, and even photons. To preserve the qubits' fragile quantum states and reduce environmental error, these systems need to be tightly regulated. Quantum gates, which are comparable to logic gates in conventional computers, are used by quantum computers to carry out calculations. Quantum gates manipulate the quantum states of the qubits, enabling operations like superposition, entanglement, and transformations of the quantum information. These gates, along with the algorithms, guide the quantum computer to perform calculations on the qubits. However, quantum systems are highly susceptible to noise and decoherence, which are major obstacles in quantum computing. Decoherence occurs when the fragile quantum states interact with the environment, leading to information loss and errors. To address these problems and guarantee the dependability of quantum calculations, several error correction strategies and fault-tolerant architectures are being created. Large-scale, useful quantum computers have not yet been completely realized since quantum computing is still in its infancy. Nevertheless, tremendous progress has been achieved, and quantum computers have already shown that they are capable of more effectively tackling a variety of challenges. International teams of scientists and engineers are attempting to scale up the possibilities of quantum computing while overcoming the technological obstacles.

1.3 INTRODUCTION TO IBM QISKIT

Quantum Computing platforms, help in simulation of quantum states, circuits and facilitate running jobs on an actual quantum computer. IBM Qiskit is an open-source framework for developing quantum computing programs and applications. It provides a comprehensive set of tools and libraries that enable researchers, developers, and enthusiasts to explore and experiment with quantum computing concepts. Qiskit allows users to design, simulate, and execute quantum circuits on real quantum hardware provided by IBM's cloud-based quantum computing platform, IBM

Quantum. Qiskit is designed to be accessible to both beginners and experienced quantum computing practitioners.

It consists of several components that work together to facilitate various aspects of quantum computing development:

1. Qiskit Terra: Terra is the foundation of Qiskit and provides the basic building blocks for creating and manipulating quantum circuits. It includes a rich set of quantum gates, circuit visualization tools, and algorithms for compiling quantum circuits into the native instructions of target quantum devices.
2. Qiskit Aer: Aer is a high-performance simulator included in Qiskit. It allows users to simulate quantum circuits on classical computers, providing insights into the behavior and performance of quantum algorithms without the need for physical quantum hardware. Aer supports different simulation modes, including statevector simulation, unitary simulation, and more.
3. Qiskit Ignis: Ignis focuses on quantum error correction and mitigation techniques. It provides tools for characterizing and calibrating quantum devices, as well as implementing error correction codes and error mitigation strategies. Ignis plays a crucial role in improving the reliability and accuracy.
4. Qiskit Aqua: Aqua is a component of Qiskit that focuses on quantum computing applications in fields such as optimization, chemistry, finance, and machine learning. It provides a set of domain-specific libraries and algorithms that leverage the power of quantum computing to solve complex problems in these areas.
5. Qiskit Optimization: Optimization is a specialized module within Qiskit Aqua that offers tools and algorithms for solving optimization problems using quantum computing techniques. It enables users to map real-world optimization problems onto quantum circuits and leverage the inherent parallelism of quantum computing to find optimal solutions more efficiently.
6. Qiskit Machine Learning: Machine Learning is another specialized module within Qiskit Aqua that provides tools and algorithms for developing quantum machine learning models. It allows users to explore the potential of quantum computing in enhancing various aspects of machine learning, such as data classification, feature mapping, and dimensionality reduction.

The working of Qiskit involves several steps:

1. **Circuit Design:** Users define quantum circuits by selecting gates and arranging them in a sequence. Qiskit provides a comprehensive set of gates, including the standard gates like Hadamard, CNOT, and Pauli gates. Circuit visualization tools assist in visualizing and verifying the correctness of the designed circuits.
2. **Circuit Compilation:** Qiskit offers various optimization techniques to compile high-level quantum circuits into low-level instructions suitable for specific quantum hardware devices. This step involves decomposing the circuit into the device's native gate set and optimizing the circuit for improved performance.
3. **Simulation:** Qiskit Aer allows users to simulate the behavior of quantum circuits on classical computers. Users can analyze the state vectors, perform unitary simulations, and estimate the probabilities of measurement outcomes. This simulation step helps in understanding the behavior and performance of quantum algorithms before running them on real quantum hardware.
4. **Execution on Quantum Hardware:** Qiskit integrates with IBM Quantum, a cloud-based quantum computing platform that provides access to real quantum devices. Users can submit their quantum circuits for execution on these devices and observe the results. Due to the limited availability of quantum hardware, job scheduling and queue management are essential aspects of this step.

Overall, Qiskit simplifies the process of developing quantum applications by providing a comprehensive framework with powerful tools and libraries. It facilitates circuit design, compilation, simulation, and execution on real quantum hardware, enabling researchers and developers to explore the potential of quantum computing and develop innovative quantum algorithms and applications.

1.4 INTRODUCTION TO GOOGLE CIRQ

Google Cirq is an open-source framework developed by Google for quantum computing. It focuses on facilitating the construction, simulation, and execution of quantum circuits, targeting both researchers and developers in the field. Cirq provides a high-level interface and a set of tools that enable users to work with quantum circuits and algorithms effectively.

Unique Features of Google Cirq:

1. **Emphasis on Low-level Operations:** Cirq is designed to provide low-level access to quantum operations, allowing users to have fine-grained control over the construction and manipulation of quantum circuits. This feature appeals to researchers and developers who want to explore the intricacies of quantum algorithms and experiment with advanced techniques.
2. **Support for Noisy Intermediate-Scale Quantum (NISQ) Devices:** Cirq acknowledges the current state of quantum hardware, known as NISQ devices, which are prone to noise and errors. It provides functionalities to incorporate noise models and simulate realistic quantum computations, helping users understand the effects of noise and develop strategies to mitigate errors in quantum circuits.
3. **Fidelity-Aware Optimizations:** Cirq offers optimization techniques that consider the noise and error rates of quantum devices. It includes tools for circuit optimization, gate decomposition, and gate synthesis, aiming to improve the fidelity and performance of quantum computations on real hardware.
4. **Access to Quantum Device Details:** Cirq provides access to device-specific information such as gate sets, connectivity graphs, and error models. This allows users to design quantum circuits tailored to the specific characteristics of the target quantum hardware, optimizing circuit execution, and enhancing the chances of successful runs on physical devices.
5. **Flexibility for Custom Gate Definitions:** Cirq allows users to define their own custom quantum gates and operations, enabling the exploration of novel gate sets and algorithms. This flexibility is particularly beneficial for researchers developing new quantum algorithms or studying specialized quantum systems.

Working of Google Cirq:

1. **Circuit Construction:** Users begin by constructing a quantum circuit using Cirq's programming interface, which offers a collection of quantum gates and operations. They can define the qubits, apply gates, and specify interactions between qubits to build the desired quantum circuit.
2. **Circuit Simulation:** Cirq provides a simulator for running quantum circuit simulations on classical computers. Users can simulate the behavior of quantum circuits and observe the state vector or obtain measurement results. Simulation allows users to test and debug their circuits, understand their behavior, and verify the correctness of their algorithms before running them on real quantum hardware.

3. **Execution on Quantum Devices:** Cirq supports running quantum circuits on various backends, including Google's Quantum Computing Service, which provides access to Google's quantum processors. Users can submit their circuits for execution, monitor the progress, and retrieve the results once the computation is completed. Cirq handles the necessary conversion and translation steps to interface with the targeted quantum devices.
4. **Noise Modeling and Error Mitigation:** To account for the inherent noise and errors in quantum devices, Cirq provides functionalities for incorporating noise models into simulations. Users can simulate the behavior of quantum circuits with realistic noise characteristics, helping them identify potential issues and develop error mitigation strategies.
5. **Optimization and Compilation:** Cirq offers optimization techniques to improve the performance and efficiency of quantum circuits. This includes gate decomposition, circuit optimization, and gate synthesis methods that aim to reduce the circuit depth, minimize the number of gates, and optimize gate sequences based on specific device constraints and characteristics.
6. **Visualization and Analysis:** Cirq provides visualization tools for circuit representation, allowing users to visualize and analyze quantum circuits. Users can plot circuit diagrams, visualize quantum states, and analyze measurement outcomes, aiding in the understanding and debugging of quantum algorithms.

Google Cirq's unique features and working make it a valuable tool for researchers and developers working in the field of quantum computing. Its emphasis on low-level operations, support for NISQ devices, fidelity-aware optimizations, access to device details, and flexibility for custom gate definitions contribute to the exploration and advancement of quantum algorithms and applications.

CHAPTER 2

LITERATURE SURVEY

2.1 A REVIEW OF METHODS FOR THE IMAGE AUTOMATIC ANNOTATION

The objective of the paper is to present a systematic review of image annotation techniques that aim to automatically assign relevant textual tags to images. Image annotation plays a crucial role in various applications, such as image retrieval, organization, and content-based image retrieval. The authors begin by introducing the importance of image annotation and the challenges associated with manual annotation. They emphasize the need for automatic methods that can efficiently and accurately annotate images based on their visual content.

The paper then presents an overview of different image annotation techniques, categorizing them into three main categories: traditional methods, machine learning-based methods, and deep learning-based methods. Traditional methods discussed in the paper include techniques such as visual features extraction, clustering, and classification. The authors provide a concise explanation of each method, along with their advantages and limitations.

The focus then shifts to machine learning-based methods, where the authors explore approaches such as support vector machines (SVM), k-nearest neighbors (KNN), and decision trees. They discuss the utilization of these algorithms for image annotation and highlight their effectiveness in certain scenarios. The final category discussed in the paper is deep learning-based methods, which have gained significant attention and achieved remarkable results in image annotation tasks. The authors delve into deep learning architectures, including convolutional neural networks (CNNs), recurrent neural networks (RNNs), and their variants. They discuss how these architectures can be utilized for image annotation tasks, including multi-label image annotation and fine-grained image annotation.

Throughout the review, the authors provide insights into the strengths and limitations of each method, discussing their computational complexity, scalability, and performance. They also highlight recent advancements and trends in the field of image annotation. In conclusion, the paper provides a comprehensive review of methods for image automatic annotation. It serves as a valuable resource for researchers and practitioners interested in understanding the different approaches used for automatically assigning textual tags to images. The paper presents an organized overview of traditional, machine learning-based, and deep learning-based methods, facilitating a better understanding of the state-of-the-art techniques in the field of image annotation.

2.2 QUANTUM IMAGE PROCESSING: OPPORTUNITIES AND CHALLENGES

This paper begins by introducing the emerging field of quantum image processing, which combines the principles of quantum mechanics with image processing techniques. It explains that quantum computing has the potential to revolutionize image processing tasks by leveraging the inherent quantum properties of superposition and entanglement. The authors discuss the advantages offered by quantum image processing over classical image processing methods.

Quantum image processing algorithms have the potential to perform certain tasks more efficiently, such as image compression, edge detection, and pattern recognition. They can also offer enhanced security through the implementation of quantum encryption and quantum watermarking techniques. The paper then delves into various quantum image processing algorithms and techniques that have been proposed in the literature.

These include quantum Fourier transform-based methods, quantum superposition-based algorithms, and quantum wavelet transform approaches. The authors provide a comprehensive overview of these techniques, discussing their underlying principles and potential applications. In addition to opportunities, the paper also addresses the challenges faced by quantum image processing. These challenges include the difficulty of quantum image acquisition due to the quantum no-cloning

theorem, limited qubit resources in current quantum computers, and the need for error correction techniques to mitigate the effects of noise and decoherence.

The authors conclude by emphasizing the immense potential of quantum image processing and its ability to outperform classical methods in certain applications. They highlight the need for further research and development in this field to overcome the existing challenges and fully exploit the advantages offered by quantum computing. In summary, the paper provides a comprehensive overview of quantum image processing, discussing its opportunities, challenges, and various algorithms. It serves as a valuable resource for researchers and practitioners interested in exploring the potential of quantum computing in the field of image processing.

2.3 QUANTUM REPRESENTATION OF INDEXED IMAGES AND ITS APPLICATIONS

This paper begins by introducing the concept of indexed images, which are commonly used in computer graphics and image processing. In an indexed image, each pixel value is represented by an index that corresponds to a color lookup table. The authors propose a quantum representation scheme for indexed images, where the pixel values are encoded as superposition states of qubits. The authors explain the advantages of quantum representation over classical representation for indexed images.

Quantum representation allows for parallel processing of multiple pixels simultaneously, enabling faster image manipulation and analysis. It also offers the potential for efficient storage and transmission of indexed images using quantum compression techniques. The paper presents a detailed description of the quantum representation scheme for indexed images, including the encoding of pixel values and the corresponding quantum operations.

The authors discuss the quantum gates and algorithms required for performing common image processing tasks on the quantum representation of indexed images, such as image rotation, scaling, and filtering. Furthermore, the paper explores the potential applications of quantum representation in the field of image processing. It discusses

how quantum representation can be leveraged for image encryption and decryption, offering enhanced security compared to classical encryption methods.

The authors also propose a quantum-based image watermarking technique that can embed and extract watermarks with high robustness and imperceptibility. The paper concludes by highlighting the significance of quantum representation in the context of indexed images and its potential impact on image processing. It suggests that further research and development are necessary to explore the full capabilities of quantum representation and to address the challenges associated with implementing quantum algorithms for image manipulation. In summary, the paper presents a novel quantum representation scheme for indexed images and discusses its applications in image processing, encryption, and watermarking. It provides valuable insights into the potential advantages of quantum computing in the realm of image representation and manipulation.

2.4 IMPROVED FRQI ON SUPERCONDUCTING PROCESSORS AND ITS RESTRICTIONS IN THE NISQ ERA

This paper begins by introducing the FRQI algorithm, which is a quantum image representation and processing technique. The authors propose an improved version of FRQI specifically tailored for implementation on superconducting processors, which are one of the prominent platforms for NISQ quantum computing. The authors outline the enhancements made to the original FRQI algorithm, including the introduction of optimized quantum gates and improved error mitigation techniques. These improvements aim to enhance the fidelity and accuracy of quantum image processing tasks performed using FRQI on superconducting processors.

The paper then delves into the experimental results obtained from implementing the improved FRQI algorithm on a superconducting quantum processor. The authors discuss the performance metrics, such as fidelity, computational time, and resource utilization, and compare them with the results obtained from the original FRQI algorithm. The experimental findings demonstrate the superiority of the improved FRQI in terms of both image quality and processing efficiency. However, the paper also

addresses the limitations and restrictions of implementing FRQI on NISQ-era superconducting processors.

The authors discuss the challenges posed by the limited qubit coherence time, gate errors, and noise in these quantum computing platforms. They highlight the impact of these limitations on the scalability and practical application of FRQI for large-scale quantum image processing tasks. The authors conclude by emphasizing the potential of the improved FRQI algorithm for quantum image processing on superconducting processors.

They acknowledge the current restrictions imposed by the NISQ era and call for further advancements in hardware technology and error mitigation techniques to fully harness the capabilities of FRQI for practical applications. In summary, the paper presents an improved version of the FRQI algorithm tailored for superconducting processors and discusses its limitations in the NISQ era. It provides valuable insights into the advancements in quantum image processing on superconducting platforms and highlights the challenges that need to be overcome for its widespread practical implementation.

CHAPTER 3

SYSTEM DESIGN

3.1 SYSTEM ARCHITECTURE

Classical System Architecture: In classical computing, the architecture is based on classical physics and the principles of classical information processing. It consists of a classical bit as the fundamental unit of information and follows the principles of Boolean logic. Classical systems utilize a sequential execution model, where instructions are executed one after another, and the state of the system is determined by the values of the classical bits. Classical architectures are deterministic, meaning that given the same input, they produce the same output every time. They are typically built using digital circuits, such as logic gates and processors.

On the other hand, quantum computing functions in accordance with the ideas of quantum mechanics. The quantum bit, also known as a qubit, is the basic building block of information in quantum computing. Qubits, in contrast to conventional bits, may exist in a superposition of states, concurrently indicating 0 and 1. Quantum systems modify the qubits and carry out calculations via quantum gates. Quantum processors, quantum registers, and quantum gates are all components of a quantum system's architecture.

Similarities: Both classical and quantum architectures involve the manipulation of information to perform computations. They can both process and store data, and their goal is to solve computational problems. Both architectures rely on principles of mathematics and logic.

Differences: The key difference lies in the nature of the fundamental units of information and the underlying principles. Classical systems operate using classical bits and follow classical physics, while quantum systems use qubits and operate based on quantum mechanics. Classical systems are deterministic, whereas quantum systems can exhibit probabilistic behavior. Quantum systems can perform certain computations exponentially faster than classical systems for specific problems due to quantum

phenomena such as superposition and entanglement. Quantum architectures require specialized hardware and are more susceptible to noise and errors compared to classical architectures, which can make quantum computation more challenging.

3.1.1 CLASSICAL CNN ARCHITECTURE

Convolutional Neural Networks (CNNs) are a subset of deep learning models that are specifically made for processing and analyzing visual input, such as pictures and videos. Tasks like picture classification, object identification, and image segmentation have been completely transformed by them. To extract pertinent characteristics and provide precise predictions, CNNs take advantage of the special qualities of convolutional layers, pooling layers, and fully connected layers. Here is a detailed explanation of the CNN architecture:

1. Input Layer:

It receives raw input data, typically an image represented as a grid of pixels. Each pixel contains color information (e.g., RGB values). The size of the input layer corresponds to the dimensions of the input image.

2. Convolutional Layer:

This process extracts local features such as edges, textures, and patterns. The convolutional layer learns to recognize different low-level features through the iterative adjustment of filter weights during training.

3. Activation Layer:

To add non-linearities, an activation function is applied elementwise to the convolutional layer's output. Rectified Linear Units (ReLU), which converts negative values to zero, and sigmoid or tanh functions are examples of common activation functions.

4. Pooling Layer:

Downsampling by pooling layers preserves the most significant data while sampling the input's spatial dimensions. Max pooling is a frequently used approach that chooses the largest value inside each non-overlapping zone after dividing the input into those parts.

By lowering the spatial resolution, this improves the computational efficiency and makes following layers less susceptible to minute spatial fluctuations. Making characteristics that are translation-invariant is another benefit of pooling.

5. Additional Convolutional and Pooling Layers:

CNNs typically have multiple stacked convolutional and pooling layers. Deeper layers learn more complex and abstract features by combining low-level features learned in previous layers. As the network becomes deeper, the number of filters usually increases to capture more sophisticated features.

6. Fully Connected Layer:

Convolutional and pooling layers are introduced before fully linked layers. These layers resemble conventional neural networks in that each neuron is linked to every other neuron in the layer below. Fully connected layers carry out the final classification using the retrieved features after learning global patterns.

7. Dropout Layer:

To mitigate overfitting, dropout layers are introduced. During training, random neurons are temporarily dropped out, meaning their outputs are set to zero. This encourages the network to rely on the remaining neurons and prevents overreliance on specific features, leading to better generalization.

8. Output Layer:

The final forecasts or classifications are produced by the output layer. When doing a classification job, there are exactly as many neurons in this layer as there are classes. The last layer activations are converted into probabilities using the activation functions SoftMax for multi-class classification and sigmoid for binary classification.

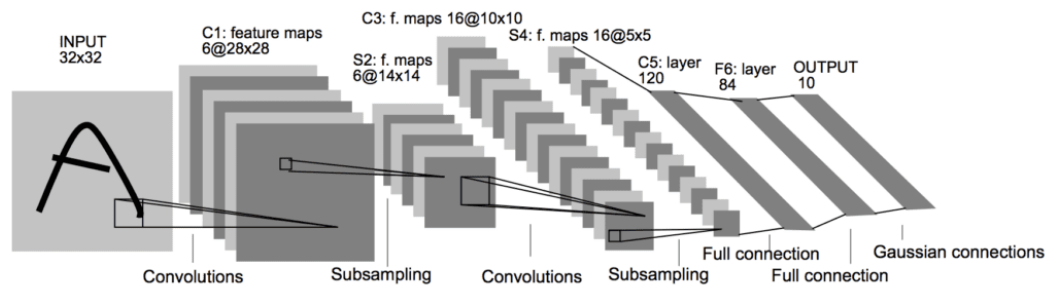


Figure 1: CNN Architecture (Ref. Machine Learning Mastery)

The CNN architecture is trained using backpropagation, which involves iteratively adjusting the weights and biases in the network to minimize a defined loss function. This process allows the network to learn the optimal set of features for accurate predictions.

This feature extraction capability, along with the translation-invariance provided by pooling layers, makes CNNs powerful tools for visual recognition tasks.

Image processing in Convolutional Neural Networks (CNNs) involves a series of operations to analyze and manipulate images.

Image processing in CNNs offers several advantages:

1. Automatic Feature Extraction: CNNs

can automatically learn relevant features from images, eliminating the need for manual feature engineering. The convolutional layers capture spatial hierarchies of features, enabling the network to recognize complex patterns and structures.

2. Translation Invariance: CNNs are invariant to small translations in the input images due to the local receptive fields and pooling operations. This property allows CNNs to recognize objects regardless of their position in the image.

3. Parameter Sharing: CNNs use parameter sharing, where the same set of weights is applied to multiple locations in the input image. This reduces the number of learnable

parameters, making CNNs more efficient and effective in handling large-scale image datasets.

4. Hierarchical Representation: CNNs learn features in a hierarchical manner, with lower layers capturing low-level features (e.g., edges, corners) and higher layers capturing high-level features (e.g., shapes, textures). This hierarchical representation enables CNNs to recognize complex and abstract concepts.

Despite their effectiveness, CNNs also have some limitations:

1. High Computational Requirements: CNNs can be computationally intensive, especially with deeper architectures and larger input images. Training and inference may require significant computational resources and time.

2. Need for Large Datasets: CNNs often require large datasets for training to avoid overfitting and achieve good generalization. Acquiring and labeling large-scale datasets can be time-consuming and costly.

3. Lack of Spatial Context: CNNs process images locally using fixed-size receptive fields, which may limit their ability to capture global spatial context and long-range dependencies in some scenarios.

In conclusion, image processing in CNNs involves a series of operations, including convolution, activation, pooling, and fully connected layers, to extract and learn features from input images. CNNs offer automatic feature extraction, translation invariance, and hierarchical representation. However, they also require computational resources, large datasets, and may lack global spatial context. Understanding the process of image processing in CNNs is crucial for leveraging their capabilities in various computer vision tasks.

It's worth noting that various modifications and advancements have been made to the basic CNN architecture over the years, such as the inclusion of skip connections (e.g., in the ResNet architecture), attention mechanisms (e.g., in the Transformer architecture), and spatial pyramid pooling (e.g., in the PSPNet architecture). These

advancements aim to further improve the performance and capabilities of CNNs in handling complex visual tasks.

3.1.2 QUANTUM CIRCUIT

Quantum gates are fundamental building blocks in quantum computing that manipulate the quantum state of qubits. They are analogous to the logical gates in classical computing, but their operations are based on the principles of quantum mechanics. Quantum gates enable the creation, manipulation, and measurement of quantum superposition and entanglement, which are the key resources for quantum computation. In this explanation, we'll explore how quantum gates work and discuss some commonly used types of quantum gates.

1. Quantum State and Qubits:

Qubits, sometimes referred to as quantum bits, are used to store information in quantum computing. Qubits can concurrently represent both 0 and 1, as contrast to traditional bits, which can only be in one of two states—a 0 or a 1—at any given time. An abstract mathematical concept known as a quantum state vector may be used to represent the state of a single qubit. For example, a qubit in a superposition state can be represented as $\alpha|0\rangle + \beta|1\rangle$, where α and β are complex probability amplitudes.

2. Quantum Gates:

Quantum gates are mathematical operations that act on the quantum state of qubits. They are represented by matrices that operate on the quantum state vector, transforming it according to the specific gate operation. The gate operation can affect the probabilities or amplitudes associated with different states of the qubits.

3. Types of Quantum Gates:

There are various types of quantum gates, each serving a different purpose in quantum computation. Here are some commonly used quantum gates:

a. Pauli Gates:

Pauli-X Gate (NOT Gate): It acts as a bit-flip gate, flipping the state of a qubit from $|0\rangle$ to $|1\rangle$ and vice versa.

Pauli-Y Gate: It combines a bit-flip and a phase-flip, transforming $|0\rangle$ to $i|1\rangle$ and $|1\rangle$ to $-i|0\rangle$.

Pauli-Z Gate: It acts as a phase-flip gate, multiplying the phase of $|1\rangle$ by -1.

b. Hadamard Gate:

The Hadamard gate creates superposition by transforming the $|0\rangle$ state to $(|0\rangle + |1\rangle)/\sqrt{2}$ and the $|1\rangle$ state to $(|0\rangle - |1\rangle)/\sqrt{2}$. It is used to prepare qubits in superposition states.

c. Phase Gates:

Phase gates introduce phase shifts to the quantum state.

Phase Shift Gate (R_ϕ): It applies a phase shift of ϕ to the $|1\rangle$ state, leaving the $|0\rangle$ state unchanged.

d. Controlled Gates:

Controlled gates act on a target qubit, conditioned on the state of a control qubit.

Controlled-NOT (CNOT) Gate: It flips the target qubit if the control qubit is in the $|1\rangle$ state. It is a widely used gate for entangling qubits.

Controlled-Phase (CPHASE) Gate: It introduces a phase shift to the target qubit if the control qubit is in the $|1\rangle$ state.

e. Swap Gate:

The Swap gate exchanges the state of two qubits.

f. Quantum Fourier Transform Gates:

These gates are used in quantum Fourier transforms, which play a crucial role in quantum algorithms such as Shor's algorithm for factoring large numbers.

Quantum Phase Estimation Gates: These gates estimate the phase of a quantum state.

g. Measurement Gate:

The measurement gate collapses the quantum state to a classical state, yielding an outcome based on the probabilities associated with the amplitudes.

h. Others:

There are many other types of gates, including Toffoli gates, controlled-phase gates with arbitrary phases, and more. These gates serve specific purposes in quantum algorithms and quantum error correction.

4. Gate Composition:

Multiple quantum gates can be combined to create more complex operations. The resulting operation is determined by the matrix multiplication of the gate matrices. For example, applying a Hadamard gate (H) followed by a CNOT gate (CX) creates a controlled-Hadamard gate (CH), which entangles two qubits.

5. Unitarity and Reversibility:

Quantum gates are required to be unitary, meaning that the gate matrix must be square and have an inverse that can be applied to revert the operation. This property ensures that the evolution of the quantum state is reversible, which is essential for quantum computation.

6. Quantum Circuit Representation:

Quantum gates can be visualized using quantum circuit diagrams, which represent the qubits as lines and the gates as boxes acting on those lines. The ordering of the gates determines the sequence of operations.













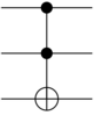
Operator	Gate(s)	Matrix
Pauli-X (X)	 	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$
Pauli-Y (Y)		$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$
Pauli-Z (Z)		$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$
Hadamard (H)		$\frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$
Phase (S, P)		$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$
$\pi/8$ (T)		$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$
Controlled Not (CNOT, CX)		$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$
Controlled Z (CZ)	 	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$
SWAP	 	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$
Toffoli (CCNOT, CCX, TOFF)		$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$

Figure 2: Quantum Logic Gates (Ref. Wikipedia Org)

Therefore, Quantum gates are mathematical operations that manipulate the quantum state of qubits in quantum computing. They enable the creation of superposition and entanglement, essential resources for quantum computation. Quantum gates can be combined to perform complex operations, and various types of gates exist for specific purposes. Understanding and effectively utilizing these gates are fundamental in designing and implementing quantum algorithms and quantum information processing tasks.

Constructing and making a quantum circuit involves designing a sequence of quantum gates to perform specific operations on qubits. Quantum circuits are a graphical representation of the sequence of gates, and they play a crucial role in

quantum algorithm design and implementation. In this explanation, we'll explore the construction and making of a quantum circuit, along with its advantages and drawbacks.

1. Designing a Quantum Circuit:

The construction of a quantum circuit begins with the design of the desired quantum algorithm or task. This involves determining the number of qubits required, the input states, the desired operations, and the final measurement. The design process is highly dependent on the specific problem or algorithm being addressed.

2. Mapping Logical Qubits to Physical Qubits:

In practical implementations, physical qubits are susceptible to noise and errors. Therefore, it is necessary to map logical qubits (used in the algorithm design) to physical qubits (available in the quantum hardware). This mapping is often influenced by the connectivity constraints of the hardware, as quantum gates can typically only operate on nearby qubits. The choice of mapping can impact the efficiency and success of the quantum circuit implementation.

3. Selecting Quantum Gates:

Once the mapping is determined, the next step is to select appropriate quantum gates to implement the desired operations. The selection of gates depends on the specific quantum algorithm or task. As discussed earlier, there are various types of quantum gates available, including Pauli gates, Hadamard gates, controlled gates, and others. The gates need to be chosen carefully to perform the desired computations or transformations on the qubits.

4. Quantum Circuit Representation:

The selected gates are then arranged in a specific order to create a quantum circuit. Quantum circuits are represented graphically using quantum circuit diagrams. In these diagrams, qubits are represented as lines, and gates are depicted as boxes acting on those lines. The ordering of the gates determines the sequence of operations. The circuit representation allows for a clear visualization of the quantum operations and their flow.

5. Circuit Execution:

After the construction of the quantum circuit, it needs to be executed on a quantum computer or simulator. In physical implementations, the gates are implemented using pulses of electromagnetic radiation, such as microwave or laser pulses, to manipulate the qubits. The qubits evolve according to the quantum gates' operations and their initial state. Measurement operations can also be included in the circuit to extract classical information from the quantum state.

Advantages of Quantum Circuits:

1. **Universal Computation:** Quantum circuits, when combined with the appropriate set of gates, can perform any computation that a classical computer can do, and in some cases, they can outperform classical computers for specific tasks.
2. **Speedup Potential:** Quantum circuits have the potential to solve certain computational problems exponentially faster than classical computers. Algorithms like Shor's algorithm for factoring large numbers and Grover's algorithm for searching an unsorted database demonstrate this speedup potential.
3. **Parallelism and Superposition:** Quantum circuits can operate on multiple qubits in parallel, thanks to the property of superposition. This enables the exploration of different computation paths simultaneously, leading to potential speedups and enhanced computational efficiency.
4. **Quantum Entanglement:** Quantum circuits can create and utilize entangled states. Entanglement allows for correlations between qubits, even when physically separated, enabling quantum algorithms like quantum teleportation and quantum cryptography.

Drawbacks of Quantum Circuits:

1. **Fragility and Sensitivity to Noise:** Quantum circuits are susceptible to noise and errors due to environmental factors and imperfections in physical qubits. This sensitivity can degrade the quality of computation and hinder the reliability of quantum circuits.
2. **Limited Qubit Availability:** Currently, practical quantum computers have a limited number of qubits available, making it challenging to perform complex computations and implement large-scale quantum algorithms. Scaling up the

number of qubits while maintaining their coherence and connectivity is a significant technological hurdle.

3. Error Correction Challenges: Quantum circuits require error correction techniques to mitigate the effects of noise and errors. Implementing error correction codes and protocols significantly increases the complexity and resource requirements of quantum circuits.

4. Measurement Limitations: Quantum circuits require measurements to extract classical information. However, measurements collapse the quantum state, destroying the superposition and entanglement properties. Careful consideration is necessary to balance the need for measurements with preserving quantum advantages.

5. Circuit Depth and Time Complexity: Quantum circuits can have a high depth, meaning that they require many gates and operations to achieve the desired computation. This can increase the time complexity and resource requirements for executing the circuit.

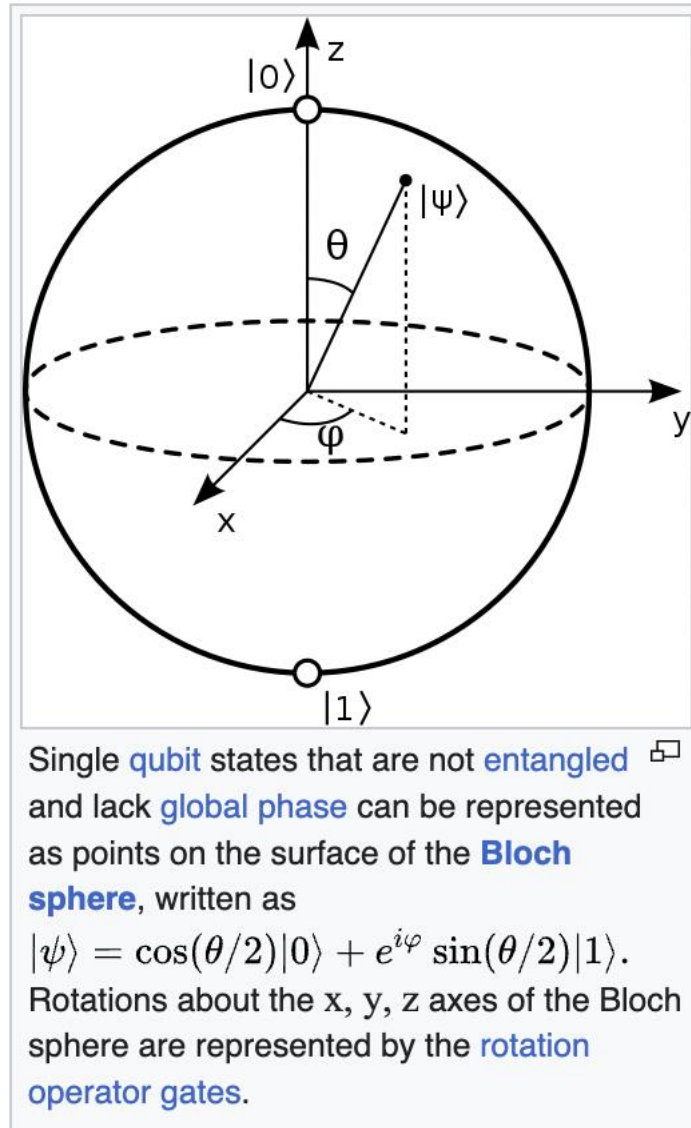


Figure 3: Bloch Sphere (Ref. Wikipedia Org)

Quantum circuits have the potential to revolutionize image processing by leveraging the unique properties of quantum mechanics. While classical methods have been highly successful in image processing tasks, quantum approaches can offer advantages such as enhanced computational power, improved feature extraction, and increased robustness to noise. In this explanation, we will explore how quantum circuits can help in image processing.

1. Quantum Superposition:

One of the key advantages of quantum circuits is the ability to represent and manipulate information in superposition. Superposition allows quantum bits (qubits) to exist in

multiple states simultaneously, providing a parallelism not available in classical computing. In the context of image processing, this parallelism can be utilized to process multiple images or image features simultaneously, potentially speeding up computations.

2. Quantum Fourier Transform (QFT):

The Quantum Fourier Transform is a quantum analogue of the classical Fourier Transform and is a powerful tool in image processing tasks such as image compression and filtering. The QFT can efficiently extract frequency components from an image, enabling the identification of important image features. By utilizing the QFT within quantum circuits, we can perform Fourier analysis on images more efficiently compared to classical methods.

3. Quantum Image Representation:

Quantum circuits can represent and process images using quantum states. Instead of encoding images as classical pixel values, quantum image representation employs quantum states to encode image data. Quantum image representation provides a different perspective on image processing, allowing for the utilization of quantum algorithms and operations for tasks like image compression, enhancement, and feature extraction.

4. Quantum Image Compression:

Quantum circuits can be employed for image compression, a fundamental aspect of image processing. Quantum image compression techniques aim to reduce the storage requirements of images while maintaining their visual quality. Quantum algorithms, such as the Quantum Singular Value Decomposition (QSVD), can efficiently compress images by utilizing the superposition and entanglement properties of qubits. QSVD-based compression techniques can potentially achieve higher compression ratios compared to classical compression methods.

5. Quantum Image Enhancement:

Image enhancement techniques are used to improve the visual quality of images by adjusting brightness, contrast, and other parameters. Quantum circuits can leverage quantum algorithms to enhance images by exploiting quantum parallelism and the

principles of quantum information theory. Quantum image enhancement techniques can provide novel ways of manipulating image data to reveal hidden details and enhance image characteristics.

6. Quantum Pattern Recognition:

Quantum circuits can also aid in pattern recognition tasks, such as object detection and image classification. Quantum algorithms, such as the Quantum Support Vector Machine (QSVM), can efficiently classify images based on their features. QSVM utilizes quantum parallelism to train and classify images, potentially providing advantages over classical machine learning algorithms. Quantum pattern recognition techniques have the potential to improve the accuracy and efficiency of image classification tasks.

7. Quantum Image Filtering:

Image filtering is a common image processing operation used for noise removal, edge detection, and image enhancement. Quantum circuits can enable quantum image filtering techniques that exploit the principles of quantum mechanics to achieve improved filtering results. Quantum filters can leverage quantum operations to process images more effectively, offering potential advantages over classical filtering methods.

8. Quantum Image Segmentation:

Image segmentation is the process of partitioning an image into meaningful regions or objects. Quantum circuits can contribute to quantum image segmentation techniques by leveraging quantum algorithms, such as the Quantum Hough Transform, to identify and segment objects in images. Quantum image segmentation techniques can offer new perspectives and potential improvements in the accuracy and efficiency of image segmentation tasks.

Advantages of Quantum Circuits in Image Processing:

1. Enhanced Computational Power: Quantum circuits have the potential to perform certain image processing tasks faster than classical methods. The parallelism inherent in quantum computing can enable simultaneous processing of multiple image features, leading to accelerated computations.

2. Improved Feature Extraction: Quantum circuits can leverage the principles of quantum mechanics, such as superposition and entanglement

, to extract and analyze image features more effectively. Quantum algorithms like the QFT can provide a more efficient representation of image frequencies, facilitating improved feature extraction.

3. Robustness to Noise: Quantum error correction techniques can enhance the robustness of quantum circuits against noise, making them potentially more resilient to image degradation caused by noise or imperfections in image acquisition.

4. Novel Perspectives: Quantum image processing approaches offer new perspectives on image representation and manipulation. By utilizing quantum states and operations, quantum circuits provide alternative methods for image compression, enhancement, and pattern recognition, potentially unlocking new possibilities in these areas.

Drawbacks and Challenges:

1. Limited Qubit Availability: Practical quantum computers currently have a limited number of qubits available, which can limit the complexity and size of images that can be processed. Scaling up the number of qubits while maintaining their coherence and connectivity remains a challenge.

2. Noise and Error Correction: Quantum circuits are susceptible to noise and errors due to environmental factors and imperfections in physical qubits. Implementing error correction techniques is crucial to mitigate the impact of noise and maintain the accuracy of image processing tasks.

3. Implementation Complexity: Designing and implementing quantum circuits for image processing tasks can be complex. Mapping image data onto quantum states, selecting appropriate quantum gates, and optimizing circuit performance require careful considerations and expertise in quantum computing.

4. Limited Quantum Algorithms: Although quantum algorithms for image processing are emerging, the field is still in its early stages. The development of efficient and robust quantum algorithms tailored for image processing tasks is an ongoing research area.

In conclusion, quantum circuits hold significant promise for image processing by leveraging the unique properties of quantum mechanics. They can enhance computational power, improve feature extraction, and provide new perspectives on image representation and manipulation. However, challenges such as limited qubit availability, noise, and error correction, as well as the complexity of implementation, need to be addressed. Continued research and advancements in quantum computing will be crucial in harnessing the full potential of quantum circuits for image processing tasks.

Making a quantum circuit involves designing the circuit based on the desired quantum algorithm, selecting appropriate gates, mapping logical qubits to physical qubits, and arranging the gates in a specific order. Quantum circuits provide the potential for universal computation, speedup in specific tasks, and utilization of quantum properties such as superposition and entanglement. However, they face challenges related to noise, limited qubit availability, error correction, measurement limitations, and circuit complexity. Overcoming these challenges is crucial for advancing the field of quantum computing and harnessing its true potential.

3.1.3 COMBINED MODEL

The model that we have formulated uses quantum circuits to accelerate and improve the existing CNN Network and help in better classification.

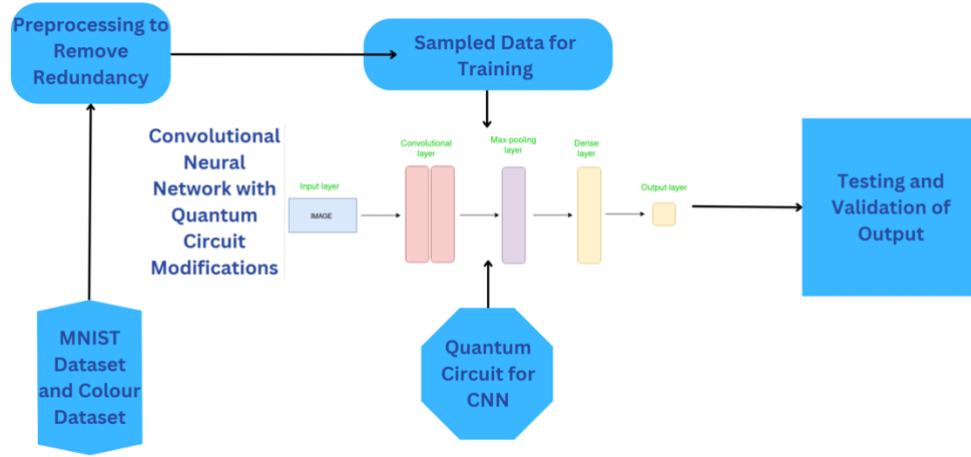


Figure 4: System Architecture

The combined modeling of quantum circuits and Convolutional Neural Networks (CNNs) holds the potential to enhance CNN performance and capabilities. By leveraging the advantages of both approaches, we can explore new avenues for improving CNNs in various aspects. Here are several ways in which the combined modeling of quantum circuits and CNNs can contribute to CNN improvement:

1. **Quantum-inspired Operations:** Quantum circuits can inspire the design of novel operations and layers within CNNs. For example, quantum-inspired convolutional layers can be designed by utilizing the principles of quantum superposition and entanglement to perform more efficient and effective feature extraction. These layers can potentially capture more complex patterns and improve CNN performance in tasks such as image classification and object detection.
2. **Quantum Feature Extraction:** Quantum circuits can be employed to enhance feature extraction in CNNs. By incorporating quantum feature extraction techniques, such as the Quantum Fourier Transform (QFT), within CNN architectures, it is possible to extract frequency-based features from images more efficiently. This can lead to improved recognition and classification accuracy, particularly in tasks that involve analyzing image patterns at different frequencies.
3. **Quantum Data Encoding:** Quantum circuits can be used to encode image data in a quantum format, providing a different representation that can be exploited by CNNs.

Quantum data encoding techniques enable the utilization of quantum properties like superposition and entanglement to represent and manipulate image information. This can potentially enhance the robustness and expressiveness of CNN models.

4. Quantum-inspired Optimization: Optimization is a critical aspect of CNN training. Quantum-inspired optimization algorithms, such as Quantum Gradient Descent, can be utilized to optimize CNN parameters. These algorithms leverage quantum principles and quantum circuit simulations to enhance gradient-based optimization, potentially leading to faster convergence and improved training performance.

5. Quantum-inspired Regularization: Regularization techniques are employed to prevent overfitting in CNN models. Quantum-inspired regularization methods can be explored to enhance CNN generalization capabilities. For example, quantum-inspired techniques based on quantum entanglement and quantum coherence principles can be developed to regulate the learning process and improve the generalization of CNN models.

6. Quantum-inspired Architectures: Quantum circuits can inspire the design of novel CNN architectures. By incorporating quantum-inspired principles, such as quantum parallelism and quantum interference, into the architecture design, it is possible to explore new CNN structures that can enhance performance. These architectures can exploit quantum properties to process image data more efficiently, potentially leading to improved accuracy and computational efficiency.

7. Hybrid Quantum-Classical Approaches: Hybrid quantum-classical approaches involve combining quantum circuits with classical neural networks. By incorporating quantum circuits as a part of the overall architecture, it is possible to leverage the advantages of both quantum and classical processing. Quantum circuits can be used for specific tasks, such as feature extraction or optimization, while classical neural networks handle other aspects, such as decision-making and final predictions. This hybrid approach can harness the strengths of quantum circuits while maintaining the practicality of classical neural networks.

While the combined modeling of quantum circuits and CNNs presents exciting opportunities, it also faces challenges:

1. **Limited Quantum Resources:** Practical quantum computers currently have limited qubit availability and are prone to noise and errors. Scaling up quantum resources and improving qubit quality is necessary to fully leverage the potential of combined models.
2. **Hardware and Software Integration:** Integrating quantum and classical hardware and software systems can be challenging. Developing efficient frameworks and tools that seamlessly integrate quantum circuits with CNN architectures is crucial for practical implementation.
3. **Training Complexity:** Training combined quantum-CNN models can be complex due to the unique properties of quantum circuits. Developing effective training algorithms and methodologies specific to these models is an ongoing research area.

The combined modeling of quantum circuits and CNNs offers exciting prospects for improving CNN performance and capabilities. By incorporating quantum-inspired operations, feature extraction, optimization, and architectures, we can explore novel approaches to enhance CNNs.

However, addressing challenges related to limited quantum resources, hardware-software integration, and training complexity is essential for the practical realization of these advancements. Continued research and development in the field of quantum computing will pave the way for fruitful integration with CNNs and further advancements in image processing and deep learning.

3.2 USE CASE DIAGRAM

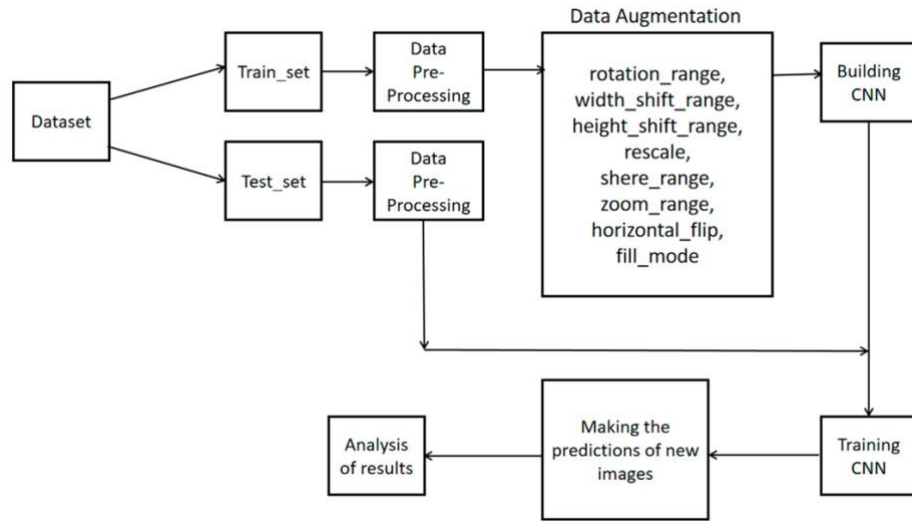


Figure 5: Use Case Diagram (Ref: MDPI)

Here is the use case for a quantum CNN classifier:

The primary actor in the use case diagram is the User, who interacts with the quantum CNN classifier system. The following use cases can be identified:

1. **Train Quantum CNN Model:** This use case represents the user's ability to train the quantum CNN classifier model. The user provides training data, selects appropriate quantum algorithms and parameters, and initiates the training process.
2. **Evaluate Quantum CNN Model:** This use case involves evaluating the performance of the trained quantum CNN model. The user provides test data and executes the model to obtain classification results.
3. **Classify Images:** This use case allows users to classify images using the trained quantum CNN model. The user provides input images to the system, and the model performs classification using quantum algorithms and returns the predicted classes.
4. **Visualize Results:** This use case enables users to visualize the classification results. The system presents the output in a user-friendly format, such as displaying the classified images or generating performance metrics.

5. Manage Training Data: This use case allows users to manage the training data used to train the quantum CNN model. It includes tasks such as data selection, preprocessing, and storage.

6. Update Quantum CNN Model: This use case represents the user's ability to update or retrain the quantum CNN model with new data. It involves selecting the updated training data, executing the training process, and replacing the previous model.

Additionally, the diagram may include other external actors or systems that interact with the quantum CNN classifier, such as a Data Provider who supplies the training and test data, or a Database System that stores and retrieves the image data.

The use case diagram visually captures the interactions and functionalities of a quantum CNN classifier system, providing an overview of how users and external entities interact with the system's various use cases.

CHAPTER 4

PROPOSED DESIGN

4.1 ESTIMATOR QUANTUM CIRCUIT

An estimator quantum circuit is a circuit used in quantum machine learning to estimate parameters or quantities of interest in a quantum system. It is designed to leverage the computational power of quantum computers to perform efficient estimation tasks. The circuit consists of a sequence of quantum gates that manipulate quantum states and measurements that extract information about the system.

The working of an estimator quantum circuit involves several key steps. First, an initial quantum state is prepared, typically a superposition of states representing different parameter values. The gates in the circuit then act on this state, effectively encoding the parameter information into the quantum state. The gates used in the circuit depend on the specific estimation task and the system being studied.

After the gates have acted on the state, measurements are performed on selected qubits in the circuit. These measurements extract information about the parameter being estimated. The outcomes of these measurements are probabilistic, and repeated measurements are performed to obtain statistics that allow for parameter estimation. Statistical techniques such as maximum likelihood estimation or Bayesian inference are typically employed to analyze the measurement outcomes and estimate the parameter of interest.

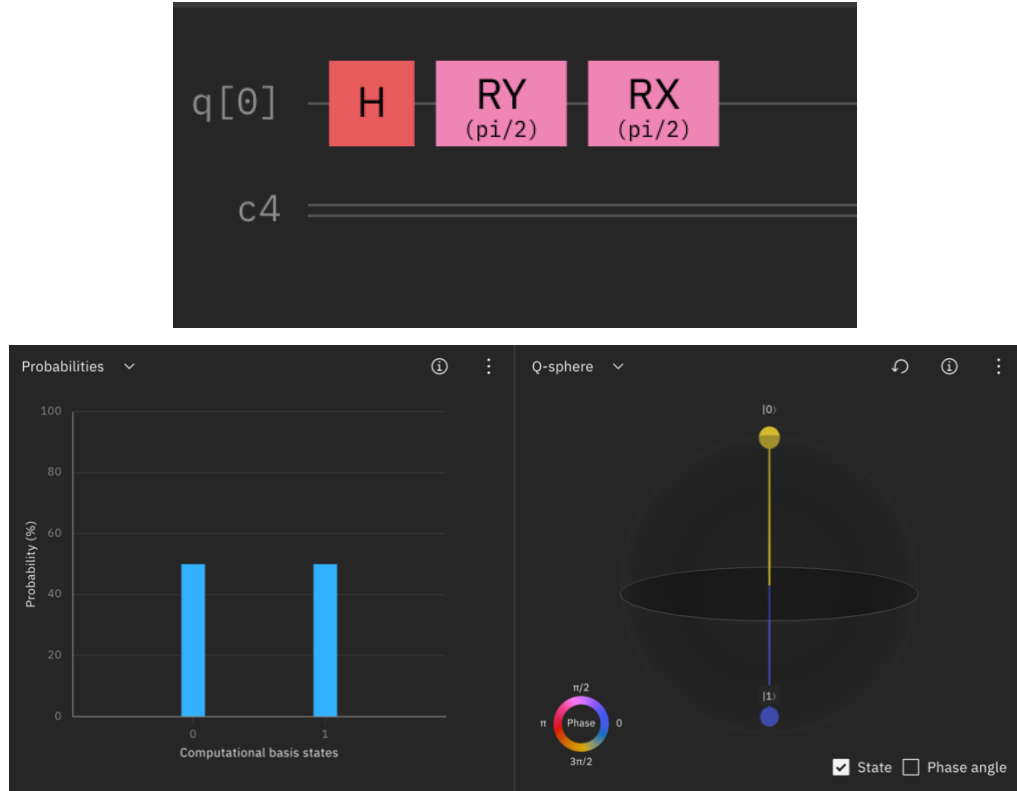


Figure 6: Estimator Quantum Circuit Simulated on IBM Quantum Composer

The performance of an estimator quantum circuit depends on various factors, including the number of qubits, the complexity of the gate operations, and the number of measurements performed. The circuit design aims to strike a balance between accurately encoding the parameter information and minimizing the computational resources required.

Estimator quantum circuits have the potential to provide advantages over classical estimation methods for certain problems. They can leverage quantum parallelism and entanglement to perform estimation tasks more efficiently than classical algorithms. However, it is important to note that the practical implementation of such circuits faces significant challenges due to noise and errors in current quantum hardware.

So, an estimator quantum circuit is a specialized circuit designed to estimate parameters or quantities of interest in a quantum system. It involves preparing an initial state, applying gates to encode parameter information, performing measurements, and employing statistical techniques to estimate the parameter. While still a developing

field, quantum machine learning holds promise for tackling complex estimation tasks in the future.

4.2 INITIATOR QUANTUM CIRCUIT

Quantum teleportation is a fundamental concept in quantum information theory that allows the transfer of quantum states from one location to another, without physically moving the particles themselves. The process involves two parties: the sender (Alice) and the receiver (Bob), along with a shared entangled pair of qubits and a classical communication channel.

The quantum teleportation circuit implements this concept and consists of the following steps:

1. Initialization: Alice and Bob share an entangled pair of qubits, usually referred to as the Bell state or EPR pair. This pair is prepared in a maximally entangled state such as $(|00\rangle + |11\rangle) / \sqrt{2}$.
2. State Preparation: The quantum state that Alice wants to teleport, let's call it $|\psi\rangle$, is prepared on a separate qubit, which we'll refer to as qubit A. The state $|\psi\rangle$ can be any arbitrary qubit state.
3. Entanglement and Measurement: Alice performs a Controlled-Not (CNOT) gate operation between qubit A (the state to be teleported) and her part of the entangled pair. She then applies a Hadamard gate (H) to qubit A, followed by a measurement in the Bell basis (measurement in the computational basis followed by a measurement in the Hadamard basis).
4. Measurement Results: Alice obtains two classical measurement outcomes, which we'll refer to as a and b.

5. Communication: Alice sends her classical measurement outcomes (a and b) to Bob over a classical communication channel. These two bits of classical information carry the necessary information to recover the teleported state.

6. State Reconstruction: Upon receiving the classical information from Alice, Bob performs operations on his entangled qubit pair based on the measurement outcomes a and b.

If $a = 0$ and $b = 0$, Bob applies no operation.

If $a = 0$ and $b = 1$, Bob applies a Pauli-X gate (bit-flip gate).

If $a = 1$ and $b = 0$, Bob applies a Pauli-Z gate (phase-flip gate).

If $a = 1$ and $b = 1$, Bob applies both the Pauli-X and Pauli-Z gates.

7. Result: After Bob performs the appropriate operations based on the received measurement outcomes, the qubit in his possession is now in the state $|\psi\rangle$, which is an exact replica of the original state prepared by Alice. The teleportation process is complete.

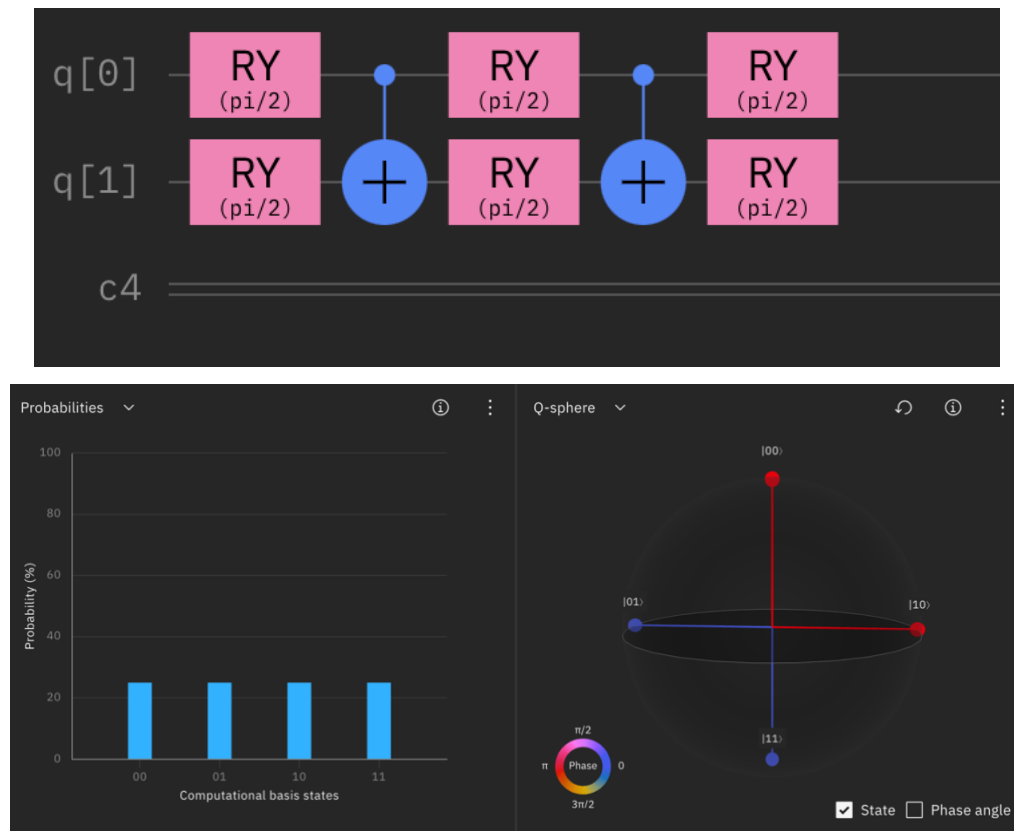


Figure 6: Initiator Quantum Circuit Simulated on IBM Quantum Composer

The quantum teleportation circuit showcases the non-local nature of entanglement and enables the transfer of quantum information between distant qubits. It has significant implications for quantum communication and quantum computation protocols, enabling the transmission of quantum states without physically moving particles through an intermediate medium.

4.3 DEUTSCH JOZSA ALGORITHM

The Deutsch algorithm is a quantum algorithm that illustrates the power of quantum computing by solving a specific problem more efficiently than classical algorithms. It is a simple example of a quantum algorithm that exhibits quantum parallelism and interference. The problem it solves is known as the Deutsch problem, which involves determining whether a given Boolean function is constant (returns the same output for all inputs) or balanced (returns different outputs for at least half of the inputs).

Here are the details of the Deutsch algorithm:

1. Initialization: The algorithm begins with the preparation of two quantum registers: one input register (consisting of two qubits) and one output register (consisting of one qubit). The qubits are initially set to the $|0\rangle$ state.
2. Superposition: A Hadamard gate (H) is applied to both qubits in the input register, creating a superposition of all possible input states. The input register is now in the state: $(|0\rangle + |1\rangle) \otimes (|0\rangle + |1\rangle)$.
3. Oracle Function: An oracle unitary operation, denoted as U_f , is applied to the input and output registers. The oracle function represents the given Boolean function that we want to evaluate. The oracle function U_f flips the phase of the output qubit for inputs that satisfy a specific property (constant or balanced) defined by the Boolean function.
4. Interference: Another Hadamard gate (H) is applied to the qubits in the input register.

5. Measurement: Finally, a measurement is performed on the two qubits in the input register. The measurement outcomes will determine the nature of the Boolean function (constant or balanced).

6. Result Interpretation: The measurement outcomes yield one of the four possible results:

If both qubits measure 0, the Boolean function is constant.

If the first qubit measures 0 and the second qubit measures 1, the Boolean function is balanced.

The reason why the Deutsch algorithm is more efficient than classical algorithms for this problem lies in the quantum parallelism and interference. In the classical case, we would need to evaluate the Boolean function twice to determine if it is balanced or constant. However, in the Deutsch algorithm, due to quantum superposition and interference, the evaluation is performed in a single step.

This algorithm demonstrates the power of quantum computing by providing a quadratic speedup compared to classical algorithms for the Deutsch problem. While this problem is simple and the speedup may not be substantial, it serves as a starting point to understand the principles of quantum computing and the advantages it can offer for more complex problems.

4.4 QUANTUM OPTIMIZED CNN

Quantum Convolutional Neural Networks (QCNNs) are a type of quantum machine learning model that leverage the principles of quantum computing to process and analyze data using convolutional neural network (CNN) architecture. QCNNs extend traditional CNNs to exploit the computational power of quantum systems, potentially offering advantages in certain domains. Here's a step-by-step description of how a quantum convolutional neural network works:

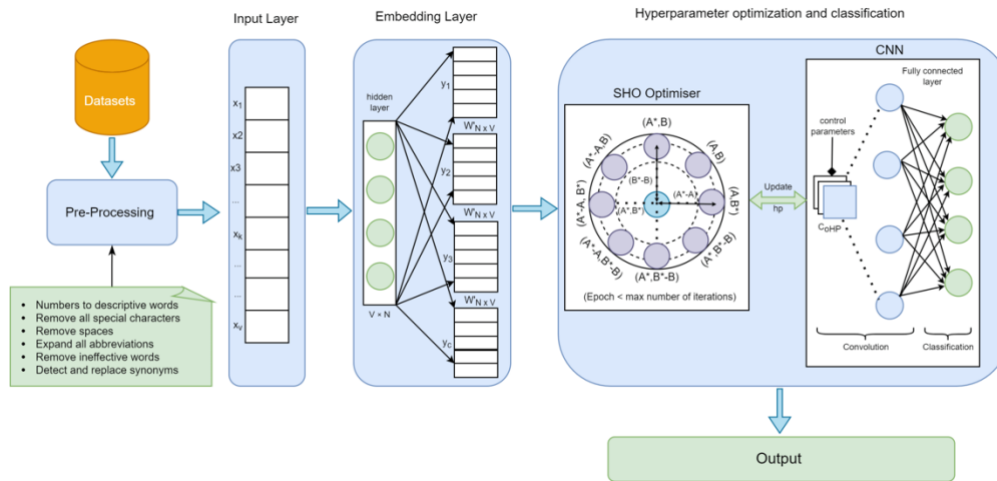


Figure 7: Quantum CNN (Ref: MDPI)

1. Quantum Data Encoding:

The input data, such as images or feature maps, is encoded into a quantum state representation suitable for quantum computation. This encoding process can involve various techniques, such as amplitude encoding or feature mapping, where the data is mapped to the amplitudes or angles of quantum states.

2. Quantum Convolution:

The quantum convolutional layer applies convolutional operations to the quantum data. This layer involves a series of quantum gates that act on the quantum state representation, mimicking the convolutional operation in classical CNNs. These gates perform operations such as controlled rotations or controlled swaps to capture local correlations and extract features from the input quantum state.

3. Quantum Pooling:

Quantum pooling operations are applied to reduce the spatial dimensions of the quantum data. Like classical pooling layers, these operations aggregate information from neighboring quantum states. Quantum pooling can be implemented using techniques like quantum phase estimation or quantum subspace expansion to extract relevant features while discarding unnecessary information.

4. Quantum Non-Linearity:

Activation functions are introduced to introduce non-linearity into the quantum convolutional layer. These functions transform the quantum states in a non-linear fashion, allowing the network to model complex relationships and capture higher-level

features. Common activation functions, such as the quantum variant of the rectified linear unit (ReLU), can be applied to the quantum states.

5. Quantum Classification:

The processed quantum data is fed into the classification layer, which maps the learned features to class labels or probabilities. This layer typically employs quantum measurements to extract information from the quantum states and perform classification. Depending on the specific task, techniques like amplitude estimation or quantum support vector machines can be used for classification.

6. Training and Optimization:

Training a QCNN involves optimizing the network parameters to minimize a chosen loss function. This process typically employs gradient-based optimization algorithms such as stochastic gradient descent. Backpropagation techniques can be used to compute gradients of the loss function with respect to the network parameters. The optimization process aims to update the parameters iteratively to improve the network's performance on the training data.

7. Quantum Circuit Compilation:

To implement the quantum operations and measurements required by the QCNN, a quantum circuit must be constructed. The circuit represents a sequence of gates and measurements that can be executed on a quantum computer or simulator. Circuit compilation techniques are employed to map the high-level QCNN architecture to the specific gate set and connectivity constraints of the target quantum hardware.

8. Quantum Hardware Execution:

Once the quantum circuit is compiled, it can be executed on a quantum computer or simulator. The quantum operations and measurements are applied to the encoded quantum data, and the resulting quantum state is processed through the network. The measurement outcomes are obtained, and the classification or regression results are derived based on these outcomes.

9. Evaluation and Validation:

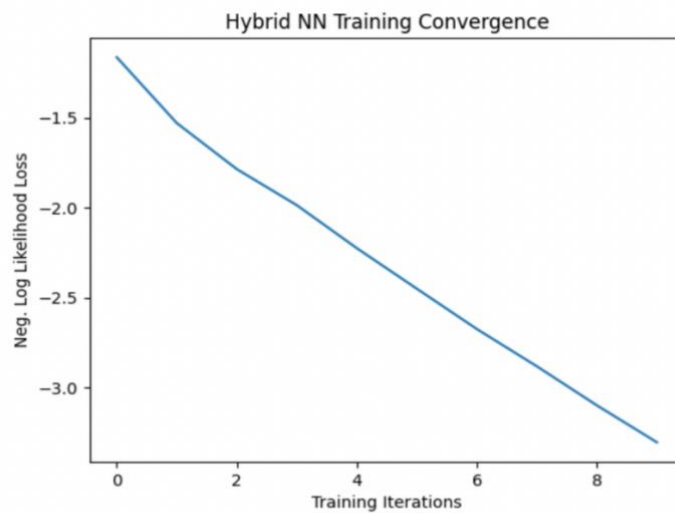
The performance of the QCNN is evaluated using metrics such as accuracy, precision, recall, or F1 score. The trained model is tested on independent test data to assess its generalization ability and its ability to make accurate predictions on unseen data. Cross-validation techniques can be used to estimate the model's performance and address issues like overfitting.

It's important to note that quantum convolutional neural networks are an active area of research and development, and their practical implementation is still limited due to the current constraints of quantum hardware, such as noise, limited qubit coherence, and gate errors. However, advancements in quantum computing technology and algorithm design are continually pushing the boundaries of QCNNs, offering potential applications in fields such as computer vision, pattern recognition, and quantum image processing.

CHAPTER 5

RESULTS

5.1 GRAYSCALE MNIST ON IBM QISKIT



100 Epochs and LR 0.001



Labelled Data



Predicted Data

Performance on test data:

Loss: -3.4254

Accuracy: 99.9%

Figure 8: Results on Grayscale MNIST Images

5.2 COLOUR MNIST ON GOOGLE CIRQ

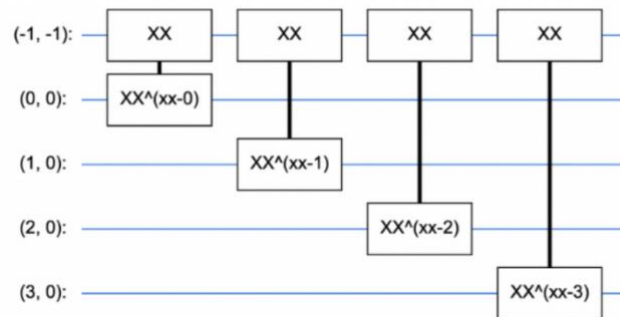
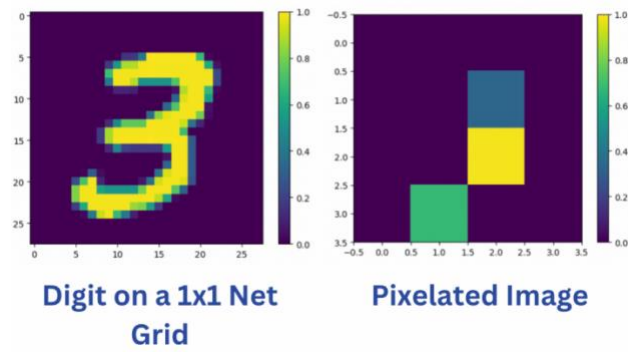


Figure 9: Results on Colour MNIST Images

5.3 CIRCUIT SIMULATION ON REAL QUANTUM COMPUTER

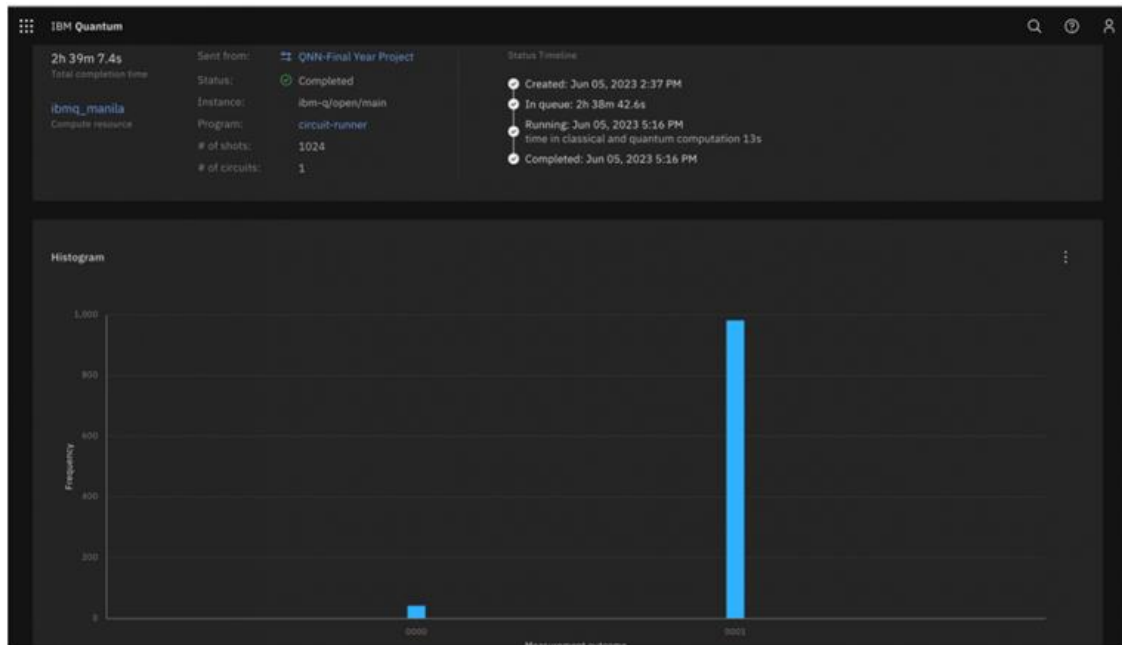


Figure 10: Simulations on IBM Manila

5.4 DIFFERENCES CAUSED BY QUANTUM ERROR AND CIRCUIT FAILURE

Quantum error refers to the occurrence of errors or disturbances in quantum systems, which can lead to deviations from the expected results on a quantum computer. These errors arise from various sources, including noise, imperfect control operations, environmental interactions, and decoherence.

Quantum systems are highly sensitive to their surroundings, making them prone to errors. The fragile nature of quantum states, known as superposition and entanglement, makes them susceptible to disturbances from factors such as thermal fluctuations, electromagnetic radiation, and interactions with neighboring particles. These disturbances introduce errors that can affect the accuracy and reliability of computations performed on a quantum computer.

One of the primary sources of quantum errors is decoherence, which refers to the loss of quantum coherence in a system. Coherence is a fundamental property of quantum

systems that enables the manipulation and storage of quantum information. However, interactions with the environment cause the system to lose coherence over time, leading to errors in quantum computations.

Decoherence arises due to the phenomenon of quantum entanglement with the environment, known as quantum entanglement or entanglement with noise. When a quantum system interacts with its environment, its state becomes entangled with the environmental degrees of freedom, leading to the loss of coherence. This entanglement spreads the quantum information into the environment, making it difficult to extract meaningful results.

The effect of quantum errors can be detrimental to the performance of quantum computations. As quantum algorithms typically rely on delicate interference patterns and precise manipulation of quantum states, even small errors can accumulate and cause significant deviations from the expected results.

These errors can manifest in several ways:

1. **Bit Flip Errors:** Bit flip errors occur when the quantum state experiences a transition from the desired state to its logical complement due to noise or unwanted interactions. For example, a qubit in the state $|0\rangle$ may flip to $|1\rangle$ or vice versa.
2. **Phase Flip Errors:** Phase flip errors involve a change in the phase of a quantum state. This can result in a rotation of the quantum state around the Bloch sphere, introducing phase shifts and altering the interference patterns.
3. **Measurement Errors:** Errors can also occur during the measurement process itself. Imperfections in measurement devices or environmental interactions can lead to incorrect measurement outcomes, affecting subsequent computations and result interpretation.

Quantum error correction (QEC) techniques are employed to mitigate the impact of quantum errors. QEC methods involve encoding quantum information redundantly, such that errors can be detected and corrected. Quantum error correction codes, such as

the surface code or the stabilizer codes, are used to protect quantum information against errors.

QEC protocols typically involve introducing additional qubits and applying error-detection and error-correction operations. These operations allow for the identification and correction of errors without directly measuring the encoded quantum information, thereby preserving coherence.

By implementing QEC techniques, quantum computers can increase the robustness of quantum computations and improve the accuracy of results. However, QEC protocols require additional computational resources, such as extra qubits and additional gate operations, which poses a challenge for current quantum hardware limitations.

These errors can cause deviations from the expected results on a quantum computer, impacting the accuracy and reliability of quantum computations. Quantum error correction techniques are being developed to mitigate the impact of these errors, enabling more reliable quantum computations, and facilitating the development of fault-tolerant quantum computers.

CHAPTER 6

CONCLUSION AND SCOPE OF FUTURE WORK

6.1 CONCLUSION

This entire project has been at the pivot of understanding and trying to simulate how quantum circuits can help us in understanding, retrieving, and working on quantum computers.

Larger scope of works such as annotations in quantum image, would involve larger data labels of all sorts and then, a classical, statistically optimized approach to achieve annotation, even after which the accuracy and data to performance load will take time to handle and measure.

This remains to be a field of larger development and scope towards major improvement in the field.

6.2 FUTURE ENHANCEMENTS

Working on images requires deep understanding of statistical measures and in this case of Quantum Image Processing, we require statistical tools to help understand image features and lattices as well as to formulate mathematical components for further analysis.

We therefore need to work on quantum image processing methods and retrieval methods to accomplish tasks like annotations and deep segmentation.

CHAPTER 7

APPENDIX

7.1 SOURCE CODE

```
!pip install qiskit
import torch
from torch import cat, no_grad, manual_seed
from torch.utils.data import DataLoader
from torchvision import datasets, transforms
import torch.optim as optim
from torch.nn import (
    Module,
    Conv2d,
    Linear,
    Dropout2d,
    NLLLoss,
    MaxPool2d,
    Flatten,
    Sequential,
    ReLU,
)
import torch.nn.functional as F
#training data
manual_seed(42)

batch_size = 1
n_samples = 100 # first 100 samples only

X_train = datasets.MNIST(
    root="./data", train=True, download=True, transform=transforms.Compose([transforms.ToTensor()])
)

# Filter out labels (originally 0-9), leaving only labels 0 and 1
idx = np.append(
    np.where(X_train.targets == 0)[0][:n_samples], np.where(X_train.targets == 1)[0][:n_samples]
)
X_train.data = X_train.data[idx]
X_train.targets = X_train.targets[idx]

# Define torch dataloader with filtered data
train_loader = DataLoader(X_train, batch_size=batch_size, shuffle=True)
n_samples_show = 10

data_iter = iter(train_loader)
fig, axes = plt.subplots(nrows=1, ncols=n_samples_show, figsize=(10, 3))

while n_samples_show > 0:
    images, targets = data_iter.__next__()

    axes[n_samples_show - 1].imshow(images[0, 0].numpy().squeeze(), cmap="gray")
    axes[n_samples_show - 1].set_xticks([])
    axes[n_samples_show - 1].set_yticks([])
    axes[n_samples_show - 1].set_title("L: {}".format(targets[0].item()))

    n_samples_show -= 1
n_samples = 500

# Use pre-defined torchvision function to load MNIST test data
X_test = datasets.MNIST(
    root="./data", train=False, download=True, transform=transforms.Compose([transforms.ToTensor()])
)

# Filter out labels (originally 0-9), leaving only labels 0 and 1
idx = np.append(
    np.where(X_test.targets == 0)[0][:n_samples], np.where(X_test.targets == 1)[0][:n_samples]
)
X_test.data = X_test.data[idx]
X_test.targets = X_test.targets[idx]

# Define torch dataloader with filtered data
test_loader = DataLoader(X_test, batch_size=batch_size, shuffle=True)
def create_qnn():
    feature_map = ZZFeatureMap(2)
    ansatz = RealAmplitudes(2, reps=1)
    qc = QuantumCircuit(2)
```

```

qc.compose(feature_map, inplace=True)
qc.compose(ansatz, inplace=True)

# REMEMBER TO SET input_gradients=True FOR ENABLING HYBRID GRADIENT BACKPROP
qnn = EstimatorQNN(
    circuit=qc,
    input_params=feature_map.parameters,
    weight_params=ansatz.parameters,
    input_gradients=True,
)
return qnn

qnn4 = create_qnn()
class Net(Module):
    def __init__(self, qnn):
        super().__init__()
        self.conv1 = Conv2d(1, 2, kernel_size=5)
        self.conv2 = Conv2d(2, 16, kernel_size=5)
        self.dropout = Dropout2d()
        self.fc1 = Linear(256, 64)
        self.fc2 = Linear(64, 2) # 2-dimensional input to QNN
        self.qnn = TorchConnector(qnn) # Apply torch connector, weights chosen
        # uniformly at random from interval [-1,1].
        self.fc3 = Linear(1, 1) # 1-dimensional output from QNN

    def forward(self, x):
        x = F.relu(self.conv1(x))
        x = F.max_pool2d(x, 2)
        x = F.relu(self.conv2(x))
        x = F.max_pool2d(x, 2)
        x = self.dropout(x)
        x = x.view(x.shape[0], -1)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        x = self.qnn(x) # apply QNN
        x = self.fc3(x)
        return cat((x, 1 - x), -1)

optimizer = optim.Adam(model4.parameters(), lr=0.001)
loss_func = NLLLoss()

# Start training
epochs = 10 # Set number of epochs
loss_list = [] # Store loss history
model4.train() # Set model to training mode

for epoch in range(epochs):
    total_loss = []
    for batch_idx, (data, target) in enumerate(train_loader):
        optimizer.zero_grad(set_to_none=True) # Initialize gradient
        output = model4(data) # Forward pass
        loss = loss_func(output, target) # Calculate loss
        loss.backward() # Backward pass
        optimizer.step() # Optimize weights
        total_loss.append(loss.item()) # Store loss
    loss_list.append(sum(total_loss) / len(total_loss))
    print("Training [{:.0f}%]\tLoss: {:.4f}".format(100.0 * (epoch + 1) / epochs, loss_list[-1]))
plt.plot(loss_list)
plt.title("Hybrid NN Training Convergence")
plt.xlabel("Training Iterations")
plt.ylabel("Neg. Log Likelihood Loss")
plt.show()
torch.save(model4.state_dict(), "model4.pt")
qnn5 = create_qnn()
model5 = Net(qnn5)
model5.load_state_dict(torch.load("model4.pt"))
model5.eval() # set model to evaluation mode
with no_grad():

    correct = 0
    for batch_idx, (data, target) in enumerate(test_loader):
        output = model5(data)
        if len(output.shape) == 1:
            output = output.reshape(1, *output.shape)

```

```

pred = output.argmax(dim=1, keepdim=True)
correct += pred.eq(target.view_as(pred)).sum().item()

loss = loss_func(output, target)
total_loss.append(loss.item())

print(
    "Performance on test data:\n\tLoss: {:.4f}\n\tAccuracy: {:.1f}%".format(
        sum(total_loss) / len(total_loss), correct / len(test_loader) / batch_size * 100
    )
)
n_samples_show = 10
count = 0
fig, axes = plt.subplots(nrows=1, ncols=n_samples_show, figsize=(10, 3))

model5.eval()
with no_grad():
    for batch_idx, (data, target) in enumerate(test_loader):
        if count == n_samples_show:
            break
        output = model5(data[0:1])
        if len(output.shape) == 1:
            output = output.reshape(1, *output.shape)

        pred = output.argmax(dim=1, keepdim=True)

        axes[count].imshow(data[0].numpy().squeeze(), cmap="gray")

        axes[count].set_xticks([])
        axes[count].set_yticks([])
        axes[count].set_title("P: {}".format(pred.item()))

        count += 1

```

7.2 FORMULAS USED

$$\begin{aligned}
 sim(d, q) &= tr(\sum_i (t_i^d \cdot f_j^d)^2 |t_i f_j\rangle \langle t_i f_j| \\
 &\quad \cdot (t_i^q \cdot f_j^q)^2 |t_i f_j\rangle \langle t_i f_j|) \\
 &= trace(\rho_d \cdot \rho_q) \\
 &= \sum_{ij} (t_i^d \cdot f_j^d)^2 (t_i^q \cdot f_j^q)^2
 \end{aligned}$$

Figure 11: Measurements in Tensor Space

$$recall(C_i) = \frac{|a_{correct}(C_i)|}{|a_{total}(C_i)|}, precision(C_i) = \frac{|a_{correct}(C_i)|}{|a_{system}(C_i)|}$$

Figure 12: Annotation Recall and Validation

$$P_{best}(t+1) = \begin{cases} X(t), & \text{if } F(X(t)) > F(P_{best}(t)) \\ P_{best}(t), & \text{if } F(X(t)) \leq F(P_{best}(t)) \end{cases}$$

Figure 13: Best Position of Particle

REFERENCES

1. A. Geng, A. Moghiseh, C. Redenbach, and K. Schladitz, “Improved FRQI on superconducting processors and its restrictions in the NISQ era,” *Quantum Inf Process*, vol. 22, no. 2, p. 104, Feb. 2023, doi: 10.1007/s11128-023-03838-0.
2. B. Wang, M. Hao, P. Li, and Z. Liu, “Quantum Representation of Indexed Images and its Applications,” *International Journal of Theoretical Physics*, vol. 59, no. 2, pp. 374–402, Feb. 2020, doi: 10.1007/s10773-019-04331-0.
3. Y. Ruan, X. Xue, and Y. Shen, “Quantum Image Processing: Opportunities and Challenges,” *Math Probl Eng*, vol. 2021, pp. 1–8, Jan. 2021, doi: 10.1155/2021/6671613.
4. M. M. Adnan, M. S. M. Rahim, M. Hasan ali, K. Al-Jawaheri, and K. Neamah, “A Review of Methods for The Image Automatic Annotation,” *J Phys Conf Ser*, vol. 1892, no. 1, pp. 012002–012010, Apr. 2021, doi: 10.1088/1742-6596/1892/1/012002.
5. Dang, Y., Jiang, N., Hu, H., Ji, Z., & Zhang, W. (2018). Image classification based on quantum K-Nearest-Neighbor algorithm. *Quantum Information Processing*, 17, 1-18.
6. Guala, D., Zhang, S., Cruz, E., Riofrío, C. A., Klepsch, J., & Arrazola, J. M. (2023). Practical overview of image classification with tensor-network quantum circuits. *Scientific Reports*, 13(1), 4427.
7. Easom-McCaldin, P., Bouridane, A., Belatreche, A., Jiang, R., & Al-Maadeed, S. (2022). Efficient Quantum Image Classification Using Single Qubit Encoding. *IEEE Transactions on Neural Networks and Learning Systems*.
8. Mogalapalli, H., Abburi, M., Nithya, B., & Bandreddi, S. K. V. (2022). Classical–quantum transfer learning for image classification. *SN Computer Science*, 3(1), 20.
9. Parisi, L., Neagu, D., Ma, R., & Campean, F. (2022). Quantum ReLU activation for Convolutional Neural Networks to improve diagnosis of Parkinson’s disease and COVID-19. *Expert Systems with Applications*, 187, 115892.
10. Chalumuri, A., Kune, R., Kannan, S., & Manoj, B. S. (2021). Quantum-enhanced deep neural network architecture for image scene classification. *Quantum Information Processing*, 20(11), 381.
11. Duan, B., Yuan, J., Liu, Y., & Li, D. (2017). Quantum algorithm for support matrix machines. *Physical Review A*, 96(3), 032301.

12. Beach, G., Lomont, C., & Cohen, C. (2003, October). Quantum image processing (quip). In 32nd Applied Imagery Pattern Recognition Workshop, 2003. Proceedings. (pp. 39-44). IEEE.
13. Su, J., Guo, X., Liu, C., & Li, L. (2020). A new trend of quantum image representations. *IEEE Access*, 8, 214520-214537.
14. Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., ... & Chen, T. (2018). Recent advances in convolutional neural networks. *Pattern recognition*, 77, 354-377.
15. O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.
16. Wille, R., Van Meter, R., & Naveh, Y. (2019, March). IBM's Qiskit tool chain: Working with and developing for real quantum computers. In 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE) (pp. 1234-1240). IEEE.
17. Singh, P. N., & Aarthi, S. (2021, February). Quantum circuits—an application in qiskit-python. In 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV) (pp. 661-667). IEEE.
18. Pattanayak, S. (2021). *Quantum Machine Learning with Python: Using Cirq from Google Research and IBM Qiskit*. Apress.
19. Isakov, S. V., Kafri, D., Martin, O., Heidweiller, C. V., Mrućkiewicz, W., Harrigan, M. P., ... & Boixo, S. (2021). Simulations of quantum circuits with approximate noise using qsim and cirq. *arXiv preprint arXiv:2111.02396*.
20. Hevia, J. L., Peterssen, G., Ebert, C., & Piattini, M. (2021). Quantum computing. *IEEE Software*, 38(5), 7-15.