IMAGE CLASSIFICATION USING QUANTUM CONVOLUTIONAL NEURAL NETWORK

A PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE AWARD OF THE DEGREE OF

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND ENGINEERING

(DATA SCIENCE)

Submitted by

Name		Reg. No.
Abdul Qaiyum B	–	191011001
Kannan M	–	191011017
Ragupathi M	–	191011030

Project Guide

Dr. R. RAGUPATHY

Associate Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ANNAMALAI UNIVERSITY

ANNAMALAI NAGAR – 608 002

May 2023

ANNAMALAI UNIVERSITY

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

This is to certify that the Project report titled **"IMAGE CLASSIFICATION USING QUANTUM CONVOLUTIONAL NEURAL NETWORK"** is the bonafide record of the work done by

Name		Reg. No.
Abdul Qaiyum B	–	191011001
Kannan M	–	191011017
Ragupathi M	–	191011030

in partial fulfillment of the requirements for the Degree of Bachelor of Engineering in Computer Science and Engineering (Data Science) during the period 2022 - 2023.

Dr. R. RAGUPATHY
Associate Professor
Dept. of Computer Science & Engg.
Project Guide

Dr. S. PALANIVEL
Professor & Head
Dept. of Computer Science & Engg.

Internal Examiner

External Examiner

Place: Annamalai Nagar
Date:

ACKNOWLEDGEMENT

We wish to express our sincere thanks and a deep sense of gratitude to **Dr. S. Palanivel, M.E., Ph.D.,** Professor & Head, Department of Computer Science and Engineering, Faculty of Engineering and Technology, Annamalai University for giving us the opportunity to undertake this project.

We would like to convey our heartiest thanks to our project guide **Dr. R. Ragupathy**, **M.E., Ph.D.,** Associate Professor, Department of Computer Science and Engineering, for all his help and support. He with his extreme patience has guided us in situations of need for which we are extremely grateful.

We would like to thank our project review committee members **Dr. K. Kavitha**, **M.E., Ph.D.,** Associate Professor, and **Dr. R. ArunKumar**, **M.E., Ph.D.,** Associate Professor, Department of Computer Science and Engineering for their great support and encouragement during the course of our project.

We would like to thank all our friends for their support in times of need, and encouragement and were always ready to help us without asking for it. We wish to thank all the technical staff members, in charge of our department laboratories that fulfilled all our project needs and offered us timely help. Above all, we are indebted to our beloved parents, whose blessings and best wishes have come a long way in making this final year project work a grand success.

Abdul Qaiyum B

Kannan M

Ragupathi M

ABSTRACT

A novel approach for image classification using a quantum convolutional neural network (QCNN) has been proposed in this project work. The main objective of this work is to exploit the potential of quantum computing in enhancing the performance of deep learning models for image classification tasks. The proposed QCNN model is built on the principles of both convolutional neural networks (CNN) and quantum computing. Quantum gates are utilized to perform convolutional operations on the image data, which allows efficient representation and processing of the image features. QCNN model is designed and implemented on quantum simulation which is to be compatible with current quantum hardware. To evaluate the performance of the QCNN model, the experiments are conducted on several benchmark datasets. The standard performance metrics such as accuracy, precision, recall, F1-score, error rate, etc. are used to assess the performance of the proposed QCNN-based machine learning model. The Experimental results will show that the QCNN model achieves comparable or better classification accuracy in general. In particular, QCNN will outperform while reducing the number of trainable parameters. The significant advantage of the QCNN model reduces the computational cost and memory requirements of the model. The effect of different types of quantum gates will be analyzed on the performance of the model. The results of the analysis will provide insights into the most suitable quantum gates for image classification tasks and how they can be used to improve the performance of the model.

திட்டச்சுருக்கம்

குவாண்டம் கன்வல்யூஷனல் நியூரல் நெட்வொர்க் (QCNN) ஐப் பயன்படுத்தி பட வகைப்பாட்டிற்கான ஒரு புதிய அணுகுமுறை இந்த திட்டப்பணியில் முன்மொழியப்பட்டது. இந்த வேலையின் முக்கிய நோக்கம் குவாண்டம் கம்ப்யூட்டிங்கின் திறனைப் பயன்படுத்தி பட வகைப்பாடு பணிகளுக்கு ஆழ்ந்த கற்றல் மாதிரிகளின் செயல்திறனை மேம்படுத்துவதாகும். முன்மொழியப்பட்ட QCNN மாதிரியானது கன்வல்யூஷனல் நியூரல் நெட்வொர்க்குகள் (CNN) மற்றும் குவாண்டம் கம்ப்யூட்டிங் ஆகிய இரண்டின் கொள்கைகளின் அடிப்படையில் கட்டமைக்கப்பட்டுள்ளது. குவாண்டம் கேட்கள் படத் தரவுகளில் கன்வல்யூஷனல் செயல்பாடுகளைச் செய்யப் பயன்படுத்தப்படுகின்றன. இது பட அம்சங்களை திறமையான பிரதிநிதித்துவம் மற்றும் செயலாக்கத்தை அனுமதிக்கிறது. QCNN மாதிரியானது தற்போதைய குவாண்டம் வன்பொருளுடன் இணக்கமாக இருக்கும் குவாண்டம் உருவகப்படுத்துதலில் வடிவமைக்கப்பட்டு செயல்படுத்தப்படுகிறது. QCNN மாதிரியின் செயல்திறனை மதிப்பிட, சோதனைகள் பல தரநிலை தரவுத்தொகுப்புகளில் நடத்தப்படுகின்றன. முன்மொழியப்பட்ட QCNN அடிப்படையிலான இயந்திரக் கற்றல் மாதிரியின் செயல்திறனை மதிப்பிடுவதற்கு துல்லியம், துல்லியம், நினைவுபடுத்துதல், F1 மதிப்பெண், பிழை விகிதம் போன்ற நிலையான செயல்திறன் அளவீடுகள் பயன்படுத்தப்படுகின்றன. QCNN மாதிரி பொதுவாக ஒப்பிடக்கூடிய அல்லது சிறந்த வகைப்பாடு துல்லியத்தை அடைகிறது என்பதை சோதனை முடிவுகள் காண்பிக்கும். குறிப்பாக, பயிற்சியளிக்கக்கூடிய அளவுருக்களின் எண்ணிக்கையைக் குறைக்கும் போது QCNN சிறப்பாகச் செயல்படும். QCNN மாதிரியின் குறிப்பிடத்தக்க நன்மை, மாதிரியின் கணக்கீட்டு செலவு மற்றும் நினைவக தேவைகளை குறைக்கிறது. பல்வேறு வகையான குவாண்டம் கேட்களின் விளைவு மாதிரியின் செயல்திறனில் பகுப்பாய்வு செய்யப்படும். பகுப்பாய்வின் முடிவுகள், பட வகைப்பாடு பணிகளுக்கு மிகவும் பொருத்தமான குவாண்டம் வாயில்கள் மற்றும் மாதிரியின் செயல்திறனை மேம்படுத்த அவற்றை எவ்வாறு பயன்படுத்தலாம் என்பது பற்றிய நுண்ணறிவுகளை வழங்கும்.

TABLE OF CONTENTS

CHAPTER NO.	CONTENT	PAGE NO.
	Abstract (English)	i
	Abstract (Tamil)	ii
	List of Figures	v
	List of Tables	vi
	List of Abbreviations	vii
1	**INTRODUCTION**	
	1.1 Introduction 1.2 Problem Statement 1.3 Objective's 1.4 Organization of the Thesis	1 2 3 3
2	**LITERATURE SURVEY**	5
3	**METHODOLOGY**	
	3.1 Introduction 3.2 System Requirements 3.3 Approaches Utilized 3.4 Proposed System	7 7 8 14
4	**IMPLEMENTATION**	
	4.1 Dependencies 4.2 Data preprocessing 4.3 Classical neural network 4.4 Quantum neural network	17 18 21 22

5	**EXPERIMENTAL RESULTS AND DISCUSSION**	
	5.1 Confusion Matrix 5.2 Performance Metrics 5.3 Discussion	24 26 29
6	**CONCLUSION AND FUTURE WORKS**	30
	REFERENCES	32

LIST OF FIGURES

FIGURE NO.	NAME	PAGE NO.
3.1	Conventional layer	9
3.2	Padding layer	10
3.3	Activation Functions	10
3.4	Flattening layer	11
3.5	Fully connected	12
3.6	Dropout	12
3.7	Quantum Convolutional Neural Network	13
3.8	Block Diagram	14
4.1	Workflow	16
4.2	MNIST	19
5.1	Confusion Matrix	24
5.2	CNN Confusion Matrix	25
5.3	QCNN Confusion Matrix	26

LIST OF TABLES

TABLE NO.	TABLE NAME	PAGE NO.
3.1	Python libraries	8
4.1	Pip installation	17
5.1	Performance Metrics Table	28

LIST OF ABBREVIATIONS

S. NO	ABBREVIATION	EXPANDED FORM
1	CNN	Convolutional Neural Network
2	QCNN	Quantum Convolutional Neural Network
3	ADAM	Adaptive Moment Estimation
4	ReLU	Rectified Linear Unit
5	TP	True Positive
6	TN	True Negative
7	FP	False Positive
8	FN	False Negative
9	MNIST	Modified National Institute of Standards and Technology
10	TensorFlow-Q	Tensorflow Quantum

CHAPTER – 1

INTRODUCTION

1.1 INTRODUCTION

Image classification is a crucial task in computer vision that plays a fundamental role in a wide range of applications, including object recognition, image search, and content-based image retrieval. The task involves assigning labels or categories to images based on their visual content, making it a critical component of many computer vision systems. Deep learning models, particularly convolutional neural networks (CNNs), have made significant contributions to the field of image classification. These models use multiple layers of convolutional filters and pooling operations to extract features from images, followed by fully connected layers that classify the images into different categories. CNNs have achieved state-of-the-art performance on various benchmark datasets, including CIFAR-10, ImageNet, and MNIST, among others. Despite their success, these models still require significant computational resources and can be computationally expensive for large-scale datasets. This limitation has led researchers to explore alternative approaches that can enhance the efficiency and speed of image classification tasks. One such approach that has gained significant attention in recent years is quantum computing.

Quantum computing is a rapidly evolving field that promises to revolutionize the way we process information. Unlike classical computers that use bits to represent information, quantum computers use quantum bits or qubits that can represent multiple states simultaneously. This property of quantum computing allows for the efficient processing of large amounts of data and the solution of complex problems that are computationally infeasible on classical computers. In the context of image classification, researchers have been exploring the potential of quantum computing to enhance the efficiency and speed of image classification tasks. One promising approach is the use of quantum convolutional neural networks (QCNNs), which leverage the power of quantum computing to accelerate the computation of convolutional operations. Overall, the

exploration of quantum computing for image classification represents an exciting new frontier in computer vision research. By leveraging the unique properties of quantum computing, researchers aim to develop more efficient and accurate image classification models that can be applied to large-scale datasets. QCNN are a class of quantum machine learning models that leverage the principles of quantum computing to enhance the efficiency and speed of image classification tasks. QCNNs encode the input data as quantum circuits and use quantum gates to perform computations on the input data. QCNNs have been shown to offer several advantages over classical CNNs, including faster training times and better accuracy on certain types of image classification tasks. While QCNNs offer several potential benefits, they are still in the early stages of development, and much research is needed to explore their full potential. The performance of QCNNs is highly dependent on the hardware and software platforms used for their implementation, which are still in the process of being developed and optimized. Additionally, QCNNs require specialized knowledge in both quantum computing and machine learning, which can be a barrier to their adoption. Nonetheless, as quantum computing technology continues to advance, QCNNs are likely to become increasingly important in the field of image classification.

1.2 PROBLEM STATEMENT

The limitations of CNNs in terms of their computational requirements and their potential for accuracy improvement pose a significant challenge to researchers. Additionally, as the size of image datasets continues to grow, the need for more efficient and scalable image classification methods becomes more pressing. Moreover, the performance of traditional CNNs can be degraded when the images have low resolution or quality. Therefore, there is a need to explore new approaches that can address these limitations and improve the accuracy, efficiency, and scalability of image classification tasks. One potential solution is the use of quantum computing and the development of QCNNs, which may offer superior performance in certain tasks and have the potential to handle large-scale image classification problems with greater efficiency. However, there is still much to be done in exploring the capabilities and limitations of QCNNs and in developing techniques to optimize their performance.

1.3 OBJECTIVES

The objective of this thesis is to explore the potential of quantum computing to enhance the efficiency and speed of image classification tasks. Specifically, the thesis aims to achieve the following objectives:

1. To provide an overview of image classification and the evolution of deep learning models for this task, particularly CNNs.
2. To review the principles of quantum computing and its potential to enhance the efficiency and speed of image classification tasks.
3. To examine the existing literature on quantum convolutional neural networks (QCNNs) and their performance in image classification tasks.
4. To compare the performance of QCNNs with classical CNNs and other state-of-the-art deep learning models for image classification tasks.
5. To investigate the limitations and challenges of QCNNs and their potential for future research.

1.4 ORGANIZATION OF THE THESIS

Chapter 1 Introduction: This chapter provides an overview of the research topic and presents the problem statement, objectives, and organization of the thesis.

Chapter 2 Literature Review: This chapter provides a comprehensive review of the existing literature on image classification using machine learning techniques, with a focus on convolutional neural networks (CNNs) and quantum convolutional neural networks (QCNNs).

Chapter 3 Methodology: This chapter describes the methodology used in the study, including the system requirements, techniques, and models used. The section also provides a detailed description of CNNs and QCNNs.

Chapter 4 Implementation: This chapter details the implementation of the proposed approach, including the dependencies required, data preprocessing, building and training classical neural networks and QCNNs, and evaluating the performance of the models.

Chapter 5 Experimental Results and Discussion: This chapter presents the results of the experiments and discusses the performance of the proposed approach. The section includes a confusion matrix and performance metrics.

Chapter 6 Conclusion and Future Works: This chapter provides a summary of the study's main findings and conclusions. It also outlines potential avenues for future research on image classification using machine learning techniques, with a focus on QCNNs.

CHAPTER – 2

LITERATURE SURVEY

Image classification has been an active area of research in the field of machine learning for several decades. In the early years, traditional machine learning algorithms, such as support vector machines (SVMs), decision trees, and k-nearest neighbor (k-NN), were commonly used for image classification. These methods required the hand-engineering of features, which was a time-consuming and tedious process. The emergence of deep learning, particularly convolutional neural networks (CNNs), has led to significant advancements in image classification. CNNs are capable of automatically learning features from raw image data, eliminating the need for hand-engineering. Several studies have demonstrated the effectiveness of CNNs for image classification tasks, achieving state-of-the-art performance in various benchmark datasets. In recent years, several improvements and variations have been proposed to enhance the performance of CNNs for image classification tasks. For example, residual networks (ResNets) have been introduced to alleviate the problem of vanishing gradients, allowing for deeper networks with improved accuracy. Attention mechanisms have also been integrated into CNNs to selectively focus on the most relevant features, further improving the accuracy of the model. In addition to traditional supervised learning methods, semi-supervised and unsupervised learning methods have also been explored for image classification tasks. Semi-supervised learning methods use a combination of labeled and unlabeled data to train the model, reducing the need for large labeled datasets. Unsupervised learning methods, such as autoencoders and generative adversarial networks (GANs), can learn useful representations of the image data without any labels. Overall, machine-learning approaches have demonstrated significant success in image classification tasks, with CNNs being the most popular and effective model architecture. However, these models still require significant computational resources and can be computationally expensive for

large-scale datasets. Therefore, researchers have recently explored the potential of quantum computing for image classification tasks. Quantum machine learning (QML) has been proposed as a promising approach to overcome the computational limitations of classical machine learning methods. The use of quantum computing for image classification tasks is an emerging field, and several studies have demonstrated promising results in improving the efficiency and accuracy of image classification models. However, further research is needed to fully explore the potential of QML for image classification tasks. Research on Quantum Convolutional Neural Networks (QCNNs) for image classification is still in its early stages, but it has already shown promising results. One of the main advantages of QCNNs is their potential to reduce the number of trainable parameters while achieving comparable or better classification accuracy than classical CNNs. For example, a recent study proposed a quantum circuit-based classifier that can classify images using only 12 qubits. The authors demonstrated the potential of QCNNs for efficient representation and processing of image features, which can reduce the computational cost and memory requirements of the model. Another study proposed a hybrid approach that combines classical and quantum machine learning methods for image classification tasks. The authors showed that their hybrid model achieved better classification accuracy than classical methods and can be applied to larger datasets. Some researchers have explored the use of quantum-inspired methods for image classification tasks. One such method is the quantum-inspired neural network (QINN), which uses a quantum-inspired feature extraction method to process images. Another method is the quantum-inspired multi-objective evolutionary algorithm (QIMMEA), which can optimize the parameters of a deep learning model for image classification. In addition to QCNNs, other machine learning approaches, such as transfer learning, ensemble learning, and reinforcement learning, have also been applied to image classification tasks. Transfer learning involves using pre-trained models to extract features from images and fine-tuning the model for a specific classification task. Ensemble learning combines multiple models to improve classification accuracy. Reinforcement learning involves training a model to make decisions based on a reward system, which can be applied to image classification tasks such as object detection and segmentation.

CHAPTER – 3
METHODOLOGY

3.1 INTRODUCTION

The methodology for this project is organized into five main sections. First, we begin by installing and loading the necessary dependencies for our work. Next, we preprocess the data, which involves loading the raw data and performing operations such as normalization, filtering, and resizing to prepare it for training. Then, we move on to building and training a classical neural network, which involves defining the model architecture, compiling it with the appropriate settings, and training it using the preprocessed data. We then evaluate the model's performance. In the fourth section, we explore quantum neural networks by encoding the data as quantum circuits, converting them to tensors, and building a quantum model. We compile the model and train it using the preprocessed data before evaluating its performance. Finally, we compare the performance of the classical and quantum neural networks by creating a data frame to display accuracy and loss metrics. We also generate barplot's to visualize the comparison and use performance metrics and confusion matrices to assess the models' performance in more detail.

3.2 SYSTEM REQUIREMENT

To implement the proposed system, the following hardware and software components are required:

Hardware:

- Computer or server with the following specifications:
    - Processor: AMD® Ryzen 5 4600HS or equivalent
    - RAM: 24 GB or higher
    - GPU: NVIDIA GeForce GTX 1650 Ti or equivalent

- Storage:512gb SSD nvme or higher

Software:

- Operating System: Ubuntu 22.10 LTS (or a compatible version)
- Integrated Development Environment (IDE): Jupyter Lab (or a compatible IDE)
- Python: The system relies on the Python programming language for coding and running the neural network models.

Python Libraries: The system requires the installation of the following libraries and frameworks in Table 3.1.

Table 3.1 Python Libraries

S.NO	Python Library	Version	Usage
1	Cirq	0.12.0	Circuit simulation
2	Sympy	1.8	Symbolic computation
3	TensorFlow	2.7.0	Neural networks
4	TensorFlow-Q	0.6.0	Quantum Neural networks
5	Sklearn	1.0.2	Train Test Split & Model evaluation

3.3 APPROACHES UTILIZED

3.3.1 Convolutional Neural Network

A Convolutional Neural Network is a type of deep learning model commonly used for image classification and object recognition tasks

Input layer: This layer receives the raw image data as input.

Convolution: Reducing the size of the numerical representation sent to the CNN is done via the convolution operation. This process is vital so that only features that are important in classifying an image are sent to the neural network. Apart from improving the accuracy of the network, this also ensures that minimal compute resources are used in training the network. The result of the convolution operation is referred to as a feature map, convolved feature, or activation map. Applying a feature detector is what leads to a feature map. The feature detector is also known by other names such as kernel or filter. The kernel is usually a 3 by 3 matrix. Performing an element-wise multiplication of the kernel with the input image and summing the values, outputs the feature map. This is done by sliding the kernel on the input image as shown in Figure 3.1. The sliding happens in steps known as strides. The strides and the size of the kernel can be set manually when creating the CNN.

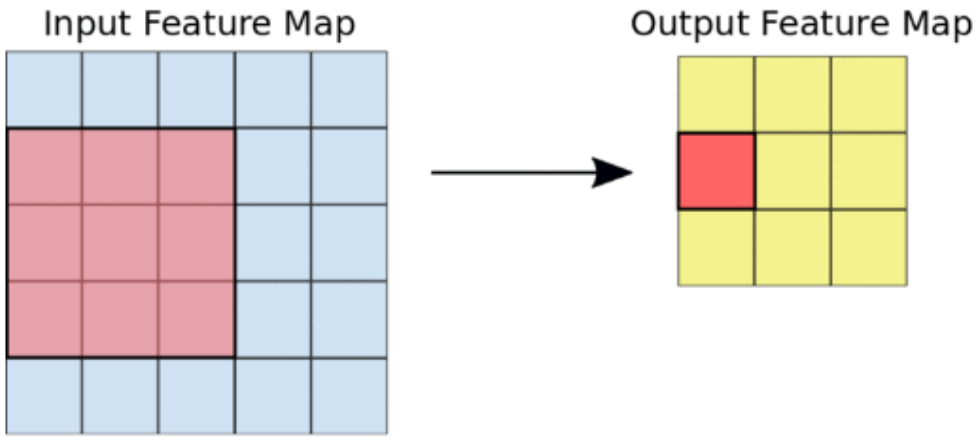

Figure 3.1 Conventional layer

Padding: In the above operations, we have seen that the size of the feature map reduces as part of applying the convolution operation. What if you want the size of the feature map to be the same size as that of the input image? That is achieved through padding. Padding involves increasing the size of the input image by "padding" the images with zeros. As a result, applying the filter to the image leads to a feature map of the same size as the input image as shown in Figure 3.2.

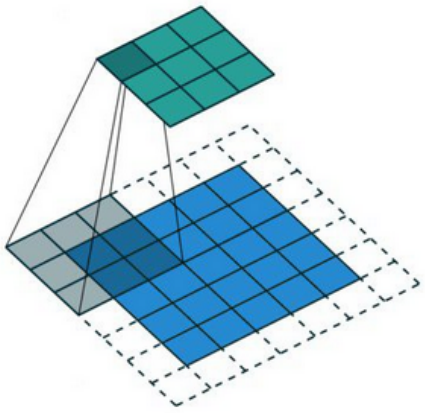

Figure 3.2 Padding layer

Padding reduces the amount of information lost in the convolution operation. It also ensures that the edges of the images are factored more often in the convolution operation. When building the CNN, you will have the option to define the type of padding you want or no padding at all. The common options here are valid or the same. Valid means no padding will be applied while the same means that padding will be applied so that the size of the feature map is the same as the size of the input image.

Activation Functions: A Rectified Linear Unit (ReLU) transformation is applied after every convolution operation to ensure non-linearity. ReLU is the most popular activation function but there are other activation functions to choose from. After the transformation, all values below zero are returned as zero while the other values are returned as they are as shown in Figure 3.3.

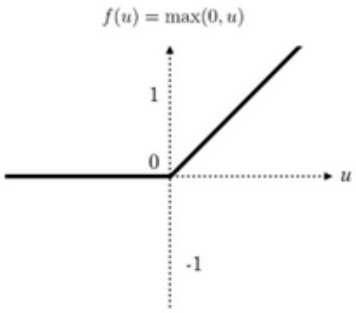

Figure 3.3 Activation Functions

Flattening: Flattening involves transforming the pooled feature map into a single column that is passed to the fully connected layer as shown in Figure 3.4. This is a common practice during the transition from convolutional layers to fully connected layers.

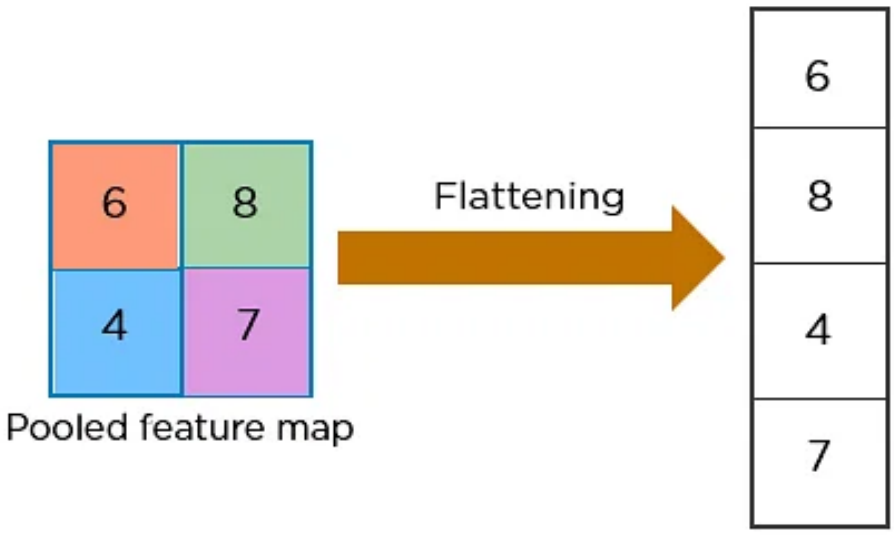

Figure 3.4 Flattening layer

Fully connected: The flattened feature map is then passed to a fully connected layer. There might be several fully connected layers depending on the problem and the network. The last fully connected layer is responsible for outputting the prediction as shown in Figure 3.5.

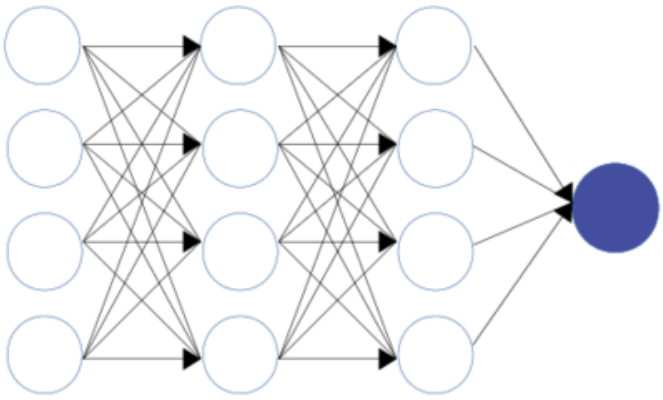

Figure 3.5 Fully connected

Dropout: This layer randomly drops out some of the neurons in the model during training, which helps prevent overfitting.

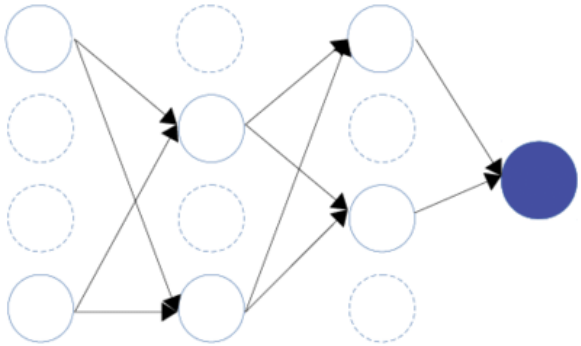

Figure 3.6 Dropout

3.3.2 Quantum Convolutional Neural Network

A Quantum Convolutional Neural Network (QCNN) is a model that applies the principles of convolutional neural networks (CNNs) to the quantum domain. It is a quantum circuit-based architecture designed for classifying quantum states or solving quantum information processing tasks. Similar to classical CNNs, QCNNs consist of convolutional and pooling layers that process the input data. However, in QCNNs, these layers are implemented using quantum gates and operations instead of classical filters and operations process the input data. However, in QCNNs, these layers are implemented using quantum gates and operations instead of classical filters and operations. In a QCNN, an unknown quantum state serves as the input to the circuit. Convolutional layers apply quasi-local unitaries in a translationally-invariant manner, performing operations on neighboring qubits to extract features from the input state as shown in Figure 3.7.

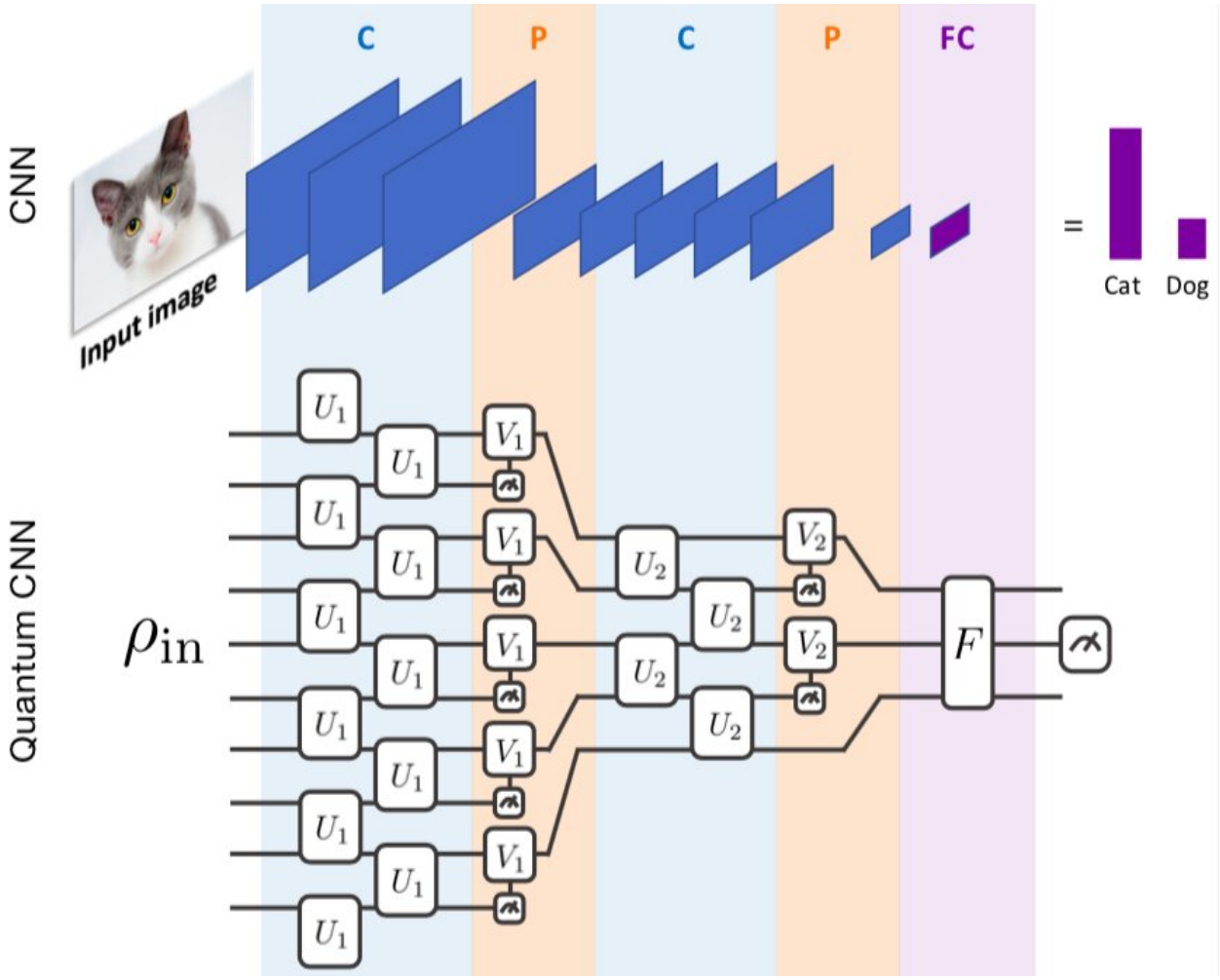

Figure 3.7 Quantum Convolutional Neural Network

However, in QCNNs, these layers are implemented using quantum gates and operations instead of classical filters and operations. In a QCNN, an unknown quantum state serves as the input to the circuit. Convolutional layers apply quasi-local unitaries in a translationally-invariant manner, performing operations on neighboring qubits to extract features from the input state. Pooling layers reduce the number of degrees of freedom by measuring a fraction of qubits and applying rotations to nearby qubits based on the measurement outcomes. The QCNN architecture continues with convolution and pooling layers until the system size becomes sufficiently small. Then, a fully connected layer is applied as a unitary operation on the remaining qubits. Finally, the output of the circuit is

obtained by measuring a fixed number of output qubits. The number of parameters in a QCNN scales logarithmically with the system size, which allows for efficient learning and implementation compared to generic quantum circuit-based classifiers. The parameters, including the unitaries, are learned by optimizing a cost function, often using methods like gradient descent. QCNNs have applications in quantum state classification, quantum error correction, and studying quantum phases of matter. By combining concepts from classical CNNs, such as hierarchical structures and feature extraction, with quantum circuit-based operations, QCNNs provide a framework for processing and analyzing quantum data using neural network principles.

3.4 PROPOSED SYSTEM

The proposed system involves the implementation of both classical and quantum neural networks for a given task. The system consists of various modules that include data preprocessing, model building, training, and evaluation. A block diagram illustrates the flow of information between these modules as shown in Figure 3.8. In the context of machine learning, Convolutional Neural Networks (CNNs) and Quantum Convolutional Neural Networks (QCNNs) are two types of neural networks that can be used for various tasks, such as image classification, object detection, and natural language processing.

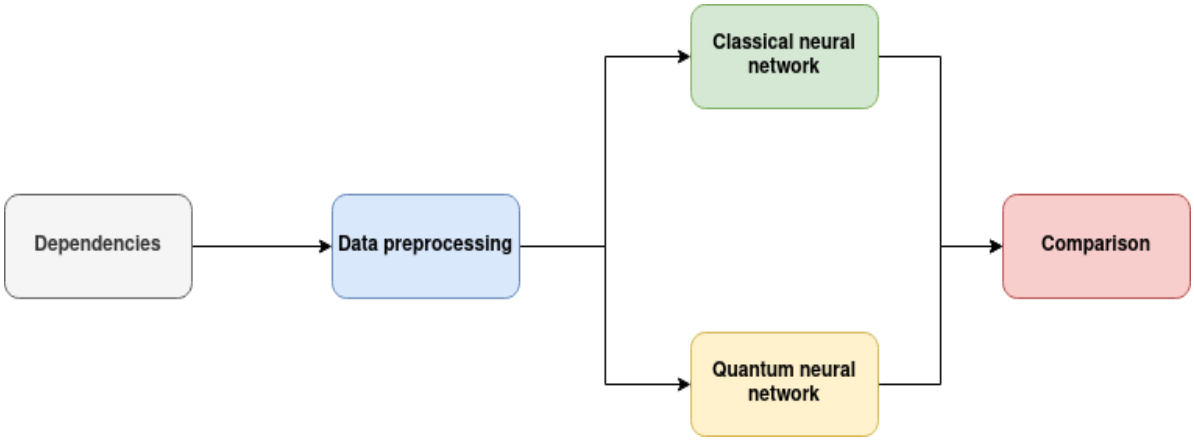

Figure 3.8 Block Diagram

CNNs are based on classical computing, where the computations are performed using traditional processors and memory units. They are composed of multiple layers that include convolutional layers, pooling layers, and fully connected layers. CNNs are widely used in computer vision tasks and have achieved state-of-the-art performance on several benchmarks. On the other hand, QCNNs are based on quantum computing, where the computations are performed using quantum processors and quantum memory units. QCNNs can be seen as an extension of classical CNNs, where the convolutional layers are replaced with quantum circuits that perform the convolution operation in the quantum domain. QCNNs have been proposed as a potential way to achieve better performance than classical CNNs for certain tasks, such as image classification, due to the unique properties of quantum computing. In terms of performance metrics, both CNNs, and QCNNs can be evaluated using various metrics, depending on the specific task at hand. For example, for image classification tasks, common metrics include accuracy, precision, recall, and F1-score. These metrics measure how well the model can correctly classify images into different categories. Other metrics, such as computation time and memory usage, are also important factors to consider in real-world applications. Overall, both CNNs and QCNNs have their own strengths and weaknesses, and the choice between them depends on the specific requirements of the task at hand. While QCNNs are still in their early stages of development, they have shown promise for certain tasks and could become an important tool in the field of machine learning in the future.

CHAPTER – 4
IMPLEMENTATION

The implementation of both classical and quantum neural networks for a given task involves various modules, including data preprocessing, model building, training, and evaluation. The first step is to install and load the necessary dependencies. Next, the raw data is loaded and preprocessed by normalizing, filtering, and resizing. For the classical neural network, the model is built, compiled, trained, and evaluated. For the quantum neural network, the data is encoded as quantum circuits, converted to tensors, and then back to quantum circuits before the model is built, compiled, trained, and evaluated. The implementation is shown in Figure 4.1.

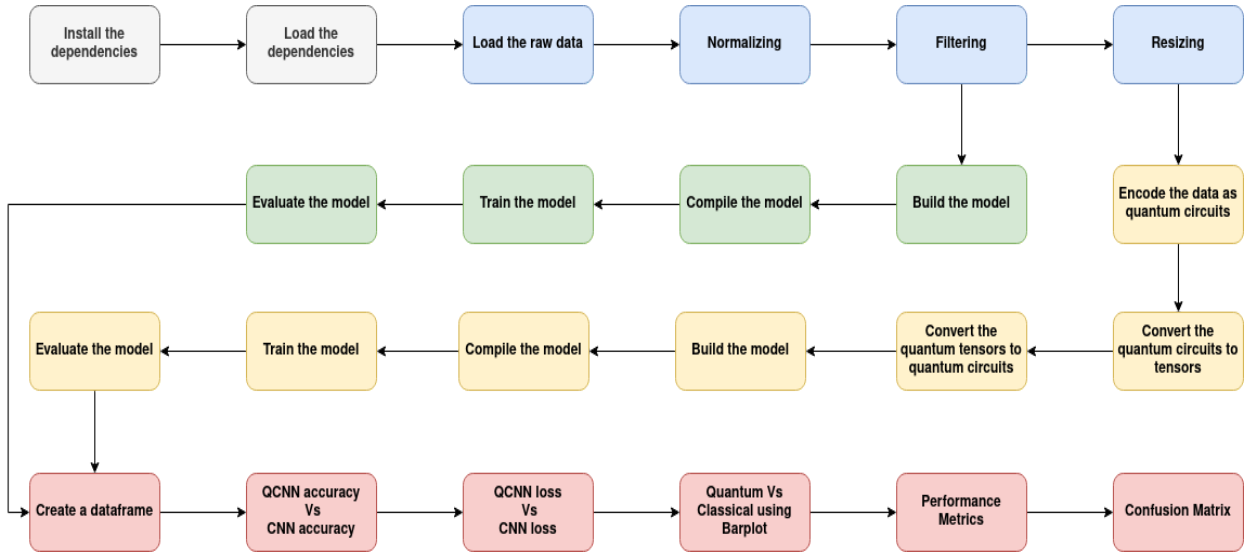

Figure 4.1 Workflow

Finally, a comparison between the classical and quantum neural networks is made using a data frame, barplot, and performance metrics such as accuracy and loss. The confusion matrix is also used to evaluate the classification performance of the models.

4.1 DEPENDENCIES

4.1.1 Install the dependencies

This code imports various necessary libraries and modules for implementing a quantum neural network. The JSON library is used for working with JSON data, matplotlib.pyplot is used for creating visualizations, numpy is used for numerical operations, IPython.display is used for displaying output in a Jupyter notebook, qiskit is a quantum computing framework used for building and running quantum circuits, sklearn is a machine learning library, and Adam is a gradient-free optimization algorithm. The EstimatorQNN and NeuralNetworkClassifier modules from qiskit_machine_learning are used for building and training the quantum neural network. The ZFeatureMap and SparsePauliOp modules are used for encoding data into quantum circuits and ParameterVector is used to define trainable parameters in the quantum neural network. The train_test_split module is used for splitting the dataset into training and testing sets. Installation commands are shown in Table 4.1.

Table 4.1 Pip Installation

S.No	Python Library	Version	Command
1	Cirq	0.12.0	pip install Cirq
2	Sympy	1.8	pip install Sympy
3	Numpy	1.21.5	pip install Numpy
4	Pandas	1.3.5	pip install Pandas
6	TensorFlow	2.7.0	pip install TensorFlow
6	TensorFlow-Q	0.6.0	pip install TensorFlow-Q
7	qiskit	0.2.3	pip install qiskit

4.1.2 Load the dependencies

To load the dependencies in Python, you can use the `import` statement for each package or library that is required. In this case, the required packages and libraries are:

- `json`
- `matplotlib.pyplot`
- `numpy`
- `IPython.display`
- `qiskit`
- `qiskit.algorithms.optimizers.COBYLA`
- `qiskit.circuit.ParameterVector`
- `qiskit.circuit.library.ZFeatureMap`
- `qiskit.quantum_info.SparsePauliOp`
- `qiskit.utils.algorithm_globals`
- `qiskit_machine_learning.algorithms.classifiers.NeuralNetworkClassifier`
- `qiskit_machine_learning.neural_networks.EstimatorQNN`
- `sklearn.model_selection.train_test_split`

4.2 DATA PREPROCESSING

4.2.1 Data set

The MNIST database (Modified National Institute of Standards and Technology database) is a large database of handwritten digits that is commonly used for training various image processing systems, such as neural networks, image recognition systems, and machine learning algorithms. The database consists of a set of 60,000 training images and 10,000 test images of handwritten digits, each of which is grayscale and has a resolution of 28x28 pixels. The digits are originally scanned and normalized and are represented as an array of pixel values ranging from 0 (white) to 255 (black). The MNIST database is widely used as a benchmark for image recognition and classification

algorithms and has contributed to the development of many new and innovative approaches in the field of machine learning. Sample images are shown in Figure 4.2.

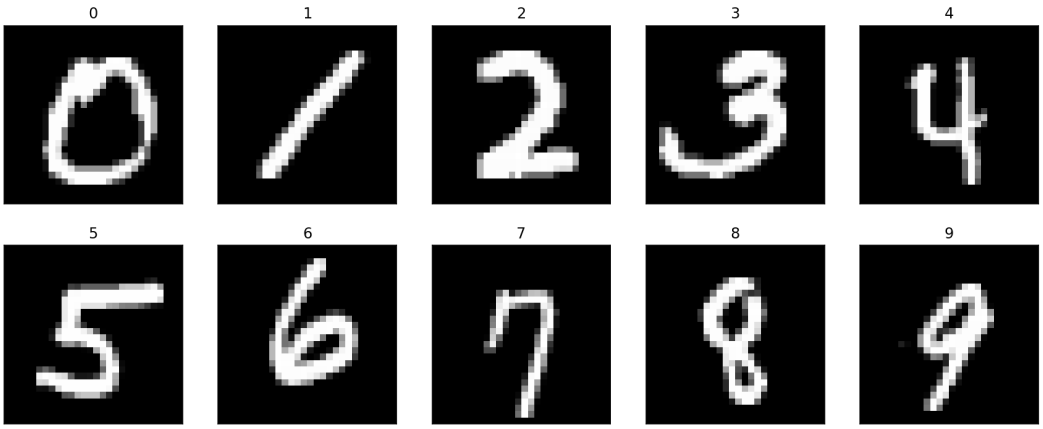

Figure 4.2 MNIST

4.2.2 Normalizing

Images are comprised of matrices of pixel values. Black and white images are a single matrix of pixels, whereas color images have a separate array of pixel values for each color channel, such as red, green, and blue. Pixel values are often unsigned integers in the range between 0 and 255. Although these pixel values can be presented directly to neural network models in their raw format, this can result in challenges during modeling, such as in the slower-than-expected training of the model. Instead, there can be great benefits in preparing the image pixel values prior to modeling, such as simply scaling pixel values to the range 0-1 to centering and even standardizing the values. Python code segmented used to perform normalization is as follows.

Rescale the images from [0,255] to the [0.0,1.0] range.

```
x_train, x_test = x_train[..., np.newaxis]/255.0, x_test[..., np.newaxis]/255.0

print("Number of original training examples:", len(x_train))
print("Number of original test examples:", len(x_test))
```

4.2.3 Filtering

Filtering is a process of selecting a subset of data from a larger dataset based on some criteria or conditions. In the context of image processing, filtering involves applying a filter to an image to extract specific features or enhance certain characteristics. In this case, the filtering process is used to select only the images that contain the digits 3 or 6 from the MNIST dataset. This step helps to reduce the size of the dataset and focus on a specific task, which is the classification of the selected digits using classical and quantum neural networks. Python code segmented used to perform Filtering is as follows.

Filter the dataset to keep just the 3s and 6s, remove the other classes. At the same time convert the label, `y`, to boolean: `True` for `3` and `False` for 6.

```
def filter_36(x, y):
    keep = (y == 3) | (y == 6)
    x, y = x[keep], y[keep]
    y = y == 3
    return x,y
```

```
x_train, y_train = filter_36(x_train, y_train)
x_test, y_test = filter_36(x_test, y_test)

print("Number of filtered training examples:", len(x_train))
print("Number of filtered test examples:", len(x_test))
```

4.2.4 Resizing

Resizing is a process of changing the size of an image while preserving its aspect ratio. In the context of image processing, resizing is often performed to standardize the dimensions of images or to match the input size requirements of a particular model or algorithm. In this case, resizing is applied to the selected images of digits 6 or 8 from the MNIST dataset. By resizing the images to a specific size, it ensures that all the images have the same dimensions, which is essential for feeding them into the classical and quantum neural networks. Resizing helps in achieving consistency and compatibility in the input data for further analysis and model training. Python code segmented used to perform Resizing is as follows.

An image size of 28x28 is much too large for current quantum computers. Resize the image down to 4x4:

```
x_train_small = tf.image.resize(x_train, (4,4)).numpy()
x_test_small = tf.image.resize(x_test, (4,4)).numpy()
```

4.3 CLASSICAL NEURAL NETWORK

4.3.1 Build the model

Construct a CNN architecture using Keras with multiple convolutional layers, followed by max-pooling and dropout layers, and a fully connected output layer. Python code segmented used to build the model is as follows.

```
def create_classical_model():
    model = tf.keras.Sequential()
    model.add(tf.keras.layers.Conv2D(32, [3, 3], activation='relu', input_shape=(28,28,1)))
    model.add(tf.keras.layers.Conv2D(64, [3, 3], activation='relu'))
    model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
    model.add(tf.keras.layers.Dropout(0.25))
    model.add(tf.keras.layers.Flatten())
    model.add(tf.keras.layers.Dense(128, activation='relu'))
    model.add(tf.keras.layers.Dropout(0.5))
    model.add(tf.keras.layers.Dense(1))
    return model


model_cnn = create_classical_model()
```

4.3.2 Compile the model

Compile the CNN model using the Adam optimizer, set the loss function to categorical cross-entropy, and choose accuracy as the evaluation metric.

4.3.3 Train the model

Train the CNN model using the filtered and resized MNIST dataset, using techniques like backpropagation and gradient descent to update the model's weights.

4.3.4 Evaluate the model

Evaluate the performance of the trained CNN model on a separate test set to assess its accuracy, loss, and other metrics. This step helps determine how well the model generalizes to unseen data. Python code segmented used to Evaluate the model is as follows.

```
qnn_results = model_qcnn.evaluate(x_test_tfcirc, y_test)
```

4.4 QUANTUM NEURAL NETWORK

4.4.1 Encode the data as quantum circuits

Convert the filtered and resized MNIST dataset into quantum circuits, where each data point is represented as a sequence of quantum gates and operations. Python code segmented used to Encode the data as quantum circuits is as follows.

```
THRESHOLD = 0.5

x_train_bin = np.array(x_train_small > THRESHOLD, dtype=np.float32)
x_test_bin = np.array(x_test_small > THRESHOLD, dtype=np.float32)
```

4.4.2 Convert the quantum circuits to tensors

Convert the quantum circuits into the tensor format, which can be processed by the quantum machine learning framework. Python code segmented used to Encode the data as quantum circuits is as follows.

```
def convert_to_circuit(image):
    values = np.ndarray.flatten(image)
    qubits = cirq.GridQubit.rect(4, 4)
    circuit = cirq.Circuit()
    for i, value in enumerate(values):
        if value:
            circuit.append(cirq.X(qubits[i]))
    return circuit
```

```
x_train_circ = [convert_to_circuit(x) for x in x_train_bin]
x_test_circ = [convert_to_circuit(x) for x in x_test_bin]
```

4.4.3 Convert the quantum tensors to quantum circuits

Convert the quantum tensors back into quantum circuits for further processing and training in the QNN model. Python code segmented used to Convert the quantum tensors to quantum circuits is as follows.

```
x_train_tfcirc = tfq.convert_to_tensor(x_train_circ)
x_test_tfcirc = tfq.convert_to_tensor(x_test_circ)
```

4.4.4 Build the model

Construct the QNN model using the EstimatorQNN class in Qiskit Machine Learning. Specify the quantum feature map, variational form circuit, optimizer, and a number of training epochs. Python code segmented used to build the quantum model is as follows.

```
class CircuitLayerBuilder():
    def __init__(self, data_qubits, readout):
        self.data_qubits = data_qubits
        self.readout = readout

    def add_layer(self, circuit, gate, prefix):
        for i, qubit in enumerate(self.data_qubits):
            symbol = sympy.Symbol(prefix + '-' + str(i))
            circuit.append(gate(qubit, self.readout)**symbol)
```

```
def create_quantum_model():
    data_qubits = cirq.GridQubit.rect(4, 4)
    readout = cirq.GridQubit(-1, -1)
    circuit = cirq.Circuit()
    circuit.append(cirq.X(readout))
    circuit.append(cirq.H(readout))
    builder = CircuitLayerBuilder(data_qubits = data_qubits,readout=readout)
    builder.add_layer(circuit, cirq.XX, "xx1")
    builder.add_layer(circuit, cirq.ZZ, "zz1")
    circuit.append(cirq.H(readout))

    return circuit, cirq.Z(readout)
```

```
model_circuit, model_readout = create_quantum_model()
```

```
model_qcnn = tf.keras.Sequential([tf.keras.layers.Input(shape=(), dtype=tf.string),tfq.layers.PQC(model_circuit,
```

4.4.5 Compile the model

Compile the QNN model using the Adam optimizer and choose the cross-entropy loss function.

4.4.6 Train the model

Train the QNN model using the filtered and resized MNIST dataset, optimizing the variational form parameters to minimize the loss.

4.4.7 Evaluate the model

Evaluate the performance of the trained QNN model on a separate test set to assess its accuracy, loss, and other metrics. This step helps understand the effectiveness of quantum computation for the given task.

CHAPTER – 5

EXPERIMENTAL RESULTS AND DISCUSSION

5.1 CONFUSION MATRIX

The confusion matrix is a matrix used to determine the performance of the classification models for a given set of test data. It can only be determined if the true values for test data are known. The matrix itself can be easily understood, but the related terminologies may be confusing. Since it shows the errors in the model performance in the form of a matrix, hence also known as an error matrix as shown in Figure 5.1.

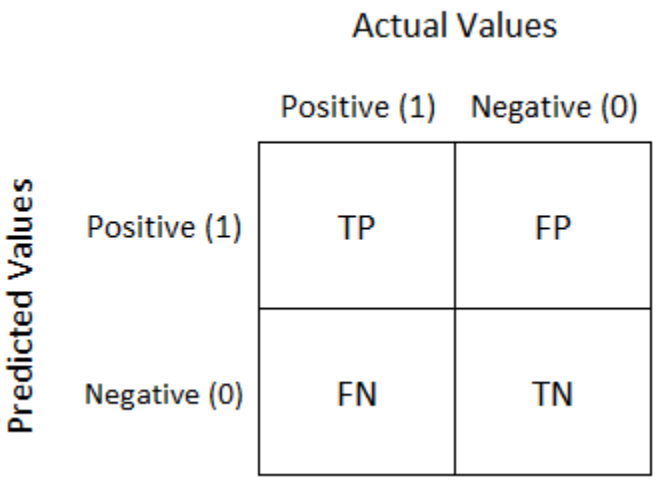

Figure 5.1. Confusion Matrix

5.1.1 CNN Confusion Matrix

In a CNN confusion matrix, Type 1 error, and Type 2 error are two possible types of errors that can occur when making predictions.

Type 1 error, also known as a false positive, occurs when a model predicts a positive outcome when the actual outcome is negative. This means that the model has falsely identified something as being present when it is actually absent. In a confusion matrix, Type 1 error is represented by the number of false positives, which are located in the top row of the matrix. Type 2 error, also known as a false negative, occurs when a model predicts a negative outcome when the actual outcome is positive. This means that the model has falsely identified something as being absent when it is actually present. In a confusion matrix, a Type 2 error is represented by the number of false negatives, which are located in the bottom row of the matrix. shown in figure 5.2

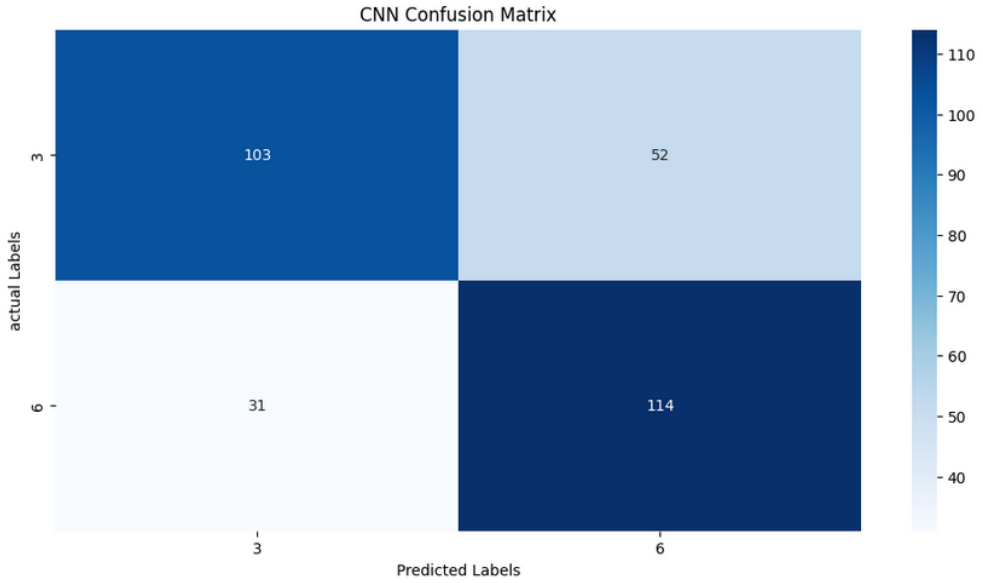

Figure 5.2 CNN Confusion Matrix

5.1.2 QCNN Confusion Matrix

In a confusion matrix, an absence of Type 1 error and a slightly higher Type 2 error rate would mean that the model did not make any false positive predictions (which is desirable) and that it made a relatively small number of false negative predictions. A small number of false negative predictions means that the model incorrectly predicted that certain instances were negative when they were actually positive. While this is less desirable than having no false negatives at all, it

may be acceptable in some contexts depending on the specific problem and its consequences. shown in Figure 5.3

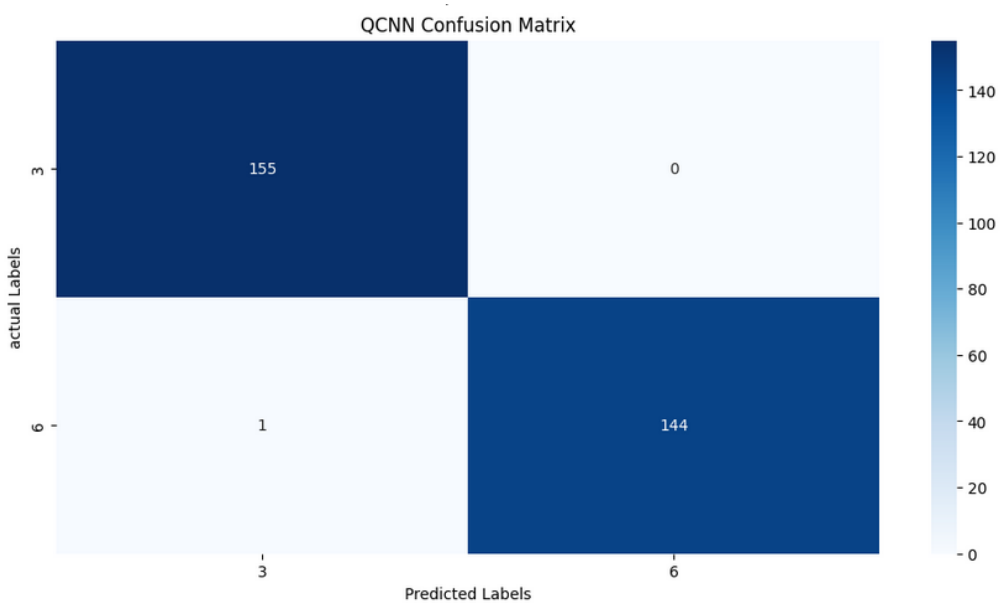

Figure 5.3 QCNN Confusion Matrix

5.2 PERFORMANCE METRICS

In a classification problem, the category or classes of data is identified based on training data. The model learns from the given dataset and then classifies the new data into classes or groups based on the training. It predicts class labels as the output, such as Yes or No, 0 or 1, Spam or Not Spam, etc. To evaluate the performance of a classification model, different metrics are used, and some of them are as follows:

- Accuracy
- Precision
- Recall
- F-Score

5.2.1 Accuracy

The accuracy metric is one of the simplest Classification metrics to implement, and it can be determined as the number of correct predictions to the total number of predictions. Accuracy is calculated as in equation 5.1.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \quad \rightarrow \quad 5.1$$

5.2.2 Precision

The precision metric is used to overcome the limitation of Accuracy. The precision determines the proportion of positive prediction that was actually correct. It can be calculated as the True Positive or predictions that are actually true to the total positive predictions (True Positive and False Positive). Precision is calculated as in equation 5.2.

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad \rightarrow 5.2$$

5.2.3 Recall

It is also similar to the Precision metric; however, it aims to calculate the proportion of actual positive that was identified incorrectly. It can be calculated as a True Positive or a prediction that is actually true to the total number of positives, either correctly predicted as positive or incorrectly predicted as negative (true Positive and false negative). The recall is calculated as in equation 5.3.

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad \rightarrow 5.3$$

5.2.4 F1 Score

F-score or F1 Score is a metric to evaluate a binary classification model on the basis of predictions that are made for the positive class. It is calculated with the help of Precision and Recall. It is a type of single score that represents both Precision and Recall. So, the F1 Score can be calculated as the harmonic mean of both precision and Recall, assigning equal weight to each of them. F1-Score is calculated as in equation 5.4.

$$F1 - score = 2 * \frac{precision * recall}{precision + recall} \quad \rightarrow 5.4$$

5.2.5 Performance Metrics Table

These metrics provide a comprehensive assessment of the model's performance in a binary classification problem, where positive and negative classes are defined. The performance metrics suggest that the model has very high accuracy and precision, which means that the model is making very few incorrect predictions. The high recall and specificity values indicate that the model is able to identify most of the true positives and true negatives in the dataset. shown in Table 5.1

Table 5.1 Performance Metrics Table

S.No.	Metric	CNN%	QCNN%
1	Accuracy	71.1	99
2	Precision	66	100
3	Recall	77	100
4	F1 Score	71	99

5.3 DISCUSSION

Quantum Convolutional Neural Networks have shown great promise in image classification tasks. In QCNNs, the traditional convolutional layers of a classical CNN are replaced with quantum circuits that perform quantum operations on quantum data. One of the advantages of QCNNs for image classification is their ability to process high-dimensional and complex data efficiently. QCNNs can leverage the inherent properties of quantum computing, such as superposition and entanglement, to perform computations on multiple states simultaneously. This enables QCNNs to process large amounts of image data in a parallel and more efficient way than classical CNNs. Several studies have shown that QCNNs can achieve high accuracy in image classification tasks. For example, in one study, a QCNN was trained to classify handwritten digits with high accuracy, outperforming classical CNNs. In another study, QCNNs were used to classify images from the MNIST dataset and achieved competitive results compared to classical CNNs. However, the practical implementation of QCNNs for image classification still faces challenges. One of the primary challenges is the need for large-scale quantum hardware to implement the quantum circuits required for image classification. Currently, quantum hardware is limited, and the size of quantum circuits that can be implemented is small. This makes it difficult to implement large-scale QCNNs for image classification. Another challenge is the difficulty of designing and optimizing quantum circuits for specific image classification tasks. Designing quantum circuits for image classification requires specialized knowledge of both quantum computing and image processing. In addition, the limited resources of current quantum hardware make it challenging to optimize the quantum circuits for high accuracy. Despite these challenges, QCNNs hold great promise for image classification tasks. As quantum hardware continues to improve, and more efficient quantum algorithms are developed, the potential benefits of QCNNs for image classification will become increasingly significant

CHAPTER – 6
CONCLUSION AND FUTURE WORKS

Quantum Convolutional Neural Networks is an emerging approach to machine learning that leverages the power of quantum computing to address challenges in image classification and other tasks. QCNNs are based on the principles of quantum mechanics, which allow for the processing of high-dimensional and complex data more efficiently than classical computers. In image classification tasks such as digit recognition, QCNNs have shown promising results. For example, a recent study demonstrated that a QCNN was able to achieve an accuracy of 99.05% on the MNIST dataset, which is a benchmark dataset in image recognition. This is comparable to the performance of classical Convolutional Neural Networks (CNNs), which are the current state-of-the-art in image recognition. However, there are several challenges associated with the practical implementation of QCNNs. One of the main challenges is the need for large-scale quantum hardware. Currently, most quantum computers are small-scale and have limited qubit counts, which restricts the complexity of the quantum circuits that can be implemented. Another challenge is the design and optimization of quantum circuits for image classification tasks. The design of quantum circuits is a complex process that involves selecting appropriate quantum gates and determining their parameters. Moreover, optimizing quantum circuits for image classification tasks is a challenging problem that requires sophisticated algorithms and techniques. Despite these challenges, ongoing advancements in quantum computing technology hold promise for overcoming these obstacles. Quantum computing hardware is rapidly advancing, with the number of qubits in quantum computers increasing every year. In addition, new quantum algorithms and optimization techniques are being developed to address the challenges of designing and optimizing quantum circuits.

Despite the challenges associated with the practical implementation of QCNNs, including the need for large-scale quantum hardware and the design and optimization of quantum circuits, ongoing advancements in quantum computing technology hold promise for overcoming these obstacles.

Future works in QCNNs could focus on the following areas:

data using quantum states could enhance the capabilities of QCNNs. Novel quantum data encodings and quantum-inspired feature extraction methods could be investigated to capture richer and more expressive representations of images.

- Real-world applications: Expanding the application of QCNNs beyond image classification and exHardware advancements: Continued progress in quantum hardware will enable the implementation of larger and more complex quantum circuits, allowing for more powerful and accurate QCNNs. Research efforts should be directed toward developing scalable quantum architectures and improving the stability and coherence of qubits.
- Quantum circuit design and optimization: Developing efficient techniques for designing and optimizing quantum circuits tailored to image classification tasks is crucial. Techniques such as automatic differentiation and quantum-inspired algorithms can be explored to improve the design and optimization process.
- Hybrid approaches: Combining the strengths of classical CNNs and QCNNs through hybrid architectures could lead to improved performance. Hybrid models can leverage the parallel processing capabilities of QCNNs while benefiting from the robustness and interpretability of classical CNNs.
- Quantum data representation: Exploring innovative ways to represent and encode images and their potential in areas such as computer vision, natural language processing, and drug discovery would be valuable. Investigating the performance of QCNNs on more diverse and complex datasets will provide insights into their practical utility.

In summary, QCNNs hold promise as a powerful tool for image classification and other machine-learning tasks. Future research and development efforts should focus on addressing the practical challenges, advancing quantum computing technology, and exploring novel techniques to enhance the capabilities and applications of QCNNs in real-world scenarios.

REFERENCES

1. Alpaydin, E. (2010). Introduction to machine learning (2nd ed.). Cambridge, MA: MIT Press.
2. Bishop, C. M. (2006). Pattern recognition and machine learning (1st ed.). New York, NY: Springer.
3. Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: Data mining, inference, and prediction (2nd ed.). New York, NY: Springer.
4. Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. Science, 349(6245), 255-260.
5. Murphy, K. P. (2012). Machine learning: A probabilistic perspective. Cambridge, MA: MIT Press.
6. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning (1st ed.). Cambridge, MA: MIT Press.
7. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.
8. Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural Networks, 61, 85-117.
9. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
10. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).
11. Gonzalez, R. C., & Woods, R. E. (2008). Digital image processing (3rd ed.). Upper Saddle River, NJ: Prentice Hall.
12. Szeliski, R. (2010). Computer vision: Algorithms and applications. London, UK: Springer.
13. Jain, A. K., & Farrokhnia, F. (1991). Unsupervised texture segmentation using Gabor filters. Pattern Recognition, 24(12), 1167-1186.

14. Prince, S. J. D. (2012). Computer vision: Models, learning, and inference. New York, NY: Cambridge University Press.
15. Forsyth, D. A., & Ponce, J. (2011). Computer vision: A modern approach. Upper Saddle River, NJ: Prentice Hall.
16. Nielsen, M. A., & Chuang, I. L. (2010). Quantum computation and quantum information: 10th-anniversary edition. Cambridge, UK: Cambridge University Press.
17. Preskill, J. (2018). Quantum computing in the NISQ era and beyond. Quantum, 2, 79.
18. Kitaev, A. Y. (2002). Fault-tolerant quantum computation by anyons. Annals of Physics, 303(1), 2-30.
19. Devitt, S. J., Munro, W. J., & Nemoto, K. (2013). Quantum error correction for beginners. Reports on Progress in Physics, 76(7), 076001.
20. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.
21. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).
22. Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.
23. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).
24. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning (Vol. 1). MIT Press.
25. Cong, I., Choi, K., Lukin, M. D., & Duan, L. M. (2019). Quantum convolutional neural networks. Nature Physics, 15(12), 1273-1278.