QUANTUM NEURAL NETWORK ARCHITECTURE TO PERFORM MACHINE LEARNING TASKS ON NISQ COMPUTERS

by

RITU DHAULAKHANDI

SUPERVISOR: PROF. P. K. PANIGRAHI
EXPERT: PROF. M. S. SANTHANAM

[Master's Thesis]

[Physics Department]

IISER PUNE

INDIAN INSTITUTES OF SCIENCE EDUCATION AND RESEARCH, PUNE

[June, 2022 — March,2023]

CERTIFICATE

This is to certify that this dissertation entitled **"Quantum Neural Network architecture to perform Machine Learning tasks on NISQ computers"** towards the partial fulfillment of the BS-MS dual degree program at the Indian Institute of Science Education and Research, Pune, represents study/work carried out by **Ritu Dhaulakhandi** at **Indian Institute of Science Education and Research, Kolkata** under the supervision of Prof. P. K. Panigrahi at the physics department of **Indian Institute of Science Education and Research, Kolkata** during the academic year of 2022-23.

Signature of [illegible]

(**Supervisor**: Prof. P. K. Panigrahi)

(IISER, Kolkata)

DECLARATION

I hereby declare that the matter embodied in the report entitled **"Quantum Neural Network architecture to perform Machine Learning tasks on NISQ computers"** are the results of the work carried out by me at the Department of Physics, **Indian Institute of Science Education and Research, Kolkata**, under the supervision **Prof. P. K. Panigrahi**, and the same has not been submitted elsewhere for any other degree.

Ritu

Signature of ____________________________

(**Student**: Ritu Dhaulakhandi)

(IISER, Pune)

ACKNOWLEDGEMENTS

I want to thank my parents and brother for being there for me through thick and thin over the years, for their companionship, support, and care, and for being the ones without whom this project would not have been possible. In addition, I want to express my gratitude to my grandmother for her encouragement over the years.

I also want to thank Prof. P. K. Panigrahi and Prof. M. S. Santhanam, who oversaw my dissertation, for their wisdom, encouragement, and knowledge-sharing that helped make this thesis feasible.

Last but not least, I want to thank my closest friends, Riya and Raikhik, for supporting me through good and terrible times.

Thank you.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

QUANTUM NEURAL NETWORK ARCHITECTURE TO PERFORM MACHINE LEARNING TASKS ON NISQ COMPUTERS

Abstract

by

RITU DHAULAKHANDI

SUPERVISOR: PROF. P. K. PANIGRAHI

EXPERT: PROF. M. S. SANTHANAM

Models known as quantum neural networks (QNNs) combine the benefits of quantum theory with those of neural networks. They are said to be more proficient in reaching the desired performance than classical models. But it is challenging to demonstrate this importance using a practically relevant problem on the current Noisy Intermediate Scale Quantum (NISQ) computers. This thesis aims to develop a QNN architecture that can be implemented on a real quantum device without significant changes in results due to errors. The limitations imposed on implementing QNNs are due to errors in NISQ computers that suffer from decoherence, gate errors, measurement errors, and cross-talk. To reduce the error introduced due to the high accumulation of gate errors and decoherence, a combination of small QNNs, Hierarchical bi-linear classification structure, and Clustering is used.

The architecture is tested for low-dimensional datasets (Iris flower species and Ripley's crab datasets) with results provided. An additional classification result for a high-dimensional dataset is provided. A standard QNN consists of an encoding circuit, parameterized quantum circuit (PQC), measurement operations, and cost function. The encoding circuit maps the classical data represented as a column vector to a quantum state. At the same time, PQC transforms the quantum state (or updates the position of the quantum state on the Bloch sphere) to obtain desired output state. The cost function is the fidelity between the QNN result and desired output subtracted from one. The parameters of PQC are updated repeatedly until the cost function is minimized, hence completing the learning process.

The quantum state obtained from the QNN circuit is reconstructed by obtaining measurement results. The statistical distance data obtained from the final measurement is used to classify the data points into one of the main clusters. The sub-clusters present in these main clusters are identified with the help of more detailed analysis and QNN unit training. The final number of QNN units required to implement the classification problem depends on the dataset's selected features and target labels. Three IBM cloud devices are used to check the performance of the QNN units against the simulator results. The variation in the results due to QV and CLOPS is also observed. Small QNN units might be a desirable alternative for applications based on realistic QML simulations, but more study is required to boost performance. This is a modest step towards a noise-resilient quantum machine learning framework.

Chapter 1

INTRODUCTION

It is essential to understand machine learning before discussing quantum machine learning (quantum neural networks in particular). Therefore, let us first understand why machine learning came into use, how it works, and why it is important. For that, we will have to briefly examine the history of artificial intelligence.

The idea of artificial intelligence (AI) [1] dates back to when philosophers contemplated the existence of artificially intelligent beings in the future. The invention of the Atanasoff Berry Computer (ABC) [2], a programmable digital computer, in the 1940s further inspired scientists to move forward with the idea of an artificially intelligent being. With each decade came new technological innovations and outcomes, shaping AI's fundamental knowledge in people. AI got developed to create computer models that exhibit "intelligent behaviors" like humans. Machine learning (ML) [3], a sub-field of AI, was part of the evolution of AI until it branched off in the late 1970s. While AI allows computers to learn from experience, ML works more like an optimization process to help AI learn from the experience faster and better with minimum human intervention.

The economic feasibility and efficiency of ML systems have made a tremendous impact on society and business. Depending on the industry's requirements, an ML system can use data to explain what happened, predict what will happen, or suggest what actions to take. Therefore, ML is suitable for gaining insight or automating decision-making for big data situations. Furthermore, with quantum computers' [4] current development, researchers are now looking at new opportunities that open up by combining machine learning and quantum computation [5]. In the future, ML and quantum computing are expected to be critical aspects of how information gets dealt with in society. Therefore, it is only natural to question how they could be combined. Quantum machine learning (QML) [6], an emerging discipline, explores the answers to those questions.

Quantum neural network (QNN) [7] is part of the QML field in which quantum computing techniques are used to train neural networks. The theory of quantum mind, which explains the role of quantum effect in cognitive function, gave rise to the idea of the quantum neural network [8, 9]. Research in QNN mainly involves incorporating the advantages of quantum information in the classical neural network to develop more efficient algorithms. The primary motivation to investigate QNN is the quantum advantage (quantum parallelism, interference, and entanglement) it provides to overcome the difficulty of training CNNs for datasets with complex patterns or "big data" [10]. Due to the lack of fault-tolerant quantum systems and robust error correction techniques, implementing a large QNN is still premature. The following sections explain the complete thesis problem, the project's scope, the inspiration behind working on the problem, the goals, and the previous works done in QNN.

1.1 Motivation and Goals

Note: The terms and notations used here will be addressed in more detail in Chapter 2.

A neural network is an abstraction of the biological nervous system in the form of a mathematical model. Neural networks mostly follow the McCulloch-Pitts neuron model, where a nerve cell is in an active or resting state based on cell activity [11]. The basic idea behind Quantum Neural Networks (QNNs) is constructing a quantum neuron as a two-level quantum system analogous to the two states of the neural networks. The importance of QNNs arises because they are more proficient in reaching the desired performance (e.g., classification of huge data samples) compared to the classical models of a similar scale (e.g., with the same number of tunable parameters) [12, 13]. However, demonstrating this importance using a practically relevant problem on the current NISQ computers is challenging as the results will get affected due to the accumulation of errors. Until universal fault-tolerant quantum computers are developed, researchers are coming up with different approaches and architectures for QNN implementation on NISQ devices to see if there is an advantage to using QNNs over classical neural networks. Drawing inspiration from it, the thesis aims to develop a QNN architecture that can be implemented on a real quantum device without significant changes in results due to errors.

The limitations imposed on implementing QNNs in Noisy Intermediate Scale Quantum (NISQ) computers are due to decoherence, gate errors, measurement errors, and cross-talk [14]. We aim to reduce the error introduced due to the high accumulation of gate errors and decoherence. Hence, obtain high accuracy results from QNN implementation on NISQ systems for machine learning tasks. The errors are reduced with the help of a combination of small Quantum Neural Networks (QNNs), Hierarchical bi-linear classification structure, and Clustering. Small QNNs built using parameterized quantum circuits (PQCs) reduce the errors due to gate and qubit cross-talk [15, 16]. The hierarchical structure allows more room for datasets with multiple classification categories. The final step of clustering allows us to separate the quantum states visually and provide a clear classification boundary. The architecture is tested for low-dimensional datasets (Iris flower species and Ripley's crab datasets) with results provided in Chapter 3. An additional classification result for a high-dimensional dataset is provided in Chapter 4 to demonstrate how the protocol can be scaled for bigger datasets.

Now, some of the questions that come to mind are: Does quantum information add something new to how machines recognize patterns in data? Can quantum computers help to solve problems faster? How can we create new machine learning methods using the concepts used in quantum computing? What are the ingredients of a QNN algorithm, and where lies the bottleneck? Before addressing these questions, let's examine the relevant background knowledge needed before discussing about QNNs.

1.1.1 Background

This subsection provides the basic background knowledge of ML problem types and quantum computation needed to prepare QNNs. First, the task (problem category) is identified. The problem is first categorized into one of the three main types to determine the type of ML model and tools that would work for the project problem [17].

Supervised ML models learn and get more accurate over time by training on labeled datasets. An algorithm can be trained to identify pictures of the cat by training on images labeled cats by humans. Classification and regression are two main categories of problems in supervised ML.

Unsupervised ML model is used when there is a requirement for finding patterns or trends in the unlabeled dataset. One of the main categories of problems in unsupervised ML is clustering.

Reinforcement ML model maximizes the reward in a situation by training using a trial and error method. It is used to train autonomous vehicles to drive by informing when it made the right decision, helping it learn to take appropriate action over time.

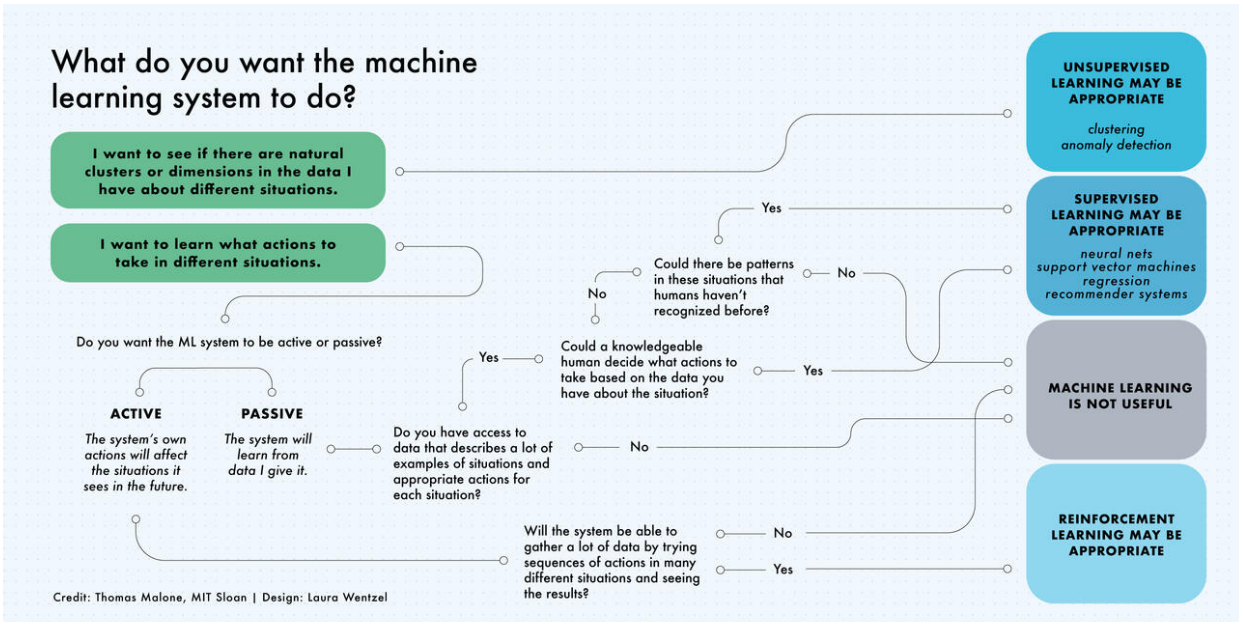

What do you want the machine learning system to do?
I want to see if there are natural clusters or dimensions in the data I have about different situations.
I want to learn what actions to take in different situations.
Do you want the ML system to be active or passive?
ACTIVE
The system's own actions will affect the situations it sees in the future.
PASSIVE
The system will learn from data I give it.
Do you have access to data that describes a lot of examples of situations and appropriate actions for each situation?
Yes
No
Could a knowledgeable human decide what actions to take based on the data you have about the situation?
Yes
Could there be patterns in these situations that humans haven't recognized before?
Yes
No
Will the system be able to gather a lot of data by trying sequences of actions in many different situations and seeing the results?
No
Yes
UNSUPERVISED LEARNING MAY BE APPROPRIATE
clustering
anomaly detection
SUPERVISED LEARNING MAY BE APPROPRIATE
neural nets
support vector machines
regression
recommender systems
MACHINE LEARNING IS NOT USEFUL
REINFORCEMENT LEARNING MAY BE APPROPRIATE
Credit: Thomas Malone, MIT Sloan | Design: Laura Wentzel

Figure 1.1: Categories of ML problems.

Depending on the type of information input and outcome required after processing (learning step), the problems can be classified into different types of ML protocols best suited for the situation. Source: Thomas Malone | MIT Sloan.

Fig. 1.1 provides a summarized flowchart of ML problem categorization. The problems in the supervised learning category are utilized for the project problem. After identifying the problem type, the next step involves using appropriate algorithms to train the model (experience) and testing the model's accuracy (performance). The specific algorithms needed to get the best model are dependent on the problem type and dataset, more details of which will be given in the next chapter.

When modeling a machine learning routine for an organization, the experts are concerned with the explainability of the ML model [18, 19]. It is the ability to explain what the ML model is doing and how it makes decisions in a way that makes sense to a person at an acceptable level. Depending on the types of algorithms used in the ML model, the level of explainability can differ.

The explainability of a model is essential to avoid bias and unintended outcomes, as systems can be fooled sometimes or make mistakes due to a programming error while performing specific tasks. Bias is often introduced due to inequalities in original data or human biases [20]. Careful vetting of data for training and ethical programming practices can help prevent the introduction of bias in the model.

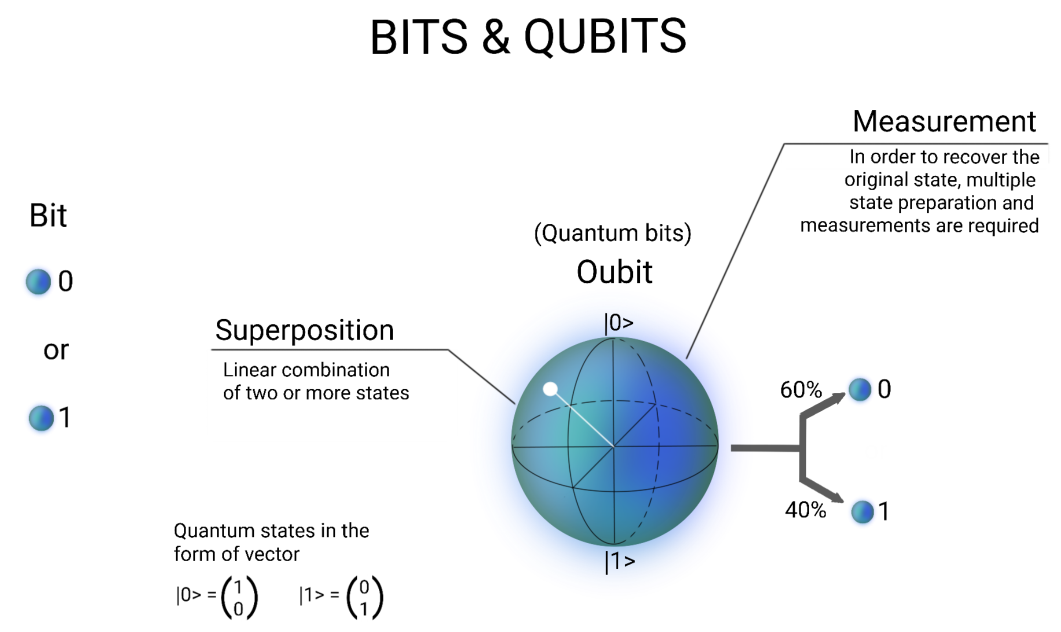

BITS & QUBITS
Measurement
In order to recover the original state, multiple state preparation and measurements are required
Bit
0
or
1
(Quantum bits)
Oubit
Superposition
Linear combination of two or more states
|0>
|1>
60%
0
40%
1
Quantum states in the form of vector
|0> = (1 0)
|1> = (0 1)

Figure 1.2: Bits and Qubits.

The difference between bits and qubits with a visual demonstration is provided.

After going through the bare minimum requirements of an ML task, we now need to understand how qubits work in order to use them for QNNs. Fig. 1.2 helps visualize how a qubit state is defined and how it differs from bits. A quantum bit is any bit created using a two-level quantum system, such as an electron or photon [21]. A quantum bit must have two states, one for "0" and one for "1," just like conventional bits. A quantum bit is capable of superposition states, incompatible measurements, and even entanglement in contrast to a classical bit. Qubits are fundamentally different from classical bits and far more powerful due to their capacity to harness the powers of superposition, interference, and entanglement. For computation purposes, the qubit states are represented by two orthonormal basis states $|0\rangle$ and $|1\rangle$ (commonly known as the computational basis or z basis states).

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Two qubit states can be represented as $|00\rangle=|0\rangle \otimes |0\rangle$, $|01\rangle=|0\rangle \otimes |1\rangle$, $|10\rangle=|1\rangle \otimes |0\rangle$, and $|11\rangle=|1\rangle \otimes |1\rangle$ (tensor product of first qubit state with the second qubit state). This is similarly extended for more qubits.

1.2 Organization of the Document/Outline

This thesis starts with a background introduction in Chapter 1, followed by the basic principles of QNN. After that, an architecture of the QNN model to implement on noisy quantum devices is proposed in Chapter 2, along with the results from IBM cloud devices in Chapter 3. The challenges and limitations of the proposed model are discussed from multiple perspectives in Chapter 4. Lastly, the concluding remarks about the work done and possible future research directions and applications of the proposed QNN model have been presented in Chapter 5.

Chapter 2

METHODS AND CALCULATIONS

2.1 Preliminaries

We need to understand the benefits of utilizing the QNN model over the CNN model for the ML problem and how the two models differ from one another before delving into the QNN architecture.

Quantum mechanical systems produce atypical patterns that are thought to be difficult for classical systems to simulate. Classical ML methods like deep neural networks can recognize these statistical patterns. With the recent advancements in quantum computing, combining quantum information processing and machine learning to identify classically complicated patterns more efficiently is becoming prevalent. The quantum model of neural networks was the earliest idea investigated in 1995, inspired by the quantum theory of how the brain works [22] (still controversial due to lack of evidence) to find explanations. The concept of associative memory always comes up during the discussion of neural networks. Hopfield's neural network [23] (or Ising model of a neural network) model served as an associative memory system (Hopfield, 1982) to understand human memory. In 1999, Dan Ventura and Tony Martinez introduced the idea of a circuit-based quantum algorithm that simulated associative memory [24]. While there is a search for quantum ML techniques to replace classical ML models due to the supposed speed-up and scaling advantage, the question of whether the advantage exists and whether there is any noticeable difference in outcomes persists.

The idea of quantum speed-up hinges on whether one adopts a formal computer science perspective (mathematical proofs) or a perspective based on what is possible with practical, finite-size devices (which requires solid statistical evidence of a scaling advantage over some finite range of problem sizes) [25]. It's not always clear how well classical algorithms will function in the context of quantum machine learning. Similar to Shor's polynomial-time quantum algorithm for integer factorization, there isn't a sub-exponential-time classical algorithm found, but it's not shown that there isn't one either. Calculating a scaling advantage between quantum and classical machine learning would require the presence of a quantum computer and is referred to as a "bench-marking" challenge. Such advantages can include increased categorization accuracy and sampling of conventionally inaccessible systems. As a result, idealized measurements from complexity theory—query complexity and gate complexity—are used to describe quantum speed-ups in machine learning. Many numerical experiments largely quantify the resources used by traditional machine learning methods. Quantifying the resources needed for quantum machine learning algorithms will probably be equally challenging. More details on the practical viability of using QML models can be studied in the references provided [26, 27].

The key difference between QNN and CNN models is how each neural network layer communicates with other layers. The output from one layer gets copied to the other in a CNN. But, in QNN, the outcome can't be duplicated due to violation of the no-cloning theorem [28, 29]. A generalized solution is to substitute the traditional fan-out technique with an arbitrary unitary that spreads out but does not transfer the output of one layer of qubits to the next layer. Information from the qubit can be passed on to the subsequent layer of qubits using this fan-out Unitary with a dummy state qubit in a known state, referred to as an Ancilla bit. This procedure complies with the reversibility condition of a quantum operation. Only two layers are active at any given time since the fan-out unitary operators in the discussed quantum neural network only affect their respective inputs. Hence, the number of qubits needed for a given step varies depending on the number of inputs in a particular layer, as no Unitary operator is active on the complete network at any given time. The effectiveness of a quantum neural network is primarily based on the number of qubits in any given layer and not on the depth of the network because quantum computers are renowned for their capacity to conduct several iterations in a short time. The references offer a more thorough examination of the quantum advantage for those interested in reading [30, 31]. After highlighting the advantages of the QNN model, the remainder of this thesis concentrates on creating a QNN architecture that can be used to build and evaluate ML problems on a NISQ device.

2.2 Data Preparation

ML algorithms gain knowledge from data [32]. Feeding the system with the proper data is essential to solve the ML problem. Even if good data is available, we still need to ensure that it is scaled, formatted, and has important features. The chances of obtaining more reliable and superior results increase with the level of data management discipline. Selecting, pre-processing, and transforming data are the three phases that make up the process of data preparation for an ML algorithm [33].

- **Selecting Data**: This step focuses on selecting the subset of all relevant data for the problem. There is always a strong desire to include all accessible data, as more is better. This could be accurate or inaccurate. Only the data required to resolve the issue should be considered. Make some assumptions about the necessary data and write them down so you can test them if necessary. When choosing data, it's important to keep certain considerations in mind.
  - How much data is currently available?
  - What data should be available but isn't?
  - What data is not required to solve the issue?
- **Pre-processing Data**: This stage aims to get the chosen data into a format that the ML model can use as input. There are three typical stages in data preprocessing:
  - *Formatting*: It's possible that the selected data isn't in an editable format. The data could be in a text file, relational database, or proprietary file format. The file type must be altered to suit the user's needs.
  - *Cleaning*: Removing or filling in blanks in data is known as cleaning. There can be instances where the necessary information to address the issue is missing. It may be required to remove these instances. It's also possible that some properties include sensitive information, in which case the data may need to be completely or anonymously erased.

  - *Sampling*: There may be more available selected data than can be used. The amount of data can cause algorithms to operate much more slowly and increase the memory and computing required. A smaller representative sample of the chosen data should be used to explore and prototype ideas more quickly before considering the entire dataset.

  The machine learning tools used on the data will likely influence the preprocessing. It is very likely to revisit this step later.

- **Transforming Data**: The processed data must be transformed as the last step. This procedure is also known as feature engineering. This step will be influenced by the particular algorithm utilized and the understanding of the problem domain. As we work on the issue, we'll probably need to go back and review various transformations of the pre-processed data. There are three typical data transformations:

  - *Scaling*: The pre-processed data may include properties that use a variety of scales for different units of measurement, such as dollars, kilograms, and sales volume. Several machine learning algorithms prefer that data characteristics maintain a consistent scale, such as a range between 0 and 1 for the least and the biggest value for a specific feature. Take into account any feature scaling as required.
  - *Decomposition*: Splitting apart some attributes that describe complicated concepts into their component bits may make them more valuable to machine learning techniques. As an illustration, consider a date that might be divided further into its day and time components. The hour may be the only time of day that matters. Think about the feature decompositions that are possible.
  - *Aggregation*: It's possible that certain features can be combined into a single feature that will have a more significant impact on the issue you're seeking to solve. For instance, data instances for each time a client logged into a system might be combined into a count for the number of logins, allowing the extra instances to be ignored. Take into account the possible feature aggregations.

A lot of time can be spent on engineering features from the data, which can help improve the performance of an algorithm. It is always best to start small and build on ideas and skills from the results.

2.3 Algorithms

In machine learning, a dataset's dimensionality is determined by how many variables represent it. Regularization techniques could help lower the risk of over-fitting, but using Feature Extraction techniques instead can also result in other benefits, such as increased accuracy, decreased over-fitting risk, accelerated training, improved data visualization, and increased model explainability. With the use of existing features, feature extraction seeks to reduce the number of features in a dataset (and then discard the original features). This new, smaller set of features should summarize most of the information in the original collection of characteristics. In this manner, combining the original set of features can result in a condensed version of the original features. Feature selection is another approach frequently used to minimize the number of features in a dataset. The goal of feature selection is to prioritize the significance of the already-present characteristics in the dataset and eliminate those that are less significant (no new features are created) [34].

2.3.1 Feature Selection

Reducing the quantity of input data features can minimize over-fitting in neural network models. The existence of redundant characteristics typically increases the variance of the result. As a result, having too many features might make a model vulnerable to noise or subtle changes in a highly correlated dataset, reducing its robustness. Following data cleaning, it is visualized using scatter plots, correlation heat maps, and a covariance matrix to look for trends, correlations, or anomalies that need further investigation. Let the data after cleaning be of the form $(\mathbf{x}_i, y_i)$, where $\mathbf{x}_i \equiv (x_{1,i}, x_{2,i}, \ldots, x_{m,i})$ is the i^{th} data feature vector ($x_{1,i}$, $x_{2,i}$,...,$x_{m,i}$ are m distinct data features/attributes) and y_i its corresponding class/target label.

One of the most effective starting tools is the pairwise plot (a scatter-plot matrix) to identify trends. With the aid of a pairwise plot, the distribution of a single variable as well as the connections between two variables can be observed. The correlation heat map helps to measure the strength of the relation between two features. Its range is $[-1,1]$, with 1 denoting perfect correlation between the two datasets and -1 denoting perfect out-of-phase correlation. Let $x1$ and $x2$ be two features with mean values $x\bar{1}$ and $x\bar{2}$, respectively.

$$r \text{ (Correlation coefficient)} = \frac{cov(x1,x2)}{\sigma_{x1}\sigma_{x2}}, \qquad \sigma_{x1} = \sqrt{\frac{\sum_{i=1}^{N}(x1i - x\bar{1})^2}{N-1}}, \qquad \sigma_{x2} = \sqrt{\frac{\sum_{i=1}^{N}(x2i - x\bar{2})^2}{N-1}} \tag{2.1}$$

$$cov(x1,x2) = \frac{1}{N-1}\sum_{i=1}^{N}(x1i - x\bar{1})(x2i - x\bar{2}) \tag{2.2}$$

2.3.2 Feature Extraction

After feature selection, feature extraction techniques are tested out to check if better features can be found. Before the application of feature extraction algorithms, the data features are normalized. This is done to put all the features in a similar range so that the true value of variables can be ascertained. Some of the ways to normalize data features are given as follows:

- **Linear transformation using mean and standard deviation**:

$$x_{Ti}^{n} = \frac{x_i^n - \bar{x}_i}{\sigma_i} \tag{2.3}$$

  x_{Ti}^{n} is the transformed n-th data point of the feature x_i.

- **Range Scaling**:

$$x_{Ti} = \frac{x_i - x_{i\ min}}{x_{i\ max} - x_{i\ min}} \tag{2.4}$$

  $x_{i\ max}$ and $x_{i\ min}$ are maximum and minimum values of the feature x_i.

The algorithms used for feature extraction are listed below:

- **Principal Components Analysis (PCA)**: One of the most common methods for reducing the number of dimensions in a linear model is PCA. While utilizing PCA, we start with our original data and search for a combination of input features that can most effectively summarize the original data distribution and minimize its original dimensions. These combinations are made so that the new variables (features selected after using extraction methods) are uncorrelated, and most of the information from the beginning variables is crammed into the first components. Principal components are the lines that, geometrically speaking, encompass most of the information in the data and reflect the directions of the data that account for the greatest amount of variance.

Before computing the covariance matrix of the features in PCA, the original data is first standardized using the scaling approach. The covariance matrix's eigenvalues and eigenvectors are obtained. The first principal component is the eigenvector with the highest eigenvalue, and the remaining components are arranged in decreasing eigenvalue order. Since PCA is an unsupervised learning technique, it doesn't care about the labels of the data but only about variation. In some instances, this may result in the incorrect classification of data.

- **Independent Component Analysis (ICA)**: ICA is a linear dimensionality reduction technique that uses a variety of independent components as input data and seeks to identify each one of them correctly. This differs from a typical PCA in seeking out non-Gaussian, statistically independent (uncorrelated) components. We need to deconstruct the data to create a set of independent factors. ICA isn't needed if there aren't numerous independent information producers to identify. ICA uses the principle of non-Gaussianity to identify independent components. The degree to which a random variable's distribution deviates from a Gaussian distribution is measured as non-Gaussianity.

  Particularly, PCA is frequently used for dimensionality reduction or data compression. By converting the input space into a maximally independent basis, ICA tries to segregate information. Both methods need input data to be scaled. It is typically advantageous to perform PCA before ICA.

- **Linear Discriminant Analysis (LDA)**: LDA is a classifier that uses supervised learning to reduce dimensionality. Each class's mean should be as far apart as possible, and spreading within each class should be minimal. Thus, LDA uses within-class and between-class comparisons as its metrics. This is a wise decision since maximizing the distance between the means of each class while projecting the data in a lower-dimensional space means the linear and nonlinear dependencies of the two input features are equal to zero. Non-Gaussian data may result in poor classification outcomes because it is anticipated that the input data when using LDA, follows a Gaussian Distribution.

After acquiring the necessary features, they are dispersed among numerous small QNNs to be trained for classification. Based on the number of features and qubits used for training in individual QNNs, the total number of QNNs is calculated. In the QNNs used for thesis work, single qubits are employed to reduce error.

2.3.3 QNN unit

A standard QNN unit consists of an encoding circuit, parameterized quantum circuit (PQC), measurement operations, and cost function [16]. The encoding quantum circuit maps the classical data represented as a column vector to a quantum state. PQC transforms the quantum state to obtain desired output state by applying unitary operators and tuning the parameters or degrees of freedom present in the operators. The output state is retrieved by performing measurements. The cost function is the fidelity between the QNN result and desired output subtracted from one, and the learning process gets repeated until the cost function is minimized. The steps involved are summarized below.

- Data encoding: Classical data as column vector : $\mathbf{X}$,
  x_k are components of $\mathbf{X}$, $x_k \xrightarrow[(Normalized\ k^{th}\ data\ component)]{Mapped\ to} \theta(x_k)$,
  therefore $\mathbf{X} \xrightarrow[(Through\ application\ of\ Quantum\ Gates\ QG(\theta(x_k)))]{Quantum\ State} |Q\rangle$.
- Applying unitary operators to the quantum state: $|Q\rangle \xrightarrow[(Initially\ guessed\ ansatz)]{PQC} |Q'\rangle$.
- Finally, the measurements are obtained to evaluate the cost function value, and the parameters of PQC are updated. After each training step, the PQC parameters are updated until the cost function value is minimum.

Encoding Circuit

The encoding quantum circuit maps the classical data represented as a column vector (each entry corresponds to a unique data feature) to a quantum state. The classical data is mapped to the quantum state by defining a suitable feature map ϕ: $\mathbb{D} \longrightarrow \mathbb{H}$, where $\mathbb{D}$ is the classical data domain and $\mathbb{H}$ is the Hilbert space of the quantum states.

For continuous variables, the classical data ($x_{j,i}$ where j represents j^{th} data feature and i represents the i^{th} data point in feature j) is first normalized, and then the angle values are obtained ($Angles(\theta)_{j,i}$) corresponding to the classical data. The angles corresponding to each data point are obtained in one of the following ways:

$$Angles(\theta)_{j,i} = \frac{2\pi(x_{j,i} - \min x_{j,k})}{\max x_{j,k} - \min x_{j,k}}, \qquad Angles(\theta)_{j,i} = 2 \times \sin^{-1} \sqrt{\left(\frac{(x_{j,i})}{\sqrt{\sum_j (x_{j,i})^2}} \right)^2} \tag{2.5}$$

These angle values ($(\theta)_{j,i}$) are used as entries for quantum gates (encoding circuit) to get the quantum state distribution of the classical data.

- Classical data as column vector : $\mathbf{X} \equiv (x_1, x_2, ..., x_n)$

- Through the application of rotation gates (like $R_y(\theta)$, $R_z(\phi)$ where θ, $\phi \in Angles(\theta)_{j,i}$), the quantum state representing a classical data vector is obtained, $\mathbf{X} \longrightarrow |\psi\rangle$.

$$|\psi\rangle = \frac{1}{\sqrt{2}} \left(\left[\cos\left(\frac{\theta}{2}\right) - e^{-i\phi} \sin\left(\frac{\theta}{2}\right) \right] |0\rangle + \left[\cos\left(\frac{\theta}{2}\right) + e^{i\phi} \sin\left(\frac{\theta}{2}\right) |1\rangle \right] \right), \quad \theta, \phi \in Angles(\theta)_{j,i} \tag{2.6}$$

The global phase will be useful when working with more than one qubit system. The quantum state in Eq. 2.6 can be obtained by application of a quantum rotation gate in the following order:

$$|\psi\rangle = R_z\left(\frac{\phi}{2}\right) R_y(\theta) R_z\left(-\frac{\phi}{2}\right) H |0\rangle \tag{2.7}$$

where the quantum gates R_y, R_z are defined as:

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad R_y(\theta) = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}, \quad R_z(\theta) = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix} \tag{2.8}$$

Many different encoding circuits were tested before finding an encoding circuit that can be used for all datasets without any changes (given in Eq. 2.6).

Parameterized Quantum Circuit (PQC)

Parameterized quantum circuits (PQC) have been suggested as machine learning models for noisy near-term intermediate-scale quantum devices because of their robustness and ease of implementation. PQC updates the quantum state (given in Eq. 2.6) by applying phase shifts on individual qubit states to change their position on the Bloch sphere after every iteration (an iteration corresponds to the application of PQC ($U_1(p_3)$, $U_2(p_2, p_3)$, $U_3(p_1, p_2, p_3)$ or a combination of them with entanglement to the encoding circuit (2.7) after updating the parameters).

$$U_1(p_3) = U_3(0,0,p_3) = \begin{bmatrix} 1 & 0 \\ 0 & e^{ip_3} \end{bmatrix}, \quad U_2(p_2,p_3) = U_3(\frac{\pi}{2},p_2,p_3) = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & -e^{ip_3} \\ e^{ip_2} & e^{i(p_2+p_3)} \end{bmatrix} \tag{2.9}$$

$$U_3(p_1,p_2,p_3) = \begin{bmatrix} \cos\left(\frac{p_1}{2}\right) & -e^{ip_3}\sin\left(\frac{p_1}{2}\right) \\ e^{ip_2}\sin\left(\frac{p_1}{2}\right) & e^{i(p_2+p_3)}\cos\left(\frac{p_1}{2}\right) \end{bmatrix} \tag{2.10}$$

The $U_1(p_3)$ gate, also known as the phase gate, rotates the quantum state by p_3 around the Z-axis direction on the Bloch sphere. $U_3(p_1, p_2, p_3)$ is the most general form of all single-qubit quantum gates.

The parameters are randomly initialized for the first iteration, measurements are obtained, and the parameters are updated (trained) using the gradient descent method. The parameters are updated to minimize the cost function value in the subsequent iterations. The circuit parameters value corresponding to the minimum cost function value is stored (remembered) to classify new data points. Depending on the data distribution, multiple applications of parameter gates and different entanglements could be required.

Measurement

It is essential to recover the quantum state prior to measurement to obtain the relative phase information that might be used to determine more accurate classification requirements. By taking measurements on a group of identical quantum states, a quantum state can be rebuilt through a technique known as quantum state tomography [35]. The measurements must be completely tomographic in order to be able to identify the state by itself. In other words, the measured operators must provide an operator basis on the system's Hilbert space, containing all the state-specific data.

One method entails taking measurements in phase space along various rotational directions. The probability density of measurements in the direction of phase space θ that result in the value q can be found as a probability distribution $w(q,\theta)$ for each direction θ. The Wigner function, $\mathrm{W}(x,p)$, is obtained by applying an inverse Radon transformation (the filtered back projection) on the function $w(q,\theta)$, and an inverse Fourier transformation transforms it into the density matrix for the state in any basis.

For the case of qubits, a single qubit's density matrix can be represented in terms of its Pauli and Bloch vectors, $\vec{\sigma}$ and $\vec{r}$, respectively:

$$\rho = \frac{1}{2}(\mathrm{I}+\vec{r}.\vec{\sigma}) = \frac{1}{2}\begin{bmatrix} 1+r_z & r_x - ir_y \\ r_x + ir_y & 1-r_z \end{bmatrix} \tag{2.11}$$

Single-qubit Pauli measurements can be used to do single-qubit state tomography.

- Make a list of three quantum circuits first, with the first measuring the qubit in the computational basis (Z-basis), the second doing a Hadamard gate prior to measurement (measurement in X-basis). The third performing the proper phase shift gate ($\sqrt{Z}^{\dagger} = |0\rangle\langle 0| + \exp(-\mathrm{i}\pi/2)|1\rangle\langle 1|$) followed by a Hadamard gate prior to measurement (measurement in Y-basis).

- The measurement results of the first circuit then used to create $\bar{z} = (n_{z,+} - n_{z,-})/(n_{z,+} + n_{z,-})$, the second circuit produces $\bar{x}$, and the third circuit produces $\bar{y}$.

- Last but not least, if $\bar{x}^2 + \bar{y}^2 + \bar{z}^2 \leq 1$, then a measured Bloch vector is formed as $\vec{r}_m = (\bar{x}, \bar{y}, \bar{z})$, and the measured density matrix is $\rho_m = \frac{1}{2}(I + \vec{r}_m \cdot \vec{\sigma})$. It will be essential to re-normalize the measured Bloch vector as $\vec{r}_m = (\bar{x}, \bar{y}, \bar{z})/\sqrt{\bar{x}^2 + \bar{y}^2 + \bar{z}^2}$ if $\bar{x}^2 + \bar{y}^2 + \bar{z}^2 > 1$ before calculating the measured density matrix using it.

Qubit tomography is based on the aforementioned algorithm, which is also used in several quantum programming operations.

Cost function

The cost function is calculated classically from repeated measurement outcomes. The cost function is defined using the QNN circuit measurement and desired output's fidelity. Fidelity conveys information about how close two quantum states are. For measurement output density matrix (ρ_{out}) and desired output density matrix (σ_{out}), the fidelity $F(\rho_{out}, \sigma_{out})$ is defined as:

$$F(\rho_{out}, \sigma_{out}) = \left(Tr(\sqrt{\sqrt{\rho_{out}}\sigma_{out}\sqrt{\rho_{out}}}) \right)^2 \tag{2.12}$$

In case of pure quantum states; $F(\rho_{out}, \sigma_{out})$=$|\langle\psi_\rho|\,|\psi_\sigma\rangle|^2$ (ρ_{out}=$|\psi_\rho\rangle\langle\psi_\rho|$, σ_{out}=$|\psi_\sigma\rangle\langle\psi_\sigma|$). It may not be obvious from the general definition of fidelity, but it is symmetric ($F(\rho_{out}, \sigma_{out})$=$F(\sigma_{out}, \rho_{out})$).

Using the given definition of fidelity, the mean of fidelity of the training data points (i going from 1 to N) is used to write down the cost function (C) such that the minimum value of the cost function (0) will be attained when the maximum fidelity value (1) is reached.

$$C = 1 - \frac{1}{N}\sum_{i=1}^{N} F(\rho_{out}, \sigma_{out}^{i}) \tag{2.13}$$

The parameters of PQC are updated using the gradient descent method and cost function. To find a local minimum of a differentiable function, gradient descent is a first-order iterative optimization technique in mathematics. As the direction of the steepest descent is in the opposite direction of the function's gradient at the present point, the idea is to move in that direction repeatedly. $C(\vec{p}_n)$

decreases fastest if one goes from $\vec{p}_n$ in the direction of the negative gradient of C at $\vec{p}_n$,$-\nabla C(\vec{p}_n)$. Therefore, the parameter is updated as:

$$\vec{p}_{n+1} = \vec{p}_n - \gamma \nabla C(\vec{p}_n) \tag{2.14}$$

where $\gamma \in R^+$ is the learning rate (step size).The term $\gamma \nabla C(\vec{p}_n)$ is subtracted from $\vec{p}_n$ to move against the gradient and reach the minimum. On reaching the cost function minimum, the training process of QNN is complete, and the final parameters are saved.

2.3.4 Statistical distances

The quantum state obtained from the QNN circuit using the optimized parameters of PQC is reconstructed by obtaining measurement results. After that, the statistical distances (ds_{PS}) [36] between the output states and the mean state of each class is obtained, where ds_{PS} is defined as:

$$\frac{1}{4}(ds_{PS})_i^2 = [\cos^{-1}(|\langle(\psi_{mean})_i|\psi_b\rangle|)]^2 = 1 - |\langle(\psi_{mean})_i|\psi_b\rangle|^2 \tag{2.15}$$

Using the statistical distance data, the main clusters are identified and separated. The data is classified into one of the identified main clusters. The sub-clusters present in these main clusters are identified with the help of another round of detailed analysis and QNN unit training. Classical clustering protocols (like k-means) are used to separate the target labels.

2.4 Tools and cloud devices

IBM Quantum aims to increase the amount of useful work that quantum computing systems can accomplish (quantum computing performance). The overarching idea is that scale, quality, and speed are the three essential components of performance. The superconducting quantum systems at IBM Quantum are prepared to lead in pushing on all three areas in order to integrate quantum computers into enterprises' computing activities. The progress in the number of qubits in the system measures the scale. Quantum volume is used to measure the quality. CLOPS is the metric used to measure speed.

All the protocols used in the work of this thesis have been implemented on IBM quantum cloud devices. It is, therefore, important to understand these devices' attributes and limitations.

- **Qubits**: We introduced qubits and discussed how they differ from conventional bits in Chapter 1. The qubit can be any two-level quantum mechanical system. If a multilevel system has two states that can be successfully decoupled from the others, it can also be employed. Some of the most prevalent qubit systems are listed below as examples.
  - *Spin*: The majority of quantum particles act like tiny magnets. This property is called spin. A spin qubit can be constructed with "0" as the spin-up state and "1" as the spin-down state.
  - *Trapped Atoms and Ions*: Qubits can be created using the energy levels of the electrons in neutral atoms or ions. These electrons are at the lowest energy levels imaginable in their native state. Lasers can "excite" them to a higher energy level. According to its energy state, the qubit can be given values with "0" denoting a low energy state and "1" denoting a high energy level.
  - *Photons*: There are various ways that photons, which are discrete units of light, might be employed as qubits. For example, A photon qubit can also be created using its arrival time (TIME QUBIT: A "photon coming early" ("0") and a "photon arriving late" ("1")).
  - *Superconducting Circuits*: Some materials allow an electrical current to pass without resistance when chilled to a low temperature. These are called superconductors. Superconducting-based electrical circuits can be created with qubit-like behavior. These artificial systems, which differ from earlier qubit instances in that they are composed of billions of atoms, still function as a single quantum system. An electrical circuit's current flow direction can be given a value in order to create a superconducting qubit ("0" as clockwise current and "1" as anticlockwise current).
- **Quantum Volume (QV)**: Quantum Volume measures the efficiency with which a quantum computer can operate a circuit of arbitrary two-qubit gates functioning concurrently on a

subset of the device's qubits. It incorporates system characteristics, including connectivity, the number of qubits, gate, spectator, and measurement errors, into a single numerical value.

Increased processor performance is directly correlated with increased quantum volumes. Because the number of transistors in a classical computer and the number of quantum bits in a quantum computer are different, IBM developed the Quantum Volume metric. Few fault-tolerant bits are more valuable as a performance indicator than a larger number of noisy, error-prone qubits because qubits decohere and suffer a corresponding performance loss.

- **Circuit Layer Operations Per Second (CLOPS)**: CLOPS is a metric proposed by IBM Quantum that measures the speed of quantum circuits, the fundamental building block of quantum computation that encompasses both the interaction of the quantum system with a classical computer and the sequence of quantum operations. A quantum program incorporates some classical computing: programmers translate instructions into a format that can be used by the quantum processing unit (QPU) and receive calculation results using classical gear, such as a laptop.

  Many quantum circuits are realized with each "question" transmitted to the quantum computer. Therefore, the entire system's performance depends on how quickly the quantum-classical interaction takes place. This speed, which takes into account not only the time spent running each circuit on the device but also the time elapsed between each circuit's shot on the system and the time spent getting the circuits ready to run, is what CLOPS measures.

- **Processor Type**: A laptop's keyboard wafer is about the size of an IBM Quantum Processor. A quantum hardware setup is also roughly the size of a vehicle and consists primarily of cooling mechanisms to maintain the superconducting processor's ultra-cold operational temperature. A conventional processor uses bits to perform its tasks. A quantum computer uses qubits to execute multidimensional quantum computations.

  The temperature of the quantum computers is lowered close to absolute zero using super-cooled super-fluids to produce superconductors. Josephson junctions, employed as super-conducting qubits, are created when two superconductors are put on opposite sides of an

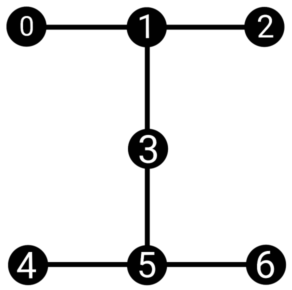


Figure 2.1: Segment H of a Falcon processor.

Source: IBM Quantum Computing

insulator at extremely low temperatures. These qubits may be made to keep, modify, and read out discrete pieces of quantum information by exposing them to microwave photons that can be controlled.

The family and revision are two general technological characteristics that go into building a processor type. Family (for instance, Falcon) describes the size and scope of circuits that can be placed on the chip. The connectivity graph and the number of qubits mostly determine this. Revisions (such as r1) are design variations within a single family, frequently resulting in performance gains or trade-offs. Within a particular family, segments are specified as chip sub-sections. For instance, segment H of a Falcon has seven qubits set up, as depicted in Fig. 2.1.

IBM Quantum's mission is to create a workable quantum computing system, and they think that superconducting qubit systems have the best chance of achieving this aim. They claim that various quantum designs can perform well in some aspects of scale, quality, and speed, but not all. As an illustration, confined ions have demonstrated the ability to reach high Quantum Volume but have difficulty achieving speed. In contrast, spin qubits can operate at high speeds but haven't been able to push quality or scale yet. In achieving further performance improvements in scalability, quality, and speed, they anticipate that superconducting qubits will offer the most potential for all three.

The IBM Perth, Oslo, and Quito cloud devices are used to check the performance of the QNN

Table 2.1: IBM cloud quantum resources and their attributes.

Cloud Devices	Qubits	Quantum Volume (QV)	Circuit Layer Operations Per Second (CLOPS)	Processor Type	Calibration Data (Link)
ibm_perth	7	32	2900	Falcon r5.11H	Perth
ibm_lagos	7	32	2700	Falcon r5.11H	Lagos
ibm_nairobi	7	32	2600	Falcon r5.11H	Nairobi
ibm_oslo	7	32	2600	Falcon r5.11H	Oslo
ibmq_jakarta	7	16	2400	Falcon r5.11H	Jakarta
ibmq_manila	5	32	2800	Falcon r5.11L	Manila
ibmq_quito	5	16	2500	Falcon r4T	Quito
ibmq_belem	5	16	2500	Falcon r4T	Belem
ibmq_lima	5	8	2700	Falcon r4T	Lima
ibmq_qasm_simulator	32	-	-	General, context-aware	-

units against the simulator results.

Chapter 3

RESULTS

Python, Jupyter Notebook, Qiskit, and IBM cloud devices are all used in the QNN architecture implementation and testing. The suggested QNN protocol implementation is described in detail in the next section. In Appendix A, the codes used are listed. Plots are used to show the distribution, and a subset of the data is given when it is pertinent. The Iris flower and Ripley's crab species are classified using the protocol. This chapter is divided into two parts, each part dedicated to the individual dataset. The layout is explained before going through the results.

First, data-related details are gathered, such as the number of features, individual feature data type, null data points, and the total number of distinct data points. After the null data points are eliminated, the statistics for the remaining features are gathered. The next step is to do a univariate and bivariate analysis to find outliers and clusters. A pairwise plot is obtained to see if the clusters belong to different target labels or if their distributions overlap. For subsequent feature extraction methods (PCA, ICA, and LDA extract features from the scaled data), the correlation heat map is utilized to determine the degree of connection between features and target labels. The selected features are used as input for the encoding circuit.

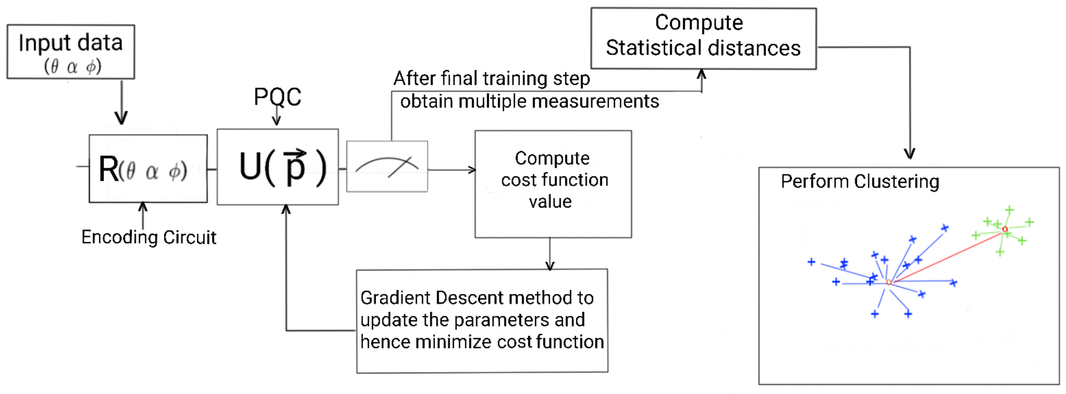

Input data
(θ α ϕ)
R(θ α ϕ)
Encoding Circuit
PQC
U(p⃗)
After final training step
obtain multiple measurements
Compute
Statistical distances
Compute
cost function
value
Gradient Descent method to
update the parameters and
hence minimize cost function
Perform Clustering

Figure 3.1: A QNN unit in the Hierarchical structure.

The input data is the angles obtained from scaling the final selected features.

The distribution from the encoding circuit is collected by performing qubit tomography to see if the mapping results are as expected. The number of QNN units required to implement the classification problem depends on the dataset's selected features and target labels. The final classification model corresponding to each dataset is provided at the end of each dataset section. All the individual QNN units, as shown in Fig. 3.1, used in the datasets are made of a single qubit to make the fidelity and updating parameter calculations easy to follow.

3.1 Iris Flower Dataset

In his 1936 publication "The use of many measures in taxonomic issues," British statistician and biologist Ronald Fisher introduced the multivariate data set for the iris flower [37]. The information was gathered to measure the morphologic variance of Iris blossoms from three related species. The three iris species in the data set have 50 samples each. From each sample, the length and width of the sepals and petals in centimeters were measured. Several machine-learning statistical classification approaches used this dataset as a standard test scenario.

3.1.1 Data Preparation

The raw data information table provides the details of the number of data points available. The data type, species label used, and null points are also provided. The data features are length and width measurements of the sepal and petal of the iris flower, which need to be scaled before using them as input for feature extraction. For the first QNN training, the complete 150 data point is taken. Only the data points corresponding to the bigger cluster (species 1 and 2) are taken for the training of the second QNN unit.

Table 3.1: Iris Flower Raw Data

S. No.	sepal_length	sepal_width	petal_length	petal_width	species	species label
1	5.1	3.5	1.4	0.2	Iris-setosa	0
2	5.4	3	4.5	1.5	Iris-versicolor	1
3	6.3	2.5	5	1.9	Iris-virginica	2

Feature	Null Data Points	Total Count	Data Type
sepal_length	0	150	float
sepal_width	0	150	float
petal_length	0	150	float
petal_width	0	150	float

Species	Total Count	Data Label Type
Iris-setosa(0)	50	integer
Iris-versicolor(1)	50	integer
Iris-virginica(2)	50	integer

It can be deduced from the raw data, univariate, and bivariate analysis (Fig. 3.2 and 3.3) that two clusters are highlighted by the petal length and width attributes. The boxplot of the sepal width feature shows four outlying points; however, those points are left in since they are not too far from

Table 3.2: Iris Flower Raw Data Statistical Information

Info Type	sepal_length	sepal_width	petal_length	petal_width
mean	5.84	3.05	3.76	1.20
std. dev.	0.83	0.43	1.76	0.76
min	4.3	2.0	1.0	0.1
25%	5.1	2.8	1.6	0.3
50%	5.8	3.0	4.4	1.3
75%	6.4	3.3	5.1	1.8
max	7.9	4.4	6.9	2.5

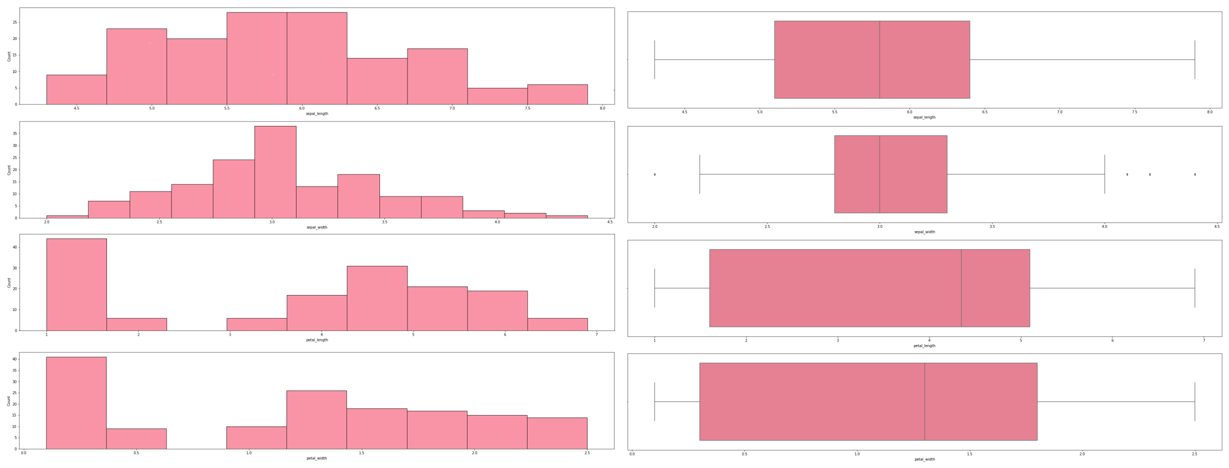


Figure 3.2: Iris Flower Features Univariate Analysis.

Sepal length is shown at the top of the univariate plot, followed by sepal width, petal length, and petal width at the bottom. The dataset's frequency distribution is shown in the left section's histogram. The boxplot in the right section displays the minimum, first quartile, median, third quartile, and maximum values.

the minimum and maximum values. Whole raw data is scaled further for feature extraction because no null values exist.

The PCA, ICA, and LDA results (Fig. 3.4) are compared with the help of a correlation heat map to select appropriate features. The final selected features are collected, stored in a separate data file, and used for the QNN encoding circuit.

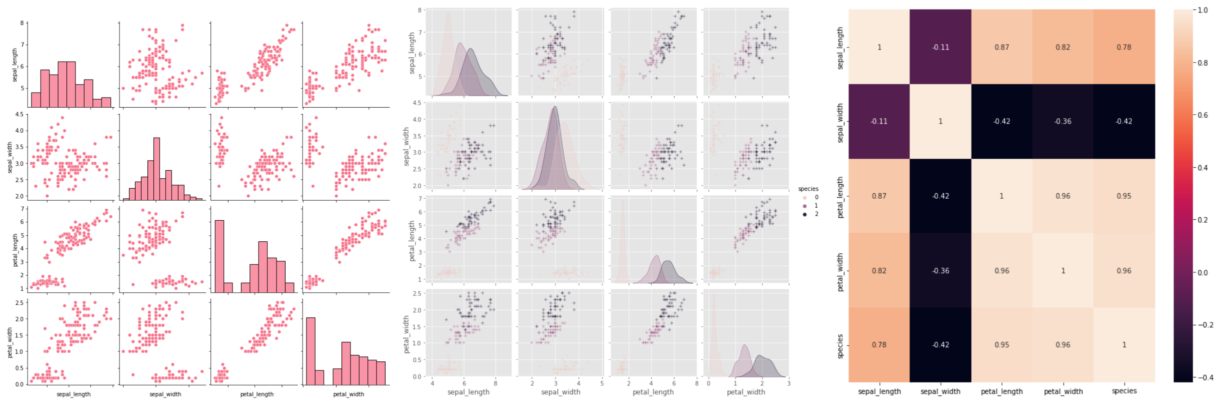


Figure 3.3: Iris Flower Features Bivariate Analysis.

The pairwise plot of petal length and width makes it obvious that two data clusters are present. A second pairwise plot is obtained to confirm that the clusters correspond to different species. The heat map of correlation reveals a strong relationship between species and petal length and width.

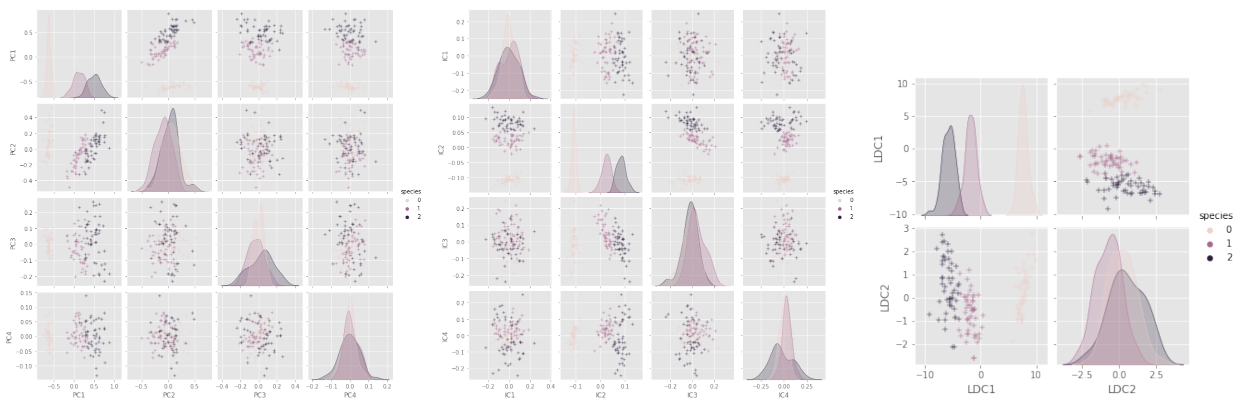


Figure 3.4: PCA, ICA, and LDA of scaled Iris Flower data for first QNN unit.

The figures demonstrate that species 0 and species 1 and 2 are separated by PC1, IC2, and LDC1. PC1 and IC2 are the input data for the encoding circuit in the first QNN unit, as they have the highest correlation with species labels for the two clusters.

3.1.2 *First QNN Unit*

The first QNN unit is used to separate species 0 from species 1 and 2. PC1 and IC2 are selected as the features to be used for the QNN unit. The PC1 and IC2 data points are multiplied by 2π to use as input (θ and ϕ respectively) for the circuit gates.

Encoding circuit

Initially, the qubit is in $|0\rangle$ state. From Chapter 2 Eq. 2.7, the quantum state after the application of the encoding circuit is given as:

$$|Q_{enc}\rangle = R_z\left(\frac{\phi}{2}\right)R_y(\theta)R_z\left(-\frac{\phi}{2}\right)H|0\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix} \cos\left(\frac{\theta}{2}\right) - e^{-i\phi}\sin\left(\frac{\theta}{2}\right) \\ \cos\left(\frac{\theta}{2}\right) + e^{i\phi}\sin\left(\frac{\theta}{2}\right) \end{bmatrix} \tag{3.1}$$

The simulator measurement results of the encoding circuit (Fig. 3.5) are used to identify centroids of clusters to be separated. These centroids are used to calculate the fidelity of the entire QNN circuit during the training steps.

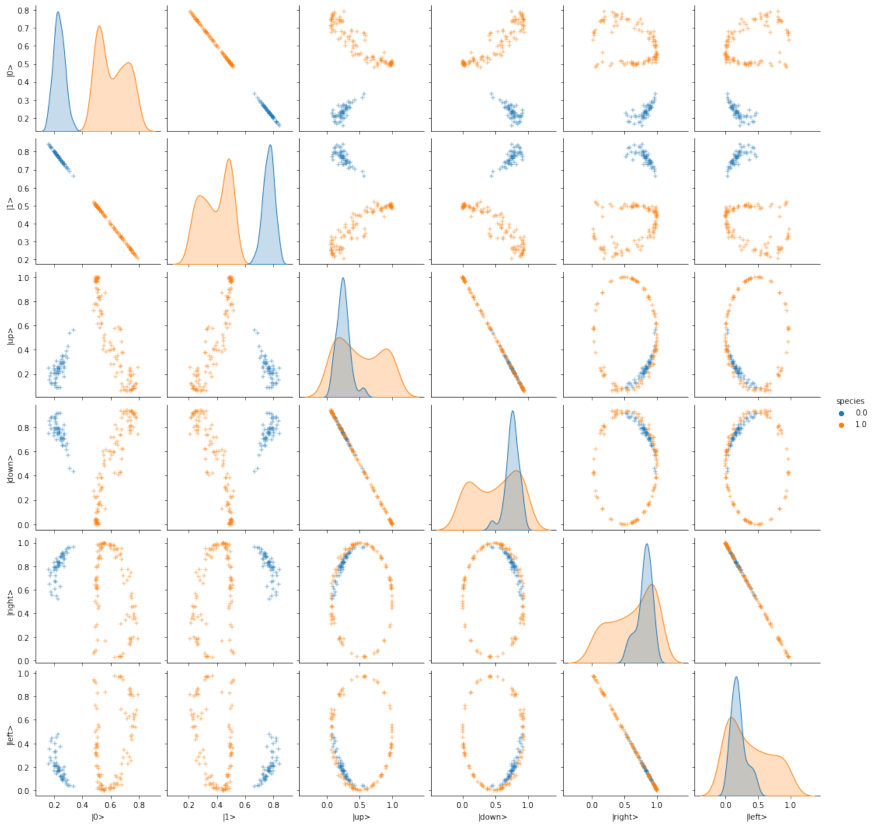


Figure 3.5: Iris Flower encoding circuit distribution for first QNN unit.

The measurement results are obtained in Z $(|0\rangle, |1\rangle)$, X $(|up\rangle, |down\rangle)$, and Y $(|right\rangle, |left\rangle)$ basis. The encoding circuit separates the two clusters. The phase gate parameter is still trained to adhere to the protocol.

PQC

The Phase gate is used to separate the two distributions of quantum states as it is only a one-qubit system. The introduction of too many parameters could lead to more errors and redundancy in the model.

$$|Q_{PQC}\rangle = U_1(\lambda)R_z\left(\frac{\phi}{2}\right)R_y(\theta)R_z\left(-\frac{\phi}{2}\right)H|0\rangle \tag{3.2}$$

$$|Q_{PQC}\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix} \cos\left(\frac{\theta}{2}\right) - e^{-i\phi}\sin\left(\frac{\theta}{2}\right) \\ e^{i\lambda}\left(\cos\left(\frac{\theta}{2}\right) + e^{i\phi}\sin\left(\frac{\theta}{2}\right)\right) \end{bmatrix} \tag{3.3}$$

The parameter λ is trained to get maximum fidelity (minimum cost function value). The training results from the simulator and cloud devices are tabulated below. The classification accuracy corresponds to the accuracy with which the two clusters can be separated using the statistical data of the final measurement results.

Table 3.3: Iris Flower First QNN Unit

The Optimized Parameter and Accuracy Results from IBM cloud quantum resources are tabulated.

Cloud Device	Parameter Value	Maximum Fidelity	Classification Accuracy
ibmq_qasm_simulator	0	0.8301741163735193	100%
ibm_perth	0.0003873399945222867	0.8299820064457042	100%
ibm_oslo	-0.0029373231940700894	0.8202756824232105	100%
ibmq_quito	-0.000349514940277223	0.8203307288621763	100%

The effect of noise can be noticed from the lower fidelity values of the cloud devices when compared to the simulator result (Table 3.3). The parameter value obtained from the training also deviated from the simulator result. The reason for the Nairobi device to achieve such a close parameter value could be sudden fluctuations in the systems caused due to large amount of usage by various users or slow calibration. The fidelity value of the Perth device is closest to the simulator result. This can be due to higher QV and CLOPS values compared to other devices. The two clusters remain separated even with deviations in the quantum state distribution for different devices. Hence the accuracy of classification remains 100% in all the cases.

Species 0 target label data points are classified and removed from further training. The other cluster is processed through a new second QNN unit to separate the remaining two species. For the second part of separating sub-clusters, the raw data of the bigger cluster is analyzed again. The extracted features from the PCA, ICA, and LDA plot given in Fig. 3.6 are used for training the second QNN unit.

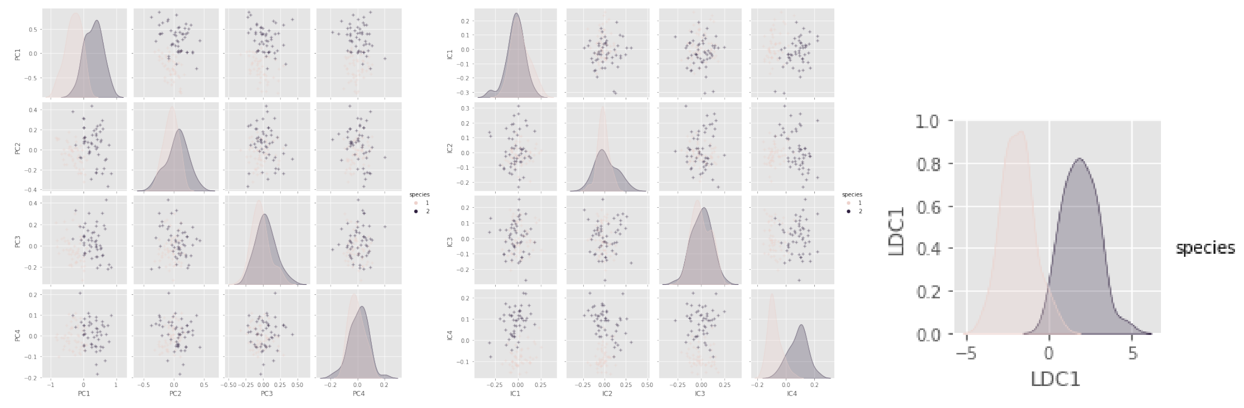


Figure 3.6: Iris Flower Features obtained from extraction.
The pairwise plot and correlation values show that IC4 and LDC1 are the best choices for features to separate the clusters of the remaining two iris flower species.

3.1.3 Second QNN Unit

The second QNN unit is used to separate species 1 (renamed as '0') and species 2 (renamed as'1'). IC4 and LDC1 are selected as input features after analyzing the data distributions again. The IC4 and LDC1 data points are multiplied by $\pi/2$ to use as input (ϕ and θ respectively) for the circuit gates.

Encoding circuit

Initially, the qubit is in $|0\rangle$ state. The quantum state after the application of the encoding circuit (same as the one given in Chapter 2, Eq. 2.7) is given as:

$$|Q_{enc}\rangle = R_z\left(\frac{\phi}{2}\right) R_y(\theta) R_z\left(-\frac{\phi}{2}\right) H\,|0\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) - e^{-i\phi}\sin\left(\frac{\theta}{2}\right) \\ \cos\left(\frac{\theta}{2}\right) + e^{i\phi}\sin\left(\frac{\theta}{2}\right) \end{bmatrix} \tag{3.4}$$

Same as the first QNN unit, the simulator measurement results of the encoding circuit (Fig. 3.7) are used to identify centroids of clusters to be separated. These centroids are used to calculate the fidelity of the entire QNN circuit during the training steps.

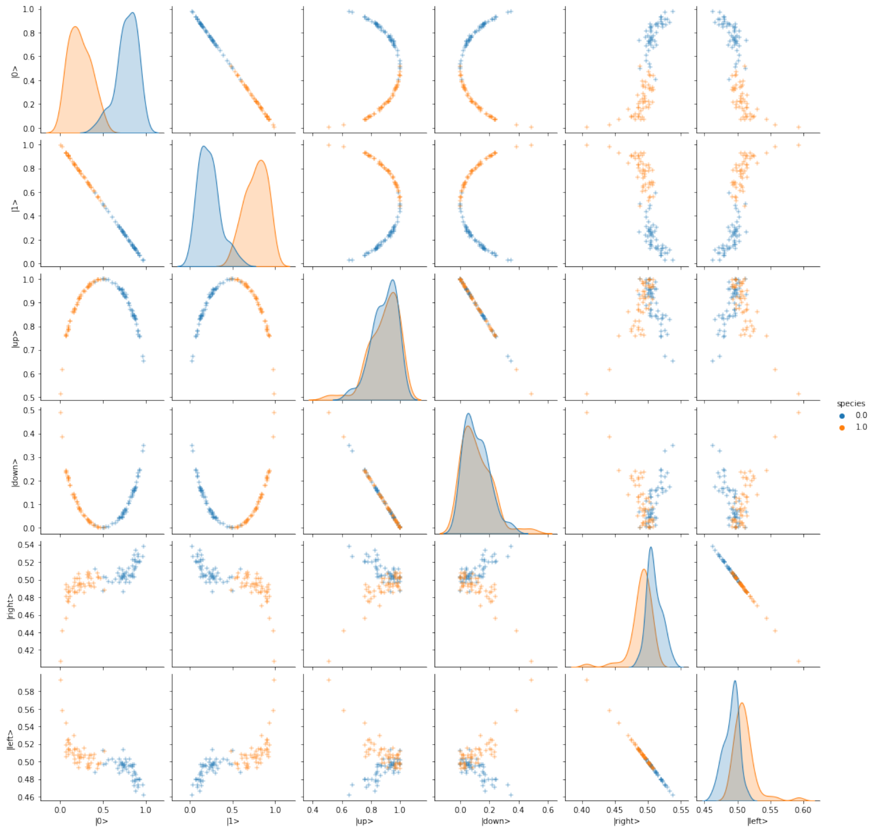


Figure 3.7: Iris Flower encoding circuit distribution for second QNN unit

The measurement results are obtained in Z $(|0\rangle, |1\rangle)$, X $(|up\rangle, |down\rangle)$, and Y $(|right\rangle, |left\rangle)$ basis. The encoding circuit separates the two clusters with a small overlap.

PQC

The two clusters have an overlap which is reduced by training the parameter of the Phase gate.

$$|Q_{PQC}\rangle = U_1(\lambda) R_z\left(\frac{\phi}{2}\right) R_y(\theta) R_z\left(-\frac{\phi}{2}\right) H|0\rangle \tag{3.5}$$

$$|Q_{PQC}\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) - e^{-i\phi}\sin\left(\frac{\theta}{2}\right) \\ e^{i\lambda}\left(\cos\left(\frac{\theta}{2}\right) + e^{i\phi}\sin\left(\frac{\theta}{2}\right)\right) \end{bmatrix} \tag{3.6}$$

Table 3.4: Iris Flower Second QNN Unit

The Optimized Parameter and Accuracy Results from IBM cloud quantum resources are tabulated.

Cloud	Parameter Value	Maximum Fidelity	Classification Accuracy
ibmq_qasm_simulator	0.5899472986937727	0.8675201572767639	98%
ibm_perth	0.5909252006559969	0.8682670104750781	97%
ibm_oslo	0.5911934427928294	0.8674517123083092	97%
ibmq_quito	0.5908367869152529	0.8666303111786496	97%

It can be observed that the overall fidelity is a bit lower than the first QNN unit (Table 3.4). This is because the raw data of species 1 and 2 have close feature values, making it hard to differentiate between some data points. In the final simulator result, two data points are not accurately classified because of the overlapping data points. Generally, the classification outcomes from the cloud devices are comparable to the outcome from the simulator. It is noticeable that the Perth device has greater fidelity than the simulator. This might be a result of the slight noise-induced fluctuation in distribution. Fig. 3.8 shows the final classification model.

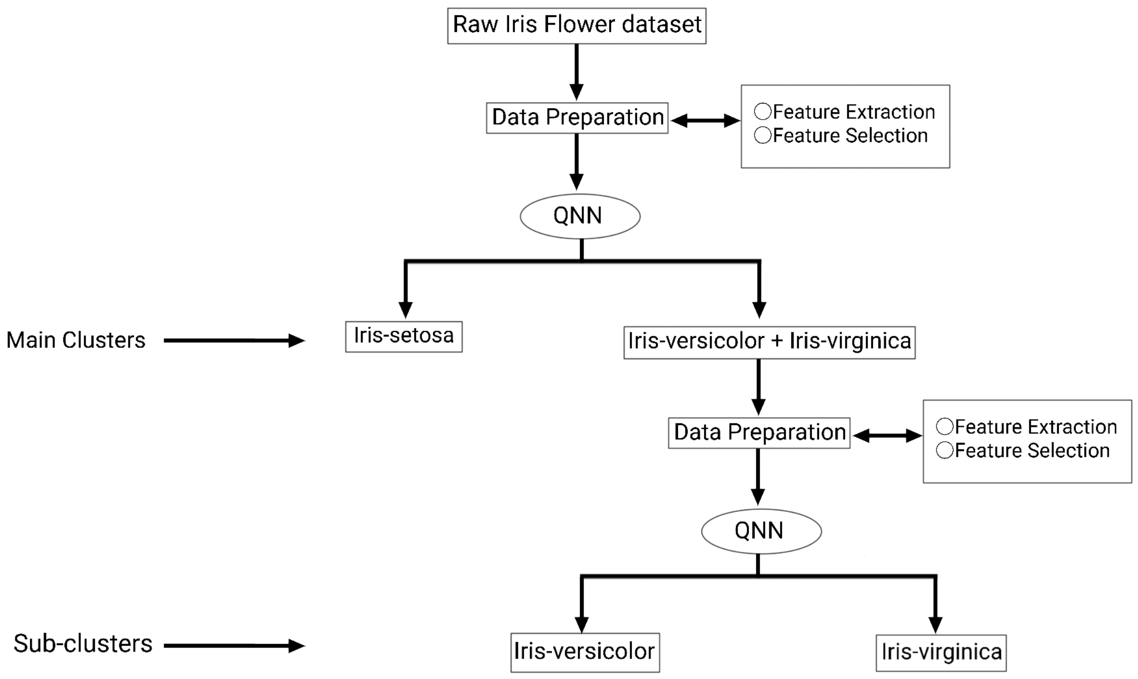


Figure 3.8: Iris Flower classification model.

3.2 Ripley's Crab Dataset

The dataset for the Leptograpsus variegatus crab species gathered in Fremantle, Western Australia, comprises 200 rows and 8 columns, describing 5 morphological measurements on 100 crabs each of the two color types (blue or orange), with 50 male and 50 female crabs in each color [38]. Mostly, the R programming language uses this dataset (R-data statistics). For the first QNN unit, only the species are separated.

3.2.1 Data Preparation

The raw data information is provided in the tables below.

Table 3.5: Ripley's Crab Raw Data

S. No.	FL	RW	CL	CW	BD	species label
1	12.8	10.2	27.2	31.8	10.9	0 (blue)
2	21.6	14.8	43.4	48.2	20.1	1 (orange)

Feature	Null Data Points	Total Count	Data Type
FL	0	200	float
RW	0	200	float
CL	0	200	float
CW	0	200	float
BD	0	200	float

Species	Total Count	Data Label Type
0 (blue)	100	integer
1 (orange)	100	integer

Table 3.6: Ripley's Crab Raw Data Statistical Information

Info Type	FL	RW	CL	CW	BD
mean	15.583000	12.73850	32.105500	36.414500	14.030500
std. dev.	3.495325	2.57334	7.118983	7.871955	3.424772
min	7.2	6.5	14.7	17.1	6.1
25%	12.9	11.0	27.275	31.5	11.4
50%	15.55	12.8	32.1	36.8	13.9
75%	18.05	14.3	37.225	42.0	16.6
max	23.1	20.2	47.6	54.6	2.521.6

It can be deduced from the raw data, univariate, and bivariate analysis (Fig. 3.9 and 3.10) that no clusters are highlighted by any of the attributes. The boxplot of RW shows one outlying point, which is not removed as it is close to the maximum value. Whole raw data is scaled further for feature extraction because no null values exist.

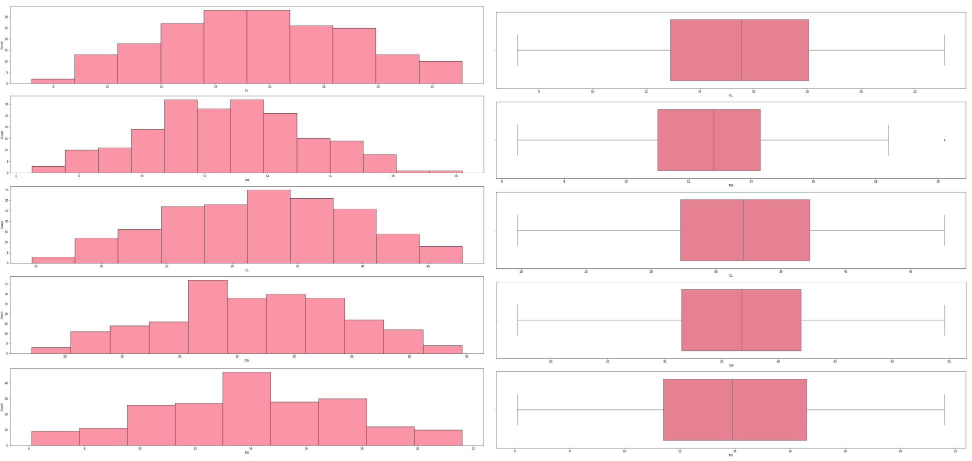


Figure 3.9: Ripley's Crab Features Univariate Analysis.

FL is shown at the top of the univariate analysis of Ripley's Crab features, followed by RW, CL, CW, and BD at the bottom. The dataset's frequency distribution is shown in the left section's histogram. The boxplot in the right section displays the minimum, first quartile, median, third quartile, and maximum values.

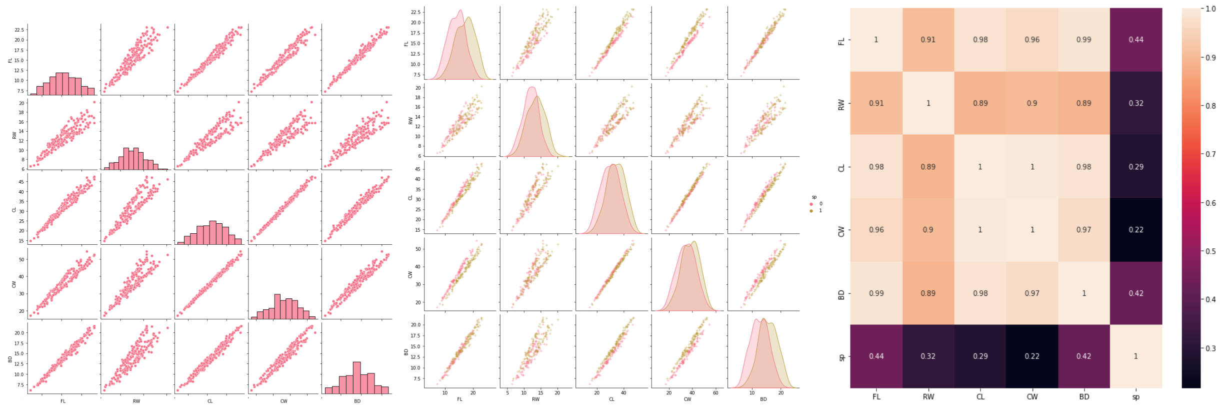


Figure 3.10: Ripley's Crab Features Bivariate Analysis.

The pairwise plot of petal length and width makes it obvious that two data clusters are present. A second pairwise plot is obtained to confirm that the clusters correspond to different species. The heat map of correlation reveals no strong relationship between species and the features.

To choose the best features, the PCA, ICA, and LDA outputs are compared (Fig. 3.11). In order to employ the final chosen features for the QNN encoding circuit, they are gathered and saved in a separate data file.

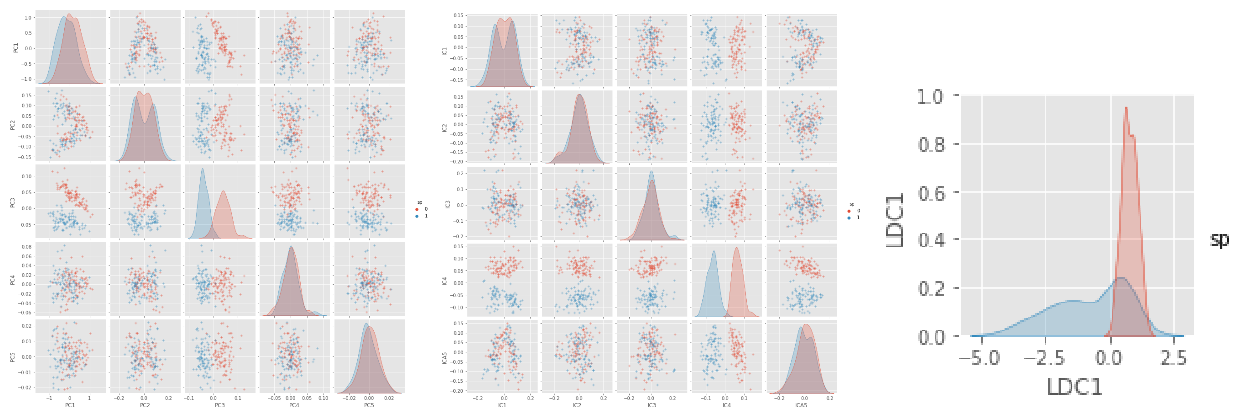


Figure 3.11: PCA, ICA, and LDA of scaled Ripley's crab data for the first QNN unit.

The figures demonstrate that species 0 and 1 are separated by PC3 and IC4. Thus, PC3 and IC4 are used as the input data for the encoding circuit in the first QNN unit.

3.2.2 First QNN Unit

The first QNN unit is used to separate species 0 and species 1. The PC3 and IC4 data points are multiplied by 2π to use them as input (θ and ϕ respectively) for the circuit gates.

Encoding circuit

Initially, the qubit is in $|0\rangle$ state. The quantum state after the application of the encoding circuit is given as follows:

$$|Q_{enc}\rangle = R_z\left(\frac{\phi}{2}\right)R_y(\theta)R_z\left(-\frac{\phi}{2}\right)H\,|0\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix} \cos\left(\frac{\theta}{2}\right) - e^{-i\phi}\sin\left(\frac{\theta}{2}\right) \\ \cos\left(\frac{\theta}{2}\right) + e^{i\phi}\sin\left(\frac{\theta}{2}\right) \end{bmatrix} \tag{3.7}$$

Similar to iris flower QNN units, the simulator measurement results of the encoding circuit (Fig. 3.12) are used to identify centroids of clusters to be separated. These centroids are used to calculate the fidelity of the entire QNN circuit during the training steps.

PQC

The Phase gate separates the quantum states as it is only a one-qubit system.

$$|Q_{PQC}\rangle = U_1(\lambda)R_z\left(\frac{\phi}{2}\right)R_y(\theta)R_z\left(-\frac{\phi}{2}\right)H\,|0\rangle \tag{3.8}$$

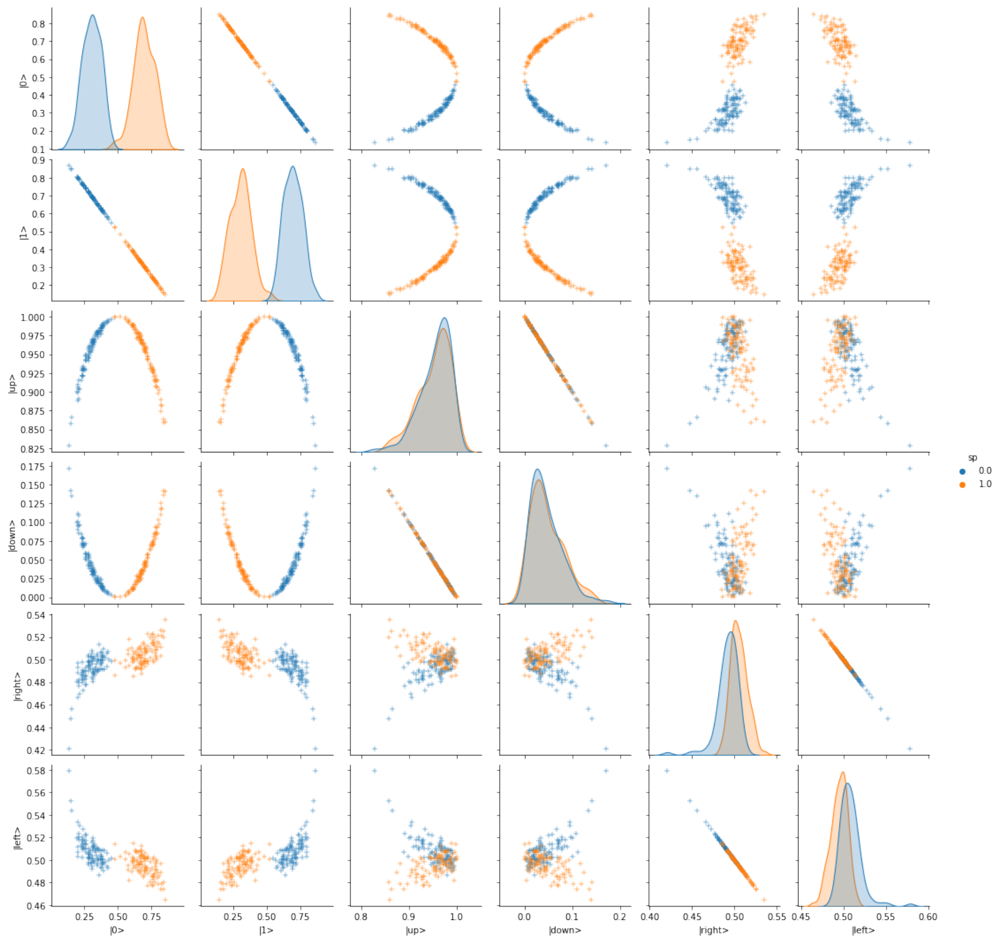


Figure 3.12: Ripley's Crab encoding circuit distribution for first QNN unit.

The measurement results are obtained in Z ($|0\rangle, |1\rangle$), X ($|up\rangle, |down\rangle$), and Y ($|right\rangle, |left\rangle$) basis. The encoding circuit separates the two clusters. The distance between the two cluster distributions will be increased in the QNN training.

$$|Q_{PQC}\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) - e^{-i\phi}\sin\left(\frac{\theta}{2}\right) \\ e^{i\lambda}\left(\cos\left(\frac{\theta}{2}\right) + e^{i\phi}\sin\left(\frac{\theta}{2}\right)\right) \end{bmatrix} \tag{3.9}$$

Table 3.7: Ripley's Crab First QNN Unit

The Optimized Parameter and Accuracy Results from IBM cloud quantum resources are tabulated.

Cloud	Parameter Value	Maximum Fidelity	Classification Accuracy
ibmq_qasm_simulator	0.39269908169872414	0.9057063636584465	100%
ibm_perth	0.39164589418564727	0.9055940024104965	100%
ibm_oslo	0.3909671490780341	0.9051565946302962	100%
ibmq_quito	0.39123400125948793	0.9054003512805725	100%

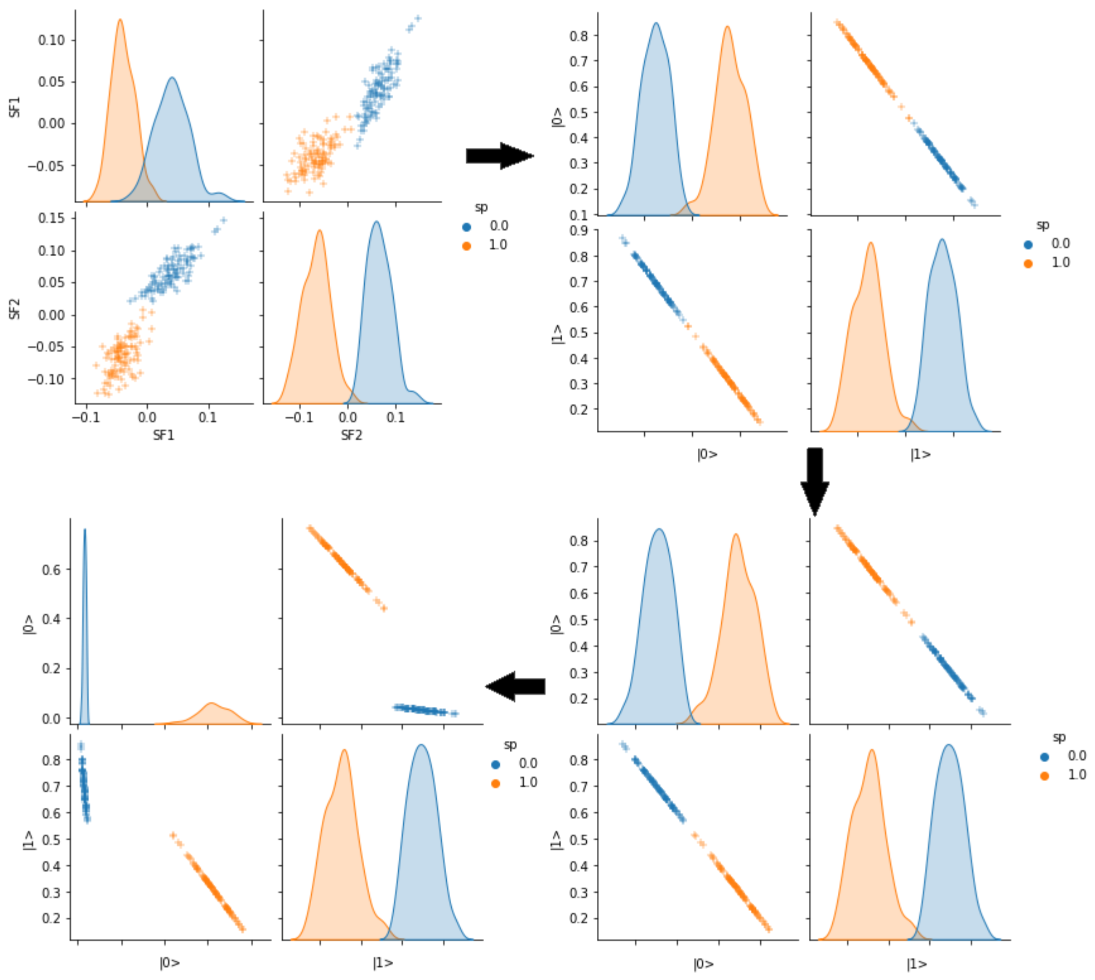


Figure 3.13: The final distribution transition

The first figure shows the distribution of the encoding circuit for the chosen features, followed by the optimized QNN circuit and the clustered target label plot.

From the transition plot (Fig. 3.13), it can be seen that during the training, the species 0 distribution is moved closer to its distribution center. This allowed the proper clustering of the two species from the statistical distance data. The parameter and fidelity levels of the cloud devices are reasonably close to those of the simulator (Table 3.7). It could be significant to note that there was relatively little device usage on the day the results for this QNN unit were acquired. The classification accuracy for all the cases remained 100%.

The data points from the two species are sorted into their appropriate groupings. A second set of QNN units is used to process sub-clusters, the gender population in each species, for classification. A second analysis of the raw data is performed in order to separate sub-clusters. The second QNN unit (blue species) is trained using the retrieved features from the PCA, ICA, and LDA plot shown in Fig. 3.14.

3.2.3 Second QNN Unit for Blue Species

The second QNN unit is used to separate the gender population in the Blue species cluster. The IC4 and LDC1 data points are multiplied by $\pi/2$ to use as input (θ and ϕ respectively) for the circuit gates.

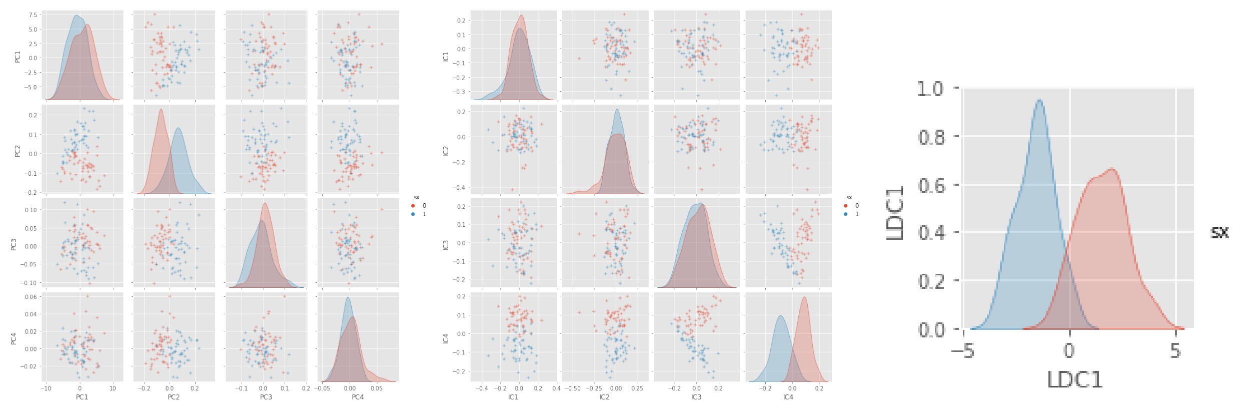


Figure 3.14: PCA, ICA, and LDA of Ripley's Crab (Blue species) Features for second QNN unit. The pairwise plot shows that IC4 and LDC1 are the best choices for features to separate the clusters as they have a lesser overlap of the clusters.

The same steps as before are repeated for the encoding circuit and the PQC layer. The final results are provided in Table 3.8.

Table 3.8: Ripley's Crab Second QNN Unit (Blue species)

The Optimized Parameter and Accuracy Results from IBM cloud quantum resources are tabulated.

Cloud	Parameter Value	Maximum Fidelity	Classification Accuracy
ibmq_qasm_simulator	0.3911143420932814	0.8883250498721621	95%
ibm_perth	0.3899032384903946	0.8882313614700156	93%
ibm_oslo	0.3923461950298759	0.8885650863233748	94%
ibmq_quito	0.3898183660348335	0.8888366600319081	95%

The extracted features from the PCA, ICA, and LDA plot given in Fig. 3.15 are used for training the second QNN unit (orange species).

3.2.4 Second QNN Unit for Orange Species

The second QNN unit is used to separate the gender population in the Orange species cluster. The IC3 and LDC1 data points are multiplied by 2π to use them as input (θ and ϕ respectively) for the circuit gates.

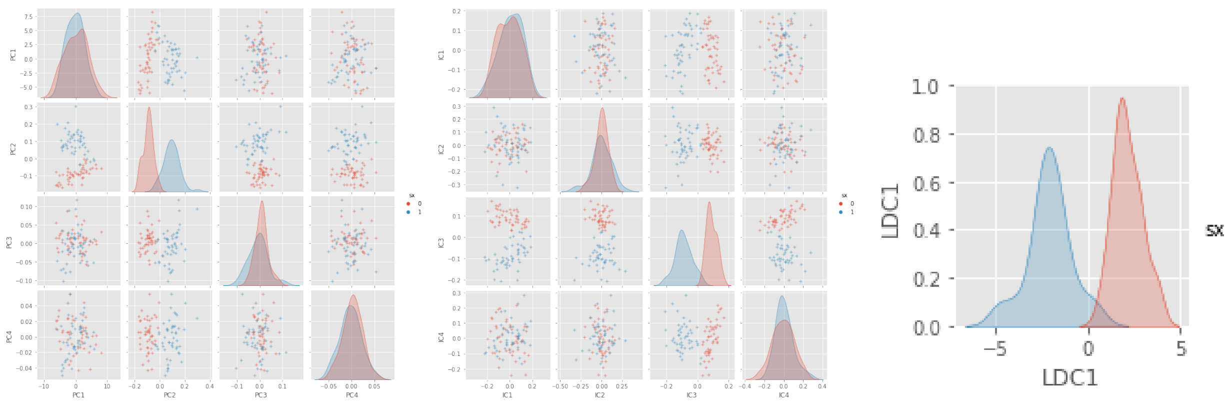


Figure 3.15: PCA, ICA, and LDA of Ripley's Crab (Orange species) Features for second QNN unit. The pairwise plot shows that IC3 and LDC1 are the best choices for features to separate the clusters.

Repeating the same steps as before for the encoding circuit and the PQC layer. The final results are provided in Table 3.9.

Table 3.9: Ripley's Crab Second QNN Unit (Orange species)

The Optimized Parameter and Accuracy Results from IBM cloud quantum resources are tabulated

Cloud	Parameter Value	Maximum Fidelity	Classification Accuracy
ibmq_qasm_simulator	0.9817477042468103	0.6608653762732526	99%
ibm_perth	0.9837888092029138	0.6611557532178729	99%
ibm_oslo	0.9834065701392277	0.6595346588059554	99%
ibmq_quito	0.9842795328356394	0.6614605093160912	100%

The complete Ripley crab classification model (Fig. 3.16) results were obtained on the same day. The wait time was very low due to the less usage of the device on that day. The fidelity and parameter values are again found to be close to the simulator outcome. It can be seen from Table 3.9 that noise fluctuations sometimes work in favor of optimization protocols.

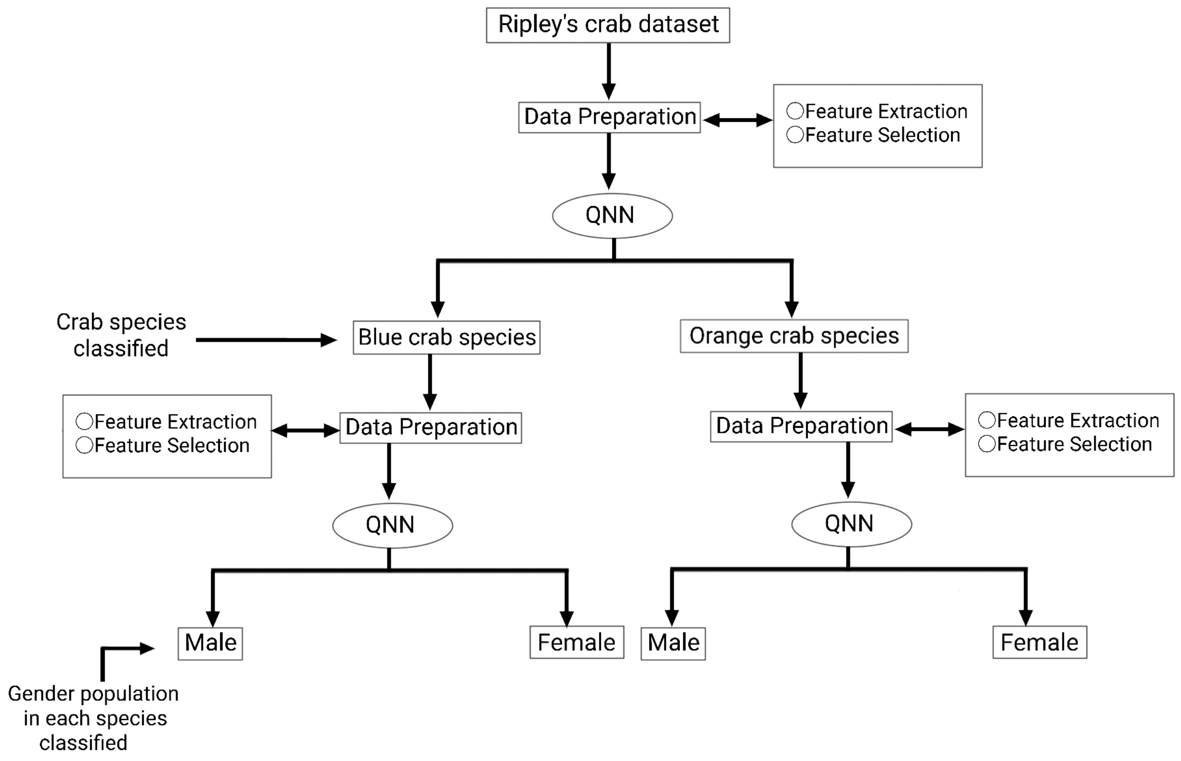


Figure 3.16: Ripley's Crab classification model.

Chapter 4

DISCUSSION

What is the proposed architectures' advantage, and is it scalable?

The hierarchical structure of classification using small QNN units proved to work for NISQ devices within a reasonable range of deviation from the simulator results. Like any other ML algorithm, the efficiency of the protocol depends on how well the data features are prepared and how well the model itself can separate the data clusters. Of course, a deep ML model will beat this approach as the classical ML protocols have been studied and refined over a few decades of research, and it would look unfair to suggest that QNNs provide an advantage. Still, drawing inspiration from classically connecting the results from small quantum ML setups can also help improve the efficiency of deep ML models for problems where the patterns are too complex for classical systems. Additionally, this model can be reliably implemented in existing NISQ devices due to the small size of the QNN unit (1 qubit) and layer depth (1 or 2 PQC layers). The small depth of the circuit helps avoid the accumulation of errors due to a large number of gates and decoherence. This architecture can be used to train with large datasets like the breast cancer dataset or the MNIST handwritten digits dataset. The proposed structure for the classification of the breast cancer dataset is provided, which can be tested along with the codes provided in the appendix. Therefore, this model can be scaled to fit a variety of different sizes of the dataset.

Constructing a QNN units hierarchical structure for big dataset.

For any dataset being used, each QNN unit consisting of the encoding circuit, the PQC layer, and measurements performed are the same as the details provided in Chapter 2 and Chapter 5. Depending on the number of features extracted from the dataset, multiple QNNs could be required in one layer of the hierarchical structure. The model's performance could depend on the number of QNN units used and the steps needed to separate the clusters of target labels. The results

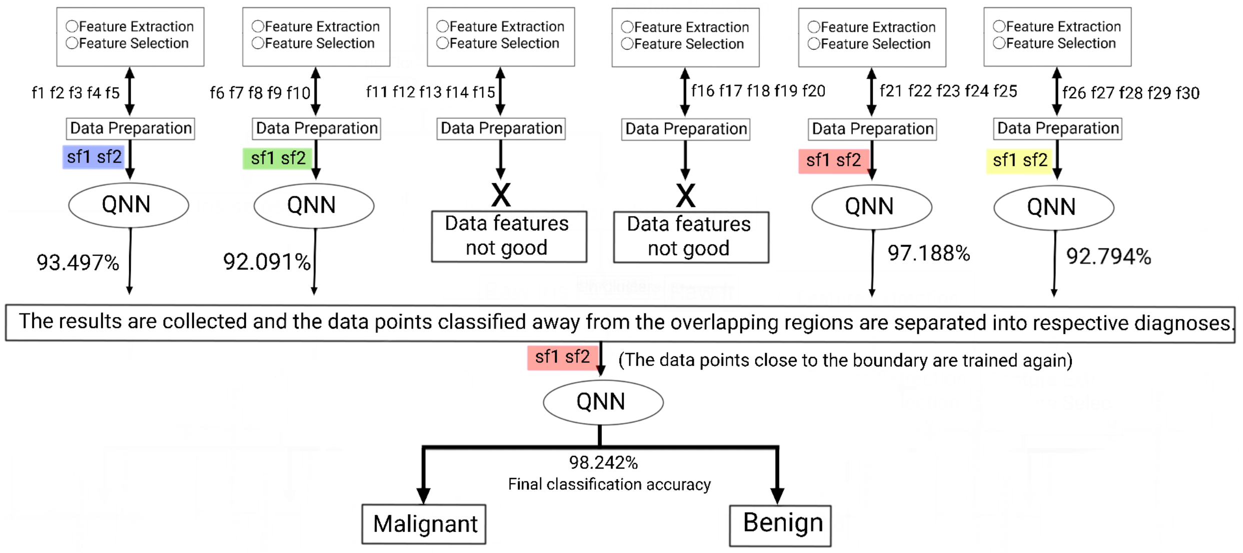


Figure 4.1: Breast Cancer dataset classification model.

f1 to f30 are original data features. sf1 and sf2 are the selected features from each subset of the original features where the colors are used to identify them differently for each QNN unit.

from these datasets can be analyzed classically to identify clusters and separate them in steps. The QNN units can be implemented one after the other (this will lead to increased run-time) or simultaneously depending on the user's available hardware resources. Each publicly available IBM 5-qubit and 7-qubit system can be used to train many single-qubit QNN units. This model can also be tested on other quantum computing systems of different companies like IonQ (trapped-ion system) or Honeywell. The QNN units will require modification tailored to the hardware like IonQ has different types of gate operators than the IBMQ gates. The PQC will also need modification accordingly while considering using a minimum number of parameters.

Current limitations due to the quantum devices.

Quantum mechanical processes like superposition and entanglement are used to create quantum computing processing devices. As a result, the qubits in these processors can hold an exponential number of states. For example, n-qubits is comparable to 2^n bits. To get an idea of the scale, 100 qubits can hold the state information of all the hard disks in the world, and 300 qubits can hold more information than the number of atoms in the universe. But, the qubits decohere to a single state due to small environmental disturbances. This means the amount of data retrieved from n-qubits cannot be more than the information obtained from n-bits. This is also referred to as the no-cloning theorem. Due to this reason, the QNN units are reconstructed multiple times to get the complete qubit tomography. The reconstructed QNNs are the same in theory. But in reality, due to varying environmental influences on the cloud system, the original state can never be replicated. Hence, qubit tomography is imperfect when measurements are carried out on cloud devices. The errors and delay by applying quantum gates add to the issue of recovering the quantum state.

The above limitations don't make quantum computing the best option for all computational problems. Quantum protocols are useful in solving problems that have very complex relations between the data elements. It makes a good candidate for traveling salesman and logistic problems, chemical and physical simulations, binary optimization, and AI and ML problems where a large amount of data needs to be processed. This leads to the question of why not use quantum simulators instead to get the work done. The quantum simulators that replicate a quantum system's behavior face the memory bottleneck problem. To simulate a n-qubit system, a 2^n dimensional complex state vector is needed. Depending on the number of qubits (n), the memory requirement can surpass the limits of classical resources (a lot of bits are required to account for entanglement, interference, and superposition behavior of qubits) needed to simulate the system.

Limitations of the proposed architecture.

The proposed architecture uses a large number of small QNN units, giving the user an advantage if they have access to a large number of small quantum processors for parallel computation. Having access to only a few can create a problem, as the user will need to implement the QNN units in sequence. Furthermore, they will need to reset the qubits between each call to the quantum processor, which may raise the overall implementation time because qubit resets can take much longer than actual gate executions. A way to overcome this problem could be to use QNN units with more qubits while keeping the number of PQC layers to the minimum. The number of measurements will increase, leading to increased measurement errors. However, measurement errors have decreased significantly in the IBM quantum devices in the past few years. On top of that, one can reduce measurement errors by applying classical post-processing. The cost and gradient calculations will become more complex but can be handled using different techniques. The advantages of less run time, reduced depth, and gate count may outweigh the additional measurement noise. The user can also implement the big QNN unit network first on simulators to test the performance before running it on the actual device.

Chapter 5

CONCLUSION AND OUTLOOK

This thesis presents a hybrid QNN architecture that can be implemented on real cloud quantum devices. Three datasets (Iris flower, Ripley's crab, and Breast cancer) and three noisy IBM devices are used for this work. The accuracy obtained from the results of the real devices of the QNN architecture is within an acceptable range of error from the simulator results. The framework can be used for developing various QML models that can withstand noise by combining several tiny quantum neural networks. It has the potential to impact the study of quantum machine learning significantly.

This study used quantum simulators from the qiskit framework to demonstrate hybrid QNN architecture for classification problems. The run time can be sped up using more sophisticated computing methods such as multi-processing units or distributed computing and allocating QNN units to multiple simulator instances. The memory bottleneck influences the proposed model less than the others due to the small number of gates implemented in the individual circuit. As a result, small QNN units may be an intriguing option for realistic QML simulation-based applications. Additional applications may include utilizing QNN units for various QML tasks (such as regression, auto-encoder, and other things) by modifying the cost functions. Further research on these areas may be fruitful. Another potential research problem could be incorporating quantum fluctuation devices for optimization and connecting the QNN units network using weights and bias of the quantum fluctuation model rather than clustering.

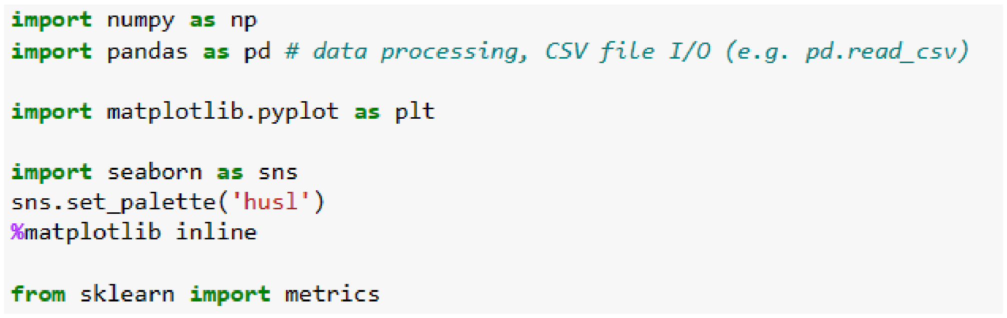
The fields of quantum computing and quantum machine learning are still young. The current generation of classical ML algorithms results from several decades of research. As a result, it may be unfair to expect QML protocols to perform better than traditional algorithms. Yet, the proposed architecture performed about as well in this investigation as conventional neural networks do. It might get even better with the proper encoding techniques, parametric layer architecture, and measurements. This is just a small step towards a general framework for building noise-resilient quantum machine learning models. In the near future, we anticipate it will have an essential role in quantum machine learning research.

All the datasets used in this study are provided in online repositories. The links to the datasets have been provided in the bibliography.

Appendix A

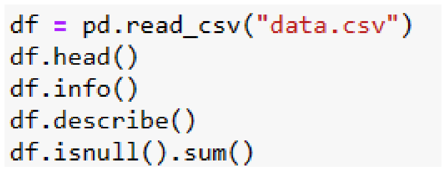
CODES

A.1 Data Preparation



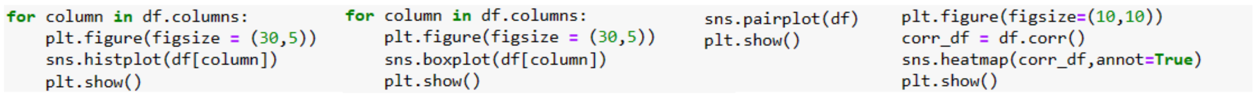
```python
import numpy as np
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

import matplotlib.pyplot as plt

import seaborn as sns
sns.set_palette('husl')
%matplotlib inline

from sklearn import metrics
```

Figure A.1: The modules required for data preparation.



```python
df = pd.read_csv("data.csv")
df.head()
df.info()
df.describe()
df.isnull().sum()
```

Figure A.2: Loading the raw dataset and information about null data points.



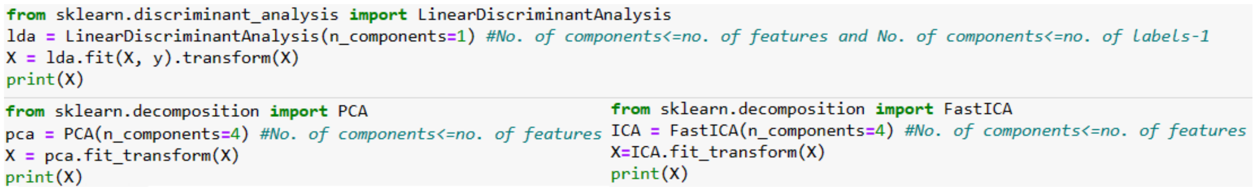
```python
for column in df.columns:
    plt.figure(figsize = (30,5))
    sns.histplot(df[column])
    plt.show()
```

```python
for column in df.columns:
    plt.figure(figsize = (30,5))
    sns.boxplot(df[column])
    plt.show()
```

```python
sns.pairplot(df)
plt.show()
```

```python
plt.figure(figsize=(10,10))
corr_df = df.corr()
sns.heatmap(corr_df,annot=True)
plt.show()
```

Figure A.3: Univariate and Bivariate analysis of the raw dataset.



```python
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
lda = LinearDiscriminantAnalysis(n_components=1) #No. of components<=no. of features and No. of components<=no. of labels-1
X = lda.fit(X, y).transform(X)
print(X)
```

```python
from sklearn.decomposition import PCA
pca = PCA(n_components=4) #No. of components<=no. of features
X = pca.fit_transform(X)
print(X)
```

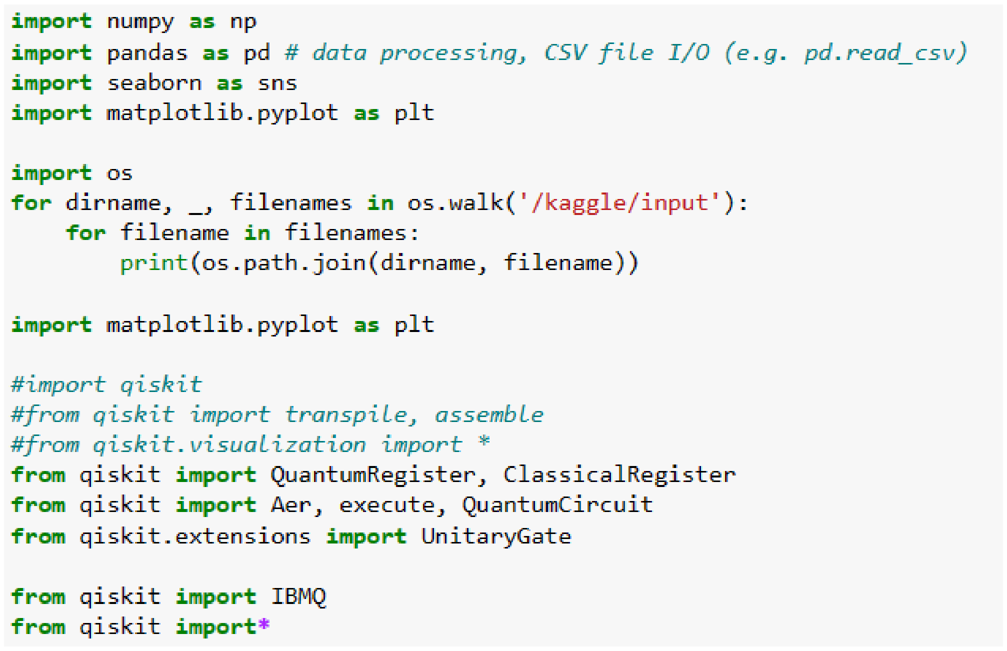
```python
from sklearn.decomposition import FastICA
ICA = FastICA(n_components=4) #No. of components<=no. of features
X=ICA.fit_transform(X)
print(X)
```

Figure A.4: Feature extraction techniques: LDA, PCA, and ICA.

The final selected features are stored in a new CSV file accessed by the QNN unit notebooks.

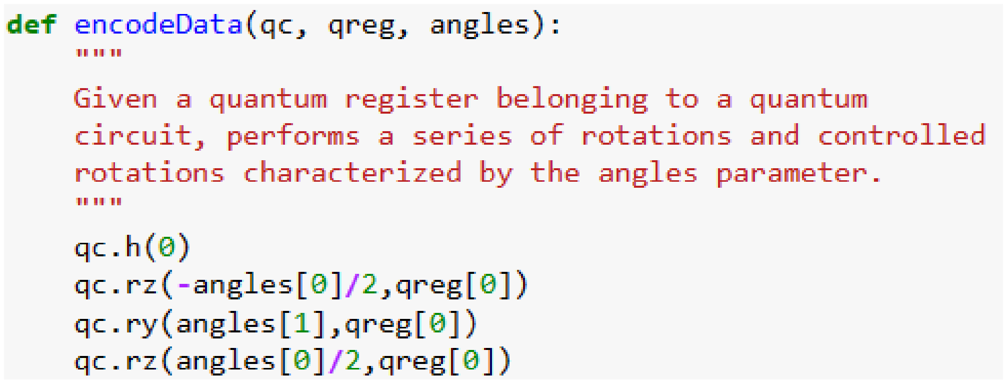
A.2 QNN unit

The codes are combined in the end in one function defined as QNN, where they are individually called with varying input data for the encoding circuit.



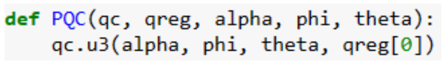
```python
import numpy as np
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

import matplotlib.pyplot as plt

#import qiskit
#from qiskit import transpile, assemble
#from qiskit.visualization import *
from qiskit import QuantumRegister, ClassicalRegister
from qiskit import Aer, execute, QuantumCircuit
from qiskit.extensions import UnitaryGate

from qiskit import IBMQ
from qiskit import*
```

Figure A.5: The modules required to run the quantum circuit.



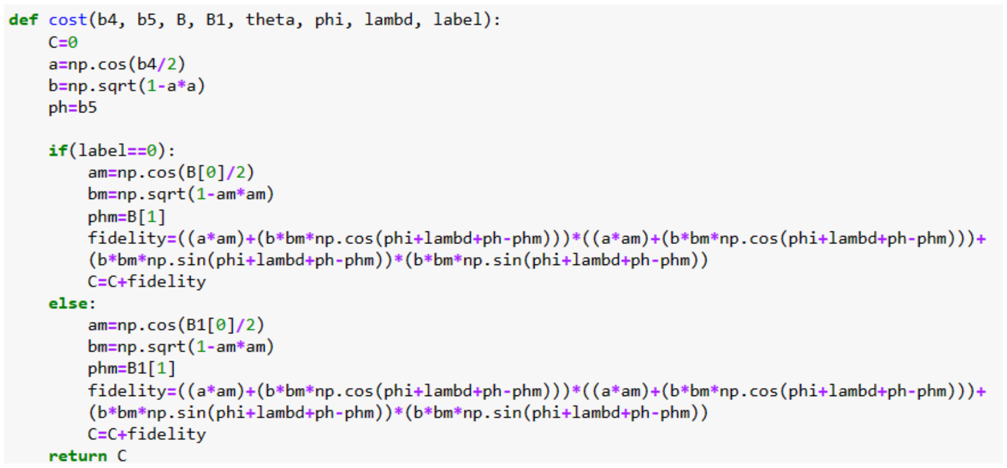
```python
def encodeData(qc, qreg, angles):
    """
    Given a quantum register belonging to a quantum
    circuit, performs a series of rotations and controlled
    rotations characterized by the angles parameter.
    """
    qc.h(0)
    qc.rz(-angles[0]/2,qreg[0])
    qc.ry(angles[1],qreg[0])
    qc.rz(angles[0]/2,qreg[0])
```

Figure A.6: Encoding Circuit.



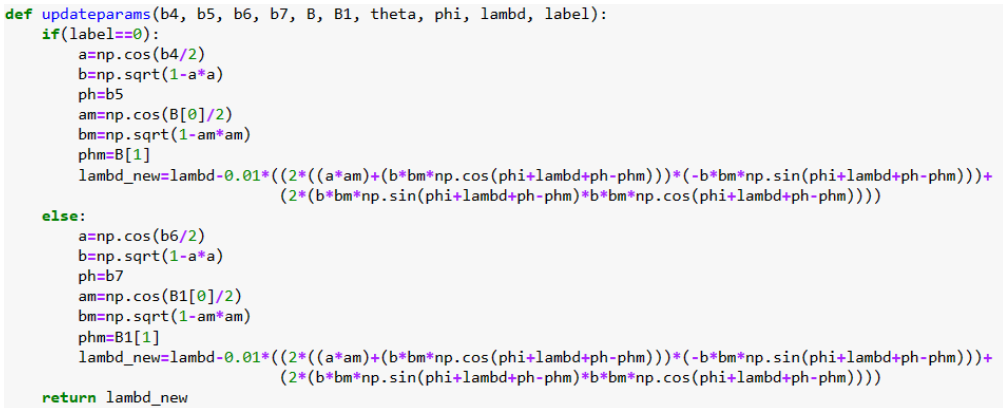
```python
def PQC(qc, qreg, alpha, phi, theta):
    qc.u3(alpha, phi, theta, qreg[0])
```

Figure A.7: PQC



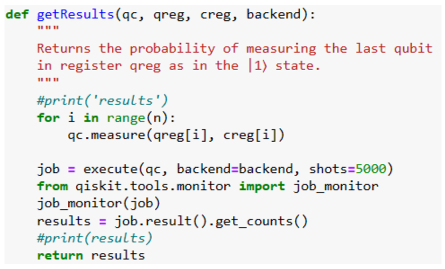
```python
def cost(b4, b5, B, B1, theta, phi, lambd, label):
    C=0
    a=np.cos(b4/2)
    b=np.sqrt(1-a*a)
    ph=b5

    if(label==0):
        am=np.cos(B[0]/2)
        bm=np.sqrt(1-am*am)
        phm=B[1]
        fidelity=((a*am)+(b*bm*np.cos(phi+lambd+ph-phm)))*((a*am)+(b*bm*np.cos(phi+lambd+ph-phm)))+
        (b*bm*np.sin(phi+lambd+ph-phm))*(b*bm*np.sin(phi+lambd+ph-phm))
        C=C+fidelity
    else:
        am=np.cos(B1[0]/2)
        bm=np.sqrt(1-am*am)
        phm=B1[1]
        fidelity=((a*am)+(b*bm*np.cos(phi+lambd+ph-phm)))*((a*am)+(b*bm*np.cos(phi+lambd+ph-phm)))+
        (b*bm*np.sin(phi+lambd+ph-phm))*(b*bm*np.sin(phi+lambd+ph-phm))
        C=C+fidelity
    return C
```

Figure A.8: Cost Function



```python
def updateparams(b4, b5, b6, b7, B, B1, theta, phi, lambd, label):
    if(label==0):
        a=np.cos(b4/2)
        b=np.sqrt(1-a*a)
        ph=b5
        am=np.cos(B[0]/2)
        bm=np.sqrt(1-am*am)
        phm=B[1]
        lambd_new=lambd-0.01*((2*((a*am)+(b*bm*np.cos(phi+lambd+ph-phm)))*(-b*bm*np.sin(phi+lambd+ph-phm)))+
                              (2*(b*bm*np.sin(phi+lambd+ph-phm)*b*bm*np.cos(phi+lambd+ph-phm))))
    else:
        a=np.cos(b6/2)
        b=np.sqrt(1-a*a)
        ph=b7
        am=np.cos(B1[0]/2)
        bm=np.sqrt(1-am*am)
        phm=B1[1]
        lambd_new=lambd-0.01*((2*((a*am)+(b*bm*np.cos(phi+lambd+ph-phm)))*(-b*bm*np.sin(phi+lambd+ph-phm)))+
                              (2*(b*bm*np.sin(phi+lambd+ph-phm)*b*bm*np.cos(phi+lambd+ph-phm))))
    return lambd_new
```

Figure A.9: Update Parameter



```python
def getResults(qc, qreg, creg, backend):
    """
    Returns the probability of measuring the last qubit
    in register qreg as in the |1⟩ state.
    """
    #print('results')
    for i in range(n):
        qc.measure(qreg[i], creg[i])

    job = execute(qc, backend=backend, shots=5000)
    from qiskit.tools.monitor import job_monitor
    job_monitor(job)
    results = job.result().get_counts()
    #print(results)
    return results
```

Figure A.10: Get Results

BIBLIOGRAPHY

[1] Nils J. Nilsson. *The Quest for Artificial Intelligence*. Cambridge University Press, July 2009. ISBN: ISBN 978-0-521-12293-1. DOI: 10.1017/CBO9780511819346.

[2] John Vincent Atanasoff. "Milestones:Atanasoff-Berry Computer, 1939." In: *IEEE Global History Network* (July 2011).

[3] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, July 2016. ISBN: ISBN 978-1-4939-3843-8.

[4] Richard P. Feynman. "Simulating physics with computers." In: *International Journal of Theoretical Physics* 21 (June 1982). URL: https://doi.org/10.1007/BF02650179.

[5] Phillip Kaye, Raymond Laflamme, and Michele Mosca. *An Introduction to Quantum Computing*. Oxford Academic, Nov. 2006. URL: https://doi.org/10.1093/oso/9780198570004.001.0001.

[6] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. "Quantum machine learning." In: *Nature* 549 (Sept. 2017). URL: https://doi.org/10.1038/nature23474.

[7] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. "The quest for a Quantum Neural Network." In: *Quantum Information Processing* 13 (Aug. 2014). URL: https://doi.org/10.1007/s11128-014-0809-8.

[8] Subhash C. Kak. "Quantum neural computing." In: *Advances in Imaging and Electron Physics* 94 (1995). URL: https://doi.org/10.1016/S1076-5670(08)70147-2.

[9] Ronald Chrisley. "Quantum Learning." In: *New directions in cognitive science: Finnish Society for Artificial Intelligence* 4 (June 1995).

[10] Chris Snijders, Uwe Matzat, and Ulf-Dietrich Reips. ""Big Data": Big Gaps of Knowledge in the Field of Internet Science." In: *International Journal of Internet Science* 7 (Feb. 2013), pp. 1–5. ISSN: ISSN 1662-5544. URL: http://www.ijis.net/ijis7_1/ijis7_1_editorial_v1.pdf.

[11] Zhih-Ahn Jia, Biao Yi, Rui Zhai, Yu-Chun Wu, Guang-Can Guo, and Guo-Ping Guo. "Quantum Neural Network States: A Brief Review of Methods and Applications." In: *Advanced Quantum Technologies* (Mar. 2019). URL: https://doi.org/10.1002/qute.201800077.

[12] Diego Ristè, Marcus P. da Silva, Colm A. Ryan, Andrew W. Cross, Antonio D. Córcoles, John A. Smolin, Jay M. Gambetta, Jerry M. Chow, and Blake R. Johnson. "Demonstration of quantum advantage in machine learning." In: *npj Quantum Information* 3 (Apr. 2017). URL: https://doi.org/10.1038/s41534-017-0017-3.

[13] Amira Abbas, David Sutter, Christa Zoufal, Aurelien Lucchi, Alessio Figalli, and Stefan Woerner. "The power of quantum neural networks." In: *Nature Computational Science* 1 (June 2021), pp. 403–409. URL: https://doi.org/10.1038/s43588-021-00084-1.

[14] Frank Leymann and Johanna Barzen. "The bitter truth about gate-based quantum algorithms in the NISQ era." In: *Quantum Science and Technology* 5 (Sept. 2020). DOI: 10.1088/2058-9565/abae7d.

[15] Kishor Bharti, Alba Cervera-Lierta, Thi Ha Kyaw, Tobias Haug, Sumner Alperin-Lea, Abhinav Anand, Matthias Degroote, Hermanni Heimonen, Jakob S. Kottmann, Tim Menke, Wai-Keong Mok, Sukin Sim, Leong-Chuan Kwek, and Alán Aspuru-Guzik. "Noisy intermediate-scale quantum algorithms." In: *Reviews of Modern Physics* 94 (Feb. 2022). DOI: 10.1103/RevModPhys.94.015004.

[16] Mahabubul Alam and Swaroop Ghosh. "QNet: A Scalable and Noise-Resilient Quantum Neural Network Architecture for Noisy Intermediate-Scale Quantum Computers." In: *Frontiers in Physics* 9 (Jan. 2022). URL: https://doi.org/10.3389/fphy.2021.755139.

[17] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning From Theory to Algorithms.* Cambridge University Press, July 2014. ISBN: 9781107298019. URL: https://doi.org/10.1017/CBO9781107298019.

[18] Nadia Burkart and Marco F. Huber. "A Survey on the Explainability of Supervised Machine Learning." In: *Journal of Artificial Intelligence Research* 70 (May 2021). URL: https://doi.org/10.1613/jair.1.12228.

[19] Or Biran and Courtenay V. Cotton. “Explanation and Justification in Machine Learning : A Survey Or.” In: *IJCAI-17 Workshop on Explainable AI (XAI)* (2017).

[20] Alexandra Chouldechova. “Fair Prediction with Disparate Impact: A Study of Bias in Recidivism Prediction Instruments.” In: *Big Data* 5 (June 2017). URL: https://doi.org/10.1089/big.2016.0047.

[21] Benjamin Schumacher. “Quantum coding.” In: *Physical Review A* 51 (Apr. 1995), pp. 2738–2747. DOI: 10.1103/PhysRevA.51.2738.

[22] Christof Koch and Klaus Hepp. “Quantum mechanics in the brain.” In: *Nature* 440 (Mar. 2006). URL: https://doi.org/10.1038/440611a.

[23] J J Hopfield. “Neural networks and physical systems with emergent collective computational abilities.” In: *PNAS* 79 (Apr. 1982). URL: https://doi.org/10.1073/pnas.79.8.2554.

[24] Dan Ventura and Tony Martinez. “Quantum Associative Memory.” In: (July 1998). URL: https://doi.org/10.48550/arXiv.quant-ph/9807053.

[25] Troels F. Rønnow, Zhihui Wang, Joshua Job, Sergio Boixo, Sergei V. Isakov, David Wecker, John M. Martinis, Daniel A. Lidar, and Matthias Troyer. “Defining and detecting quantum speedup.” In: *Science* 345 (June 2014), pp. 420–424. URL: https://www.science.org/doi/abs/10.1126/science.1252319.

[26] Yunchao Liu, Srinivasan Arunachalam, and Kristan Temme. “A rigorous and robust quantum speed-up in supervised machine learning.” In: *Nature Physics* 17 (July 2021), pp. 1013–1017. URL: https://doi.org/10.1038/s41567-021-01287-z.

[27] Andrew J. Daley, Immanuel Bloch, Christian Kokail, Stuart Flannigan, Natalie Pearson, Matthias Troyer, and Peter Zoller. “Practical quantum advantage in quantum simulation.” In: *Nature* 607 (July 2022), pp. 667–676. URL: https://doi.org/10.1038/s41586-022-04940-6.

[28] W. K. Wootters and W. H. Zurek. “A single quantum cannot be cloned.” In: *Nature* 299 (Oct. 1982), pp. 802–803. URL: https://doi.org/10.1038/299802a0.

[29] Sanjay Gupta and R.K.P. Zia. "Quantum Neural Networks." In: *Journal of Computer and System Sciences* 63 (Nov. 2001), pp. 355–383. URL: https://doi.org/10.1006/jcss.2001.1769.

[30] Diego Ristè, Marcus P. da Silva, Colm A. Ryan, Andrew W. Cross, Antonio D. Córcoles, John A. Smolin, Jay M. Gambetta, Jerry M. Chow, and Blake R. Johnson. "Demonstration of quantum advantage in machine learning." In: *npj Quantum Information* 3 (Apr. 2017). URL: https://doi.org/10.1038/s41534-017-0017-3.

[31] Wen Guan, Gabriel Perdue, Arthur Pesah, Maria Schuld, Koji Terashi, Sofia Vallecorsa, and Jean-Roch Vlimant. "Quantum machine learning in high energy physics." In: *Machine Learning: Science and Technology* 2 (Mar. 2021). DOI: 10.1088/2632-2153/abc17d.

[32] Carlo Batini and Monica Scannapieco. *Data and Information Quality.* Springer, June 2016. ISBN: 978-3-319-24104-3.

[33] Isabelle Guyon and Andre Elisseeff. "An introduction to variable and feature selection." In: *Journal of Machine Learning Research* 3 (2003), pp. 1157–1182.

[34] Verónica Bolón-Canedo, Noelia Sánchez-Maroño, and Amparo Alonso-Betanzos. *Feature Selection for High-Dimensional Data.* Springer, Aug. 2015. ISBN: 978-3-319-36643-2.

[35] Roman Schmied. "Quantum state tomography of a single qubit: comparison of methods." In: *Journal of Modern Optics* 63:18 (2016), pp. 1744–1758. URL: https://doi.org/10.1080/09500340.2016.1142018.

[36] Samuel L. Braunstein and Carlton M. Caves. "Statistical distance and the geometry of quantum states." In: *Physical Review Letters* 72 (May 1994), pp. 3439–3443. URL: https://doi.org/10.1103/PhysRevLett.72.3439.

[37] R. A. Fisher. "The use of Multiple Measurements in Taxonomic Problems." In: *Annals of Eugenics* 7 (1936), pp. 179–188. URL: https://doi.org/10.1111/j.1469-1809.1936.tb02137.x.

[38] Brian D. Ripley. *Pattern Recognition and Neural Networks.* Cambridge University Press, 1996. ISBN: 9780511812651.