Alma Mater Studiorum · University of Bologna

School of Science
Department of Physics and Astronomy
Master Degree in Physics

Quantum Neural Networks for Data-Efficient Image Classification

Supervisor:

Prof. Stefano Lodi

Submitted by:

Francesco Aldo Venturelli

Co-supervisors:

Dr. Antonio Macaluso,
Dr. Riccardo Mengoni.

Academic Year 2022/2023

Abstract

In our constantly evolving world, an overwhelming influx of data permeates every moment—be it daily, hourly, or even by the second. We communicate, share links, images, and opinions, disseminating a trail of traces, representing not just the vastness of our natural surroundings but also reflecting our thoughts, preferences, and sentiments. Recognizing the significance of these data, the field of Data Science has emerged, dedicated to unveiling the concealed insights embedded within. Machine Learning (ML) has become a captivating realm of research[8], gaining prominence for its capacity to extract knowledge from extensive datasets[20]. ML has played a pivotal role in bridging the gap between our understanding of nature and its intricacies. Deep Learning (DL), particularly Neural Networks (NNs), has revolutionized classical ML, serving as non-linear structures for modeling statistical data[23]. NNs, and notably convolutional neural networks (CNNs), simulate intricate relationships between inputs and outputs[8], excelling at tasks such as image-based pattern recognition inspired by the structure of the visual cortex. While NNs, especially multilayered ones, have demonstrated remarkable power, their trainability posed challenges. The advent of back-propagation mitigated this issue, but training difficulties persisted, necessitating solutions like rectifier neuron activation functions and layer-wise training. Quantum Machine Learning (QML) has introduced new avenues, leveraging noisy intermediate-scale quantum computers for computational problems involving quantum data. Variational quantum algorithms (VQAs) and quantum neural networks (QNNs) offer promising applications, utilizing classical optimizers to train parameters in a quantum circuit. QNNs present a distinctive advantage over classical models by analyzing systems with polynomial complexity[2][6], which would be exponentially complex in classical ML, providing a computational edge. Notably, QNNs exhibit faster learning capabilities compared to classical counterparts, attributed to the entanglement discussed later in chapters 1 and A. Previous studies highlight the efficacy of QNNs in learning from limited data, reducing time and energy in training processes. This Master thesis delves into efficient image classification possibilities using various quantum models trained with minimal images, concluding with a direct comparison against classical CNN performance. Two diverse datasets are employed for training, subsequently scaled down to explore the QNN models' potential to predict more images than CNNs.

Contents

Chapter 1

Introduction

Motivation

Machine Learning (ML) is widely used in several fields of research nowadays. Even if it was held by the most renowned companies or research centers at the beginning, people now have access to an enormous quantity of Generative AI models that can help in developing their activities such as writing a paper, generating weird images, coding, or even taking decisions over some event. ML has completely changed the way people approach problems in Science. Before his arrival, scientists usually faced a problem and, based on the data collected, made hypothesis to explain the observed phenomena, built a mathematical model that could have been representative of it, and subsequently translated the model to a computer to perform simulations to get the answers they were looking for. The primary challenge with this mode of interaction arises when the initial hypothesis is incorrect, rendering the simulation meaningless and yielding inconclusive results. The significance of the initial guess, or hypothesis, is evident in this particular approach. However, a notable shift has occurred since the inception of ML. Nowadays, there has been a turnaround: a ML model is employed to ingest and train on the data, aiming to unveil the inherent rules or hypotheses governing the underlying phenomena. In this sense, it is a process that goes backward: from data to hypothesis/rules. If in the first approach, the balance leans much more on the hypothesis humans guess on a particular phenomena, for the second one the data has greater weight in the analysis and it is the only element that reveals and explains the insights behind the event.
ML is generally composed of three main parts: the starting point is the data collection, in which practitioners prepare, clean, and organize plenty of data that will be used by a model to reveal the inner logic behind them. In the second step the model is trained, i.e. it starts to classify as many training data-points as possible to learn the intrinsic bounds in common. Ultimately, the model is evaluated on a new set of data that it hasn't seen yet: the test data. The testing process is probably the most representative part of ML, where the model has to be able to predict new data based on what has

been learned previously. Already from this point, we can wonder how much data are necessary for the model to reach a high level of accuracy during the prediction. As we may think, good performances are usually accompanied by a long training phase. In fact, if we think about a classification problem with two different classes of items (as it has been done in this project) where the goal is to classify which item belongs to which class, it makes sense that the more images we provide the model, the more it's able to recognize details representative to each class. The development of ML, in general, would be unimaginable without the invention of the modern computer[8]. Living in the age

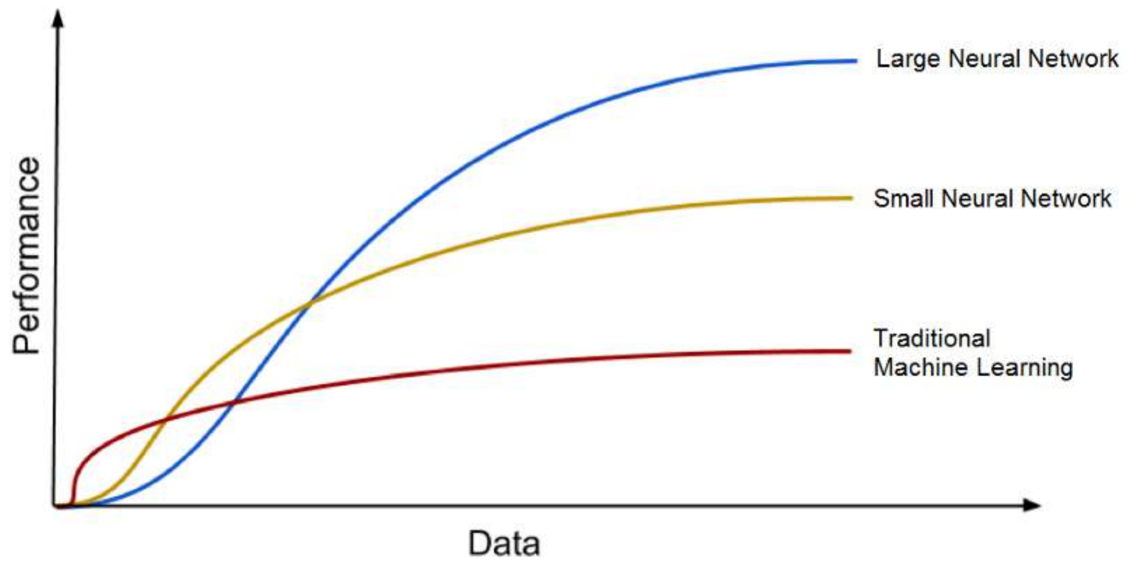


Figure 1.1: ML performances over how much data they require. It's visible that more and more data are essential for good predictions and the amount of memory used to store these data and the number of processors that can execute the task should increase over time.

of laptops, smartphones, and smartwatches and constantly observing the arrival of new technologies can give the expectation of endless growing computational potential[8].
As we have pointed out, Machine Learning is everywhere. Biology, Medicine, High-Tech industries, Physics, Engineering, etc. are all sectors that have experienced a rapid growth in the last years thanks to the spread of Artificial Intelligence (AI). The most updated and powerful ML models used these days require long time for being trained, energy, and enormous computational resources due to the complexity of the problem, which usually cannot be provided from our everyday computers, instead, they are furnished most likely by super-computing centers. The idea of looking for something different arises directly from this point: it's very expensive to wait for a long time and to make use of supercomputers, GPUs, hardware accelerators, and other advanced architectures to obtain slightly

better accuracy compared to simple models. The demand for new device architectures and information processing methods is even more motivated through the exponentially increasing amount of data created every day[8]. This is in fact, one of the reasons why the innovative quantum models can come out as a possible way of solving, or reducing the current problem.
With the advent of Quantum Mechanics, new technologies such as quantum models and quantum computers are under investigation and are proposed as innovative devices that offer a possibility of addressing open existing problems for classical computers. In this context, one of the main area where quantum computers promise to give their contribution to reach a significant speed-up is the combinatorial and optimization set of problems. In this set the power of qubits to be in a superposition of states (and to have a probability amplitude associated to those states) permits to scale with the system's size of the given problem with the result of finding a solution with lower computational costs. Additionally, they are able to perform computations faster and reach better results than classical devices on specific problems and many quantum algorithms have shown to break down widely used classical encryption protocols. Such algorithms demonstrated mathematically that one could gain more information by querying a black box with a quantum state in superposition, sometimes referred to as "quantum parallelism".
In principle, a classical computer can solve the same computational problems as a quantum computer, given enough time.
Quantum advantage, as we said, comes in the form of time complexity rather than computability, and quantum complexity theory shows that some quantum algorithms, for carefully selected tasks, require exponentially fewer computational steps than the best known non-quantum algorithms. Such protocols can, in theory, be solved on a large-scale quantum computer whereas classical computers would not finish computations in any reasonable amount of time. Quantum computers, containing up to hundreds of quantum bits, became experimentally realizable in the last years and give the opportunities to exploit the laws of Quantum Mechanics to avoid the limits of classical computing[8]. They offer the promise of dramatically improving ML through speed-ups in computation and improved model scalability[2] (ML models can handle increasing amounts of data and perform many computations in a cost-effective and time-saving way). In the field of ML, the capacity of a model relates to its ability to express a variety of functions. The higher a model's capacity, the more functions it can fit[1]. By introducing the concept of generalization as the way a model can learn from data and generalize unseen data[2], many capacity measures have been shown to mathematically bind the error a model makes when performing a task on new data. With the introduction of the global effective dimension [1] presented for the first time in [1], authors have tried to set up an indicator of how well a particular model will be able to perform predictions and express new data. As they remarked in the article, quantum neural networks can achieve a significantly better effective dimension than comparable classical neural networks that results in a wider capacity of fitting the data[2]. Indeed, certain QNNs can

train faster than classical models due to their favorable optimization landscapes, captured by a more evenly spread Fisher information spectrum[2], which they used in their experiments. From this point of view, it is evident that QNNs, together with quantum computers, can break down the horizons hitherto known.
Another principal characteristic that distinguishes quantum from classical models is intrinsically contained in Quantum Mechanics: the entanglement. It is a pure quantum effect, not present in classical nature, responsible for correlations between quantum systems, or quantum data, even at long distances. When two particles, such as a pair of photons or electrons (or qubits), become entangled, they remain connected even when placed at infinity one from the other. In Section 2.2 of Chapter 2, a concise definition is presented regarding the conditions specified by the superposition principle in Quantum Mechanics. In scenarios where two or more physical systems act as subsystems within a larger one, and the quantum state of the overall system is expressed as a combination of their individual states, the measurement of an observable for one system (subsystem) simultaneously establishes the value of the same observable for the others. Since the state of quantum superposition is independent of a spatial separation of such systems (subsystems), entanglement counter-intuitively implies the presence of distance correlations between them and, consequently, the non-local character of physical reality[8]. In the field of Quantum Computing, the entanglement is realized between qubits and quantum information is shared among entangled qubits. The ability to make connections between the data is exploited by models simulated on a quantum computer to acquire deeper and finer information about the data themselves and how they are connected; classical models instead, cannot make use of entanglement.
It's clear that we are in front of two separated worlds, generated by different theoretical aspects and that will give different results. From now on, the project will introduce the fundamentals of Quantum Mechanics from the basics to the QNN models. Before that, we are going to spend few words on the idea behind this work, explaining our idea, what we would like to test, what we would expect and the experiments we are going to illustrate. Eventually, we will provide specific results about the experiments we carried out, focusing on the ability of QNN to recognize images compared with an over-studied classical CNN.

Objectives

This thesis seeks to illuminate the advantages derived from the generalization process when training quantum models with minimal data. By subjecting various quantum architectures to testing in a supervised binary problem within Computer Vision, our goal is to discern the potential advantages stemming from the principles of Quantum Mechanics1. To rigorously challenge the capabilities of these models, we deliberately reduced the training dataset to a scant number of instances (6, 10, 20, 30, 40, and 50). This intentional reduction in data aims to assess the resilience and efficiency of quantum models when confronted with sparse training datasets. The evaluation process involves comparing the performance of these quantum models with that of a classical counterpart, represented by a CNN. The objective is not to anticipate a clear superiority of QNNs over the CNN employed; instead, we are seeking to identify a particular regime associated with "Data-efficiency." In this context, Data-efficient implies achieving robust predictive capabilities while utilizing a minimal number of images and features, thereby diminishing the computational resources required for model training. Our focus is on exploring situations where quantum models exhibit a relative advantage in managing and learning from sparse datasets. The subsequent section provides a more detailed breakdown of the analysis conducted to delve into these aspects.

Contributions

During the realization of the thesis, we have tried to contribute to the vast research activities that involve quantum models, aiming to illustrate that such original devices provide a new way of performing image classification, especially when we have few data and features. We started from an existing work that has shown a little tutorial on binary image classification where only a selected quantum circuit has been tested. We expanded the current work putting more attention on testing different quantum architectures that emulate the QNN and also a quantum convolutional neural network (QCNN) inspired by a similar architecture proposed in [12]. Thus, we built a simple and fair classical neural network for image classification to have a direct comparison to the quantum algorithms. We have followed the starting work by adapting our different models to a real dataset, but we have also introduced a standard toy-dataset as Mnist, appropriately reduced in the number of features to be encoded by the feature map circuit that we describe in the section 2.2.4.
The idea behind the project is to explore the performances of quantum models based on what has been realized in [10] where has been revealed that a quantum model can learn and generalize faster, compared to a classical algorithm, with less training (quantum) data-points. This last statement is investigated in the thesis and applied to classical data images, to include the state preparation process that characterizes and differentiates any quantum model concerning its classical counterpart. The "modus operandi" consisted of building five different sets of a selected number of training images, encoding the classical data into their corresponding quantum states, training multiple variational ansatz circuits, and making the binary classification over a fixed test set, by using the number of corrected predictions over the total number of predictions as a metric. After collecting the respective accuracy on the training and test set, the number of images is raised and the same process has been applied for a total number of six training sets containing few images (of the order of tens). The predictions are then plotted by varying the number of training data the model has seen in each iteration. We have managed to see important results about the generalization abilities of each quantum model. As we were expect-

ing, the systematic reduction of the over-fitting is reached when more images are used by the model for training. In other words, as the reader will have the opportunity to notice, the more images are seen by the QNN, the less the distance between the train and test accuracy curve will be, until almost an overlapping between the two. Besides this foregone result, also difficulties came out. The selected quantum models are still far away from the excellent performance of the classical CNN, meaning that probably some classical adjustment could have been done at the beginning, for example tuning some hyper-parameters as the learning rate (that has kept fixed in our simulations), using a different classical optimizer, introducing dimensionality reduction techniques, increasing the number of epochs used within the training, etc. The intent here is not to demonstrate if quantum neural networks can overcome classical neural networks in the performances in principle, rather we are interested to see how faster the generalization can be realized by those quantum models. Anyway, we are forced to compare the learning abilities concerning classical models and we have to admit that is not completely possible to uphold the thesis supported in [10] so far, since the introduction of the embedding circuit, responsible to turn classical data into quantum states, in the model is not present there. Quantum computing tells us that adding qubits and quantum gates to our circuit, as well as being expensive to simulate, will generate errors that can be traduced into the impossibility of a QNN model to learn the best parameters for the classification. To tackle this issue, one possible approach could be the reduction of the number of features of the image to feed the model with, to decrease the number of quantum operations (quantum gates) necessary to encode each pixel. Along this way we have proposed another experiment that reveals interesting facts. This new task aims to reproduce the accuracy performances for selected training sets where the number of features is considerably lowered. In a certain sense, we challenged quantum and classical models by making the images worse. As we will display later in 5.2, for some reduced-in-features training sets, the quantum models work better than the classical ones. This can be another viewpoint of the same thesis supported in the article[10]. In this sense, we can observe if we are able to obtain some improvements with an efficient way of providing data to quantum models over classical. Eventually, an additional task has been set up and it wants to reproduce another comprehensive result: the more parameters are added in the ansatz, i.e. more layers the circuit is composed by, the more the accuracy increases.

Chapter 2

Background

2.1 Elements of Machine Learning

Machine learning is a branch of Artificial Intelligence and Computer Science that focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy. As already explained in the 1, it's a new way of approaching problems, from the data we can measure, back to the rules that govern the phenomena itself. Through the use of statistical methods, algorithms are trained to make classifications or predictions and to uncover key insights in data mining projects. These insights subsequently drive decision-making within applications and businesses, ideally impacting key growth metrics. As big data continues to expand and grow, Machine Learning techniques are increasingly in demand to reveal the links that join such data. The most used algorithms in this field can be divided into three main groups:

Supervised learning;

Unsupervised learning;

Reinforcement learning.

2.1.1 Supervised Learning

In SL, input objects (for example, a vector of predictor variables) and a desired output value (also known as human-labeled supervisory signal) train a model. The training data is processed, building a function that maps new data on expected output values. An optimal scenario will allow for the algorithm to correctly determine output values for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way. This statistical quality of an algorithm is measured through the so-called generalization error. By introducing some Mathematics in the definition of SL, given a set of N training examples of the form $\{(x_1, y_1), ..., (x_N, y_N)\}$ such that x_i is the feature vector of the i-th example and y_i is its label (i.e., class), a learning algorithm seeks a function $g : X \rightarrow Y$, where X is the **input space** and Y is the **output space**. The function g is an element of some space of possible functions G, usually called the **hypothesis space**. It is sometimes convenient to represent g using a scoring function $f : X \times Y \rightarrow \mathbb{R}$ such that g is defined as returning the y value that gives the highest score: $g(x) = argmax_y f(x, y)$. SL is a part of the Computer Vision domain, where scientists train several ML models with train images and the corresponding labels and test them on new images asking the model to attach them at each image the correct label. The fraction of images classified correctly gives the accuracy of the model that should come closer to 100% ideally.

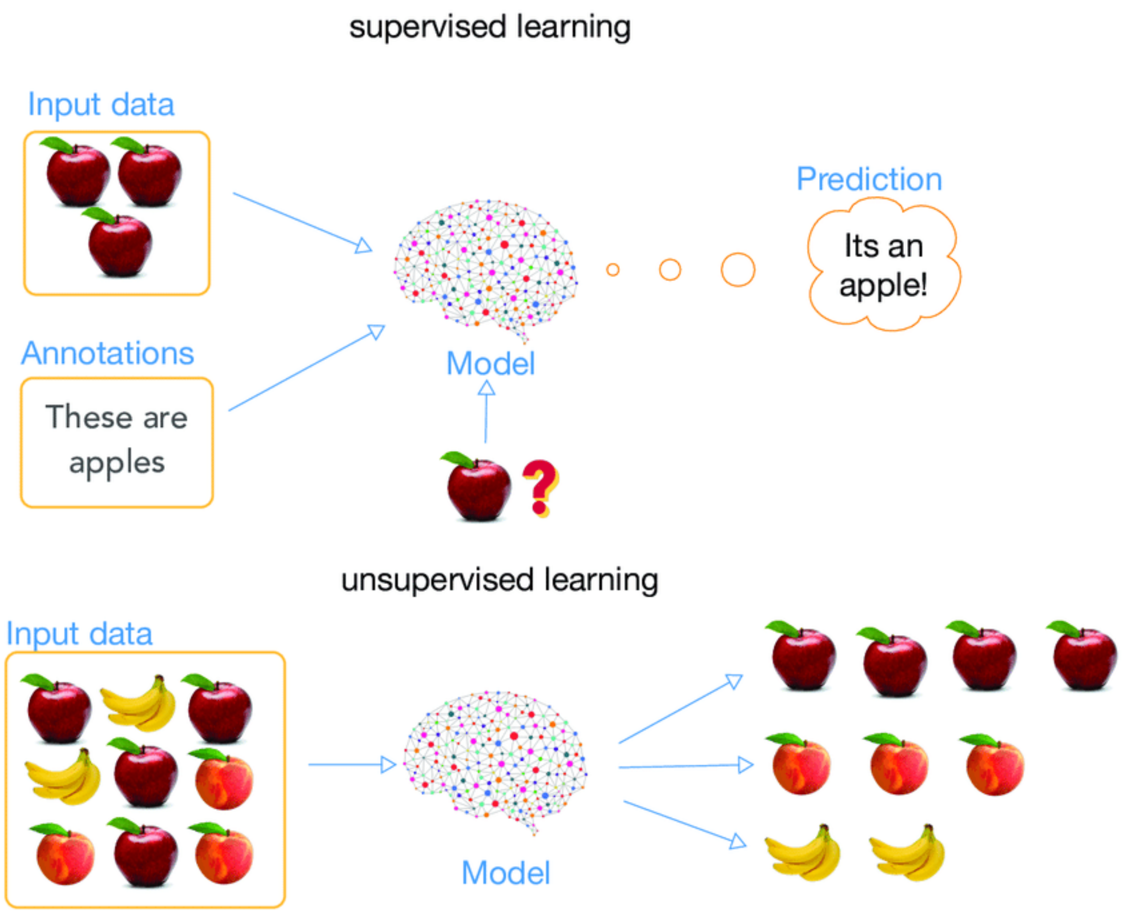


Figure 2.1: Example of SL vs UL.

2.1.2 Unsupervised Learning

In Unsupervised Learning (UL) no labels are given to the learning algorithm, leaving it on its own to find structure in its input. In some pattern recognition problems, the training data consists of a set of input vectors $\vec{x}$ without any corresponding target values. The goal in such UL problems may be to discover groups of similar examples within the data, which is called clustering, or to determine how the data is distributed in the space, known as density estimation. To put forward in simpler terms, for a n-sampled space $\{x_1,..., x_n\}$, true class labels are not provided for each sample, and is the algorithm itself to recover some logic in the data. UL is mostly used in the field of Data Mining where users don't know, a-priori, what classes or subgroups the data is divided into, or where it wouldn't be possible to annotate large datasets by hand.

2.1.3 Reinforcement learning

Reinforcement Learning (RL) is an area of Machine Learning concerned with how intelligent agents ought to take actions in an environment to maximize the notion of cumulative reward. RL differs from SL in not needing labeled input/output pairs to be presented, and in not needing sub-optimal actions to be explicitly corrected. Instead, the focus is on finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge).

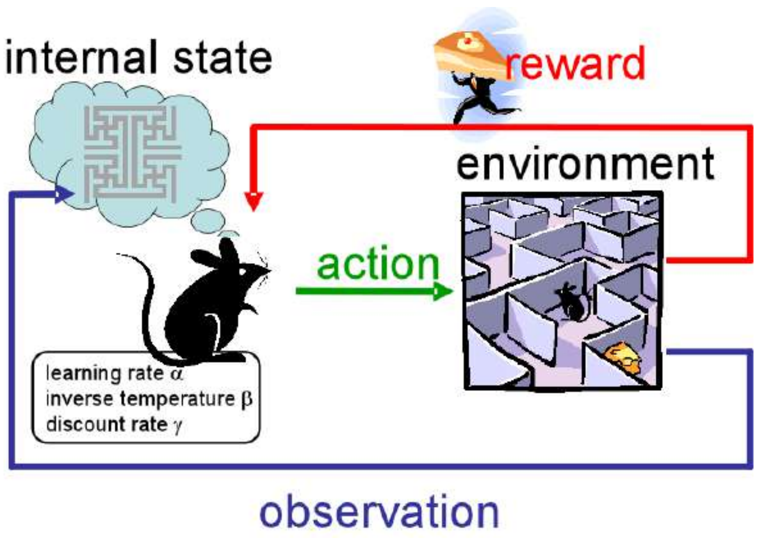


Figure 2.2: Example of Reinforcement Learning: a mouse has to find the piece of cheese missing in a libyrinth. Once the mouse chooses a way, a positive or negative feedback will be sent to him in the case it has guessed the corrected path or not.

2.1.4 Neural Networks and Deep Learning

Neural Networks

For all vertebrate animals, the brain is the nervous system's centre. This complex organ is built of billions of fundamental units, referred to as neurons. The connection of one of these building blocks to others through so-called synapses allows interactions. Whereas the connections within smaller groups of neurons can be recorded, studying the communication between a larger population of these units is very tough. In the middle of the 20th century, the first computational models for neural networks were proposed. Based on these ideas, artificial neural networks arose and could be set-up on the available electronic computers[8] at that time. The perceptron was the first artificial neural network with complex adaptive behaviour. Versions of this building block are used for NN architectures still today, and referred to as artificial neurons. A neuron takes n inputs $x_1, ..., x_n$ and has a single binary output y, also called activation. Every input has an assigned weight $w_i \in \mathbf{R}$. Additionally, the neuron is equipped with a bias $b \in \mathbf{R}$. The neuron's output is computed through

$$y = \kappa(\sum_i w_i x_i + b) \tag{2.1}$$

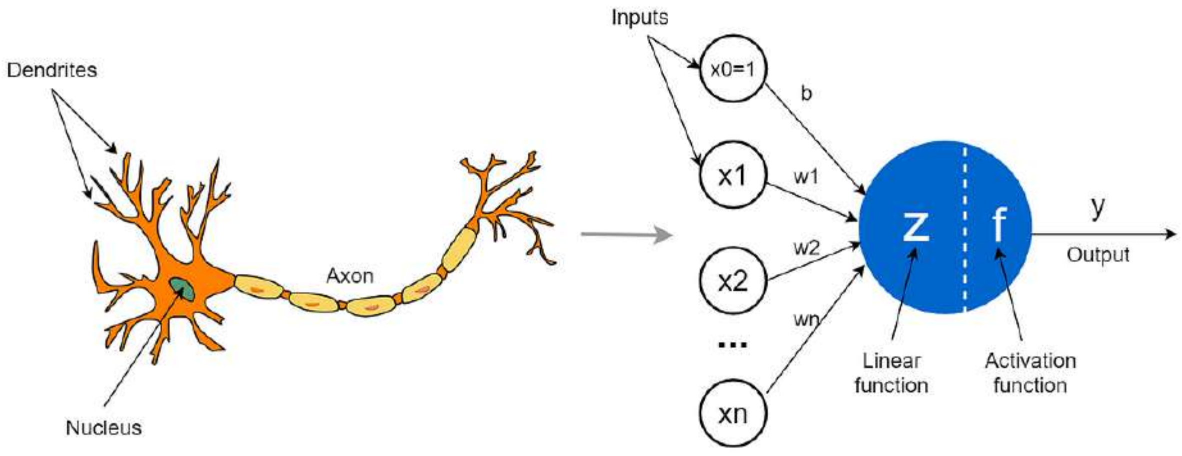


Figure 2.3: Comparison between a biological brain neuron and an artificial neuron.

where κ denotes an activation function. To intuitively understand the method of a neuron, we can imagine the perceptron's task as deciding between two choices, 0 and 1. The inputs x_i can be seen as arguments with different importances w_i, where an argument with $w_i x_i < 0$ is pro-choice 0 and $w_i x_i > 0$ is pro-choice 1, respectively. The activation function and the bias b describe a threshold. Depending on which side of the threshold the weighted sum $\sum_i w_i x_i$ of the arguments is, the perceptron "decides" for the output 0 or 1. In the context of NNs, where neurons are layered, we can say that the activation function, as the word says, activates the neuron if the input is above a certain threshold[8]. It' evident that choosing the activation functions wisely is crucial for good training results since the activation functions decide if an input of a neuron is relevant. After the first enthusiasm about perceptrons, it got rapidly clear that these one-layer NNs were quite limited in computational power[8]. It was discovered that stacking these early artificial neurons in layers increases the performances. Whereas with only one layer the learning of linearly separable classes can be performed, with multi-layer perceptrons also non-linear classification problems can be tackled and solved. A generic Neural Network's architecture, constituted by multiple layers, depicted in 2.4

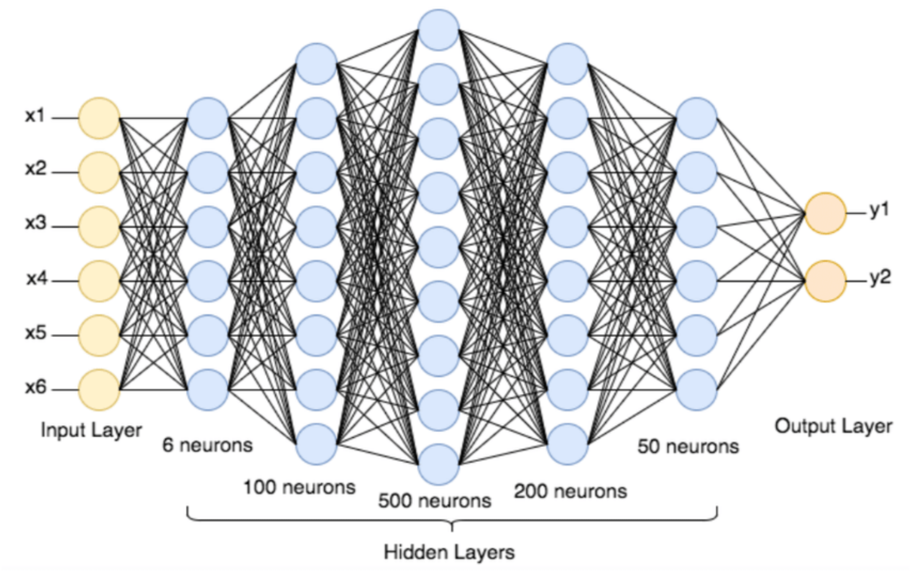


Figure 2.4: Multi-layers perceptron scheme.

The first layer of neurons, which get the initial input, are called the input layer, the last layer of neurons is named output layer, while the layers in between are called hidden layers. While working through a considerable amount of the layers, the original information gets more and more abstract, and in that way, the complex data processing is divided into a series of simple nested assignments. The simplest NN architecture can be found in feed-forward neural networks. The neurons get the output of previous layers as inputs, and no loops are built-in. Furthermore, such NNs are often built of fully connected layers, i.e. where all the inputs from one layer are connected to every neuron of the next layer. Below we present some of the most famous and used activation functions that introduce the non-linearity within the networks.

$$\textbf{Sigmoid} = \frac{1}{1+e^{-x}}; \tag{2.2}$$

$$\textbf{ReLU} = max(0, x); \tag{2.3}$$

$$\textbf{Leaky-ReLU} = max(\frac{1}{10}x, x); \tag{2.4}$$

$$\textbf{TanH} = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{2.5}$$

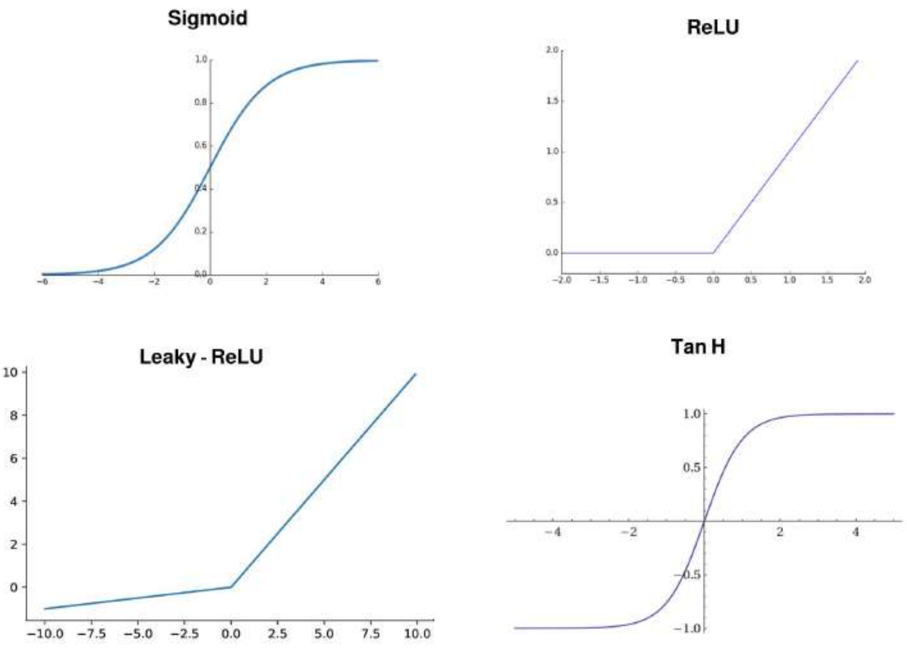


Figure 2.5: Sigmoid, ReLU, Leaky-ReLU and the Tanh activation functions.

Neural Networks are particular architectures organized by layers, that emulate the human brain's way of learning. These non-linear models receive data as input and process them, inside each neuron placed on a layer. The "signal" at a connection is a real number, and the output of each neuron is computed by some non-linear function of the sum of its inputs. Such purely mathematical tools constituted a turning point in Computer Science since they introduced the non-linearity contained within data. It means that NNs can successfully approximate functions that do not follow linearity or it can successfully predict the class of a function that is divided by a decision boundary which is not linear (for example a logistic-regression (LoR)). Deep[1] Learning (DL) models are evolutions of NNs where more hidden layers are added inside the architecture with the result of increasing the possibility of representing more complex data, as they have many more non-linear functions to work with. DL is part of a broader family of Machine Learning methods, which is based on NNs. DL architectures such as Deep Neural Networks (DNNs), Deep Belief Networks (DBNs), Deep Reinforcement Learning (DRL), Recurrent Neural Networks (RNNs), Convolutional Neural Networks (CNNs) and Transformers have been applied to fields including Computer Vision, Speech Recog-

[1]The adjective **"Deep"** refers to the use of multiple layers within the network structure of the model, building increasingly sophisticated models.

nition, Natural Language Processing (NLP), Bioinformatics, drug design, medical image analysis, climate science, material inspection and board game programs, like Chess, where they have produced extraordinary results surpassing human expert performances. NNs were inspired by information processing and distributed communication nodes in biological systems, where the information passes through each node and subsequently is processed until reaches the final step, where the classification is made [8]. In the SL setting (predicting y from the input x), models like LoR and Linear Regression (LiR) are linear in the number of parameters θ that are guessed to obtain the best fit of the data points. In DL those models instead are non-linear in both the parameters θ and the inputs x. Suppose $\{(x_i, y_i)\}_{i=1}^{n}$ are the training examples. The parameters θ are the learning weights the model should guess to make good predictions over the data. After the first inference, it usually happens that the learnt parameters θ couldn't be the best possible ones that ensure no error in the classification. In fact, algorithms such as NNs and CNNs work iteratively: the learning parameters are updated and re-calculated at each step, or iteration, as result of a minimization of a specific cost function, i.e. loss function. The choice of the cost (or loss) function belongs to the set of tunable hyper-parameters that can be modified in order to look for improvements in the generalization task. As mentioned earlier, the loss function serves as a measure to assess the error computed by the model. When the loss function is defined as the mean squared error (MSE), it takes a specific form for the i-th data point and its corresponding label, denoted as (x_i, y_i). The MSE loss function is denoted by the following equation:

$$J_i(\theta) = \frac{1}{2}(h_\theta(x_i) - y_i)^2. \tag{2.6}$$

h_θ represents an equation that characterizes the model and makes predictions based on the parameters θ. This specific cost function is suitable for the LiR problem. Squaring this difference serves two primary purposes: it penalizes larger errors more significantly, and it ensures that all errors contribute positively to the overall loss. In summary, this mean squared error loss function quantifies the squared difference between the model's prediction and the actual label for a given data point, providing a measure of how well or poorly the model is performing, but others can be considered for more sophisticated experiments. In fact, additional cost functions are more likely for the NNs, taking into account the non-linearity of the model as the Cross-Entropy, the Likelihood, the Negative Log Likelihood function, etc. The minimization of the cost functions is performed by the so-called optimizers. They are basically used to update weights and biases, the internal parameters of a model, to reduce the error computed by the model. We anticipate that either in the quantum algorithms we have used to make the experiment, the optimization is done classically. The most important technique used to make the optimization is the Gradient Descent method. As it can be seen from the curve above, there exists a value of parameters w which has the minimum value of the cost function, J_{min}. In the Gradient Descent algorithm, in order to find the minimum, we give the model random parameters

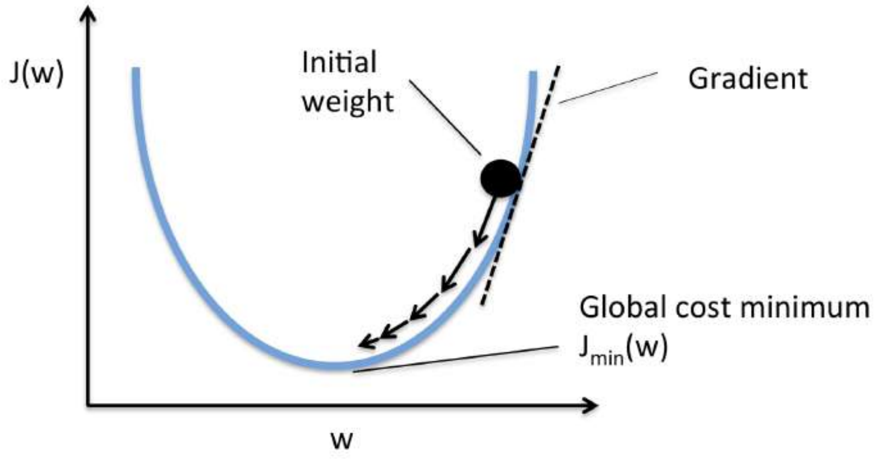


Figure 2.6: Illustrative picture of how GD works.

to compute the error for each learning iteration, updating the parameters as to move closer to the values that minimize the cost function. After the j-th iteration the model's parameters are updated according to

$$w_j = w_j - \alpha \frac{\partial J(w)}{\partial w_j} \tag{2.7}$$

where the second term of the equation calculates the slope, or gradient, of the curve at each iteration as represented in 2.6.
The gradient arises as a result of calculating the partial derivative (∂J) of the cost function with respect to each model parameter w_j, where j varies from 1 to n. Here, α represents the learning rate i.e. determining the speed at which we approach the minimum. If α is excessively large, there is a risk of overshooting. On the other hand, if it's too small, meaning small steps of learning, the overall training will take more time. One of the most used Gradient Descent Method is the Stochastic Gradient Descent Method (SGD) where the optimizer updates the parameters using only a single training instance in each iteration. The training instance is usually selected randomly. Stochastic gradient descent is often preferred to optimize cost functions when there are hundreds of thousands of training instances or more, as it will converge more quickly than other gradient descent methods. In the next chapters it will be introduced the specific optimizer that has been used during the experiments.

2.1.5 A common issue: over-fitting

Over-fitting is a concept in ML, which occurs when a statistical model fits exactly the training data. When this happens, the algorithm unfortunately cannot perform accurately unseen data, defeating its purpose. Generalization of a model to new data is ultimately what allows us to use ML algorithms every day to make predictions and classify data. However, when the model trains for too long on sample data or when the model is too complex, it can start to learn the "noise", or irrelevant information, within the dataset. When the model memorizes the noise and fits too closely to the training set, the model becomes "overfitted", and it is unable to generalize well to new data. If a model cannot generalize well to new data, then it will not be able to perform the classification or prediction tasks that it was intended for. Low error rates and a high variance are good indicators of overfitting. In order to prevent this type of behavior, part of the training dataset is typically set aside as the "test set" to check for overfitting. If the training data has a low error rate and the test data has a high error rate, it signals overfitting.

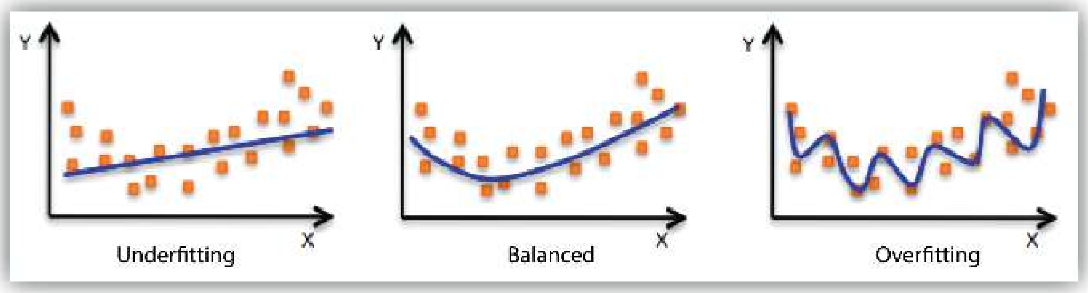


Figure 2.7: Illustration of under-fitting vs over-fitting.

2.1.6 Convolutional Neural Networks (CNN)

Besides fully connected layers, as we used in the feed-forward NNs, also convolutional layers are used especially in image classification experiments. These layers use the convolution of the layer's input, often inserted in matrix form, with another matrix, called kernel[8]. A Convolutional Neural Network is a class of neural networks that are specialized in processing data that has a grid-like topology, such as an image. A digital image is a binary representation of visual data and contains a series of pixels arranged in a grid-like fashion where pixel values denote brightness and color of each pixel[3]. A convolution is the simple application of a filter to an input that results in an activation. Repeated application of the same filter to an input results in a map of activation called a feature map, indicating the locations and strength of a detected feature in an input, such as an image. The innovation of convolutional neural networks is the ability to automatically learn a large number of features (i.e. **feature extraction**) by using numerous filters in parallel and to reduce the aforementioned number of features layer by layer, ending with less parameters used to make the classification. The result is highly specific features that can be detected anywhere on input images. In order to stress the concept behind these innovative architectures, we are going to divide the characteristic of each element that compose the CNN.

- **Convolutional layer:** A convolution is a linear operation that involves the multiplication of a set of weights with the input, much like a traditional neural network. Given that the technique was designed for two-dimensional input, the multiplication is performed between an array of input data and a two-dimensional array of weights, known as kernel. The filter is smaller than the input data and the type of multiplication applied between a filter-sized patch of the input and the filter is a dot product. A dot product is the element-wise multiplication between the filter-sized patch of the input and filter, which is then summed, always resulting in a single value. Because it results in a single value, the operation is often referred to as the "scalar product". This systematic application of the same filter across an image is a powerful idea. If the filter is designed to detect a specific type of feature in the input, then the application of that filter systematically across the entire input image allows the filter to discover a feature anywhere in the image. This capability is commonly referred to as translation invariance. The output from the multiplication between the filter with the input array one time is a single value. As the filter is applied multiple times to the input array, the result is a two-dimensional vector of output values that represent a filtering of the input. Once the feature map is created (2-D output), we can pass its value through a nonlinear function, such as ReLU, much like we do for the outputs of a fully connected layer. The convolution layer uses filters that perform convolution operations as it is scanning the input image with respect to its dimensions. Its hyper-parameters include the filter size

F, which is the matrix that passes through the image and registers the pixel values in that position, and the stride S that is the step with which the convolutional layer moves pixel by pixel. The resulting output is called feature map or activation map. The goal of the convolutional layer is to capture as many information as it can and to send forward to the remain layers until it's reduced in a way that only the essential information (that are characteristic for the image) are left and are used for the classification.

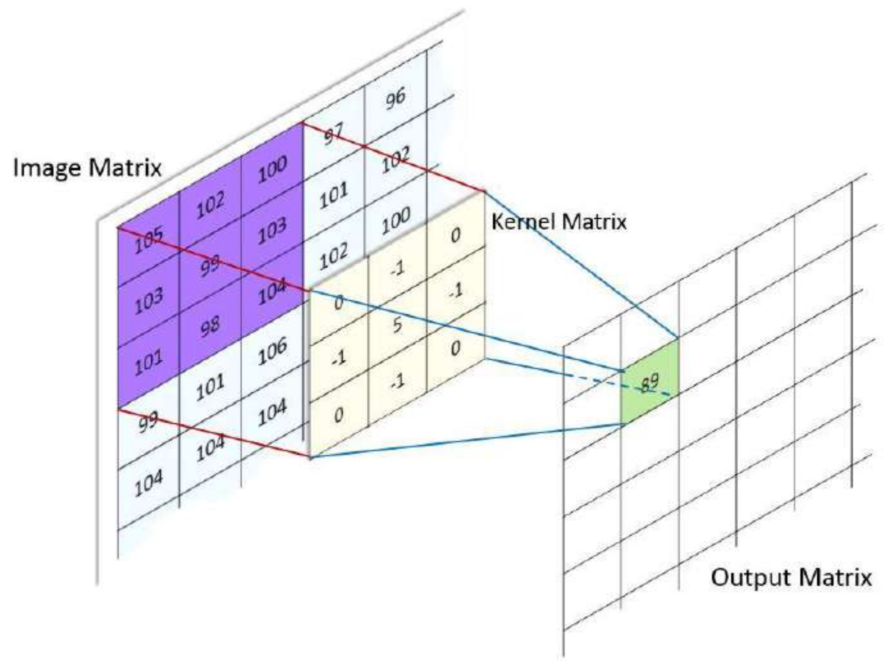


Figure 2.8: Representation of what happens in a convolutional filter. A convolution converts all the pixels in its receptive field into a single value, according to the kernel matrix that passes through, and collect a reduced in number pixel into the corresponding position of a secondary matrix or vector. For example, by applying a convolution to an image, the image size will decrease as well as bringing all the information in the field together into a single pixel.

- **Pooling layer:** The pooling layer is a downsampling operation, typically applied after the convolution layer. It replaces the output of the network at certain locations by deriving a summary statistic of the nearby outputs. This helps in reducing the spatial size of the representation, which decreases the required amount of computation and weights. The pooling operation is processed on every slice of the representation individually. There are several pooling functions such as the average of the rectangular neighborhood, L2 norm of the rectangular neighborhood, and a weighted average based on the distance from the central pixel. However, the most popular process is max pooling, which reports the maximum output from the neighborhood.

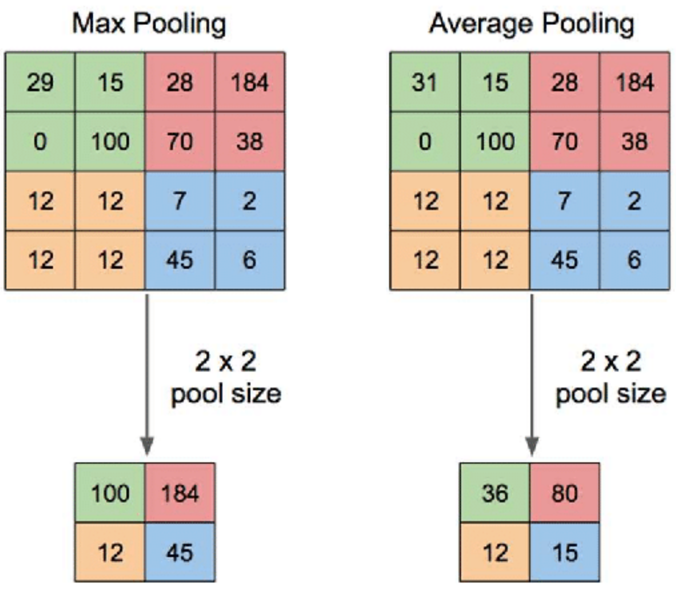


Figure 2.9: Pooling layers are used to reduce the dimensions of the feature maps. Thus, it reduces the number of parameters to learn and the amount of computation performed in the network. The pooling layer summarises the features present in a region of the feature map generated by a convolution layer..

- **Fully connceted layer (FC):** The fully connected layer (FC) operates on a flattened input where each input is connected to all neurons. Neurons in this layer have full connectivity with all neurons in the preceding and succeeding layer. The FC layer helps to map the representation between the input and the output.

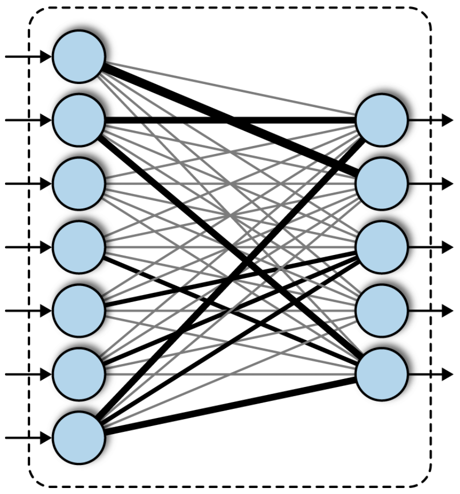


Figure 2.10: A fully connected layer refers to a neural network in which each input node is connected to each output node. The classification, in its true sense of the word, is made at this level: the pixel value computed by the previous filters is linked to the correspondent output label.

The CNN has multiple parameters that are calculated and optimized during the epochs in the training phase. The number of parameters in a given layer is the count of "learning" elements of a filter. Parameters, in general, are statistical weights that are learned during training which contribute to model's predictive power, and they change during back-propagation process.

2.2 Elements of Quantum Computing

Quantum Mechanics has the curious distinction of being simultaneously the most successful and the most mysterious of our scientific theories. It was developed in fits and starts over a remarkable period from 1900 to the 1920s, maturing into its current form in the late 1920s. Physicists had great success applying Quantum Mechanics to understand the fundamental particles and forces of Nature, culminating in the development of the Standard Model of Particle Physics. Over the same period, its application brought to new discoveries about astonishing phenomena in our world, from polymers to semiconductors, from superfluids to superconductors. But, while these developments profoundly advanced our understanding of the natural world, they did only a little to improve our understanding of Quantum Mechanics [30]. This began to change in the 1970s and 1980s, when a few pioneers were inspired to ask whether some of the fundamental questions of Computer Science and Information Theory could be applied to the study of quantum systems. Instead of looking at quantum systems purely as phenomena to be explained as they are found in nature, they looked at them as systems that can be designed[29]. Thanks also to the birth of this new technology, new questions combining Physics, Computer Science, and Information theory rose up.[30] These include questions such as: what are the fundamental physical limitations on the space and time required to construct a quantum state? How much time and space are required for a given dynamical operation? What makes quantum systems difficult to understand and simulate by conventional classical means?[30] These questions are not completely solved, but one thing is certain: if, at the beginning of 20s, Quantum Mechanics and especially Quantum Computers (QCs) were merely theoretical thoughts, today they are real and constitute one of the most prominent research sector in which people are involved. Nowadays scientists are strongly convinced that quantum computers will be able to overcome classical computers and open the way to new answers that can solve today's most tedious open questions. What many years ago was a stylized drawing on a chalk-stained black slate is now becoming a reality. From this point, some of the basics concepts and applications of QCs are presented in order to gently introduce how QCs work.

2.2.1 Quantum bits

The bit is the fundamental concept of classical computation and classical information. Quantum computation and quantum information are built upon an analogous concept, the quantum bit, or qubit for short. In this section we introduce the properties of single and multiple qubits, comparing their properties to those of classical bits. Just as a classical bit has a state – either 0 or 1 – a qubit also has a state. Two possible states for a qubit are the states $|0\rangle$ and $|1\rangle$, which correspond to the states 0 and 1 for a classical bit. The main difference between classical bits and qubits is that a qubit can be in a linear combination of state, often called superpositions:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{2.8}$$

The numbers α and β are complex and their modulus square defines the probability for a system described by the above equation, of being in the $|0\rangle$ or $|1\rangle$ state. The state of a qubit is a vector in a two-dimensional complex vector space. The special states $|0\rangle$, $|1\rangle$ are known as *computational basis* states, and form an orthonormal basis for this vector space. We can examine a bit to determine whether it is in the state 0 or 1. For example, computers do this all the time when they retrieve the contents of their memory. Rather remarkably, we cannot examine a qubit to determine its quantum state, that is, the values of α and β. Instead, quantum mechanics tells us that we can only acquire much more restricted information about the quantum state. When we measure a qubit we get either the result 0, with probability $|\alpha|^2$, or the result 1, with probability $|\beta|^2$. Naturally, $|\alpha|^2+|\beta|^2 = 1$, since the probabilities must sum to one. Geometrically, we can interpret this as the condition that the qubit's state be normalized to length 1. Thus, in general a qubit's state is a unit vector in a two-dimensional complex vector space. This dichotomy between the unobservable state of a qubit and the observations we can make lies at the heart of quantum computation and quantum information. As we have said, a qubit can exist in a continuum of states between $|0\rangle$ and $|1\rangle$ – until it is observed. Classical bits instead are like a coin: could be only 0 or 1, head or tail. For example, a qubit can be in the state

$$|\psi\rangle = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle, \tag{2.9}$$

which, when measured, gives the result 0 50% ($|1/\sqrt{2}|^2$) of the time, and the result 1 50% of the time.
Because $|\alpha|^2 + |\beta|^2 = 1$, it's possible to re-write as

$$|\psi\rangle = e^{i\gamma}(\cos\frac{\theta}{2}|0\rangle + e^{i\phi}\sin\frac{\theta}{2}|1\rangle) \tag{2.10}$$

The numbers θ and ϕ define a point on the unit three-dimensional sphere, as shown below. This sphere is often called the Bloch sphere. Despite this strangeness, qubits are

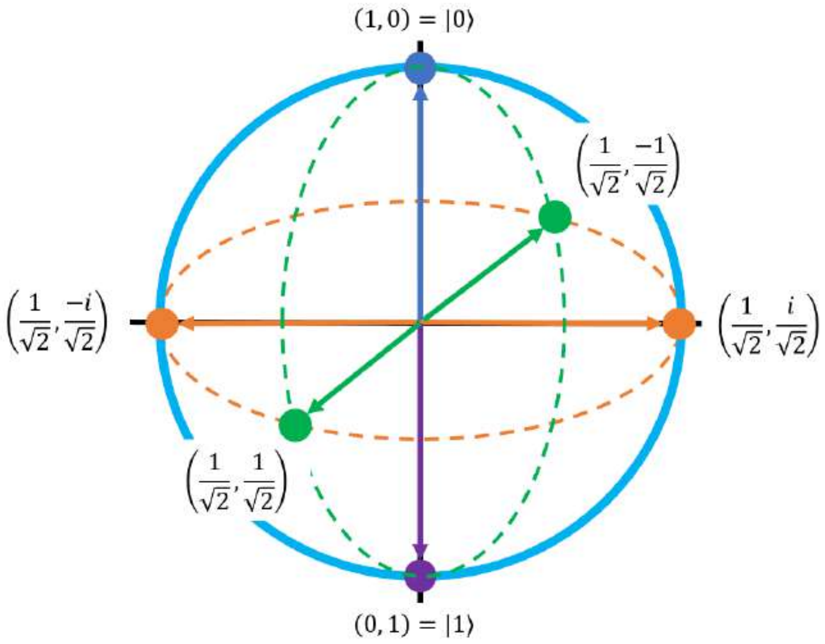

$(1,0) = |0\rangle$
$\left(\frac{1}{\sqrt{2}}, \frac{-1}{\sqrt{2}}\right)$
$\left(\frac{1}{\sqrt{2}}, \frac{-i}{\sqrt{2}}\right)$
$\left(\frac{1}{\sqrt{2}}, \frac{i}{\sqrt{2}}\right)$
$\left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$
$(0,1) = |1\rangle$

Figure 2.11: Bloch sphere representation.

decidedly real, their existence and behavior extensively validated by experiments, and many different physical systems can be used to realize qubits. To get a concrete feel for how a qubit can be realized it may be helpful to list some of the ways this realization may occur: as the two different polarizations of a photon; as the alignment of a nuclear spin in a uniform magnetic field; as two states of an electron orbiting a single atom[30].

2.2.2 The postulates of Quantum Mechanics

Quantum mechanics is a mathematical framework for the development of physical theories. On its own quantum mechanics does not tell you what laws a physical system must obey, but it does provide a mathematical and conceptual framework for the development of such laws. These postulates provide a connection between the physical world and the mathematical formalism of Quantum Mechanics[30].

Postulate 1: Associated to any isolated physical system is a complex vector space with inner product (that is, a Hilbert space) known as the state space of the system.

The system is completely described by its state vector, which is a unit vector in the system's state space. The simplest quantum mechanical system, and the system which we will be most concerned with, is the qubit. A qubit has a two-dimensional state space. Suppose $|0\rangle$ and $|1\rangle$ form an orthonormal basis for that state space. Then an arbitrary state vector in the state space can be written

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{2.11}$$

where α and β are complex numbers. The condition that $|\psi\rangle$ be a unit vector, $\langle\psi|\psi\rangle = 1$, is therefore equivalent to $|\alpha|^2 + |\beta|^2 = 1$.

Postulate 2: The evolution of a closed quantum system is described by a unitary transformation. That is, the state $|\psi\rangle$ of the system at time t_1 is related to the state $|\psi'\rangle$ of the system at time t_2 by a unitary operator U which depends only on t_1 and t_2.

Just as Quantum Mechanics does not tell us the state space or quantum state of a particular quantum system, it does not tell us which unitary operators U describe real world quantum dynamics. Quantum Mechanics merely assures us that the evolution of any closed quantum system may be described in such a way. The second postulate requires that the system being described be closed. That is, it is not interacting in any way with other systems. In reality, of course, all systems (except the Universe as a whole) interact at least somewhat with other systems. Nevertheless, there are interesting systems which can be described to a good approximation as being closed, and which are described by unitary evolution to some good approximation. Furthermore, at least in principle every open system can be described as part of a larger closed system (the Universe) which is undergoing unitary evolution. Postulate 2 describes how the quantum states of a closed quantum system at two different times are related[30]. A more refined version of this postulate can be given which describes the evolution of a quantum system in continuous time. Indeed, the final version of the second postulate can be rearranged as follow: he time evolution of the state of a closed quantum system is described by the

Schrodinger equation,

$$-i\hbar\frac{d}{dt}|\psi\rangle \;= H|\psi\rangle \tag{2.12}$$

where H is the Hamiltonian of the system and $\hbar$ is the Planck's constant[30].

Postulate 3: Quantum measurements are described by a collection M_m of measurement operators.

These are operators acting on the state space of the system being measured. The index m refers to the measurement outcomes that may occur in the experiment. If the state of the quantum system is $|\psi\rangle$ immediately before the measurement, then the probability that result m occurs is given by

$$p(m) = \langle\psi|M_m^\dagger M_m|\psi\rangle, \tag{2.13}$$

and the state after the measurement will be

$$\frac{M_m|\psi\rangle}{\sqrt{\langle\psi|M_m^\dagger M_m|\psi\rangle}}. \tag{2.14}$$

The measurement operators satisfy the completeness equation

$$\sum_m M_m^\dagger M_m = I. \tag{2.15}$$

The completeness relation expresses the fact that probabilities sum to one. For example, after a measurement in the computational basis, defined by the two operators $M_0 = |0\rangle\langle 0|, M_1 = |1\rangle\langle 1|$, a generic state of the form $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ will give the outcome 0 with a probability

$$p(0) = \langle\psi|M_0^\dagger M_0|\psi\rangle = \langle\psi|M_0|\psi\rangle = |\alpha|^2 \tag{2.16}$$

and will be projected onto the state

$$\frac{M_0|\psi\rangle}{|\alpha|} = \frac{\alpha}{|\alpha|}|0\rangle. \tag{2.17}$$

Postulate 4: The state space of a composite physical system is the tensor product of the state spaces of the component physical systems. Moreover, if we have systems numbered 1 through n, and system number i is prepared in the state $|\psi\rangle$, then the joint state of the total system is $|\psi_1\rangle \bigotimes |\psi_2\rangle \bigotimes ...|\psi_n\rangle$

Regarding what has been said before, if we suppose to have a quantum system with state space Q, and we want to perform a measurement described by measurement operators M_{m*} on the system Q, we introduce an ancilla system, with state space M, having an orthonormal basis $|m\rangle$ in one-to-one correspondence with the possible outcomes of the measurement we wish to implement. This ancilla system can be interpreted physically as an extra quantum system introduced into the problem, which we assume has a state space with the required properties. Letting $|0\rangle$ be any fixed state of M, define an operator U on products $|\psi\rangle|0\rangle$ of states $|\psi\rangle$ from Q with the state $|0\rangle$ by

$$U|\psi\rangle|0\rangle = \sum_m M_m|\psi\rangle|m\rangle. \tag{2.18}$$

We can delve further into the fundamentals of Quantum Mechanics, introducing additional key concepts and expanding on what has already been discussed, but that is not the primary goal of the project. It was designed with the intent of providing readers with foundational knowledge to initiate their understanding and prepare them for what comes next.

2.2.3 Quantum computation

Quantum gates

Changes occurring to a quantum state can be described using the language of quantum computation. Analogous to the way a classical computer is built from an electrical circuit containing wires and logic gates, a quantum computer is built from a quantum circuit containing wires and elementary quantum gates to carry around and manipulate the quantum information. Classical computer circuits consist of wires and logic gates. The wires are used to carry information around the circuit, while the logic gates perform manipulations of the information, converting it from one form to another. For a Quantum Computer the same story can be applied thanks to the linearity of Quantum Mechanics, in fact the only non-trivial member of logic gates class is the NOT gate, whose operation is defined by its truth table, in which $0 \rightarrow 1$ and $1 \rightarrow 0$, that is, the 0 and 1 states are interchanged. For Quantum Mechanics is possible to have a quantum NOT gate in which a process takes the state $|0\rangle$ to the state $|1\rangle$, and vice versa. However, specifying the action of the gate on the states $|0\rangle$ and $|1\rangle$ does not tell us what happens to superpositions of those states, without further knowledge about the properties of quantum gates. By the linearity, the quantum version of NOT gate takes the state

$$\alpha|0\rangle + \beta|1\rangle \tag{2.19}$$

to the corresponding state in which the states have been interchanged

$$\alpha|1\rangle + \beta|0\rangle \tag{2.20}$$

There is a convenient way of representing the NOT quantum gate in matrix form, which follows directly from the linearity of quantum gates:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{2.21}$$

Without going into the very basics of Quantum Computing, it has been preferred to present some other single-qubit gates just to give a quick overview of the structures that have been used in the experiment. Other two important quantum gates are:

Y gate whose matrix has the form

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}; \tag{2.22}$$

Z gate:

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{2.23}$$

The next quantum gate is the Hadamard gate, responsible for the creation of a superposition of states for qubits in the $|0\rangle$ or $|1\rangle$ states and has the form

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} . \tag{2.24}$$

The resulting states from the application of the Hadamard are in the X basis denoted by the vectors $|+\rangle$ (if the starting qubit is in the state $|0\rangle$) and $|-\rangle$ (if the starting qubit is in the state $|1\rangle$).

Parameterized gates

There are other kinds of quantum gates that are functions of some unspecified parameters (angles) which are estimated by a variational approach. These unitary gates rotate the quantum state along a specific axis such as X, Y, or Z according to θ. They can be defined as exponential matrices with the corresponding Pauli matrix passed as an argument:

$$R_X(\theta) = e^{-i\frac{\theta}{2}X} = \begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} \tag{2.25}$$

$$R_Y(\theta) = e^{-i\frac{\theta}{2}Y} = \begin{pmatrix} \cos\frac{\theta}{2} & \sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} \tag{2.26}$$

$$R_Z(\theta) = e^{-i\frac{\theta}{2}Z} = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix} . \tag{2.27}$$

Together with the Hadamard, a multiple qubits-quantum gate, known as CNOT gate, generates entanglement between quantum states, and for the moment where is applied, the states will be connected in such a way that who is going to measure one of the qubit will influence the outcome of the second qubit, owned by another person. This gate has two inputs, known as the control and the target qubit, respectively and has the form

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} . \tag{2.28}$$

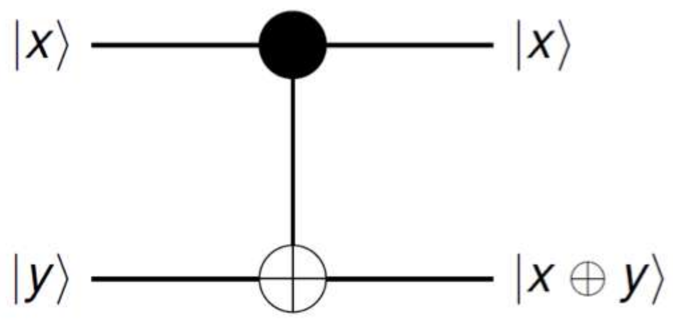


Figure 2.12: Example of the controlled gate.

The action of this particular quantum gate works as follow: if the control qubit is set to 0, then the target qubit is left alone. If the control qubit is set to 1, then the target qubit is flipped.
Another way of describing the CNOT is as a generalization of the classical gate, since the action of the gate may be summarized as $|A, B\rangle \ \rightarrow |A, A \bigoplus B\rangle$, where $\bigoplus$ is addition modulo two, which is exactly what the gate does. That is, the control qubit and the target qubit are XORed and stored in the target qubit. As for the single qubit case, the requirement that the probability be conserved is expressed in the fact that U_{CNOT} is a unitary matrix, that is, $U^{\dagger}_{CNOT}U_{CNOT} = I$.
We noticed that the CNOT can be regarded as a type of generalized gate. Can other classical gates be understood as unitary gates in a sense similar to the way those gates represent classical gates? It turns out that this is not possible. The reason is because the and gates are essentially irreversible or non-invertible. For example, given the output $A \bigoplus B$ from XOR gate, it is not possible to determine what the inputs A and B were; there is an irretrievable loss of information associated with the irreversible action of the gate. On the other hand, unitary quantum gates are always invertible, since the inverse of a unitary matrix is also a unitary matrix, and thus a quantum gate can always be inverted by another quantum gate. However, in a sense the controlled-NOT and single qubit gates are the prototypes for all other gates because of the following remarkable universality result: any multiple qubit logic gate may be composed from CNOT and single qubit gates.
To conclude the current session, it is worth saying that any kind of rotation we encountered before can be realized as a control gate: in fact, one of the next architecture used as ansatz for learning the parameters necessary to detect the images, makes use of controlled rotations that act on a specific qubit if and only if the corresponding state is equal to $|1\rangle$, otherwise they are not applied. To explicitly demonstrate the effect of combining Hadamard and CNOT gate to realize entanglement, it's presented a quick

application that should illustrate what happens on a quantum state below.
Suppose to start with a two-qubit quantum state $|00\rangle$ and apply an Hadamard gate on the first qubit, followed by a CNOT where the controlled qubit is again the first one: the application of the Hadamard results into

$$|00\rangle \xrightarrow{H_1} \frac{(|0\rangle + |1\rangle)}{\sqrt{2}}|0\rangle = \frac{(|00\rangle + |10\rangle)}{\sqrt{2}} \tag{2.29}$$

This will then act as a control input to the CNOT gate, which only inverts the target (the second qubit) when the control (the first qubit) is 1. Thus, the controlled NOT gate transforms the second qubit as follows

$$\frac{(|0\rangle + |1\rangle)}{\sqrt{2}}|0\rangle \xrightarrow{CNOT_{(1,2)}} \frac{(|00\rangle + |11\rangle)}{\sqrt{2}} = |\phi^+\rangle. \tag{2.30}$$

The last state is known as Bell's state, one of the maximal entangled two-qubit state.

2.2.4 Quantum Machine Learning

The structures of the neural networks proposed in the Quantum Machine Learning (QML) field are basically of two kinds: the Standard QNN and the QCNN. Before going deep into the main characteristics of each quantum model, is necessary to say that quantum neural networks act as a quantum analog to the classic NNs. Since all quantum gates are unitary and hence linear, the main difficulty of building a QNN is introducing the non-linearity that is rather present in the activation functions used by such devices. We solve this problem by encoding the input vector to a quantum state non-linearly with a parameterized quantum circuit: the feature map. The advantage of introducing QNN layers is that we can access vectors of exponential dimensional Hilbert spaces with only polynomial resources on a quantum computer[44].

QNN

QNN, or quantum neural networks, are particular architectures in which, the above-mentioned artificial neurons present in the classical models, are replaced by the elements of Quantum Computing: quantum bits organized in quantum circuits. In this case, qubits are used as the fundamental carriers of information (represented by the image feature), as the bits were so in the classical counterparts, while quantum gates are used to learn the transferred information until each qubit is measured and the result (in form of probabilities) used to perform the prediction. A general QNN is usually composed of two kinds of quantum circuits, one responsible for the encoding of classical features into quantum states (as we are going to explain) and a variational quantum circuit where each gate learns the parameter that represents the encoded classical data. The word "variational" means that the parameter, calculated at the first iteration, will be calculated again in the successive iteration by minimizing the value of an error/cost function. The choice of embedding is usually geared toward enhancing the performance of the quantum model and is typically neither optimized nor trained[2]. Once data is encoded, as we already said above, the variational model containing parameterized gates is applied and optimized for a particular task. QNNs can be implemented on today's quantum computers as variational quantum algorithms (VQA). Their process is a quantum-classical hybrid: the algorithms themselves are executed on a quantum computer, but the optimization process is done classically[8]. Further, a quantum algorithm computes the training loss function. During the training, the parameters are updated classically such that the training loss is optimized since this task can be efficiently fulfilled in this way. A substantial benefit of VQAs is that they can be successfully executed on NISQ devices. These devices are highly impaired through noise entering with each quantum gate. This limits the number of quantum gates within one quantum circuit before the noise outweighs the algorithm's performance.

QCNN

The next architecture known as QCNN, or quantum convolutional neural network, is motivated by the classical CNN we have told previously and introduces a quantum circuit model extending the classical key properties to the quantum domain[12]. It utilizes tree-like (or hierarchical) structures with which the number of qubits from a preceding layer is reduced by a factor of two for the subsequent layer. Such architectures consist of $O(log(n))$ layers for n input qubits, thereby permitting shallow circuit depth[17]. The progressive reduction of the number of qubits is analogous to the pooling operation in the CNN. A distinct feature of the QCNN architecture is the translational invariance, which forces the blocks of parameterized quantum gates to be identical within a layer[17]. In the QCNN architecture, the unitary U_i consists of two-qubit quantum circuit blocks, and the convolution and pooling part each uses identical quantum circuit blocks within the given layer. Since a two-qubit gate requires 15 parameters at most[42], in the ith layer consisting of $l_i > 0$ independent convolutional filter and one pooling operation the maximum number of parameters subject to optimization is $15(l_i + 1)$. Then the total number of parameters is at most $15 \sum_{i=1}^{log_2(n)} (l_i + 1)$ if the convolution i=1 and pooling operations are iterated until only one qubit remains[17]. The model of QCNN applies the convolution layer and the pooling layer, which are the main features of CNN, to quantum systems. The concept can be summarized as follows:

- The convolution circuit finds the hidden state by applying multiple qubit gates between adjacent qubits;

- The pooling circuit reduces the size of the quantum system by observing the fraction of qubits or applying 2-qubit gates such as CNOT gates;
- Repeat the convolution circuit and pooling circuit;
- When the size of the system is sufficiently small, the fully connected circuit predicts the classification result.

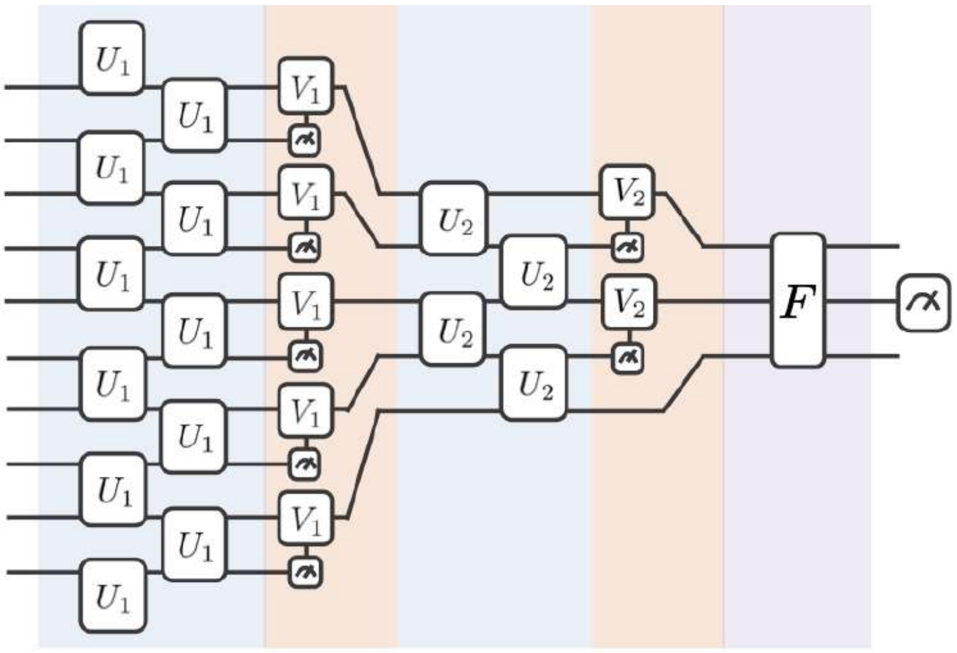


Figure 2.13: Iris Cong QCNN scheme[12].

Note that some of the QNNs and QCNNs are designed for pure quantum tasks, whereas others can be exploited for classical inputs as has been done in the current project. In the latter case interestingly, the classical data has to be encoded first, i.e. turned into quantum states. Such encoding, or embedding process that has been come out refers to the fact that is necessary, as Quantum Mechanics requires, to convert the classical information to a quantum state, moving from a classical vector-space to the larger Hilbert space where they can be analyzed via quantum algorithms.

Quantum scheme for Neural Networks

The initial point of doing Machine Learning by exploiting the advantage of Quantum Mechanics is the transformation of the classical data into quantum states. At this stage, it's convenient to provide a summary of the main processing steps involved in the QML:

- **State preparation / Encoding:** The state preparation, or encoding / embedding, can be considered the essential part that characterizes quantum models. As has been already anticipated, this is the phase in which classical data, which can represent a gray-scale or an RGB image, or any other kind of data for example, are converted into quantum states. A quantum circuit known as *feature map* is the circuit that makes this task. It's composed of a different type of quantum gates: the parameterized quantum gates. Such gates are unitary matrices that depend on a specific parameter, like the angle that the quantum state spans with the vertical Z axis of the Bloch sphere, or a time or a frequency for a specific wave packet that represents a qubit that can excite a well-known Hamiltonian (mostly used in Neutral Atoms architectures), that needs to be guessed and updated to perform an accurate prediction. This parameter can vary along the whole domain where is defined and, for this reason, is called a variational parameter. Roughly speaking, the feature map assigns to a classical data point its quantum state representation to find a representation of the data such that the known metric of the Hilbert space faithfully reproduces the unknown metric of the original data. More formally, let χ be a set of input data. A feature map ϕ is a function that acts as

  $$\phi : \chi \rightarrow F \tag{2.31}$$

  where F is the feature space. The outputs of the map on the individual data points. The feature map transforms an input vector x into

  $$\phi : U_\phi(x) \rightarrow |\phi(x)\rangle. \tag{2.32}$$

  There are three main methods for embedding classical data into quantum states:

  - **Basis embedding:** In basis embedding, the data has to be in the form of a binary string to get embedded. The idea behind basis embedding is using a computational basis. Approximating a scalar value to its binary form and then transforming it to a quantum state. The first step is to approximate a number by a binary bit string and the second step is encoding it by a computational basis state

    $$|D\rangle = \frac{1}{\sqrt{M}} \sum_{m=1}^{M} |x^{(m)}\rangle. \tag{2.33}$$

    Generally, basis embedding is not used at all since is uncomfortable to represent data as bit strings of the computational basis.

  - **Angle embedding:** Angle encoding is a simple and efficient method for embedding data, it is one of the most basic forms of encoding that transforms classical data into a quantum state, but it is not robust, in the sense that, when multiple features are encoded by the mentioned scheme, the application of many unitary gates could be hard to simulate and could produce errors. The angle embedding is performed by applying rotations on the x-axis or y-axis using quantum gates along with the values that have to be encoded. If we want to apply angle embedding on a dataset the number of rotations will be the same as the number of features in the dataset the n-dimensional sample would take a number n of qubits to generate the set of quantum states. The advantage of using angle embedding is that, for each qubit is possible to encode different features by applying as numerous quantum gates as the number of used qubits, parallelizing the embedding to minimize the circuit depth. On the contrary, to encode multiple features like 784 (28,28 Mnist images) we need a large number of qubits that ordinary computers are not able to simulate and we fall into the vicious circle in which we need to ask whether it is worth making use of supercomputers to gain maybe slightly better results.

  – **Amplitude embedding :** The amplitude embedding is also known as wave function embedding. In simple terms, the amplitude is the height of a wave. In this kind of embedding the data points are transformed into amplitudes of the quantum state. Let's consider a dataset $\alpha = \alpha_1, ..., \alpha_n$ having n number of dimensions, the dataset is initially normalized to length 1. The number of amplitudes to be encoded is the product of the number of dimensions and the number of samples. The encoding is less dense as compared to the basis or angle encoding. It requires a $\log_2(n_{features})$ number of qubits to encode such several features.

$$U_\phi(x) : x \in \mathcal{R}^N \rightarrow |\phi(x)\rangle = \frac{1}{||x||} \sum_{i=1}^{N} x_i |i\rangle \tag{2.34}$$

    Clearly, with amplitude encoding, a quantum computer can represent exponentially large classical data. This can be of great advantage in QNN algorithms. Since the number of parameters subject to optimization scales as $O(\log(n))$, the amplitude encoding reduces the number of parameters doubly exponentially with the size (i.e. dimension) of the classical data. However, the quantum circuit depth for amplitude encoding usually grows as $O(poly(N))$ and it can be responsible for slowness and error occurrence.

- **Variational ansatz**: The variational ansatz is the quantum circuit responsible for learning the quantum parameters, such as angles, that characterize the quantum states representative of the classical data point that composes the dataset. The angles in this case, as for the classical models, are learned thanks to the parameterized quantum gates following a minimization of a suitable loss function such as the Cross-entropy, Mean Squared Error, etc. Once the parameters are learned (after the measurement) in the first epoch, the loss function is evaluated and the parameters are stored and used to compute the prediction. In the next epoch, the optimizer re-calculates and updates the parameters to reach a minimum value in the loss-landscape function already evaluated in the previous epoch. The entire process proceeds according to the number of epochs that are set in the algorithm, or it stops where the cost function is no longer able to decrease. During each iteration, together with the value of the loss, the corresponding value of the accuracy (number of correct predictions over the total number of predictions) is printed to check if the model is learning from data. If everything works properly, the accuracy of the training should increase. At the end of the process, the model is tested on a new set of data it has not seen during the training, and the correct number of predictions it makes is an indicator of how good it is at classifying new data. Here the entanglement is exploited to catch information faster, with fewer iterations and making use of few qubits.

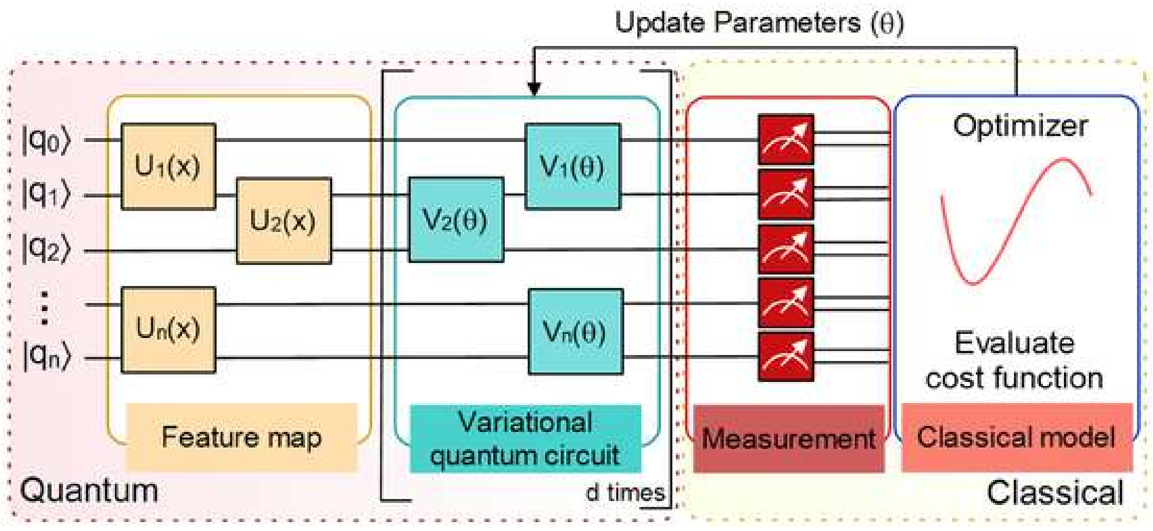


Figure 2.14: Representation of a quantum neural network composed by the feature map circuit, the parameterized circuit (variational ansatz), the measurement operation, and the classical optimization part. As it can be seen, the parameters inside the two parameterized quantum circuits (feature map and variational ansatz) are different, i.e. x and θ. This difference reflects the fact already explained in the upper chapter. The feature map expects the features as input parameters that represent the corresponding angle spanned by the quantum state correspondent of specific data. The ansatz, on the contrary, expects the learning parameters that are angles too, but are essentially the weights that it needs to learn to predict the data itself. The measurement is applied to all the qubits giving the parameters for the specific quantum state composed of those qubits. In the end, the optimization is performed according to gradient descent through the loss function.

- **Measurement:** The measurement can be considered the last part in which a Quantum Machine Learning model differs from a classical one. The fundamental concept behind measuring a specific qubit is that, following the "data acquisition" process, where parameterized quantum gates manipulate the vector representing a classical data point within the Hilbert space, the wave function of the state is a superposition composed of individual wave functions created for each qubit. When the measurement is performed, the wave function would collapse onto the most probable quantum state (represented by a sequence of 0s and 1s). The result of a measurement operation is a collection of probabilities associated with the set of quantum states originated by the quantum circuit. These last probabilities are connected to the parameters that characterize each input data, thus they will be used to understand which class the measurement's output will belong to. Moreover, the importance of the measurement resides in the possibility of evaluating the cost function. The value of each single angle parameter comes out thanks to the measurement, and will be directly re-calculated by the optimizer according to a loss function minimization. Generally, the measurement is performed on each qubit at the end of the ansatz, but for architectures such as QCNN, the measurement operation is applied on the last single qubit at the end of the last layer of the model, since the majority of the unitary gates (containing other learning weights) is applied to it. To leave the reader with clarity, we summarize the mathematical description of the model's parts, emphasizing the measure. Initially, the feature map F maps a real-valued classical data point x into a d-qubit quantum state $|\psi\rangle$

$$|\psi\rangle = F(x)|0\rangle^{\otimes d} \tag{2.35}$$

Next, an ansatz A manipulates the prepared quantum state through a series of entanglements and rotation gates. The angles of the ansatz's rotations are parameterized by a vector θ

$$|\phi(x,\theta)\rangle = A(\theta)|\psi(x)\rangle \tag{2.36}$$

Finally, an observable O is measured, and the eigenvalue corresponding to the resultant quantum state is recorded. In most ML applications, a variational quantum circuit is run many times using a particular input x and parameter vector θ so that the circuit's expectation value, denoted by f, can be approximated.

$$f(x,\theta) = \langle\phi(x,\theta)|\hat{O}|\phi(x,\theta)\rangle. \tag{2.37}$$

When a variational quantum circuit is used for machine learning, this approximated expectation value is typically treated as the output of the model.

Barren plateaus

Barren plateaus are introduced in the Quantum Computing field for the first time in [27]. In classical optimization, it is suggested that saddle points, not local minima, provide a fundamental impediment to rapid high-dimensional non-convex optimization. The ongoing development of Noisy Intermediate-Scale Quantum (NISQ) computers has led to considerable excitement about the potential quantum advantage that can be obtained in several optimization problems that are used in almost all fields of science today as discussed in 1. Various hybrid quantum-classical algorithms give the opportunity to utilize NISQ computers and variational quantum algorithms (VQAs), in this sense, offer a potential way of leveraging quantum computers alongside classical computers in a hybrid fashion[22]. When it comes to deployment of VQAs on NISQ devices, the limitations of existing quantum computers (i.e., noisy gates and limited circuit depth) restrict the overall potential of these algorithms. Moreover, VQAs themselves are not without issues. The parameters of a Variational quantum classifier (VQC) are initialized from a random distribution. Over the course of training, the output of the VQC is measured with respect to the observable $\hat{O}$ and the gradients of every unitary gate are estimated. The set of parameters at a given time step and the gradients of the unitary gates are then passed to a classical computer which updates these parameters according to a gradient descent rule by an optimizer. Ideally, for an optimization problem, the set θ^* should correspond to the best set of parameters that minimizes the cost function. However, it has been observed that when the VQC is complex either in number of qubits used to represent the input or the number of layers of unitaries, the optimization halts at a sub-optimal set of parameters that do not correspond to a minima in the optimization surface. This situation occurs when the circuit gets stuck on a plateau from where there are no good descent directions and is commonly referred to as being stuck in a **"barren-plateau"**[22].

Chapter 3

Related works

In this Chapter, we want to discuss the *State of the art* of the described models, starting from ML models like Support Vector Machines (SVM) to reach the more sophisticated and complex DL models, i.e. the classical CNN and, finally, the novel QNNs. We would like to clarify that we documented multiple articles and papers about the topic, but these four cited projects constitute the heart of our background from which the simulations took inspiration.

State of the art

In the field of ML, there are plenty of models dedicated to image recognition and classification. One of the most simple algorithm is the Decision Tree. Decision tree-based algorithms are an important part of the classification methodology. Their main advantage is that there is no assumption about data distribution, and they are usually very fast to compute. In image classification, the decision trees are mostly reliable and easy to interpret, as their structure consists of a tree with leaves which represent class labels, and branches that use logical conjunction to produce a value based on an "if-then" rule. These values produce a set of rules that can be used to interpret the instances in a given class[18]. The Random Forest (RF) algorithm are built upon the concept of decision tree learning. The RF relies on many self-learning decision trees which in their sum make up a "Forest". The idea behind using many decision trees (i.e. an ensemble) is that many base learners can come to one strong and robust decision compared to a single Decision Tree. The RF uses self-learning decision trees and involves automatically defining rules at each node based on a training dataset for which feature inputs and labels are known. One way to define the optimal split given a set of input features and training points, would be trying to minimize the heterogeneity (i.e. class-mixtures) of the two resulting subsets of data. Models such as DTs and RFs are considerably improved over time, until a new class of algorithms has been brought to light: SVM. Support vector machines (SVM) are supervised learning models with associated learning algorithms that analyze data for classification and regression analysis. In the case of SVM, a data point is viewed as a p-dimensional vector, and we want to know whether we can separate such points with a (p-1)-dimensional hyperplane. This is called a linear classifier. There are many hyperplanes that might classify the data. One reasonable choice as the best hyperplane is the one that represents the largest separation, or margin, between the two classes. So we choose the hyperplane so that the distance from it to the nearest data point on each side is maximized. If such a hyperplane exists, it is known as the maximum-margin hyperplane and the linear classifier it defines is known as a maximum-margin classifier. Even though SVM are still used today, they are limited by their linearity and cannot solve completely more complex problems that uses images with large number of features. In this context people veered towards a new approach: Deep Learning. As already explained in the Introduction 1, the principal model in DL is the CNN. Convolutional Neural Networks have been implemented to solve various visual problems[28]. At the beginning, these kinds of models were used on specific toy and simple datasets such as MNIST, CIFAR, BREASTCANCER..., that usually are provided by the well-known ML tools as *Tensorflow*, *Scikit-Learn* and *PyTorch*. On the contrary, nowadays people use the most sophisticated versions on real images representing medical plates, urban areas (as ESA has done in [37]), fractures of objects, benign or malignant tumors etc... From the end of the 20th century, several CNN models have been realized and tested in order to obtain high results in the generalization problem. The main models for capacity, ability and success available right now are *LeNet*, *AlexNet*, *VGG* and *ResNet*. LeNet-5 architecture is perhaps the most widely known CNN architecture. It was created by Yann LeCun[24] in 1998 and widely used for written digits recognition (MNIST). Starting with a grayscale tensor image of shape (32×32x1), the goal of LeNet-5 was to recognize handwritten digits. In the first step, LeNet-5 uses a set of six 5×5 filters with a stride of one. Because authors in [24] used six filters, it ends up with a shape of (28x28x6) and with a stride of one and no padding. Then the Le-Net neural network applies pooling to reduce again the size of the data. The application of convolutional and average pooling filters proceeds until the dimension of the image would be reduced to (5x5x16) which gives a total of 400 parameters. In the end, it has 2 fully connected layers where the first one fully connects each of these 400 nodes with every one of 120 neurons. Then, the same with the last fully connected layer that fully connects each of these 120 nodes with every one of 84 nodes. Finally, we have the output layer, where a softmax activation function is used for predictions. The AlexNet CNN architecture was developed by Alex Krizhevsky[21], Ilya Sutskever, and Geoffrey E. Hinton. It is quite similar to LeNet-5, only much larger and deeper, and it was the first to stack convolutional layers directly on top of each other, instead of stacking a pooling layer on top of each convolutional layer. As seen in the AlexNet architecture, CNNs were starting to get deeper and deeper. The most straightforward way of improving the performance of deep neural networks is by increasing their depth known as the number of layers between the input data and a final dense layer used for the classification. Visual Geometry Group (VGG) invented the

VGG-16, which has 13 convolutional and 3 fully-connected layers, carrying with them the Relu activation function from AlexNet. The last principal architecture that has gained lot of popularity is the Residual Network (ResNet), developed by Kaiming He et al., which uses an extremely deep CNN composed of 152 layers. The key to being able to train such a deep network is the skip connections: The signal feeding into a layer is also added to the output of a layer located a bit higher up the stack.
Motivated by the success of classical DL as well as advances in quantum computing, quantum neural networks, which share similarities with classical neural networks and contain variational parameters, have drawn a wide range of attention. There are multiple reasons to develop the quantum version of neural networks, first, quantum computers hold the potential to outperform classical computers from several aspects: some quantum Fourier transform based algorithms, such as Shor's factoring algorithm, can achieve exponential speedups compared with the best known classical methods; in addition, the possibility of approaching the so called NP-Hard problems (computational problems whose complexity increases as more inputs are added to the system) and to obtain a solution in a reasonable time constitutes an enticing service we could investigate. Lastly, one of the most open problem where lots of work is developing, focuses on ML. The ability of ML models, as we have anticipated before, lies in its ability to fit a variety of functions[2]. Deep NNs have proven to be extremely powerful models, capable of capturing intricate relationships by learning from data while QNNs serve as a newer class of ML models that are deployed on quantum computers and use quantum effects such as superposition, entanglement, and interference, to do computation. Some proposals for QNNs hint at potential advantages, such as speed-ups in training, faster processing thanks to wider Hilbert space where do computations. These fascinating results stimulate the exploration of potential advantages using QNN models, especially in the current age of big data. The early QNNs developed in the past shared the same structures as illustrated in 2.2.4. They combine an encoding quantum circuit that represents classical data as quantum states, a trainable quantum circuit, known as ansatz, that learns the parameters to update as to minimize a cost function, a classical optimization process where these parameters are re-calculated and the measurement operation necessary to evaluate the cost function and ultimately to perform the prediction.
The QNN models we are going to present are built from multiple quantum circuits that are combined together. The adopted scheme follows the *Data re-uploading* technique described in [31] in which encoding quantum gates are grouped together and followed by the variational gates to ensure the learning. The article is probably one of the most important, in which authors have proved that a single qubit provides sufficient computational capabilities to construct a universal quantum classifier when assisted with a classical subroutine[31]. Although it may seem counter-intuitive that a single qubit can be universal since it only offers a simple superposition of two states, but combining it with multiple re-uploads of data can in principle circumvent this limitation. A quantum circuit can then be organized as a series of data re-uploading and single-qubit processing

units.[31]. The cited paper constitutes a milestone in the sector and is considered the starting point for any research in QML, it gives a suggestion of a new encoding strategy that we have used for our project, combining groups of unitaries that encode the features with quantum gates that learn to classify them, leading to use a shallower circuit composed by only one qubit, or few ones, that can significantly have a good impact on the necessary resources used to simulate the process.
A more suited article to approach to image classification with quantum models is [37] realized by European Space Agency (ESA). The article aims to investigate how hybrid quantum convolutional neural networks (h-QCNNs) can be successfully employed as image classifiers in the context of remote sensing[37]. The research conducted by ESA in fact focuses on Earth observations (EO), where pictures of vast areas of our Planet are scanned by a satellite, sent to data centers, and then analyzed with a ML model to reveal peculiar characteristics of it. The question formulated by the ESA refers to the possibility of introducing a quantum layer within the classical convolutional structure of the neural network to obtain improvements in the classification to the speed of learning. This study underlines the potentialities of applying quantum computing to an EO case study and provides the theoretical and experimental background for future investigations[37]. The work is divided into two parts: one in which three different quantum circuits are tested on a 10 class remote-sensing image dataset, EuroSAT, where an increasing classification accuracy is obtained by including a more complex level of entanglement within the circuit, and a second part where three difficult subsets for images of visually similar classes are created and then have been used to train three hybrid QCNNs, to classify finer and detailed images, namely fine-grain classifiers[37]. In this way, the four-qubit and the entanglement have been applied within the selected macro-classes and their inherent complexity used to encode details finer than in the overall setup[37].
To conclude, it's interesting to notice that the importance of this article resides in the results: the multi-class classification has demonstrated that the QCNNs performances are higher than the classical counterparts and it's reasonable to proceeding in the investigation of such quantum models in the future.
The [17] article describes the performances of a QCNN for hand-written digits and fashion items image recognition. Two models, a QCNN and a CNN, are realized and compared as to obtain which performs better on purely classical data. Different strategies of quantum embedding are used and some dimensionality reduction techniques (such as PCA and auto-encoder) are applied to reduce the dimensions of the data itself in order to use a small number of qubits to perform the simulation. The challenge lies in accurately assessing the quantum model's performance relative to the classical one, as the impact of reduction techniques on the models' behaviors cannot be easily unpacked. In other words, we've asked ourselves if the good accuracy obtained during the process is a direct consequence of the power of Quantum Computing, or have been, in a certain way, disguised as the classical data pre-processing techniques. The PCA, is a method to reduce the dimensions of large data sets, by transforming a large set of variables into a smaller

one that still contains most of the information in the large set. Reducing the number of variables of a data set naturally comes at the expense of accuracy, but the trick in dimensionality reduction is to trade a little accuracy for simplicity. Because smaller data sets are easier to explore and visualize and make analyzing data points much easier and faster for ML algorithms without extraneous variables to process. Even if in many cases the use of PCA is recommended and can considerably improve the performance of the model, in the Computer Vision domain can be sometimes dangerous since can discard important features that ensure the spatial components of the image.
An additional used technique with the same goal is the auto-encoder: a type of artificial neural network (ANN) used to learn efficient coding of unlabeled data. It compresses the input into a lower-dimensional code and then reconstructs the output from this representation. In this way is it possible to reduce the dimension of the data by creating more easy ones. Indeed, we decided to be as fair as possible during the experiment, without contaminating the QNN's abilities with classical data pre-processing. The only reduction that has been introduced is a simple feature-downsampling process. In this way, the pixels of the images used for training have been reduced in number as to use less qubits and a shallower circuit in the number of gates. In the following chapter all the schemes and the pre-processing of the data are explained in detail. The last quantum scheme we would talk about is presented in the article written by I. Cong[12]. It consists of a QCNN model, where quantum convolutional and pooling layers formed by unitary operators are respectively interchanged. The substantial difference between pooling layers belonged to the CNNs and QNNs is that the latter should contain a measurement operation, or a controlled operation on the next qubit, with which the number of qubits, used in the successive layer, is reduced. The result of the single-qubit measurement produces a state ($|0\rangle$ or $|1\rangle$) whose value determine or not an operation (as a rotation) on a successive target qubit. The benefits of the QCNN are, aside from an exponential speedup, a better feature extraction: quantum convolutional layers are able to capture the essence of information from the data in a better manner as compared to classical CNN due to the presence of entanglement and have the potential to transcend classical CNN that is only able to capture local correlations[17]. Thus, another advantage of QCNN models for NISQ computing is their intrinsically shallow circuit depth realized by reducing the number of qubits by a factor of 2 at each pooling layer.

Generalization from few training data

There exist numerous methods and experiments to assess the power of QNN models with respect to classical ML algorithms. Our main goal is to formulate an answer to this question by investigating the number of training data points required for a good generalization. A this purpose, the recent [10] article published in 2022 on *Nature*, in which the authors investigate the performances of quantum architectures in generalizing unseen data and have tried to study, under a mathematical perspective, the behaviour of the **generalization error**[1], was a guide for us.
The ultimate goal of ML is to make accurate predictions on unseen data and this is known as generalization. Roughly speaking, when a model is able to generalize well on a certain dataset it means that, once it has learnt the pattern of the training set, it has managed to predict a significantly high number of unseen images during the test process. Moreover, the generalization error is a way of detecting the model's ability of a correct classification of data. Problems such as *over-fitting* already discussed in 2.7 can be detected by evaluating the generalization error's trend. In fact, training a model with a dataset composed by a large set of items should reduce the error computed during the test, since the model sees a large set of images and learns how to appropriately classify them. Unfortunately, there might be some cases where the model recognizes a large quantity of data belonging to the train set, but can be no longer able to recognize images contained in the test set: the over-fitting has occurred. Significant effort has been expended to understand the generalization capabilities of classical Machine Learning models, for example, theoretical results have been formulated and constitute upper bounds on the generalization error as a function of the training data size and the model complexity[10]. Such bounds provide guidance about how much training data is required and/or sufficient to achieve accurate generalization. Moving into the QML field, little is known about the conditions needed for accurate generalization and there is no theoretical explanation yet.
Naively, one could expect that an exponential number of training points are needed when training a function acting on an exponentially large Hilbert space, but this is a concerning hypothesis, since it would imply exponential scaling of the resources required for QML, which is precisely what the field of quantum computation would like to avoid[10]. In the report, authors focused on two main tasks. They started to train a unitary gate to represent a large number of data, showing that is possible to train an arbitrary polynomial-depth unitary gate that can be efficiently implemented on a quantum computer. More generally, one could consider a QML model with T parameterized quantum gates and relate the training data size N needed for generalization to T.

[1]The difference between the model's performance on the true data distribution and the performance estimated from our training data is called the generalization error, and it indicates how well the model has learned to generalize to unseen data.

More specifically, they proved highly general theoretical bounds on the generalization error that can be upper bounded by the quantity

$$\epsilon_{gen} = \sqrt{\frac{T}{N}}, \tag{3.1}$$

where T is the number of parameterized gates and N are the data points used. They showed that generalization improves if only some parameters have undergone substantial change during the optimization. Even if they used a number of parameters T larger than the training data size N, the QML model could still generalize well if only some of the parameters have changed significantly. To showcase these results, in the second task they considered a quantum convolutional neural network (QCNN) that has only $T = O(\log(n))$ parameters and yet it is capable of classifying quantum states into distinct phases. Their theory guarantees that QCNNs have good generalization error for quantum phase recognition with only polylogarithmic training resources[10], $N \in O(\log(n))$, supporting the primary goal of quantum computation that is reducing the number of computational resources to perform the simulations. At support to this statement, a numerical demonstration suggests that even constant-size training data can suffice. However, the only problem that get in the way in the development of an experiment supporting their work, is the fact that the data used were already quantum states, so the more impact and delicate phase where classical data are represented as quantum data is absent in the work. Indeed, is not possible to fully quantify the good predictions of the model and to relate their results with ours completely.

Chapter 4

Methods

4.1 Feature map

In this section, we illustrate the multiple architectures used to perform the image classification. We start with a description of the feature maps that are tested with a deep and entangling angle embedding scheme, and then get to the architectures used as ansatz, where we used a shallower number of quantum gates in order to make it possible to run on currently available hardware.

4.1.1 Ring-like architecture

The first QNN architecture we tested is the **Ring-like** model that is composed of 6 qubits and the same quantum circuit for both the feature map and the ansatz. The number of features and learning parameters that are fed into the two model's components are different: the feature map encodes 256 features while the ansatz only has 36 weight parameters. We design the circuits to exploit the angle embedding scheme as it is easy to implement and is more efficient to be executed on a quantum hardware. We apply a parameterized R_y gate to each qubit, then we entangle a qubit with its neighbor until the last one, which has been entangled with the first, closing into a ring. After the entanglement scheme, we apply other R_y quantum gates the entangling apparatus reverse routed. This set of operations characterizes a quantum layer that is repeated 22 times for the feature map and only 3 times for the ansatz. This architecture is a variation of data re-uploading illustrated in [31]. Encoding 12 features per layer and repeating each process 22 times gives an estimated number of features equal to 264. We drop the extra features by making an initial guess: we set the first eight parameters to be $\pi/2$ and they will be updated in the successive iterations of the training process. The motivation that induced us to use this specific quantum circuit relies on [38]. Introduced within the article, the concept of **expressibility** is presented as a circuit's ability to

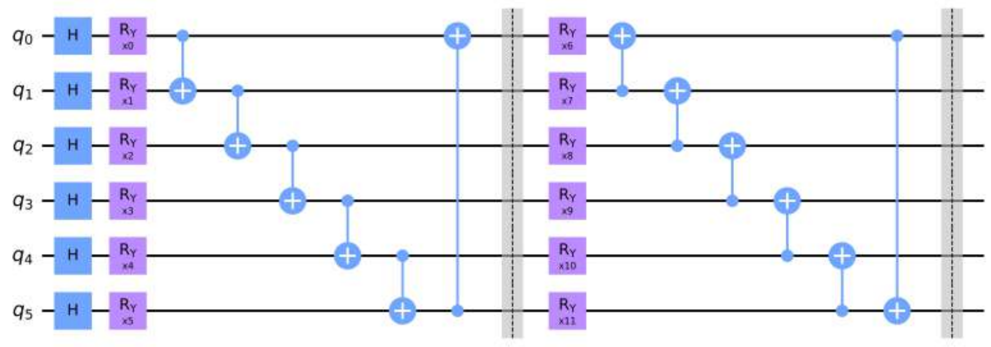


Figure 4.1: The Ring-like quantum circuit is used as a feature map. In this picture, only the first part of the whole circuit has been depicted. A part from the Hadamard gates that create superposition, the structure of each layer includes parameterized $R_Y(x)$ rotations along the angle correspondent to each feature, five CNOT between a qubit and its next one and reversed entangling gate between the last and the first qubit. After, another sequence of rotations is applied to each qubit. Each layer is composed of 12 features and the current scheme is repeated a number of times to encode all the required features. With this approach and with all the following, we have a unitary gate that encodes one feature.

generate (pure) states and in the case of a single qubit, this corresponds to a circuit's ability to explore the Bloch sphere[38]. On the other side, the ability of a circuit to generate entangled states is referred to as the **entangling capability**. In the context of variational quantum algorithms, potential advantages of generating highly entangled states with low-depth circuits include the ability to efficiently represent the solution space for tasks such as ground state preparation or data classification and result in capture non-trivial correlation within quantum data, can offer a potential advantage.

4.1.2 Waterfall architecture

The **Waterfall** architecture corresponds to a quantum circuit made of 6 qubits fully entangled one with another, given a specific qubit, its entanglement is shared with the successive qubits, and we will refer to Waterfall architecture. The idea of exploiting this kind of architecture comes from [37] where it has been introduced in experiments aimed to test the performance of a hybrid approach composed by a quantum and a classical neural network. In the article, authors presented the **Real Amplitude**[37][5] circuit, reduced to only four qubits, to realize the embedding of many resized image batches. In that case, the performance has effectively improved but is again affected by the sequence of classical convolutional filters that reduced the number of features and simplified the data fed into the quantum circuit before the classification. The hypothesis that has been advanced in support of the waterfall architecture relies on the increasing level of entanglement the circuit creates. We wonder if the circuit can catch finer details, producing more correlations between pixels, which may result in a higher performance. On the opposite way, the considerably increased amount of entanglement could generate a more difficult quantum state to learn by the ansatz since has only 3 layers against the 22 of the feature map.

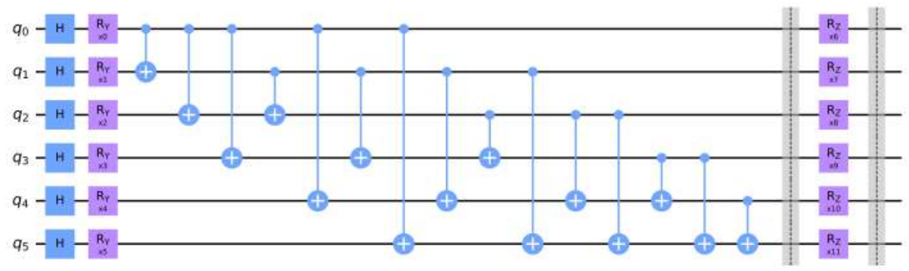


Figure 4.2: Waterfall entangling feature map. As it can seen, the entanglement is realized between one qubit and all the others following a waterfall. A parameterized R_Y gate is applied to each qubit and the parameters of these unitaries are the first d parameters of the ansatz. Subsequently, two-qubit CNOT gates are inserted in the middle of the circuit and connect a qubit with all the others. Finally, each qubit is subjected to another parameterized R_Z gate. The parameters of these gates are the second d parameters of the ansatz.

4.2 Variational ansatz

4.2.1 Mixing architectures

Since we built two different architectures composed of a couple of circuits each, we decided to mix them to observe the performances. We started to use the Ring-like quantum scheme as a feature map composed of 22 layers together with the Waterfall qubits arrangement as the variational ansatz with 3 layers, i.e. 36 learning weights. By following the same approach we used also the Waterfall quantum circuit as a feature map and the Ring-like circuit as a variational ansatz with the same number of layers and learning parameters. After testing these mixed architectures, we added quantum variational ansatz in the experiment that was used in [39] and it is presented as a hardware-efficient quantum circuit as we would describe below in 4.2.2. After that, we are going to introduce in 4.2.3 a simple quantum version of the classical CNN, i.e. the QCNN, to report the results of the generalization.

4.2.2 CZ circuit

Another circuit used to implement a variational ansatz to connect with the above-mentioned feature maps is the CZ circuit. This hardware-efficient quantum circuit is known to be highly expressive, and is susceptible to the barren plateau described in 2.2.4 phenomenon for a large number of qubits and layers. In each layer all the qubits undergo through parameterized rotations along Y and Z directions, i.e, R_Y, R_Z and a chain of CZ gates[39] generates the entanglement in a similar way to the Ring-like architecture. An important difference concerning 4.1, apart from the specific entangling quantum gates used to correlate data, is the fact that in the Ring-like quantum circuit, we apply a first column of rotations then, once the first weights are learned, they are subsequently entangled with the successive ones coming from the second column, while for the CZ circuit all the weights are learned and the entanglement is realized between layers.

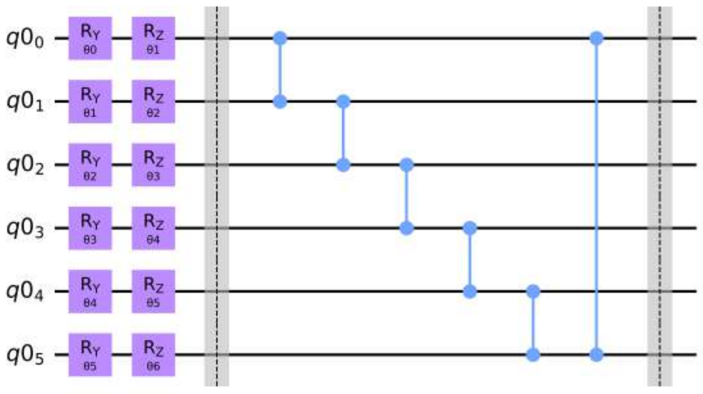


Figure 4.3: CZ variational ansatz. This time the entanglement is placed at the end of the two columns where parameterized quantum gates are applied. After two quantum gates, respectively R_Y and R_Z rotations applied to each qubit, the entanglement is realized with controlled-Z gates following the same ring scheme encountered for the Ring-like quantum circuit.

4.2.3 QCNN-like circuit

The next scheme we tested is represented by an 8-qubit quantum circuit of quantum gates such as R_Y, R_Z, and a controlled version of the two. This different approach exploits the fact that, once a layer is applied, the number of qubits used in the next one is systematically reduced by a factor 2. In particular, we composed the current architecture with 4 layers, the first of which starts with 8 qubits, the second has 4 qubits, the third layer has 2 and the last one has only one qubit remaining. The same function provided by Qiskit has been used to obtain the result of a general measurement performed on all qubits that works out the probabilities of detecting a specific class of images. It can be seen from the figure below 4.4 that a similar *IsingZZ* entangling gate has been realized and acts as a source of entanglement between qubits. In this new approach inspired by classical CNNs, all the information acquired by the variational ansatz is collected in only one qubit, the last one that should be the most representative of the corresponding quantum state generated. Repeating the same logic, we tested the QCNN variational ansatz both with the Waterfall and Ring-like feature map to see which results in the best quantum circuit combination.

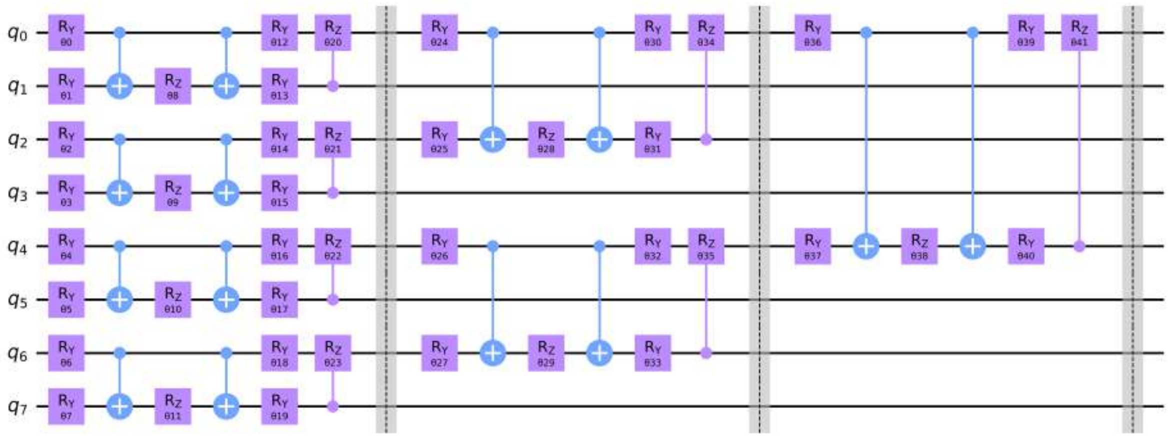


Figure 4.4: QCNN variational ansatz in which we can observe four layers where the number of parameterized quantum gates is reduced by a factor of 2. The entangling gates we have used in this approach are constituted by the *IsingZZ* two-qubits gate, where two CNOTs wrap a R_Y rotation gate applied on the next wire. Starting from 8 qubits we end up with only one over which is performed the last R_Y rotation. In this particular architecture all the 8 qubits are then measured, even if the idea behind this circuit is to measure only the last remained qubit in order to extract the resulting quantum state directly from it. In the Appendix D.1 we display the results where two different sets of qubits are measured. In the first experiment we collect the result out from all the 8 qubits (and this is one of the reasons why the process took several time, i.e. the resulting state is of the order of 2^8) and from the last single-qubit (in which the measured state is of the form 2^1).

4.2.4 Characteristics

In the figure 4.5 we show the trend of the size of each ansatz we have used to make the predictions. It explains how the circuit's size, defined as the number of gates applied to a specific quantum circuit, varies as more layers are added to the circuit. The QNN ansatzes manifest a linear behaviour of the size function S as more layers are added to the circuit. In fact, working with a large number of features, each variational ansatz requires more layers, i.e. more unitary gates applied to it in order to increase the number of optimizing parameters (weights) used to learn the characteristic that distinguish each image that has been represented by the feature map as a quantum state. Totally different behaviour can be seen for the QCNN-like ansatz, whose number of gates scales logarithmically. By reducing the number of qubits considered within each layer by a fac-

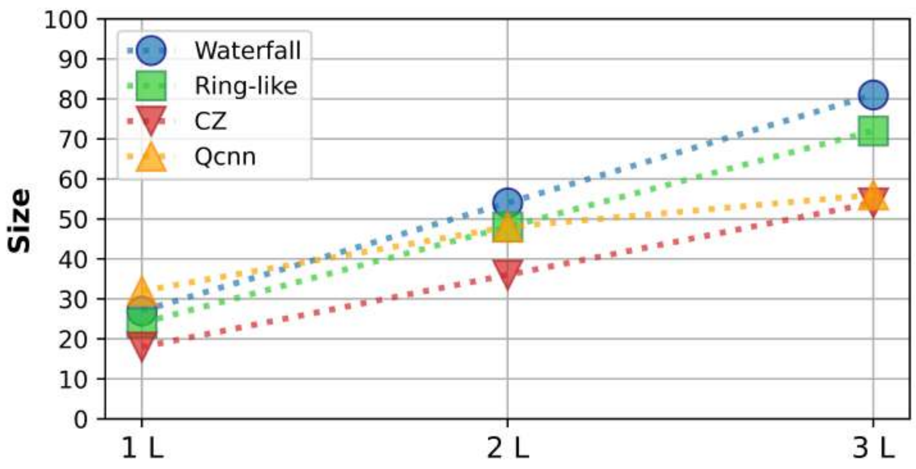


Figure 4.5: Progression of the number of quantum gates as a function of the number of layers a circuit is composed by, denoted as the **size** of a quantum circuit. Note that the QCNN variational ansatz shows a logarithmic behaviour due to the progressively reduction of the number of qubits in each layer by a factor of 2.

tor of 2 translates to a base-2 logarithmic behaviour for the circuit's size. With the help of support instruments we could in principle expand the variational ansatz by adding additional layers and emphasize this behaviour.

Chapter 5

Settings and experiment

5.1 Experimental settings

5.1.1 Framework

Quantum scheme

To perform the simulation we used different quantum/classical tools. In principle we selected *Qiskit* as the candidate to build the quantum circuits and, more generally, to train the quantum neural network. Qiskit is an open source Python library developed by IBM mostly used in Quantum simulations, Quantum Machine Learning, Quantum Compiling, Quantum Optimization etc. Qiskit allows developers to simulate the quantum circuit directly on IBM machines to have an idea about the behaviour of a quantum circuit, if executed on a real quantum hardware. In this way people can be aware of the errors that occur during the simulation process. Errors in fact, acquire a significant weight in Quantum Computing applications since they cannot be separated completely, and a-priori, from any experiment that uses quantum algorithms. Even if a possible way of circumvent the problem exists and relies in simulating a specific quantum circuit with a zero-noise simulator (like the *aer-simulator*), real existing quantum devices are not free from errors, indeed are susceptible to noise because of physical disturbances. The error we are talking about is an intrinsic property of matter that cannot be eliminated: qubits in fact, are quantum systems highly sensible to the environment. For instance, if some molecule in the surrounding air comes into contact with qubit it would transfer some kinetic energy and potentially affect the generated quantum state. Another example involves the interference between adjacent qubits; if they collide into one another their wave-functions can constructively or destructively interfere to give an odd result. This is the reason why adding more qubits into the system greatly increases the noise. There's much higher chances of a qubit "spilling over" onto another one because there's more energy contained. One of the difficulties that accompanied the development of quantum computers is the source of error that generates from its constituents and a process aims to clean the effects of this noise, known as Error Correction, is necessary.

Classical scheme

The rest of the simulation exploits suitable and known classical tools such as *Tensorflow* and *PyTorch*. ML tasks frequently leverage each library because they house functions designed to manipulate tensors representing image data. These libraries also contain a variety of built-in functions, including loss functions, training and testing evaluation functions, facilitating the classification of computed output against corresponding labels. These tools are known for their simplicity and for the numerous tutorials and documentation they are equipped with, making them accessible even to those not well-versed in the field. Furthermore, these libraries offer a wealth of ready-to-use datasets that can be easily downloaded, imported, and employed in various applications. We used *Tensorflow* for the data preparation part of the classification over the MNIST dataset. *PyTorch* has been used to perform the matching between output and corresponding label by the loss minimization. For the classical CNN we used Keras, the most popular library for ML for image classification, contained within *Tensorflow*. Keras was suitable to create a very simple and fair convolutional neural network with a number of parameters comparable with the corresponding quantum ones.

5.1.2 Dataset

MNIST

The first dataset used to carry out the Machine Learning task is the MNIST. It consists of thousands of gray-scale figures representing handwritten digits from 0 to 9 of 28x28 features. The advantages of using MNIST are multiples:

- It's a well-studied toy dataset suitable for image-recognition tasks;
- We have a large quantity of images available (60000 for training, 10000 for testing) that permits to select images not repeating images belonged to the same class;
- It's already splitted in train and test sets, in such a way that the images contained in the train set are not present also in the test set.

In the pre-processing part, only 0 and 1 digit-images have been selected to build the train and test sets. In a second moment, we have reduced the number of features from 784 (28x28 images) to 256 (16x16 images) to facilitate the next quantum encoding part and to use a small number of qubits for the circuit's creation. The features reduction has been done internally to a function that first expand the dimension of each matrix-object that represent the image, by adding an additional dimension (usually images are represented as tensors whose dimensions are (feature, feature, channel), where we could have in principle 1 channel for gray-scale images or 3 channels for RGB images) to each picture, and subsequently uses the *resize* functionality of Tensorflow to scale the features to the desired number. We are going to further reduce the features of the train and test sets to small numbers such as 12, 8 and 6, in order to see if there's an improvement when working with a number of optimizing parameters comparable to the small number of features used in the experiment.

GALAXY

The galaxy dataset includes gray-scale images representing different kind of galaxy that are recovered by a single image we have previously downloaded from Internet and that has been cropped in multiple items. With this strategy we have created a custom dataset consisting in images which contain a galaxy and others that do not contain any kind of subject, yielding to a binary dataset composed by two classes, 0 and 1. The figures are then down-sampled to have 256 pixels (16x16 features) in total. Due to the random cropping process, three main possible kinds of images have been formed: a type that contains a galaxy or at least a star, one that contains a partial galaxy or star, and the last one that does not contain any galaxy at all. A multi-class SL classification could be useless, since there are figures where the subject is not completely distinguished and three main classes can be identified, but for simplicity, we restrict to a binary classification problem where the pictures that contain a partial galaxy are considered as belonging to the same class.
Differently from what has been done for MNIST dataset, a binarization function, whose effects are displayed in BB.5, has been applied to each figure in order to clean and correct pixel values that create confusion. When lowering the number of features down to 256, the pixels' dimension increases until becoming significantly visible. Some images exhibit fractional pixels blending with the subject pixels that are a part of the galaxy. This phenomenon leads to a degradation in the overall quality of the image and can constitute a problem for the QNNs. To avoid this apparent problem we decided to set a minimal threshold - a specific pixel value equals to 0.25 - below which each pixel is set to 0 giving a black patch-square at the corresponding place. In this way we have removed the area that mix with the object contained inside the image, leading to a more distinguishing picture.
Afterwards, a function has been devised to guarantee the balance between each class within the dataset prior to commencing the analysis. This approach eliminates potential bias introduced by the random generation of MNIST images, as we explicitly specify in advance the desired quantity of images for each of the two classes under consideration. Balancing a dataset is generally an important task to implement before the creation of the model, since if not present can significantly affect the model's performance.

5.1.3 Dataset preparation

After the data pre-processing, we thought about a common scheme to start the analysis. Since the idea behind this project is to evaluate the effects coming from training the QNN with a low number of data points, the way we proceed to test the performances consists of:

- Setting a number of data-points to train the model;
- Training and testing the model on the train and test dataset;
- Collecting the result of the test accuracy and repeat the process to increase the training data size.

We remind that this project has the scope to open new ways of investigation of such QNN architectures and to emphasize the possible gain in the accuracy when training a model with few data, thus the number of images chosen to perform the training is not as large as one could expect, furthermore, since simulations required an intensive use of computational resources, it has been decided to reduce the number of points to use for the process. Six training sets have been chosen and are composed of 6, 10, 20, 30, 40 and 50 image data. The test set instead consisted of 140 images for the Mnist dataset and only 100 for the Galaxy. This discrepancy is due to the way we extracted the images from the galaxy-field figure used as proxy. After selecting a set of images, the training process is iterated five times using an identical number of training data-points. However, in each iteration, a different seed is used to create a training set of the same length but with distinct images. In this way, we literally expand the train set every time we invoke a different seed value. For example, once we choose a seed, each of the different train sets contains the same data of the previous one plus random images to complete the dataset. The value of the test accuracy we obtain at the end of each training dataset is the average value over five seeds.

5.1.4 Metrics, loss, optimizer and interpret function

In this section we examine the metrics that characterize both quantum and classical Machine Learning problems.

5.1.5 Metrics

The confusion matrix is the first element that give us an estimation of how well the model performs the prediction. The evaluation of the performance of the model is defined by some metrics. The most important ones are the following: **Accuracy**, **Precision**, **Recall** and **F1-score**. To describe them, it can be useful to define the Confusion matrix, which is a matrix used for evaluating the performance of the model. In this matrix, the real labels are placed on the column, while the predicted ones on the rows. Then, the result of a binary classification (-1, 1) can lead to:

- **True Positive** (TP), if the data is correctly predicted as 1;
- **True Negative** (TN), if the data is correctly predicted as -1;
- **False Positive** (FP), if the model misses the prediction and classifies the point as 1;
- **False Negative** (FN), if the model misses the prediction and classifies the point as -1.

It is desired to obtain the highest possible value of True Positive and True Negative data, minimizing model errors. Roughly speaking, the ideal case is obtained when only the main diagonal of the Confusion matrix contains values, which means are correctly classified, while the off-diagonal elements should be 0.

Accuracy

The Accuracy of the model is the number of data-points well predicted over the entire set of elements evaluated.

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

This metric is useful to evaluate the model performance if the two classes of elements are appropriately balanced.

Precision

The Precision is the number of correctly predicted positives over the number of positive predictions.

$$Pr = \frac{TP}{TP + FP} \tag{5.2}$$

Recall

The Recall is the number of correctly predicted positive classes over the total positive.

$$Re = \frac{TP}{TP + FN} \tag{5.3}$$

F1-score

In some cases, the Precision and the Accuracy are not good metrics, especially for imbalanced datasets. For this reason, to choose the best model, F1-score can be defined as an additional metric to achieve the best trade-off between the Precision and the Accuracy. It is obtained as the harmonic mean of Precision and Recall. This metric is maximized if the Precision and Recall are similar to each other.

$$F1 = 2\frac{Pr * Re}{Pr + Re}. \tag{5.4}$$

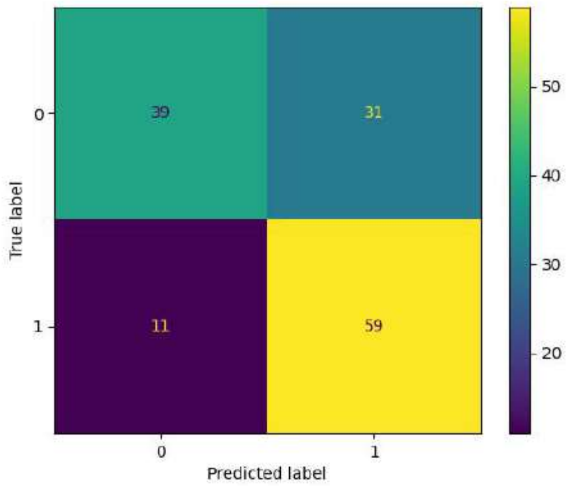


Figure 5.1: Example of a Confusion matrix (CM) calculated in the testing phase. Since the problem is a binary classification, the CM is a 2x2 square matrix. The first matrix element represents how many 0-digit images the model has correctly classified. The second element instead (with 31 displayed) represents the number of 0-digit images wrongly classified as 1-digit images. The third one instead says how many 1-digit images are misclassified as 0-digit images. The last element represents the number of 1-digit images correctly recognized. The optimal CM should have zero or less items in the off-diagonal elements meaning the model has been able to correctly classify everything he has seen.

5.1.6 Loss

Loss functions quantify the error between the predicted class of a sample and the actual class. This can be done in several ways, which creates different loss landscapes. These loss landscapes have different shapes and curves for example they could be flat plateaus, local minima, and global minima, hills, ravines, etc[16]. Even though there are several loss functions that can be used during the training, in this little section we are going to present the one adopted for the binary classification problem.

Cross-entropy loss

Cross-entropy Loss is used for those classifications in which the result of the model is a probability value. For binary classification, in which the classes are 0 and 1, it is:

$$L = -y \log p - (1 - y) \log(1 - p). \tag{5.5}$$

where p is a probability distribution representative of the events we are measuring, while y are the actual class present in the problem.

5.1.7 Optimizer

The task of the optimizer is to find updates to the parameters based on the outcome of the loss function in such a way that, after repeated runs, the loss is minimized[16]. The optimizer used for the training of the quantum neural network is the Limited-memory BFGS (L-BFGS or LM-BFGS). LBFGS is an optimization algorithm in the family of quasi-Newton methods that approximates the Broyden–Fletcher–Goldfarb–Shanno algorithm (BFGS) using a limited amount of computer memory. It is a popular algorithm for parameter estimation in ML. The algorithm's target problem is to minimize $f(\mathbf{x})$ over unconstrained values of the real-vector $\mathbf{x}$ where f is a differentiable scalar function. Like the original BFGS, L-BFGS uses an estimate of the inverse Hessian matrix to steer its search through variable space, but storing only a few vectors that represent the approximation implicitly. Due to its resulting linear memory requirement, the L-BFGS method is particularly well suited for optimization problems with many variables.

5.1.8 Interpret function

The measurement process is a necessary step to fulfill in order to evaluate the cost function. In this way the model learns how to classify data and the accuracy can be finally extracted. Such measurement operation is applied directly by a specific built-in Qiskit's class that has been used, i.e. *CircuitQNN*, during the entire work. It samples a neural network given a quantum circuit which is composed by the feature map and the variational ansatz together. It takes in parameters such as the *input params* (feature map quantum features), *weight params* (variational ansatz's learning parameters) and *output shape* which is the shape of the computed result, that is a (1,2) array formed by two different probabilities of interpreting the image as belonging to a 0 or a 1 class. The last parameter we focus on is the **interpret function**, here the parity function is used. The parity interpret function is used to convert the output state in binary strings and counts the number of 1s present in the string and performs a modulo - division -, to get weather the binary representation of a specific state is even or odd. In this case, the parity function maps a quantum state formed by a sequence of 0s and 1s to an integer value, i.e. to a class index. The strategy adopted to evaluate the interpret function is the following. It has been used as a standard form of the parity function displayed below

```
lambda x: "{:b}".format(x).count("1") %2
```

for all the QNN architecture except for the QCNN model. For this kind of architecture we decided to use two types of interpret functions: the first one remains unchanged since all the qubits are measured. The other one instead is simply an identity interpret function as

```
def interpret(x):
return lambda x: x
```

This choice is justified by the fact that measuring only one qubit we can access only two states, i.e. $|0\rangle$ and $|1\rangle$, the parity function will be useless because it will no longer able to count how many 0s or 1s are present in the final quantum state this time is a single quantum state. So the function will be simply an identity.

5.2 Results

The goal of the analysis is to test the performances of different quantum architectures by varying the number of training images as to look for an improvement in learning and generalization. Once the experiment has been done for the quantum neural networks, we focus on the classical model and use it as a direct comparison. In principle, we expect the classical model to be able to surpass the the quantum counterpart since the number of parameters that go under optimization is significantly higher than what has been used for QNNs. In addition, adding more parameters to the variational quantum circuit could turn into a non-trivial process for the simulation itself, due to the noise that could be generated from the high-depth quantum circuit. To address this obstacle, we built a simple classical model ensuring the number of parameters equals or, at least of the same order, of the number of quantum weights the ansatz circuit learns during each epoch. In this way, we kept the challenge as fair as possible. Anyway, quantum models are still in-development and they are not able to compete against the already established classical models yet; this is another reason for which the investigation is gaining much more attention on QNNs this time. The experiments conducted on the MNIST dataset have seen a decrease in the number of images since the second set, the Galaxy set, was generated with fewer images compared to its predecessor. The data in this set were obtained by cropping existing initial pictures as we said, making it less suitable for use as a genuine testing dataset. Despite this limitation, we have chosen to present the results for the Galaxy set because we find it intriguing to observe the variations that arise when different sets of images are taken into consideration, even though we have focused much more on the MNIST for the reasons we have explained in 5.1.2.

5.2.1 Generalization with few images

The first task of the project is based on the following idea: train each model with an increasing number of training data and collect the corresponding result coming from either the train and test accuracy, in order to study the performances. Each training set is placed on the x axis while the corresponding y value (train and test accuracy) is represented as a dot-marker. In the course of the process we tested all the models by choosing an initial training set consisting of 6 images, that is a small and challenging number of items. Afterwards, we enlarged it to 10, 20, 30, 40 until 50 items. Few considerations can be done already at this point. First, we decided to use only six training sets in account of two main reasons: all the simulations took a considerable time until they finished since we run multiple low and high depth circuits with the *aer-simulator* on an IBM machine and generally is not a quick process especially when numerous unitaries are applied within the circuit. Indeed, we have forced the circuits to learn from very few number of training data. Secondly, we repeated the same process five times for each model, every time using a different train set as to plot the average value and the standard deviation computed on these different runs. Due to this last way of proceeding, the task itself was really time and computational consuming. The tables that summarize the outputs of the numerous simulations we have performed are placed below. Each column contains the current **Dataset** that has been used for the simulation, the number of **Data-points** we selected for training, the name of the **Model** we tested, the number of parameters (**Params.**) the variational ansatz optimizes and the corresponding **Train acc.** and **Test acc.**, i.e. train and test accuracy, respectively.

5.2.2 Tables of results: MNIST

Quantum Ring-like (RL) model

Ring-like model's performances					
Dataset	**Data-points**	**Model**	**Params.**	**Train acc.**	**Test acc.**
Mnist	6	RLRL1	12	**0.86 ±0.19**	0.59±0.06
Mnist	10	RLRL1	12	0.74±0.18	0.57±0.11
Mnist	20	RLRL1	12	0.82±0.06	0.68±0.04
Mnist	30	RLRL1	12	0.80±0.04	**0.70±0.05**
Mnist	40	RLRL1	12	0.60±0.07	0.52±0.04
Mnist	50	RLRL1	12	0.71±0.14	0.69±0.11
Mnist	6	RLRL2	24	0.90±0.20	0.61±0.09
Mnist	10	RLRL2	24	**0.96±0.05**	0.65±0.10
Mnist	20	RLRL2	24	0.84±0.10	0.71±0.07
Mnist	30	RLRL2	24	0.82±0.07	0.71±0.05
Mnist	40	RLRL2	24	0.83±0.04	0.73±0.03
Mnist	50	RLRL2	24	0.86±0.03	**0.76±0.01**
Mnist	6	RLRL3	36	0.96±0.60	0.70±0.04
Mnist	10	RLRL3	36	**1.00±0.00**	0.68±0.05
Mnist	20	RLRL3	36	0.95±0.09	0.73±0.02
Mnist	30	RLRL3	36	0.92±0.04	0.72±0.01
Mnist	40	RLRL3	36	0.84±0.09	0.75±0.06
Mnist	50	RLRL3	36	0.89±0.03	**0.77±0.03**

The Ring-like architecture reaches good levels of train accuracy, while the test accuracy remains just under the 80%. Anyway, it is possible to notice the progressive increment in the test accuracy when more layers are considered in the circuit (leading to more parameters) and, on average, when more images are used to train the model. This particular behavior can be explained as follows: the feature map has as many quantum gates as the number of features of each image. Thus, increasing the number of gates used in the variational ansatz can significantly improve the performance on train and test accuracy since more parameters are learned at each iteration. It would be interesting to add more additional layers to the circuit and perform the same analysis looking for further improvement. It should be remarked that the model has been trained with few images that have a large number of features that could be hard to learn by using a significant reduced number of optimizing parameters. There should be a trade-off between the depth of the feature map that encode all the necessary features an image is composed of, and the depth of the variational ansatz that should be able to represent and classify those images.

Waterfall model

Waterfall model's performances					
Dataset	**Data-points**	**Model**	**Params.**	**Train acc.**	**Test acc.**
Mnist	6	WFWF1	12	0.63 ±0.24	0.60±0.13
Mnist	10	WFWF1	12	0.64±0.12	0.57±0.13
Mnist	20	WFWF1	12	0.60±0.19	**0.67±0.11**
Mnist	30	WFWF1	12	0.66±0.07	0.67±0.04
Mnist	40	WFWF1	12	0.61±0.07	0.61±0.02
Mnist	50	WFWF1	12	**0.68±0.06**	0.65±0.03
Mnist	6	WFWF2	24	**0.77±0.14**	**0.69±0.03**
Mnist	10	WFWF2	24	0.70±0.09	0.62±0.14
Mnist	20	WFWF2	24	0.72±0.04	0.63±0.06
Mnist	30	WFWF2	24	0.75±0.03	0.65±0.02
Mnist	40	WFWF2	24	0.77±0.03	0.63±0.07
Mnist	50	WFWF2	24	0.76±0.08	0.67±0.03
Mnist	6	WFWF3	36	0.90±0.13	0.64±0.08
Mnist	10	WFWF3	36	**0.86±0.08**	**0.70±0.05**
Mnist	20	WFWF3	36	0.59±0.15	0.59±0.11
Mnist	30	WFWF3	36	0.67±0.08	0.67±0.03
Mnist	40	WFWF3	36	0.67±0.15	0.64±0.10
Mnist	50	WFWF3	36	0.67±0.05	0.68±0.05

These are the performances of the Waterfall architecture. Compared to the previous one, the values reached during the generalization are lower together with the train accuracy values. We can observe the progressively increasing level of both train and test accuracy when a more deep circuit is considered, even though the the performances over the test set are still under 70%.
In the next page we observe the tables for the QCNN and CZ models. The combination between the WF feature map and the QCNN variational ansatz outcomes promising results with respect to the same variational ansatz combined to RL feature map. On the contrary, the CZ shows better results when is combined with the RL circuit, especially when a large set of images is used for training.

QCNN with the measurement of all the qubits

QCNN model's performances					
Dataset	**Data-points**	**Model**	**Params.**	**Train acc.**	**Test acc.**
Mnist	6	WFQCNN	42	**0.90±0.13**	0.72±0.04
Mnist	10	WFQCNN	42	0.76±0.10	0.65±0.05
Mnist	20	WFQCNN	42	0.76±0.09	0.65±0.06
Mnist	30	WFQCNN	42	0.72±0.15	0.64±0.03
Mnist	40	WFQCNN	42	0.77±0.04	**0.76±0.03**
Mnist	50	WFQCNN	42	0.74±0.05	0.72±0.03
Mnist	6	RLQCNN	42	**0.87±0.12**	0.54±0.05
Mnist	10	RLQCNN	42	0.80±0.20	0.55±0.10
Mnist	20	RLQCNN	42	0.74±0.15	0.64±0.07
Mnist	30	RLQCNN	42	0.74±0.07	0.62±0.04
Mnist	40	RLQCNN	42	0.73±0.08	0.65±0.06
Mnist	50	RLQCNN	42	0.69±0.13	**0.70±0.12**

CZ circuit

CZ model's performances					
Dataset	**Data-points**	**Model**	**Params.**	**Train acc.**	**Test acc.**
Mnist	6	WFCZ	36	**0.94±0.08**	0.60±0.07
Mnist	10	WFCZ	36	0.88±0.11	0.66±0.09
Mnist	20	WFCZ	36	0.82±0.05	**0.68±0.03**
Mnist	30	WFCZ	36	0.71±0.10	0.61±0.10
Mnist	40	WFCZ	36	0.78±0.08	0.66±0.03
Mnist	50	WFCZ	36	0.68±0.12	0.61±0.10
Mnist	6	RLCZ	36	0.83±0.27	0.62±0.07
Mnist	10	RLCZ	36	0.78±0.32	0.64±0.08
Mnist	20	RLCZ	36	**0.91±0.02**	0.73±0.05
Mnist	30	RLCZ	36	0.83±0.07	0.71±0.04
Mnist	40	RLCZ	36	**0.91±0.02**	0.73±0.05
Mnist	50	RLCZ	36	0.80±0.03	**0.75±0.02**

Mixed models's performances					
Dataset	**Data-points**	**Model**	**Params.**	**Train acc.**	**Test acc.**
Mnist	6	RLWF1	12	0.73 ±0.22	0.63±0.02
Mnist	10	RLWF1	12	**0.74±0.16**	0.57±0.10
Mnist	20	RLWF1	12	0.64±0.10	0.59±0.06
Mnist	30	RLWF1	12	0.68±0.06	0.64±0.03
Mnist	40	RLWF1	12	0.64±0.06	**0.65±0.01**
Mnist	50	RLWF1	12	0.65±0.08	0.61±0.09
Mnist	6	RLWF2	24	**0.83±0.15**	0.60±0.05
Mnist	10	RLWF2	24	0.70±0.09	0.58±0.03
Mnist	20	RLWF2	24	0.62±0.06	0.53±0.03
Mnist	30	RLWF2	24	0.71±0.11	0.59±0.06
Mnist	40	RLWF2	24	0.72±0.08	0.63±0.04
Mnist	50	RLWF2	24	0.69±0.07	**0.65±0.03**
Mnist	6	RLWF3	36	**0.86±0.12**	0.63±0.04
Mnist	10	RLWF3	36	0.78±0.04	0.62±0.04
Mnist	20	RLWF3	36	0.79±0.06	0.66±0.03
Mnist	30	RLWF3	36	0.76±0.06	0.66±0.03
Mnist	40	RLWF3	36	0.75±0.04	**0.67±0.04**
Mnist	50	RLWF3	36	0.74±0.05	**0.67±0.01**
Mnist	6	WFRL1	12	**0.80±0.12**	0.64±0.02
Mnist	10	WFRL1	12	0.70±0.14	0.62±0.03
Mnist	20	WFRL1	12	**0.80±0.04**	0.63±0.02
Mnist	30	WFRL1	12	0.69±0.09	0.63±0.10
Mnist	40	WFRL1	12	0.76±0.06	**0.64±0.02**
Mnist	50	WFRL1	12	0.71±0.06	0.63±0.01
Mnist	6	WFRL2	24	**1.00±0.00**	**0.69±0.03**
Mnist	10	WFRL2	24	0.82±0.22	0.64±0.15
Mnist	20	WFRL2	24	0.87±0.05	0.68±0.02
Mnist	30	WFRL2	24	0.78±0.06	0.67±0.04
Mnist	40	WFRL2	24	0.73±0.09	0.65±0.04
Mnist	50	WFRL2	24	0.69±0.19	0.61±0.13
Mnist	6	WFRL3	36	**1.00±0.00**	0.71±0.05
Mnist	10	WFRL3	36	0.94±0.05	0.67±0.04
Mnist	20	WFRL3	36	0.77±0.17	0.67±0.08
Mnist	30	WFRL3	36	0.89±0.06	**0.74±0.06**
Mnist	40	WFRL3	36	0.83±0.08	**0.74±0.07**
Mnist	50	WFRL3	36	0.90±0.05	0.70±0.04

5.2.3 Figures

In this section we report only the figures that refer to the circuit composed by a 3-layers variational ansatz, since we consider them as the architectures from which the best results could have been obtained for the current experiment, where we use the largest number of variational parameters. Further graphs are shown in the Appendix E. The table 5.2.2 shows the trend of the accuracy when the number of items contained in each train set increases. RLRL-1 LAYER presents an oscillatory behaviour and it does not reach a high test accuracy even with more images. The reason for which this happens probably relies in the number of layers used by the variational ansatz to learn the parameters: it seems they are not sufficient, compared to the number of layers used in the feature map, to learn all the features that distinguish each image and to make a correct classification. Indeed, more parameters are needed to improve its performance. A common behaviour

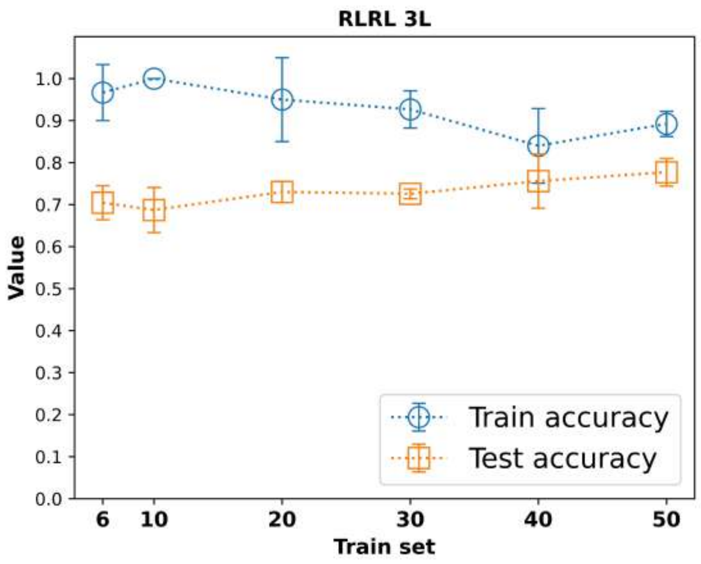


Figure 5.2: RLRL 3 layers architecture. The trend exhibited by the RLRL-3 LAYERS approaches the 80% of accuracy on the test evaluation and probably it could increase for larger training sets, much more compared to the previous two.

that is visible from the tables is the over-fitting phenomena generated when the model has been trained with only few images. As a consequence, the model is not able to recognize anything it has not seen. Despite the over-fitting's occurrence, the RLRL-2 LAYERS model, as visible in the table, seems to reduce its effect faster than the others and, surprisingly, with less optimizing parameters is able to reach a value of test accuracy comparable with the 3-layers model for large data-points.

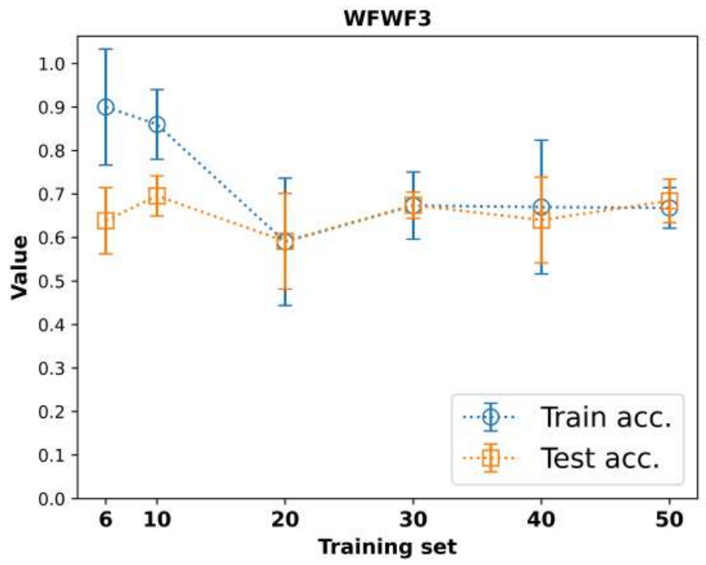


Figure 5.3: WFWF model with 3 layers. Apart from an initial over-fitting, it shows a good convergence between train and test accuracy already with 20 images. It is visible from the table 5.2.2 that the Waterfall model with only 1 layer converges to lower values of test accuracy with respect to other models, even though the over-fitting is considerably reduced. The high level of entanglement present in this circuit seems to introduce too much complexity, between the 256 features passed by the feature map, and the low number of parameters learned by the variational ansatz. The two following models perform better than the previous one already with less data and the one depicted in the figure 5.3 is able to converge to 70%.

Mixed architectures

The subsequent figures illustrate the mixed architectures composed by RL and WF quantum circuits combined together with 3 layers per variational ansatz. In the figure below 5.4 the Ring-like quantum circuit has been used to encode the features while the Waterfall as the variational ansatz with 3 layers. The next figure 5.5 shows the results for the reversed architecture that has been composed by the Waterfall circuit as feature map and the Ring-like as variational ansatz. These last three quantum neural networks present inside the table 5.2.2 (respectively composed by 1, 2 and 3 layers per variational ansatz) provide probably the lowest results so far. If for the WFRL-1 Layer the test accuracy does not improve over number of data, for the subsequent model it really decreases for 50 images.

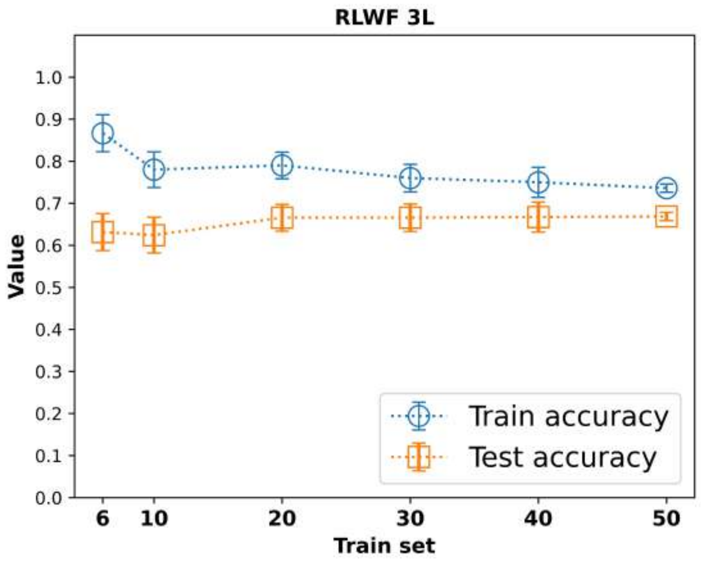


Figure 5.4: RLWF 3 layers model. We observe an almost flat trend of the test accuracy reflecting the slow decline of the train accuracy, meaning that e meaningful learning is absent.

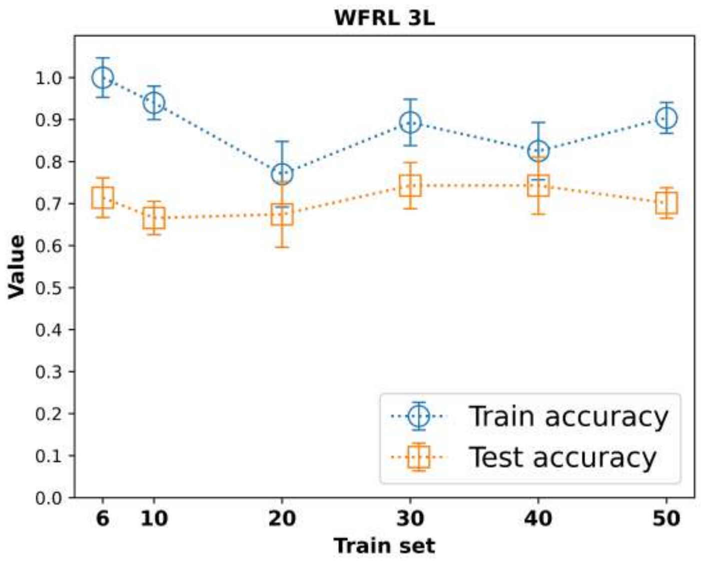


Figure 5.5: WFRL model combining a Waterfall feature map with 3 layers Ring-like architecture. The test accuracy starts from a higher value compared to the previous architecture, but we cannot see a significant improvement.

The next two figures represented in 5.6 and 5.7 displays the performances of the QCNN variational ansatz combined respectively with the Ring-like and the Waterfall feature maps. The first model in 5.6 presents an increasing accuracy on the test set, and seems able to grow-up for more data-points. In both cases the over-fitting is vanishing when more images are seen by the two models. For the WF-QCNN there is an overlapping between train and test accuracy, leading to hypothesize the train accuracy as a good predictor for the test accuracy. The QCNN variational ansatz seems to reach good results independently from the encoding circuit and could be considered another suitable candidate for image recognition experiments.

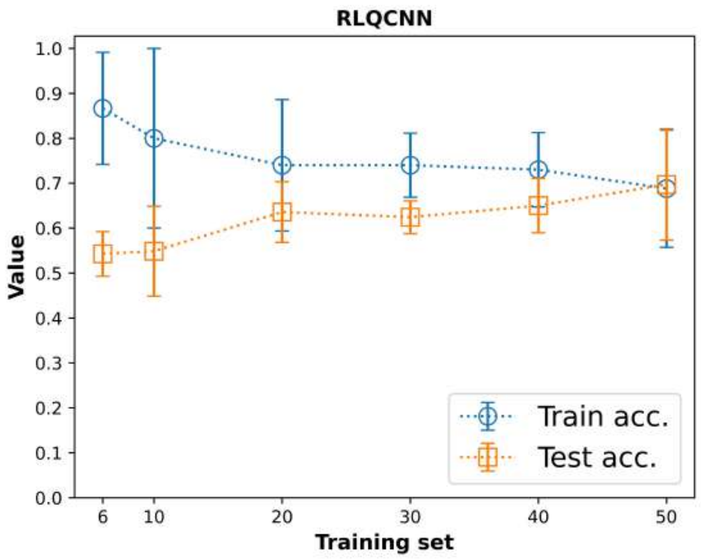


Figure 5.6: Ring-like feature of 22 layers map combined with the QCNN variational ansatz where the test accuracy seems to reach promising thresholds when the train set is enlarged.

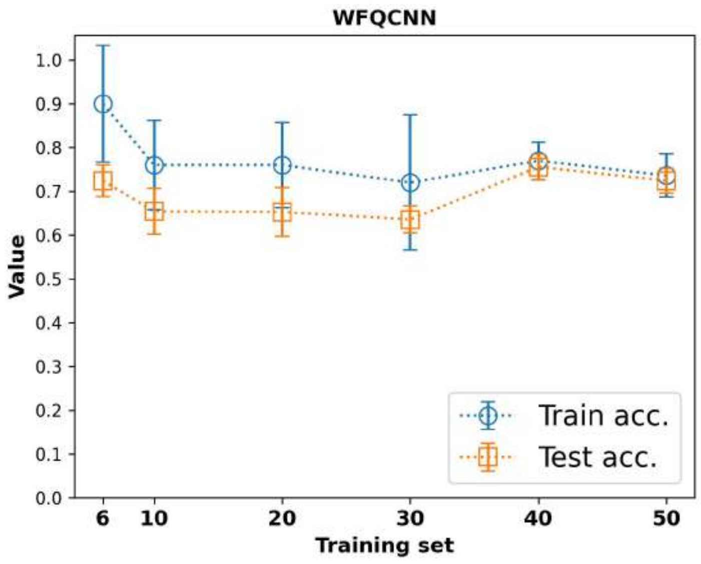


Figure 5.7: Waterfall feature map composed of 22 layers combined with the QCNN variational ansatz. After a faint peek corresponding to 40 images, the test accuracy decreases until 50 images.

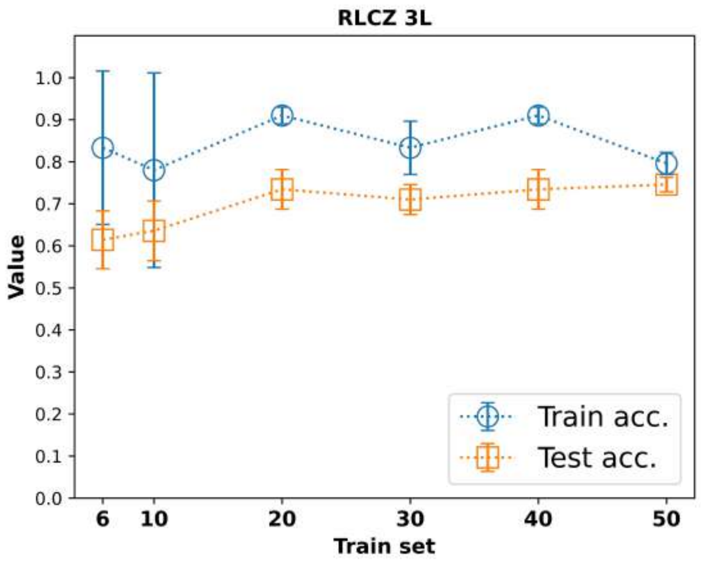


Figure 5.8: RLCZ model with 3 layers. Despite the visible gap between train and test accuracy, the latter one seems to slightly increase for many data-points.

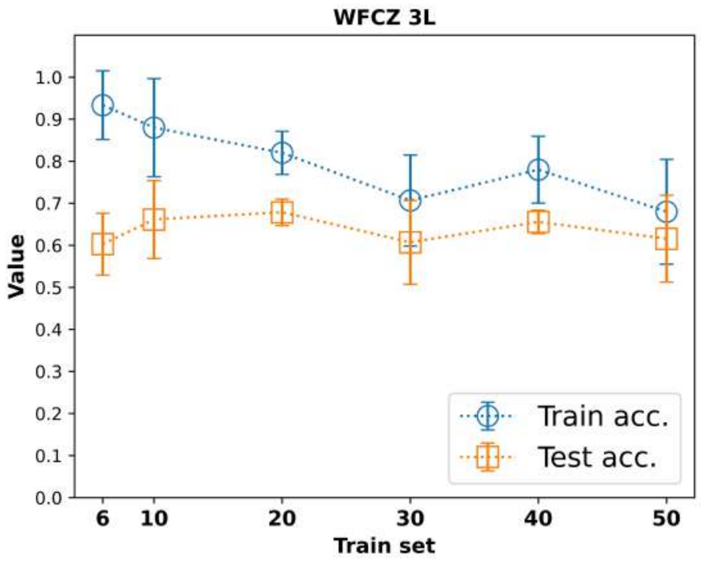


Figure 5.9: Waterfall feature map connected with the CZ quantum circuit with 3 layers used as variational ansatz. The test accuracy does not seem to exhibit a learning behaviour.

The CZ variational ansatzes lead to different results as a consequence of which feature embedding circuit has been used to encode the data-points. They are displayed in 5.8 and 5.9. The RL-CZ model reaches better results compared to the WF-CZ, and the overall test accuracy is closer to the highest level obtained from the RL-RL quantum neural network. On the other hand, the accuracy reached by the WFCZ seems to decrease even when the largest dataset has been used for training.

5.3 Accuracy vs number of layers

In this short section we show the effect of reducing the number of variational ansatz's layers on the accuracy both for training and testing. The goal is to find which is the best size (as known as the number of gates used in a given circuit) a quantum circuit should have in order to learn sufficiently from images when a large number of features must be encoded and should be learned. We analyzed the performances on the first three layers for three specific training sets: 6, 30 and 50 data. We could expect, basing on the experience in classical ML, especially with neural networks, that an increasing level of accuracy could be obtained when more layers are added to the circuit since it would be more favourable that a higher-depth circuit, with many parameters, was able to learn more features and to classify a wealth of images. The current task has been performed for the MNIST dataset. The effects on the generalization should be more visible for a complex dataset as the MNIST. Moreover, increasing the number of parameters that undergoes to optimization will produce a higher level of model's performances in terms of accuracy on new sets of data. In the tables reported in the following lines we have 4 columns to represent the number of images we have chosen for training (**Images**) and the number of layers used within the variational ansatz we have tested, respectively **1 Layer**, **2 Layers** and **3 Layers**.

RLRL model

RLRL accuracy per layer			
Images	**1 Layer**	**2 Layers**	**3 Layers**
6	0.59	0.62	**0.70**
30	0.70	0.71	**0.73**
50	0.69	0.76	**0.78**

WFWF model

WFWF accuracy per layer			
Images	**1 Layer**	**2 Layers**	**3 Layers**
6	0.60	**0.69**	0.64
30	**0.67**	0.65	**0.67**
50	0.65	0.67	**0.68**

RLWF model

RLWF accuracy per layer			
Images	**1 Layer**	**2 Layers**	**3 Layers**
6	**0.63**	0.60	**0.63**
30	0.64	0.59	**0.67**
50	0.61	0.65	**0.67**

WFRL model

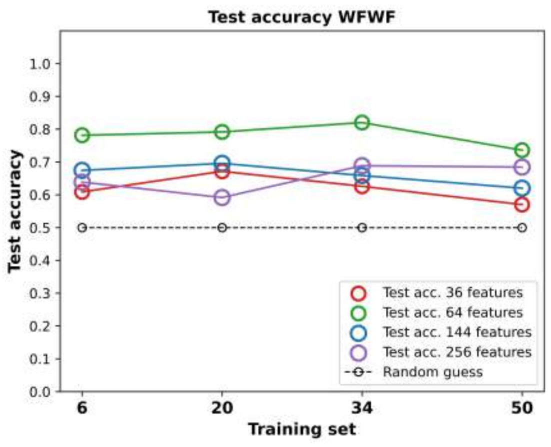
WFRL accuracy per layer			
Images	**1 Layer**	**2 Layers**	**3 Layers**
6	0.64	0.70	**0.71**
30	0.63	0.67	**0.74**
50	0.63	0.61	**0.70**

5.4 Generalization from less features

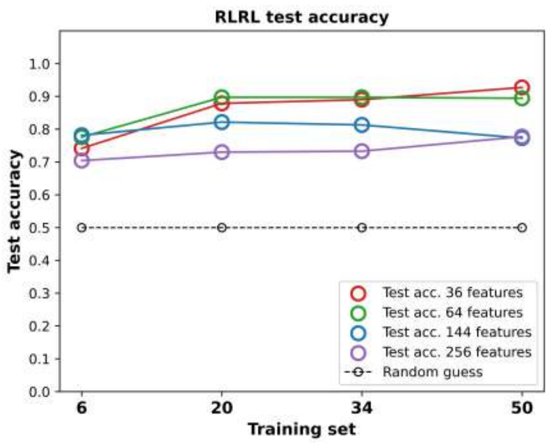
In the second task, we wonder about the model's ability to detect the test images when the number of features is considerably reduced. When we have a group of data from which we extract fewer features and we look for a function able to fit them, we generally start from simple polynomials that have few optimizing parameters. In this sense, the number of features available should induce a bound on the number of parameters subjected through optimization, even though for complex and non-linear models such as CNNs, the number of optimizing parameters grows rapidly when more layers are added to the architecture. Our QNN's ansatzes are not deep in the number of parameters, so the reduction in feature size is justified thanks to the aforementioned reason. Following this logic, it's interesting to look for an improvement, in the test accuracy, when the same train set is composed of images with fewer and fewer pixels. By following this idea, we start to think if a sort of threshold, below which the model fails in predicting the exact label of the image it sees, exists. From the further results we are going to present below, we can make a connection with what has been reported in [10]. In the article, the authors emphasized the power of QNNs involved in an SL problem that uses less training data than classical algorithms. The question we can ask at this point could be "What could happen if we use *few features*?". In this sense, we are potentially able to generalize the concept of less data saying that good accuracy can be obtained by using few images in the training phase composed of fewer features. Following the introduction to this experiment, it could turn out in a simplification for the trainable model to handle less complex images in the training process, translating into an increment in the generalization abilities.

The next figures 5.22a, 5.22b, 5.11b, and the following represent the generalization performances of the QNNs using images with less information contained into them. One of the additional advantages that could arise is the fact that, by reducing the number of features, we reduce the number of gates contained within each feature map, and consequently the possible noise occurrence associated with them. Furthermore, we would have two quantum circuits (the feature map and the variational ansatz) with a comparable number of learning parameters. If with 256 features we need 264 gates for a 6-qubit circuit with angle encoding and only 36 gates used in the variational ansatz are insufficient to learn the images with high accuracy within the train set and classify all the test images, with a reduced number of pixels encoded by the feature map we could expect a significant improvement since the number of optimizing parameters are of the same order of the features.

Lastly, quantum devices would also be able to carry out the learning in less time and using fewer resources since less number of gates are simulated.

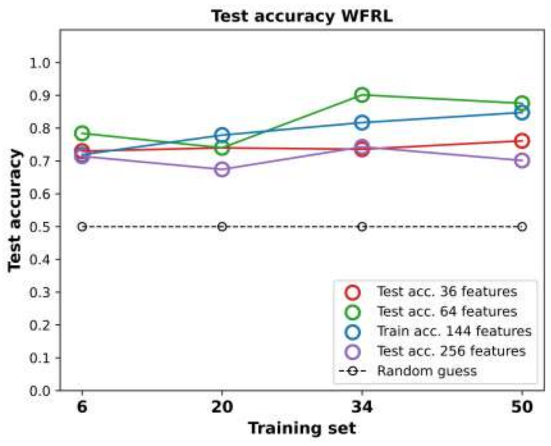


(a) WFWF model.

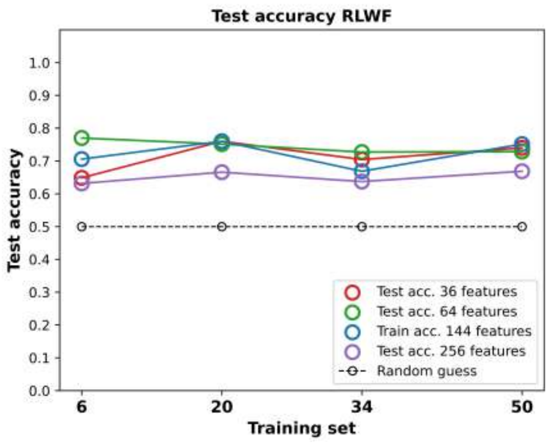


(b) RLRL model.

Figure 5.10: As we can observe in the figures, the combination of the WF circuit either for feature map and variational ansatz shows not as high improvement compared to the same model evaluated to 16x16 images. On the contrary, the RLRL reaches very high accuracy when the dataset is reduced in features, until surpassing 90% for 6x6 and 8x8 images.

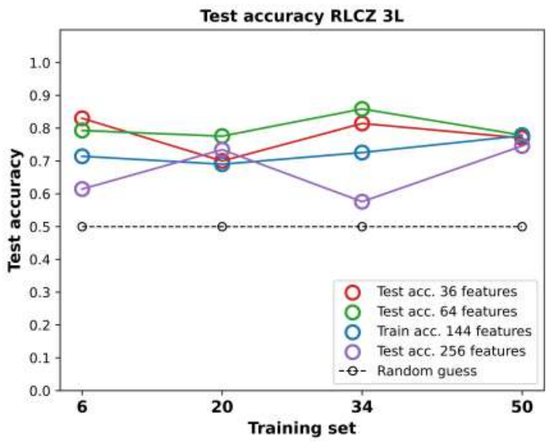


(a) WFRL model.

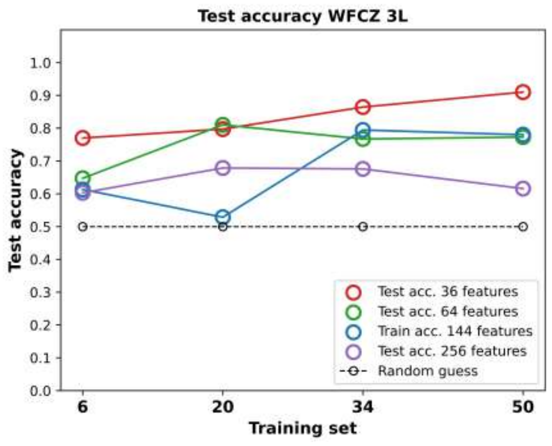


(b) RLWF model.

Figure 5.11: The WFRL model obtained better test accuracy compared to the RLWF; experiments reveal that WF circuit is not able to obtain significant high accuracy when used as a variational ansatz, while if appropriately combined with a highly expressive circuit could bring to better results.

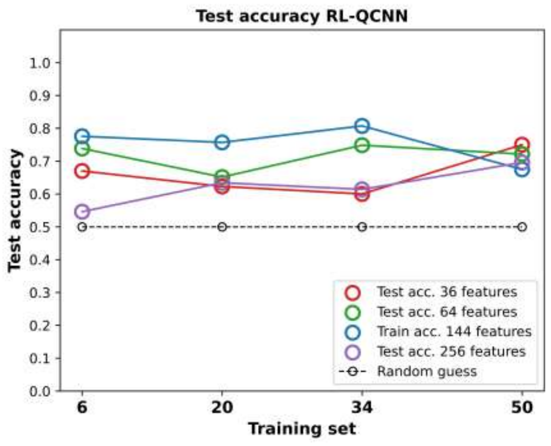


(a) RLCZ model.

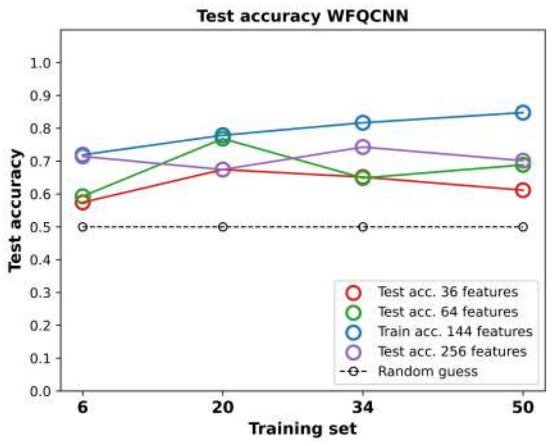


(b) WFCZ model.

Figure 5.12: From these two figures we can say these models reflect in a certain way what we have already got for the RLWF and WFRL; here again we assist to a significant high performance carried out by the WFCZ model except for dataset with a large number of features.



(a) RL-QCNN model.



(b) WF-QCNN model.

Figure 5.13: The displayed results do not suggest the QCNN ansatz to be a powerful candidate for reduced-in-features dataset, since the test accuracy obtained with these architectures are not sufficiently high compared to the previous ones.

5.5 Galaxy results

For the Galaxy dataset we decided to reduce the number of tasks. Each model has been evaluated with the same number of layers in the variational ansatz which is 3, because is the optimal number that gave best results also for Mnist dataset. We then plotted the accuracy's trend on the train and test sets when more images are seen by a selected model. The reason why we omitted the shallower ansatzes is because results are better when more parameters undergo optimization. For a dataset as the Galaxy, we directly tested the best QNN arrangements for image detection. We decided to discard the second task for the current set of data since we are expecting a similar behaviour compared to the previous one (Mnist).
Finally, even though resizing each Galaxy image should not change too much the pixel's displacement or the image content, we are going to show the third and last experiment (where we compare the performances on multiple reduced-in-size versions of the same dataset) for this dataset. We show the pictures, from 5.14 and next, for the models that obtained the best results on the previous set of data.

Tables

Ring-like model's performances					
Dataset	**N° data**	**Model**	**Params.**	**Train acc.**	**Test acc.**
Galaxy	6	RLRL3	36	**1.00 ±0.00**	0.73±0.05
Galaxy	10	RLRL3	36	0.98±0.04	0.76±0.04
Mnist	20	RLRL3	36	0.82±0.06	**0.85±0.03**
Galaxy	30	RLRL3	36	0.93±0.05	0.73±0.06
Galaxy	40	RLRL3	36	0.82±0.06	0.81±0.04
Galaxy	50	RLRL3	36	0.91±0.04	0.79±0.01

Waterfall model's performances					
Dataset	**N° data**	**Model**	**Params.**	**Train acc.**	**Test acc.**
Galaxy	6	WFWF3	36	0.70 ±0.12	0.63±0.06
Galaxy	10	WFWF3	36	**0.78±0.07**	0.60±0.05
Galaxy	20	WFWF3	36	0.60±0.07	**0.64±0.05**
Galaxy	30	WFWF3	36	0.65±0.05	0.58±0.05
Galaxy	40	WFWF3	36	0.63±0.05	0.58±0.02
Galaxy	50	WFWF3	36	0.55±0.10	0.52±0.13

Mixed model's performances					
Dataset	**N° data**	**Model**	**Params.**	**Train acc.**	**Test acc.**
Galaxy	6	RLWF3	36	0.60 ±0.34	0.58±0.21
Galaxy	10	RLWF3	36	0.72±0.21	0.68±0.10
Mnist	20	RLWF3	36	**0.76±0.02**	0.72±0.02
Mnist	30	RLWF3	36	0.75±0.02	0.73±0.03
Mnist	40	RLWF3	36	0.74±0.05	0.74±0.03
Mnist	50	RLWF3	36	0.75±0.04	**0.75±0.02**
Galaxy	6	WFRL3	36	**1.00 ±0.00**	0.68±0.07
Galaxy	10	WFRL3	36	0.88±0.12	0.67±0.04
Mnist	20	WFRL3	36	0.80±0.08	0.66±0.03
Mnist	30	WFRL3	36	0.76±0.02	0.67±0.04
Mnist	40	WFRL3	36	0.71±0.17	0.59±0.10
Mnist	50	WFRL3	36	0.79±0.02	**0.68±0.03**

RLCZ model's performances					
Dataset	**N° data**	**Model**	**Params.**	**Train acc.**	**Test acc.**
Galaxy	6	RLCZ3	36	0.87 ±0.12	**0.77±0.04**
Galaxy	10	RLCZ3	36	**0.88±0.07**	0.75±0.04
Galaxy	20	RLCZ3	36	0.71±0.04	0.76±0.03
Galaxy	30	RLCZ3	36	0.72±0.23	0.66±0.15
Galaxy	40	RLCZ3	36	0.76±0.08	0.76±0.03
Galaxy	50	RLCZ3	36	0.80±0.07	0.71±0.05

WFCZ model's performances					
Dataset	**N° data**	**Model**	**Params.**	**Train acc.**	**Test acc.**
Galaxy	6	WFCZ3	36	**0.76 ±0.17**	0.64±0.10
Galaxy	10	WFCZ3	36	0.66±0.08	0.62±0.07
Galaxy	20	WFCZ3	36	0.62±0.20	0.58±0.13
Galaxy	30	WFCZ3	36	0.71±0.07	**0.71±0.04**
Galaxy	40	WFCZ3	36	0.73±0.16	0.68±0.09
Galaxy	50	WFCZ3	36	0.74±0.06	0.67±0.07

QCNN model's performances					
Dataset	**N° data**	**Model**	**Params.**	**Train acc.**	**Test acc.**
Galaxy	6	RLQCNN	42	**0.90 ±0.13**	0.73±0.02
Galaxy	10	RLQCNN	42	0.86±0.14	**0.75±0.03**
Galaxy	20	RLQCNN	42	0.74±0.08	**0.75±0.02**
Galaxy	30	RLQCNN	42	0.80±0.04	0.74±0.03
Galaxy	40	RLQCNN	42	0.79±0.07	0.73±0.03
Galaxy	50	RLQCNN	42	0.70±0.14	0.64±0.10
Galaxy	6	WFQCNN	42	**0.80 ±0.12**	0.58±0.08
Galaxy	10	WFQCNN	42	0.70±0.11	0.59±0.14
Galaxy	20	WFQCNN	42	0.77±0.07	0.56±0.07
Galaxy	30	WFQCNN	42	0.55±0.22	0.53±0.21
Galaxy	40	WFQCNN	42	0.66±0.06	0.68±0.06
Galaxy	50	WFQCNN	42	0.75±0.05	**0.70±0.02**

Figures

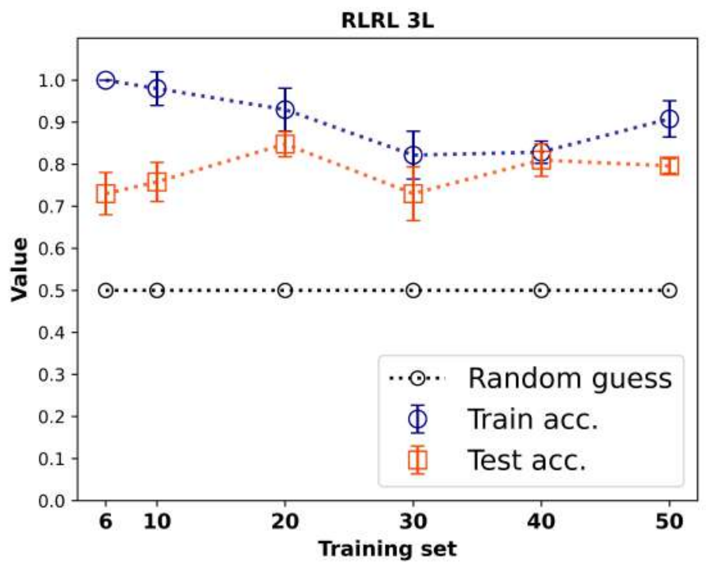


Figure 5.14: Ring-like feature map of 22 layers and Ring-like variational ansatz for Galaxy. A meaningful improvement is visible: the accuracy on the test set surpasses the 80% threshold for certain number of training data-points.

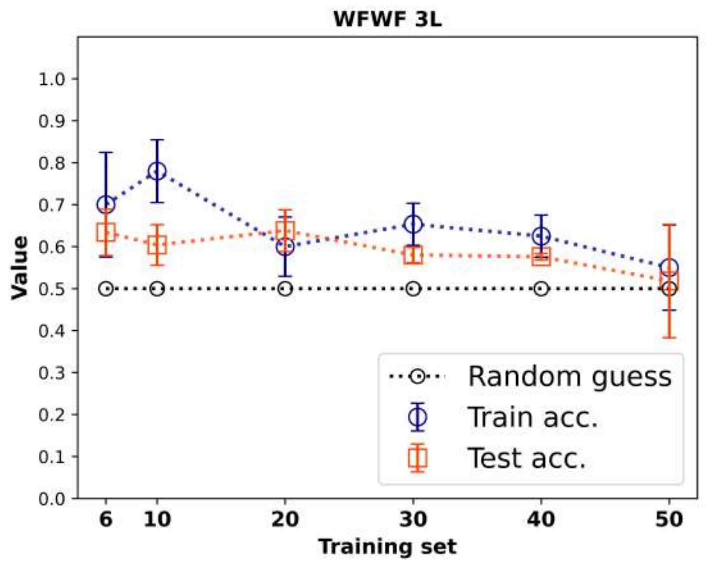


Figure 5.15: Waterfall Waterfall model. An unexpected decreasing trend for both train and test accuracy occurs even with more data-points.

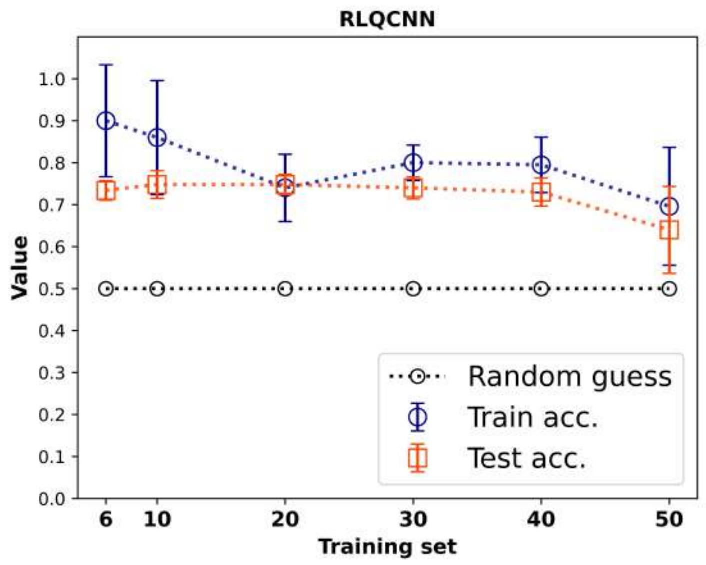


Figure 5.16: Ring-like-QCNN model. Starting from a value over 70% of test accuracy, it gets lower for a larger train set.

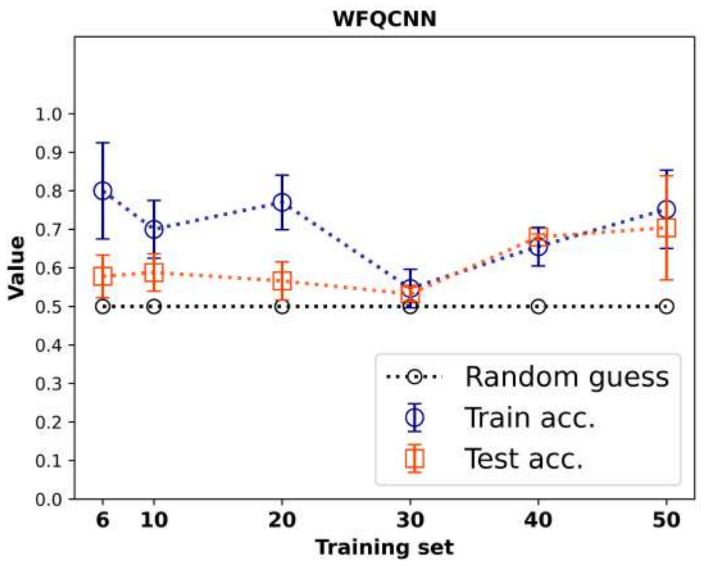


Figure 5.17: Waterfall-QCNN model. We can notice a significant growth of the test accuracy for larger train sets, while for smaller ones it is still under 60%.

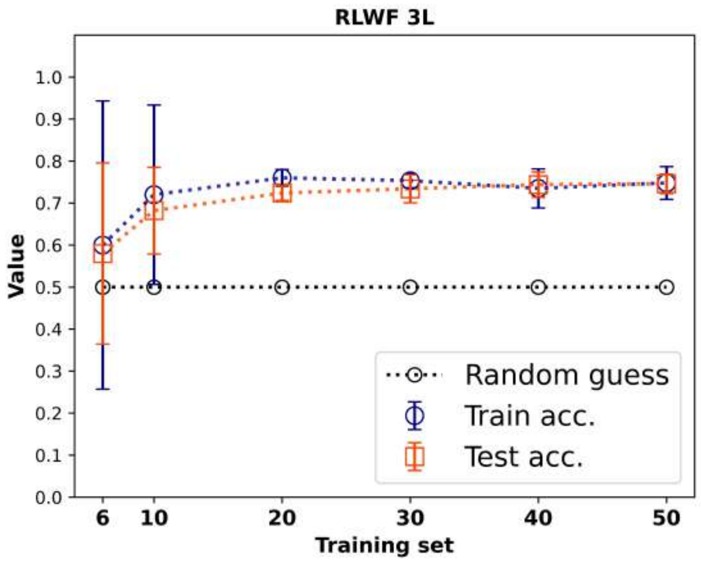


Figure 5.18: Ring-like Waterfall model. There is a high overlap between train and test accuracy combined to a progressive trend that approaches 80%. For small train sets either train and test accuracy have a high variance in common that considerably reduces when more images are considered.

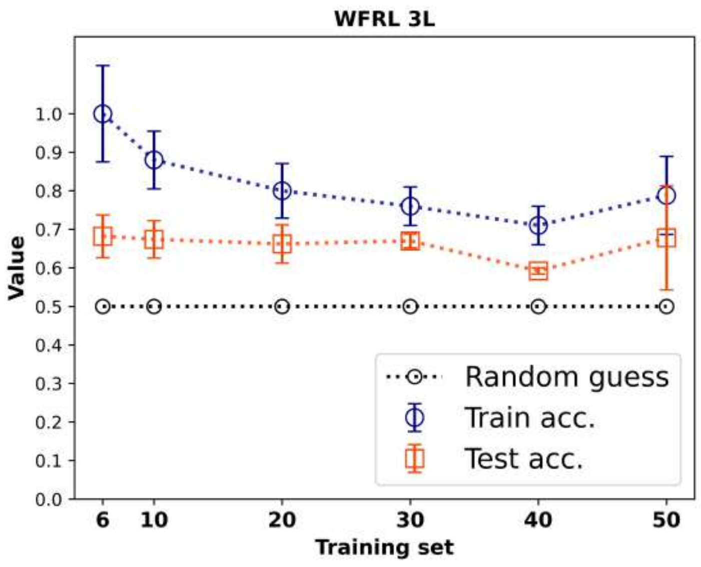


Figure 5.19: Waterfall Ring-like model. It is visible the rapid decline of the train accuracy, while the test accuracy follows the same progression smoothly.

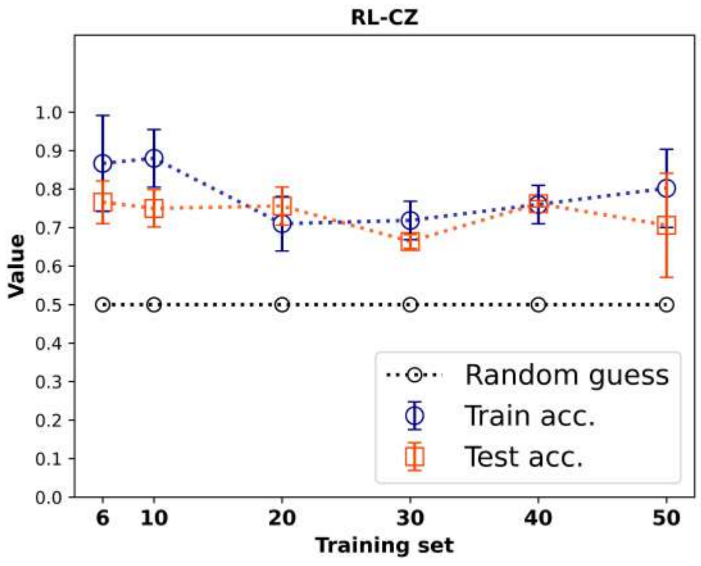


Figure 5.20: Ring-like CZ model. The accuracy on the test set keeps close to almost 80%, even though we register a drop for 50 training data-points. We can notice that the combination between the CZ circuits used as variational ansatz with the RL feature map performs better with respect to the next model that uses the WF as feature map.

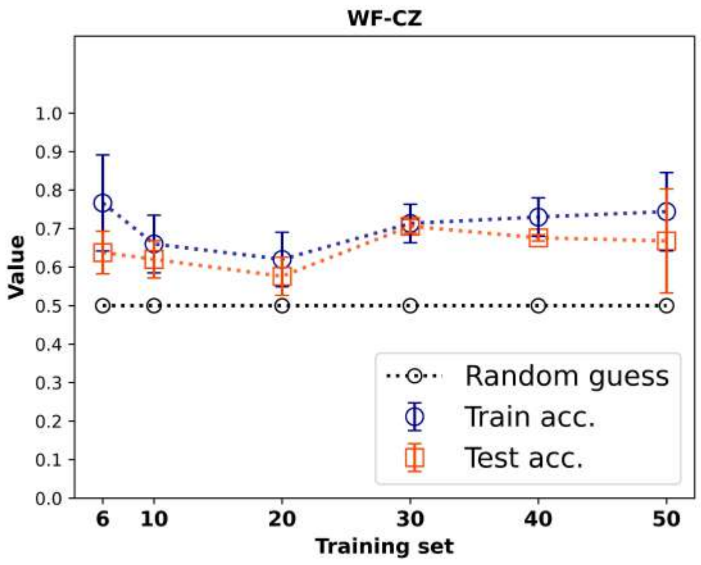
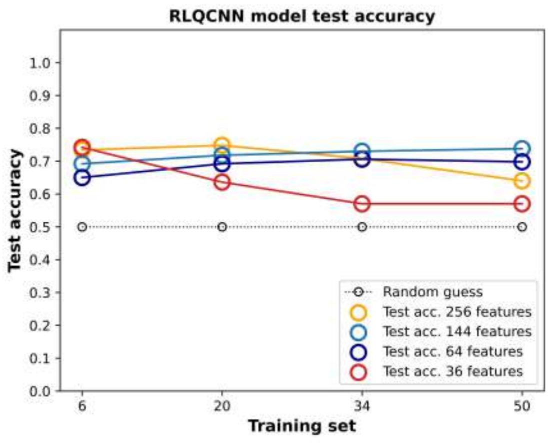
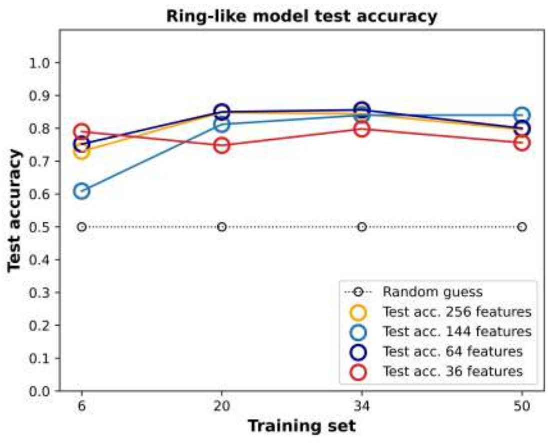


Figure 5.21: Waterfall CZ model. The unexpected smooth decrease of the test accuracy from 6 to 20 images of training is overturned for 30 training data-points even though the same trend is displayed for more images, while the train accuracy seems to slowly increase.

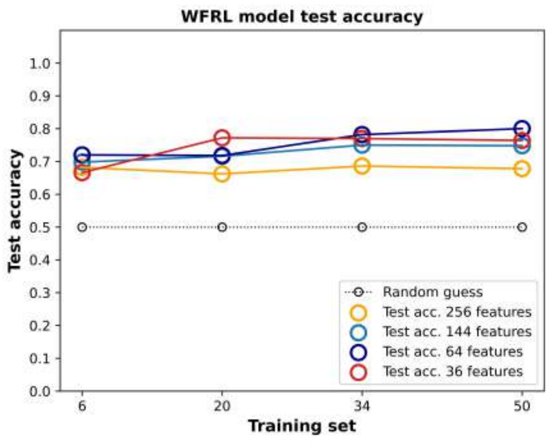


(a) RL-QCNN model.

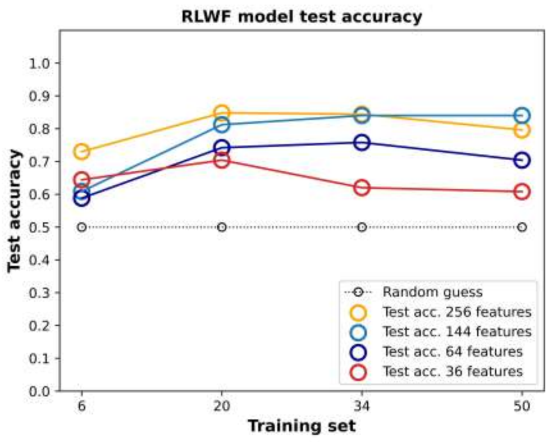


(b) RLRL model.

Figure 5.22: As we can notice, surprisingly the QCNN ansatz does not generalize unseen images as we expected from previous results5.6. On the other hand, the RLRL reaches high performances and confirms to be a suitable candidate for generalization tasks like the one we performed, even thought the results are lower than what we have obtained from MNIST dataset.



(a) WFRL model.

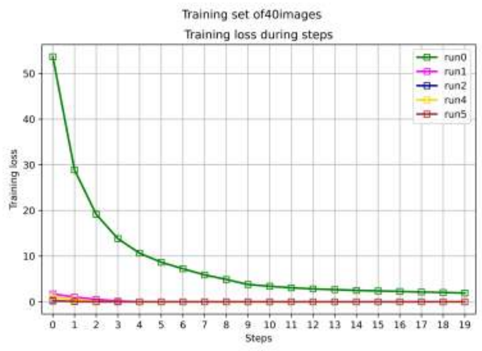


(b) RLWF model.

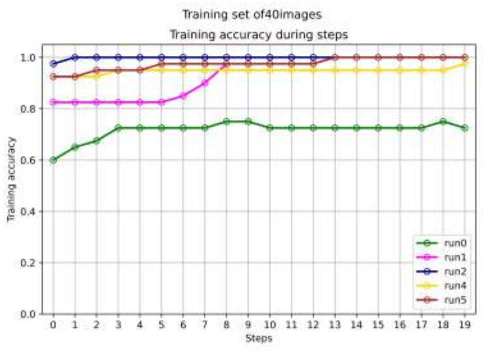
Figure 5.23: The reduction in features seems to not produce any improvements on the WFRL model, as the lines remain almost flat. The accuracy obtained converge to almost the same value within a certain range. The RLWF present unexpected behaviour, in which even less-features images are difficult to generalize, as we can see by looking at the red and dark-blue lines.

5.6 CNN

In this last section we made a CNN using Keras in order to make a comparison between classical and previous quantum models. The adjective - fair - refers to the low number of optimizing parameters the model is composed by. It is constructed by a couple of single convolutional and pooling layers, with 36 learning parameters, a flatten and a final dense layers. It has been realized with *Tensorflow* and it uses 20 epochs for each training process. We performed the training, the final evaluation on the train and test set, changing the images at each run, five times. In this way we collected an average behaviour about its capacities of learning from few data. From the following pictures 5.24, the reader can see the ability of the classical model in optimizing the parameters.



(a) Loss function.



(b) Accuracy.

Figure 5.24: Loss function and accuracy computed during the training process for the classical CNN.

The first run starts from a high value of the loss function that is progressively reduced in each iteration. This effect results in a value of the training accuracy closer to the 100%. This should constitute a test bed with which to compare and try to improve the QNN models. The train accuracy reaches very high values as we expected from such a well studied model. In the picture presented below 5.25, we used again the Mnist dataset composed of only 0 and 1 classes downscaled to 16x16 features for each picture. The classical convolutional neural network has been subjected to the same main experiment described at the beginning of 5.2 section. The figure 5.24 mirrors what we can observe in the 5.25. The train accuracy, when more images are added to the training set, is constantly 100% and the error is practically zero for each point. The test accuracy instead, starts at 75% and surpasses 90% for only 50 training images. The initial presence of over-fitting (a visible gap between train and test accuracy) is noticeably reduced when the model is fed with more data-points. The ability of such NN relies in the fact that it manages to optimize well the few parameters that are present within the network. Unlike the QNNs, from what is depicted in the Appendix C.1 and C.2, the CNN seems

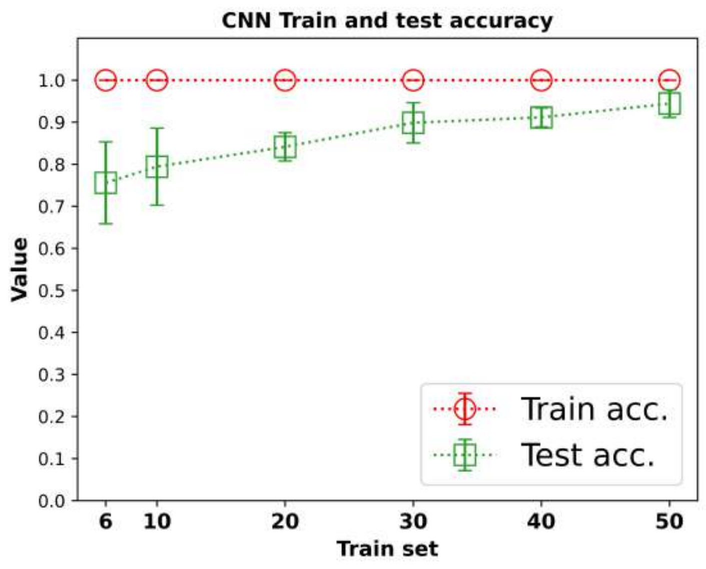


Figure 5.25: Train and test accuracy for the CNN model. For small train sets the overfitting is reduced when the train data is enlarged. For 50 images the train and test accuracy become closer.

to find the best path to minimize the loss function without being stuck in local minima. It seems that there is a completely different loss-landscape compared to what we have found for the quantum models and this could potentially be responsible for the tangible difference between the two types of devices, where we have used the same cost function to evaluate the difference between the predicted and the actual classes for both models.

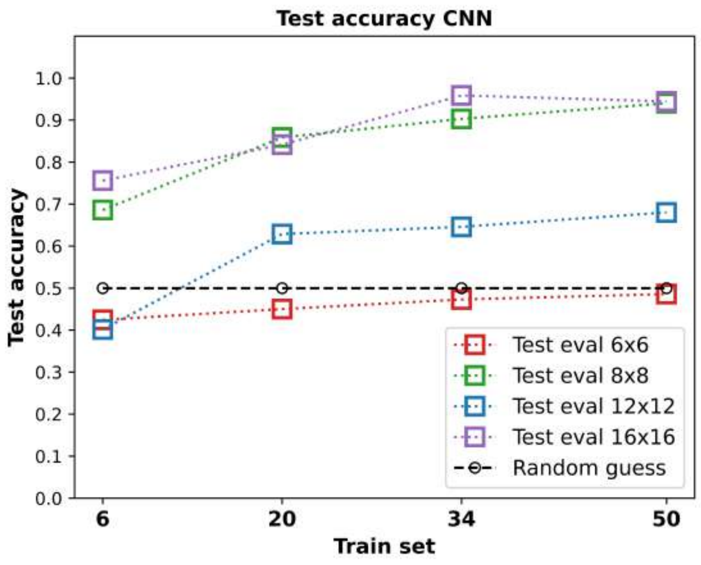


Figure 5.26: In this picture we tested the generalization abilities of the CNN, when each digit-image composing the dataset is downscaled. Apart from the best performances reached with 16x16 images, the sudden drop of the test accuracy when the data are reduced to 12x12 and 6x6 features for image, manifests the inability encountered by the classical model in generalizing new data. The only explanation here relies in the dataset itself. Probably, for those resized data the model has seen difficult images to distinguish and, due to the few number of optimizing parameters, it fails to make prediction. The worst results we have obtained are due to the smallest resized set of 6x6 and 12x12 images. The unexpected result has came from the 8x8 resized pictures where we don't see a worsening, rather we observe a significant improvement on which we are not able to give an answer. These last two will be compared more in detail with the QNNs.

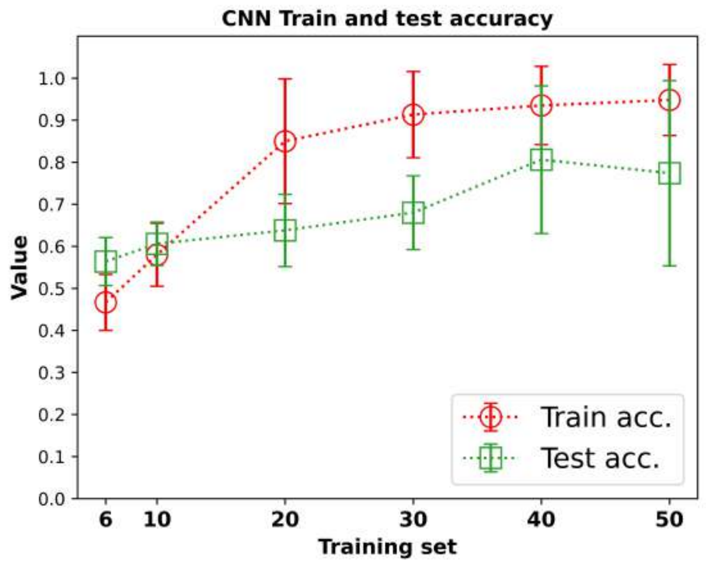


Figure 5.27: Train and test accuracy trends for the CNN that has been trained on few images of Galaxy dataset. The train accuracy, rather than MNIST showed in 5.25, starts from a lower value below 50%. This indicates the difficulty encountered by the model trained on this new dataset of learning from few data. To confirm this statement, we can see how slowly the test accuracy converges to the training one, and the value achieved by the latter surprisingly overcomes the 80% in one occasion.

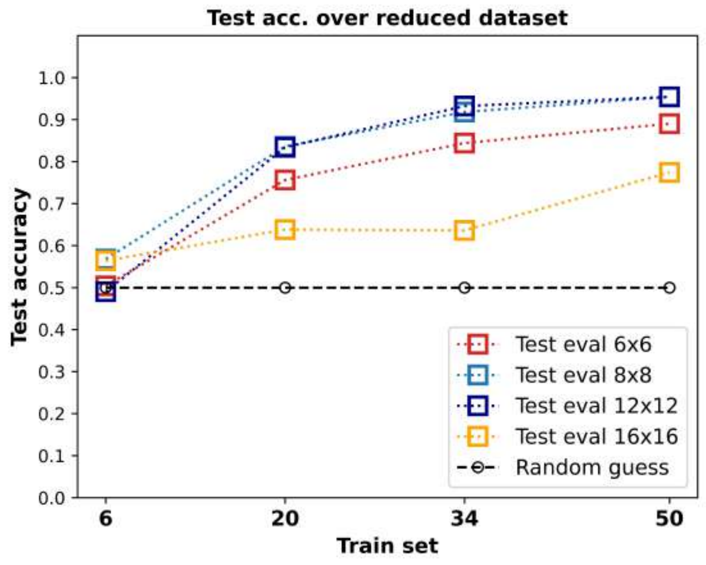


Figure 5.28: Test accuracy for multiple Galaxy images reduced in features. As we can observe, the 16x16 images are still difficult either for the CNN to be classified, while for 8 and 12 features the performances are still the same. Those referring to 6 features are slightly worse.

5.6.1 Comparison

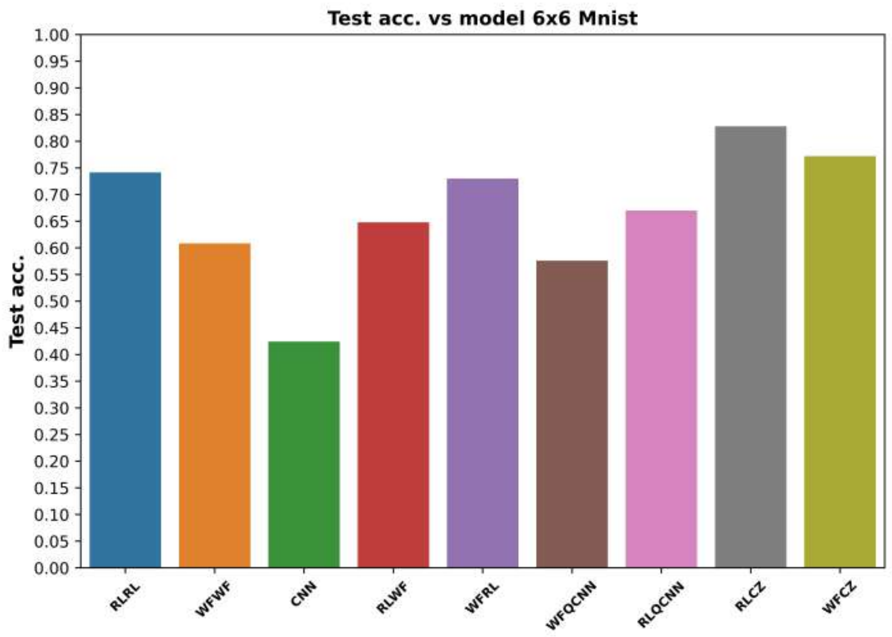


Figure 5.29: Comparison between each quantum model composed by the feature map and variational ansatz circuits and the CNN. With the present histogram we want to show the abilities of all the QNN models to generalize new data when the images are considerably reduced in features.

The picture 5.29 depicts the test accuracy reached by each model for only 6 images of training, when the images of the Mnist dataset are constituted by only 36 features. The bars represent the average value performed on the 5 runs. The RL-CZ model has reached the maximum value around 80% and is the best quantum architecture for the image recognition with only a few number of pixels. Immediately afterwards there is the WF-CZ circuit, followed by the RL-RL and the WF-RL models. From what we can observe here, we can conclude that the Ring-like architecture and the CZ quantum circuit are suitable candidates for the proposed experiment and could be tested for further investigations in this field of research. We are surprised to see the CNN below the 50% of accuracy on the test set. The result we have obtained is directly visible in the figure 5.26, where the accuracy for a reduced-in-features dataset composed of 36 pixels converges to the 50% and seems to not overcome it. This indicates the little value of features we have used is strongly challenging even for the CNN. If for 144 features the accuracy improves significantly as the number of images in the training increases, it does not seem to happen for images of only 36 features, at least for few images. This comparison confirms our initial hypothesis to obtain improved results by using quantum models when data are furnished in a data-efficient way: with few items composed by less features.

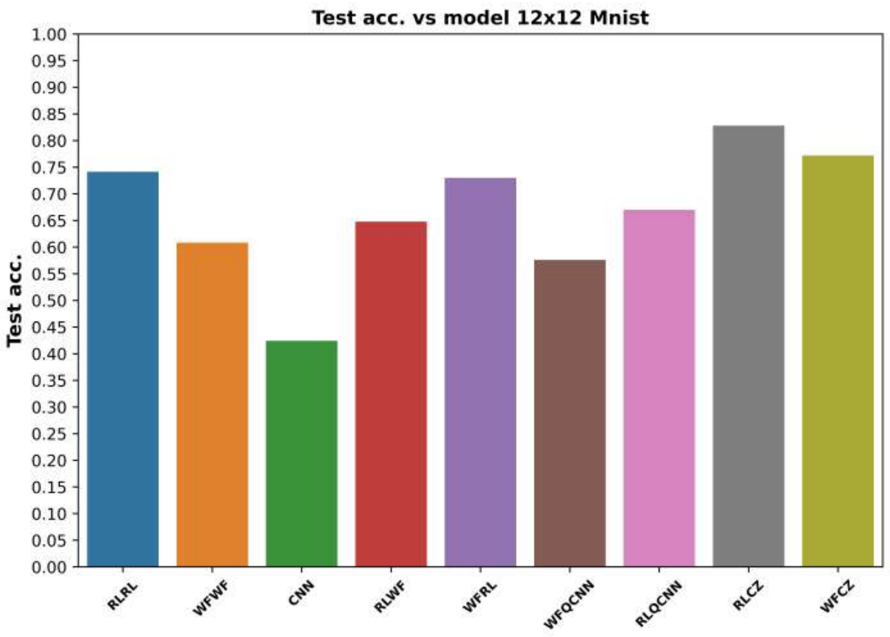


Figure 5.30: It represents the same results we have got previously in the picture 5.29 with the only difference that here the MNIST dataset has been rescaled to 12x12 features. The value of the bars indicates the accuracy on the test set when each model has seen only 6 images of training. We notice the inability of the CNN (green bar) in generalizing unseen data starting from less features images.

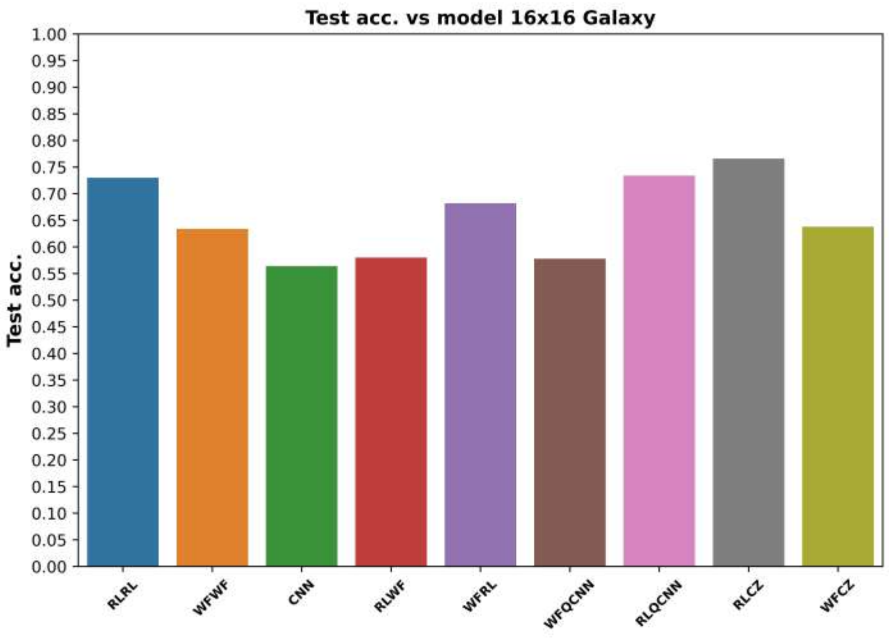


Figure 5.31: In this third histogram the RLRL and RLCZ architectures result the best models for detecting galaxy fields, followed by the QCNN architecture together with the RL feature map with images composed by 256 features. The results obtained demonstrate how much the dataset has its effect on the generalization performance of each model. Only 6 images of training for each model have been used to reach the test accuracy represented as bars in the histogram.

Chapter 6

Conclusions

6.1 Achievements and limitations

The investigation about the possibility of making accurate predictions with few data, by using quantum models, gave its results. From the starting experiment, we can conclude that obtaining a high level of test accuracy is still a challenging task for the current quantum neural networks proposed within the project. The value of accuracy obtained in the test set is converging to an upper bound of approximately 80% for the best quantum model in 5.2, while the value reached by the CNN is closer to 95% in 5.25. Already at this point, there are no clear advantages of training QNNs, with less data, for image recognition over classical models. Further exploration and research are essential in the field to identify quantum neural network architectures and, consequently, quantum circuits that are well-suited for image classification tasks. In addition, likewise we have explained in 2.2.4, the impossibility for such devices to learn over a threshold can be a consequence of the *barren plateaus* phenomena2.2.4. Moreover, increasing the number of layers contained in the variational ansatz, including the number of learning parameters, could be a possible way of studying the problem and verifying what advantages could generate from such enriched architectures. This last statement is visible in the tables of the second kind of experiment we performed 5.3, where, the more layers are added to the variational ansatz, the more accurate the model is even to a lesser extent.
A possible way of getting better results could be expanding the variational parameterized quantum circuit to include more quantum gates and parameters θ to optimize during the training process. On the contrary, we would like to avoid deep quantum circuits that are hard to be run by the available hardware. About that, the reader should take into consideration the possibility of reducing the number of features that compose each image to work with a shallower circuit, which is easy to simulate on classical computers and run on quantum hardware.
The third task we have illustrated aims to show the advantage that can be gained from

it. A possible explanation for the difficulty encountered in the main experiment might be related to the state preparation process. Turning classical data into quantum states makes use of a specific circuit that could be as deep as depending on how many features should be encoded into. Using many quantum gates could raise the quantity of errors on the real quantum hardware where the circuit is executed. In the last experiment, in order to circumvent the problem, we decided to challenge both quantum and classical models by resizing all the train sets and generating images of 6x6, 8x8, 12x12, and 16x16 features. In this way, we tested their abilities when subjected to less-features pictures. It has been demonstrated that quantum models could overcome simple classical CNNs when predicting low-resolution images. As the plot 5.29 illustrates (and the following 5.30), for only 6 images of training, multiple QNNs surpass the convolutional neural network during the test evaluation.
At this point, we can assert that the generalization from few data is possible if we refer to the low number of features to include in the image the model is fed with. We have proved that the quantum models can perform better when the pictures are considerably resized to a low number of pixels which facilitates the model's performance. This latter argument cannot be applied to all circuits we have tested, but we can confirm the power of the Ring-like (RL) variational quantum circuit that has overcome all the others. This kind of circuit demonstrated itself as a good feature map and variational ansatz, when combined with others. The QCNN, and CZ in particular, have shown good abilities in generalizing data as well as the RL and they should be taken into consideration for further experiments.
The last few words we should spend about the code reproducibility.
The main delicate part we got in trouble with was the measurement process. Once the circuit has been created, we decided to use a pre-defined Qiskit's class, *CircuitQNN*, (followed by *SamplerQNN*) to pass the feature map's parameters and the variational ansatz's weights to the model's instance that would have been trained on *PyTorch*, giving the loss function trend and the corresponding train accuracy at each step. Once the process has finished, we evaluated the goodness of the model once on training, and once on the test set. As we have mentioned, the *CircuitQNN* performs a global *measure all* on the qubits and, thanks to an interpret function5.1.8, can extract the quantum state correspondent to the learned parameters, and eventually the class of the image seen by the model. In developing the QCNN architecture instead, we should avoid the global final measurement, otherwise, the advantage behind such a device would fail, i.e. it wouldn't make sense to constantly reduce the number of qubits at each layer if we are going to measure all of them. To tackle this apparent issue, we switched the neural network class furnished by *Qiskit* and we have added a classical register composed of a classical bit, where the measurement of a specific qubit (the last remaining one) is projected to. By doing so, we were able to obtain the result coming from a measure performed on the last qubit, emulating the flow of such a quantum model more faithfully. Our QCNN model differs significantly from what is presented in [12]. In addition to considering the number

of layers and quantum gates within the circuit, the model under discussion involves a measurement operation at each pooling layer. Specifically, after applying the quantum convolutional layer, the pooling operation comprises a measurement on a designated qubit, followed by the application of a unitary operation on the subsequent qubit based on the output of the previous measurement.
However, this approach proved to be more challenging than anticipated. As a workaround, we opted to simplify the process by applying controlled rotations at each pooling layer. This involved considering fewer qubits once the layer was completed and performing the final measurement on the last remaining qubit after the entire process. Notably, we avoided measuring intermediate qubits and instead applied parameterized gates to a reduced set of qubits (reduced by a factor of 2).

6.2 Future works

From the presented results we can assert that there is still a lot of work to be done in order to efficiently adapt QNNs to solve complex problems in Machine Learning, relying on less data. Initially, as already anticipated in 6.1, lots of different quantum circuits could be mixed together and tested to obtain high predicting accuracy on new datasets. Furthermore, new kinds of architectures such as quantum convolutional neural networks based on [12] should be investigated more. For the current analysis, we suggest to introduce other datasets and test the model's generalization on them. As an example, it would be useful to train the quantum models on set of data such as FashionMnist or introducing different digits inside the process, to look for an initial bias that could affect the final results. Roughly speaking, when we tested the abilities of each quantum model on two well-separated classes of Mnist dataset, we usually faced the same problem on the test set: one class (1 digits) has been completely correctly classified, the other one (0 digits) was detected until $70 - 75\%$ of the actual 0-images more or less. This intrinsic behaviour could be tackled by including different digit images, maybe less separated like 2 and 5, 3 and 8. By doing so, one could verify if the same trend occurs or not, detecting a possible intrinsic bias. In this regard, it could be interesting to create a QNN suitable for a multi-class SL problem, i.e. introducing all the 10 classes contained by Mnist dataset and see what happens at the end. By following the logic explained in the 5.4, it is suggested to significantly resize the images in order to look for an improvement of the same kind we have showed in the last experiment and to facilitate the simulation process. Moreover, it would be possible to apply classical dimensionality reduction techniques as to use a lower number of qubits that allows us to expand the number of layers in the variational ansatz without requiring too many computational resources during the iteration. One additional idea could be to expand the problem's complexity by using RGB images instead of gray-scale, including more features and optimizing parameters inside the feature map and the variational ansatz, but taking care to maintain the depth

of the circuit as low as possible. To conclude the idea behind a multi-class SL problem, we recommend to use the adequate number of qubits required for the measurement process, i.e. with 10 classes images, we need at least 4 qubits to span all the possible outcomes of the measurement.
From the results we have obtained, it seems that the most prominent architecture able to obtain high performances does not be completely quantum-based. Rather, it could be an hybrid structure composed of quantum and classical parts combined together. In this way we could in principle use complex images, add more features and parameters to the model exploiting the enormous capacities of the convolutioanl layers to extract the features. By a following pooling operation that has to be repeated several times we can reduce the features used to feed the quantum node with the intent to use less qubits, less quantum operations, but to exploit the superposition and the entanglement effects. A further deepening around this field of research could be to study the abilities of such a classical convolutional neural network with a quantum layer at the end[37].

Appendix A

The entanglement

Quantum entanglement is the phenomenon that occurs when a group of particles are generated, interact, or share spatial proximity in such a way that the quantum state of each particle of the group cannot be described independently of the state of the others, including when the particles are separated by a large distance. The following subsection focuses on the formalism of the entanglement, illustrating with some formulae the mathematical concept behind that.

Pure states

Consider two arbitrary quantum systems A and B, with respective Hilbert spaces H_A and H_B. The Hilbert space of the composite system is the tensor product

$$H_A \otimes H_B. \tag{A.1}$$

If the first system is in state $|\psi\rangle_A$ and the second in state $|\phi\rangle_B$, the state of the composite system is the tensor product between the two

$$|\psi\rangle_A \otimes |\phi\rangle_B. \tag{A.2}$$

States of the composite system that can be represented in this form are called separable states. Not all states are separable states, in fact fixing a basis $|i\rangle_A$ for H_A and a basis $|j\rangle_B$ for H_B, the most general state in $H_A \otimes H_B$ is of the form

$$|\psi\rangle_{AB} = \sum_{i,j} c_{ij} |i\rangle_A \otimes |j\rangle_B \tag{A.3}$$

This state is separable if there exist vectors c_i^A, c_j^B such that $c_{ij} = c_i^A c_j^B$ yielding to write $|\psi\rangle_A = \sum_i c_i^A |i\rangle_A$ and $|\phi\rangle_B = \sum_j^B c_j^B |j\rangle_B$. On the contrary, if the statement is false, i.e. $c_{ij} \neq c_i^A c_j^B$, state are called "entangled states". For example, given two basis vectors $|0\rangle_A$,

$|1\rangle_A$ of H_A and two basis vectors $|0\rangle_B$, $|1\rangle_B$ of H_B, the following state is an entangled state: $\frac{1}{\sqrt{2}}(|0\rangle_A \otimes |1\rangle_B - |1\rangle_A \otimes |0\rangle_B)$.
If the composite system is in this state, it is impossible to attribute to either system A or system B a definite pure state. Another way to say this is that while the von Neumann entropy of the whole state is zero (as it is for any pure state), the entropy of the subsystems is greater than zero. In this sense, the systems are "entangled". The above example is one of four Bell states, which are (maximally) entangled pure states (pure states of the $H_A \otimes H_A$ space, but which cannot be separated into pure states of each H_A and H_B).
Now suppose Alice is an observer for system A, and Bob is an observer for system B. If in the entangled state given above Alice makes a measurement in the $|0\rangle, |1\rangle$ eigenbasis of A, there are two possible outcomes, occurring with equal probability:

- Alice measures 0, and the state of the system collapses to $|0\rangle_A|1\rangle_B$;
- Alice measures 1, and the state of the system collapses to $|1\rangle_A|0\rangle_B$.

If the former occurs, then any subsequent measurement performed by Bob, in the same basis, will always return 1. If the latter occurs, (Alice measures 1) then Bob's measurement will return 0 with certainty. Thus, system B has been altered by Alice performing a local measurement on system A. The outcome of Alice's measurement is random. Alice cannot decide which state to collapse the composite system into, and therefore cannot transmit information to Bob by acting on her system.

Appendix B

Data expansion

In this section we focus on the data preparation, especially we are going to illustrate how we generated each data set and how they are broadened during each simulation. The $16x16$ MNIST train dataset composed by 6 images, 3 belonging to the class **0**, the others to the class **1** are depicted in figures B.1 and B.2. Each set is correctly balanced, i.e. chosen a number of items, half belong to the first class, the other half to the second one in order to avoid any source of bias that can be induced by a non balanced set of data. All the images in the B.1 are randomly contained in the successive dataset of 8 items displayed in B.2. In this case, the seed is the same for the two datasets which differ only in the number of elements that are made of. By creating five different versions for each of the six train sets we can obtain an average behaviour of the train and test accuracy of a specific model. Note that the corresponding label below each digit refers to the exact class of the image, i.e. the **true label**. All of these figures are not contained in the test set, since the model should be able to generalize data it has not seen yet and the experiment is to match as many labels as possible to the random test images that are selected each time. The beauty of working with a dataset such as MNIST is that we have a large number of possible items available that can be selected randomly after a shuffle that we have introduced in the data collection.

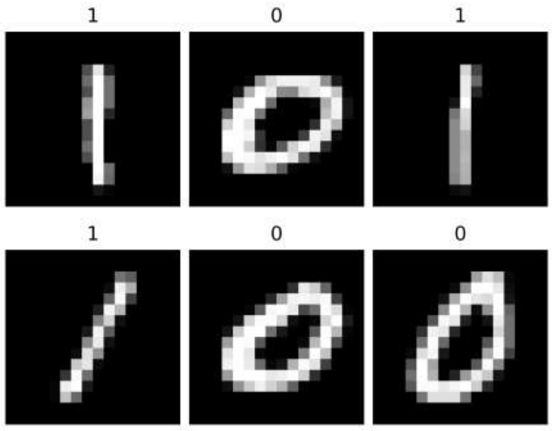


Figure B.1: 16 x 16 Mnist training data with 6 images.

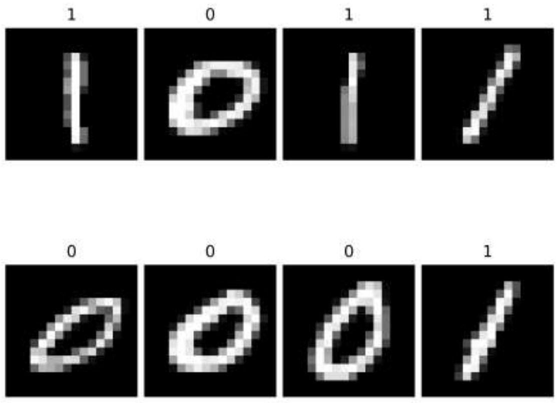


Figure B.2: Expansion of the dataset: the 6 images contained in the picture above are still contained in this new dataset, according to the chosen seed.

The effect of the downsampling

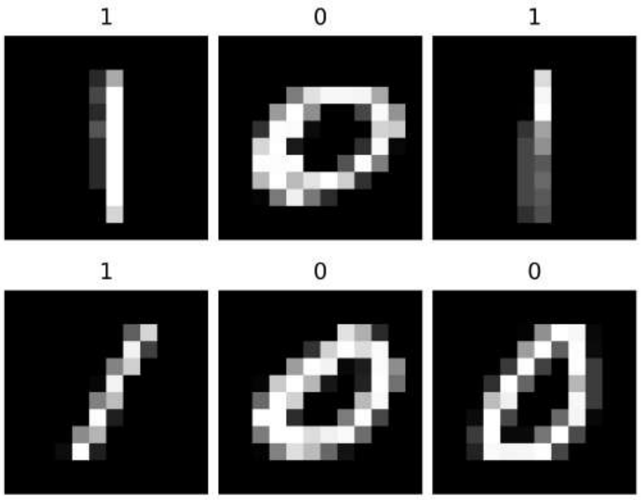


Figure B.3: 12x12 Mnist dataset.

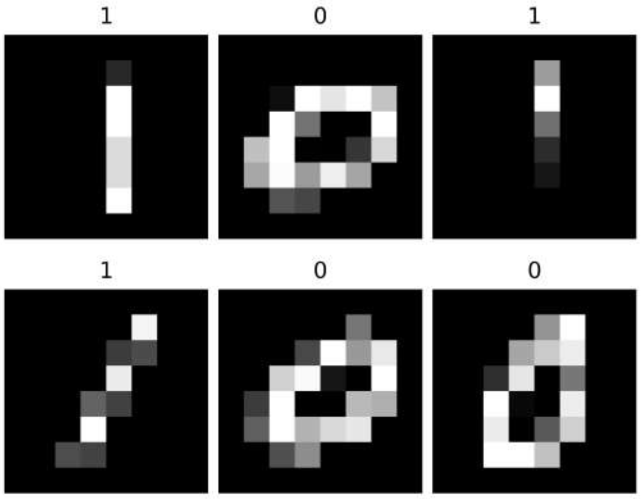


Figure B.4: 8x8 Mnist dataset.

The effects of binarization

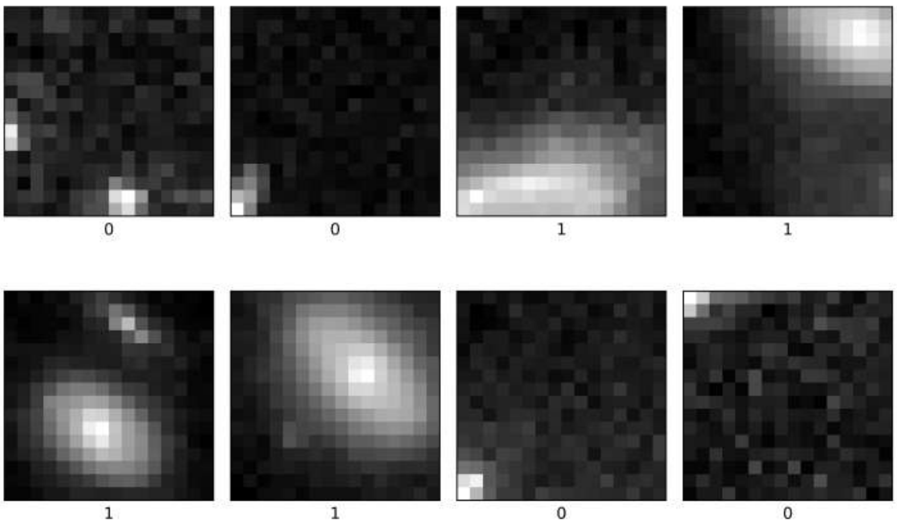


Figure B.5: Non-binarized galaxies.

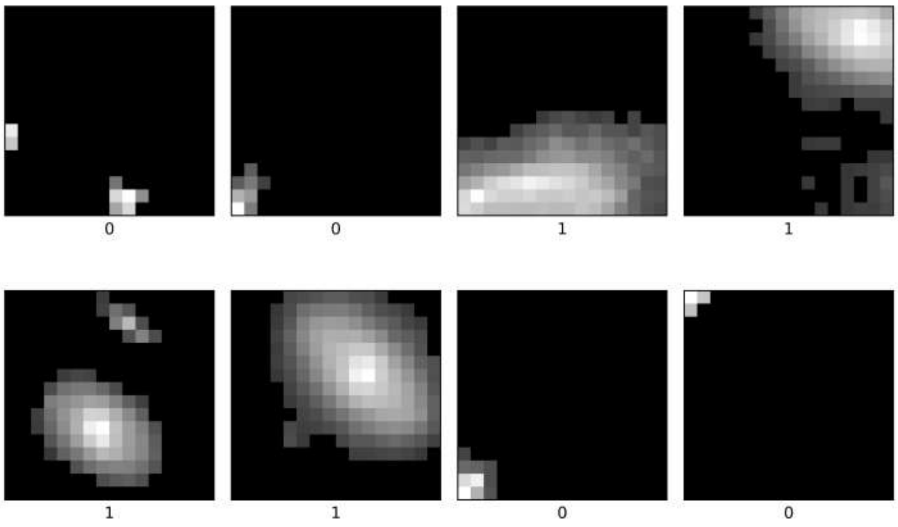


Figure B.6: Binarized galaxies.

Appendix C

Training process

Quantum

In this section we have depicted the progression of the loss and train accuracy during each run of each training process we have performed. We have set 20 epoches and a batch size equals to 1 as each model will see only one image before updating the learning parameters. The two figures represented below, C.1 and C.2, show the progression of loss and accuracy functions during the training phase of a specific model: here the QCNN

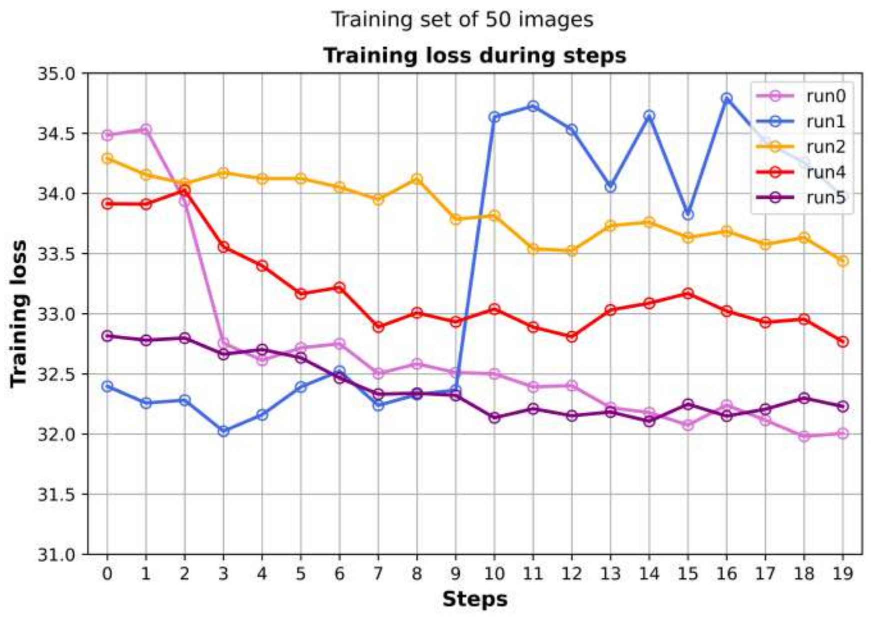


Figure C.1: Multiple loss functions corresponding to each run the model has been trained with. Despite a global descending behaviour, there is an unexpected jump, meaning that the loss has fallen into a local minima, behind which the loss landscape grows up again.

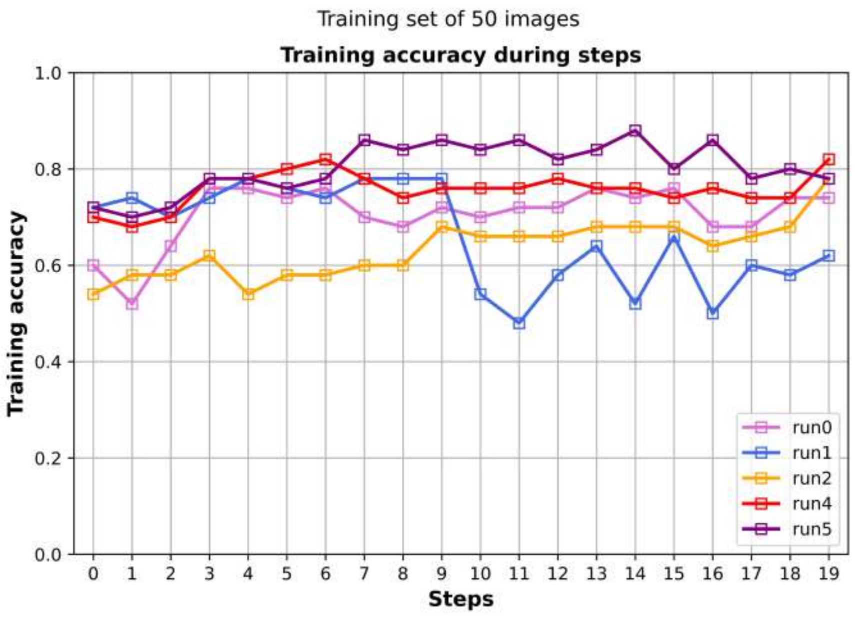


Figure C.2: The figure displays the behaviour of the accuracy during the training process. It reflects the loss trend, suddenly decreasing in the run n°1, meaning of the presence of a local minima in that point. Despite the issue encountered near the step 9, it can be said that globally the training accuracy increases as we would expect. After the training process, the model will be evaluated on the larger test set.

ansatz together with the Ring-like feature map with 22 layers with 50 images of training has been chosen and printed to illustrate the main part of the learning process. As it's possible to notice, in the first picture C.1 we visualize a - common - decreasing trend of the loss function for each run, apart from the number 1. The sudden jump that characterizes the second run in the vicinity of the 10th step.

Appendix D

QCNN measurement

Here we would like to quickly show the effects of the measurement operation on the same QCNN architecture used to classify images here. As reported in 4, this kind of QCNN inspired from [12], has been tested twice in order to see what changes in the generalization performances by measuring different qubits from which to extract the best parameters that undergo optimization. As the picture portrays in D.1, we have emphasized the single-qubit measurement by representing it within the figure. Instead of conducting a global measurement on all the qubits, we introduced a single-qubit measurement, which is passed in the *SamplerQNN* class and utilized for making predictions at the conclusion of the process. In the specific circuit discussed, we explicitly display the single classical bit where the measurement result is collected. However, for the other circuits in 4, we have omitted the mention of the eight classical bits, although they are implicitly understood. In this approach, we abstained from using the parity interpreter. Given that only one qubit is measured, we can observe only two possible single states, namely $|0\rangle$ and $|1\rangle$. Consequently, the parity would merely tally the occurrences of zeros and ones within the state without distinguishing between even or odd, as determining the parity of an 8-qubit state (which would be 2^8 states) is impossible with a measurement on a single qubit. The following image illustrates the impact on test accuracy when both a measure-all and a single-qubit measurement are executed within the same circuit. We opted to use the Galaxy dataset for testing, as the experiment took considerable time to run. To expedite the process, we selected a less complex dataset compared to MNIST, namely the Galaxy dataset.

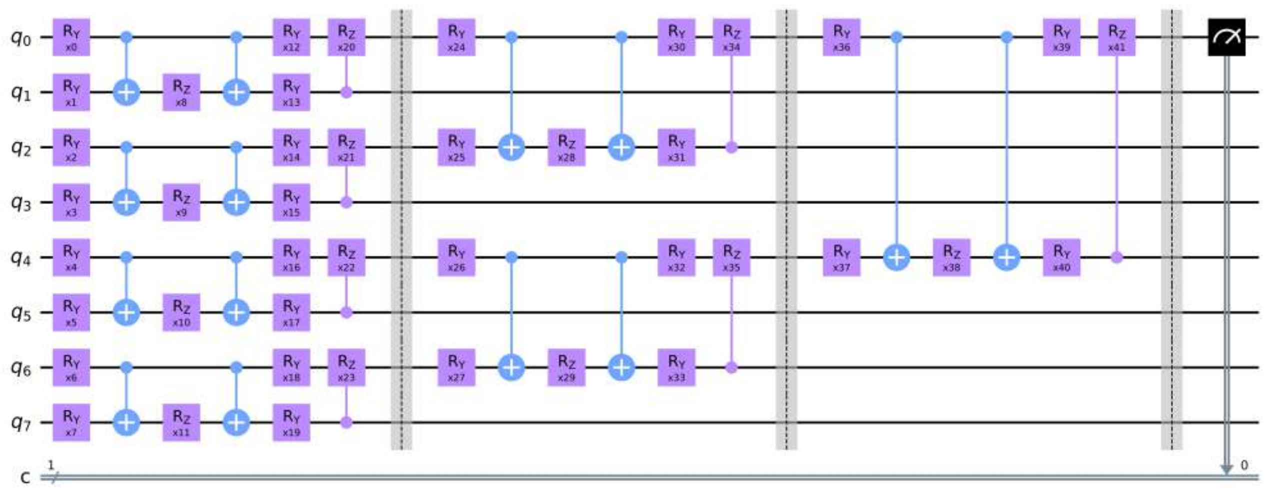


Figure D.1: Utilizing the same architecture as in 4.4, this variant involves measuring only qubit 0, and the prediction is solely derived from this measurement. The measurement result is stored in a single classical bit, which is then employed to make predictions on new data. In the context of binary classification, a single qubit is adequate to yield a state that can be either $|0\rangle$ or $|1\rangle$, corresponding to the two classes of images under consideration. However, for a multi-class problem, it is essential to consider the precise number of classes for prediction and adjust the number of specific qubits accordingly. In this regard, denoting X as the number of classes to detect at the end of training, the required number of qubits for measurement would be $n = \log_2 X$.

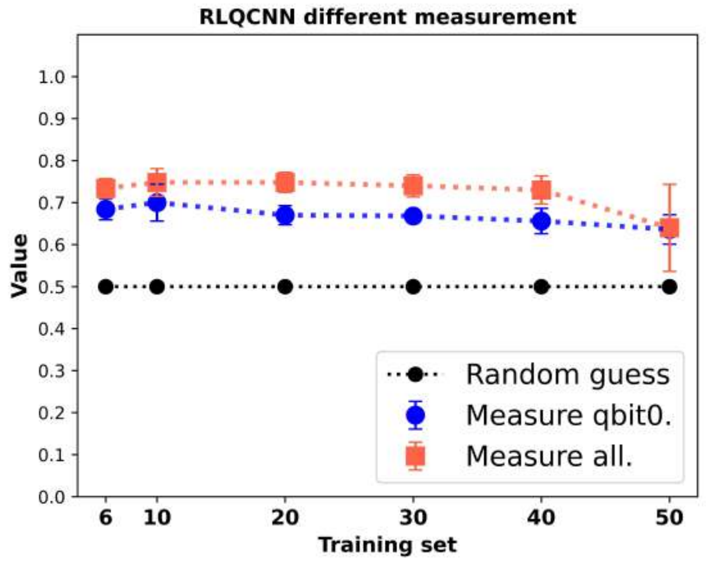
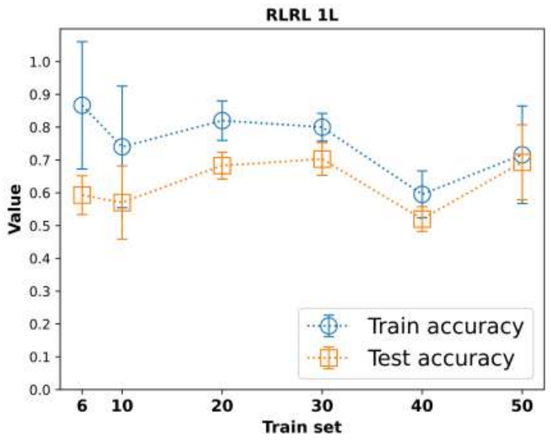


Figure D.2: In the figure we have represented the test accuracy for the QCNN with the measure all instruction as the red line, while the one coming from the measurement performed only on the qubit 0 (i.e. the last qubit remained in the circuit) as the blue line. We can see that, by measuring only one qubit, even though make much more sense for such an architecture that reduces the number of qubits by a factor of two in each layer, doesn't improve the performances.

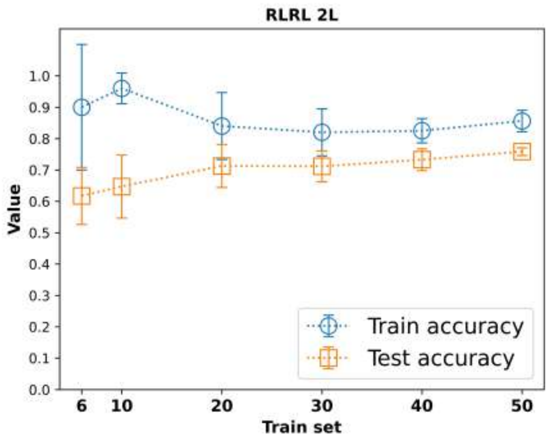
Appendix E

Plots

Here we display the plots we have omitted in the section 5.2 for the MNIST data.

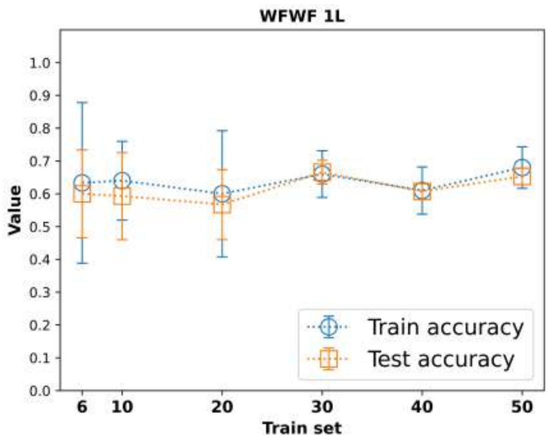


(a) RLRL-1 LAYER model.

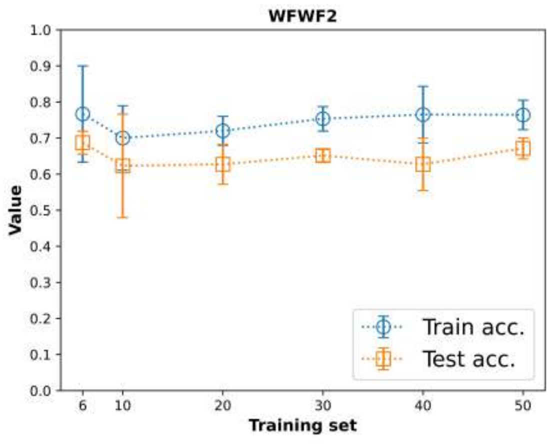


(b) RLRL-2 LAYERS model.

Figure E.1: Train and test accuracy trend for the architectures that use Ring-like circuit for feature map either for ansatz. It is visible a discrete convergence between the two functions that results in a progressively reduction of the over-fitting. We can then notice the effect of adding more layers (and variation parameters) to the circuit which permits of obtaining a better generalization accuracy.

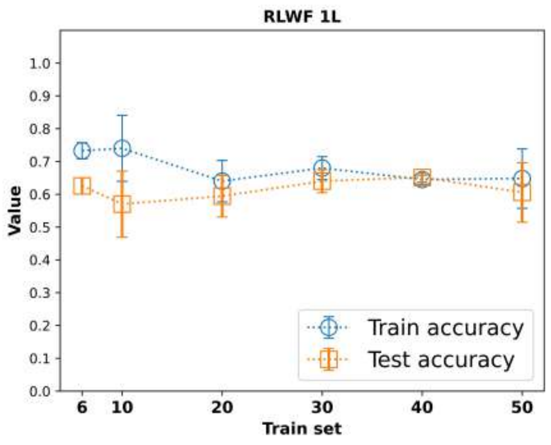


(a) WFWF-1 LAYER model.

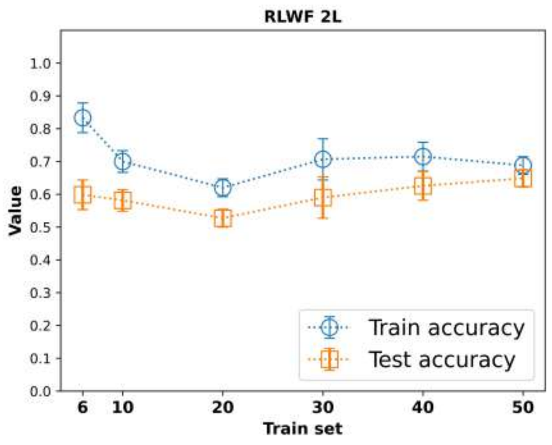


(b) WFWF-2 LAYERS model.

Figure E.2: Train and test accuracy trend for the architectures that use Waterfall circuit for feature map either for ansatz. The left-hand side plot shows a reduced over-fitting. The test accuracy of both plots are not sufficiently able to surpass the 70% threshold.

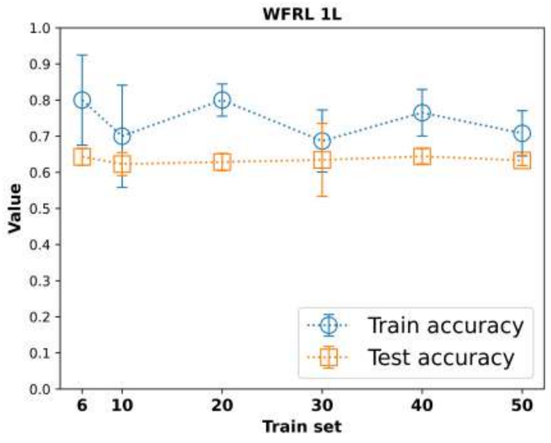


(a) RLWF-1 LAYER model.

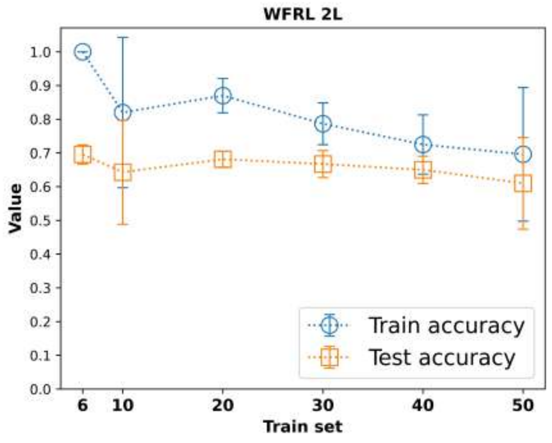


(b) RLWF-2 LAYERS model.

Figure E.3: Mixed architecture composed by the Ring-like feature map and the Waterfall variational ansatz. They start from 60% of accuracy but do not make considerable improvements during the course of the training even when an additional layer is inserted within the parameterized quantum circuit. Additionally, the left-hand side QNN shows a flat trend in the train accuracy, which usually differs significantly with respect to the test accuracy, aiming the learning process is difficult for this kind of device.



(a) WFRL-1 LAYER model.



(b) WFRL-2 LAYERS model.

Figure E.4: Train and test accuracy trend for the architectures that use Waterfall circuit for feature map and Ring-like for ansatz. The right-hand figure we observe an unexpected decreasing accuracy for larger sets of data. The left-hand side seems to not be able to generalize unseen data after has been trained with more data-points.

With the hope that this thesis has been of interest for the reader, I would like to strongly thank multiple people that has made part of my two-years university course. The first person I would like to thank is myself, which is the one that has has been the protagonist of this long course and he made his way from all the difficulties that the university could generate: from the amount of study, to having to take the train to reach the lesson location. I dedicate it to me, the only person who above all and everything has always believed in successfully completing this course, knowing himself and his weaknesses. Subsequently, I'd like to thank all my family: my mother, my father and my strong sister who accompanied me without ever leaving me alone, supported me anyway and anytime, they who cheered me up when I had bad days, but they knew how to congratulate me after good days and good results, and gave me the opportunity to conclude this study. Another person who deserves to be here in this list is my girlfriend, who supported me anytime, showing her total kindness in listening to my preparation before an exam, even though she is not interested on this subject. I will always be grateful to her for the time, patience and the commitment she manifested to me. I have also met a wealth of people during the university which I will never forget, and for this purpose I explicitly want to thank to all of them. In particular, I have bonded a lot with some people who I want to mention here, but referring to them with the initial letter of their name to avoid possible embarrassment and above all to keep the mystery alive. Thanks to the smart, admirable and super-kind Mr. A., whom was a Physics' mate I have met after a QFT lesson, but now he's a friend and whom I keep in touch daily, I discuss, I have fun, I torment him with doubts, requests, perplexities, etc. Thanks to another smart guy, Mr. Ch., for whom I have strong admiration, who helped me a lot and with whom I had different conversations. Thanks to one of the best Chess-player I've ever met during my long and endless career, Mr. M., genius, polite, good friend, Caro-Kann expert and totally feel in love with Quantum. Last but not least, thanks to Mr. A. who has deeply transferred the passion for Computer Science that I have hidden inside myself, covered by lots of stones I have finally dropped out. Another beautiful and smart person who I'd like to know more. Thanks to one of my friend, known for being a tireless walker, who supported and encouraged me to aim high and forward in the last period, thanks Mr. F. (Ch. are the initial letters of his nickname). Thanks to the University that gave me the opportunity to grow and to meet smart and curious people that have influenced me during my career. Thanks to the University where I got in contact with high competent professors. Ultimately, great thanks to professor Lodi, great thanks to Antonio, great thanks to Riccardo, great thanks to Filippo (to whom a special mention goes) and great thanks to the guys within the group of research in QML (Supreeth, Matteo and Simone) who helped me a lot during the research, have always been very helpful and have transmitted the passion and curiosity aim to pursuit this career in Science.
Thanks to all of you that left me something.

FAV.

Bibliography

[1] Amira Abbas, David Sutter, Alessio Figalli, and Stefan Woerner. Effective dimension of machine learning models. *arXiv preprint arXiv:2112.04807*, 2021.

[2] Amira Abbas, David Sutter, Christa Zoufal, Aurélien Lucchi, Alessio Figalli, and Stefan Woerner. The power of quantum neural networks. *Nature Computational Science*, 1(6):403–409, 2021.

[3] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*, pages 1–6. Ieee, 2017.

[4] MV Altaisky. Quantum neural network. *arXiv preprint quant-ph/0107012*, 2001.

[5] Davis Arthur et al. A hybrid quantum-classical neural network architecture for binary classification. *arXiv preprint arXiv:2201.01820*, 2022.

[6] Marco Ballarin, Stefano Mangini, Simone Montangero, Chiara Macchiavello, and Riccardo Mengoni. Entanglement entropy production in quantum neural networks. *Quantum*, 7:1023, 2023.

[7] Leonardo Banchi, Jason Pereira, and Stefano Pirandola. Generalization in quantum machine learning: A quantum information standpoint. *PRX Quantum*, 2(4):040321, 2021.

[8] Kerstin Beer. Quantum neural networks. *arXiv preprint arXiv:2205.08154*, 2022.

[9] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

[10] Matthias C Caro, Hsin-Yuan Huang, Marco Cerezo, Kunal Sharma, Andrew Sornborger, Lukasz Cincio, and Patrick J Coles. Generalization in quantum machine learning from few training data. *Nature communications*, 13(1):4919, 2022.

[11] Marco Cerezo de La Roca, Akira Sone, Kunal Sharma, Tyler Volkoff, Lukasz Cincio, and Patrick Coles. Barren plateaus in quantum neural networks. In *APS March Meeting Abstracts*, volume 2021, pages S32–008, 2021.

[12] Iris Cong, Soonwon Choi, and Mikhail D Lukin. Quantum convolutional neural networks. *Nature Physics*, 15(12):1273–1278, 2019.

[13] Alexandr A Ezhov and Dan Ventura. Quantum neural networks. *Future Directions for Intelligent Systems and Information Sciences: The Future of Speech and Image Technologies, Brain Computers, WWW, and Bioinformatics*, pages 213–235, 2000.

[14] Hironobu Fujiyoshi, Tsubasa Hirakawa, and Takayoshi Yamashita. Deep learning-based image recognition for autonomous driving. *IATSS research*, 43(4):244–252, 2019.

[15] Sanjay Gupta and RKP Zia. Quantum neural networks. *Journal of Computer and System Sciences*, 63(3):355–383, 2001.

[16] Thomas Hubregtsen, Josef Pichlmeier, Patrick Stecher, and Koen Bertels. Evaluation of parameterized quantum circuits: on the relation between classification accuracy, expressibility, and entangling capability. *Quantum Machine Intelligence*, 3:1–19, 2021.

[17] Tak Hur, Leeseok Kim, and Daniel K Park. Quantum convolutional neural network for classical data classification. *Quantum Machine Intelligence*, 4(1):3, 2022.

[18] Radmila Jankovic. Classifying cultural heritage images by using decision tree classifiers in weka. In *Proceedings of the 1st international workshop on visual pattern extraction and recognition for cultural heritage understanding co-located with 15th Italian research conference on digital libraries (IRCDL 2019), Pisa, Italy*, pages 119–127, 2019.

[19] Joseph-Maria Jauch. The problem of measurement in quantum mechanics. *Helv. Phys. Acta*, 37(CERN-TH-389):293–316, 1964.

[20] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[21] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[22] Ankit Kulshrestha and Ilya Safro. Beinit: Avoiding barren plateaus in variational quantum algorithms. In *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 197–203. IEEE, 2022.

[23] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[24] Yann LeCun et al. Lenet-5, convolutional neural networks. *URL: http://yann. lecun. com/exdb/lenet*, 20(5):14, 2015.

[25] Antonio Macaluso, Luca Clissa, Stefano Lodi, and Claudio Sartori. A variational algorithm for quantum neural networks. In *Computational Science–ICCS 2020: 20th International Conference, Amsterdam, The Netherlands, June 3–5, 2020, Proceedings, Part VI 20*, pages 591–604. Springer, 2020.

[26] Stefano Mangini. Variational quantum algorithms for machine learning: theory and applications. *arXiv preprint arXiv:2306.09984*, 2023.

[27] Jarrod R McClean, Sergio Boixo, Vadim N Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature communications*, 9(1):4812, 2018.

[28] Iftikhar Naseer, Sheeraz Akram, Tehreem Masood, Arfan Jaffar, Muhammad Adnan Khan, and Amir Mosavi. Performance analysis of state-of-the-art cnn architectures for luna16. *Sensors*, 22(12):4426, 2022.

[29] Michael A Nielsen and Isaac L Chuang. Quantum computation and quantum information. *Phys. Today*, 54(2):60, 2001.

[30] Michael A Nielsen and Isaac L Chuang. Quantum computation and quantum information, 2010.

[31] Adrián Pérez-Salinas, Alba Cervera-Lierta, Elies Gil-Fuster, and José I Latorre. Data re-uploading for a universal quantum classifier. *Quantum*, 4:226, 2020.

[32] Maniraman Periyasamy, Nico Meyer, Christian Ufrecht, Daniel D Scherer, Axel Plinge, and Christopher Mutschler. Incremental data-uploading for full-quantum classification. In *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 31–37. IEEE, 2022.

[33] Bob Ricks and Dan Ventura. Training a quantum neural network. *Advances in neural information processing systems*, 16, 2003.

[34] Maria Schuld. Supervised quantum machine learning models are kernel methods. *arXiv preprint arXiv:2101.11020*, 2021.

[35] Maria Schuld and Francesco Petruccione. *Supervised learning with quantum computers*, volume 17. Springer, 2018.

[36] Maria Schuld, Ilya Sinayskiy, and Francesco Petruccione. The quest for a quantum neural network. *Quantum Information Processing*, 13:2567–2586, 2014.

[37] Alessandro Sebastianelli, Daniela Alessandra Zaidenberg, Dario Spiller, Bertrand Le Saux, and Silvia Liberata Ullo. On circuit-based hybrid quantum neural networks for remote sensing imagery classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 15:565–580, 2021.

[38] Sukin Sim, Peter D Johnson, and Alán Aspuru-Guzik. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Advanced Quantum Technologies*, 2(12):1900070, 2019.

[39] Andrea Skolik, Sofiene Jerbi, and Vedran Dunjko. Quantum agents in the gym: a variational quantum algorithm for deep q-learning. *Quantum*, 6:720, 2022.

[40] Andrew Steane. Quantum computing. *Reports on Progress in Physics*, 61(2):117, 1998.

[41] Matthew Stewart. Simple introduction to convolutional neural networks. *Towards Data Science*, 27, 2019.

[42] Farrokh Vatan and Colin Williams. Optimal quantum circuits for general two-qubit gates. *Physical Review A*, 69(3):032315, 2004.

[43] ShiJie Wei, YanHu Chen, ZengRong Zhou, and GuiLu Long. A quantum convolutional neural network on nisq devices. *AAPPS Bulletin*, 32:1–11, 2022.

[44] Chen Zhao and Xiao-Shan Gao. Qdnn: Dnn with quantum neural network layers. *arXiv preprint arXiv:1912.12660*, 2019.