POLITECNICO DI TORINO

Master degree course in Electronic Engineering

Master Degree Thesis

Encoding techniques for Quantum Machine Learning

Supervisors
Prof. Maurizio Zamboni
Prof.ssa Mariagrazia Graziano
Prof.ssa Giovanna Turvani

Candidate
Antonio Tudisco

December 2022

Summary

Nowadays, Quantum Computing (QC) and Machine Learning (ML) are two of the most innovative research fields of information technologies. Quantum Machine Learning (QML) merges these two topics, developing models for ML tasks whose computational complexity can be reduced with QC techniques. A relevant ML application is Classification, which identifies the class to which new input data belong, according to a model built during a preliminary learning process. This is achieved on a training dataset composed of features (numerical vectors describing data) and labels (the expected output class). The accuracy of a classifier can be quantified in terms of the total number of correctly predicted outcomes over the total number of processed data. For near-term applications, the limits of current quantum hardware, in terms of execution reliability and scalability, promote the definition of hybrid QML solutions making the best of quantum and classical processing. Among these, the Variational Quantum Circuit and the quantum-kernel-estimation-based Support Vector Machine can be mentioned. The former implements the classification model with a parameterized quantum circuit optimized classically to achieve better accuracy. The other tries to maximize the distinguishability of data belonging to two different classes with a classical optimizer, assisted by quantum computing, mapping features in a higher dimensional space. In both cases, a preliminary encoding operation to represent classical data onto a quantum system is required. Then, specific quantum and classical operations complete Classification according to the hybrid solution and how the information has been represented. This thesis aims to verify that the data encoding strategy influences the model's accuracy, so it must be treated as an optimizable degree of freedom for QML algorithms. In particular, the Amplitude and Angle Encodings, which have the most promising scalability, have been considered. The first one maps data features to the probability amplitudes of the qubits state vector, while the other consists of embedding the data as the angle parameter for rotational gates. In this work, new Angle Encoding techniques have been explored and compared with those already present in the literature to observe the impact on accuracy, examining sixty different strategies. The derived models have been developed and simulated with the Pennylane QML library, while the tests have considered the Iris and Wine datasets to prove the dependence of classification accuracy on encoding. For each

case study, the best encoding strategy has been identified as the best compromise between learning performance requirements and execution time. For each dataset, three different benchmarking classifications have been performed, considering the available classes: 1vsAll, where the model has to identify if new data belong to class 0; 1vs1, in which the classification is accomplished by taking just two among all classes; and Multi-class, for which all classes are evaluated concurrently. From the obtained results, it can be concluded that the best encoding strategy cannot be chosen from previous analyses but depends on the specific case study. Moreover, some encodings can be discarded because they do not separate data effectively. Therefore, the encoding choice is a crucial preliminary operation to be properly pondered for efficient QML model development.

Contents

Part I

Fundamentals

Chapter 1

Quantum Computing

Quantum Computing is an innovative field of computation that tries to **reduce the complexity** of some problems considered hard using classical methods.
It exploits quantum physics phenomena, such as **superposition** and **entanglement** to speed up the processing.
In this chapter, the fundamentals and the characteristics of this approach are presented. Further details can be found in the bibliography [1].

1.1 Qubit

The unit of information for Quantum Computing is the **qubit**, such as the bits for Classical computation. Differently from classical bits, which can only represent 0 or 1 values, the qubit can potentially assume **infinite states** as a linear combination of its basis states, $|0\rangle$ and $|1\rangle$.

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{1.1}$$

The state vector of a single qubit can be written as:

$$|\psi\rangle = \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} = c_0 |0\rangle + c_1 |1\rangle, \tag{1.2}$$

where c_0 and c_1 are complex numbers, called **probability amplitudes**, because $|c_0|^2$ represents the probability of the qubit of measuring it as $|0\rangle$, while $|c_1|^2$ the probability of measuring it as $|1\rangle$. The sum of these probabilities must be equal to 1 and, for this reason, the norm of the state vector is equal to 1.

$$|c_0|^2 + |c_1|^2 = 1 \tag{1.3}$$

The state of the qubit can be also represented graphically using the **Bloch Sphere**. Each point of the Bloch sphere represents a possible state, and it can be represented using the projection along the three axes X, Y, and Z.

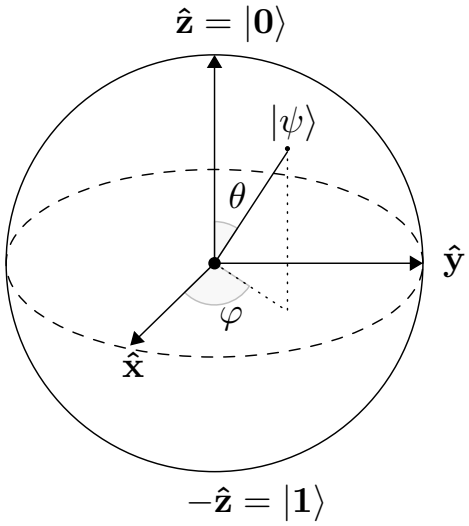


Figure 1.1: Bloch Sphere.

To map the quantum information of a qubit in the sphere, the generic state $|\psi\rangle$ is also expressed as:

$$|\psi\rangle = \cos\theta\,|0\rangle + e^{i\phi}\sin\theta\,|1\rangle \tag{1.4}$$

Where θ and ϕ are the two angles, as defined in fig. 1.1. While ϕ is comprised in the interval $[0, 2\pi]$, θ belongs to $[0, \frac{\pi}{2}]$. On the top of the sphere is mapped the state $|0\rangle$, while on the bottom $|1\rangle$.

1.2 Measurement

Differently from classical computation, **measurement** in quantum computing is an operation that **alters the system** and is a **non-deterministic process**. The measurement operation is done by using an observable Ω. The **reading value** would be the **eigenvalue**, λ, of the observable defined, and the **resulting state** after the operation is the **eigenvector** associated.

Example Consider the measurement of the state $|\psi\rangle$ and the observable Ω.

- $|\psi\rangle = \frac{1}{2}\begin{pmatrix}1\\1\end{pmatrix}$
- $\Omega = \begin{pmatrix}-1 & -i\\ i & 1\end{pmatrix}$

The eigenvalues associated to this specific observable are λ_1 and λ_2 while the eigenvector are e_1 and e_2.

- $\lambda_1 = -\sqrt{2}$
- $\lambda_2 = \sqrt{2}$
- $e_1 = \begin{pmatrix} -0.9239i \\ -0.3827 \end{pmatrix}$
- $e_2 = \begin{pmatrix} -0.3827i \\ 0.9239 \end{pmatrix}$

Then, the probability of the state $|\psi\rangle$ to collapse in $|e_1\rangle$ is calculated doing the square of the inner product of the two states, so the square of the product of $|e_1\rangle$ by the conjugate and transpose of the state vector $|\psi\rangle$. The same is done for the probability of $|\psi\rangle$ to go into $|e_2\rangle$.

$$p_1 = |\langle\psi|e_1\rangle|^2 \qquad p_2 = |\langle\psi|e_2\rangle|^2 \tag{1.5}$$

The **expected value** of the measurement, denoted by the expression $\langle\psi|\Omega|\psi\rangle$, can be determined by summing the contributions of each eigenvalue multiplied by the probability of the state to collapse in the corresponding eigenvector state.

$$\langle\psi|\Omega|\psi\rangle = p_1 \times \lambda_1 + p_2 \times \lambda_2 = 0. \tag{1.6}$$

1.3 The basis of Quantum Computing

The two most important quantum mechanics properties exploited in Quantum Computation are:

- **Superposition**
- **Entanglement**

The **superposition** is achieved when the qubit has a no-null probability of being in both the state $|0\rangle$ and $|1\rangle$. By exploiting this effect, it is possible to represent 2^n states simultaneously from a system with n qubits. A graphical representation of the superposition is shown in fig. 1.2.

The other property examined is the **entanglement**. It involves the correlation of two qubits. The most iconic example that describes this phenomenon is the **Bell States**, which represents a correlation between two qubits. The correlation can be **positive** if the two qubits at the output are in the same state, or **negative** if the two qubits are in the opposite state. A graphical representation is shown in 1.3 .

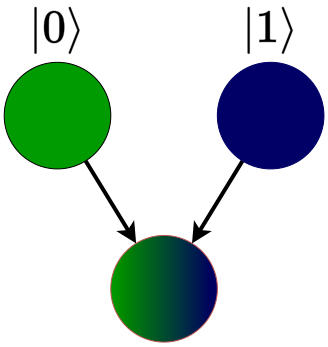


Figure 1.2: Superposition representation. The green color represents the state $|0\rangle$, while the blue is associated with state $|1\rangle$. The other ball, colored with shades of green and blue, is in superposition.

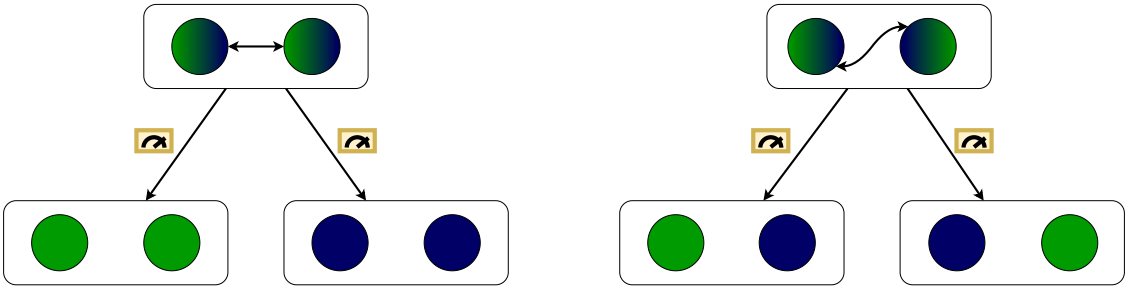


Figure 1.3: Representation of the Bell states. On the left, an example of a positive correlation of the two qubits is shown, which collapse both in $|0\rangle$ or $|1\rangle$ when measured. On the right, the two qubits are anti-correlated and collapse in the opposite state.

1.4 Quantum gates

The qubits are manipulated by applying transformations called quantum gates. They are associated with matrices that, by multiplying the initial state, transform it into a new one, as shown in the following equation, where G is the quantum gate applied.

$$|\psi^{(t+1)}\rangle = G\,|\psi^{(t)}\rangle \tag{1.7}$$

The main properties related to these transformations are:

- Linearity
- Unitary
- Reversibility

Linearity The quantum gates are linear transformations that are applied to the state vector.

Unitary A matrix is unitary if:

$$UU^\dagger = U^\dagger U = I \tag{1.8}$$

where $U^\dagger$ is the conjugate and transpose matrix of U. It guarantees that the state vector norm remains constant.

Reversibility Quantum gates are reversible and their input quantum state can be always reconstructed through inverse transformation.

The gates presented in this thesis are the most common for Quantum Machine Learning and do not cover the totality of the possible gates. The presented ones are:

- The Pauli gates
- The Hadamard gate
- The Rotational gates
- The Controlled gates

Pauli, Hadamard, and Controlled gates belong to the Clifford set, while the others compose the Rotational set. Each gate is associated with a matrix, which describes the operation.

Pauli gates The application of the Pauli gates rotates the state of the qubit of 180° on the Bloch Sphere around the corresponding axis X, Y, or Z.

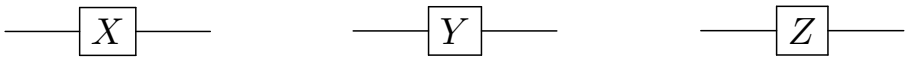


Figure 1.4: Graphical representation of the Pauli gates X, Y, and Z.

The corresponding matrices are

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{1.9}$$

Considering the Bloch sphere, if a qubit is in $|0\rangle$ and a Pauli gate is applied, the qubits will be in $|1\rangle$ due to a rotation of 180° concerning the X-axis. Vice versa, using a Pauli gate to a qubit in $|1\rangle$, it will rotate to $|0\rangle$. For this reason, this transformation is associated to the classical NOT gate.

Hadamard The Hadamard, shown in fig. 1.5, is one of the most exploited gates in Quantum Computing. It permits obtaining the uniform superposition starting from the ground state, where all qubits are in $|0\rangle$.

Figure 1.5: Representation of the Hadamard gate.

The matrix that describes this behavior is presented in 1.10.

$$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix} \tag{1.10}$$

If this gate is applied to a qubit in $|0\rangle$, it will be rotated to $|+\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\1\end{pmatrix}$, while if the initial state is $|1\rangle$, the resulting vector is $|-\rangle = \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix}$

Rotational gates The rotational gates, shown in fig. 1.6, rotate the state of the qubit along one of the three axes X, Y, and Z by an angle θ passed as a parameter.

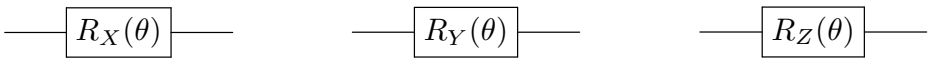


Figure 1.6: Representation of the rotational gates, which rotates the state along the X, Y, and Z axis respectively.

They can be defined as exponential matrices with the corresponding Pauli matrix passed as an argument:

$$\begin{aligned}
R_X(\theta) &= e^{-i\frac{\theta}{2}X} = \begin{pmatrix}\cos\frac{\theta}{2} & -i\sin\frac{\theta}{2}\\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2}\end{pmatrix}\\
R_Y(\theta) &= e^{-i\frac{\theta}{2}Y} = \begin{pmatrix}\cos\frac{\theta}{2} & \sin\frac{\theta}{2}\\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2}\end{pmatrix}\\
R_Z(\theta) &= e^{-i\frac{\theta}{2}Z} = \begin{pmatrix}e^{-i\frac{\theta}{2}} & 0\\ 0 & e^{i\frac{\theta}{2}}\end{pmatrix}
\end{aligned} \tag{1.11}$$

Controlled gates The controlled gates are multiple input gates in which there are one or more qubits used as control and a single target qubit. The example reported in figs. 1.7 and 1.8 is the C-NOT gate, where the Pauli-X gate implements the transformation if the control qubit is in the state $|1\rangle$.

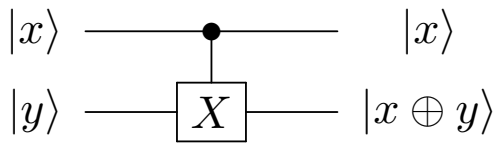


Figure 1.7: Graphical representation of the C-NOT gate, where the top qubit in state $|x\rangle$ is the control, while the bottom qubit in state $|y\rangle$ is the target.

$\|x\rangle$	$\|y\rangle$	$\|\psi_{out}\rangle$
$\|0\rangle$	$\|0\rangle$	$\|0\rangle$
$\|0\rangle$	$\|1\rangle$	$\|1\rangle$
$\|1\rangle$	$\|0\rangle$	$\|1\rangle$
$\|1\rangle$	$\|1\rangle$	$\|0\rangle$

Figure 1.8: Representation of the truth table related to the circuit in fig. 1.7. The input qubits are in state $|x\rangle$ and $|y\rangle$, while $|\psi_{out}\rangle$ is the state of the bottom qubit after the application of the CNOT.

The resulting operation for the target qubit is similar to the XOR function because it is in the state $|0\rangle$ if both the input qubit are in the same state, else the output qubit state is $|1\rangle$.

Also, considering the controlled gates, another example is the Toffoli Gate. For this gate, the Pauli-X gate is active if both the 2-control qubits are in the state $|1\rangle$.

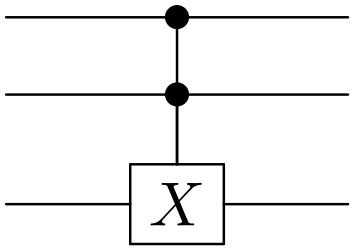


Figure 1.9: Graphical representation of the Toffoli gate. The bottom qubit is the target, while the others are the controls.

1.5 Encoding

The goal of encoding is to represent the data in a quantum system. Different methods can be implemented but the ones explored in this thesis are **Basis**, **Amplitude**, and **Angle** encoding, which would be presented in the following sections.

The information described can be deepened in [2].

1.5.1 Basis Encoding

Basis Encoding consists in mapping the data, written in binary, by maximizing the probability amplitude of the corresponding basis state. In order to do this, Pauli-X gates are applied.

Example Consider to encode a vector x = (0.1 , -0.6 , 1.0). The elements are encoded using the sign and magnitude technique, and for this example the precision is supposed to be $\tau = 4$.

$$0.1 \rightarrow 0\,0001$$
$$-0.6 \rightarrow 1\,1001$$
$$1.0 \rightarrow 0\,1111$$

The resulting state will be:

$$|\psi\rangle = |00001\,11001\,01111\rangle$$

The limiting aspect of this encoding strategy is related to the number of qubits, which are limited in current devices. Supposing to encode the feature -0.6, it would be necessary to apply a Pauli-X gate for each non-zero bit of the value, as shown in fig. 1.10.

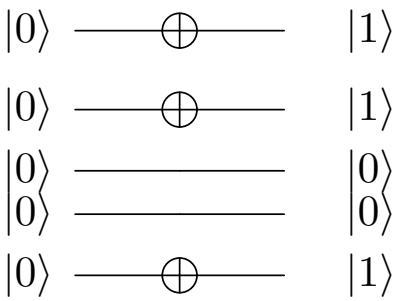


Figure 1.10: Example of basis encoding for the parameter -0.6.

1.5.2 Amplitude Encoding

Amplitude Encoding is another encoding technique that consists of encoding a value as a probability amplitude associated with the reference state vector.

Example Consider the same case of the Basis encoding. Firstly, it is necessary to normalize the vector, in order to embed it as a state vector. For the normalization, each element is divided by the norm-2 of the vector.

$$\begin{aligned} \frac{x}{norm(x)} &= \frac{1}{\sqrt{0.1^2 + 0.6^2 + 1.0^2}} \begin{pmatrix} 0.1 & 0.6 & 1.0 & 0 \end{pmatrix} = \\ &= \begin{pmatrix} 0.085 & -0.513 & 0.854 & 0.000 \end{pmatrix} \end{aligned} \tag{1.12}$$

The resulting state is:

$$|\psi\rangle = 0.0854\,|00\rangle - 0.513\,|01\rangle + 0.854\,|10\rangle + 0\,|11\rangle \tag{1.13}$$

To prepare the desired state, RY and CNOT gates can be used and an example is proposed in fig. 1.11 whose results in output in terms of amplitudes and probabilities are shown in fig. 1.12.

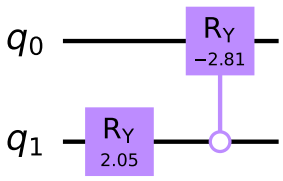


Figure 1.11: Example of a quantum circuit capable of preparing the state $|\psi\rangle = 0.0854\,|00\rangle - 0.513\,|01\rangle + 0.854\,|10\rangle + 0\,|11\rangle$

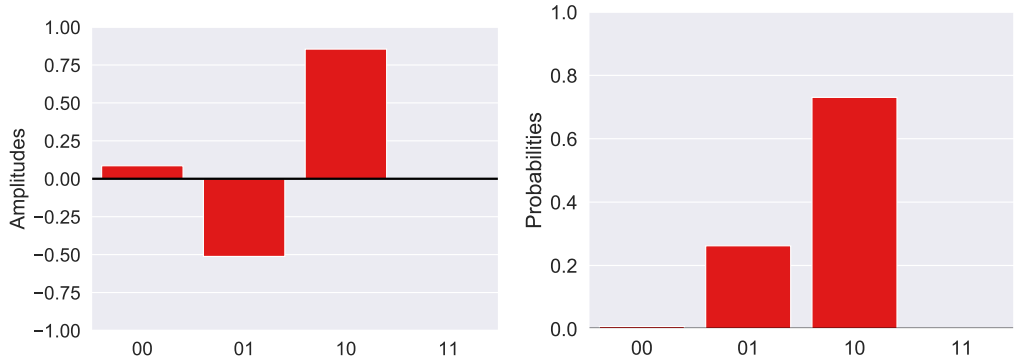


Figure 1.12: Amplitudes and probabilities for the desired encoded state

The number of qubits required for this encoding is $\log_2 N$, where N is the number of elements of the vector. It represents an advantage with respect to the other techniques, but it requires a huge number of gates that will slow the computation.

1.5.3 Angle encoding

Angle Encoding consists in embedding the data using rotational gates. The value to encode (i.e. the feature) is passed as an angle parameter, and the resulting state depends on the rotational gate applied.

Example Supposing to encode the value -0.513 using RY gates, the resulting state will be:

$$|\psi(-0.513)\rangle = \cos(-0.513/2)\,|0\rangle + \sin(-0.513/2)\,|1\rangle \tag{1.14}$$

For this encoding technique, the number of qubits required is proportional to the number of elements. For this reason, it requires more qubits, but it is faster than the Amplitude Encoding state preparation.

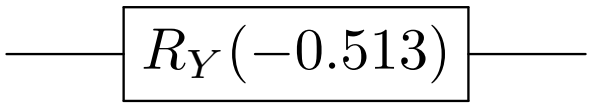


Figure 1.13: Example of angle encoding for the parameter 0.513 using the R-Y gate

Chapter 2

Introduction to Machine Learning

Machine Learning is a research field of Artificial Intelligence that became popular in the last few years. It can be used for solving problems that are too computationally expensive using classical programming techniques. The basic idea of Machine Learning is to define models that generalize from data to solve these kinds of problems. The information described in this chapter is taken from [2].
A generic definition, proposed by Mitchell [3], of Machine Learning is:

> A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

According to this definition, it is possible to analyze the principal elements of a Machine Learning algorithm, that would be:

- The Task
- The Model
- The Training process
- The Metrics

2.1 Task

The objective of each Machine Learning model is to create a model capable of producing an answer for a problem, given an initial dataset.
The learning problem can be categorized into three different approaches:

- Supervised Learning

- Unsupervised Learning
- Reinforcement Learning

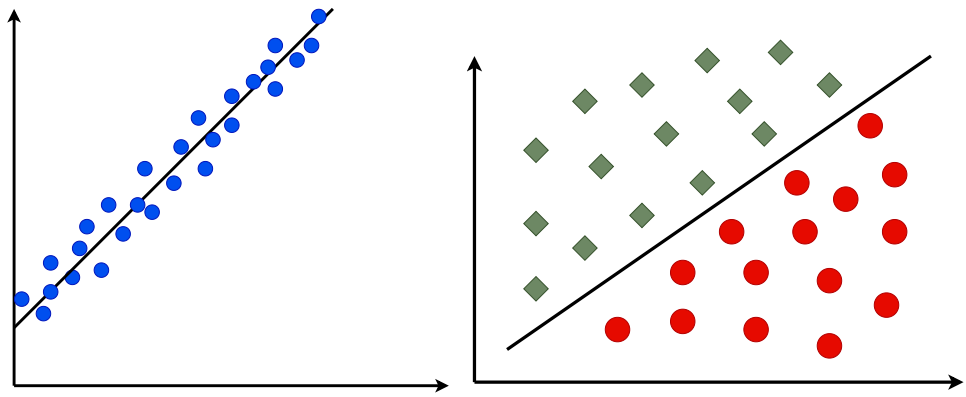
For this thesis, it is analyzed just Supervised Learning.

Supervised Learning Its goal is to predict the label of a new unclassified data vector, starting from a dataset composed of features and target labels.
The main tasks for this kind of learning are:

- Regression
- Classification

For the regression, the predicted value is a real number. A typical example of this task is the value of houses, which can depend on different features like where the house is located (the latitude and longitude), the number of bedrooms, the number of bathrooms, and others.

Differently from the regression, the predicted label of a classification problem is a discrete value. The classification can be **binary**, **multiclass**, or **multilabel**. Binary Classification consists of predicting the label of a data sample from two mutually exclusive classes. For Multiclass classification, the possible classes are more than two. Finally, multi-label classification assigns zero or more labels to each data sample.



(a) Example of a regression problem.

(b) Example of a binary classification.

Unsupervised Learning It has to recognize the characteristics of the dataset. The dataset is composed of features with no labels.
The most diffused tasks are the **Principal Component Analysis (PCA)**, where

the algorithm has to identify the principal components of a dataset, and the **Clustering**, in which, given the number of clusters, the algorithm finds the data group similar to each other.

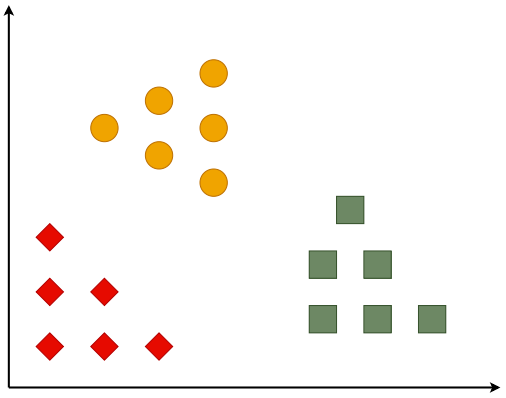


Figure 2.2: Example of a Clustering task in which the number of clusters is set to 3.

Reinforcement Learning Here, the model is inserted in a close loop with the environment. The model has to generalize from experience and with the environment. The most iconic problem is the computer that plays chess.

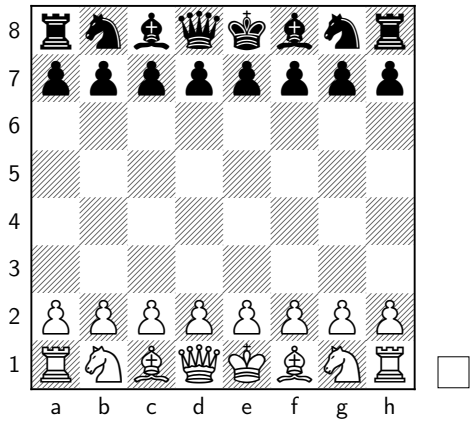


Figure 2.3: Example of a chess board that represents Reinforcement Learning.

2.2 Model

As previously stated, it is critical in Machine Learning to define a model, which is a collection of functions, algorithms, or rules that define a relationship between input data and the output outcomes. This relationship is also defined by a set of trainable parameters that will be optimized to get the predicted value as close to the real one as possible.

2.3 Train

The training part is the heart of Machine Learning algorithms. In this process, the parameters are optimized to achieve the best performance possible.
To do that, it is important to first define the Cost function. It is constituted of two contributions:

- The Loss
- The Regularisation

2.3.1 Loss function

The **Loss** term is used to calculate the difference between predicted and actual values. There are several loss functions, some of which are the following.

Squared Loss The Squared Loss function is the most well-known, which computes the **squared distance** between the prediction and the expected value.

$$L = \frac{1}{2}(f(x;\theta) - y)^2 \tag{2.1}$$

where $f(x;\theta)$ is the outcome of the model which depends on the input data x and on the parameter θ, while y is the expected label.

Absolute Loss The Absolute Loss is similar to the squared loss, substituting the square with the absolute value.

$$L = \frac{1}{2}|f(x;\theta) - y| \tag{2.2}$$

Hinge Loss The Hinge Loss is useful for binary classification, in which the classes are 1 or -1.

$$L = \max(0, 1 - f(x;\theta)y) \tag{2.3}$$

The result is in the interval [0, 2], with 0 indicating that the prediction is equal to the true value and 2 indicating that the prediction class differs from the expected.

Logistic Loss Logistic Loss is determined starting from the logistic function. It is used for binary classification in which the classes are positive and negative. If the prediction and the target are different, the error is maximized.

$$L = log(1 + e^{-yf(x;\theta)} \tag{2.4}$$

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:

$$-y \log p - (1 - y) \log(1 - p) \tag{2.5}$$

For **multi-class** classification, the classifications in which the outputs of the model are mutually exclusive, the loss function will be equal to the logarithm of the probability of obtaining the class as result.

$$L = \log(p(x, y|\theta)) \tag{2.6}$$

While for **multi-label** classification, in which the outputs are not mutually exclusive, the loss function is:

$$L = \sum_{d=1}^{D} y_d \log p \tag{2.7}$$

2.3.2 Regularisation term

The **regularisation** is an additional term that can be added to reduce the **overfitting**. Overfitting is a Machine Learning problem in which trained models are incapable of generalizing from data. To determine whether this phenomenon occurs, dataset is divided into two parts: training and testing. The training section is used to train the model, and the test section is used to evaluate its performance. Another Machine Learning problem is the **underfitting**, which happens when the model is not capable of visualizing the relationship between inputs and outputs.
The regularisation term is associated with a parameter λ that if it is too low, the model may overfit; if it is too large, the model may underfit.

$$Cost = Loss + \lambda \text{Reg} \tag{2.8}$$

2.3.3 Optimization function and Gradient Descent

As previously stated, the objective of the training is the minimization of the errors accomplished by the Machine Learning models, that in this case is represented by the cost function.

To do that, at each step every parameter θ is subtracted by the derivative of the cost function to the parameter analyzed.

$$\theta^{(t+1)} = \theta^{(t)} - \eta \frac{dC}{d\theta}, \tag{2.9}$$

where η is the learning rate, a parameter that set the step size.
The algorithm presented is related to the Gradient Descent [4]. The entire training set is run through the model, and after each iteration, the derivatives and parameters are calculated and updated. Other approaches would include the Stochastic Gradient Descent [5], which speeds up the algorithm by passing only a subset of the entire training set and updating the parameters each time. The problem with this algorithm is that it does not guarantee that the cost function will be reduced each iteration.

2.4 Important Metrics

The evaluation of the performance of a Model is defined by some metrics. The most important ones are the following:

- Accuracy
- Precision
- Recall
- F1-score

To describe them, it can be useful to define the **Confusion matrix**, which is a matrix for evaluating the performance of the model. In this matrix, the real labels are defined on the column, while the predicted ones on the rows. Then, the results of a binary classification (-1, 1) can be defined as:

- **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.

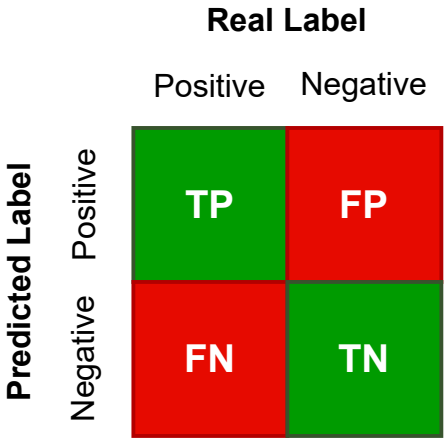


Figure 2.4: Confusion matrix

2.4.1 Accuracy

The accuracy of the model is the number of data well predicted over the entire set of elements evaluated.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2.10}$$

This metric is useful to evaluate the model performance if the number of elements belonging to a class is the same for the other. Otherwise, it's possible to achieve a good result of accuracy in classifying the whole dataset with just one class.

2.4.2 Precision

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

$$Precision = \frac{TP}{TP + FP} \tag{2.11}$$

2.4.3 Recall

The Recall is the number of corrected predicted positives over the number for which the real label is the positive one.

$$Recall = \frac{TP}{TP + FN} \tag{2.12}$$

2.4.4 F1-Score

In some cases, the Precision and the Accuracy are not well balanced, for this reason, to choose the best model, F1-score can be defined as an additional metric to achieve

the best trade-off. 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.

$$F1 = 2\frac{Precision * Recall}{Precision + Recall} \tag{2.13}$$

Chapter 3

Quantum Machine Learning

The most important models for Quantum Machine Learning, related to classification tasks, are the Variational Quantum Classifier [6] and the Support Vector Machine with the Quantum Kernel Estimation [7]. The models proposed are hybrid, therefore, there is an evaluation task based on Quantum devices while the optimization one is done classically. These models can be developed also in near-term quantum devices because require a limited number of qubits.

3.1 Variational Quantum Classifiers

The variational quantum classifier is a model composed of an encoding part, to represent the input classical data in the quantum devices, and a variational block, composed of rotational gates in which the parameters are the rotational angles and whose values are optimized by classical methods.
The training set is passed to the variational quantum circuit to optimize the parameters, and then, for the evaluation, just the encoding part is changed.

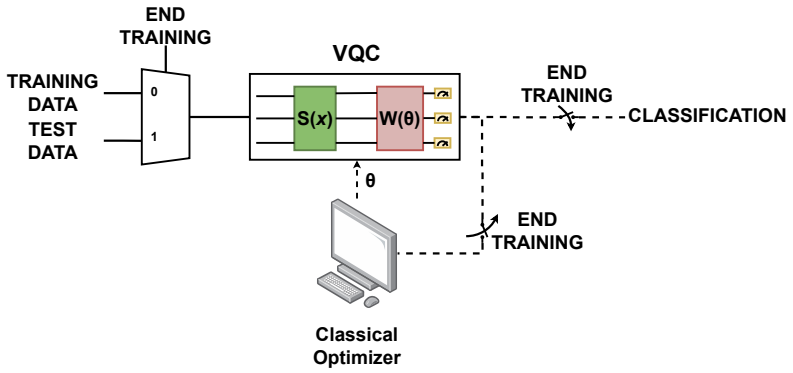


Figure 3.1: Variational Quantum Circuit

3.1.1 Measurement

For the binary classifications, the measurement is done on a qubit and the expected value is read using a Pauli gate as observable, generally, the Pauli-Z that has the eigenvalue 1 for the state $|0\rangle$ and -1 for the state $|1\rangle$. The equation that describes the measurement operation is the following:

$$Tr(\rho\sigma_z) = \langle \rho \mid \sigma_z \mid \rho \rangle \tag{3.1}$$

Where ρ represents the density matrix for the measured qubit. To describe the measurement, using the observable Pauli-Z, the following image is used. The probabilities of the qubit to be in states $|0\rangle$ and $|1\rangle$ are calculated, then they are multiplied by the corresponding eigenvalue for the states $|0\rangle$ and $|1\rangle$. Then, the two quantities are summed with a trainable parameter b to obtain the prediction value.

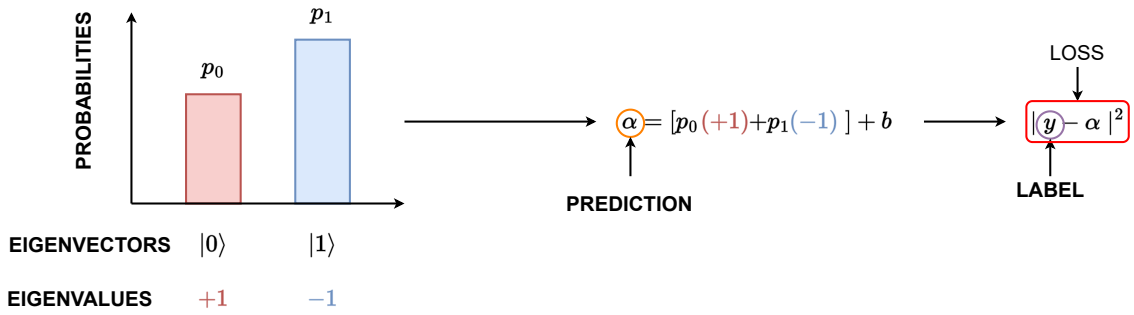


Figure 3.2: Representation of how the measurement is done for the VQC. The two probabilities of the qubit to collapse to the eigenvectors $|0\rangle$ and $|1\rangle$ are multiplied by the corresponding eigenvalue and summed. Then, the sum with a trainable bias parameter b gives the result of the prediction.

3.2 Support Vector Machine with quantum kernel estimation

The Support Vector Machine model is a type of model that is also used in classical computation. The idea, for the classification task, is to maximize the margins, which are the distances between the two classes.
For non-linearly separable classes, the kernel functions are used. These functions map the data points in a higher-dimensional space. Originally the dot product is applied for each data point to each data of the training set.
Using the kernel trick, other kernel functions can be used, like the Gaussian. The resulting data are then saved in a $M \times M$ matrix and the optimization started.

In quantum, there are two different methods in the State-of-Art, one explicit and the other one implicit. The explicit method consists in apply also the optimization task in quantum, which is the method proposed by Rebentrost [8]. The implicit one consists of doing the kernel evaluation in quantum and the optimization classically. To do that, a possible solution consists in applying an encoding circuit to encode the data x, and an adjoint circuit to encode the data t. Then, measure the probability of getting the ground state in output.

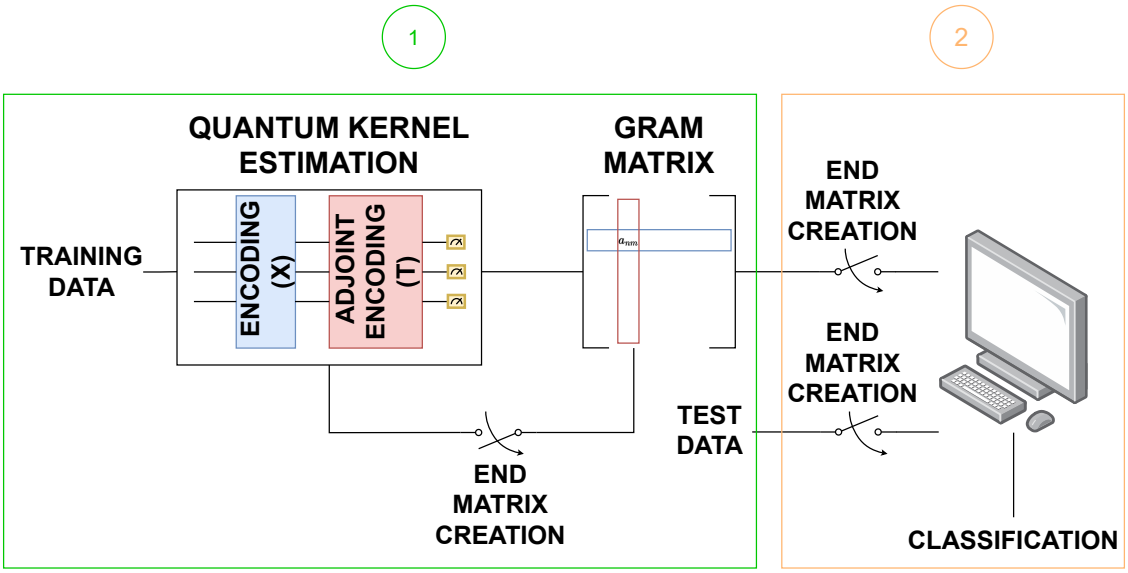


Figure 3.3: Support Vector Machine with quantum kernel estimation

3.3 Figures of Merits

Considering both the approaches described, some Figures of Merit can be described. The ones in common are **depth**, **complexity**, and **number of qubits**. All of them influence the simulation time.

Depth

The depth is a relevant Figure of Merit that represents the longest path of a quantum circuit. If the depth increases, the overall processing requires more sequential transformations, enlarging the simulation time and the probabilities of suffering non-ideal phenomena, like decoherence and relaxation. For these reasons, if two models guarantee the same accuracy for the classification in an ideal simulation, the one with the less depth can be considered as more profitable.

Complexity

The complexity represents the total number of quantum gates applied to the quantum circuit. This metric influences the simulation time and is often correlated with the depth.

Number of qubits

The number of qubits required by the system is an important metric, especially for the limited capabilities of near-term devices. It is related to the encoding strategy adopted. For example, defining N as the number of features, the number of qubits needed using the Angle encoding strategies is equal to N, while are necessary $log_2 N$ qubits using the Amplitude approach.

Part II

Implementation and result of the Quantum Machine Learning Models

Chapter 4

Implementation of the thesis

4.1 Goal of the Thesis

As described in the previous chapters, the objective of Machine Learning is to define the best possible model that better classifies data. This thesis aims to verify how the choice of the **encoding mechanism influences the model performance**.
This hypothesis is verified using two different approaches, the **Variational Quantum Circuit** and the **Support Vector Machine with the quantum kernel estimation**, hybrid models that do not require many qubits and, for this reason, can be implemented in actual devices.
For the simulation of the developed circuits, the Pennylane [9] library is used, which permits the management of the training operation as in the classical neural network case with the PyTorch [10] and Tensorflow [11] interface commonly employed in classical Machine Learning.

4.2 Variational Quantum Classifier

As mentioned in section 3.1, the **Variational Quantum Classifier** (VQC) is composed of an **encoding** circuit used to represent data and a **parametric** part to process them. The variational block is defined by rotational gates, in which the parameter angles associated are optimized at each step to achieve a better classification. After defining the quantum circuit, the top C qubits of the system are measured using the Pauli-Z observable. The parameter C depends on the classification task, equal to 1 for the binary classification; otherwise, equivalent to the number of classes.

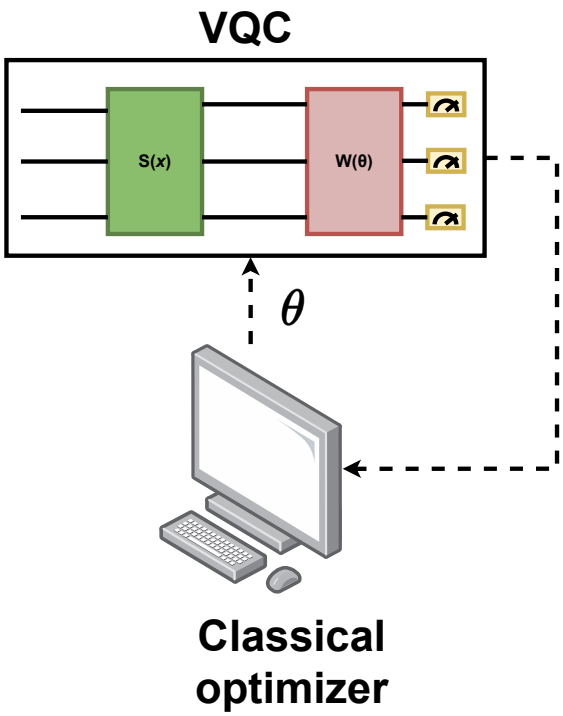


Figure 4.1: The figure represents a Variational Quantum Circuit model. The green rectangle represents the encoding part, while the red is the parametric block. The parameters of the variational block are updated using a classical optimizer.

4.2.1 Encoding strategies for Variational Quantum Classifiers

The goal of the encoding circuit is to represent data in the quantum circuit, as mentioned in 1.5. The strategies evaluated are **Angle** and **Amplitude** encoding. Due to the limited number of qubits of the actual devices, the Basis encoding strategy is not exploited in this thesis.

Angle Encoding

As described in section 1.5.3, using the Angle encoding technique, the numerical value associated with a specific characteristic of the data analyzed, i.e. the feature, is embedded as a parametric angle for a set of rotational gates. This approach requires N-qubit systems to encode a vector, where N is the number of elements.

The following sections describe the developed strategies related to this encoding strategy.

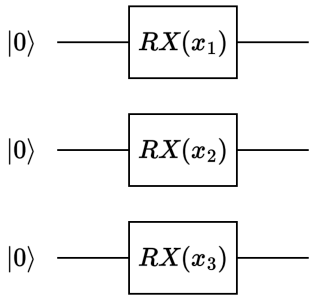


Figure 4.2: Representation of a quantum circuit for which the feature vector, composed of three values, is embedded using the RX gates.

Rotational gates used for the data encoding The performance of the model can depend on the type of rotational gate used for data encoding. This thesis evaluates strategies with a series of single, double, or triple rotational gates, considering a combination of RX, RY, and RZ gates.
Figure 4.3 shows the exploited strategies related to the rotational part.

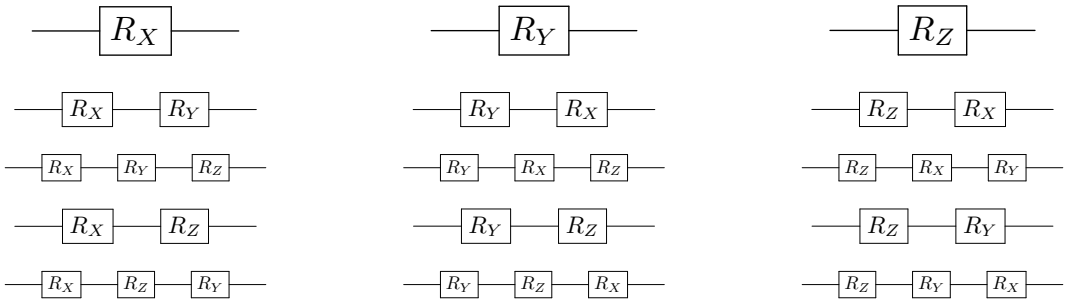


Figure 4.3: The possible combinations of rotational gates are here reported. The parameter angle is the same for all the rotational gates placed on the same qubit. On the top of the figure, there are the single gate strategies, using a single rotational gate. Then, the strategies that use 2 or 3 rotational gates are presented.

Uniform superposition strategy An exploited strategy consists of applying the uniform superposition before the rotational gates. The idea is to remove the offset on the measurement that uses the Pauli-Z observable, as described in the section 1.2. The initial state is in a uniform superposition of $|0\rangle$ and $|1\rangle$, which guarantees to measure the expected value 0 using the Pauli-Z observable. The models that use this property have an initial layer made by Hadamard gates applied on each qubit before the rotational encoding gates.

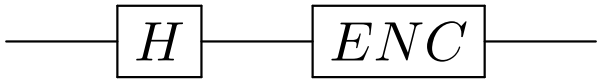


Figure 4.4: Uniform superposition strategy that consists of the application of the Hadamard gate on each qubit of the system right before the rotational gates.

Analysis of the encoding techniques

For the goal of this thesis, an **initial study of the encoding circuit is crucial**. Each encoding circuit is evaluated by changing the angle parameter in the interval $[0, \pi]$ and observing the effect on the output using the Pauli-X, Pauli-Y, and Pauli-Z observables. The results are shown in Figures figs. 4.5 to 4.13.

Among the different graphs, there are some which deserve an additional consideration. For example, let's consider the curves in the presented images are constant, the model is not working correctly, considering that the encoding does not map each feature in a different point of the block sphere. For this reason, **RZ and H-RX strategies**, whose state vector is on the same axis of rotation, **can be discarded** for the realization of the VQC model.
Moreover, it is also possible to observe that some encoding strategies are similar to others. For example, let's consider the RX and RZ-RX encodings.

$$RX(\theta)\left|0\right\rangle = \begin{pmatrix} cos(\frac{\theta}{2}) & -isin(\frac{\theta}{2}) \\ -isin(\frac{\theta}{2}) & cos(\frac{\theta}{2}) \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} cos(\frac{\theta}{2}) \\ -isin(\frac{\theta}{2}) \end{pmatrix} \tag{4.1}$$

$$\begin{aligned} RX(\theta)RZ(\theta)\left|0\right\rangle &= \begin{pmatrix} cos(\frac{\theta}{2}) & -isin(\frac{\theta}{2}) \\ -isin(\frac{\theta}{2}) & cos(\frac{\theta}{2}) \end{pmatrix} \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} cos(\frac{\theta}{2}) & -isin(\frac{\theta}{2}) \\ -isin(\frac{\theta}{2}) & cos(\frac{\theta}{2}) \end{pmatrix} \begin{pmatrix} e^{-i\frac{\theta}{2}} \\ 0 \end{pmatrix} \\ &= e^{-i\frac{\theta}{2}} \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) \end{pmatrix} \end{aligned} \tag{4.2}$$

As can be seen from eqs. (4.1) and (4.2), the **difference** between the two encoding strategies is simply a phase rotation that **would not influence the measurement**, and, therefore, the two techniques are equivalent. See eqs. (4.3) and (4.4)

for the mathematical proof.

$$\begin{aligned} p_1 = |\langle\psi|v_1\rangle|^2 &= \left|\begin{pmatrix}\cos\left(\frac{\theta}{2}\right) & i\sin\left(\frac{\theta}{2}\right)\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}\right|^2 = \left|\cos\left(\frac{\theta}{2}\right)\right|^2 \\ p_2 = |\langle\psi|v_2\rangle|^2 &= \left|\begin{pmatrix}\cos\left(\frac{\theta}{2}\right) & i\sin\left(\frac{\theta}{2}\right)\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix}\right|^2 = \left|\sin\left(\frac{\theta}{2}\right)\right|^2 \\ p_1 \times \lambda_1 + p_2 \times \lambda_2 &= \left|\cos\left(\frac{\theta}{2}\right)\right|^2 \times 1 + \left|\sin\left(\frac{\theta}{2}\right)\right|^2 \times -1 = \\ &= \left|\cos\left(\frac{\theta}{2}\right)\right|^2 - \left|\sin\left(\frac{\theta}{2}\right)\right|^2 \end{aligned} \tag{4.3}$$

$$\begin{aligned} p_1 = |\langle\psi|v_1\rangle|^2 &= \left|e^{i\frac{\theta}{2}}\begin{pmatrix}\cos\left(\frac{\theta}{2}\right) & i\sin\left(\frac{\theta}{2}\right)\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}\right|^2 = \left|\cos\left(\frac{\theta}{2}\right)\right|^2 \\ p_2 = |\langle\psi|v_2\rangle|^2 &= \left|e^{i\frac{\theta}{2}}\begin{pmatrix}\cos\left(\frac{\theta}{2}\right) & i\sin\left(\frac{\theta}{2}\right)\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix}\right|^2 = \left|\sin\left(\frac{\theta}{2}\right)\right|^2 \\ p_1 \times \lambda_1 + p_2 \times \lambda_2 &= \left|\cos\left(\frac{\theta}{2}\right)\right|^2 \times 1 + \left|\sin\left(\frac{\theta}{2}\right)\right|^2 \times -1 = \\ &= \left|\cos\left(\frac{\theta}{2}\right)\right|^2 - \left|\sin\left(\frac{\theta}{2}\right)\right|^2 \end{aligned} \tag{4.4}$$

All the equivalent encodings strategies are shown in table 4.1. During the tests, only the more compact circuit representations are taken into account (the longest encoding circuits from the list can be eliminated).

Table 4.1: Equivalent encoding circuit strategies.

Circuit	Equivalent Circuit
RZ-RX	RX
RZ-RX-RY	RX-RY
RZ-RY	RY
RZ-RY-RX	RY-RX
H-RX-RY	H-RY
H-RX-RY-RZ	H-RY-RZ
H-RX-RZ	H-RZ
H-RX-RZ-RY	H-RZ-RY

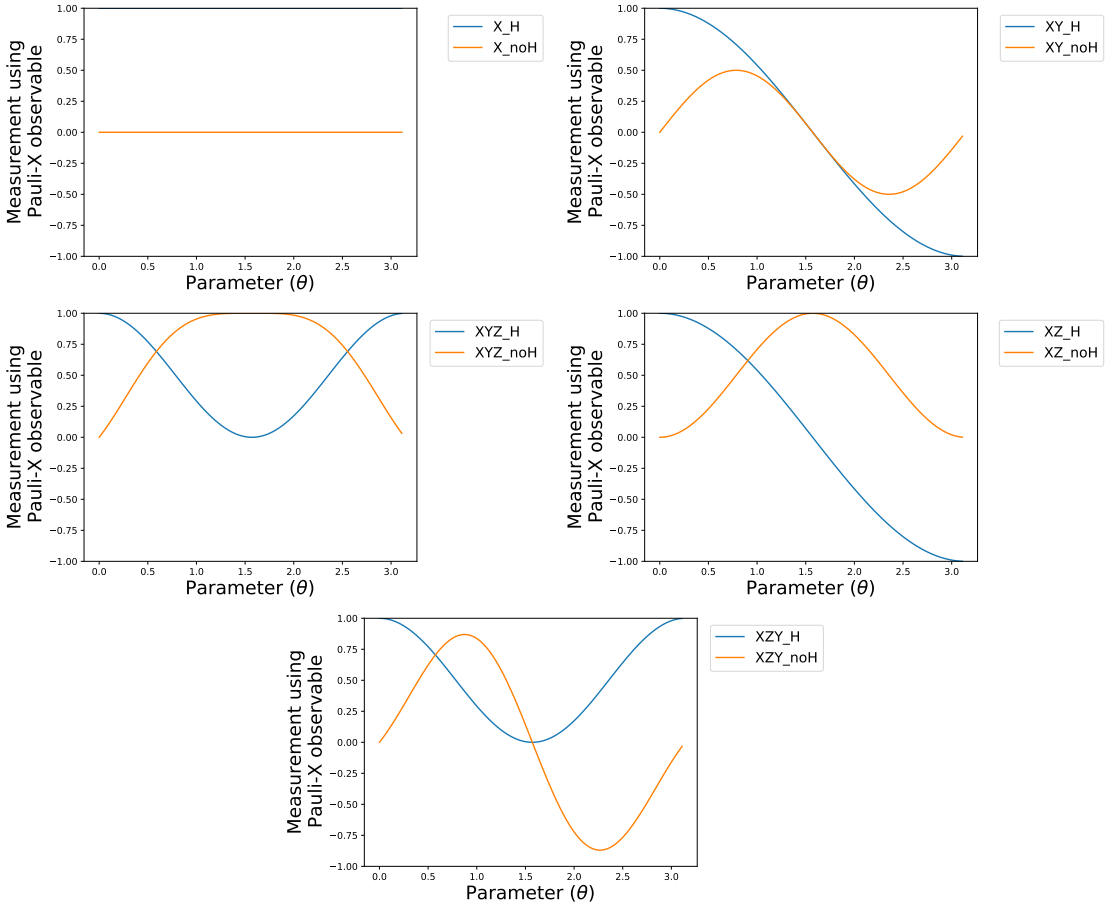


Figure 4.5: Evaluation of encoding strategies that use the rotational gates RX, RX-RY, RX-RY-RZ, RX-RZ, RX-RZ-RY measured using the Pauli-X observable.

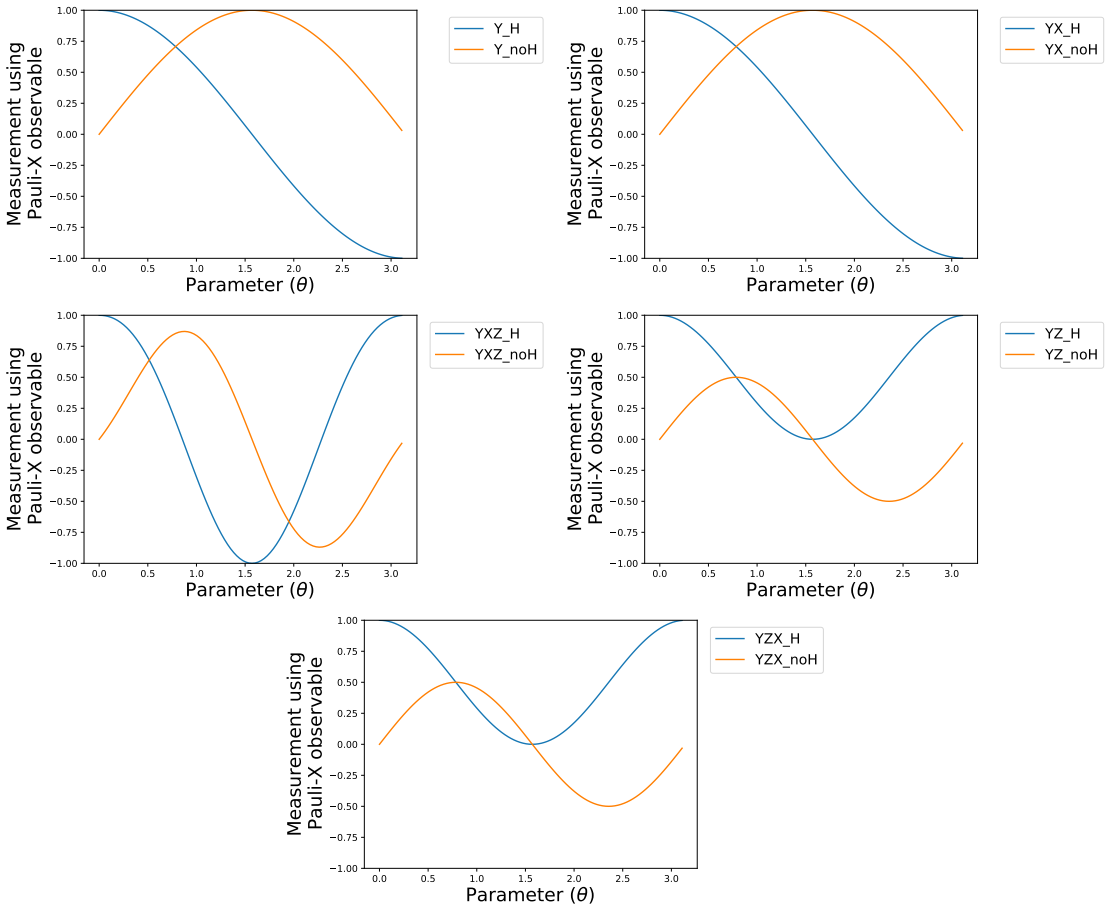


Figure 4.6: Evaluation of encoding strategies that use the rotational gates RY, RY-RX, RY-RX-RZ, RY-RZ, RY-RZ-RX measured using the Pauli-X observable.

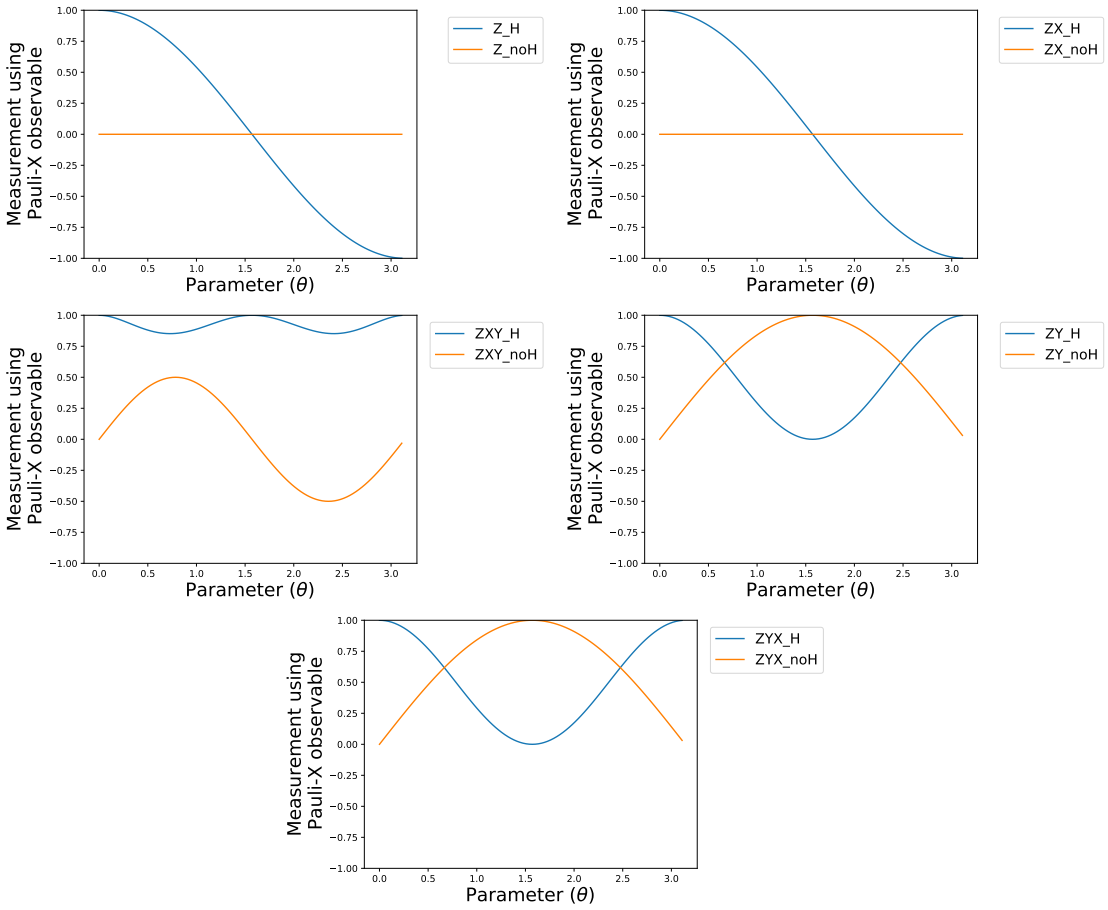


Figure 4.7: Evaluation of encoding strategies that use the rotational gates RZ, RZ-RX, RZ-RX-RY, RZ-RY, RZ-RY-RX measured using the Pauli-X observable.

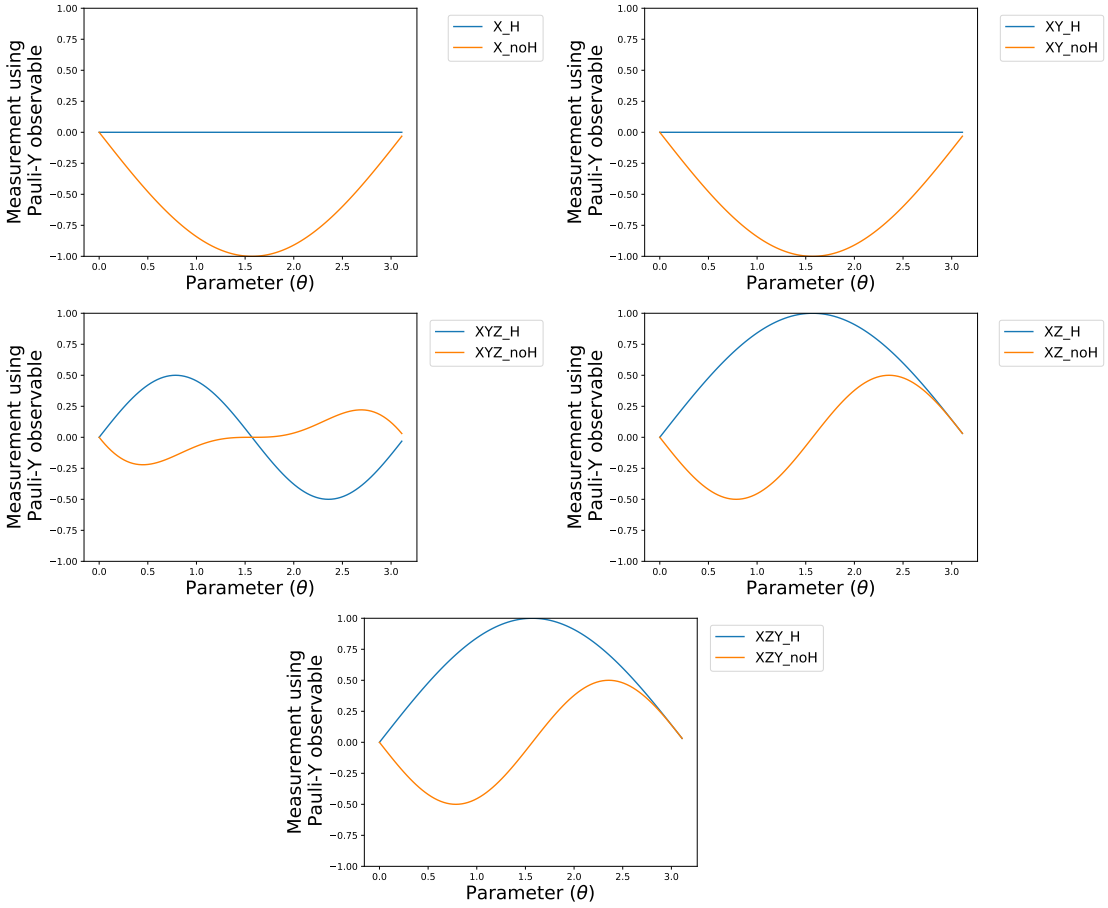


Figure 4.8: Evaluation of encoding strategies that use the rotational gates RX, RX-RY, RX-RY-RZ, RX-RZ, RX-RZ-RY measured using the Pauli-Y observable

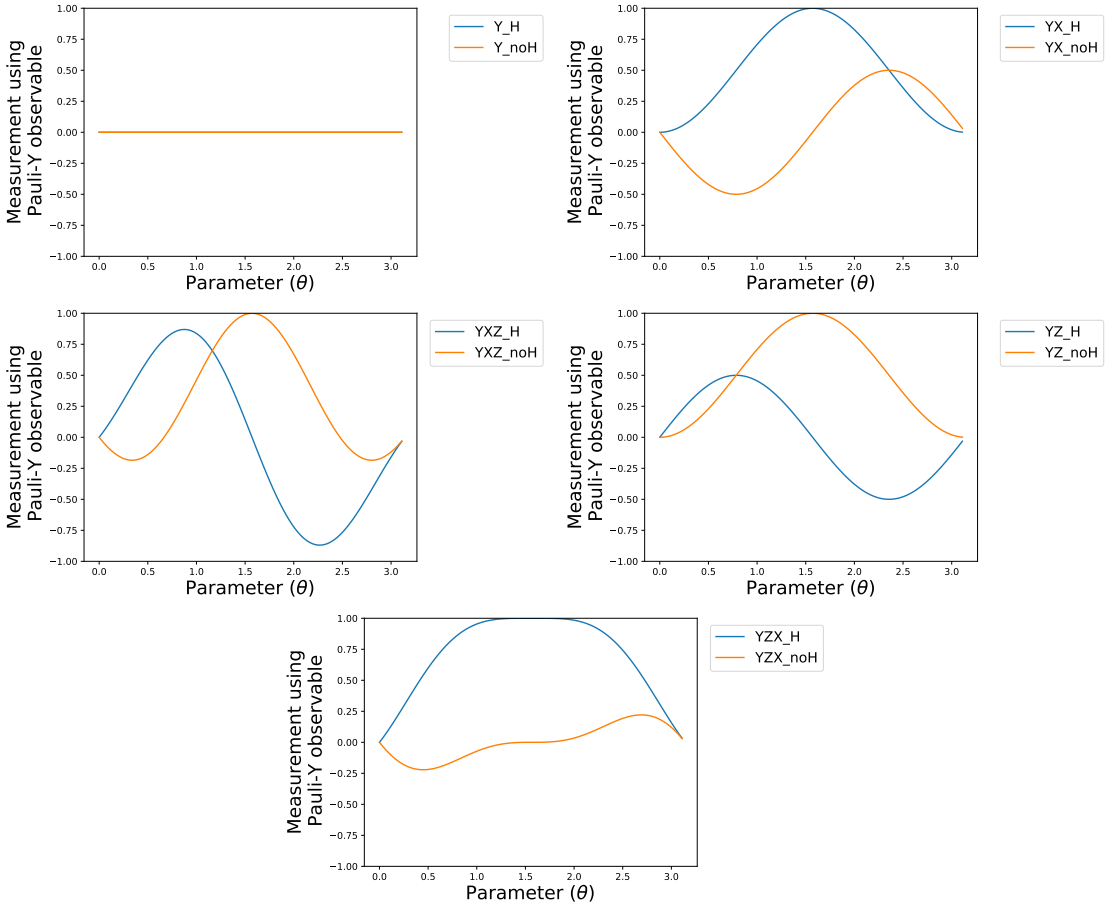


Figure 4.9: Evaluation of encoding strategies that use the rotational gates RY, RY-RX, RY-RX-RZ, RY-RZ, RY-RZ-RX measured using the Pauli-Y observable.

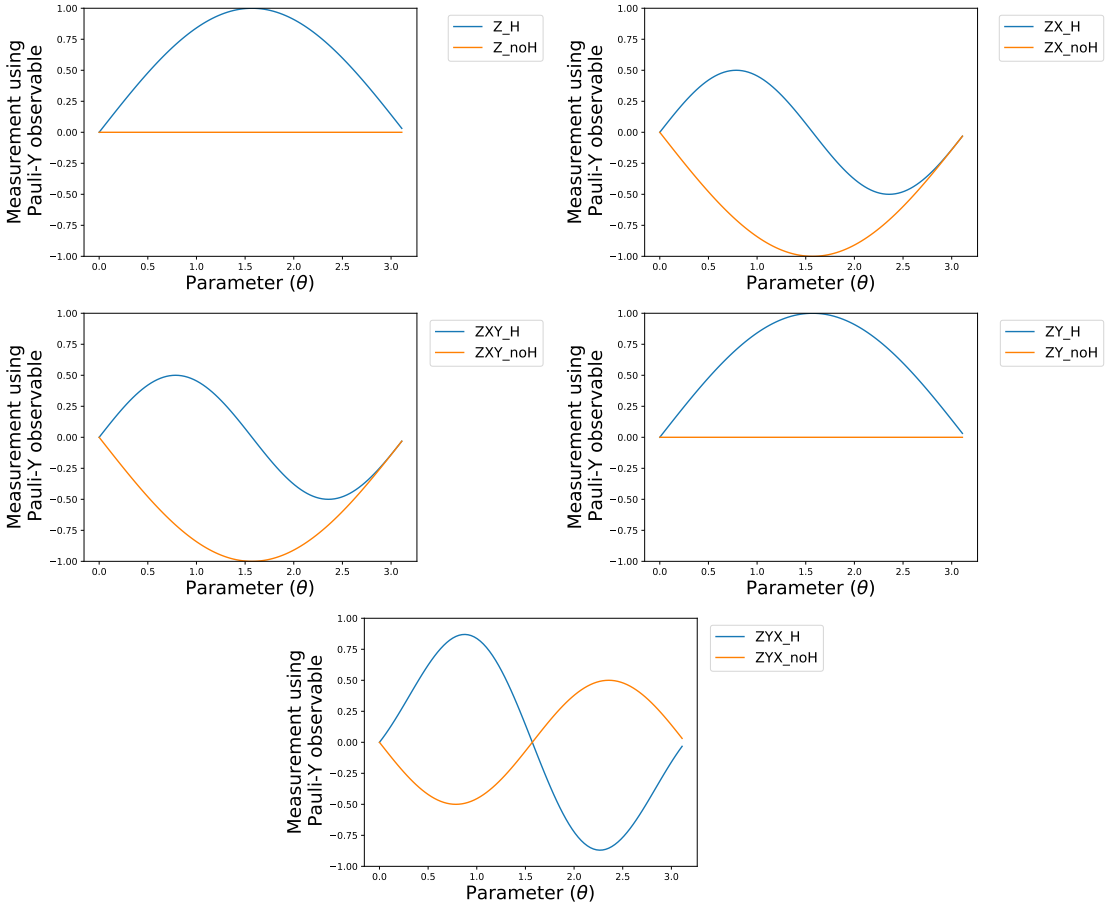


Figure 4.10: Evaluation of encoding strategies that use the rotational gates RZ, RZ-RX, RZ-RX-RY, RZ-RY, RZ-RY-RX measured using the Pauli-Y observable.

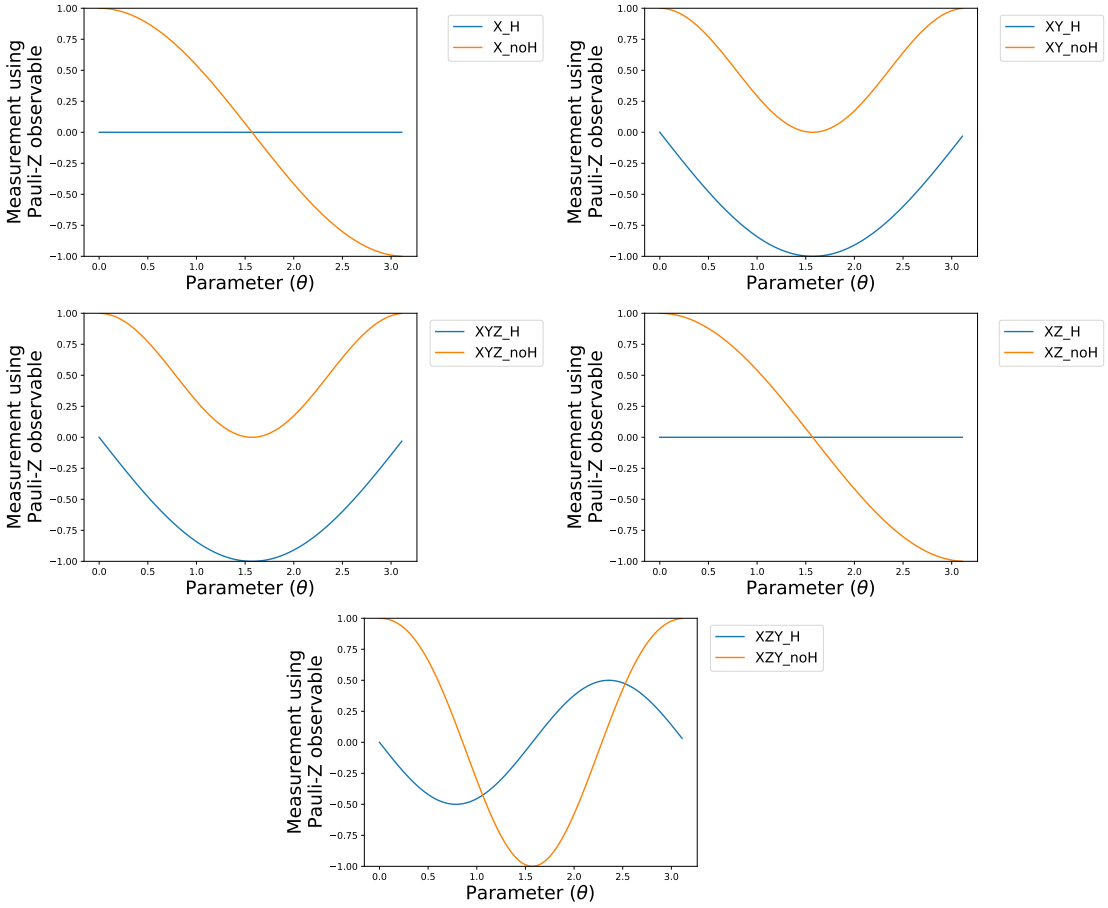


Figure 4.11: Evaluation of encoding strategies that use the rotational gates RX, RX-RY, RX-RY-RZ, RX-RZ, RX-RZ-RY measured using the Pauli-Z observable.

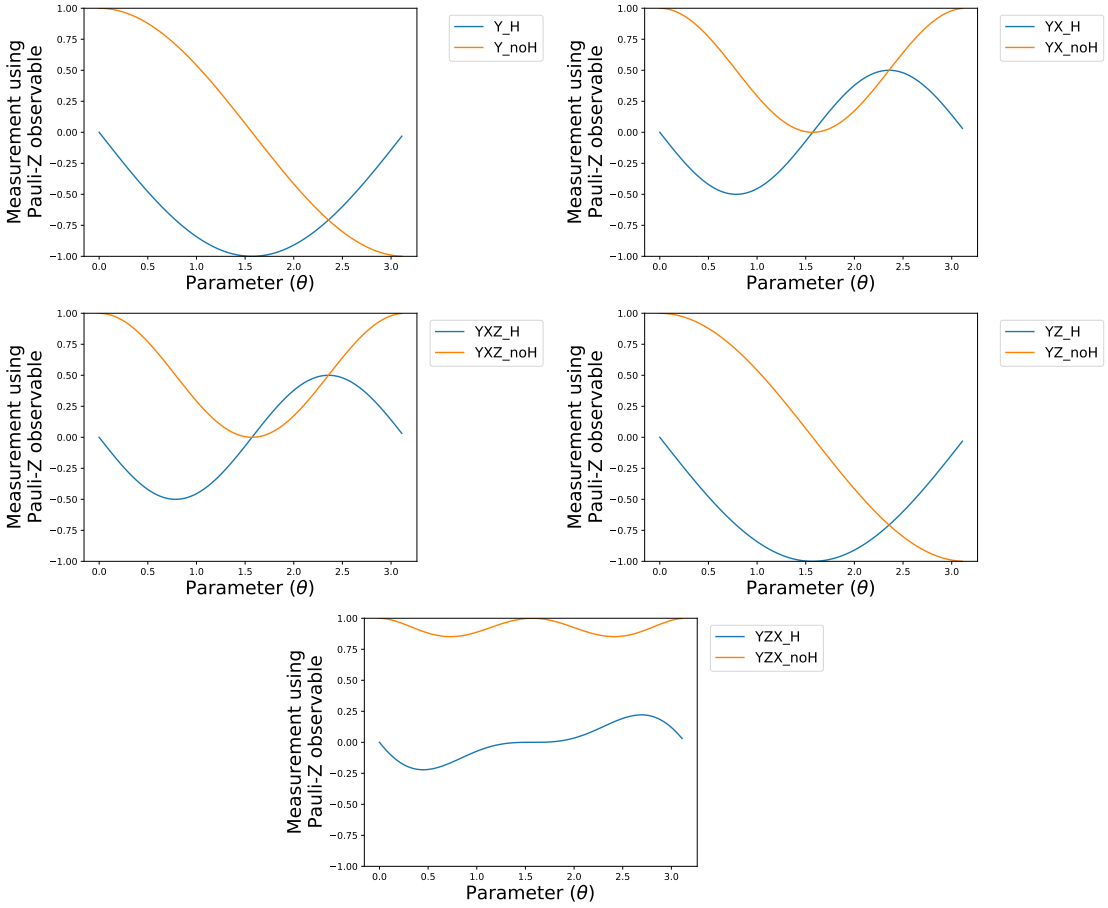


Figure 4.12: Evaluation of encoding strategies that use the rotational gates RY, RY-RX, RY-RX-RZ, RY-RZ, RY-RZ-RX measured using the Pauli-Z observable.

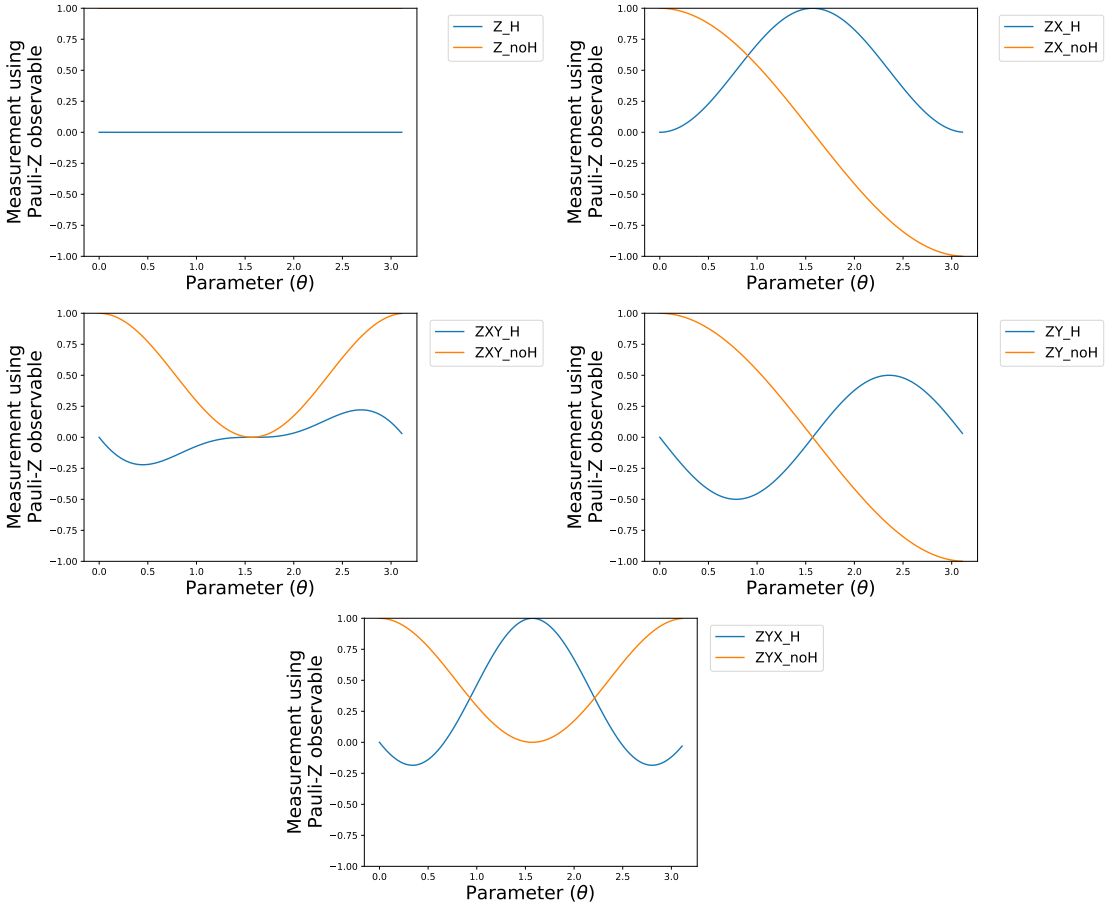


Figure 4.13: Evaluation of encoding strategies that use the rotational gates RZ, RZ-RX, RZ-RX-RY, RZ-RY, RZ-RY-RX measured using the Pauli-Z observable.

Amplitude encoding

Amplitude encoding involves mapping the classical value into one of the probability amplitudes of the state vector. The Mottonen State Preparation[12], also developed in the Pennylane library[9], can be used to prepare a system in a specific state.

Mottonen state preparation Mottonen state preparation use C-NOT and Rotational gates, particularly the RY gate, to prepare the state to the desired one. If all the probability amplitudes of the desired state are positive, the resulting state at the output is equal to the one desired and it is obtained using just RY and CNOT gates. Otherwise, the state obtained will have an additional phase variation and it would be necessary to use also RZ gates, increasing the number of qubit transformations and slowing the model. For this reason, in this thesis, the value of the initial features are scaled in the interval [0, 1].
The feature vector must then be transformed again to be embedded, as it is necessary to ensure that its norm equals 1. Therefore, each feature is divided by the norm-2 of the vector.
After these transformations, it is required to calculate the parameters angles associated with the RY gates as:

$$\beta_j^s = 2 \arcsin \frac{\sqrt{\sum_{l=1}^{2^{s-1}} \left| a_{(2j-1)2^{s-1}+l} \right|^2}}{\sqrt{\sum_{l=1}^{2^s} \left| a_{(j-1)2^s+l} \right|^2}} \tag{4.5}$$

where

- β_j^s is the rotation angle for the RY gate of the jth qubit
- s is the control state qubit
- a_j is the probability amplitude desired

As an example, supposing to encode the state $|\psi\rangle = \sqrt{0.2}\,|000\rangle + \sqrt{0.5}\,|010\rangle + \sqrt{0.2}\,|110\rangle + \sqrt{0.1}\,|111\rangle$, the resulting angles are reported in Tab 4.2.

$\beta_1^1 = 0$	$\beta_2^1 = 0$
$\beta_3^1 = 0$	$\beta_4^1 = 1.231$
$\beta_1^2 = 2.014$	$\beta_2^2 = 3.142$
$\beta_1^3 = 1.159$	

Table 4.2: Resulting angle parameters associated with the RY gates in the Mottonen state preparation to obtain the state $|\psi\rangle = \sqrt{0.2}\,|000\rangle + \sqrt{0.5}\,|010\rangle + \sqrt{0.2}\,|110\rangle + \sqrt{0.1}\,|111\rangle$

The graphical representation of the quantum circuit which can be derived is presented in figure 4.14. The state obtained by applying this circuit is coherent with the expected values of probability amplitudes, as shown from the results reported in Fig 4.15.

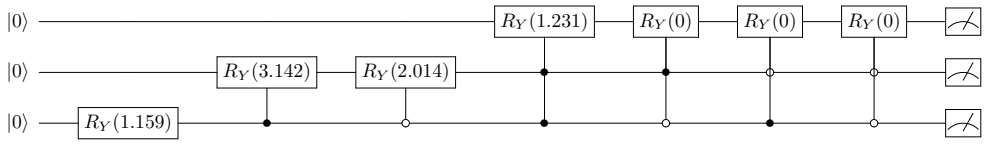


Figure 4.14: Representation of the Mottonen's state preparation circuit capable to get the state $|\psi\rangle = \sqrt{0.2}\,|000\rangle + \sqrt{0.5}\,|010\rangle + \sqrt{0.2}\,|110\rangle + \sqrt{0.1}\,|111\rangle$ starting from the ground state.

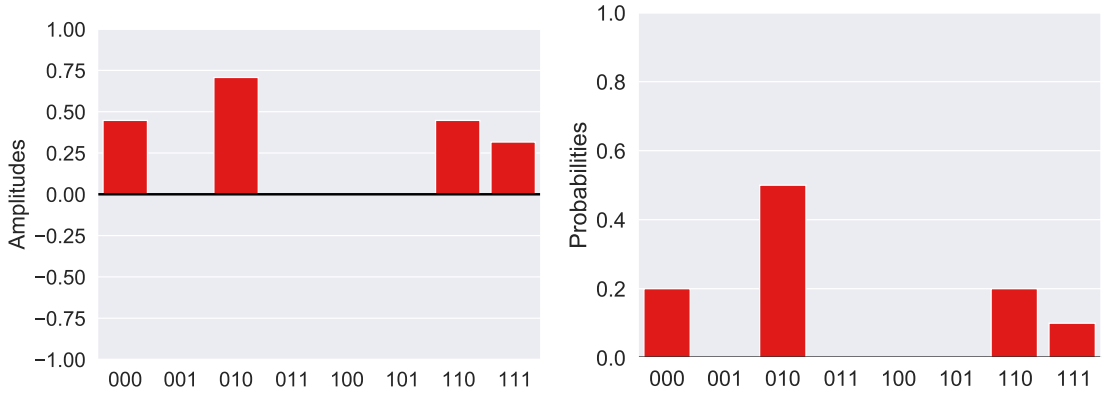
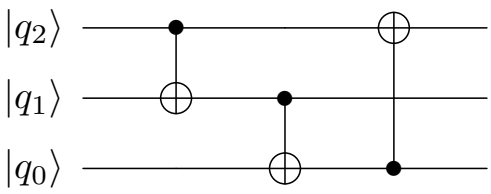


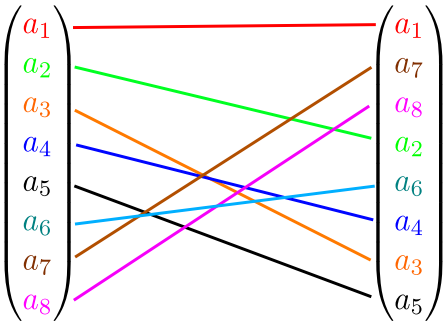
Figure 4.15: Mottonen state preparation encoding results

Entangling block used for data encoding

For both the encoding strategies, Angle and Amplitude, an entangling block after the rotational gates can be added. This strategy adds another degree of complexity to the model, obtaining a higher correlation of the features, and, in theory, could be helpful for complex classifications.
Figure 4.16a shows an example of an entangling circuit obtained by applying the C-NOT on each of the qubits of the quantum system in a round configuration. The effect of this block is a shuffle of the probability amplitudes of the state vector. To better describe the outcome of the circuit, Figure 4.16b represents how the state vector changes by adding this block.



(a) Entangling block circuit defined in a 3-qubits system. The C-NOT gates are applied on each of the qubits in a round configuration. The top qubit represents the most significant one.



(b) Transformation on the probability amplitudes of the state vector applying the entangling block.

4.2.2 Analysis of the variational block

The variational block is composed of the Rotational gate $Rot(\phi, \theta, \omega)$, which results equal to the composition of $RZ(\phi)RY(\theta)RZ(\omega)$, and an entangling block defined by the C-NOT placed in a round configuration. During the training step, the angles associated with the Rotational gates are optimized to achieve better accuracy for the classification task.
The circuit, described by the StronglyEntanglingLayer class by Pennylane[9], is presented in Fig 4.17.

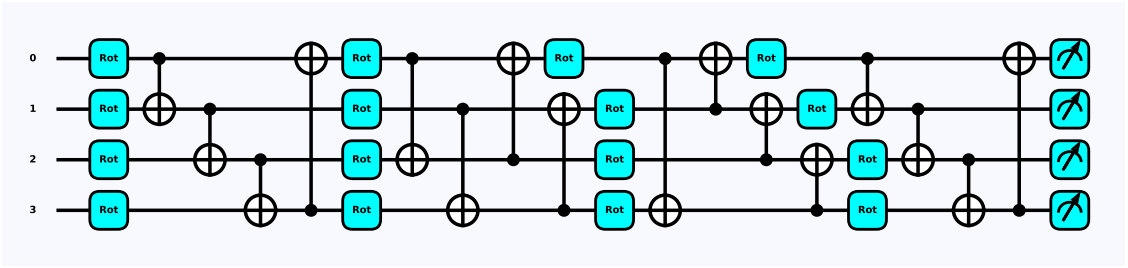


Figure 4.17: A representation of the variational block for a system with four qubits and four-parameter layers, described using the PennyLane library. The rotational gate Rot is equal to the application of the RZ, RY, and RZ gates.

4.2.3 Re-uploading tecnique

The re-uploading technique is a strategy that consists of encoding the data every time a new parameter block is defined.
Applying this strategy, the model adds another degree of complexity, allowing better performance for hard-classification tasks. The circuit obtained is shown in Fig. 4.18.

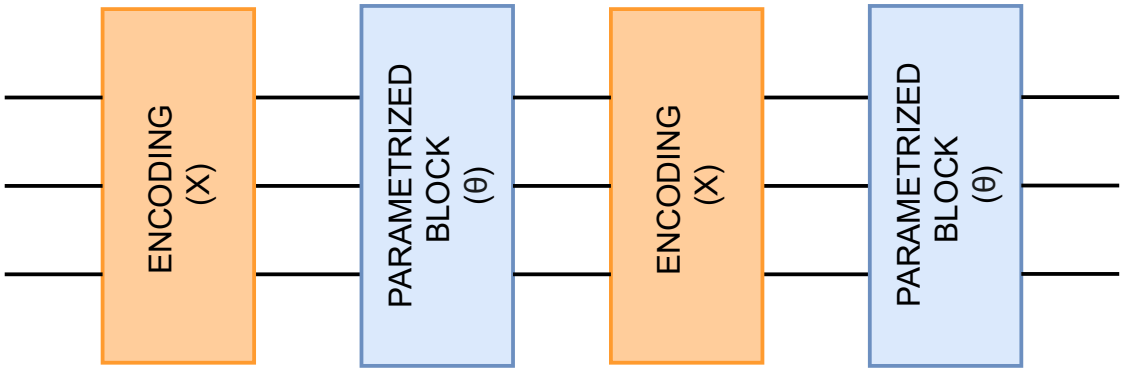


Figure 4.18: A graphical representation of the reuploading technique, which consists of encoding the data every time a new parameter block is added.

4.2.4 Figure of Merits for the evaluation of the Variational Quantum Circuit model

Referring to Section 3.3, the most important Figure of Merits related to the Variational Quantum Circuit can be analyzed.

Depth

Figure 4.21 shows how to calculate the depth of a variational quantum circuit. This metric is influenced by the techniques applied, the data encoding strategies and the number of parameter layers. For the techniques that use the Angle encoding strategies, the depth results are summed in tables 4.3 to 4.6.

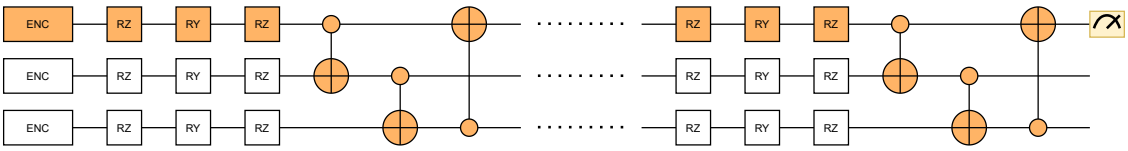


Figure 4.19: The figure shows how the depth is calculated for a Variational Quantum Circuit defined with three qubits and without applying the entangling block and the re-uploading technique. The highlighted path represents the longest one.

Table 4.3: Report of the depths calculated for models that use the Angle encoding strategy in which the entangling block and the re-uploading technique are not applied. N stands for the number of qubits of the system.

Encoding	L1	L2	L4	L8
X	$4+1\times N$	$7+2\times N$	$13+4\times N$	$25+8\times N$
X_H	$5+1\times N$	$8+2\times N$	$14+4\times N$	$26+8\times N$
XY	$5+1\times N$	$8+2\times N$	$14+4\times N$	$26+8\times N$
XY_H	$6+1\times N$	$9+2\times N$	$15+4\times N$	$27+8\times N$
XYZ	$6+1\times N$	$9+2\times N$	$15+4\times N$	$27+8\times N$
XYZ_H	$7+1\times N$	$10+2\times N$	$16+4\times N$	$28+8\times N$
XZ	$5+1\times N$	$8+2\times N$	$14+4\times N$	$26+8\times N$
XZ_H	$6+1\times N$	$9+2\times N$	$15+4\times N$	$27+8\times N$
XZY	$6+1\times N$	$9+2\times N$	$15+4\times N$	$27+8\times N$
XZY_H	$7+1\times N$	$10+2\times N$	$16+4\times N$	$28+8\times N$
Y	$4+1\times N$	$7+2\times N$	$13+4\times N$	$25+8\times N$
Y_H	$5+1\times N$	$8+2\times N$	$14+4\times N$	$26+8\times N$
YX	$5+1\times N$	$8+2\times N$	$14+4\times N$	$26+8\times N$
YX_H	$6+1\times N$	$9+2\times N$	$15+4\times N$	$27+8\times N$
YXZ	$6+1\times N$	$9+2\times N$	$15+4\times N$	$27+8\times N$
YXZ_H	$7+1\times N$	$10+2\times N$	$16+4\times N$	$28+8\times N$
YZ	$5+1\times N$	$8+2\times N$	$14+4\times N$	$26+8\times N$
YZ_H	$6+1\times N$	$9+2\times N$	$15+4\times N$	$27+8\times N$
YZX	$6+1\times N$	$9+2\times N$	$15+4\times N$	$27+8\times N$
YZX_H	$7+1\times N$	$10+2\times N$	$16+4\times N$	$28+8\times N$
Z	$4+1\times N$	$7+2\times N$	$13+4\times N$	$25+8\times N$
Z_H	$5+1\times N$	$8+2\times N$	$14+4\times N$	$26+8\times N$
ZX	$5+1\times N$	$8+2\times N$	$14+4\times N$	$26+8\times N$
ZX_H	$6+1\times N$	$9+2\times N$	$15+4\times N$	$27+8\times N$
ZXY	$6+1\times N$	$9+2\times N$	$15+4\times N$	$27+8\times N$
ZXY_H	$7+1\times N$	$10+2\times N$	$16+4\times N$	$28+8\times N$
ZY	$5+1\times N$	$8+2\times N$	$14+4\times N$	$26+8\times N$
ZY_H	$6+1\times N$	$9+2\times N$	$15+4\times N$	$27+8\times N$
ZYX	$6+1\times N$	$9+2\times N$	$15+4\times N$	$27+8\times N$
ZYX_H	$7+1\times N$	$10+2\times N$	$16+4\times N$	$28+8\times N$

Table 4.4: Report of the depths calculated for models that use the Angle encoding strategy in which the entangling block is applied but not the re-uploading technique. N stands for the number of qubits of the system.

Encoding	L1	L2	L4	L8
X	$4+2\times N$	$7+3\times N$	$13+5\times N$	$25+9\times N$
X_H	$5+2\times N$	$8+3\times N$	$14+5\times N$	$26+9\times N$
XY	$5+2\times N$	$8+3\times N$	$14+5\times N$	$26+9\times N$
XY_H	$6+2\times N$	$9+3\times N$	$15+5\times N$	$27+9\times N$
XYZ	$6+2\times N$	$9+3\times N$	$15+5\times N$	$27+9\times N$
XYZ_H	$7+2\times N$	$10+3\times N$	$16+5\times N$	$28+9\times N$
XZ	$5+2\times N$	$8+3\times N$	$14+5\times N$	$26+9\times N$
XZ_H	$6+2\times N$	$9+3\times N$	$15+5\times N$	$27+9\times N$
XZY	$6+2\times N$	$9+3\times N$	$15+5\times N$	$27+9\times N$
XZY_H	$7+2\times N$	$10+3\times N$	$16+5\times N$	$28+9\times N$
Y	$4+2\times N$	$7+3\times N$	$13+5\times N$	$25+9\times N$
Y_H	$5+2\times N$	$8+3\times N$	$14+5\times N$	$26+9\times N$
YX	$5+2\times N$	$8+3\times N$	$14+5\times N$	$26+9\times N$
YX_H	$6+2\times N$	$9+3\times N$	$15+5\times N$	$27+9\times N$
YXZ	$6+2\times N$	$9+3\times N$	$15+5\times N$	$27+9\times N$
YXZ_H	$7+2\times N$	$10+3\times N$	$16+5\times N$	$28+9\times N$
YZ	$5+2\times N$	$8+3\times N$	$14+5\times N$	$26+9\times N$
YZ_H	$6+2\times N$	$9+3\times N$	$15+5\times N$	$27+9\times N$
YZX	$6+2\times N$	$9+3\times N$	$15+5\times N$	$27+9\times N$
YZX_H	$7+2\times N$	$10+3\times N$	$16+5\times N$	$28+9\times N$
Z	$4+2\times N$	$7+3\times N$	$13+5\times N$	$25+9\times N$
Z_H	$5+2\times N$	$8+3\times N$	$14+5\times N$	$26+9\times N$
ZX	$5+2\times N$	$8+3\times N$	$14+5\times N$	$26+9\times N$
ZX_H	$6+2\times N$	$9+3\times N$	$15+5\times N$	$27+9\times N$
ZXY	$6+2\times N$	$9+3\times N$	$15+5\times N$	$27+9\times N$
ZXY_H	$7+2\times N$	$10+3\times N$	$16+5\times N$	$28+9\times N$
ZY	$5+2\times N$	$8+3\times N$	$14+5\times N$	$26+9\times N$
ZY_H	$6+2\times N$	$9+3\times N$	$15+5\times N$	$27+9\times N$
ZYX	$6+2\times N$	$9+3\times N$	$15+5\times N$	$27+9\times N$
ZYX_H	$7+2\times N$	$10+3\times N$	$16+5\times N$	$28+9\times N$

Table 4.5: Report of the depths calculated for models that use the Angle encoding strategy in which the re-uploading technique is applied but not the entangling block. N stands for the number of qubits of the system.

Encoding	L2	L4	L8
X	$8 + 2 \times N$	$16 + 4 \times N$	$32 + 8 \times N$
X_H	$10 + 2 \times N$	$20 + 4 \times N$	$40 + 8 \times N$
XY	$10 + 2 \times N$	$20 + 4 \times N$	$40 + 8 \times N$
XY_H	$12 + 2 \times N$	$24 + 4 \times N$	$48 + 8 \times N$
XYZ	$12 + 2 \times N$	$24 + 4 \times N$	$48 + 8 \times N$
XYZ_H	$14 + 2 \times N$	$28 + 4 \times N$	$56 + 8 \times N$
XZ	$10 + 2 \times N$	$20 + 4 \times N$	$40 + 8 \times N$
XZ_H	$12 + 2 \times N$	$24 + 4 \times N$	$48 + 8 \times N$
XZY	$12 + 2 \times N$	$24 + 4 \times N$	$48 + 8 \times N$
XZY_H	$14 + 2 \times N$	$28 + 4 \times N$	$56 + 8 \times N$
Y	$8 + 1 \times N$	$16 + 4 \times N$	$32 + 8 \times N$
Y_H	$10 + 2 \times N$	$20 + 4 \times N$	$40 + 8 \times N$
YX	$10 + 2 \times N$	$20 + 4 \times N$	$40 + 8 \times N$
YX_H	$12 + 2 \times N$	$24 + 4 \times N$	$48 + 8 \times N$
YXZ	$12 + 2 \times N$	$24 + 4 \times N$	$48 + 8 \times N$
YXZ_H	$14 + 2 \times N$	$28 + 4 \times N$	$56 + 8 \times N$
YZ	$10 + 2 \times N$	$20 + 4 \times N$	$40 + 8 \times N$
YZ_H	$12 + 2 \times N$	$24 + 4 \times N$	$48 + 8 \times N$
YZX	$12 + 2 \times N$	$24 + 4 \times N$	$48 + 8 \times N$
YZX_H	$14 + 2 \times N$	$28 + 4 \times N$	$56 + 8 \times N$
Z	$8 + 1 \times N$	$16 + 4 \times N$	$32 + 8 \times N$
Z_H	$10 + 2 \times N$	$20 + 4 \times N$	$40 + 8 \times N$
ZX	$10 + 2 \times N$	$20 + 4 \times N$	$40 + 8 \times N$
ZX_H	$12 + 2 \times N$	$24 + 4 \times N$	$48 + 8 \times N$
ZXY	$12 + 2 \times N$	$24 + 4 \times N$	$48 + 8 \times N$
ZXY_H	$14 + 2 \times N$	$28 + 4 \times N$	$56 + 8 \times N$
ZY	$10 + 2 \times N$	$20 + 4 \times N$	$40 + 8 \times N$
ZY_H	$12 + 2 \times N$	$24 + 4 \times N$	$48 + 8 \times N$
ZYX	$12 + 2 \times N$	$24 + 4 \times N$	$48 + 8 \times N$
ZYX_H	$14 + 2 \times N$	$28 + 4 \times N$	$56 + 8 \times N$

Table 4.6: Report of the depths calculated for models that use the Angle encoding strategy where both the re-uploading technique and the entangling block are applied. N stands for the number of qubits of the system.

Encoding	L2	L4	L8
X	$8 + 4 \times N$	$16 + 8 \times N$	$32 + 16 \times N$
X_H	$10 + 4 \times N$	$20 + 8 \times N$	$40 + 16 \times N$
XY	$10 + 4 \times N$	$20 + 8 \times N$	$40 + 16 \times N$
XY_H	$12 + 4 \times N$	$24 + 8 \times N$	$48 + 16 \times N$
XYZ	$12 + 4 \times N$	$24 + 8 \times N$	$48 + 16 \times N$
XYZ_H	$14 + 4 \times N$	$28 + 8 \times N$	$56 + 16 \times N$
XZ	$10 + 4 \times N$	$20 + 8 \times N$	$40 + 16 \times N$
XZ_H	$12 + 4 \times N$	$24 + 8 \times N$	$48 + 16 \times N$
XZY	$12 + 4 \times N$	$24 + 8 \times N$	$48 + 16 \times N$
XZY_H	$14 + 4 \times N$	$28 + 8 \times N$	$56 + 16 \times N$
Y	$8 + 4 \times N$	$16 + 8 \times N$	$32 + 16 \times N$
Y_H	$10 + 4 \times N$	$20 + 8 \times N$	$40 + 16 \times N$
YX	$10 + 4 \times N$	$20 + 8 \times N$	$40 + 16 \times N$
YX_H	$12 + 4 \times N$	$24 + 8 \times N$	$48 + 16 \times N$
YXZ	$12 + 4 \times N$	$24 + 8 \times N$	$48 + 16 \times N$
YXZ_H	$14 + 4 \times N$	$28 + 8 \times N$	$56 + 16 \times N$
YZ	$10 + 4 \times N$	$20 + 8 \times N$	$40 + 16 \times N$
YZ_H	$12 + 4 \times N$	$24 + 8 \times N$	$48 + 16 \times N$
YZX	$12 + 4 \times N$	$24 + 8 \times N$	$48 + 16 \times N$
YZX_H	$14 + 4 \times N$	$28 + 8 \times N$	$56 + 16 \times N$
Z	$8 + 4 \times N$	$16 + 8 \times N$	$32 + 16 \times N$
Z_H	$10 + 4 \times N$	$20 + 8 \times N$	$40 + 16 \times N$
ZX	$10 + 4 \times N$	$20 + 8 \times N$	$40 + 16 \times N$
ZX_H	$12 + 4 \times N$	$24 + 8 \times N$	$48 + 16 \times N$
ZXY	$12 + 4 \times N$	$24 + 8 \times N$	$48 + 16 \times N$
ZXY_H	$14 + 4 \times N$	$28 + 8 \times N$	$56 + 16 \times N$
ZY	$10 + 4 \times N$	$20 + 8 \times N$	$40 + 16 \times N$
ZY_H	$12 + 4 \times N$	$24 + 8 \times N$	$48 + 16 \times N$
ZYX	$12 + 4 \times N$	$24 + 8 \times N$	$48 + 16 \times N$
ZYX_H	$14 + 4 \times N$	$28 + 8 \times N$	$56 + 16 \times N$

The depth increases linearly with the number of qubits of the system and the number of parameter layers. The strategies that use the Re-uploading and the ones that use more rotational gates have a higher depth.
The same metric is considered for the Amplitude encoding models. However, it is evaluated using a specific PennyLane function for difficulties in generalizing to a generic N-qubit system. The results are reported in Tables 4.7 and 4.8.

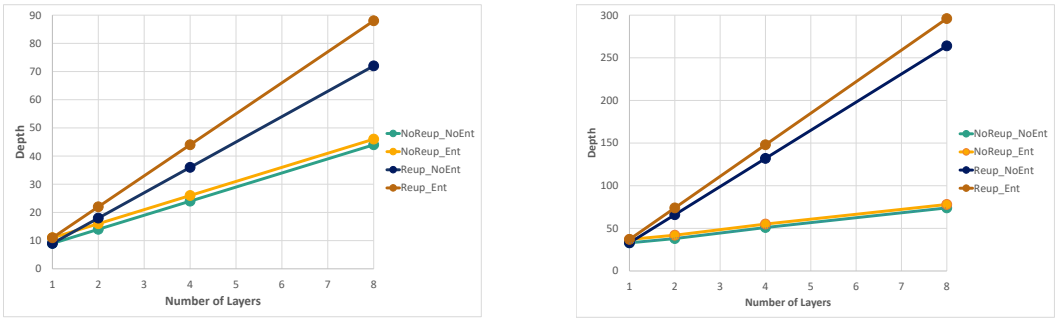
Table 4.7: Report of the depths for models that use the Amplitude encoding strategy for a two-qubit system calculated by using the PennyLane library.

Strategy adopted	L1	L2	L4	L8
NoReup_NoEnt	9	14	24	44
NoReup_Ent	11	16	26	46
Reup_NoEnt	9	18	36	72
Reup_Ent	11	22	44	88

Table 4.8: Report of the depths for models that use the Amplitude encoding strategy for a four-qubit system calculated using the PennyLane library.

Strategy adopted	**L1**	**L2**	**L4**	**L8**
NoReup_NoEnt	33	38	51	74
NoReup_Ent	37	42	55	78
Reup_NoEnt	33	66	132	264
Reup_Ent	37	74	148	296

To better analyze the linear behavior of the depth with the number of parameter layers, the plots in Fig. 4.20 are reported that show the depth results for a two-qubit system and a four-qubit system for each strategy adopted.



(a) (b)

Figure 4.20: Plot of the variation of the depth with respect to the number of layers for a two-qubit system, in Figure 4.20a, and a four-qubit model, in Figure 4.20b. The strategies evaluated consider the presence of the entangling block and the application of the re-uploading technique.

Complexity

For the analysis of the complexity, the contributions of the gates are categorized into Rotational gates, decomposing also the Rot gate of the variational part into two RZ gates and an RY gate, Hadamard, and C-NOT. Figure 4.21 shows an example of how to compute this metric.

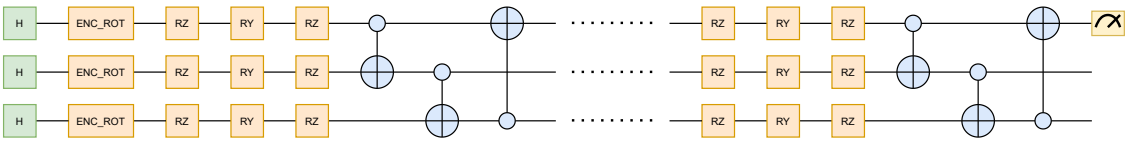


Figure 4.21: The figure shows how the complexity metric can be calculated for a Variational Quantum Circuit defined with three qubits and without applying the entangling block and the re-uploading technique. The green block represents the contributions of the Hadamard, the yellow rotational gates, and the light blue one the CNOT.

The Angle encoding strategies complexities are reported in Tables tables 4.9 to 4.12.

Table 4.9: Complexity for models in which the **entangling block and the re-uploading technique are not applied**. Each column stands for the number of layers for the model analyzed, while the results of the same row share the encoding strategy.

Encoding	L1	L2	L4	L8
X	$(4R+1CX)\times N$	$(7R+2CX)\times N$	$(13R+4CX)\times N$	$(25R+8CX)\times N$
X_H	$(1H+4R+1CX)\times N$	$(1H+7R+2CX)\times N$	$(1H+13R+4CX)\times N$	$(1H+25R+8CX)\times N$
XY	$(5R+1CX)\times N$	$(8R+2CX)\times N$	$(14R+4CX)\times N$	$(26R+8CX)\times N$
XY_H	$(1H+5R+1CX)\times N$	$(1H+8R+2CX)\times N$	$(1H+14R+4CX)\times N$	$(1H+26R+8CX)\times N$
XYZ	$(6R+1CX)\times N$	$(9R+2CX)\times N$	$(15R+4CX)\times N$	$(27R+8CX)\times N$
XYZ_H	$(1H+6R+1CX)\times N$	$(1H+9R+2CX)\times N$	$(1H+15R+4CX)\times N$	$(1H+27R+8CX)\times N$
XZ	$(5R+1CX)\times N$	$(8R+2CX)\times N$	$(14R+4CX)\times N$	$(26R+8CX)\times N$
XZ_H	$(1H+5R+1CX)\times N$	$(1H+8R+2CX)\times N$	$(1H+14R+4CX)\times N$	$(1H+26R+8CX)\times N$
XZY	$(6R+1CX)\times N$	$(9R+2CX)\times N$	$(15R+4CX)\times N$	$(27R+8CX)\times N$
XZY_H	$(1H+6R+1CX)\times N$	$(1H+9R+2CX)\times N$	$(1H+15R+4CX)\times N$	$(1H+27R+8CX)\times N$
Y	$(4R+1CX)\times N$	$(7R+2CX)\times N$	$(13R+4CX)\times N$	$(25R+8CX)\times N$
Y_H	$(1H+4R+1CX)\times N$	$(1H+7R+2CX)\times N$	$(1H+13R+4CX)\times N$	$(1H+25R+8CX)\times N$
YX	$(5R+1CX)\times N$	$(8R+2CX)\times N$	$(14R+4CX)\times N$	$(26R+8CX)\times N$
YX_H	$(1H+5R+1CX)\times N$	$(1H+8R+2CX)\times N$	$(1H+14R+4CX)\times N$	$(1H+26R+8CX)\times N$
YXZ	$(6R+1CX)\times N$	$(9R+2CX)\times N$	$(15R+4CX)\times N$	$(27R+8CX)\times N$
YXZ_H	$(1H+6R+1CX)\times N$	$(1H+9R+2CX)\times N$	$(1H+15R+4CX)\times N$	$(1H+27R+8CX)\times N$
YZ	$(5R+1CX)\times N$	$(8R+2CX)\times N$	$(14R+4CX)\times N$	$(26R+8CX)\times N$
YZ_H	$(1H+5R+1CX)\times N$	$(1H+8R+2CX)\times N$	$(1H+14R+4CX)\times N$	$(1H+26R+8CX)\times N$
YZX	$(6R+1CX)\times N$	$(9R+2CX)\times N$	$(15R+4CX)\times N$	$(27R+8CX)\times N$
YZX_H	$(1H+6R+1CX)\times N$	$(1H+9R+2CX)\times N$	$(1H+15R+4CX)\times N$	$(1H+27R+8CX)\times N$
Z	$(4R+1CX)\times N$	$(7R+2CX)\times N$	$(13R+4CX)\times N$	$(25R+8CX)\times N$
Z_H	$(1H+4R+1CX)\times N$	$(1H+7R+2CX)\times N$	$(1H+13R+4CX)\times N$	$(1H+25R+8CX)\times N$
ZX	$(5R+1CX)\times N$	$(8R+2CX)\times N$	$(14R+4CX)\times N$	$(26R+8CX)\times N$
ZX_H	$(1H+5R+1CX)\times N$	$(1H+8R+2CX)\times N$	$(1H+14R+4CX)\times N$	$(1H+26R+8CX)\times N$
ZXY	$(6R+1CX)\times N$	$(9R+2CX)\times N$	$(15R+4CX)\times N$	$(27R+8CX)\times N$
ZXY_H	$(1H+6R+1CX)\times N$	$(1H+9R+2CX)\times N$	$(1H+15R+4CX)\times N$	$(1H+27R+8CX)\times N$
ZY	$(5R+1CX)\times N$	$(8R+2CX)\times N$	$(14R+4CX)\times N$	$(26R+8CX)\times N$
ZY_H	$(1H+5R+1CX)\times N$	$(1H+8R+2CX)\times N$	$(1H+14R+4CX)\times N$	$(1H+26R+8CX)\times N$
ZYX	$(6R+1CX)\times N$	$(9R+2CX)\times N$	$(15R+4CX)\times N$	$(27R+8CX)\times N$
ZYX_H	$(1H+6R+1CX)\times N$	$(1H+9R+2CX)\times N$	$(1H+15R+4CX)\times N$	$(1H+27R+8CX)\times N$

Table 4.10: Complexity for models in which **the re-uploading technique is applied but not the entangling block**.

Encoding	L2	L4	L8
X	$(8R + 2CX) \times N$	$(16R + 4CX) \times N$	$(32R + 8CX) \times N$
X_H	$(2H + 8R + 2CX) \times N$	$(4H + 16R + 4CX) \times N$	$(8H + 32R + 8CX) \times N$
XY	$(10R + 2CX) \times N$	$(20R + 4CX) \times N$	$(40R + 8CX) \times N$
XY_H	$(2H + 10R + 2CX) \times N$	$(4H + 20R + 4CX) \times N$	$(8H + 40R + 8CX) \times N$
XYZ	$(12R + 2CX) \times N$	$(24R + 4CX) \times N$	$(48R + 8CX) \times N$
XYZ_H	$(2H + 12R + 2CX) \times N$	$(4H + 24R + 4CX) \times N$	$(8H + 48R + 8CX) \times N$
XZ	$(10R + 2CX) \times N$	$(20R + 4CX) \times N$	$(40R + 8CX) \times N$
XZ_H	$(2H + 10R + 2CX) \times N$	$(4H + 20R + 4CX) \times N$	$(8H + 40R + 8CX) \times N$
XZY	$(12R + 2CX) \times N$	$(24R + 4CX) \times N$	$(48R + 8CX) \times N$
XZY_H	$(2H + 12R + 2CX) \times N$	$(4H + 24R + 4CX) \times N$	$(8H + 48R + 8CX) \times N$
Y	$(8R + 2CX) \times N$	$(16R + 4CX) \times N$	$(32R + 8CX) \times N$
Y_H	$(2H + 8R + 2CX) \times N$	$(4H + 16R + 4CX) \times N$	$(8H + 32R + 8CX) \times N$
YX	$(10R + 2CX) \times N$	$(20R + 4CX) \times N$	$(40R + 8CX) \times N$
YX_H	$(2H + 10R + 2CX) \times N$	$(4H + 20R + 4CX) \times N$	$(8H + 40R + 8CX) \times N$
YXZ	$(12R + 2CX) \times N$	$(24R + 4CX) \times N$	$(48R + 8CX) \times N$
YXZ_H	$(2H + 12R + 2CX) \times N$	$(4H + 24R + 4CX) \times N$	$(8H + 48R + 8CX) \times N$
YZ	$(10R + 2CX) \times N$	$(20R + 4CX) \times N$	$(40R + 8CX) \times N$
YZ_H	$(2H + 10R + 2CX) \times N$	$(4H + 20R + 4CX) \times N$	$(8H + 40R + 8CX) \times N$
YZX	$(12R + 2CX) \times N$	$(24R + 4CX) \times N$	$(48R + 8CX) \times N$
YZX_H	$(2H + 12R + 2CX) \times N$	$(4H + 24R + 4CX) \times N$	$(8H + 48R + 8CX) \times N$
Z	$(8R + 2CX) \times N$	$(16R + 4CX) \times N$	$(32R + 8CX) \times N$
Z_H	$(2H + 8R + 2CX) \times N$	$(4H + 16R + 4CX) \times N$	$(8H + 32R + 8CX) \times N$
ZX	$(10R + 2CX) \times N$	$(20R + 4CX) \times N$	$(40R + 8CX) \times N$
ZX_H	$(2H + 10R + 2CX) \times N$	$(4H + 20R + 4CX) \times N$	$(8H + 40R + 8CX) \times N$
ZXY	$(12R + 2CX) \times N$	$(24R + 4CX) \times N$	$(48R + 8CX) \times N$
ZXY_H	$(2H + 12R + 2CX) \times N$	$(4H + 24R + 4CX) \times N$	$(8H + 48R + 8CX) \times N$
ZY	$(10R + 2CX) \times N$	$(20R + 4CX) \times N$	$(40R + 8CX) \times N$
ZY_H	$(2H + 10R + 2CX) \times N$	$(4H + 20R + 4CX) \times N$	$(8H + 40R + 8CX) \times N$
ZYX	$(12R + 2CX) \times N$	$(24R + 4CX) \times N$	$(48R + 8CX) \times N$
ZYX_H	$(2H + 12R + 2CX) \times N$	$(4H + 24R + 4CX) \times N$	$(8H + 48R + 8CX) \times N$

Table 4.11: Complexity for models in which **the entangling block strategy is applied but not the re-uploading technique**.

Encoding	L1	L2	L4	L8
X	$(4R+2CX)\times N$	$(7R+3CX)\times N$	$(13R+5CX)\times N$	$(25R+9CX)\times N$
X_H	$(1H+4R+2CX)\times N$	$(1H+7R+3CX)\times N$	$(1H+13R+5CX)\times N$	$(1H+25R+9CX)\times N$
XY	$(5R+2CX)\times N$	$(8R+3CX)\times N$	$(14R+5CX)\times N$	$(26R+9CX)\times N$
XY_H	$(1H+5R+2CX)\times N$	$(1H+8R+3CX)\times N$	$(1H+14R+5CX)\times N$	$(1H+26R+9CX)\times N$
XYZ	$(6R+2CX)\times N$	$(9R+3CX)\times N$	$(15R+5CX)\times N$	$(27R+9CX)\times N$
XYZ_H	$(1H+6R+2CX)\times N$	$(1H+9R+3CX)\times N$	$(1H+15R+5CX)\times N$	$(1H+27R+9CX)\times N$
XZ	$(5R+2CX)\times N$	$(8R+3CX)\times N$	$(14R+5CX)\times N$	$(26R+9CX)\times N$
XZ_H	$(1H+5R+2CX)\times N$	$(1H+8R+3CX)\times N$	$(1H+14R+5CX)\times N$	$(1H+26R+9CX)\times N$
XZY	$(6R+2CX)\times N$	$(9R+3CX)\times N$	$(15R+5CX)\times N$	$(27R+9CX)\times N$
XZY_H	$(1H+6R+2CX)\times N$	$(1H+9R+3CX)\times N$	$(1H+15R+5CX)\times N$	$(1H+27R+9CX)\times N$
Y	$(4R+2CX)\times N$	$(7R+3CX)\times N$	$(13R+5CX)\times N$	$(25R+9CX)\times N$
Y_H	$(1H+4R+2CX)\times N$	$(1H+7R+3CX)\times N$	$(1H+13R+5CX)\times N$	$(1H+25R+9CX)\times N$
YX	$(5R+2CX)\times N$	$(8R+3CX)\times N$	$(14R+5CX)\times N$	$(26R+9CX)\times N$
YX_H	$(1H+5R+2CX)\times N$	$(1H+8R+3CX)\times N$	$(1H+14R+5CX)\times N$	$(1H+26R+9CX)\times N$
YXZ	$(6R+2CX)\times N$	$(9R+3CX)\times N$	$(15R+5CX)\times N$	$(27R+9CX)\times N$
YXZ_H	$(1H+6R+2CX)\times N$	$(1H+9R+3CX)\times N$	$(1H+15R+5CX)\times N$	$(1H+27R+9CX)\times N$
YZ	$(5R+2CX)\times N$	$(8R+3CX)\times N$	$(14R+5CX)\times N$	$(26R+9CX)\times N$
YZ_H	$(1H+5R+2CX)\times N$	$(1H+8R+3CX)\times N$	$(1H+14R+5CX)\times N$	$(1H+26R+9CX)\times N$
YZX	$(6R+2CX)\times N$	$(9R+3CX)\times N$	$(15R+5CX)\times N$	$(27R+9CX)\times N$
YZX_H	$(1H+6R+2CX)\times N$	$(1H+9R+3CX)\times N$	$(1H+15R+5CX)\times N$	$(1H+27R+9CX)\times N$
Z	$(4R+2CX)\times N$	$(7R+3CX)\times N$	$(13R+5CX)\times N$	$(25R+9CX)\times N$
Z_H	$(1H+4R+2CX)\times N$	$(1H+7R+3CX)\times N$	$(1H+13R+5CX)\times N$	$(1H+25R+9CX)\times N$
ZX	$(5R+2CX)\times N$	$(8R+3CX)\times N$	$(14R+5CX)\times N$	$(26R+9CX)\times N$
ZX_H	$(1H+5R+2CX)\times N$	$(1H+8R+3CX)\times N$	$(1H+14R+5CX)\times N$	$(1H+26R+9CX)\times N$
ZXY	$(6R+2CX)\times N$	$(9R+3CX)\times N$	$(15R+5CX)\times N$	$(27R+9CX)\times N$
ZXY_H	$(1H+6R+2CX)\times N$	$(1H+9R+3CX)\times N$	$(1H+15R+5CX)\times N$	$(1H+27R+9CX)\times N$
ZY	$(5R+2CX)\times N$	$(8R+3CX)\times N$	$(14R+5CX)\times N$	$(26R+9CX)\times N$
ZY_H	$(1H+5R+2CX)\times N$	$(1H+8R+3CX)\times N$	$(1H+14R+5CX)\times N$	$(1H+26R+9CX)\times N$
ZYX	$(6R+2CX)\times N$	$(9R+3CX)\times N$	$(15R+5CX)\times N$	$(27R+9CX)\times N$
ZYX_H	$(1H+6R+2CX)\times N$	$(1H+9R+3CX)\times N$	$(1H+15R+5CX)\times N$	$(1H+27R+9CX)\times N$

Table 4.12: Complexity for models in which both **the re-uploading technique and the entangling block are applied**.

Encoding	L2	L4	L8
X	$(8R + 4CX) \times N$	$(16R + 8CX) \times N$	$(32R + 16CX) \times N$
X_H	$(2H + 8R + 4CX) \times N$	$(4H + 16R + 8CX) \times N$	$(8H + 32R + 16CX) \times N$
XY	$(10R + 4CX) \times N$	$(20R + 8CX) \times N$	$(40R + 16CX) \times N$
XY_H	$(2H + 10R + 4CX) \times N$	$(4H + 20R + 8CX) \times N$	$(8H + 40R + 16CX) \times N$
XYZ	$(12R + 4CX) \times N$	$(24R + 8CX) \times N$	$(48R + 16CX) \times N$
XYZ_H	$(2H + 12R + 4CX) \times N$	$(4H + 24R + 8CX) \times N$	$(8H + 48R + 16CX) \times N$
XZ	$(10R + 4CX) \times N$	$(20R + 8CX) \times N$	$(40R + 16CX) \times N$
XZ_H	$(2H + 10R + 4CX) \times N$	$(4H + 20R + 8CX) \times N$	$(8H + 40R + 16CX) \times N$
XZY	$(12R + 4CX) \times N$	$(24R + 8CX) \times N$	$(48R + 16CX) \times N$
XZY_H	$(2H + 12R + 4CX) \times N$	$(4H + 24R + 8CX) \times N$	$(8H + 48R + 16CX) \times N$
Y	$(8R + 4CX) \times N$	$(16R + 8CX) \times N$	$(32R + 16CX) \times N$
Y_H	$(2H + 8R + 4CX) \times N$	$(4H + 16R + 8CX) \times N$	$(8H + 32R + 16CX) \times N$
YX	$(10R + 4CX) \times N$	$(20R + 8CX) \times N$	$(40R + 16CX) \times N$
YX_H	$(2H + 10R + 4CX) \times N$	$(4H + 20R + 8CX) \times N$	$(8H + 40R + 16CX) \times N$
YXZ	$(12R + 4CX) \times N$	$(24R + 8CX) \times N$	$(48R + 16CX) \times N$
YXZ_H	$(2H + 12R + 4CX) \times N$	$(4H + 24R + 8CX) \times N$	$(8H + 48R + 16CX) \times N$
YZ	$(10R + 4CX) \times N$	$(20R + 8CX) \times N$	$(40R + 16CX) \times N$
YZ_H	$(2H + 10R + 4CX) \times N$	$(4H + 20R + 8CX) \times N$	$(8H + 40R + 16CX) \times N$
YZX	$(12R + 4CX) \times N$	$(24R + 8CX) \times N$	$(48R + 16CX) \times N$
YZX_H	$(2H + 12R + 4CX) \times N$	$(4H + 24R + 8CX) \times N$	$(8H + 48R + 16CX) \times N$
Z	$(8R + 4CX) \times N$	$(16R + 8CX) \times N$	$(32R + 16CX) \times N$
Z_H	$(2H + 8R + 4CX) \times N$	$(4H + 16R + 8CX) \times N$	$(8H + 32R + 16CX) \times N$
ZX	$(10R + 4CX) \times N$	$(20R + 8CX) \times N$	$(40R + 16CX) \times N$
ZX_H	$(2H + 10R + 4CX) \times N$	$(4H + 20R + 8CX) \times N$	$(8H + 40R + 16CX) \times N$
ZXY	$(12R + 4CX) \times N$	$(24R + 8CX) \times N$	$(48R + 16CX) \times N$
ZXY_H	$(2H + 12R + 4CX) \times N$	$(4H + 24R + 8CX) \times N$	$(8H + 48R + 16CX) \times N$
ZY	$(10R + 4CX) \times N$	$(20R + 8CX) \times N$	$(40R + 16CX) \times N$
ZY_H	$(2H + 10R + 4CX) \times N$	$(4H + 20R + 8CX) \times N$	$(8H + 40R + 16CX) \times N$
ZYX	$(12R + 4CX) \times N$	$(24R + 8CX) \times N$	$(48R + 16CX) \times N$
ZYX_H	$(2H + 12R + 4CX) \times N$	$(4H + 24R + 8CX) \times N$	$(8H + 48R + 16CX) \times N$

Analyzing the tables, it could be observed how the complexity increases linearly with the number of qubits of the system and the number of parameters. Also, the application of the re-uploading technique and entangling block increases the result of this metric.
For the Amplitude Encoding, the results, illustrated in Tables 4.13 and 4.14 are calculated using, also in this case, the PennyLane library. Even in this case, the number of qubits and the techniques adopted influence this metric.

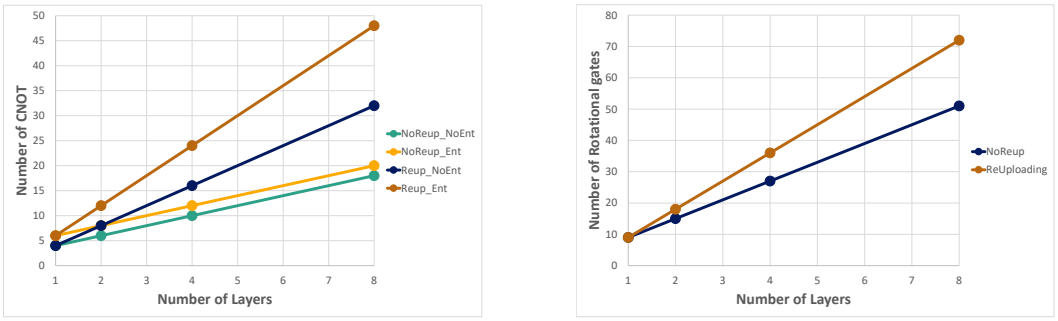
Table 4.13: Report of the complexity for models that use the Amplitude encoding strategy for 2-qubits system calculated by using PennyLane library.

Strategy adopted	**L1**	**L2**	**L4**	**L8**
NoReup_NoEnt	$9R + 4CX$	$15R + 6CX$	$27R + 10CX$	$51R + 18CX$
NoReup_Ent	$9R + 6CX$	$15R + 8CX$	$27R + 12CX$	$51R + 20CX$
Reup_NoEnt	$9R + 4CX$	$18R + 8CX$	$36R + 16CX$	$72R + 32CX$
Reup_Ent	$9R + 6CX$	$18R + 12CX$	$36R + 24CX$	$72R + 48CX$

Table 4.14: Report of the complexity for models that use the Amplitude encoding strategy for 4-qubits system calculated by using PennyLane library.

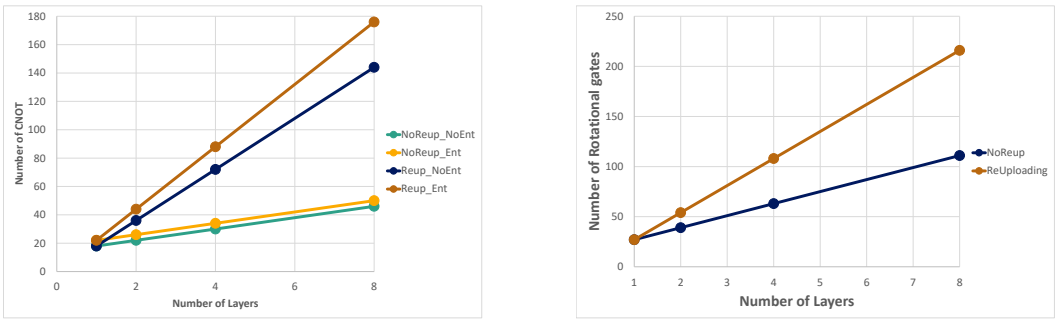
Strategy adopted	**L1**	**L2**	**L4**	**L8**
NoReup_NoEnt	$27R + 18CX$	$39R + 22CX$	$63R + 30CX$	$111R + 46CX$
NoReup_Ent	$27R + 22CX$	$39R + 26CX$	$63R + 34CX$	$111R + 50CX$
Reup_NoEnt	$27R + 18CX$	$54R + 36CX$	$108R + 72CX$	$216R + 144CX$
Reup_Ent	$27R + 22CX$	$54R + 44CX$	$108R + 88CX$	$216R + 176CX$

To detail this behavior, the values of CNOT and rotational gates for the Amplitude encoding for a 2-qubit system and a 4-qubit system are also plotted in Figures 4.22 and 4.23. In particular, the number of Rotational gates depends only on the number of layers and the application of the re-uploading technique. The same dependencies are valid for the number of CNOT, and also it depends on the presence of the entangling layer.



(a) (b)

Figure 4.22: Reporting graph of the complexity for a two-qubit Variational Quantum Circuit. In Figure 4.22a the number of CNOT for each strategy concerning the number of layers is reported. In Figure 4.22b the number of Rotational gates considering just the re-uploading technique to the number of layers.



(a) (b)

Figure 4.23: Reporting graph of the complexity for a four-qubit Variational Quantum Circuit. In Figure 4.23a the number of CNOT for each strategy to the number of layers is reported. In Figure 4.23b, the number of Rotational gates considering just the re-uploading technique to the number of layers.

Number of parameters

Another Figure of Merits evaluated for the Variational Quantum Circuit is the number of parameters of the parametric layer. This value represents a hyper-parameter to consider for the definition of the model; in fact, a higher number of parameters implies a higher simulation and training time. It increases linearly with the number of qubits and layers of the system.

4.3 Support vector machine with Quantum Kernel Evaluation

The Support Vector Machine algorithm with quantum kernel estimation is also developed in the present thesis. The SVM is a classical algorithm used for the classification of data. For this method, kernel functions, which map features in a higher dimensional space, are used. They consist of estimating the distance between two points x_i and x_j. The resulting feature vector is an M-vector, where M is the number of training points, in which each element is evaluated as the distance between a given point and another one of the training set. Then, the results are saved in the Gram matrix, composed of the new M-feature vectors. After the definition of this matrix, a classical optimizer develops and tests the model. The representation of this approach is shown in Fig. 4.24.

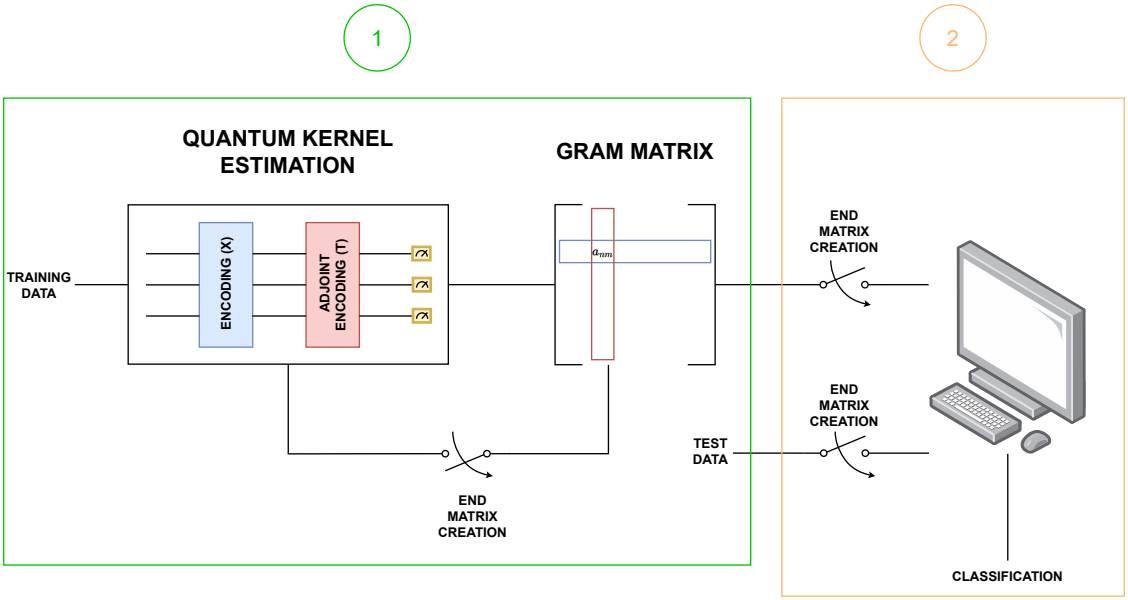


Figure 4.24: Representation of the Kernel evaluation. The steps required are two, the quantum kernel estimation part used to calculate the Gram matrix (1) and the classical optimization (2).

4.3.1 Definition of the Kernel Function

The basic strategy to define a kernel function using quantum devices consists of applying the encoding circuit for the x_i value and the adjoint of the encoding circuit for x_j. If x_i is equal to x_j, the probability of obtaining the ground state at the output of the circuit is maximized, using the Unitary property of the gate matrix. The resulting value, saved in the Gram matrix, corresponds to the probability of

measuring the ground state at the end of the circuit. A generic representation of the Kernel function is displayed in Fig. 4.25.

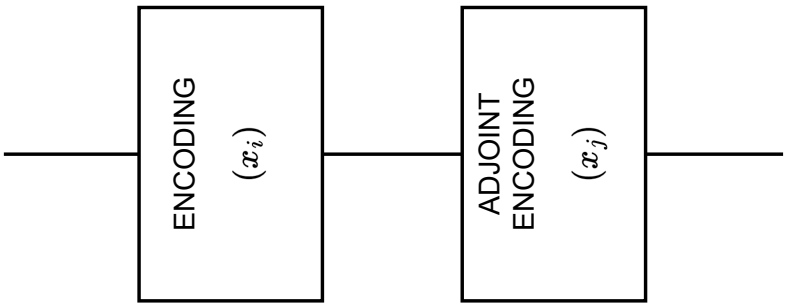


Figure 4.25: Kernel evaluation

4.3.2 Encoding strategies used for the Kernel evaluation

The data encoding techniques exploited for this approach are the same described for the Variational Quantum Circuit in section 4.2.1. The kernel circuit can be defined in a multi-layer approach generating different kernel functions. Considering the time required for the simulation, the circuits defined are composed of a number of layers in the interval [1, 4].

4.3.3 Analysis of the kernel functions defined

The kernel functions defined are analyzed by visualizing their behavior. In this study, the feature vector x_i is fixed while x_j changes in the interval $[0, \pi]$.
The results are plotted from figs. 4.26 to 4.37.

It is important to note that some encodings do not guarantee a correct estimation of the distance between the points. In particular, the problems can be classified into two types: those with constant kernel functions and those with many peaks. The encoding strategies that suffer the first problem are the ones that do not map each feature in a different point of the Bloch Sphere, described also in the Variational Quantum Circuit section. The same encoding strategies that have problems in classifying data in the Variational Quantum Classifier approach suffer from the first problem. The same consideration for the Variational Classifier can be done. The other issue is related to the fact that the kernel function of the basic block is periodic, so repeating the block many times raises the frequency of the kernel function obtained and the estimation of the distance would not be correct.

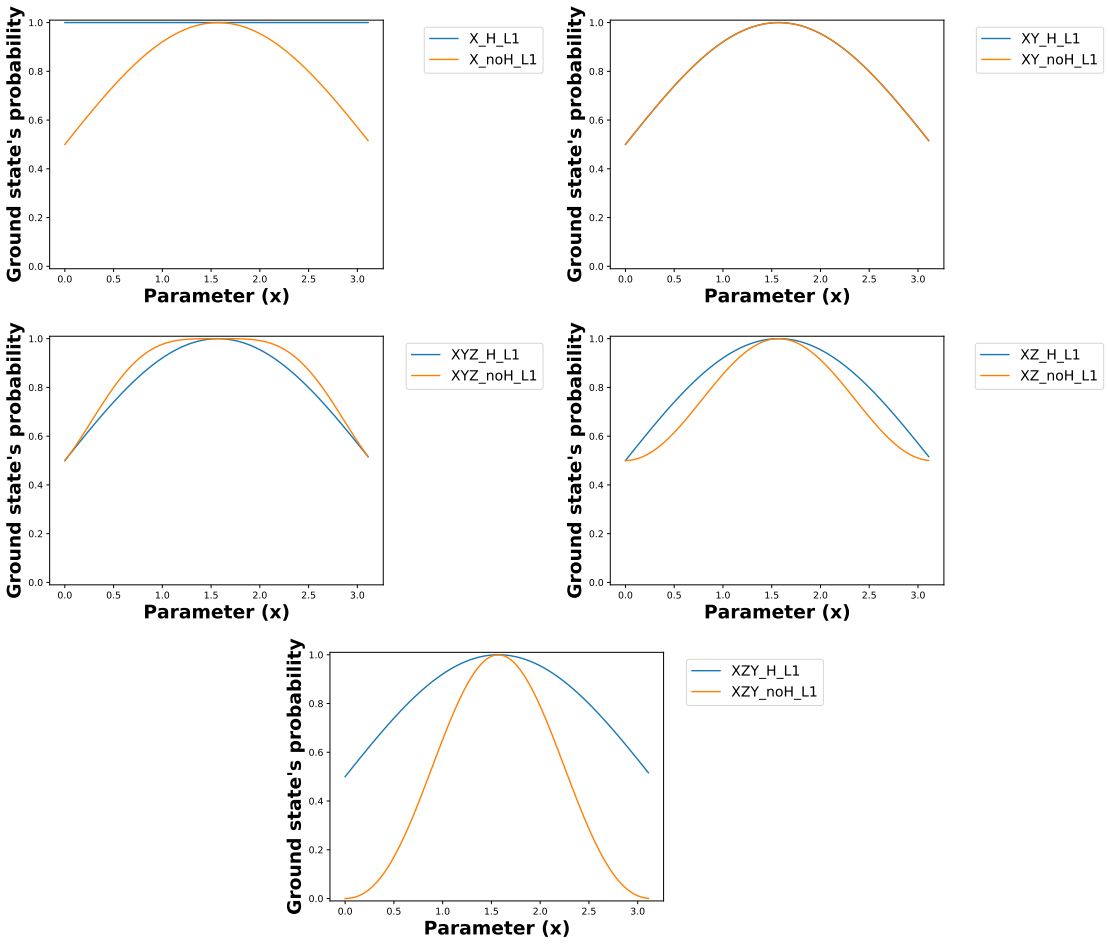


Figure 4.26: Evaluation of kernel function measured using the probability of getting the ground state for the encoding strategies RX, RX-RY, RX-RY-RZ, RX-RZ, RX-RZ-RY using just 1 layer.

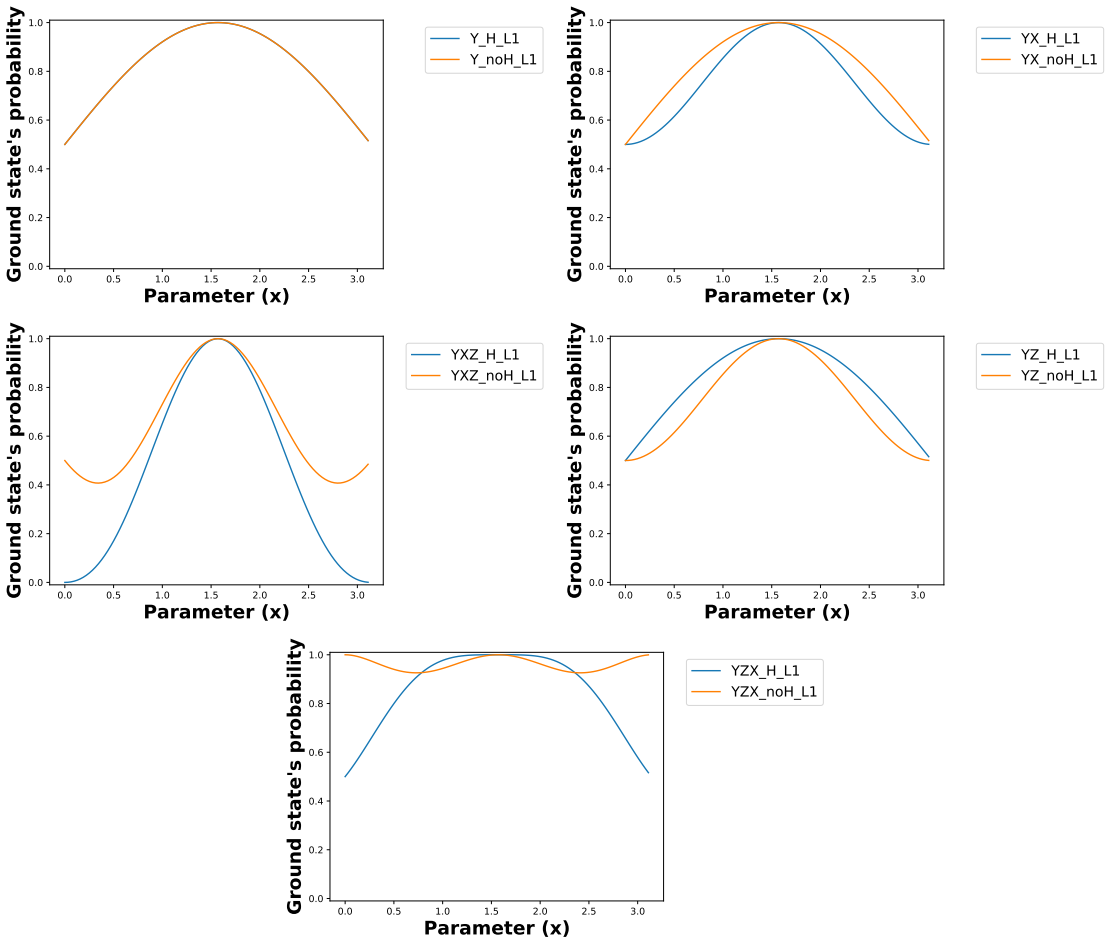


Figure 4.27: Evaluation of kernel function measured using the probability of getting the ground state for the encoding strategies RY, RY-RX, RY-RX-RZ, RY-RZ, RY-RZ-RX using just 1 layer.

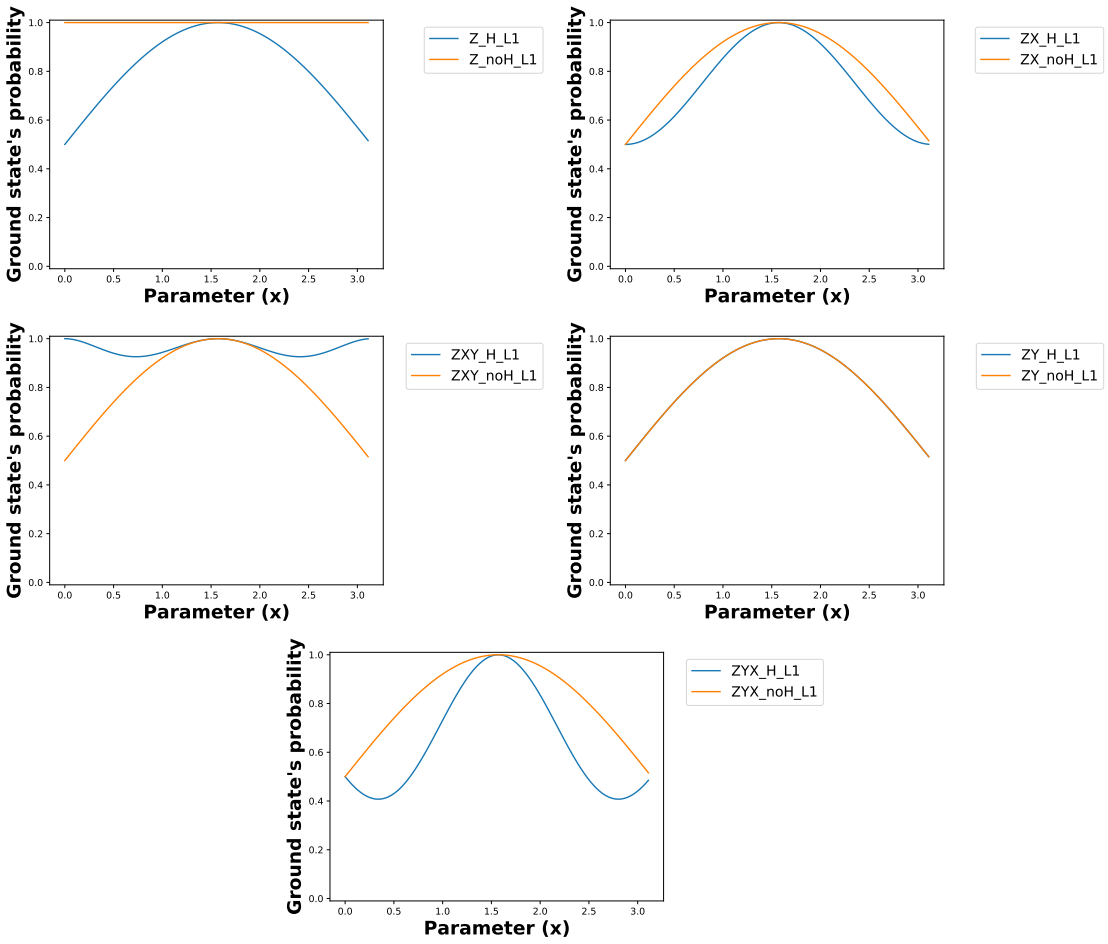


Figure 4.28: Evaluation of kernel function measured using the probability of getting the ground state for the encoding strategies RZ, RZ-RX, RZ-RX-RY, RZ-RY, RZ-RY-RX using just one layer.

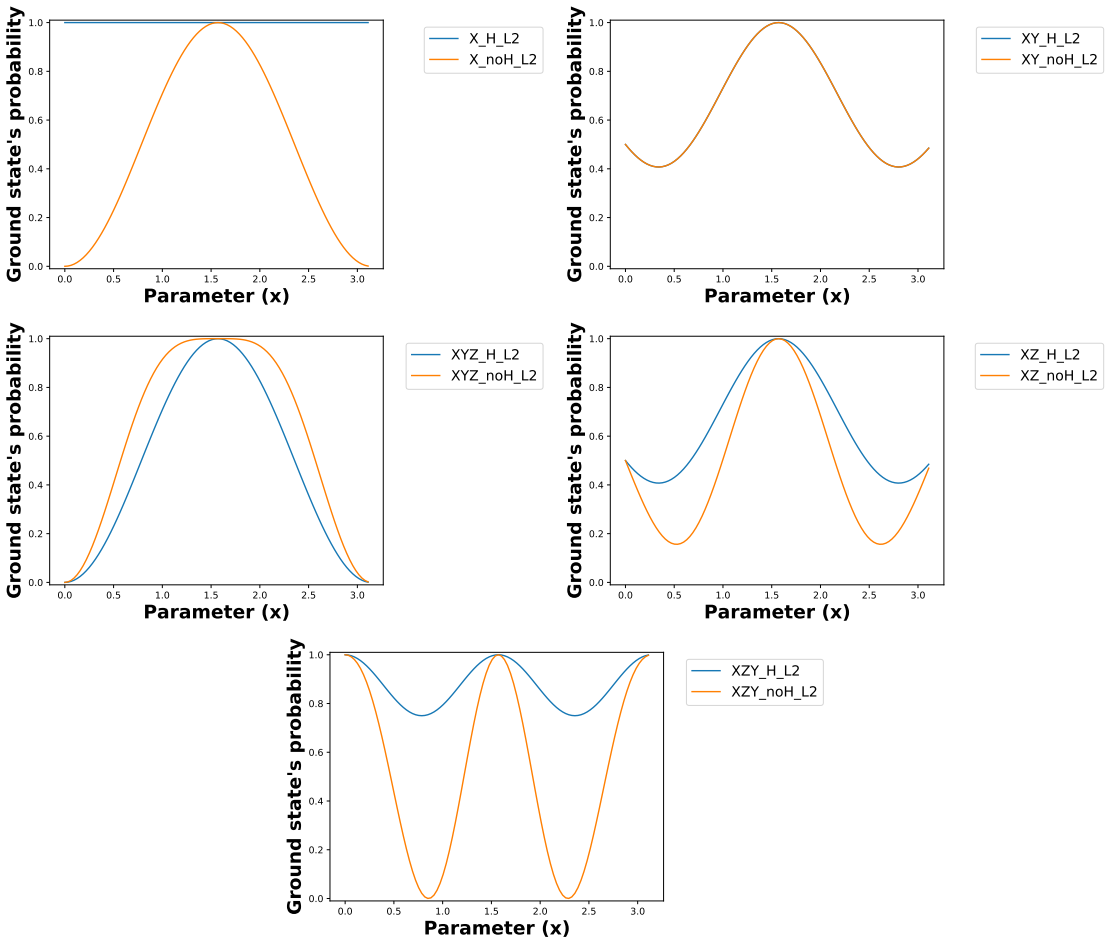


Figure 4.29: Evaluation of kernel function measured using the probability of getting the ground state for the encoding strategies RX, RX-RY, RX-RY-RZ, RX-RZ, RX-RZ-RY using two layers.

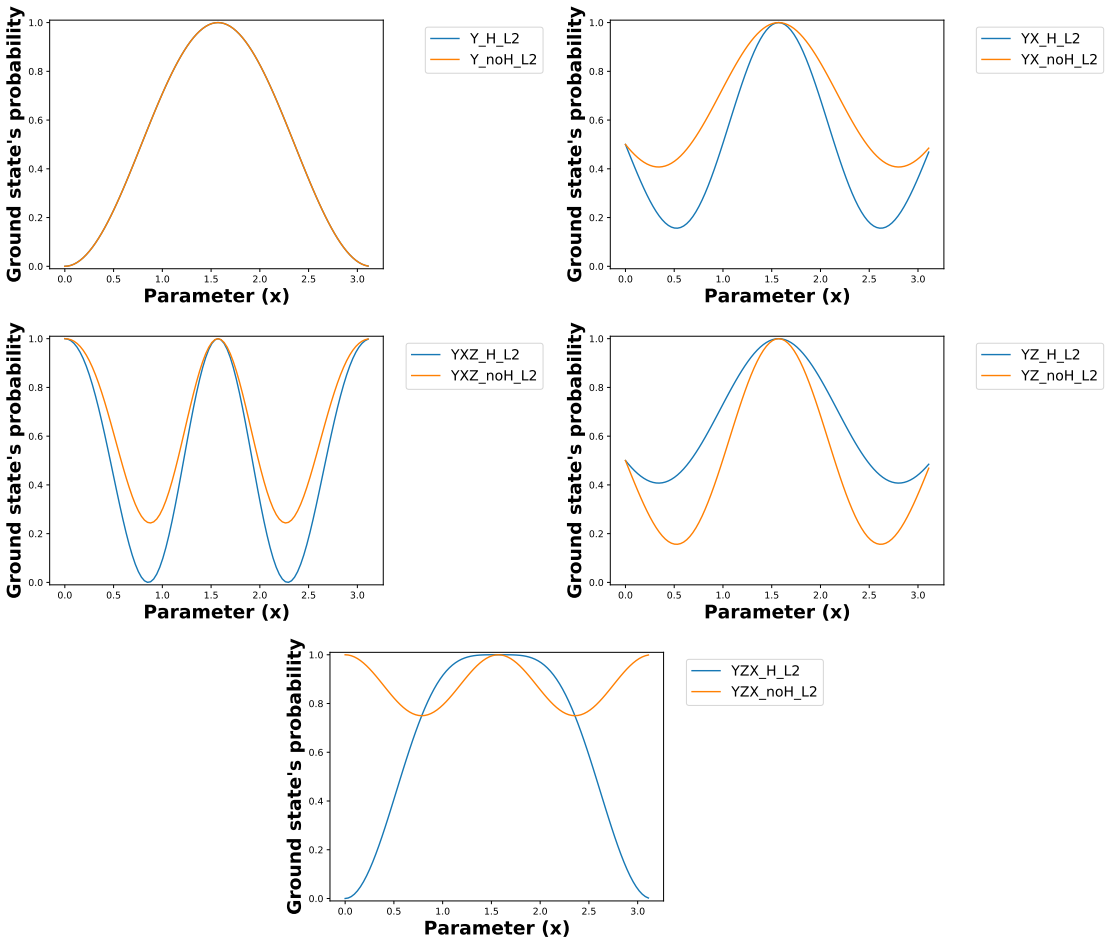


Figure 4.30: Evaluation of kernel function measured using the probability of getting the ground state for the encoding strategies RY, RY-RX, RY-RX-RZ, RY-RZ, RY-RZ-RX using two layers.

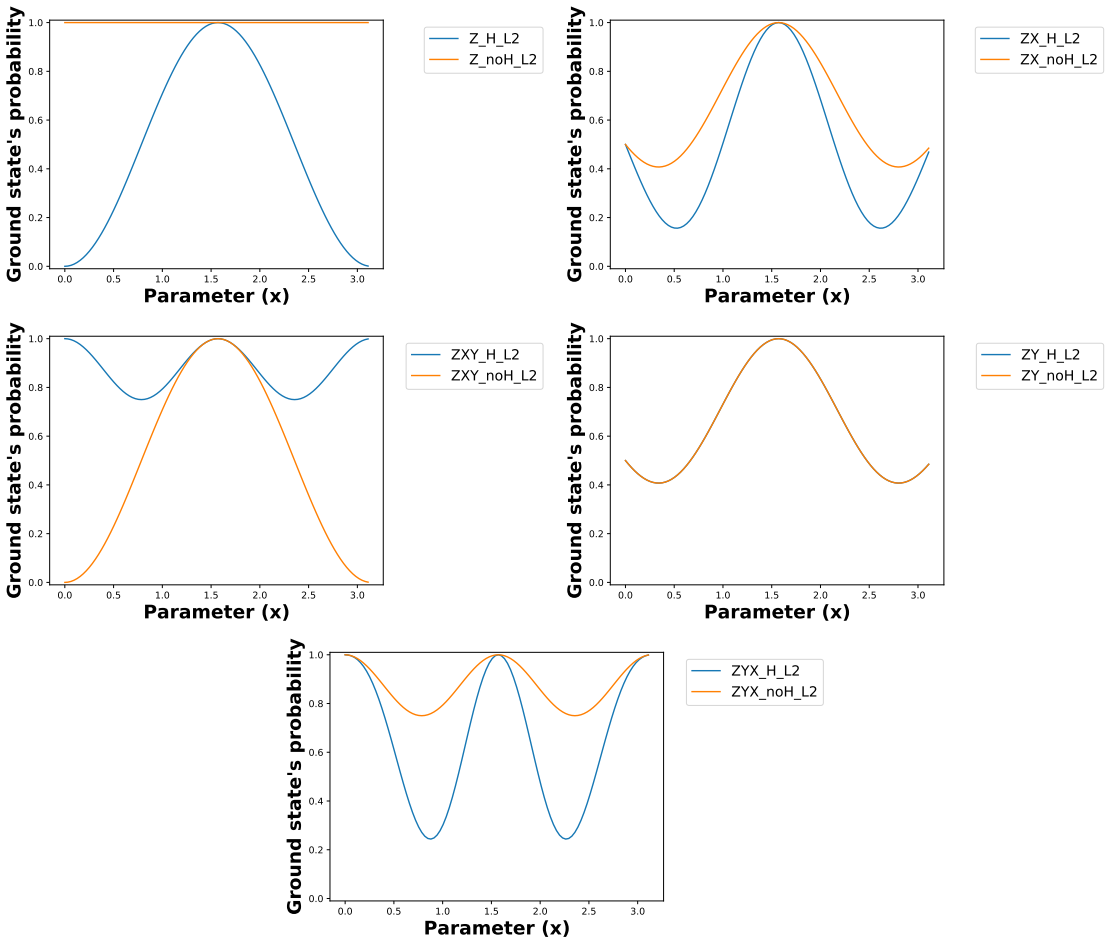


Figure 4.31: Evaluation of kernel function measured using the probability of getting the ground state for the encoding strategies RX, RX-RY, RX-RY-RZ, RX-RZ, RX-RZ-RY using two layers.

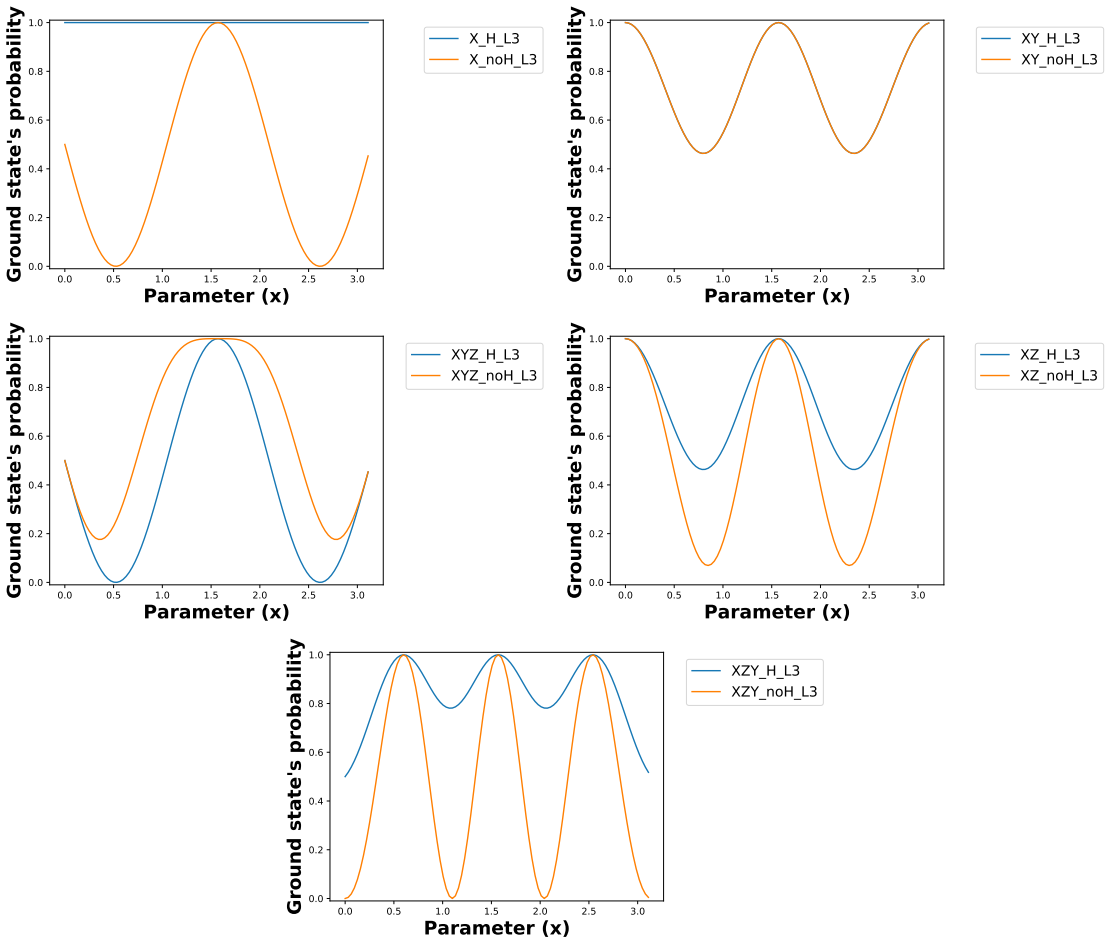


Figure 4.32: Evaluation of kernel function measured using the probability of getting the ground state for the encoding strategies RX, RX-RY, RX-RY-RZ, RX-RZ, RX-RZ-RY using three layers.

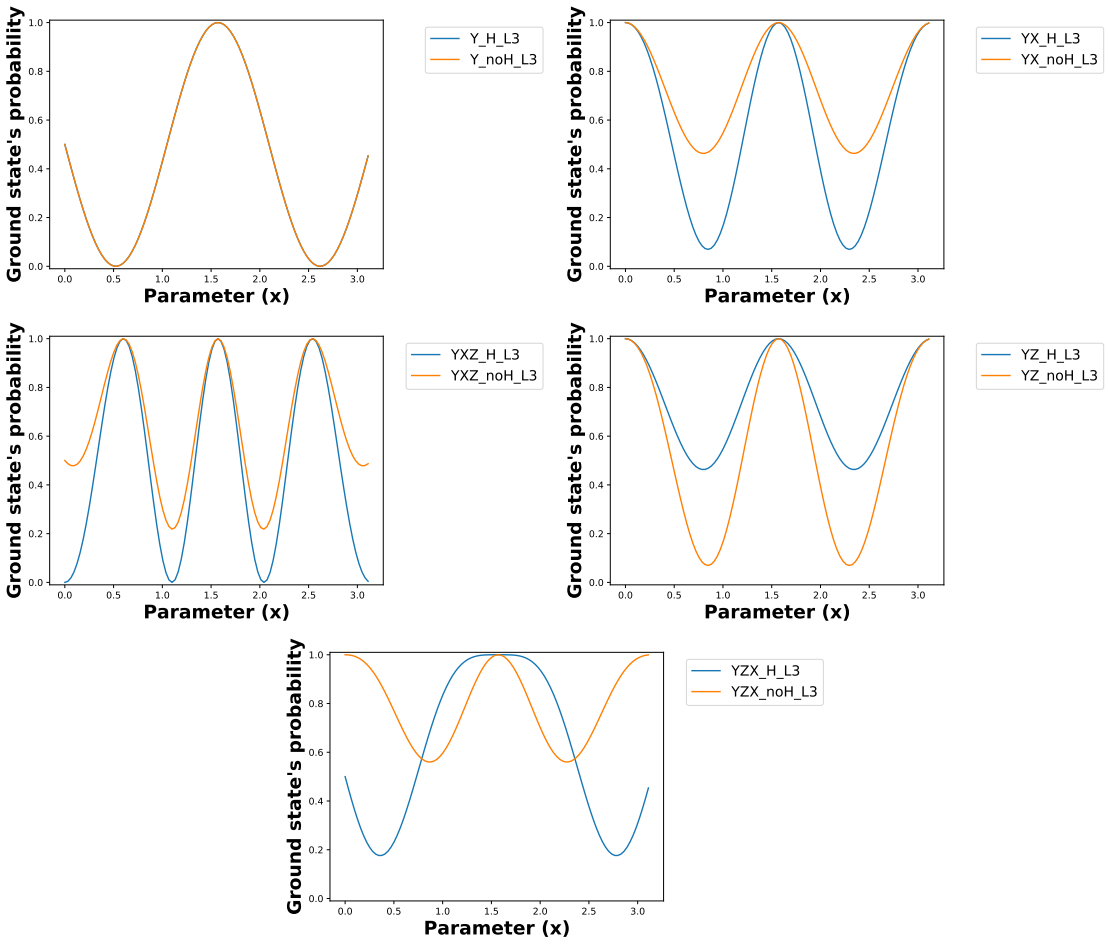


Figure 4.33: Evaluation of kernel function measured using the probability of getting the ground state for the encoding strategies RY, RY-RX, RY-RX-RZ, RY-RZ, RY-RZ-RX using three layers.

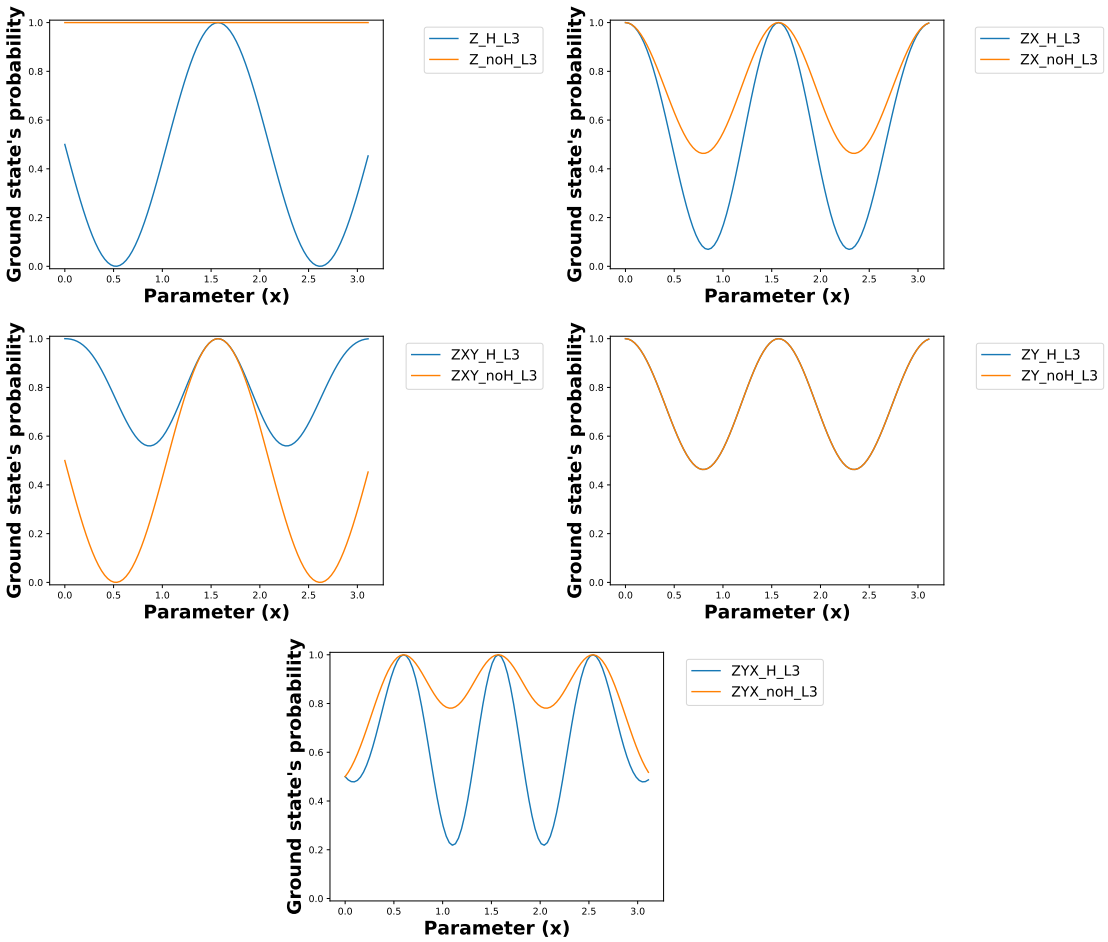


Figure 4.34: Evaluation of kernel function measured using the probability of getting the ground state for the encoding strategies RX, RX-RY, RX-RY-RZ, RX-RZ, RX-RZ-RY using three layers.

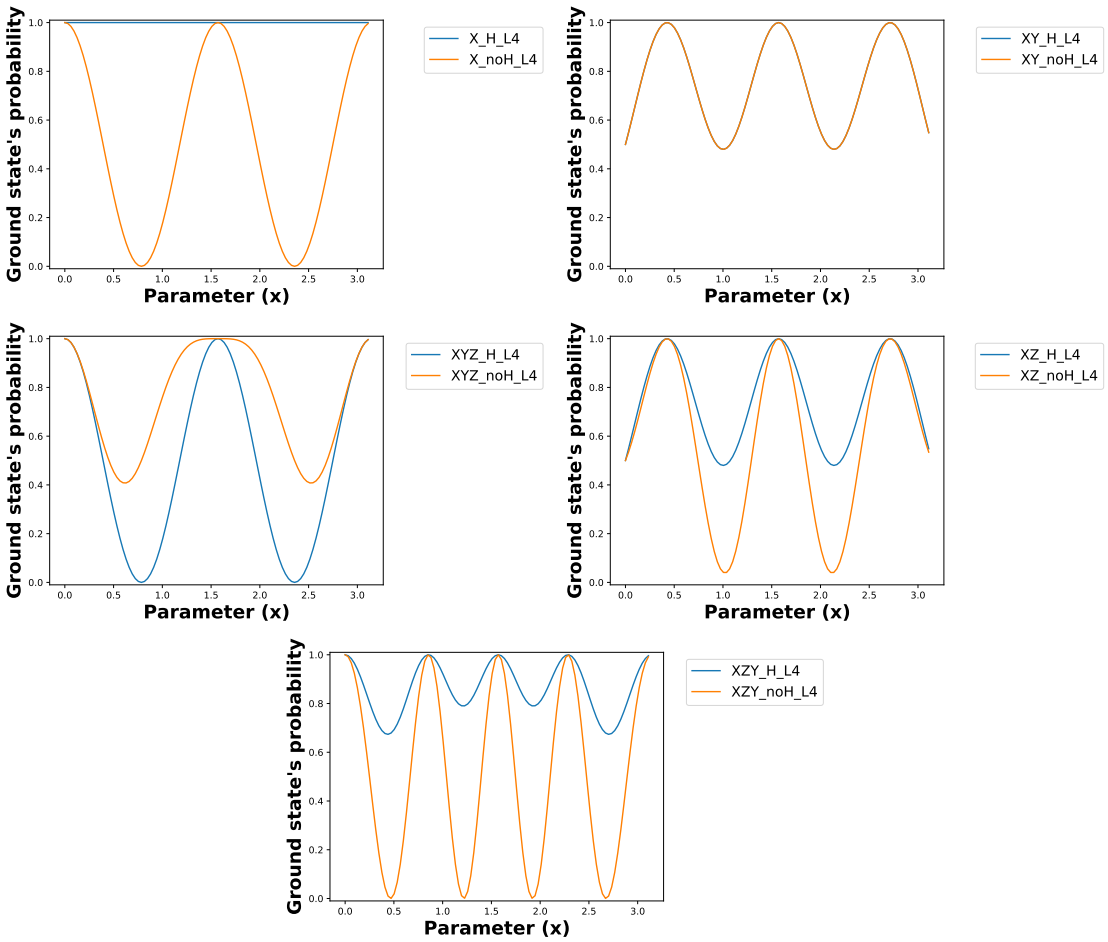


Figure 4.35: Evaluation of kernel function measured using the probability of getting the ground state for the encoding strategies RX, RX-RY, RX-RY-RZ, RX-RZ, RX-RZ-RY using four layers.

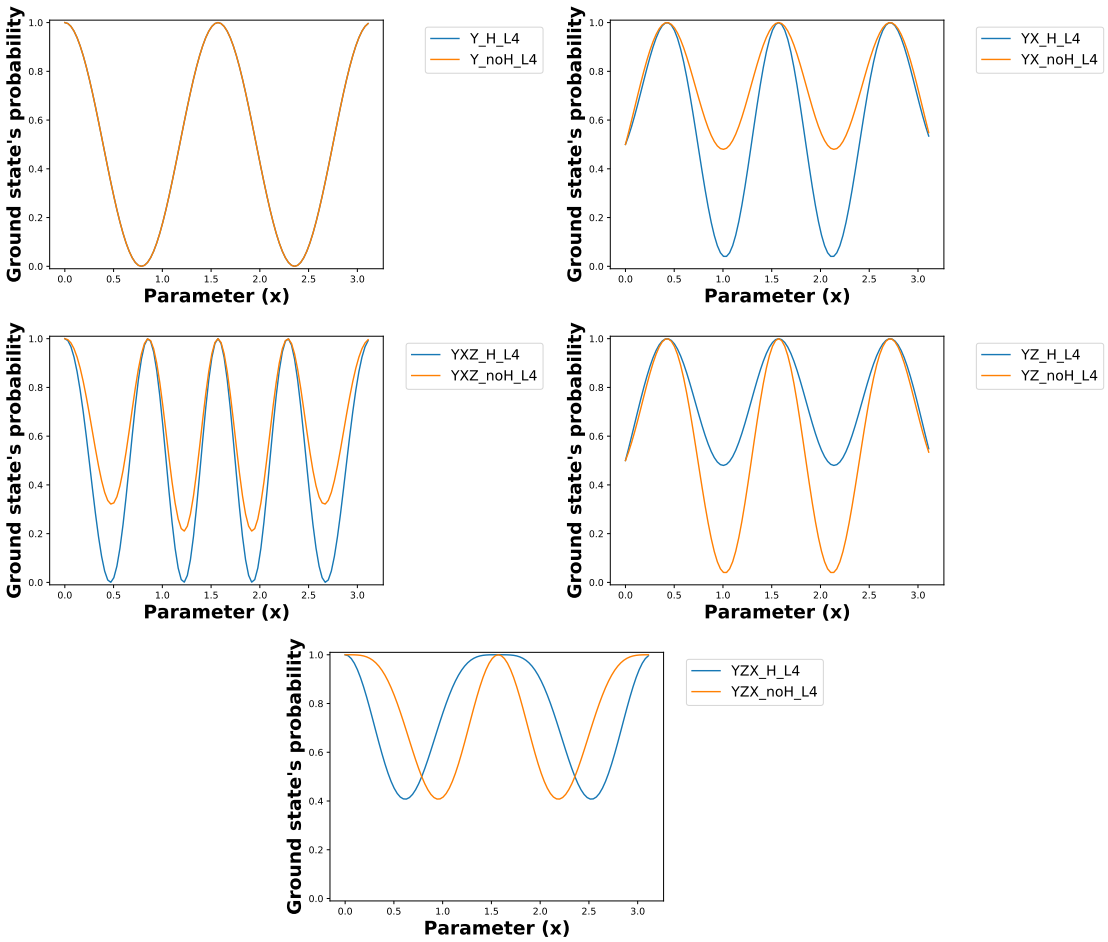


Figure 4.36: Evaluation of kernel function measured using the probability of getting the ground state for the encoding strategies RY, RY-RX, RY-RX-RZ, RY-RZ, RY-RZ-RX using four layers.

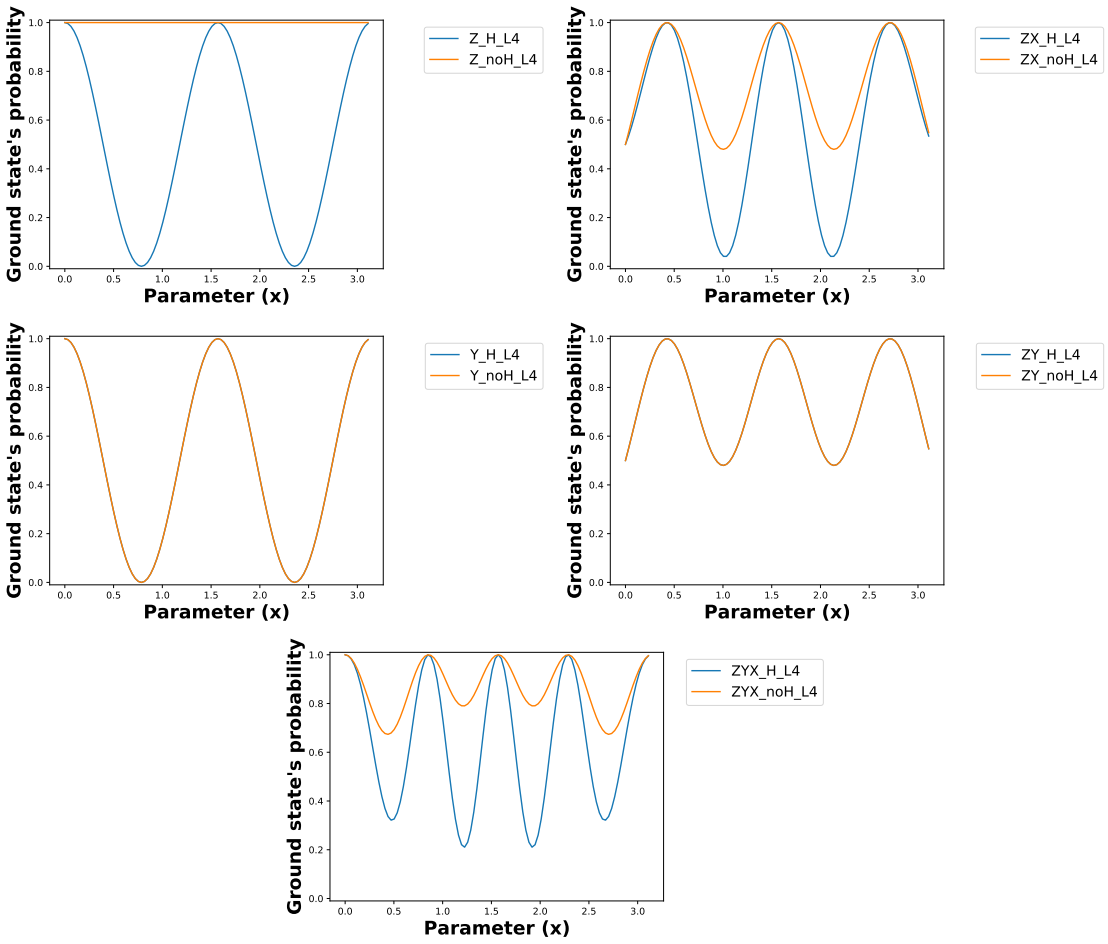


Figure 4.37: Evaluation of kernel function measured using the probability of getting the ground state for the encoding strategies RX, RX-RY, RX-RY-RZ, RX-RZ, RX-RZ-RY using four layers.

4.3.4 Figure of Merits for the evaluation of SVM models

The Figure of Merits described in section 3.3 can be analyzed also for the SVM models, in particular focusing on the **depth** and the **complexity**, both depending on the encoding strategy used.

Depth The depth depends on the encoding gates considered. In particular, it is equal to $E \times L$, where E stands for the number of gates while L is the number of repeated layers for the definition of the kernel circuit.

Complexity For the evaluation of complexity, the contributions of the Hadamard and the Rotational gates are considered. In this case, it is equal to $(2 \times R \times L + 2 \times$

$H) \times N$, where R stands for the number of rotational gates needed for the encoding, L the number of repeated Layers, N is the number of qubits and H is a value equal to one if uniform superposition is applied; otherwise, the contribution is null.

Chapter 5

Results and Conclusions

The previously defined models are tested on real datasets. This chapter describes the methodology adopted and analyzes the results obtained.

5.1 Methodology adopted

Each classification task is characterized through the same methodology. In particular, firstly it is necessary to define the dataset and preprocessing techniques to apply, such as feature scaling and Principal Component Analysing [13] (used to reduce the number of features the model has to manage).

The entire dataset is then split into two parts, the Training set, used for the optimization of the model, and the Test set, a smaller portion used to test the performance of the model. For this thesis, the Training set includes 70% of the entire dataset, while the other 30% is used for the Test portion. The percentage of this split represents a trade-off between the classical 80% Train and 20% Test and 60% Train, 20% Validation (dataset used for choosing the best model considering the hyper-parameters), and 20% Test.

Then, the approach used, the Variational Quantum Circuit or the Support Vector Machine with Quantum Kernel Estimation, is defined. The quantum circuit and the optimization technique adopted for each of the two methods are defined. The optimizer chosen for the Variational Quantum Circuit is the Adam [14], which is a stochastic gradient descent algorithm in which the learning rate, which represents the length of the step, changes at each iteration. For the Support Vector Machine, the optimization is done by using a specific Sci-Kit learn library [15] comprises in the definition of the model.

In the end, the models are tested on the training and test set. This procedure is repeated for each model 10 times, to obtain a mean of the results, reducing the dependency on the randomness of the choice of the parameters.

For the choice of the best model, the following priority of the metrics is defined:

1. Accuracy on the test set
2. Accuracy on the training set
3. Depth and Complexity

This priority order guarantees to choose the model that can better classify data, and, if the accuracy performance is the same, it is selected the one with reduced simulation time.

5.2 Analysis of the dataset used and the classification executed

The first step described in the methodology part is the definition of the dataset. The datasets employed are the Iris[16] and the Wine dataset [17].

The Iris dataset is composed of 150 features vectors, each characterized by four features. Moreover, they are categorized into three classes, each constituted of 50 data. Classes 1 and 2, related to the Versicolour and Virginica Iris, are not linearly separable.
Using this dataset, the classification tasks executed are related to identifying if data belongs to class 0, classifying between classes 1 and 2, and implementing a Multiclass classification, based on evaluating simultaneously all the possible output classes.

The Wine dataset is composed of 178 features vector, characterized by thirteen features and divided into three classes. Also in this case, the classifications are the one that identifies if data belongs to class 0, the classification between class 0 and 1, and the Multiclass classification.

5.3 How the results are represented

The results obtained for each classification are saved in a matrix form using the heatmap, as shown in Figure 5.1. In this way, it is easier to compare the accuracy of the different models; indeed, the yellow cells represent the best ones for this task, while the blue ones are the worst. The encoding gates used are reported on the y-axis, while on the x-axis are defined the other adopted strategies.

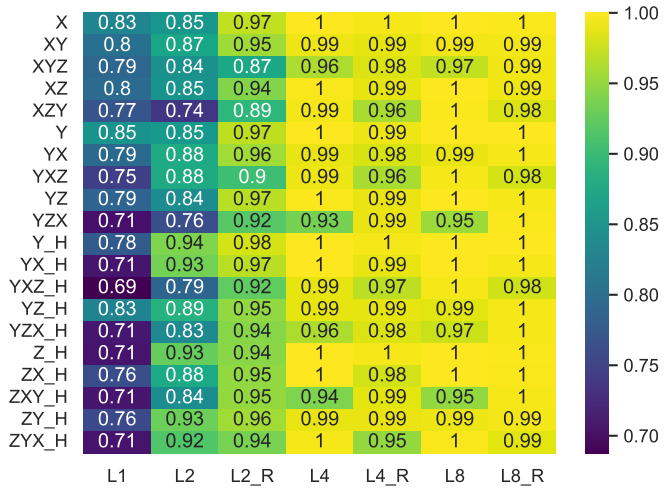


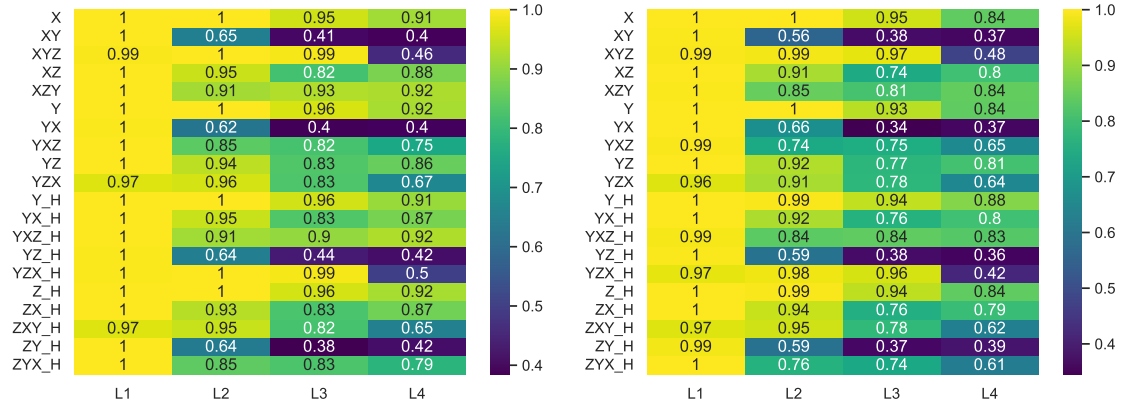
Figure 5.1: Example of a heatmap in which the accuracy results are reported. On the y-axis, there are the encoding gates adopted, while on the x-axis, the strategies exploited.

5.4 Classify if data belongs to class 0 for Iris dataset

The analysis proposed is related to SVM and VQC models, which determine if data belongs to class 0 of the Iris dataset. These models are characterized by the encoding strategy adopted, which influences the number of qubits of the system (equal to two for the Angle Encoding and four for the Amplitude).

5.4.1 Support Vector Machine

For the SVM models, the results are shown in Figure 5.2. For this classification, many strategies reach an accuracy of 100% in both the train and test sets. For this reason, the models are compared by using depth and complexity. The best encoding strategies are the ones that employ RX gates or RY gates, whose depth is equal to 2. The kernel circuits are shown in fig. 5.3



(a) Training set. (b) Test set.

Figure 5.2: Report of the accuracy results for SVM models that classify data belonging to class 0 of the Iris dataset. On the left 5.2a, the models are evaluated on the training set are shown, while on the right 5.2b, the results obtained using the test set.

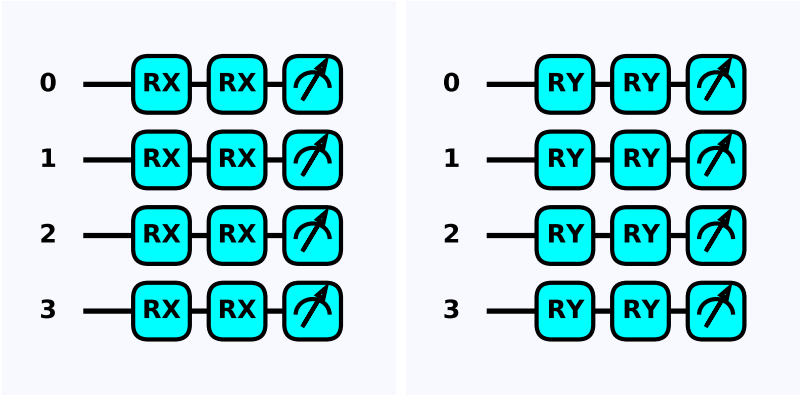
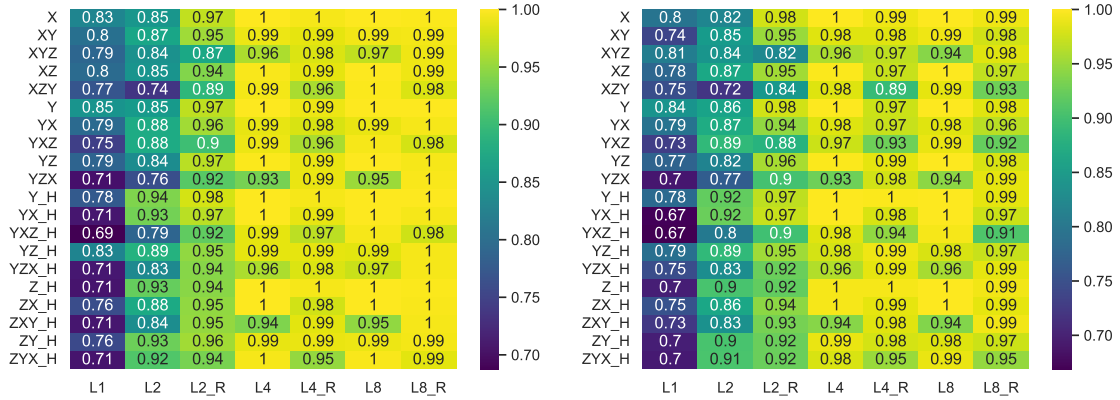


Figure 5.3: Representation of the best kernel circuits for the classification of class 0 data of the Iris dataset. On the left the one that uses RX gates to encode data is reported, while the other uses RY gates.

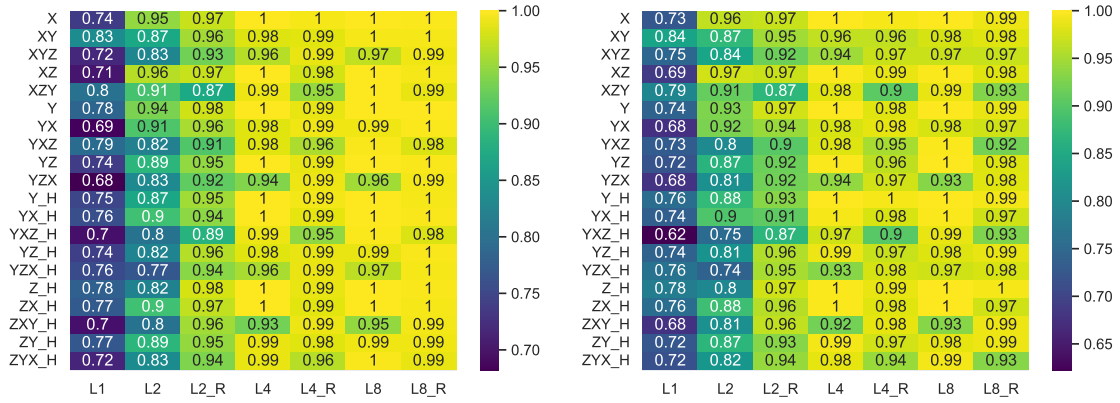
5.4.2 Variational Quantum Classifier

The accuracy results related to the VQC models for classifying data belonging to class 0 are reported from figs. 5.4 to 5.7. Considering that most of them have an accuracy of 100% in both training and test sets, the best models are collected in Table 5.1, considering the techniques that use four parametric layers to maximize the accuracy while minimizing the depth. In the table, the best models are the ones that use the RX or RY gates for embedding data, as shown in Figure 5.8.



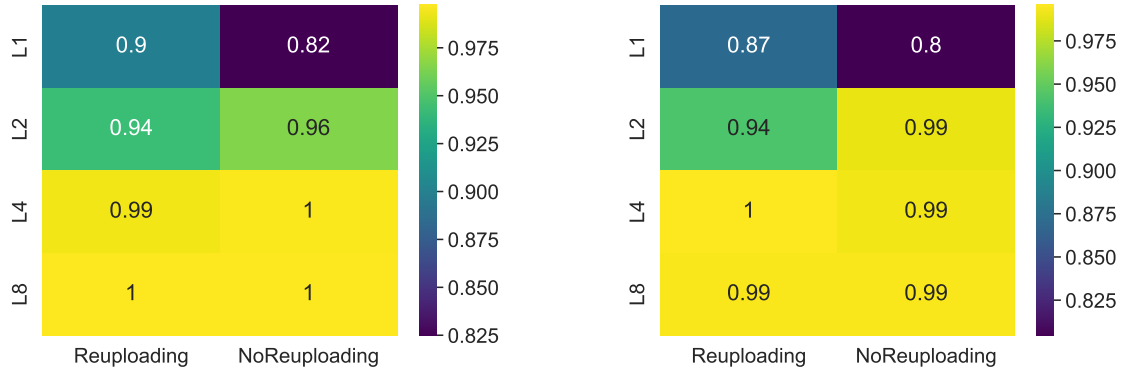
(a) Training set. (b) Test set.

Figure 5.4: Report of the accuracy results for Angle encoding VQC models that classify if data belongs to class 0 of the Iris dataset. On the left 5.4a, the models are evaluated on the training set are shown, while on the right 5.4b, the results related to the test set.



(a) Training set. (b) Test set.

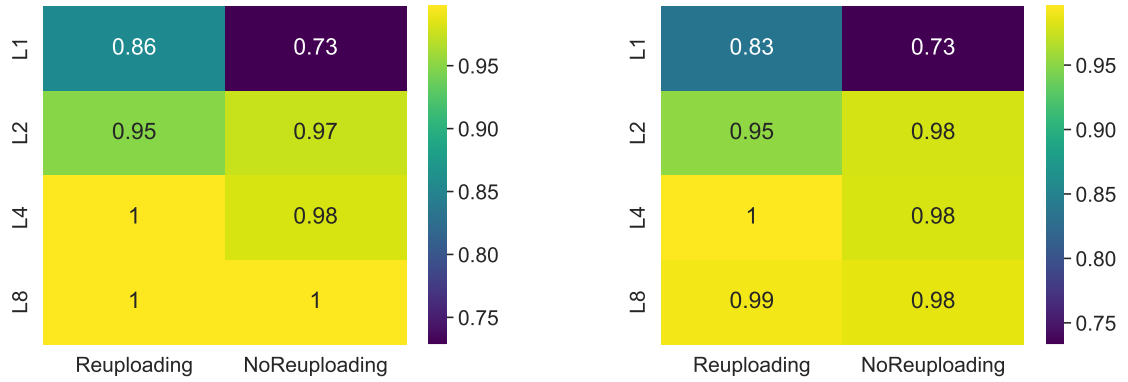
Figure 5.5: Report of the accuracy results for Angle encoding VQC models that use the entangling block that classify if data belongs to class 0 of the Iris dataset. On the left 5.5a, the models are evaluated on the training set are shown, while on the right 5.5b, the results related to the test set.



(a) Training set.

(b) Test set.

Figure 5.6: Report of the accuracy results for Amplitude encoding VQC models that classify if data belongs to class 0 of the Iris dataset. On the left 5.6a, the models evaluated on the training set are shown, while on the right 5.6b, the results related to the test set.



(a) Training set.

(b) Test set.

Figure 5.7: Report of the accuracy results for Amplitude encoding VQC models that uses the entangling block and classify if data belongs to class 0 of the Iris dataset. On the left 5.7a, the models are evaluated on the training set are shown, while on the right 5.7b, the results related to the test set.

Table 5.1: Report of the best VQC models that achieve an accuracy of 100% in both the training and test set to classify data belonging to class 0. As can be seen, the best encoding strategies are characterized by simple RX gate or RY.

Strategy	**Depth**
RX_L4	29
RX-RZ_L4	30
RY_L4	29
RY-RZ_L4	30
RZ-RX_L4	30
RZ-RY_L4	30
H-RY_L4	30
H-RY-RX_L4	31
H-RZ_L4	30
H-RZ-RX_L4	31
RX_ENT_L4	33
RX-RZ_ENT_L4	34
RY_ENT_L4	33
RY-RZ_ENT_L4	34
RZ-RX_ENT_L4	34
RZ-RY_ENT_L4	34
H-RY_ENT_L4	34
H-RY-RX_ENT_L4	35
H-RZ_ENT_L4	34
H-RZ-RX_ENT_L4	35
Amplitude_Reuploading_ENT_L4	44

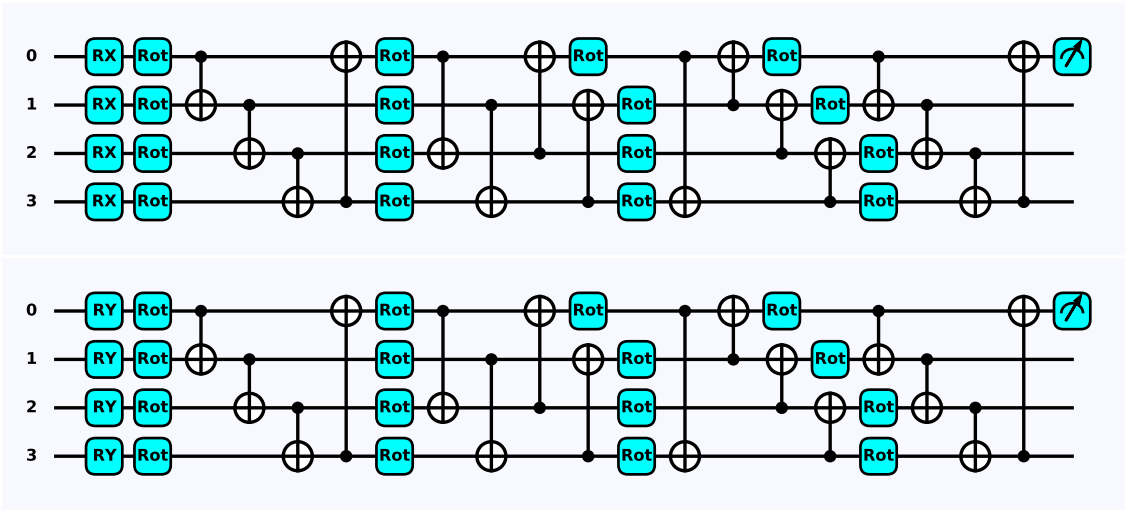


Figure 5.8: Representation of the best VQC for the classification of class 0 data of the Iris dataset. On the top is reported the one that uses RX gates to encode data, while the other uses RY gates. The rotational gate Rot is equal to the application of the RZ, RY, and RZ gates.

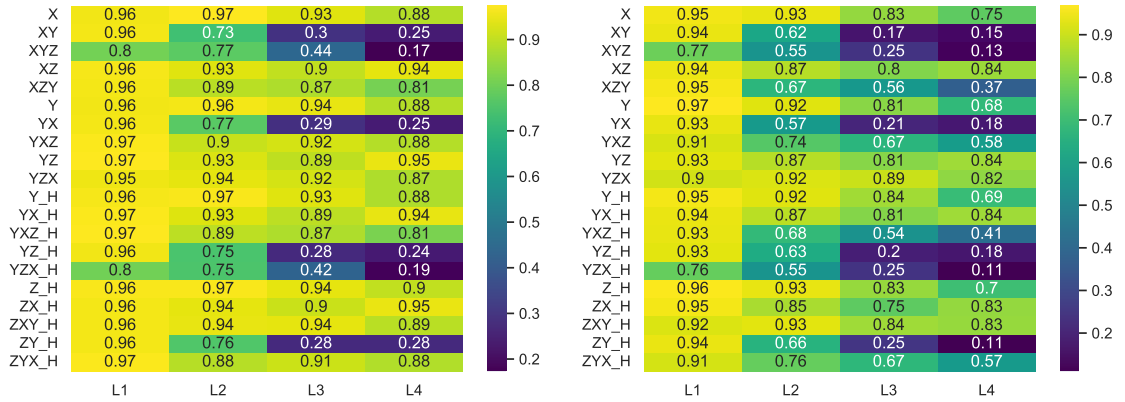
5.5 Classification between classes 1 and 2 of the Iris dataset

The analysis proposed is related to VQC and SVM models, which try to classify non-linearly separable data belonging to classes 1 and 2. These models are characterized by the encoding strategy exploited, which influences the number of qubits of the system (equal to two for Angle Encoding and to four for Amplitude Encoding).

5.5.1 Support Vector Machine

The accuracy results for models that use the Support Vector Machine approach are reported in Figure 5.9.

The best accuracy obtained on the test set is equal to 0.97. It is achieved by the single-layer circuit which uses the RY gate to embed the data, as shown in fig. 5.10.



(a) Training set. (b) Test set.

Figure 5.9: Report of the accuracy results for SVM models that classify data between non-separable classes 1 and 2 of Iris dataset. On the left 5.9a, the models evaluated on the training set are shown, while on the right 5.9b, the results obtained using the test set.

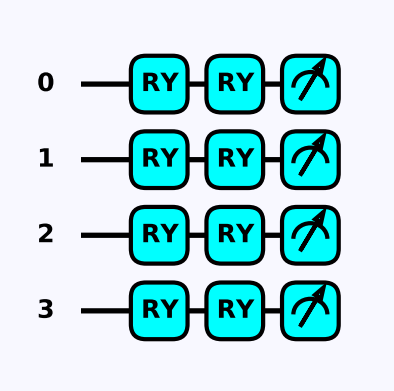
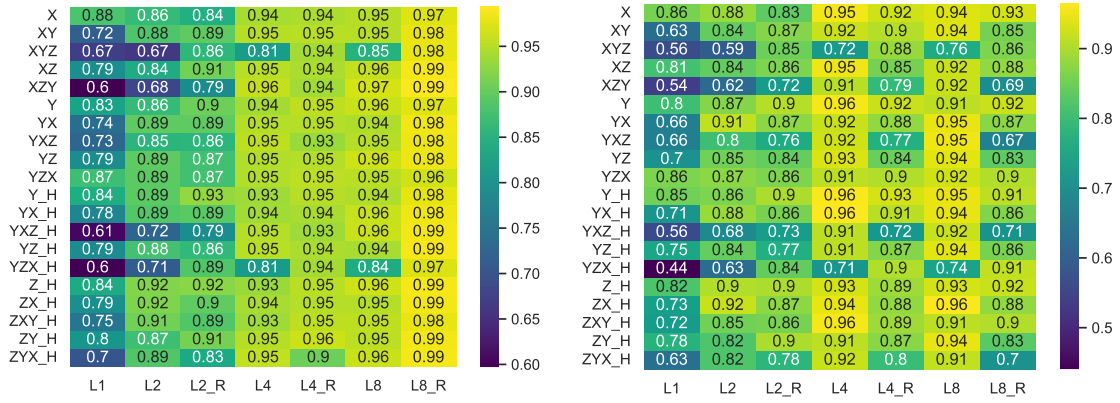


Figure 5.10: Representation of the best kernel circuit for the classification between non-linearly separable classes 1 and 2 of the Iris dataset. The encoding circuit used is characterized by RY gates.

5.5.2 Variational Quantum Classifier

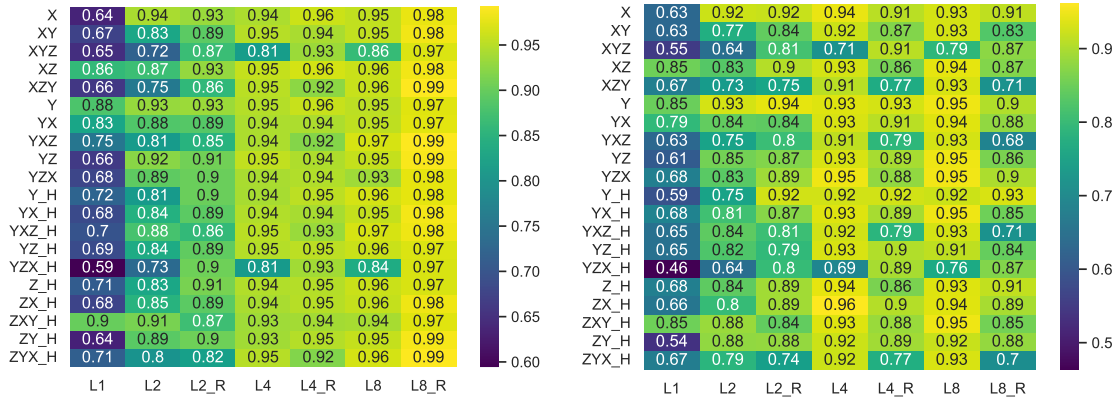
The accuracy results related to the VQC model for the classification between class 1 and 2 data of the Iris dataset are shown from figs. 5.11 to 5.14. From the heatmaps, the strategies with the best performance in the test set are reported in Table 5.2. There, the VQCs are compared using the accuracy in the training set and the depth. The best model identified is constituted of RY gates for the encoding and four parametric layers, shown in fig. 5.15.



(a) Training set.

(b) Test set.

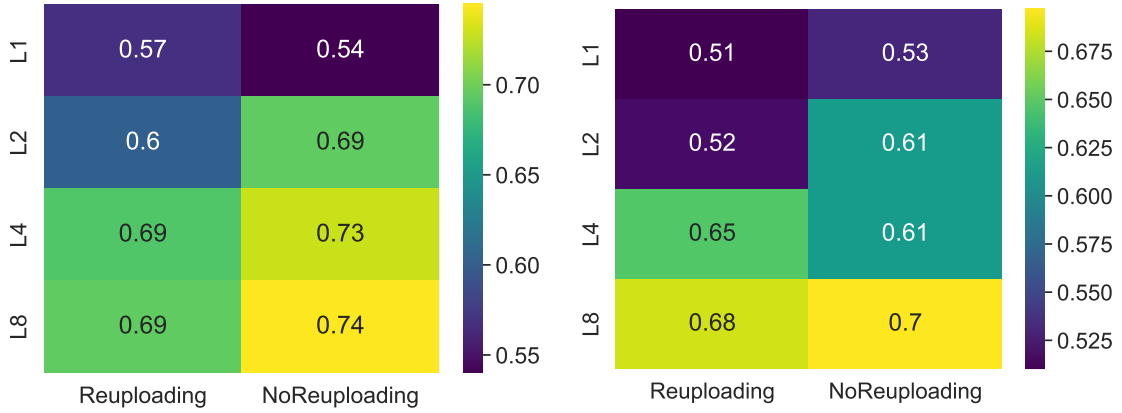
Figure 5.11: Report of the accuracy results for Angle encoding VQC models that classify data between non-separable classes 1 and 2 of the Iris dataset. On the left 5.11a, the models evaluated on the training set arr shwon, while on the right 5.11b, the results related to the test set.



(a) Training set.

(b) Test set.

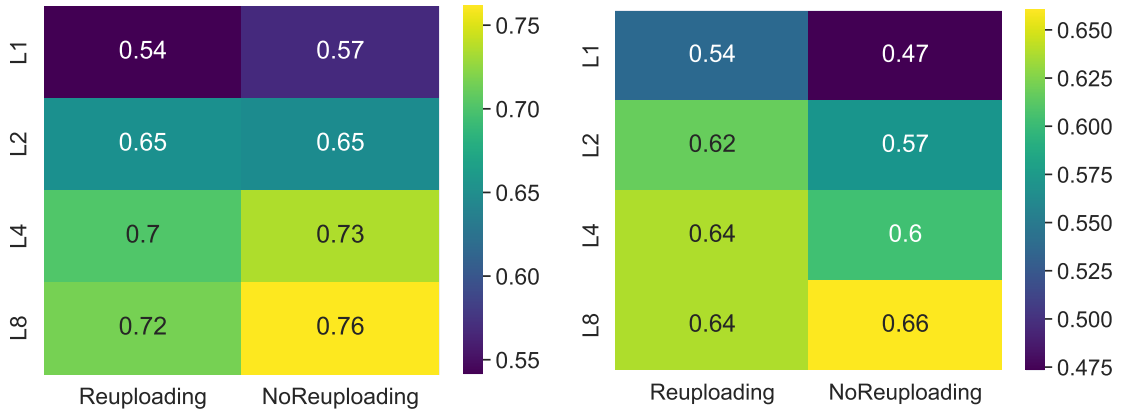
Figure 5.12: Report of the accuracy results for Angle encoding VQC models with the entangling block used for classifying data between non-separable classes 1 and 2 of the Iris dataset. On the left 5.12a, the models evaluated on the training set are shown, while on the right 5.12b, the results related to the test set.



(a) Training set.

(b) Test set.

Figure 5.13: Report of the accuracy results for Amplitude encoding VQC models used for classifying data between non-separable classes 1 and 2 of the Iris dataset. On the left 5.13a, the models evaluated on the training set are shown, while on the right 5.13b, the results related to the test set.



(a) Training set.

(b) Test set.

Figure 5.14: Report of the accuracy results for Amplitude encoding VQC models with the entangling block used for classifying data between non-separable classes 1 and 2 of the Iris dataset. On the left 5.14a, the models evaluated on the training set are shown, while on the right 5.14b, the results related to the test set.

Table 5.2: Report of the best encoding strategies for accuracy in the test set. As can be seen, the model that uses RY gates for the encoding and four parametric layers is the best one in terms of accuracy and depth.

Encoding Strategy	Accuracy Test	Accuracy Train	Depth	Complexity
H-RY_L4	0.96	0.94	30	4H + 52R + 16CX
RY_L4	0.96	0.95	29	52R + 16CX
H-RY-RX_L4	0.96	0.94	31	4H + 56R + 16 CX
H-RZ-RX_L8	0.96	0.95	59	4H + 104R + 32CX
H-RZ-RX-RY_L4	0.96	0.93	32	4H + 60R + 16CX
H-RZ-RX_ENT_L4	0.96	0.94	35	4H + 56R + 20CX

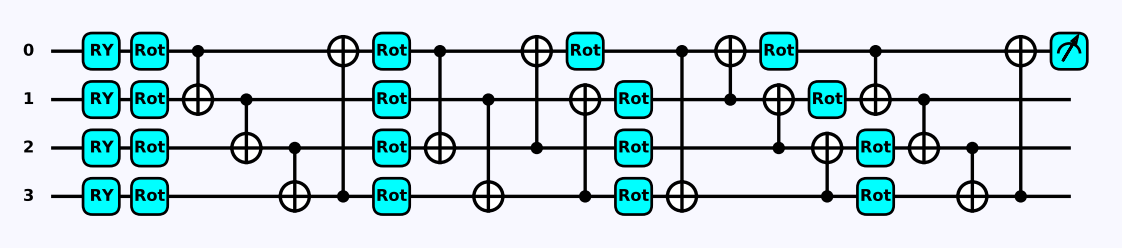
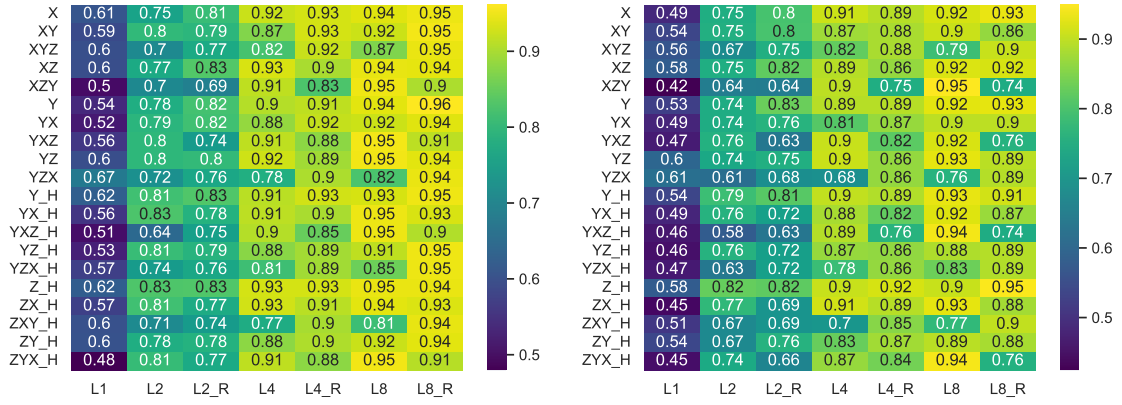


Figure 5.15: Representation of the best VQC for the classification of data belonging to the non-separable classes 1 and 2 of the Iris dataset. The rotational gate Rot is equal to the application of the RZ, RY, and RZ gates.

5.6 Multiclass classification for Iris dataset

The objective of multiclass classification is to determine to which class data belong among the three available. In this case, the only approach tested is the VQC. In fact, SVM is based on binary classification and, for the multiclass task, requires the implementation of three different models.

Moreover, the only encoding strategies used are Angle Encoding. The reason is that this particular classification circuit must have at least three qubits, each associated with a class, and by using Amplitude encoding, the system would have only two qubits. The results of this classification task are reported in figs. 5.16 and 5.17b. To visualize the models with the best accuracy in the test set, they are reported in table 5.3. The best model found is composed of an encoding part realized using RX-RY-RZ gates and eight parametric layers.



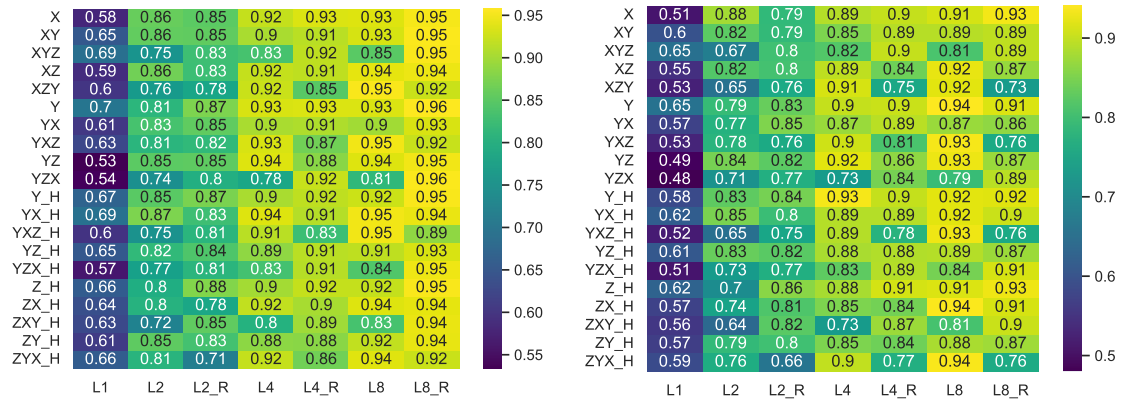
(a) Training set.

(b) Test set.

Figure 5.16: Heatmaps representing the accuracy results for Angle encoding VQC models for multiclass classification of the Iris dataset. On the left 5.16a, the models evaluated on the training set are shown, while on the right 5.16b, the results related to the test set.

Table 5.3: Best accuracy result in Test set for the Multiclass classification. The best model is the one that uses RX-RZ-RY gates for the encoding and eight parametric layers.

Strategy	Accuracy Test	Accuracy Train
H-Z_Reuploading	0.95	0.94
RX-RZ-RY_L8	0.95	0.95



(a) Training set.

(b) Test set.

Figure 5.17: Heatmaps representing the accuracy results for Angle encoding VQC models with the entangling block for multiclass classification of the Iris dataset. On the left 5.17a, the models evaluated on the training set are shown, while on the right 5.17b, the results related to the test set.

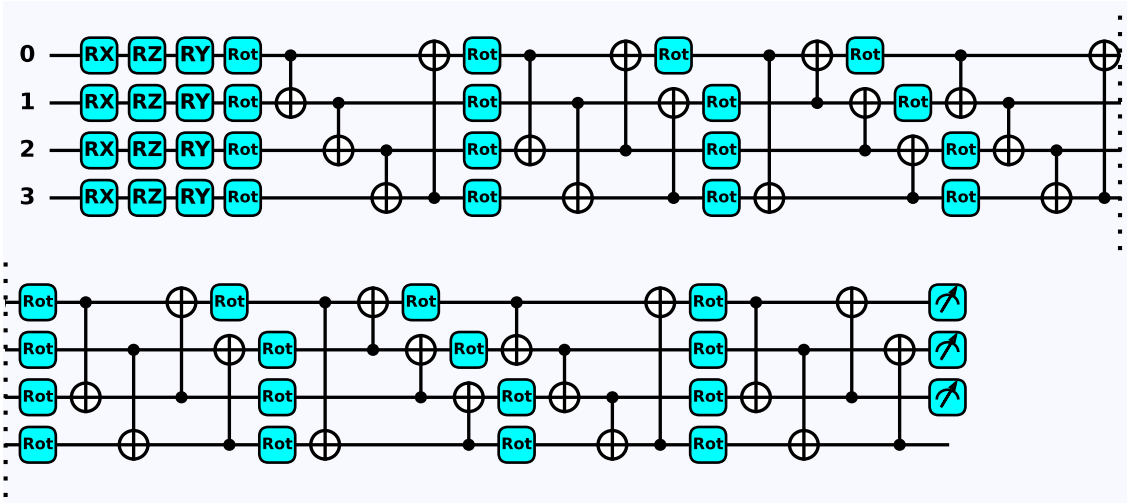


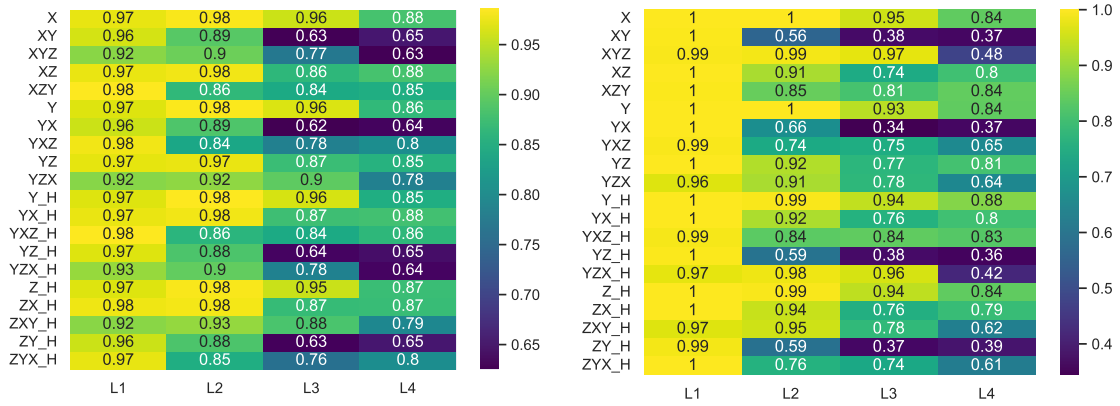
Figure 5.18: Representation of the best VQC model for the multiclass classification of the Iris dataset, characterized by RX-RZ-RY gates for the encoding, and eight parametric layers. The Rot gate corresponds to the sequential application of RZ-RY-RZ gates.

5.7 Classify if data belongs to class 0 for Wine dataset

The analysis proposed is related to both the VQC and SVM models to determine if data belongs to class 0 of the Wine dataset. The number of qubits and the preprocessing techniques depend on the encoding strategies adopted. For Angle Encoding, a Principal Component Analysis is applied to reduce the number of features from thirteen to five. For Amplitude Encoding, no other preprocessing techniques over feature scaling are applied, and the number of qubits of the system is equal to four.

5.7.1 Support Vector Machine

The results of this classification for the SVM model are reported in fig. 5.19. The encoding strategies used by the circuit associated with the SVM model with better performance on the test set are highlighted in table 5.4. From this subset, the best one found is the circuit that uses RX gates for the encoding and the layer is repeated 2 times, as shown in fig. 5.20.



(a) Training set.

(b) Test set.

Figure 5.19: Report of the accuracy results for Support Vector Machine models for the classification data belonging to class 0 of the Wine dataset. In the left, 5.19a, the models evaluated on the training set are shown, while on the right, 5.19b the results related to the test set.

Table 5.4: Best SVM model identified to classify if data belongs to class 0 of the Wine dataset. The best circuit is characterized by RX gates for the encoding and the layer is repeated two times.

Strategy	Accuracy test	Accuracy train	Depth
RY_L1	0.97	0.97	2
H-RY_L1	0.97	0.97	4
H-RZ-RX_L1	0.97	0.98	6
RX_L2	0.97	0.98	4

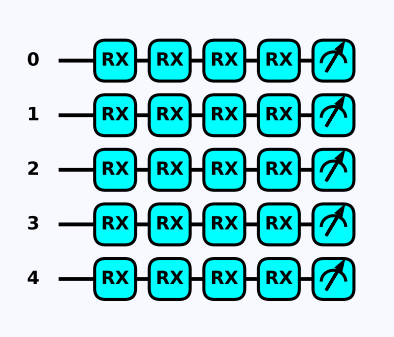
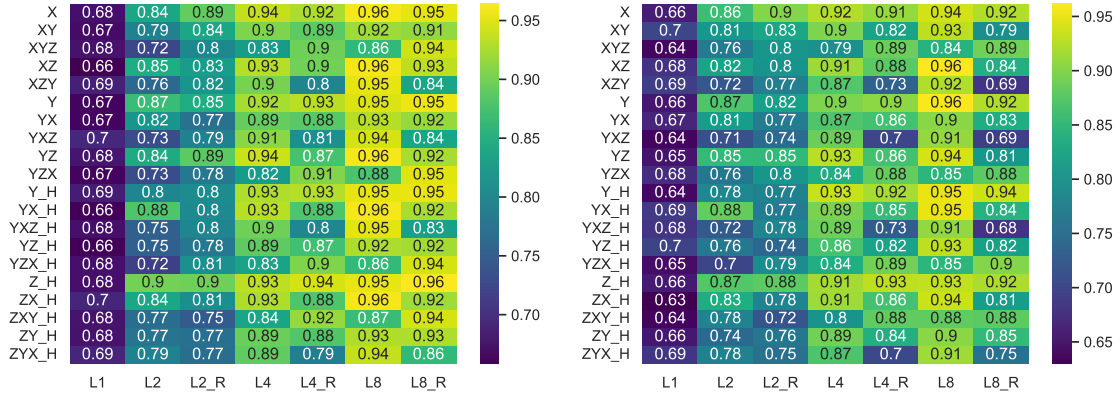


Figure 5.20: Representation of the best kernel circuit to classify if data belongs to class 0. The encoding circuit used is characterized by RX gates and the circuit is repeated 2 times.

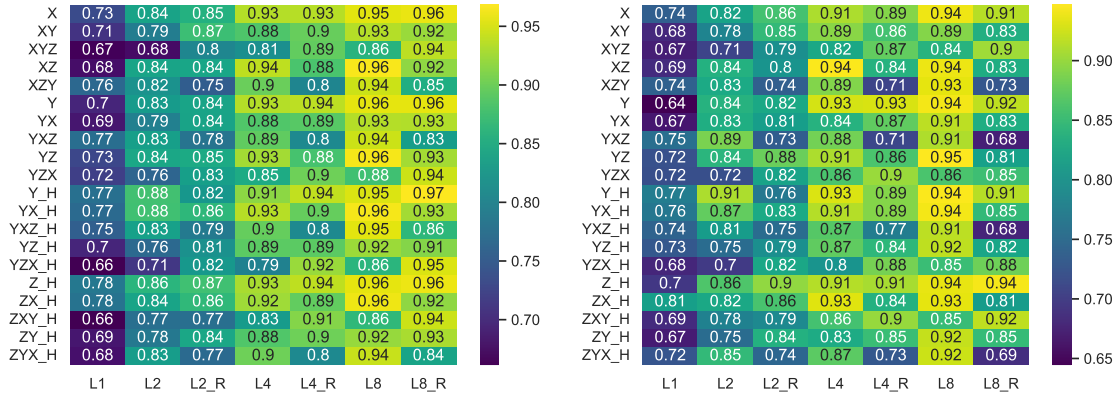
5.7.2 Variational Quantum Circuit

The accuracy results related to the VQC models for classifying if data belongs to class 0 are reported from figs. 5.21 to 5.24. As can be seen, the Amplitude Encoding models, which do not require Principal Component Analysis, perform better than the Angle ones. The strategies that achieve the highest value of accuracy on the training and test sets are compared through the depth values in Table table 5.5. As can be seen, the best choice consists in applying Amplitude Encoding followed by four parametric layers, as shown in fig. 5.25.



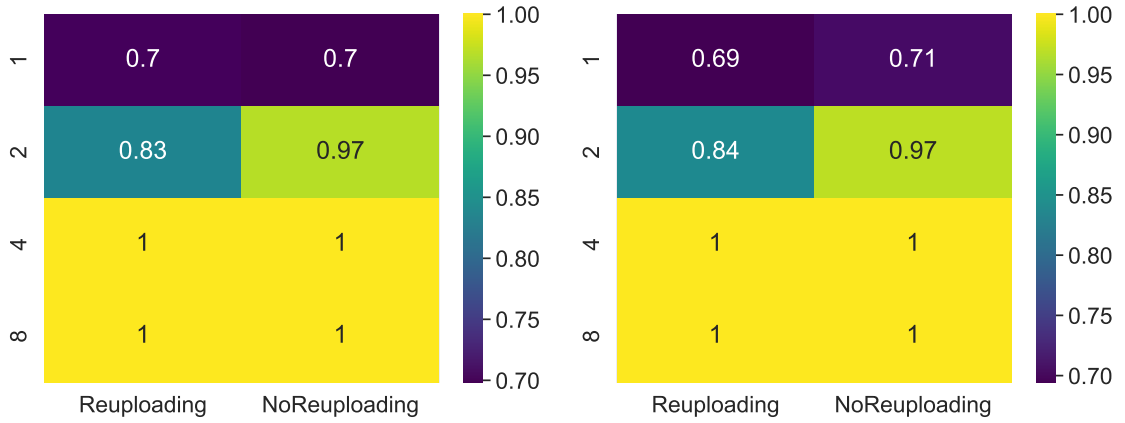
(a) Training set. (b) Test set.

Figure 5.21: Report of the accuracy results for Angle encoding VQC models that classify data belonging to class 0 of the Wine dataset. On the left 5.21a, the models evaluated on the training set are shown, while on the right 5.21b, the results related to the test set.



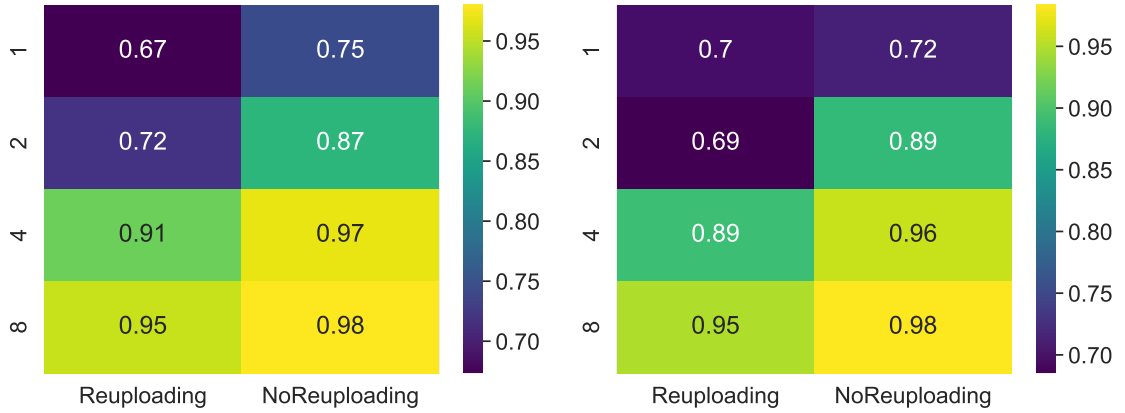
(a) Training set. (b) Test set.

Figure 5.22: Report of the accuracy results for Angle encoding VQC models that use the entangling block that classify data belonging to class 0 of the Wine dataset. On the left 5.22a, the models evaluated on the training set are shown, while on the right 5.22b, the results related to the test set.



(a) Training set. (b) Test set.

Figure 5.23: Report of the accuracy results for Amplitude encoding VQC models that classify data belonging to class 0 of the Wine dataset. On the left 5.23a, the models evaluated on the training set are shown, while on the right 5.23b, the results related to the test set.



(a) Training set. (b) Test set.

Figure 5.24: Report of the accuracy results for Amplitude encoding VQC models that uses the entangling block and classify data belonging to class 0 of the Wine dataset. On the left 5.24a, the models evaluated on the training set are shown, while on the right 5.24b, the results related to the test set.

Table 5.5: Best encoding strategies for the Variational Quantum Circuit used to classify if data belongs to class 0 of the Wine dataset. The best model is defined using Amplitude Encoding and four parametric layers.

Strategy	Depth
Amplitude_L4	51
Amplitude_Reuploading_L4	132
Amplitude_L8	74
Amplitude_Reuploading_L8	264

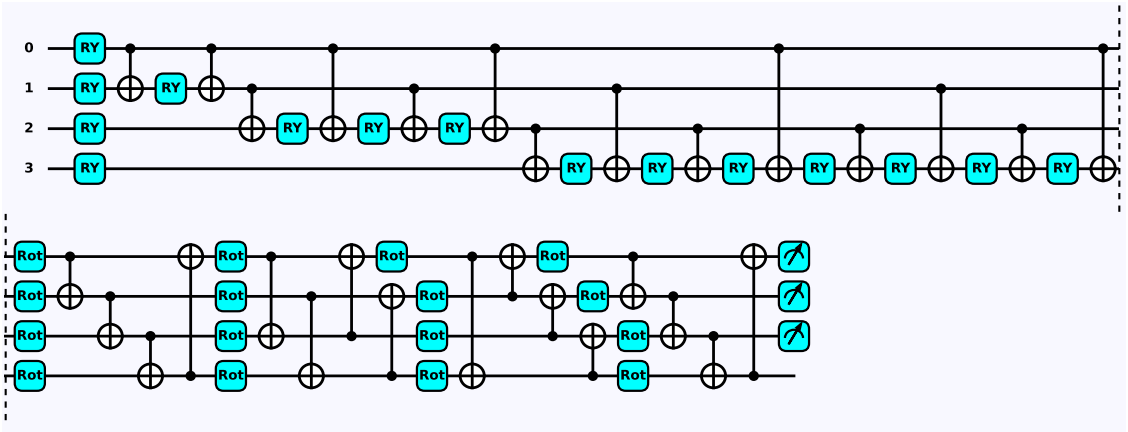


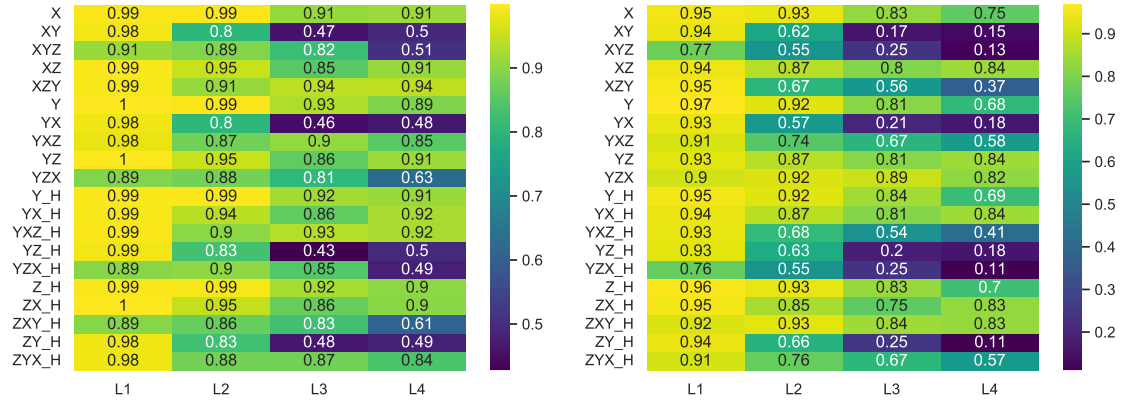
Figure 5.25: Representation of the best VQC model used for classifying if data belongs to class 0 of the Wine dataset. It is characterized by the Amplitude encoding circuit and four parametric layers.

5.8 Classification between classes 0 and 1 of the Wine dataset

The analysis is based on classifying if data belongs to class 0 or 1 of the Wine dataset. The number of qubits and the preprocessing techniques considered depend on the encoding strategies adopted. Also in this case, for Angle Encoding, a Principal Component Analysis is applied to reduce the number of features from thirteen to five, while for Amplitude Encoding, no other preprocessing techniques over feature scaling, and the number of qubits of the system is equal to four.

5.8.1 Support Vector Machine

The results related to the classification of data between class 0 and 1 are shown in Figure 5.26. From the heatmaps, the models with the best performance in the test set can be extracted and compared, evaluating the accuracy in the training set and the depth, as shown in Table 5.6. The best encoding technique, considering the metrics analyzed, is a single-layer circuit that uses RX gates to embed the data, as shown in fig. 5.27.



(a) Training set.

(b) Test set.

Figure 5.26: Report of the accuracy results for Support Vector Machine models for the classification of data belonging to class 0 or 1 of the Wine dataset. In the left, 5.26a, the models evaluated on the training set are shown, while on the right, 5.26b the results related to the test set.

Table 5.6: Best model identified to classify if data belongs to class 0 or 1 of the Wine dataset. The strategies are compared to the accuracy of the training set and the depth. The best circuit is characterized by a single layer with the RX gates for the encoding.

Strategy	Accuracy test	Accuracy train	Depth
RX	0.99	0.99	2
H-RY	0.99	0.99	4
H-RY-RX	0.99	0.99	6
H-RZ	0.99	0.99	4

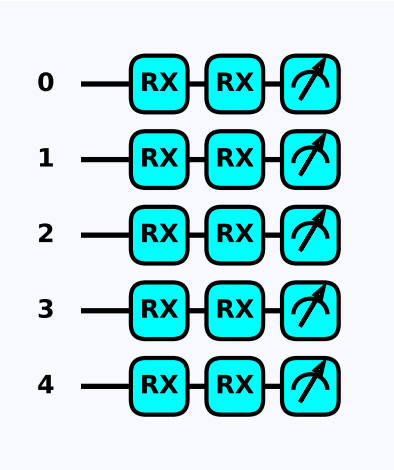
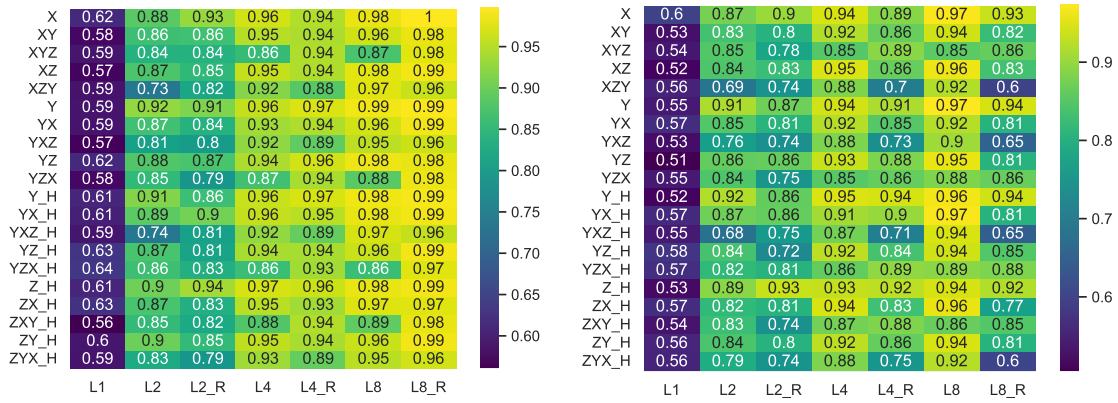


Figure 5.27: Representation of the best kernel circuit to classify if data belongs to class 0 or class 1 of the Wine dataset. The circuit is single-layer and the encoding is done by RX gates.

5.8.2 Variational Quantum Circuit

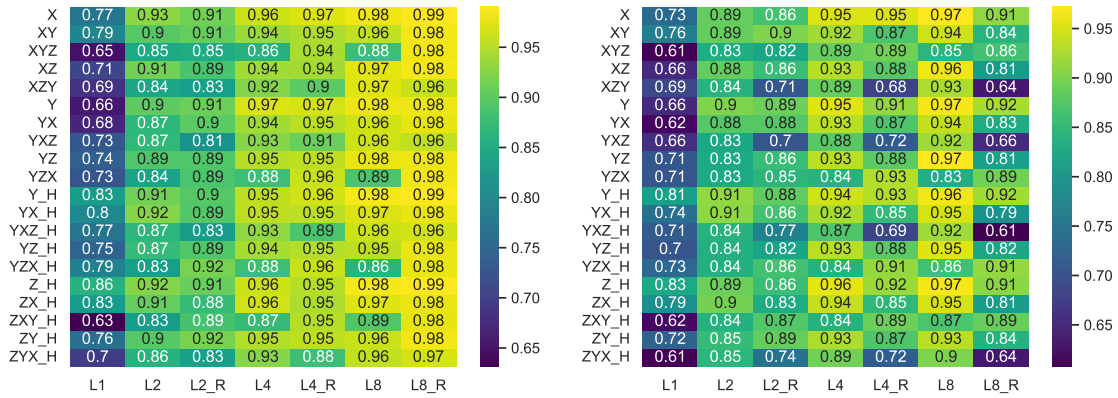
The results of the VQC model for this classification are reported from figs. 5.28 to 5.31. From the results obtained, the models with the better performance on the test set are indicated in Table 5.7 and the best one found is the one that uses RY for encoding the data and eight parametric layers, as shown in fig. 5.32.



(a) Training set.

(b) Test set.

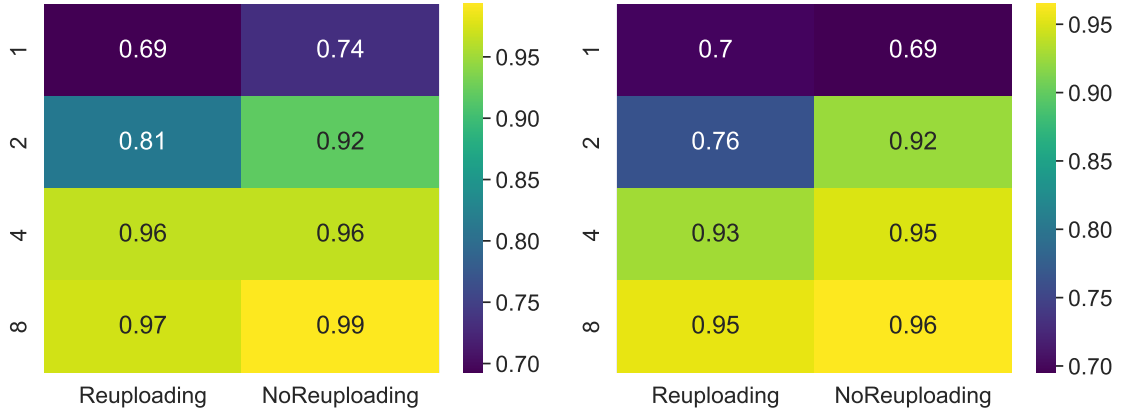
Figure 5.28: Report of the accuracy results for Angle encoding VQC models that classify data between classes 0 and 1 of the Wine dataset. On the left 5.28a, the models evaluated on the training set are shown, while on the right 5.28b, the results related to the test set.



(a) Training set.

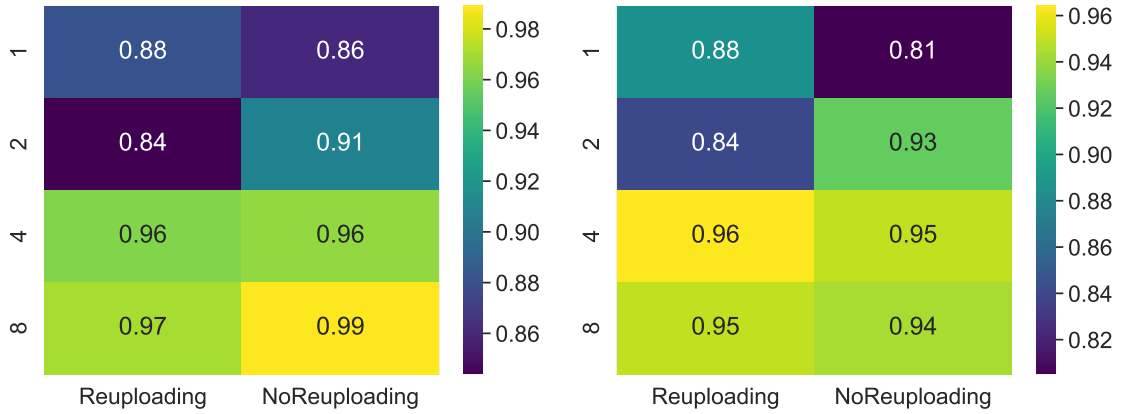
(b) Test set.

Figure 5.29: Report of the accuracy results for Angle encoding VQC models with the entangling block used for classifying data between classes 0 and 1 of the Wine dataset. On the left 5.29a, the models evaluated on the training set are shown, while on the right 5.29b, the results related to the test set.



(a) Training set. (b) Test set.

Figure 5.30: Report of the accuracy results for Amplitude encoding VQC models used for classifying data between classes 0 and 1 of the Wine dataset. On the left 5.30a, the models evaluated on the training set are shown, while on the right 5.30b, the results related to the test set.



(a) Training set. (b) Test set.

Figure 5.31: Report of the accuracy results for Amplitude encoding VQC models with the entangling block used for classifying data between classes 0 and 1 of the Wine dataset. On the left 5.31a, the models evaluated on the training set are shown, while on the right 5.31b, the results related to the test set.

Table 5.7: Best model identified to classify if data belongs to class 0 or 1 of the Wine dataset. In this case, it is chosen as the preferred circuit the one characterized by the RY gates for the encoding and eight parametric layers.

Strategy	Accuracy test	Accuracy train
RX_L8	0.97	0.98
RY_L8	0.97	0.99
H-RY-RX_L8	0.97	0.98
RX_ENT_L8	0.97	0.98
RY_ENT_L8	0.97	0.98
RY-RZ_ENT_L8	0.97	0.98
H-RZ_ENT_L8	0.97	0.98

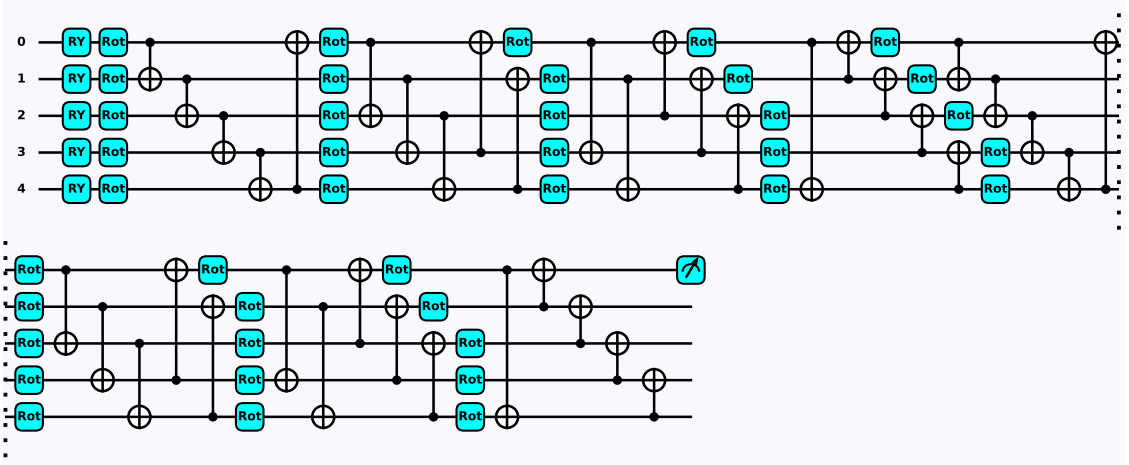
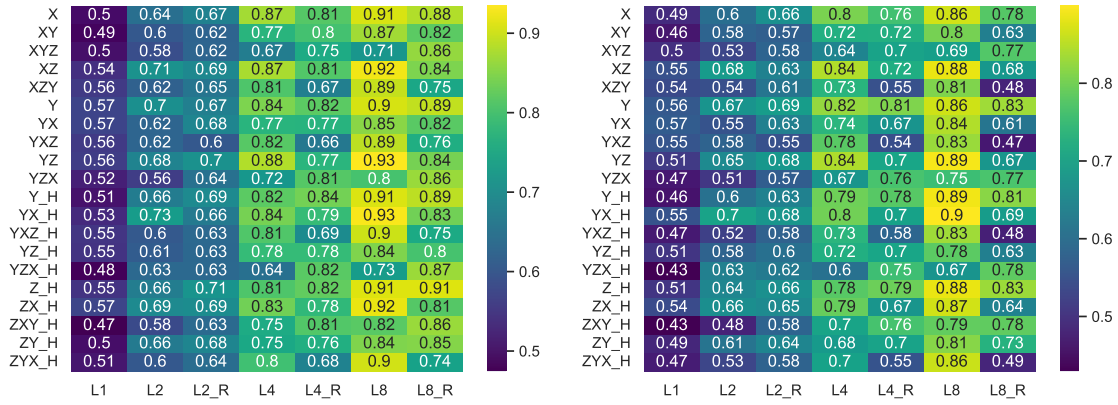


Figure 5.32: Representation of the best VQC model for the classification between classes 1 and 2 of the Wine dataset. It is composed of RY gates to embed data and the rotational gate Rot corresponds to the application of the RZ, RY, and RZ gates.

5.9 Multiclass classification for Wine datset

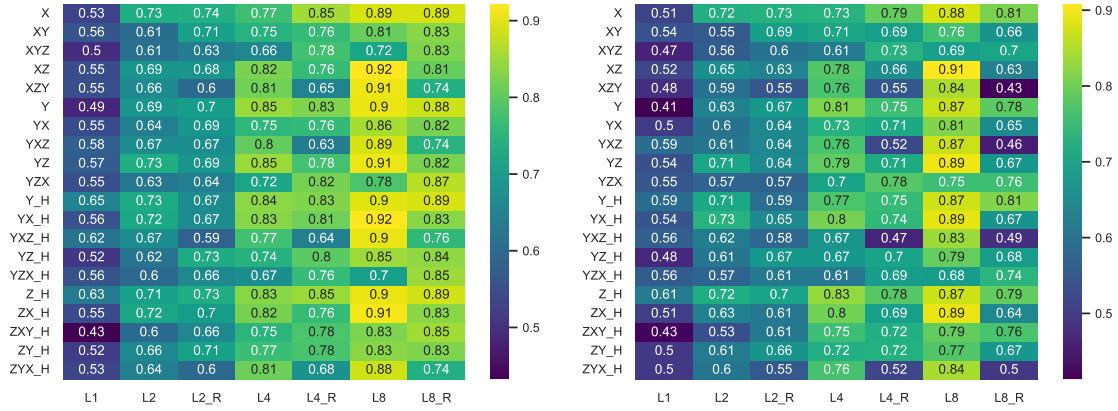
The task of this classification is to identify to which of the three classes data belongs simultaneously. In this case, only the Angle Encoding VQC models are evaluated and the results obtained are reported from figs. 5.33 and 5.34. The best model identified is the one that uses the RX-RZ strategy to encode the data, the entangling block, and eight parametric layers, as represented in fig. 5.35.



(a) Training set.

(b) Test set.

Figure 5.33: Report of the accuracy results for Angle encoding VQC models for multiclass classification of the Wine dataset. On the left 5.33a, the models evaluated on the training set are shown, while on the right 5.33b, the results related to the test set.



(a) Training set.

(b) Test set.

Figure 5.34: Report of the accuracy results for Angle encoding VQC models with the entangling block for multiclass classification of the Wine dataset. On the left 5.34a, the models evaluated on the training set are shown, while on the right 5.34b, the results related to the test set.

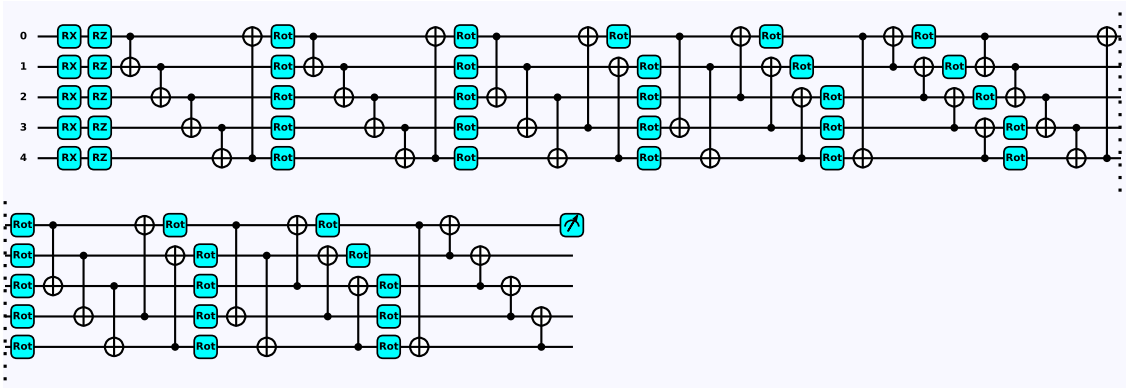


Figure 5.35: Best VQC model for multiclass classification of the Wine dataset. It is composed of RX-RZ gates for the encoding, the entangling block, and eight parametric layers.

5.10 Final Observations

From the results collected, some final considerations can be drawn. As pointed out during the previous analysis on the results obtained the best encoding strategy for both the VQC and SVM models depends on the case study. For this reason, it has to be considered as a hyper-parameter to optimize during the cross-validation step. Another conclusion is related to the models that exploit the re-uploading approach. They suffer from overfitting, considering the high variation from the accuracy of the training and test set. For this reason, regularisation techniques can be adopted to reduce this phenomenon.
In the end, it can be seen that SVM methods permit achieving higher accuracy in predicting the label of data with respect to VQC models.

Chapter 6

Conclusion and Future perspective

The objective of this thesis is to verify the dependence between the accuracy and the encoding strategy adopted in near-term Quantum Machine Learning models. In particular, the Variational Quantum Circuit and the Support Vector Machine with Quantum Kernel Estimation have been taken as a reference for the analyses conducted. The encoding approaches explored are the Angle and Amplitude strategies, which are embedded in more than 280 Variational Quantum Circuits and 80 Support Vector Machines models. These models are evaluated on both the Iris and Wine dataset, executing binary and multiclass classifications.
The results demonstrate that the best encoding strategy depends on the specific case study and, for this reason, it has to be considered as a hyper-parameter to optimize for the choice of the best model. Moreover, some additional considerations can be drawn.

Models that exploit re-uploading suffer from overfitting and, for them, it could be convenient to apply regularisation techniques and reduce the number of parametric layers (this phenomenon becomes more relevant by increasing the number of parameters of the same model).

Comparing the accuracy results of the SVM models to the VQC ones, it can be seen that the former achieve higher accuracy than the others. For this reason, it would be useful to invest in implementing the entire algorithm in Quantum, even if the current quantum devices represent a limit for its development.

The research on this topic has just started. New horizons could be expanded, developing new models or accelerating the existing classical ones by exploiting the quantum advantage. Starting from this thesis, various future works can be foreseen.

First, the number of measurements for the evaluation of the quantum circuit has to be considered for the choice of the best model, because it represents a cost to be optimized as much as possible, especially in near-term devices. Another proposal, related to what concerns this thesis, is to implement new parametric layers for the Variational Quantum Circuits models to improve the classification.

In general, other explorations can be conducted on the Quantum Machine Learning topic. For what concerns the definition of new models, Quantum Generative Adversarial Networks and Quantum Convolutional Networks can be implemented. They are the quantum counterpart of the classical algorithm which tries to be more computationally efficient.
Considering the optimization step, a good approach could be to describe the Machine Learning problem in the QUBO form and optimize it using techniques like the Grover Adaptive Search and Quantum Approximate Optimization Algorithm. In this way, the entire problem will be integrated into Quantum devices.

Machine Learning is a topic that is expanding day by day, involving, for example, domotic and biomedical applications, becoming even more computationally expensive. For this reason, it is looking for a quantum advantage capable of opening the doors to new Machine Learning algorithms with unimaginable capabilities today.

Bibliography

[1] N. S. Yanofsky and M. A. Mannucci, *Quantum Computing for Computer Scientists*. Cambridge University Press, 2008. DOI: 10.1017/CBO9780511813887.

[2] M. Schuld and F. Petruccione, *Supervised learning with quantum computers*. Springer, 2018, vol. 17.

[3] T. M. Mitchell and T. M. Mitchell, *Machine learning*, 9. McGraw-hill New York, 1997, vol. 1.

[4] S. Ruder, *An overview of gradient descent optimization algorithms*, 2016. DOI: 10.48550/ARXIV.1609.04747. [Online]. Available: https://arxiv.org/abs/1609.04747.

[5] H. E. Robbins, "A stochastic approximation method", *Annals of Mathematical Statistics*, vol. 22, pp. 400–407, 2007.

[6] M. Schuld, A. Bocharov, K. M. Svore, and N. Wiebe, "Circuit-centric quantum classifiers", *Physical Review A*, vol. 101, no. 3, Mar. 2020. DOI: 10.1103/physreva.101.032308. [Online]. Available: https://doi.org/10.1103%2Fphysreva.101.032308.

[7] T. Hubregtsen, D. Wierichs, E. Gil-Fuster, P.-J. H. S. Derks, P. K. Faehrmann, and J. J. Meyer, "Training quantum embedding kernels on near-term quantum computers", *Physical Review A*, vol. 106, no. 4, Oct. 2022. DOI: 10.1103/physreva.106.042431. [Online]. Available: https://doi.org/10.1103%2Fphysreva.106.042431.

[8] P. Rebentrost, M. Mohseni, and S. Lloyd, "Quantum support vector machine for big data classification", *Physical Review Letters*, vol. 113, no. 13, Sep. 2014. DOI: 10.1103/physrevlett.113.130503. [Online]. Available: https://doi.org/10.1103%2Fphysrevlett.113.130503.

[9] V. Bergholm, J. Izaac, M. Schuld, *et al.*, *Pennylane: Automatic differentiation of hybrid quantum-classical computations*, 2018. DOI: 10.48550/ARXIV.1811.04968. [Online]. Available: https://arxiv.org/abs/1811.04968.

[10] A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library", in *Advances in Neural Information Processing Systems 32*, Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[11] Martín Abadi, Ashish Agarwal, Paul Barham, *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. [Online]. Available: https://www.tensorflow.org/.

[12] M. Mottonen, J. J. Vartiainen, V. Bergholm, and M. M. Salomaa, "Transformation of quantum states using uniformly controlled rotations", 2004. DOI: 10.48550/ARXIV.QUANT-PH/0407010. [Online]. Available: https://arxiv.org/abs/quant-ph/0407010.

[13] H. Abdi and L. J. Williams, "Principal component analysis", *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[14] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2014. DOI: 10.48550/ARXIV.1412.6980. [Online]. Available: https://arxiv.org/abs/1412.6980.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python", *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[16] *Iris dataset description uci*, [Online] https://archive.ics.uci.edu/ml/datasets/iris, accessed 25-October-2022.

[17] *Wine dataset description uci*, [Online] https://archive.ics.uci.edu/ml/datasets/wine, accessed 25-October-2022.