



UNIVERSITÀ
DEGLI STUDI
DI PADOVA



Dipartimento
di Fisica
e Astronomia
Galileo Galilei

MSC PROGRAMME IN PHYSICS OF DATA

SAVERIO MONACO

STUDY OF QUANTUM CORRELATIONS IN LHCb SIMULATED HEAVY FLAVOUR
JETS

FINAL PROJECT

SUPERVISORS:

PROF. DONATELLA LUCCHESI
DR. DAVIDE ZULIANI
DR. LORENZO SESTINI

ACADEMIC YEAR 2022-2023

Contents

Introduction	1
1 Quantum Computing for High Energy Physics	3
1.1 Introduction to Machine Learning	3
1.2 Quantum Machine Learning	8
1.2.1 Parametrized Quantum Circuit	9
1.2.2 Data encoding	10
1.2.3 Optimization loop	12
1.3 Entanglement	14
1.3.1 Von Neumann Entropy as measure of Entanglement	16
1.3.2 Example: (Parametrized) Bell Circuit	17
2 Jet Physics at LHCb	21
2.1 The LHCb experiment and jet identification	22
2.1.1 The Large Hadron Collider	22
2.1.2 LHCb	22
2.2 b - and c -jets formation	25
2.2.1 Quarks and gluon production	26
2.2.2 Fragmentation	27
2.2.3 Hadronization	27
2.3 Jets reconstruction	28
2.3.1 Particle flow	28
2.3.2 Clustering	28
2.3.3 E-recombination scheme	30

2.3.4	Jet energy correction	30
2.3.5	Secondary Vertex finding	30
3	Jet identification with Quantum Machine Learning	33
3.1	Data-set	33
3.1.1	Secondary Vertex features	34
3.1.2	Data set selection	35
3.2	Classical model: (Gradient) Boosted Decision Trees	37
3.3	Software implementation for jet tagging on quantum models	38
3.3.1	PennyLane	38
3.3.2	JAX	40
3.4	Quantum Models	40
3.5	QML Performance	41
3.5.1	Metrics	42
3.5.2	Scaling with number of qubits N and layers L :	43
3.5.3	Comparison between QPC and BDT	44
4	Benchmarking of entropy production in a quantum classifier	47
4.1	The Barren Plateau problem in QML	48
4.2	Entropy productions	50
4.2.1	Ansatz	52
4.2.2	Parameters initialization	57
4.2.3	Cost function	59
5	Entanglement-based models for b- and c-jet tagging	63
5.1	Characteristic circuits	63
5.1.1	Training of the circuits	64
5.1.2	Classification phase	66
5.2	Feature importance	68
5.3	Correlation from entanglement	71
	Conclusions	77
C.1	Future developments in heavy jet tagging	78
	Bibliography	79
6	Appendix	87
A.1	Quantum Computing	87

Introduction

Quantum Machine Learning (QML) is gaining significant traction as a powerful framework for data analysis, specifically for High Energy Physics (HEP) tasks at the Large Hadron Collider (LHC). Experiments taking data at this collider are starting the so called Run 3 [1], the third period of data collection characterized by a center-of-mass energy of 13.6 TeV and an increased number of collisions that can be produced in a detector per cm^2 and per second (luminosity).

Concurrently, the detectors have been upgraded in order to cope with the accelerator improved performance to increase measurements accuracy and potentially revealing new physics.

However, Run3 data-taking starts posing computational challenges, that will need to be addressed for High Luminosity LHC [2], due to the increased complexity and amount of data generated, necessitating both more sophisticated algorithms and greater computational power for its successful analysis. Quantum Computing, particularly QML, emerges as a potential solution for addressing these demands.

This thesis work focuses on jet flavour tagging, specifically distinguishing between b - and c -quark jets, using QML techniques. Jet tagging is known to be a challenging classification task that could potentially benefit from a quantum implementation. In particular, there may be correlations among the particles forming the jet, that are hidden or washed out by the methods used so far to simulate jets formation. The research starts from the existing literature [3, 4] on jet tagging using new quantum techniques that could help to implement technologies to identify and use new correlations. The methods are here applied to simulated data with a future possibility to compare results with jets collected in actual proton-proton collisions.

The primary technique examined in this study revolves around analyzing the entan-

glement generated within a quantum circuit during its training phase. The hypothesis is that the entanglement measure holds valuable information that can be leveraged to improve the training process on a quantum computer. By delving into the dynamics of entanglement production during the training loop, this research aims to enhance the performance of jet flavour tagging and potentially unlock more accurate and efficient results for this complex classification task.

The outcomes in this work pertaining to jet flavor classification and entanglement study are organized as follows:

- **Chapter 1:** provides an overview of the techniques in quantum computing and QML specifically tailored for data analysis in HEP tasks.
- **Chapter 2:** explores the physics phenomena of jets formation at LHCb. It begins with a brief description of the LHCb detector and the functioning of each sub-detector. The physical processes that produce jets are then explained, from the quarks and gluons production followed then by a description of jet formation with particular attention to b - and c -jets. A chapter subsection is dedicated to jets reconstruction and jet identification (jet tagging) from where the features used in the rest of the work are extracted.
- **Chapter 3:** introduces the data set, providing insights into each feature chosen to characterize the jets. Additionally, it presents and compares both the Classical and Quantum state-of-the-art classifiers for the b - vs c -jets identification.
- **Chapter 4:** presents the findings related to entanglement formation within a quantum classifier, with the goal in mind to distinguish b - from c -jets. Numerous hyper-parameters of the learning models were taken under analysis to investigate their impact on entanglement production.
- **Chapter 5:** showcases two models developed that leverage the information on the generated entanglement. The first model is capable of estimating feature importance, while the second model provides information on data correlation.
- **Conclusions:** summarizes the work done the and draws the major conclusions I can extract from the work performed. In addition the next steps of this work is presented together with possible future applications when hardware with enough qubits will be available.

Quantum Computing for High Energy Physics

The upcoming chapter will explore the methods of Machine learning (ML) on a Quantum Computer [5, 6]. Initially, an introduction to classical ML is provided, exploring the fundamental concepts shared to its quantum counterpart. Subsequently, the optimization procedure of a general quantum model is presented. Lastly, some considerations are made towards the quantum-exclusive phenomenon of entanglement formation, a pivotal factor behind the high expressiveness of quantum models.

1.1 Introduction to Machine Learning

ML is a branch of Artificial Intelligence which involves developing models that are able learn from data to address specific problems without the need to provide an explicit solving algorithm. Its objective is to automate the process of drawing generalizations from experience, enabling the generation of new plausible predictions in unfamiliar scenarios. ML proves particularly powerful in tasks where the relationships within the input data are highly intricate, where it is challenging or even unfeasible to derive analytical algorithms. The trained model, due to its intricate use of data correlation, becomes challenging to interpret in non-trivial cases, leading to its characterization as a *black box*. This property was historically the primary reason for the initial negative perception of ML models in the scientific community. Even today, considerable efforts are invested in developing interpretable machine learning models to address this limitation [7].

ML models fall into three main categories:

- **Supervised Machine Learning:** The primary objective of this branch is to develop models that can learn from labeled data to make accurate predictions on unseen data. In supervised learning, the algorithm is "supervised" during the training phase: the data set contains both the inputs and their respective ground-truths and the model adapts itself in order to match the prediction to its true label. Supervised ML models are primarily utilized for classification and regression tasks, with a notable example being the implementation of sophisticated and rapid GPU triggers for signal-background noise classification [8].
- **Unsupervised Machine Learning:** The end goal of unsupervised machine learning is to discover patterns, structures, or relationships in data. Unlike supervised learning, where the algorithm learns from labeled data, unsupervised learning works with unlabeled data, meaning that the algorithm has no predefined correct answers to learn from. It has found valuable applications in various HEP tasks including clustering and anomaly detection. Notably, models for these two tasks have already found use in enhancing the jet identification pipeline [9]. Moreover, it has enabled the development of numerous generative models. These novel approaches have been proven to be effective in simulating collision events with high accuracy while requiring significantly less simulation time compared to conventional Monte Carlo methods [10].
- **Reinforcement Learning:** The primary goal of reinforcement machine learning is to create an agent model capable of taking decisions within an environment to achieve specific objectives. This framework involves training the model through interactions with the environment, employing a reward/punishment system to guide its learning process. For instance, a reinforcement learning agent has been employed at LHC to address the task of *jet grooming*. This involves improving the purity of jet events derived from collisions [11].

The upcoming part will focus on elucidating the fundamentals of Supervised ML, as it serves as the primary method employed throughout this thesis.

Supervised Machine Learning

The main structure of a generic supervised machine learning algorithm is the following:

1. **Problem definition:** given an input domain \mathcal{X} and an output domain \mathcal{Y} , the training data-set is $\mathcal{D} = \{(x_1, y_1), \dots, (x_M, y_M)\}$ with each element $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$ is the pair of training input (event) and expected output (label). The relationship between each event x_i and the label y_i is unknown and finding the function that best represents this relationship is the goal of the training.
2. **Choice of the model family:** a *model family* is a function

$$y = f(x, \vec{\theta}) \tag{1.1}$$

with $x \in \mathcal{X}$, $y \in \mathcal{Y}$ and $\vec{\theta} = \{\theta_1, \dots, \theta_D\}$ is a set of parameters that will be optimized during the training phase such that the errors during prediction are minimized.

3. **Training of the model:** the model is trained by fitting the parameters and the hyper-parameters to the data \mathcal{D} . A test data set is then used to validate the performance of the model on new, unseen data.
4. **Employment of the model:** once trained, the model can be used to predict output values.

The training phase holds utmost importance in a ML algorithm. Its objective is to meticulously adjust the parameters $\vec{\theta}$ of the model in order to minimize an error function defined as **loss function** $L(\vec{\theta})$ that quantifies the discrepancy between the model's predictions and the true labels. A model achieving a low (minimized) error indicates a strong alignment with the input data.

Different choices of the loss function lead to different training strategies and results. For a regression task, the most widely used is the *mean squared error* loss defined as:

$$L^{MSE}(\vec{\theta}) = \frac{1}{M} \sum_{i=1}^M \left(f(x^i, \vec{\theta}) - y^i \right)^2 \tag{1.2}$$

While for binary (and multi-class) classification problems, the most common loss function is the *cross-entropy* loss, defined as:

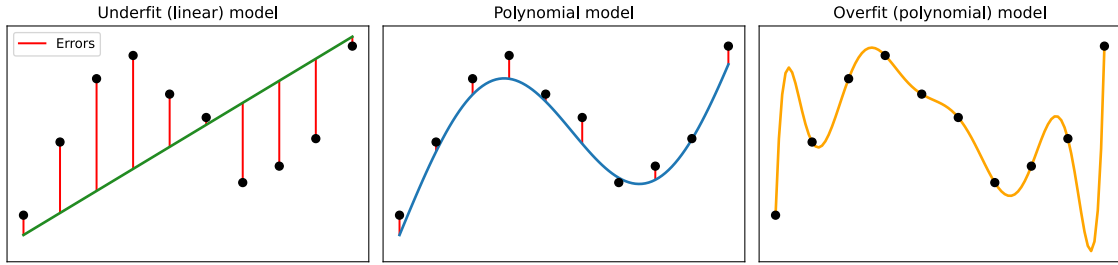


Figure 1.1: Performances of three different models on the same input data. On the left, a linear model which results in a high error, in the middle a polynomial model with little training error, on the right an *overfit* model with no error over the training set.

$$L^{CE}(\vec{\theta}) = - \sum_{i=1}^M y^i \log p^i + (1 - y^i) \log(1 - p^i) \quad (1.3)$$

Where $p^i = f(x^i, \vec{\theta})$ are the outputs of the model, which are assumed to be probabilities of belonging to a given class.

The loss function $L(\vec{\theta})$ has to be minimised with respect to the parameters of the model to obtain the optimal parameters configuration $\vec{\theta}^*$:

$$\vec{\theta}^* = \arg \min_{\theta} L(\theta) \quad (1.4)$$

In equation 1.4, the training objective is defined as finding the parameters that lead to the lowest value of the loss function. However, an essential consideration must be made regarding the choice of the model family. If the selected model is overly complex, it may achieve a very low error during training, but when used on unseen data, it can lead to a high prediction error. Thus, while minimizing the loss function is the primary goal of training, achieving a low error on the training set does not necessarily guarantee a good model that will generalize well to new data. An overly simplistic model exhibits limited expressibility, potentially failing to capture the complex tendencies the data (as shown in the left plot in Figure 1.1). On the other hand, excessively complex models can result in overfitting (as illustrated in the right plot of Figure 1.1), where the model memorizes the training data rather than

extracting meaningful patterns applicable to unseen data. Thus, achieving a balance between model complexity and its ability to generalize is important in developing effective ML models, as in the scenario depicted in the middle plot of Figure 1.1.

Gradient descent: Finding the minimum of the loss function analytically becomes impractical due to the high dimensionality of its space, which depends on the number of model parameters and can be very high, even reaching millions. To address this, gradient descent offers a smart approach. Its main goal is to iteratively adjust the model's parameters $\vec{\theta}$ following the direction of the steepest descent (opposite to the gradient) until it reaches a local or global minimum. By doing so, gradient descent efficiently seeks the optimal point without requiring an explicit analytical solution. The compact formula for gradient descent is:

$$\vec{\theta}^{(t+1)} = \vec{\theta}^{(t)} - \eta \nabla L(\vec{\theta}^{(t)}) \quad (1.5)$$

Where $\vec{\theta}^{(t+1)}$ represents the updated parameters at step $t+1$, $\vec{\theta}^{(t)}$ denotes the parameters at iteration t . The gradient vector of the loss function at iteration t , denoted as $\nabla L(\vec{\theta}^{(t)})$, indicates the direction of the descent, and η stands for the coefficient determining the magnitude of change at each step, referred to as the *learning rate*. A typical optimization procedure through gradient descent is described through pseudo-code in the Algorithm block 1.1.

Even after careful tuning of the learning rate parameter (η), plain gradient descent might not be enough because of flat regions and local minima present in the loss function (as in figure 1.2) which can limit the effectiveness of this approach.

To mitigate these effects, several methods have been developed from the naive gradient descent, in this work the ADAM [13] optimizer has been used.

The main features of the ADAM optimizer are:

- **Adaptive Learning Rate:** ADAM uses adaptive learning rates for each parameter. It calculates individual learning rates based on the historical gradients of each parameter. This helps to adjust the step size for each parameter, providing faster convergence for different features.
- **Momentum:** ADAM incorporates the concept of momentum. Instead of relying solely on the current gradient value, momentum considers the accumulated

Algorithm 1.1: Pseudo-code for Gradient Descent

Function Gradient Descent:

Initialization: choose initial values for the parameters $\vec{\theta}$

for $epoch = 1 \dots n_epochs$ **do**

Compute the gradient: Compute the gradient of the loss function $L(\vec{\theta})$ relative to each parameter θ_i : $\partial L / \partial \theta_i$.

 The gradient represents the direction of the steepest increase in the function and it represents the change of the loss function upon small variations in each parameter.

Update the parameters: Each parameter is updated by subtracting a fraction of the gradient from the current parameter values. The fraction is called learning rate:

$$\theta_i = \theta_i - \eta * \partial L / \partial \theta_i$$

end

effect of past gradients to guide the updates to the model's parameters which it helps both in escaping from local minima (thus premature optimization) and escaping flat regions of the loss landscape, which gradients have a value close to zero.

1.2 Quantum Machine Learning

QML has emerged as a promising paradigm for data analysis merging Quantum Computing (QC) and ML, offering novel approaches to address complex challenges in research fields as for example in HEP at particle colliders. With the potential to exploit quantum properties, such as superposition and entanglement, QML holds the promise of enhancing analysis and computational capabilities crucial to HEP experiments.

These novel models incorporate the essential techniques and concepts from their classical counterpart, as explained in the previous section, within a framework governed by the principles of quantum mechanics.

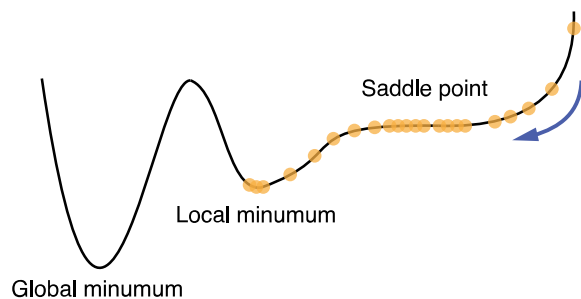


Figure 1.2: During gradient descent, various scenarios can occur in the optimization process. Saddle points are locations in the landscape where the gradient value is low. Local minima represent situations that often lead to premature optimization, causing the model’s parameters to become trapped in these points and preventing further updates. [12]

1.2.1 Parametrized Quantum Circuit

Every QML model is structured as a Parametrized Quantum Circuit (PQC) which functions as the graphical representation of the parametrized model. An instance of this circuit is depicted in Figure 1.3. It visually portrays the evolution of a system of spin particles, or qubits, with each horizontal line—also referred to as a wire—representing one of the spins. The x-axis signifies time, where actions to the right occur after those on the left.

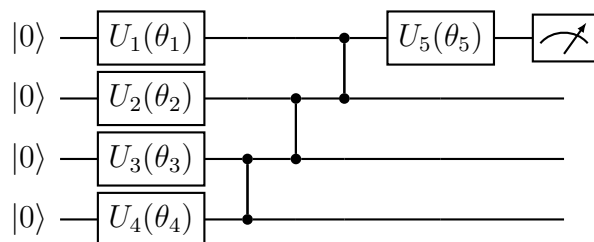


Figure 1.3: Example of a parametrized quantum circuit with 4 number of qubits.

The initial (leftmost) state is referred to as $|0\rangle^{\otimes N}$, where N is the number of qubits. This state serves as the starting point for every quantum circuit. In practical implementations, it signifies that each qubit is initialized to its ground state.

The circuit is then composed of blocks, representing actions applied to specific qubits denoted as gates, while elements spanning multiple wires denote multi-qubit gates. The former constitute the parametrized elements that enable model adaptation, while the latter are responsible for generating entanglement.

The output of the circuit is obtained through the measurement of the wavefunction and it is depicted by the gate .

The optimization process, namely the update of the parameters, is executed through the aid of a classical computer which obtain the gradients through multiple measurements and sets the new angles for the parameterized rotations. The following section will explore the initial segment of the quantum circuit, the embedding block.

1.2.2 Data encoding

In QML, the process of *embedding* involves loading classical data, which the model uses as input for learning and make predictions, into the quantum circuit as a quantum state. Conventionally, the initial state of a quantum circuit is $|0\rangle^{\otimes N}$ (the state in which all N qubits are in the state $|0\rangle$), which then it is evolved by a parametrized circuit block \mathcal{F} . The parameters in \mathcal{F} differ from the trainable parameters since they are used to represent an input event. To input a single event \vec{x}_i into a circuit, the state $|0\rangle^{\otimes N}$ is evolved into a state representing that event $|\vec{x}_i\rangle$ through the embedding block $\mathcal{F}(\vec{x}_i)$, where the parameters correspond to the values of the features of the input events. This feature map takes a classical data point \vec{x}_i and converts it into a set of gate parameters in a quantum circuit, resulting in a quantum state $|\vec{x}_i\rangle$:

$$\vec{x}_i \xrightarrow{\text{Embedding}} |\vec{x}_i\rangle \quad (1.6)$$

Through the process of embedding, a d -dimensional input vector (defined in the space \mathbb{R}^d , is mapped into the significantly larger Hilbert space \mathcal{H} . Proper selection of the *quantum embedding process* can result in feature maps that exhibit a greater expressiveness compared to their classical counterparts, leading to advantages in classifications in selected scenarios [6, 14].

The typical strategies for embedding classical data into a quantum circuit are:

Angle Embedding: The angle embedding is the most straightforward approach to input classical data into a quantum circuit. This method requires a circuit with the number of qubits matching the dimension of the input vector. It revolves around applying rotations on the x-axis or y-axis, utilizing parametrized RX or RY gates, respectively. The angle of these rotations is determined by the values present in the feature vector, thus establishing the quantum representation of the classical data.

The operation of angle encoding is then the following:

$$|\psi_{x_i}\rangle = \bigotimes_{k=1}^N R(x_i[k])|0^k\rangle \quad (1.7)$$

where R can be either RX or RY and $x_i[k]$ is the k -th element of the vector \vec{x}_i . An example of angle encoding for $\vec{x}_i = (0.2, 1.5, 3.1)$ is shown in Figure 1.4.

Rotational gates are periodic unitary operator with respect to the parameter θ : $R(\theta) = R(\theta + 4\pi)$. Therefore, each feature value must be scaled to the $[0, 4\pi]$ range.

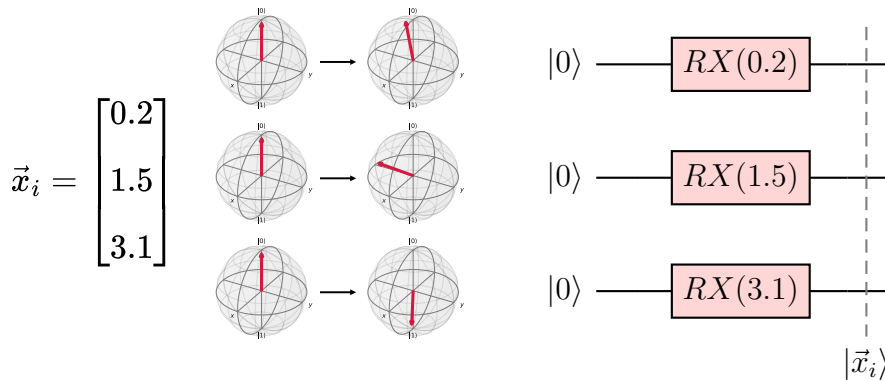


Figure 1.4: On the left, the action of the angle embedding on the $|0\rangle^{\otimes 3}$ state for the vector $\vec{x}_i = (0.2, 1.5, 3.1)$. On the right the graphical representation of the angle embedding circuit.

Amplitude encoding: an alternative and widely used approach for data encoding within a quantum circuit is known as amplitude encoding. In this method, a normalized N -dimensional vector is encoded into a circuit comprising n qubits, where $n = \log_2(N)$. By employing this scheme, the amplitudes of the quantum states

are manipulated to represent the data, providing an efficient and compact quantum representation of the classical information.

For instance we define a map which represents the state $|\psi_{x_i}\rangle$ such that:

$$|\psi_{x_i}\rangle = \sum_{k=1}^N x_i[k]|k\rangle \quad (1.8)$$

where $|k\rangle$ is the k -th computational basis state (namely $\{|k\rangle\} \equiv \{|0\dots00\rangle, |0\dots01\rangle, |0\dots10\rangle \dots |1\dots11\rangle\}$)

As an example, the vector $x_i = [0.2, 1.5, 3.1]$ can be represented in just 2 qubits through amplitude encoding such that the embedded state is:

$$\vec{x}_i = \begin{bmatrix} 0.2 \\ 1.5 \\ 3.1 \end{bmatrix} \Rightarrow |\psi_{x_i}\rangle = 0.2 * |00\rangle + 1.5 * |01\rangle + 3.1 * |10\rangle + 0.0 * |11\rangle \quad (1.9)$$

The main advantage of amplitude embedding over angle embedding is that it requires significantly fewer qubits (exponentially less) to represent the same amount of data.

However, in this study, Angle embedding will be the primary choice due to its unique property of dedicating each wire to a specific feature. This property will be particularly advantageous for the entanglement-based models presented in chapters 4 and 5.

1.2.3 Optimization loop

Variational circuits: Variational circuits play a central role in QML. These circuits consist of single-qubit gates, such as parametrized gates like RX, RY, and RZ, which offer flexibility and trainability through the optimization of their parameters. Additionally, variational circuits incorporate multiple-qubit gates, which are typically non-parametrized. These multiple qubit gates are crucial for establishing intricate relationships between qubits through entanglement, enabling the solution of complex tasks in quantum machine learning.

A general Variational Quantum Circuit (VQC) is made of the following parts as represented in Figure 1.5:

- **Feature Map:** An Embedding block is positioned before the variational part in the quantum circuit. Its purpose is to transform the initial state $|0\rangle^{\otimes N}$ into

a state that effectively represents the given input vector:

$$|0\rangle^{\otimes N} \rightarrow |\vec{x}_i\rangle = \mathcal{F}(\vec{x}_i)|0\rangle^{\otimes N}$$

- **Variational gates:** The state is then evolved further through a complex unitary operation represented by $U(\vec{\theta})$. This unitary operation incorporates the variational parameters $\vec{\theta}$.
- **Measurement gate(s):** Depending on the selected loss function, there are two approaches:
 - Global Measurement: In this approach, all the qubits are measured collectively.
 - Local measurement: Alternatively, the measurement can be performed only on a subset of all the qubits. This approach is referred to as a local cost function. As highlighted in later sections, local cost functions can exhibit greater robustness against noise and may yield better-behaved gradients, hence better training.

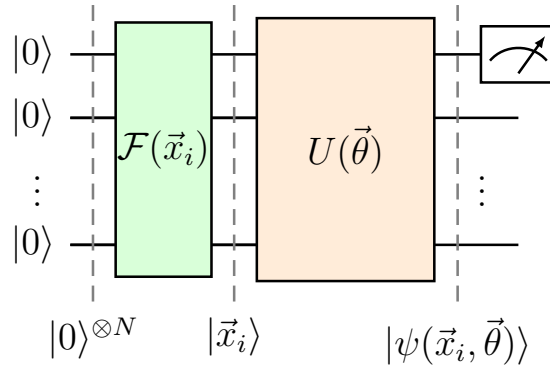


Figure 1.5: General QML circuit, the first block represents the embedding, the light yellow block $U(\vec{\theta})$ is the variational part, then the wavefunction is measured through local measurement.

Computation of the gradient: A crucial element for any QML model implementation is the integration of a gradient descent algorithm. The computation of partial derivatives of the loss function with respect to each parameter can be entirely carried out directly on a quantum circuit using the *parameter shift rule*, which is a

method of gradient computation exclusive to quantum computers.

Keeping in mind that $L(\vec{\theta})$ is the loss function, where $\vec{\theta}$ are the current parameters of the model, the parameter shift rule is given by the equation:

$$\frac{\partial L(\vec{\theta})}{\partial \theta_i} = \frac{1}{2} \left[L\left(\vec{\theta} + \frac{\pi}{2} \cdot \vec{e}_i\right) - L\left(\vec{\theta} - \frac{\pi}{2} \cdot \vec{e}_i\right) \right] \quad (1.10)$$

This rule enables the calculation of exact gradients with just two evaluations per parameter. By shifting the given parameter by $+\frac{\pi}{2}$ and $-\frac{\pi}{2}$. Therefore, to calculate the gradient for the parameter θ_i , it is necessary to evaluate the loss of the quantum circuit with the parameter $\vec{\theta}$ while first shifting the i -th parameter θ_i to $+\frac{\pi}{2}$ and then to $-\frac{\pi}{2}$. The optimization routine for each parameter involves evaluating the two expectation values using the parameter shift rule using the Quantum Hardware, then calculating the difference, and subsequently updating the parameters using a classical optimizer as in Figure 1.6.

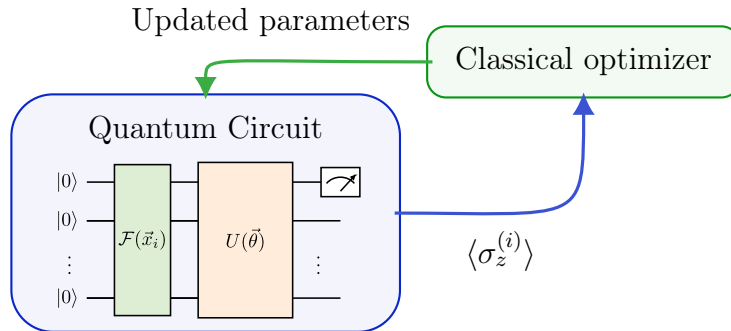


Figure 1.6: Optimization routine, multiple shots are performed within a quantum circuit to compute the gradient for each parameter, then the parameters are updated through the computation of a classical computer.

1.3 Entanglement

Entanglement is a fundamental phenomenon that plays a crucial role in quantum mechanics. It refers to the non-classical correlations that can exist between the quantum states of two or more particles. When two or more particles get entangled, their states become inextricably linked, behaving as one single system.

When a composite quantum system is entangled, its overall quantum state cannot be expressed as a simple product of individual states of its constituent parts.

To provide more clarity, two examples will be presented, one involving an entangled state, and the other concerning an unentangled state:

- **Entangled State (Bell State):**

The Bell state is a maximally entangled state between two qubits and is given by:

$$|\psi^{\text{Bell}}\rangle = \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle) \quad (1.11)$$

Here, $|00\rangle$ represents both first and the second qubit being in the state $|0\rangle$. Similarly, $|11\rangle$ represents both qubits being in the state $|1\rangle$. The coefficient $\frac{1}{\sqrt{2}}$ ensures that the state is properly normalized.

This state cannot be represented as a product of individual states. If it were separable, we could write it as $|\psi^{\text{Bell}}\rangle = |q_1\rangle \otimes |q_2\rangle$, where $|q_1\rangle$ and $|q_2\rangle$ are individual states of the first and second qubit, respectively. However, it is impossible to find such states $|q_1\rangle$ and $|q_2\rangle$ that satisfy this condition for the Bell state $|\psi^{\text{Bell}}\rangle$.

- **Unentangled State (Separable State):**

A simple unentangled state can be given by the following product state:

$$|\psi^{\text{sep}}\rangle = \frac{1}{2} ((|0\rangle + |1\rangle) \otimes (|0\rangle - |1\rangle)) \quad (1.12)$$

In this case, the state $|\psi^{\text{sep}}\rangle$ is separable because it can be expressed as the product of individual qubit states. The first qubit is in the state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) := |+\rangle$ and the second qubit is in the state $\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) := |-\rangle$.

As previously indicated, qubit entanglement plays a fundamental role in generating intricate behaviors among qubits, enabling the development of highly expressive models.

Furthermore, entanglement between qubits also represents a form of shared information. Analyzing entanglement production within a quantum circuit can provide valuable insights into the input data and the classification task, as discussed in Chapter 5.

1.3.1 Von Neumann Entropy as measure of Entanglement

Von Neumann Entropy [15] is a quantity named after John Von Neumann, one of the pioneers of quantum mechanics. It is a concept borrowed from the field of classical information theory with its relative counterpart, the Shannon Entropy [16] and it is used to quantify the amount of uncertainty or information content of a random variable.

In the context of entanglement, the Von Neumann Entropy is a measure of quantum entanglement shared between two bipartitions of a quantum state.

Computation of the Von Neumann Entropy: Von Neumann Entropy between two bipartitions A and B is computed starting from the the density matrix (or density operator) ρ :

$$\rho = |\psi\rangle\langle\psi| \tag{1.13}$$

where $|\psi\rangle$ is the wavefunction representing the global state.

Given the two bipartitions of the system, it is possible to marginalize one of the two to obtain what is called the *reduced density matrix*. The reduced density matrix describes the state of that specific subsystem by marginalizing components of the other subsystem.

The formula to compute the Von Neumann entropy between two quantum bipartitions A and B is defined as follows:

$$S(\rho_A) = -\text{Tr}(\rho_A \log(\rho_A)) \tag{1.14}$$

where ρ_A is the reduced density matrix of subsystem A , obtained by tracing out the degrees of freedom of subsystem B from the joint density matrix of the composite system. In this context, the logarithm can be computed using either base 2 or e . However, throughout this work, the natural logarithm \log_e will be utilized unless otherwise specified.

Similarly, we can calculate the Von Neumann entropy $S(\rho_B)$ for subsystem B by taking the reduced density matrix ρ_B obtained from the joint density matrix by tracing out the degrees of freedom of subsystem A .

Properties:

- $S(\rho)$ is zero if and only if ρ represents a separable state. By definition, a separable state does not have entanglement.
- Given the two bipartitions A and B , $S(\rho_A) = S(\rho_B)$, in other words, the amount of entanglement obtained by tracing out respectively the system B and A are the same.

1.3.2 Example: (Parametrized) Bell Circuit

The Parametrized Bell Circuit offers a straightforward illustration of entanglement between qubits as shared information. This circuit relies solely on a single parameter, θ :

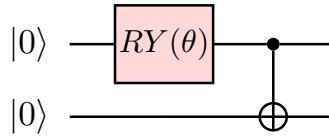


Figure 1.7: Circuit that outputs the parametrized Bell State $|\psi^{\text{Bell}}(\theta)\rangle$.

The unitary representation of the circuit ($\hat{B}(\theta)$) is

$$\hat{B}(\theta) = (CNOT)(RY(\theta) \otimes \mathbb{I}) \tag{1.15}$$

Which applied to the initial state $|00\rangle$, it outputs the desired evolved state:

$$\begin{aligned} \hat{B}(\theta)|00\rangle &= \begin{bmatrix} \cos(\theta/2) & 0 & \sin(\theta/2) & 0 \\ 0 & \cos(\theta/2) & 0 & \sin(\theta/2) \\ 0 & -\sin(\theta/2) & 0 & \cos(\theta/2) \\ -\sin(\theta/2) & 0 & \cos(\theta/2) & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} \cos(\theta/2) \\ 0 \\ 0 \\ -\sin(\theta/2) \end{bmatrix} = \cos(\theta/2)|00\rangle - \sin(\theta/2)|11\rangle \end{aligned} \tag{1.16}$$

1.3. ENTANGLEMENT

The reduced density matrix $\tilde{\rho}(\theta)$ of the state $|\psi^{\text{Bell}}(\theta)\rangle$ is:

$$\tilde{\rho}(\theta) = \begin{bmatrix} \cos^2(\theta/2) & 0 \\ 0 & \sin^2(\theta/2) \end{bmatrix} \quad (1.17)$$

From which it is possible to compute the Von Neumann Entropy of the state:

$$\begin{aligned} S(\theta) &= -\text{Tr}[\tilde{\rho}(\theta) \log(\tilde{\rho}(\theta))] = \\ &= -\cos^2(\theta/2) * \log(\cos^2(\theta/2)) - \sin^2(\theta/2) * \log(\sin^2(\theta/2)) \end{aligned} \quad (1.18)$$

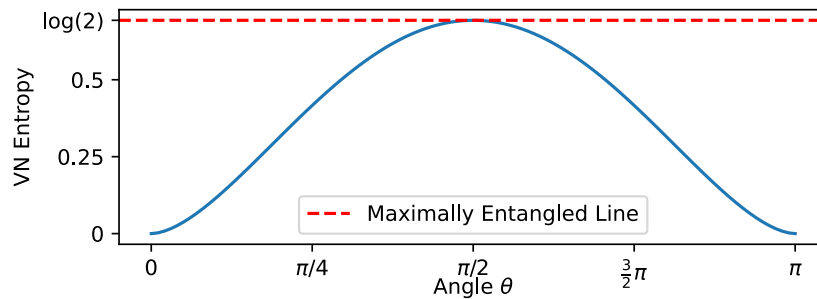


Figure 1.8: Value of Von Neumann Entropy for $\theta \in [0, \pi]$.

The Von Neumann entropy of the circuit shown is dependent on the parameter θ . Figure 1.8 illustrates the entropy (quantifying the amount of entanglement) concerning various values of θ within the range $[0, \pi]$. Notably, as θ approaches $\pi/2$, the entropy increases, reaching its maximum for the Bell state with maximum entanglement. Conversely, the entropy returns to 0 at $\theta = \pi$.

Three distinct noteworthy cases can be identified:

- When $\theta = 0$, the state is simply $|00\rangle$, displaying no entanglement. From an information theory perspective, measuring one qubit provides no useful information about the other, as the result is always 0.
- At $\theta = \pi$, the state is $-|11\rangle$, which is equivalent to the $\theta = 0$ case, no information of the other qubit is gained upon measuring one.
- For $\theta = \pi/2$, the state is the bell state with maximum entanglement. From an information theory standpoint, maximum entanglement implies the most information is conveyed upon measurement. When measuring one qubit, the

other collapses into the same state, providing a complete description of the global state.

The techniques outlined in this chapter constitute the fundamental components of data analysis employing QML models for HEP. This framework is progressively gaining ground as a possible successor to conventional ML, showing remarkable potential for advancing data analysis in this field.

The integration of ML in the field of HEP has resulted in significant boosts in the efficiency of the experiments. This resulted not only in enhanced data analysis, but it has also proven highly beneficial for tasks such as calibrating various modules of the experiments for better performances [17]. ML results effective primarily thanks to its property of being able to tackle problems that do not have an analytical definition, which it can be assumed to be the majority of tasks faced nowadays in HEP.

Fundamentally, even with its exclusive characteristics, QML can be applied to the same problem domains currently addressed by conventional ML algorithms. Just as Quantum Computation showcased superior performance for specific tasks such as *factorization* [18], QML could serve as the solution to address tasks that proved to be challenging even for current ML models.

Furthermore, as Quantum Sensing [19] technology advances and more robust quantum hardware is developed, the true capabilities of quantum models will become more and more prominent.

One of the tasks within the realm of HEP that experienced a substantial improvement in efficiency through the incorporation of ML is the classification of jet events. However, even with the implementation of these techniques, the results are not yet fully optimal. The upcoming chapter will concentrate on introducing this particular task which will be the focus of the thesis. Additionally depiction of the LHCb experiment is provided, as it stands as the most suitable detector to date for investigating jet phenomena.

Chapter 2

Jet Physics at LHCb

The phenomena studied in this thesis work regard the formation of jets subsequent to proton collisions. These jets, which manifest as clusters of particles, can emerge in tandem with important physics events such as the Higgs decay $H \rightarrow b\bar{b}$ and $H \rightarrow c\bar{c}$.

The ML techniques presented in Chapter 1 have already been employed in the realm of jet tagging, yielding results surpassing those achieved by conventional algorithms [4, 20]. Given the intricate nature of jet formation and the amount of information embedded in a single event, the integration of a quantum variant represents an attempt to amplify even further the classification efficiency and purity, i.e. the capability to correctly identify the flavour of the quark originating the jet.

The forthcoming chapter will explore the architecture of the LHCb experiment, elucidating the functioning of all the sub-detectors used for the data collection. Subsequently, it will explore the underlying physics governing the formations of the studied events. Lastly, the focus will shift to the algorithms that process the raw detectors information into high level features information like track/jets position, momentum and energy.

2.1 The LHCb experiment and jet identification

2.1.1 The Large Hadron Collider

The Large Hadron Collider (LHC) [21], is the world's largest and most powerful particle accelerator, with a nominal center-of-mass energy of 14 TeV and stretching about 27 km in length [22]. Within the LHC, two beams of either protons or heavy ions circulate in separate beam pipes, where a vacuum is maintained in the order of 10^{-10} to 10^{-11} mbar. These beams intersect at four interaction points housing the main experiments. Figure 2.1 shows the CERN accelerators complex with the LHC machine, the Super Proton Synchrotron (SPS), the Proton Synchrotron (PS), and other extracted beams used for fixed target experiments and or test beams facilities. Protons are produced by ionizing hydrogen atoms, then they undergo several pre-acceleration stages in the synchrotrons. After this series of accelerations, the protons are injected into the collider, where each beam can reach a final energy of approximately 7 TeV, resulting in a center-of-mass energy at the collision of approximately $\sqrt{s} \approx 14$ TeV.

The LHC accommodates four main experiments at its four collision points: ATLAS (A Toroidal LHC ApparatuS) [23] and CMS (Compact Muon Solenoid) [24] serve as the two General Purpose Detectors with a cylindrical structure surrounding the collision points. ALICE (A Lead Ions Collision Experiment) [25] focuses on the study of phases of matter where quarks and gluons are free, particularly in heavy ions collisions. Lastly, LHCb is dedicated to studying the properties of b - and c -quarks, which are the subject of investigation and discussion in this thesis work.

2.1.2 LHCb

In this section, a more in-depth description of the LHCb detector is done, as the thesis work involves analyzing Monte Carlo data of this experiment.

LHCb is a single-arm spectrometer with a forward pseudo-rapidity η^1 coverage between 1.8 and 4.9. It uses a coordinate system with the z -axis parallel to the beam direction, the y -axis opposite to gravity, and the x -axis perpendicular to the yz -plane. Starting from the interaction point the main sub-detectors are:

¹The pseudo-rapidity, denoted as η , is defined as $\eta = -\ln \tan \theta/2$, where θ represents the polar angle formed by the particle momentum with respect to the beam axis.

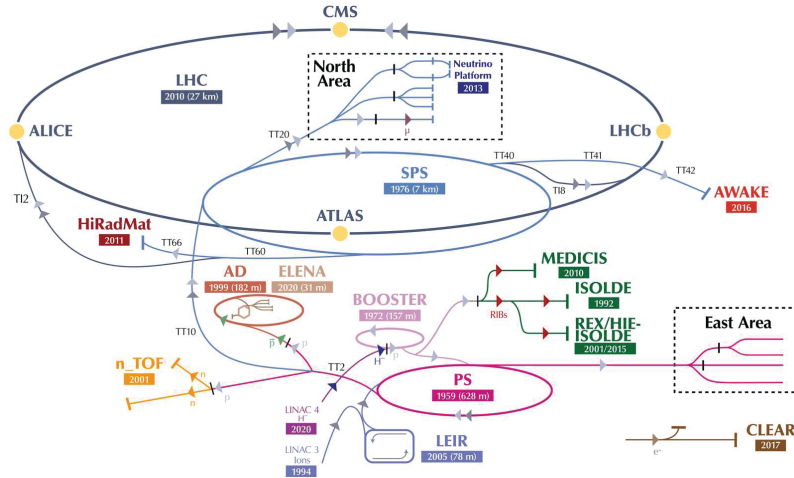


Figure 2.1: Schematic representation of accelerator facilities at CERN. The entire infrastructure depicted is interconnected, incorporating various facilities like PS, SPS responsible for accelerating particles from a source before injecting them into the LHC and other experiments including ELENA (utilized for trapping antiprotons) and AWAKE (employing plasma to accelerate particles). [26]

- **Vertex Locator [VELO]:** The VELO measures track coordinates near the interaction point. It provides precise information enabling the identification of secondary vertices from b - and c -hadron decays.
- **Tracking stations [TT, T1, T2, T3]:** Four planar stations positioned perpendicular to the beam axis. One of them, the TT, is positioned upstream of the dipole magnet (see definition in the last bullet) and the remaining T1-3 stations are positioned downstream of the magnet. The primary purpose of the tracking system is to accurately reconstruct charged-particle tracks, enabling the determination of their momenta.
- **Ring Imaging Cherenkov [RICH1, RICH2]:** The RICH detectors allow to distinguish between charged hadrons (π, K, p). Moreover, the RICH system can also help to identify charged leptons (e, μ), providing additional information to complement the data from downstream calorimeters and the muon system.

The fundamental principle underlying the RICH system relies on the Cherenkov effect. This effect takes place when charged particles traverse a transparent medium at a velocity higher than the phase velocity of light in that medium.

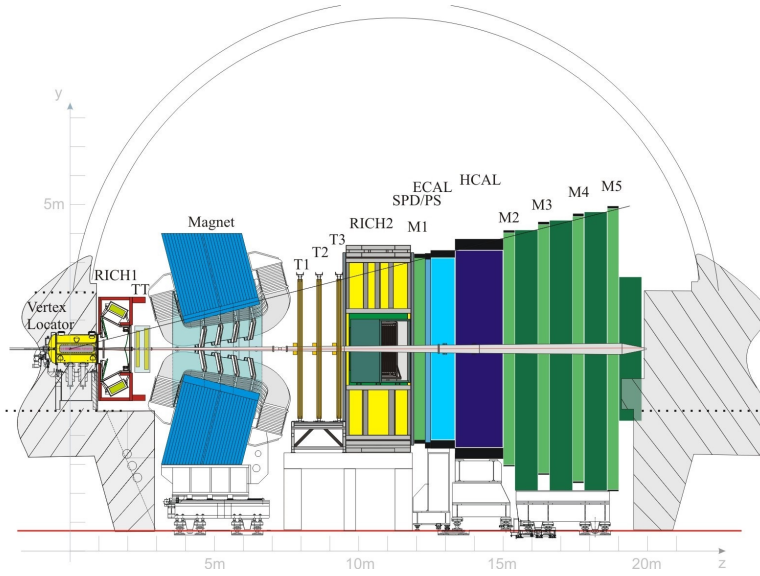


Figure 2.2: Structure of the LHCb detector. [27]

As a result, the charged particles emit Cherenkov radiation, which forms the basis for particle identification in the RICH detector.

- **Scintillator Pad Detector and Preshower [SPD/PS]:**

- The primary role of the SPD is to provide precise and fast information for real time event acquisition decision (trigger) . It is designed to detect charged particles passing through its scintillator pads, which are plastic materials that emit light when charged particles traverse them. By quickly identifying charged particles, the SPD helps initiate the data readout process for the full detector when interesting collision events occur, ensuring that relevant data is not missed.
- The PS stands for "Preshower Detector." It is another sub-detector of LHCb, positioned downstream of the SPD. The main function of the PS is to enhance the particle identification capabilities of the LHCb detector, particularly for electrons and photons

- **Calorimeters [ECAL, HCAL]:** ECAL, the electromagnetic calorimeter, serves the purpose of detecting photons and electrons by measuring the energy they deposit. Meanwhile, the HCAL, the hadronic calorimeter, is responsible for

capturing the energy of both charged and neutral hadrons.

Both calorimeters are of the sampling type, characterized by alternating layers of different materials. Instead of immediately absorbing the entire energy of the incoming particle, these layers are designed to initiate a particle shower and then measure the energy deposit [28].

- **Muon detection system [M1-M5]:** The muon detection system, positioned at the outermost part of the detector, is specifically designed for the exclusive purpose of detecting muons.
- **Magnet:** A dipole magnet is positioned between the tracking stations TT and T1. The magnet allows the measurement of charged particle momentum by observing the curvature induced by the magnetic field parallel to the y -axis, resulting in bent trajectories on the xz -plane. For 10 m long tracks, the integrated magnetic field is approximately 4 Tm, while the residual magnetic field inside the two RICHs is approximately 2 Tm. The coverage of the magnet spans a forward acceptance of ± 250 mrad vertically and ± 300 mrad horizontally.

2.2 b - and c -jets formation

Jets are narrow cones of particles observed following collisions between protons with sufficient high energy. These collisions, occurring at a center-of-mass energy of 13.6 TeV at LHC, results in strong interaction between the protons. The collision triggers a complex chain reaction leading to the formation of cone-shaped patterns composed of long lived and stable particles.

The quarks composing each proton possess a portion of the overall electric charge, but more importantly, they carry a distinctive charge known as the color charge, which determines the strength of the strong interaction force. Due to the principle of color confinement, it is impossible to isolate individual particles with a non-zero color charge, rendering them undetectable.

The *tagging* of jets is the process of determining the type of quark that originated the jet, which cannot be directly measured due to color confinement. Instead, this identification relies on studying the stable particles forming the jet.

Accurately classify the types of jets is a crucial task in HEP for two main reasons. b - and c -jets may be produced by direct parton-parton interaction but also by the

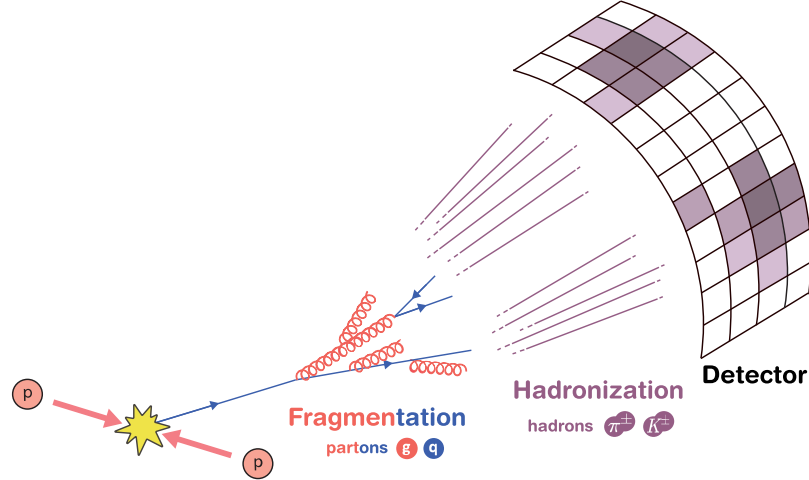


Figure 2.3: Formation of Jets: Partons inside protons collide, producing other partons or particles decaying to partons, of particular interest b and c quarks. These quarks then immediately undergo the process of fragmentation, which results in more quarks and gluons produced. Finally, they hadronize into short, long lived or/and stable particles. [29].

decay of newly discovered particles like the Higgs and being able to distinguish between $H \rightarrow b\bar{b}$ and $H \rightarrow c\bar{c}$ is one of the most challenging task in LHC experiments. Moreover the underlying mechanisms involved in the formation of jets, namely fragmentation and hadronization, are not yet fully determined.

The upcoming three sections will delve into the processes that give rise to the observed jets: Quarks and gluon production, Fragmentation and Hadronization.

2.2.1 Quarks and gluon production

When protons collide at high energies ($\sqrt{s} \approx 13$ TeV), their constituent partons can undergo a wide range of interactions and produce various physics phenomena. The kinematics of this collision are encoded in the Parton Distribution Functions (PDFs) and in the two partons elementary cross section. $d\sigma_{pp \rightarrow Q\bar{Q}}$, represents the probability density of a process where a quark pair $Q\bar{Q}$ is produced from the interaction of two protons pp :

$$d\sigma_{pp \rightarrow Q\bar{Q}} = \sum_{n,m} \int \int dx_n dx_m f_{n/p}(x_n, \mu) f_{m/p}(x_m, \mu) d\hat{\sigma}_{nm \rightarrow Q\bar{Q}} \quad (2.1)$$

Where n and m take into account all possible partons with a non-vanishing contribution at the parton level in $d\hat{\sigma}_{nm \rightarrow Q\bar{Q}}$, the inclusive elementary cross section, and μ is a renormalization parameter.

2.2.2 Fragmentation

After the production, the quark shifts away few millimetres and start fragmenting originating the jet.

The fragmentation is a process defined by the QCD, perturbative and non. During this process, a chain reaction of sub-processes are involved that results in the production of many color-charged particles. Schematically it can be summarized as:

- Emission of a gluon from a quark.
- Generation of a quark-antiquark pair by a gluon.
- Emission of a gluon by a gluon.

2.2.3 Hadronization

Due to color confinement, isolated quarks and gluons resulting from the fragmentation process, cannot be directly observed in nature. Instead, they quickly rearrange themselves to form color-neutral bound states, which are the detectable hadrons. This mechanism guarantees that only color-neutral particles, like mesons and baryons, are observed in experiments.

The result of these processes is the formation of hadrons narrow clustered, within the cone of a jet.

By studying the properties of jets, it is possible to infer the flavor of the quark originating the jet itself. Here, ML techniques are typically employed to exploit the correlations of all the particles inside the jet.

These techniques use high level jets information that are obtained with the event reconstruction software. Here the jets reconstruction is described.

2.3 Jets reconstruction

The algorithms used for jet reconstruction at LHCb combine the information from the sub-detectors listed in 2.1.2 to construct quantities representing each single jet event. The reconstruction of jets consists of the following steps:

- **Particle Flow** provides the list of particles reconstructed, eventually eliminating any false positives;
- **Clustering** is performed using the anti- k_T [30] algorithm, where particles are separated into different jets based on their association.
- **E-recombination scheme** where the four-momentum of the jet is computed as a function of the particles momenta;
- **Jet Energy Correction** is then applied to the final energy of the jets, based on Monte Carlo simulations.

2.3.1 Particle flow

The first step of the event reconstruction is the Particle flow algorithm. Its aim is to produce a singular list of the reconstructed particles such as photons, charged hadrons, neutral hadrons, muons, and electrons [31].

The output list is constructed to be as accessible as the list of actual particles derived from event reconstruction. Its purpose is to serve as input for the next reconstruction algorithms.

While applying the particle flow algorithm, certain detector hits might be discarded to guarantee the accuracy and reliability of the resulting data. These exclusions are based on quality criteria, based on data acquisition and previous reconstruction steps.

2.3.2 Clustering

Aggregating the reconstructed particles into jets/clusters is a complex task since particles belonging to separate jets could potentially overlap. There exist diverse strategies to estimate the number of clusters and subsequently assign each particle

to the most appropriate cluster.

The anti- k_t is the clustering algorithm employed at LHCb and many other experiments within LHC [30]. The main idea is to merge particles together into jets, according to a metric d_{ij} , that takes into account both the position of the particles and their momenta.

The anti- k_t algorithm works as follows:

1. For each pair of particles the distance d_{ij} is calculated:

$$d_{ij} = \min(k_{t,i}^{-2}, k_{t,j}^{-2}) \frac{\Delta_{ij}^2}{R^2} \quad \text{where} \quad \Delta_{ij}^2 = (y_i - y_j)^2 + (\phi_i - \phi_j)^2 \quad (2.2)$$

$K_{t,i}$, y_i and ϕ_i are the transverse momentum, the rapidity² and the azimuthal angle of the particle i , respectively. R instead is a tunable parameter of the algorithm called *radius*.

2. Once each distance d_{ij} is computed, a similar distance is calculated between each particle i and the beam axis

$$d_{iB} = k_{t,i}^{-2} \quad (2.3)$$

3. For each i , the distances d_{ij} and d_{iB} are compared:
 - 3.1 If $d_{ij} < d_{iB}$, the particle i is combined into j making a new particle (summing their four-momenta) and removed from the list of input particles.
 - 3.2 Otherwise the particle i is defined as a jet and removed from the list of particles.
4. The algorithm repeats steps 2 and 3 until any particle is remaining to be clustered.

The anti- k_t algorithm only has a free parameter R . It can be tuned in order to optimize the jet energy resolution depending on the type of experiment and detector properties. At LHCb, typical values are between 0.5 and 0.7 [30], For the specific data set used in this analysis, the jets were clustered setting the value to 0.5.

²rapidity y is defined as $y = (1/2) \ln((E + p_z c)/(E - p_z c))$

2.3.3 E-recombination scheme

The four-momenta $(E_{\text{jet}}, p_{\text{jet}})$ of the reconstructed jets after the clustering process is defined as the sum of energy and momenta of each particle:

$$E_{\text{jet}} = \sum_i E_i \quad p_{\text{jet}} = \sum_i p_i \quad (2.4)$$

The described algorithm is applied both to simulated and $p - p$ collisions jets. In this thesis the jet_{reco} are simulated jets reconstructed as the experimental ones while jet_{MC} are based on truth level Monte Carlo information. The main difference between the two classes lies in the particles used: jet_{MC} is clustered using all stable truth-level particles with truth-level kinematical values, while jet_{reco} is formed using reconstructed particles with reconstructed quantities.

To associate a jet_{MC} with a jet_{reco} , the distance between them in the $\eta - \phi$ plane, $\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2}$, must be smaller than 0.4. If more than one compatible jet is found within this range, the jet with the closest distance is chosen.

2.3.4 Jet energy correction

An important step in the jet reconstruction is the energy correction. Such a correction is needed to take into account several factors: partial containment of the jets in the calorimeter, not perfect calibration, energy losses due to sampling, event pile-up, etc. In this thesis, a correction factor is applied to the energy of the reconstructed jets, denoted as E_{reco}^{jet} . This correction is achieved using a factor called *Jet Energy Correction*, represented as k^{MC} , and it follows the equation:

$$E_{\text{MC}}^{\text{jet}} = k^{\text{MC}} E_{reco}^{\text{jet}} \quad (2.5)$$

2.3.5 Secondary Vertex finding

b - and c -jets exhibit the distinct behaviour of a secondary decay "within" the jet structure. The b and c hadrons undergo a decay process located in the Secondary Vertex (SV) which is few millimeters beyond the primary interaction point, named Primary Vertex (PV).

While analyzing the tracks stemming from the fragmentation process for these types of jets, it is possible to identify a secondary point where the tracks converge, the SV. This is exemplified in Figure 2.4.

This specific structure is unique to heavy quark (b and c) jets, as the resulting hadron in the collision possesses a sufficiently long lifetime to decay at a distance far enough from the PV [32]. The ability to detect the SV within a jet constitutes the primary strategy for distinguishing between heavy quark jets and light quark jets at LHCb. The characteristics of jets with a SV are presented in the next chapter.

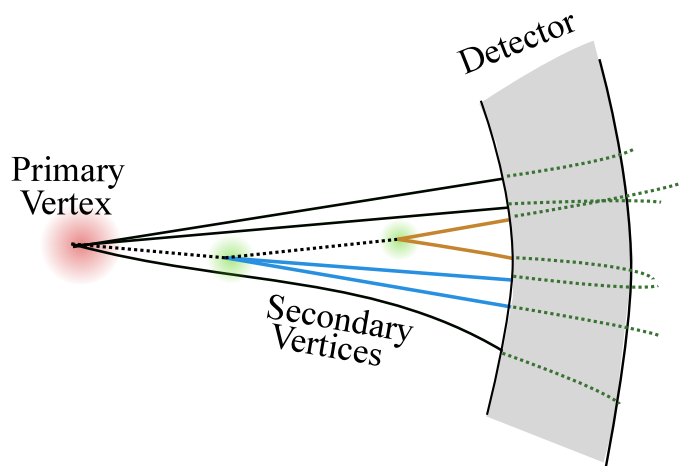


Figure 2.4: Illustration of a jet with secondary and tertiary decay vertices. [33]

b - and c -jet identification with Quantum Machine Learning

In this chapter, the focus initially lies in the presentation of the data set employed and analyzed for the classification of b - vs c - jets. Initially, the chapter presents the source of the data set, followed by an explanation of each individual feature present in the data set.

Afterwards, both classical ML models and QML models for b - vs c -jets classification will be introduced, to facilitate a comparative analysis between the two approaches.

3.1 Data-set

The data set samples are generated using the official LHCb Monte Carlo simulations, focusing on jets arising from proton-proton collisions with a center-of-mass energy of $\sqrt{s} = 13$ TeV (under Run 2 conditions).

The Monte Carlo simulation process involves the utilization of three distinct programs. Initially, Pythia [34] is employed to generate the hard process occurring during the collision. EvtGen [35] simulates the subsequent decay of b - and c - hadrons. Finally, Geant4 [36] is employed to simulate the interactions of the hadronized particles with the detector. The events, then, undergo processing through the pipeline showcased in Section 2.3 with the application of the ParticleFlow algorithm and the anti- k_T clustering algorithm.

3.1.1 Secondary Vertex features

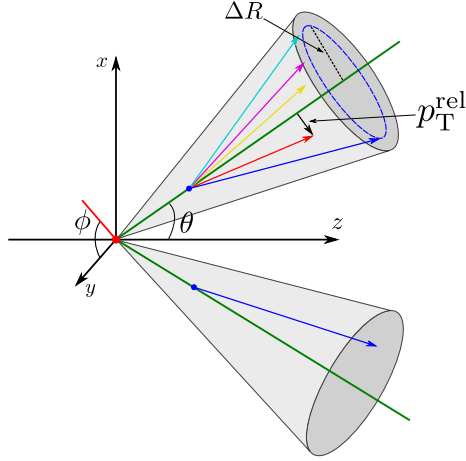


Figure 3.1: The jet on top illustrates a secondary vertex, SV, with some of the constituent features. [4]

Since the tagging in this thesis focuses on heavy quark jets, events are distinguished by the presence of the SV. To the SV, are associated the following quantities, which will form the utilized data set:

- **vtx_fdrMin:** the transverse flight distance of the two-track SV closest to the PV;
- **vtx_ptSvrJet:** The fraction of the component of the momentum transverse to the beam p_T carried by the SV over the total transverse momentum, $p_T(SV)/p_T(jet)$;
- **vtx_nTrk:** the number of tracks forming the SV;
- **vtx_drSvrJet:** the ΔR between the SV flight direction and the jet (as in Figure 3.1);
- **vtx_nTrkJet:** the number of SV tracks with $\Delta R < 0.5$ relative to the jet axis;
- **vtx_absQSum:** the net charge of the tracks that form the SV;
- **vtx_m:** the SV mass assuming all particles being a pion;

- **vtx_mCor:** it takes into account the invariant mass of the particles constituting the SV, as well as the momentum and the angle between the momentum vector and the direction of the SV trajectory.;
- **vtx_fdChi2:** the flight distance χ^2 ;
- **vtx_ipChi2Sum:** the sum χ_{IP}^2 of all tracks associated with the SV. χ_{IP}^2 refers to the difference in the χ^2 value of the PV reconstructed with and without the considered track.
- **vtx_tau:** the SV lifetime;
- **vtx_z:** the SV z -position;
- **vtx_pt:** the SV transverse momentum p_T ;

It is important to highlight that not all of these quantities will be employed in the subsequent methods due to the considerable computational demands of simulating and training a quantum circuit able to embed this high number of features. The implementation of the subsequent models rely on the Angle Embedding method outlined in Section 1.2.2. Consequently, to handle N features, N qubits are needed to be simulated.

Figure 3.2 illustrates the distribution of the features of the SV. The SV corrected mass distribution exhibits the most significant differences between b -jets and c -jets. Because of this, the corrected mass is anticipated to yield a substantial score in the feature importance analysis conducted in Section 5.2. Additionally, in the data set, each jet contains information about its quark origin, which is determined by the matched Monte Carlo jet. The label takes a value of 0 if the jet originates from a b -quark or 1 if it comes from a c -quark. This "Jet Label" variable serves as the target feature for the tagging algorithms aimed at distinguishing between the two types of jets.

3.1.2 Data set selection

The data set derived from the Monte Carlo events is subjected to a set of filters. These filters are commonly applied in the context of Jet data analysis at LHCb as their purpose is maintaining a high data quality due to limitations arising from detector and the reconstruction algorithm.

3.1. DATA-SET

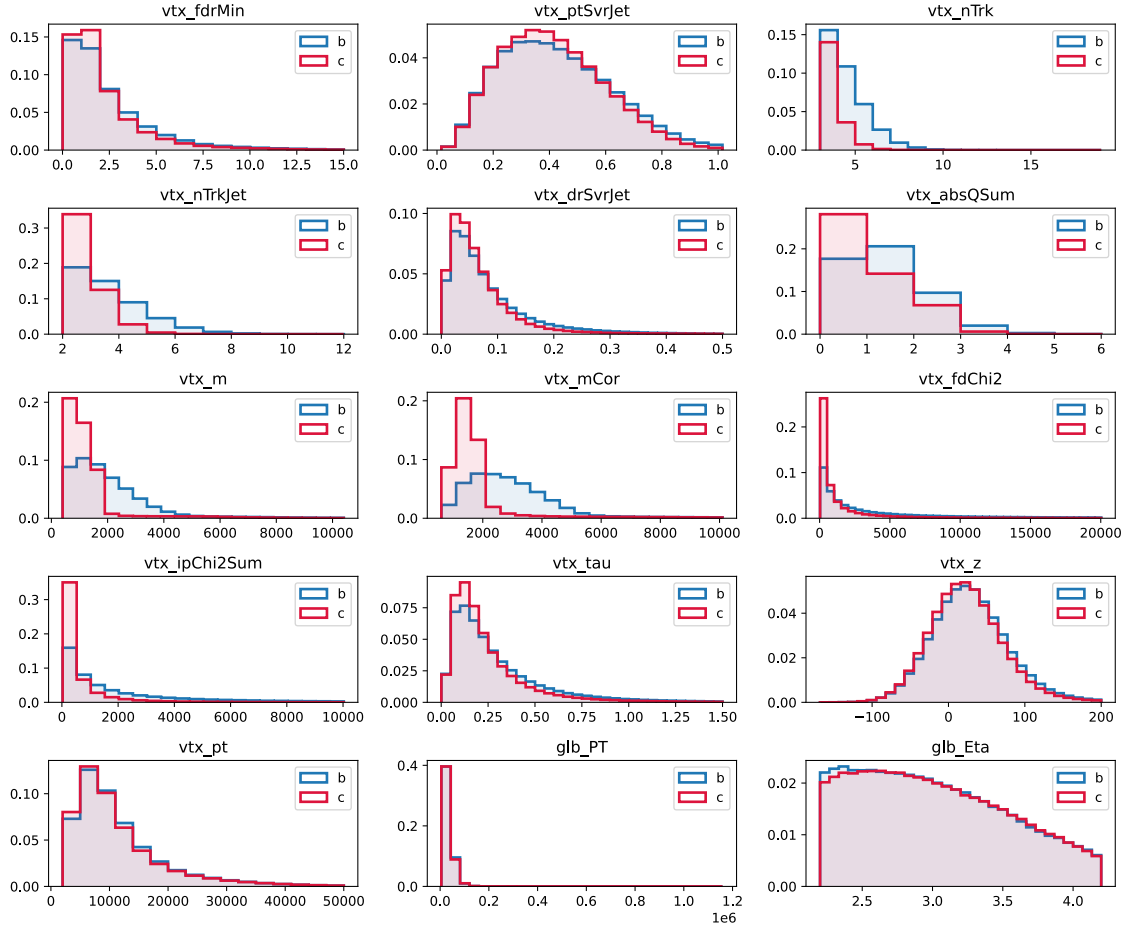


Figure 3.2: Distribution of the features of the b - jets (blue) and c - jets, in red. Description of each feature in the text.

- Transverse Momentum: $20 < p_T < 1000$ GeV/c. This criterion ensures the accurate reconstruction of the jet, as the reconstruction algorithm tends to be less reliable outside this range.
- Pseudo-rapidity: $2.2 < \eta < 4.2$. This condition guarantees that the jet remains entirely within the angular acceptance of the actual detector.

3.2 Classical model: (Gradient) Boosted Decision Trees

The classical model selected for the comparative analysis is the Boosted Decision Tree (BDT) [37], which serves as a standard in jet flavor tagging at LHCb [38]. BDTs are an enhancement of simple decision tree models that incorporate gradient descent techniques (gradient boosting) for optimization.

Decision tree: A decision tree is a straightforward machine learning model, taking the form of a tree-like structure. In this structure, each internal node represents a data split based on a specific value of a given feature. The process of splitting continues until the training data set is sufficiently divided, ideally leading to fully separated classes in a classification task, and ultimately concluding with leaf nodes that represent class labels. The training of these trees involves identifying the most suitable values for data splitting at each node.

Decision trees are models that offer high interpretability; however, they are susceptible to overfitting. Their full potential is realized when multiple decision trees, forming a "forest," are aggregated using a voting system. When presented with an input, each decision tree provides a predicted class, and the final assigned class is determined by the majority of predictions among the constituent trees. This ensemble approach enhances generalization and reduces the risk of overfitting, making random forests a powerful and widely-used tool.

Gradient boosting: Gradient boosting is a technique that combines the concept of gradient descent to the construction of new trees in a forest. The process involves adding new trees based on the prediction error of the preceding tree, updating the weight parameters to minimize the error of the next individual decision tree. This iterative approach allows the next tree to focus on the patterns that were not captured

3.3. SOFTWARE IMPLEMENTATION FOR JET TAGGING ON QUANTUM MODELS

well by the previous trees, leading to a more accurate and robust model. A visual representation of this method is depicted in Figure 3.3.

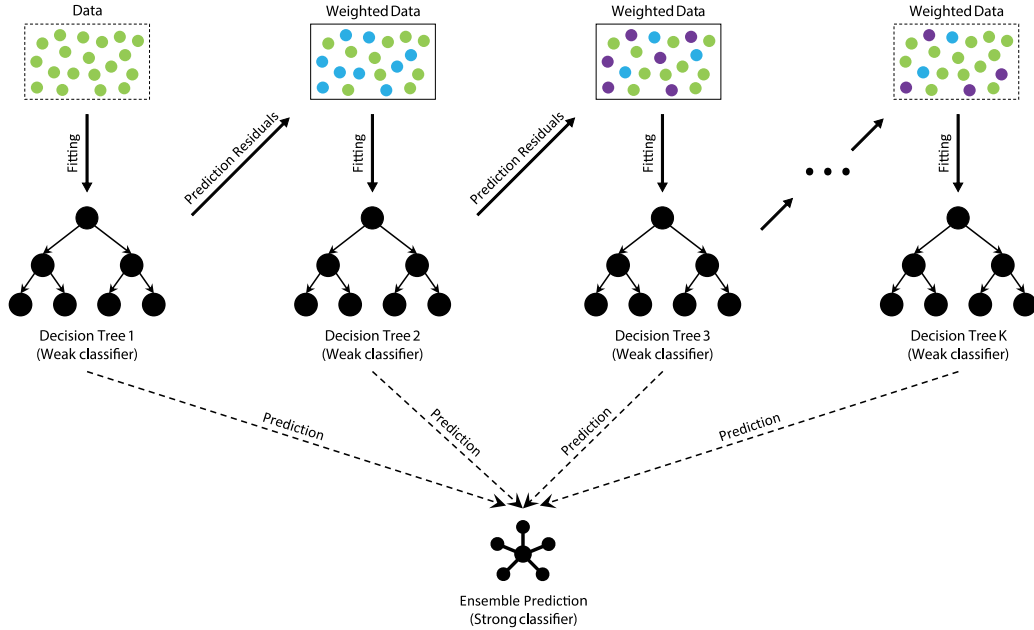


Figure 3.3: Diagrammatic representation of a Boosted Decision Tree. [39]

3.3 Software implementation for jet tagging on quantum models

3.3.1 PennyLane

The quantum models discussed in the following sections have been implemented in Python using the PennyLane library [40]. This library enables the implementation of the quantum techniques elucidated in Chapter 1. However, simulating a quantum circuit within a classical computer entails notable limitations that effectively nullify the exponential advantage. Notably, simulating a quantum system with 10 or more qubits in a classical computer necessitates significant memory allocation, making it as well computationally expensive due to the multiplication of large matrices.

In Figure 3.4 shows a basic example of a QML model. The first line of code loads the PennyLane package, while the second line defines a quantum device which determines

the overall behavior of the simulator.

In Figure 3.4, a 1-qubit circuit is defined using the `circuit` Python function, constructed with a parametrized $R_X(x)$ gate (where x is the trainable parameter). After applying the rotation, the measurement is performed on the only wire, computing the Z -expectation value. This circuit is executed alongside the `qml.qnode(dev)` decorator, which maps the circuit to the specified device, creating a Quantum Computation Node (QNode). By calling the `circuit` function, the quantum circuit is evaluated on the selected device.

The `cost` function serves as the loss function explained in Chapter 1. Within this framework, the cost function is essential for the explicit definition of its gradient.

Finally, the last three lines of code implement an optimization step of the cost function concerning the weight parameters, utilizing a simple Gradient Descent optimizer.

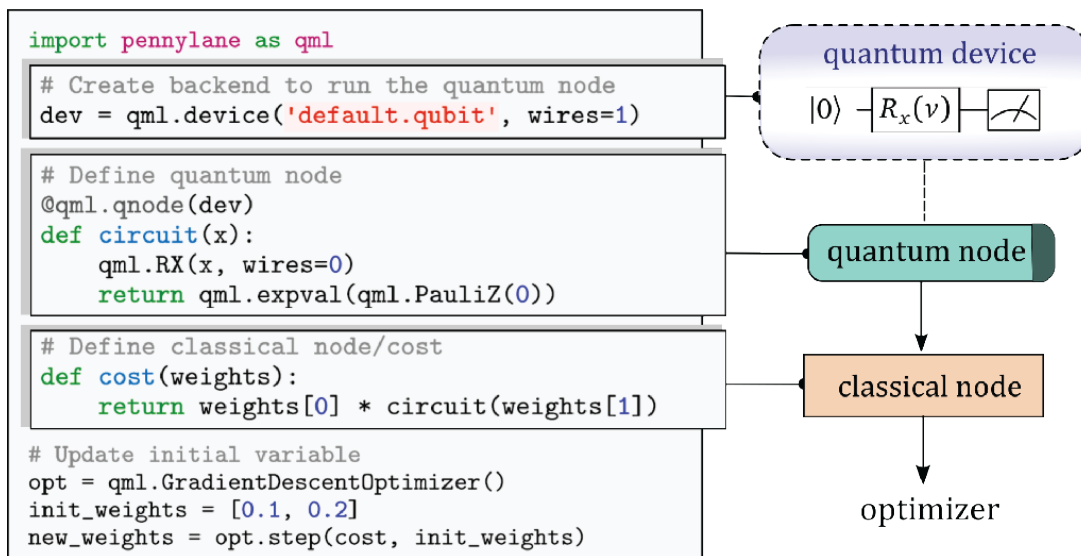


Figure 3.4: An illustrative instance of a QML model pipeline, comprising a quantum node and a subsequent classical node. The outcome obtained from the classical node serves as the optimization target. [40]

In this work, the "JAX" backend (see next paragraph) is employed for simulating the devices, particularly for its high-speed automatic differentiation capabilities. The functions `jit` and `vmap` provided by the JAX library [41] enable efficient execution on accelerators like GPUs, contributing to improved overall performance.

3.3.2 JAX

JAX, developed by Google, is an open-source library designed for high-performance numerical computing and machine learning. In this work, the following features offered by JAX are used:

- **Automatic Differentiation:** JAX provides efficient and accurate methods for computing gradients. Specifically, the function `jax.value_and_grad` is used to compute the gradient function of a given circuit for the optimization loop.
- **Vectorization:** Through the function `jax.vmap`, it is possible to vectorize a circuit, allowing different executions of the same quantum circuit to be applied in parallel with no significant increase in execution time.
- **Just-in-Time Compilation (JIT):** JAX's `jax.jit` feature enables to accelerate the execution of Python functions (including circuit functions) by converting them into optimized machine code and executing them on a GPU.

By leveraging these features, the work gains advantages such as efficient gradient computation, parallel execution of quantum circuits, and accelerated execution on GPUs, leading to enhanced performance of the QML algorithms.

3.4 Quantum Models

A thorough investigation has been conducted to identify the best set of hyper-parameters for a QML classifier employed in the *b*- vs. *c*-jet tagging [3].

The analysis centered specifically on the hyper-parameters: Ansatz and Data Scaling. The latter refers to the function applied to the input data prior to embedding it into the circuit.

This investigation resulted to a model configured as follows:

QAOA Ansatz: The parametrized circuit design draws inspiration from the combinatorial problem QAOA [42]. This circuit is designed to encode N features in N qubits and encloses a total of $2 \cdot L \cdot N$ trainable parameters, where L is the number of iterations of a single QAOA block. The complete Ansatz is visually presented in Figure 3.5. In each iteration of the block, the data is initially embedded using the

standard angle embedding (refer to 1.2.2). Subsequently, a parametrized multi-qubit gate R_{ZZ} is applied, followed by an independent parametrized R_y rotation on each qubit. After the L repetitions, an additional final embedding is applied [43] and then the Pauli-Z expectation value is measured.

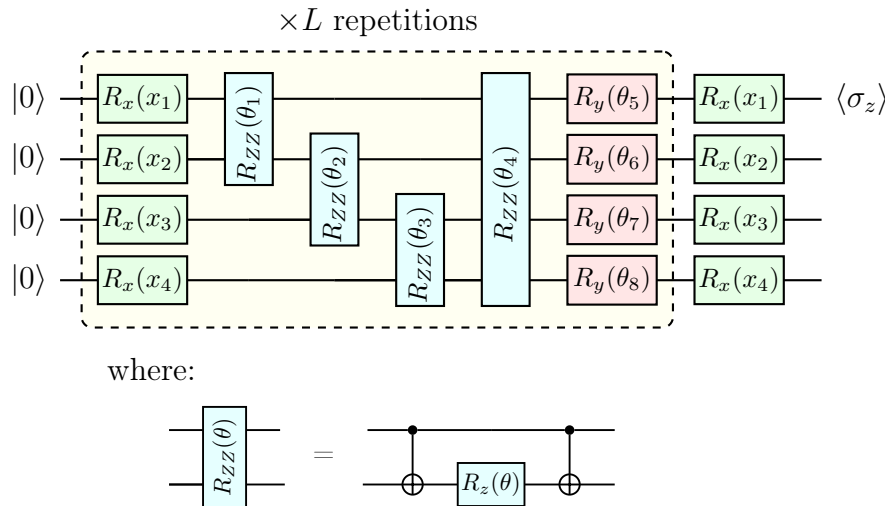


Figure 3.5: QAOA Ansatz.

Power Transformer: The PowerTransformer is a scaler function provided by the Python library sklearn. Its purpose is to transform each feature across all events in a way that they exhibit a Gaussian-like distribution with a mean of 0 and a standard deviation of 1 [44]. An example of the transformation on the input feature `vtx_pt` can be seen in Figure 3.6.

3.5 QML Performance

In the following section, multiple models are compared with different hyper-parameters. To assure a fair comparison, the following models underwent training using the identical training set. Following their training, their performance has been evaluated using an identical test set.

The training set consists of 276,790 jets , and the test set comprises 126,228. Both the training and test sets have a balanced distribution of the two classes, with a split

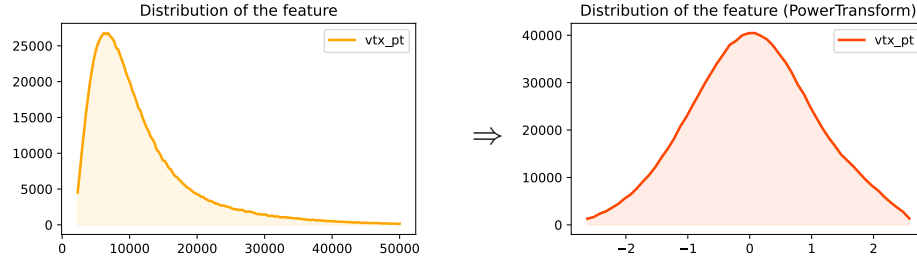


Figure 3.6: Distribution of the values of the feature `vtx_pt` before (left) and after (right) applying `PowerTransformer`

of 50% b-jets and 50% c-jets.

The performances have been evaluated using two metrics on the test set: the standard accuracy and the ROC AUC.

3.5.1 Metrics

Accuracy: One of the simplest metric is the accuracy, defined as the ratio of all correctly classified samples over all the samples:

$$\text{Accuracy: } \frac{\sum_{i=1}^{|M|} \delta(y_i - \tilde{y}_i)}{|M|} \quad (3.1)$$

where M is the set of input events and y_i and \tilde{y}_i are respectively the true and the predicted label of the jet i .

When provided with input, the models generate a score. A score close to 0 indicates higher confidence that the input event belongs to class 0, whereas a score near 1 signifies greater confidence in classifying it as class 1. A firm threshold is set at 0.5. Therefore, if the score falls below this threshold, the model predicts the input belongs to class 0; otherwise, it predicts it belongs to class 1.

ROC AUC: One of the most utilized metrics for assessing the performance of a ML classifier is the area under the curve (AUC) of the Receiver Operating Characteristic (ROC) curve. The ROC curve is generated by plotting the true positive rate (TPR) against the false positive rate (FPR) at different threshold settings across the prediction distributions [45]. In Figure 3.7, several ROC curves are illustrated for

different model examples. In the case of a random classifier, the ROC curve takes the form of a straight line from the points (0,0) to (1,1). Conversely, for an ideal classifier, the ROC curve starts directly from (0,0) to (0,1) and then proceeds to (1,1). The area beneath the ROC curve indicates the performance of a given model, with a larger area signifying better performance.

The AUC metric is computed by evaluating the integral (i.e., the area) beneath the ROC curve. Unlike plain accuracy, a high AUC score not only implies that the classifier accurately predicted most test events, but also that these predictions were made with high confidence and precision.

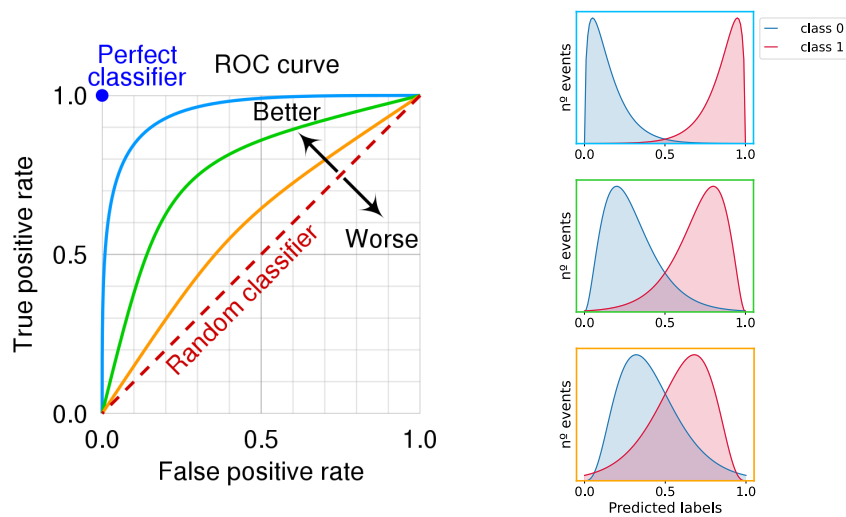


Figure 3.7: ROC curves of three models, the blue line is the ROC curve associated to the classification on top, the green line is the middle model, the orange line is related to the bottom model. [46]

3.5.2 Scaling with number of qubits N and layers L :

The effectiveness of the QAOA Ansatz was evaluated across various values of N (number of qubits) and L (number of layers/iterations). Given that the QAOA Ansatz embeds a feature in each wire, the value N corresponds to the quantity of features from the SV utilized in the analysis as well. Increasing N results in the incorporation of a greater number of features, while raising L maintains the same feature count but raises the number of trainable parameters of the model.

3.5. QML PERFORMANCE

The result of this analysis is summarised in Figure 3.8. As expected, elevated values of both N and L yield to improved model performance. It is worth noticing that a crucial number of layers threshold exists, beyond which accuracy plateaus due to the model reaching a saturation point in parameter necessity. On average, this the layer count is at 7 for all the models.

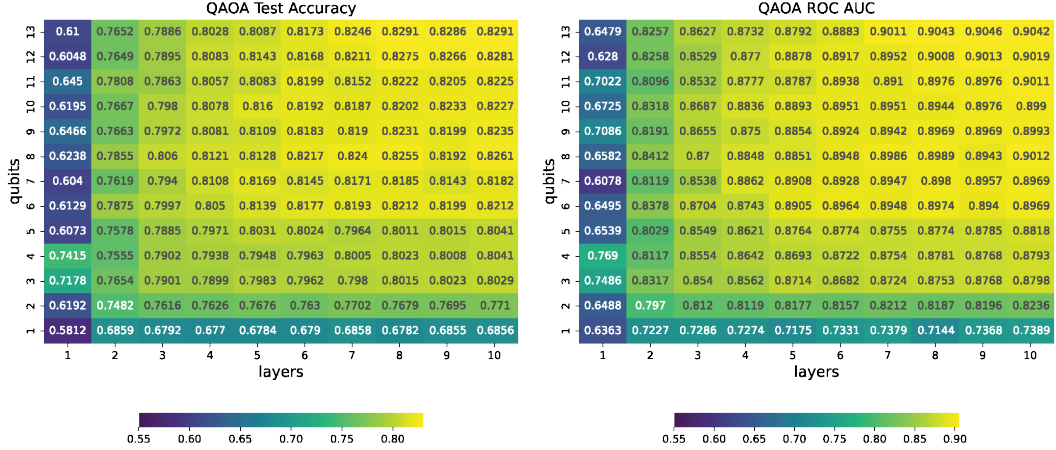


Figure 3.8: Test accuracy (left) and its corresponding ROC area under the curve (right) as a function of number of qubits and number of layers using the QML model. [3]

3.5.3 Comparison between QPC and BDT

In this section, a trained QAOA circuit is compared with a classical BDT presented in Section 3.2. In order to guarantee a fair comparison, the number of qubits (and features) for the QAOA circuit is set to 8 as well as the number of features for the BDT model.

Figure 3.9 shows the prediction distributions of the BDT and QAOA. The BDT model performs a better job in effectively distinguishing between the two classes, achieving correct predictions for the majority of events with high confidence. Conversely, while the QAOA circuit correctly classify a considerable portion of events, it consistently maintains a conservative approach in its predictions.

CHAPTER 3. JET IDENTIFICATION WITH QUANTUM MACHINE LEARNING

To quantitatively assess the aforementioned assertion, the Figure 3.10 displays ROC curves and AUC scores of the two models. As anticipated, the BDT model demonstrates a slightly superior performance, as evidenced by its ROC curve consistently above of that of the parametrized quantum circuit. The AUC score for the BDT model is 0.922, while for the quantum circuit, it stands at 0.896.

The classical BDT exhibits overall superior performance in the tagging task. How-

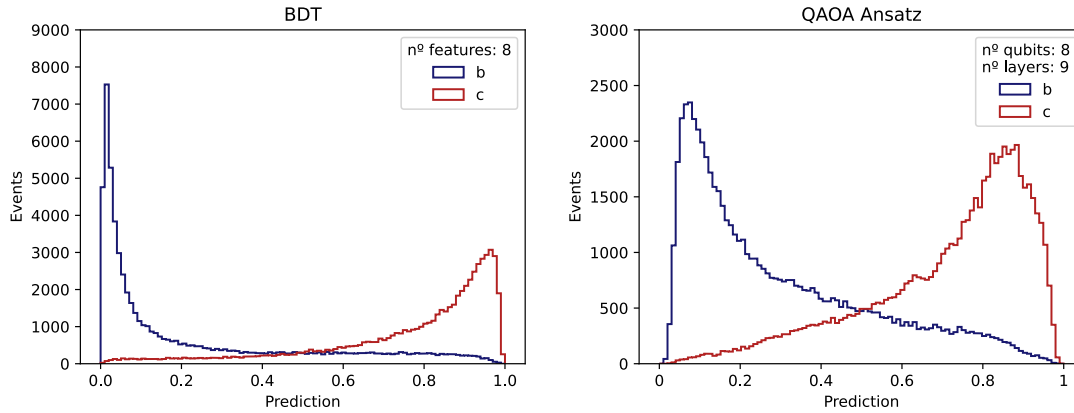


Figure 3.9: Predictions of the classical (left) and quantum model (right)

ever, an implementation of a quantum classifier with a higher number of features could potentially diminish the disparity between the quantum and classical models. The difference in outcomes achieved at this limited scale constitutes the primary factor driving the integration of unused information within the quantum classifiers, which it will investigate in the subsequent two chapters.

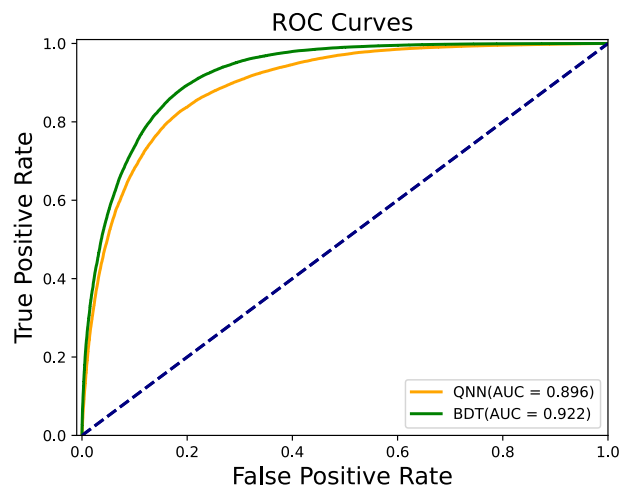


Figure 3.10: ROC curves of the Boosted Decision Tree model (green) and QML model(orange) for 8 features/qubits

Benchmarking of entropy production in a quantum classifier

Entanglement is a vital component in QML models as it enables them to achieve high *expressiveness* (as defined below), even beyond classical models. By allowing qubits to exhibit intricate and non-classical behaviors, entanglement unlocks the potential to process highly complex data, which could prove advantageous, particularly in the context of tagging heavy quark jets. Currently, this task poses significant challenges that are not easily overcome even with advanced models like BDT.

The general assumption is that the expressiveness of a quantum model increases with higher entanglement capability. This enables the quantum model to explore larger portions of the highly-dimensional Hilbert space, allowing it to represent more complex and diverse quantum states.

Expressibility: In the context of a PQC it refers to its capability to generate states that effectively represent the entire Hilbert space. For instance, for a single qubit, this translates to the circuit's ability to explore the entire Bloch sphere as in Figure 4.1. In the case of two or more qubits, expressibility refers to the circuit's capacity to fully explore its entire Hilbert space. Unlike the single-qubit scenario, the exploration of this multi-qubit Hilbert space cannot be visually represented through graphical illustrations like Bloch spheres.

However, the high expressivity offered by QML compared to ML poses a major

4.1. THE BARREN PLATEAU PROBLEM IN QML

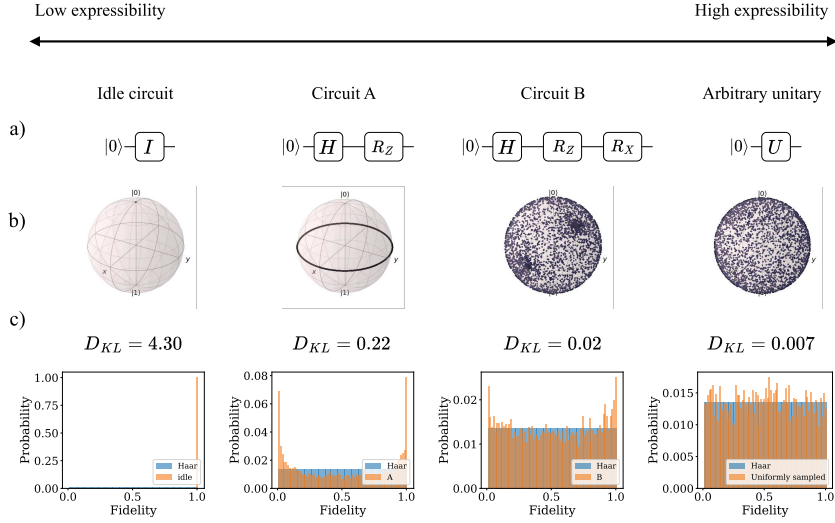


Figure 4.1: Examples of four 1-qubit circuits with different expressive powers. [47]

training challenge known as the "*Barren Plateau*" problem [48]. Currently, the primary obstacle to achieving deep quantum circuits is the presence of these plateaus alongside the present limitation by the physical hardware.

4.1 The Barren Plateau problem in QML

PQC exhibit high expressivity, mainly because of the vast dimensionality of the Hilbert space they can explore. While this high expressiveness offers advantages, it also poses challenges during the training phase. The high dimensionality of the Hilbert space can lead to exceedingly flat loss functions, known as Barren Plateau. In this scenario, the gradients necessary for efficient optimization become vanishingly small, making it extremely difficult and computationally expensive to train the quantum circuit effectively.

Overcoming the Barren Plateau problem in QML is an active area of research. While there is no one-size-fits-all solution, several approaches have been proposed to mitigate or address this challenge.

The main proposed ideas revolve around limiting the portions of the Hilbert space accessible to the trainable quantum model to restrict it to regions that potentially contain the desired optimal solution.

Some of the strategies to overcome the barren plateau include:

- **Circuit design:** The placement and number of parametrized single-qubit gates and multi-qubit gates play a fundamental role in determining the level of entanglement generated within a quantum circuit. The arrangement of these gates is critical as it directly influences the amount formed and the specific pair of qubits involved.
The Quantum Convolutional Neural Network (QCNN) [49], the quantum counterpart of a classical convolutional neural network, stands out as a notable example of an architecture that has demonstrated resilience against the barren plateau problem [50].
- **Initialization Scheme:** The general consensus is that a Gaussian Distribution with a mean 0 is the preferred choice for parameter initialization. However, determining the appropriate standard deviation has been a subject of investigation. Existing literature [51] proposes a formula that takes into account the number of parametrized gates. This initialization is proven to ensure that the expectation value of the gradient of the loss function remains above a certain lower bound, thereby preventing the loss function from becoming flat during optimization.
- **Optimisation strategy:** The types of optimization algorithms and loss formulas used play a critical role in the formation of entropies during QML tasks. Key choices are:
 - Parameters update algorithm: such as plain Gradient Descent, ADAM Gradient Descent or even gradient-free methods such as SPSA [52].
 - Cost function: the function that determines the error in the prediction of the model is responsible as well for the update of the parameters, hence the formation of the entropies.
- **Data Preprocessing:** Data preprocessing is a vital factor in determining the quality of any ML model. In the context of QML models, data preprocessing becomes even more crucial due to the specific requirements of data embedding, such as addressing periodicity. The impact of data preprocessing is significant as it directly affects the model's training process and also influences the generation of entropies within the quantum circuit.

In the following section, an analysis will be conducted on how the different strategies listed above influence the production of entropy (entanglement) within a quantum circuit. This relationship is inherently linked to the barren plateau problem, as the objective during training is to find a sweet spot where the entropy is sufficient to ensure model expressiveness, while also maintaining a low enough amount of entropy for effective training. Achieving this balance is crucial for optimizing the quantum circuit and addressing the challenges posed by the barren plateau.

4.2 Entropy productions

The entropy production has been investigated under various hyperparameter configurations of the quantum circuits.

Bipartitions chosen: As elucidated in Chapter 1, entanglement within a quantum circuit can be computed between two bipartitions of the whole system. A bipartition refers to a pair of subsets of wires, whose combination covers all the wires, as illustrated in Figure 4.2. The selection of the specific bipartitions for inspection is entirely arbitrary, and in this study, the following bipartitions have been chosen for examination:

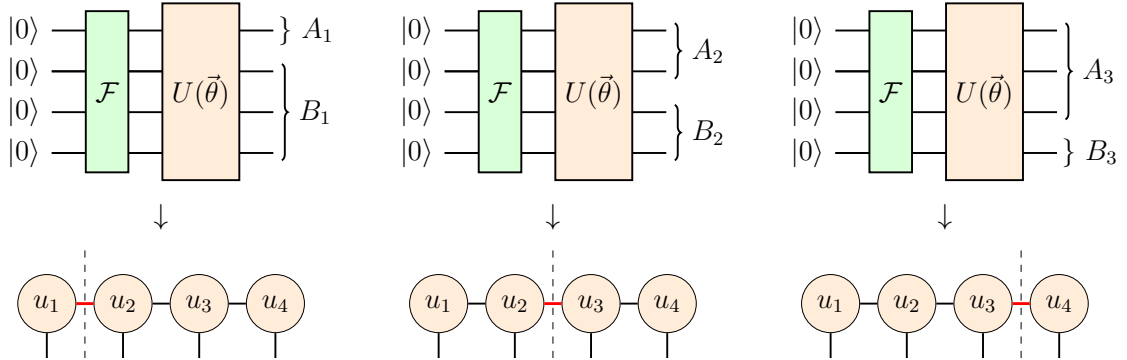


Figure 4.2: Bipartitions chosen of a 4-qubit quantum circuit (above) and their respective bipartitions in a Tensor Network representation (below).

For such bipartitions, the entanglement established within a quantum circuit across the chosen bipartition is linked to a parameter referred to as the *bond dimension*. This parameter gains significance when contemplating an analogous framework called

CHAPTER 4. BENCHMARKING OF ENTROPY PRODUCTION IN A QUANTUM CLASSIFIER

the *Matrix Product State* Tensor Network [37]. The bond dimension quantifies the degree of quantum entanglement present in the described state. A state with a low bond dimension exhibits limited entanglement and can be represented using a small number of parameters, making it relatively efficient. Conversely, a state with a high bond dimension indicates a highly entangled and complex quantum state, necessitating an exponentially larger number of parameters for a complete description [37].

Entropy computation framework: The approach for computing and analyzing the entropies shares similarities with the methodology presented in the paper titled "*Entanglement entropy production in Quantum Neural Networks*" (Ballarin et al., 2022) [53]. In the cited paper, the entropy production was analyzed by computing the *average entropies* for multiple trainings. Additionally, our investigation further involved studying the evolution of the entropies at each training step and exploring the changes in entropy as the single wavefunction evolved through the circuit.

To elaborate further, let us consider the training step t , where the circuit's trained parameters are denoted as $\vec{\theta}^{(t)}$. The examined circuit can be viewed as comprising L iterations of the same *circuit block*. Each circuit block has only a fraction of the complete parameter vector $\vec{\theta}^{(t)}$, which is decomposed into L vectors: $\vec{\theta}_1^{(t)}, \vec{\theta}_2^{(t)}, \dots, \vec{\theta}_L^{(t)}$, with each vector representing the parameters for one circuit block. An example for $L = 3$ is depicted in Figure 4.3.

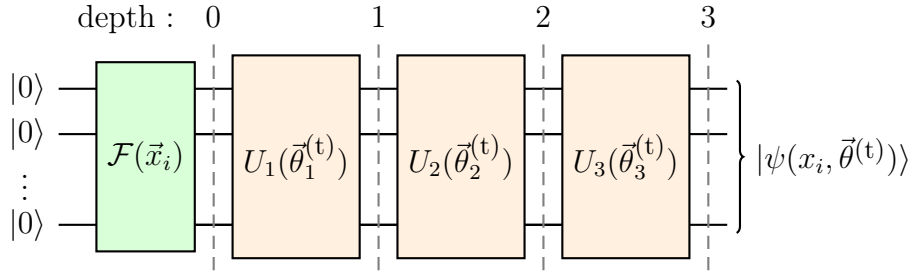


Figure 4.3: Circuit of Depth 3 at Training Step t . The first one is the embedding block, then 3 iterations of an Ansatz block U .

In addition to examining the entropies concerning the output wavefunction at each training step, I also analyzed the evolution of entropies as the wavefunction evolved

after the application of each individual circuit block.

At the training step t , the considered wavefunction are the following:

1. $\mathcal{F}(\vec{x}_i)|0\rangle$, namely the wavefunction by only applying the embedding (depth= 0)
2. $U_1(\vec{\theta}_1^{(t)})\mathcal{F}(\vec{x}_i)|0\rangle$, the wavefunction after only the first iteration of the circuit block (depth= 1)
- ⋮
- $L + 1$. The output wave-function $|\psi(x_i, \vec{\theta}^{(t)})\rangle = U_L(\vec{\theta}_L^{(t)})\dots U_1(\vec{\theta}_1^{(t)})\mathcal{F}(\vec{x}_i)|0\rangle$ (depth= L)

The algorithm used to compute all the essential entropies for the data analysis is elaborated further with pseudo-code in Algorithm 4.1.

The subsequent sections provide an analysis of entropy productions, comparing how changes in the following hyper-parameters impact them: Circuit design, Parameters Initialization, and Cost Function. These hyper-parameters were observed to have the most significant influence on the entropy variations.

4.2.1 Ansatz

The Ansatz is certainly the most impactful factor in the production of entanglement, as it decides where to place each unitary operation, and most importantly, how to connect a pair of qubits through multi-qubit gates, shaping the formation of entanglement.

In this study, three different circuit designs are compared, as seen in Figure 4.4. These circuits are intended to exhibit different qualities:

- **QAOA Ansatz:** This is the same circuit inspected in Chapter 3. It is the most well-rounded among the analyzed circuits, having only $2N$ parameters per iteration and connecting qubits only in subsequent pairs.
- **Circuit 8:** The most parameterized, with $(9/2)N$ parameters per iteration for even N . However, connections between qubits are limited to nearest neighbors.

Algorithm 4.1: Framework of entropy production analysis

Initialization: Choose variational circuit block U and number of iterations of the block L

for $model = 1 \dots n_models$ **do**

 Set the initial random parameters of the given model $\vec{\theta}^{(0)}$

for $t = 1 \dots n_epochs$ **do**

 Update the training parameter in accordance with the optimization loop:

$$\vec{\theta}^{(t)} = \vec{\theta}^{(t-1)} - \eta \nabla L(\vec{\theta}^{(t-1)})$$

for $depth = 0 \dots L$ **do**

for each input event \vec{x}_i **do**

 Compute the relative wavefunction given the current parameters

$\vec{\theta}^{(t)}$, L and \vec{x}_i :

$$|\psi(\vec{\theta}_L^{(t)}, \dots, \vec{\theta}_1^{(t)}, \vec{x}_i)\rangle = U_L(\vec{\theta}_L^{(t)}) \dots U_1(\vec{\theta}_1^{(t)}) \mathcal{F}(x_i) |0\rangle$$

for each bipartition $(A|B)$ **do**

 Compute the entropy between the subsystems A and B from the relative wavefunction

end

end

 Average among all the input events \vec{x}_i for each bipartition

end

end

end

Average among all models

- **Circuit RY full:** This circuit has the lowest number of parameters per iteration, being just N . At every iteration, every qubit is interconnected with each other through CZ gates.

The top set of plots, depicted in Figure 4.5, illustrates the training progress of the three models, distinguished by their respective colors (blue for QAOA Ansatz, green for circuit 8, and violet for circuit RY full). Each column of sub-figures corresponds to a specific bipartition, as previously illustrated in Figure 4.2. Conversely, each row represents a distinct depth value.

To gather meaningful statistics due to the strong dependence of entropy production on initial parameters, each configuration underwent 100 separate training runs with

4.2. ENTROPY PRODUCTIONS

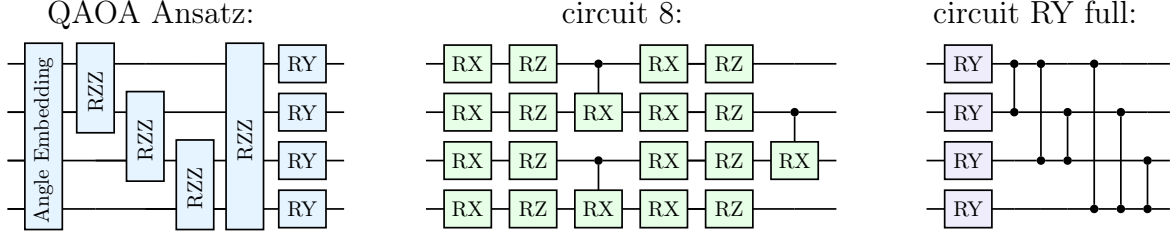


Figure 4.4: Circuit blocks examined. The "QAOA Ansatz" involves a total of $2N$ parameters per iteration, whereas "circuit RY full" utilizes $(9/2)N$ parameters, and "circuit 8" employs only N parameters.

varying initial random parameters. Each thin line in the graph represents the average entropy value among the whole input events at a specific training step t , indicated on the x-axis. The thick lines indicate the average entropy among all models for a given set of hyper-parameters, hence the line obtained averaging the y-values of the thin lines.

The bottom plot follows a similar structure, but it showcases the entropy values of the already-trained models. In this case, the x-axis represents the *depth* coordinate, providing insight into the evolution of entropy productions as the wavefunctions traverse the trained circuits.

Each plot features a red line representing the entropy value of a maximally entangled state, alongside an orange line representing the Haar Random Entropy [54]. The latter is the average entropy value of a state randomly selected from a uniform distribution across the Hilbert space. The values of these lines depend on the size (number of qubits) of the two bipartitions involved.

The three circuits exhibit varying levels of average entropy generation. "circuit RY full" possesses the highest values of entropy across all bipartitions, followed by "circuit QAOA," and then "circuit 8."

To clarify the disparity in entropies between the "circuit 8" and "circuit RY full" models, it is important to address the influence of over-parametrization leading to low trainability and potentially resulting in a barren plateau phenomenon (thus, high entropy). Interestingly, the actual results contrast this notion, showing that the lower-parametrized model has the highest overall entropy.

CHAPTER 4. BENCHMARKING OF ENTROPY PRODUCTION IN A QUANTUM CLASSIFIER

In reality, it is not merely the quantity of parameters within a circuit that dictates the average entropy production. Rather, it is the number of multi-qubit gate connections that plays a crucial role. Notably, "circuit RY full" possesses the highest number of these connections, contributing to its elevated entropy levels.

As stated previously, achieving enhanced performance involves striking a balance between high entropy production (indicating high expressivity but lower trainability) and low entropy production (denoting lower expressivity but better trainability). To support this assertion, the accuracies are presented. Notably, "circuit QAOA" emerges as the better model in terms of accuracy, with 81.00%. This places it ahead of the other two models, with "circuit 8" achieving 78.50% and "circuit RY full" with 77.50% accuracy.

4.2. ENTROPY PRODUCTIONS

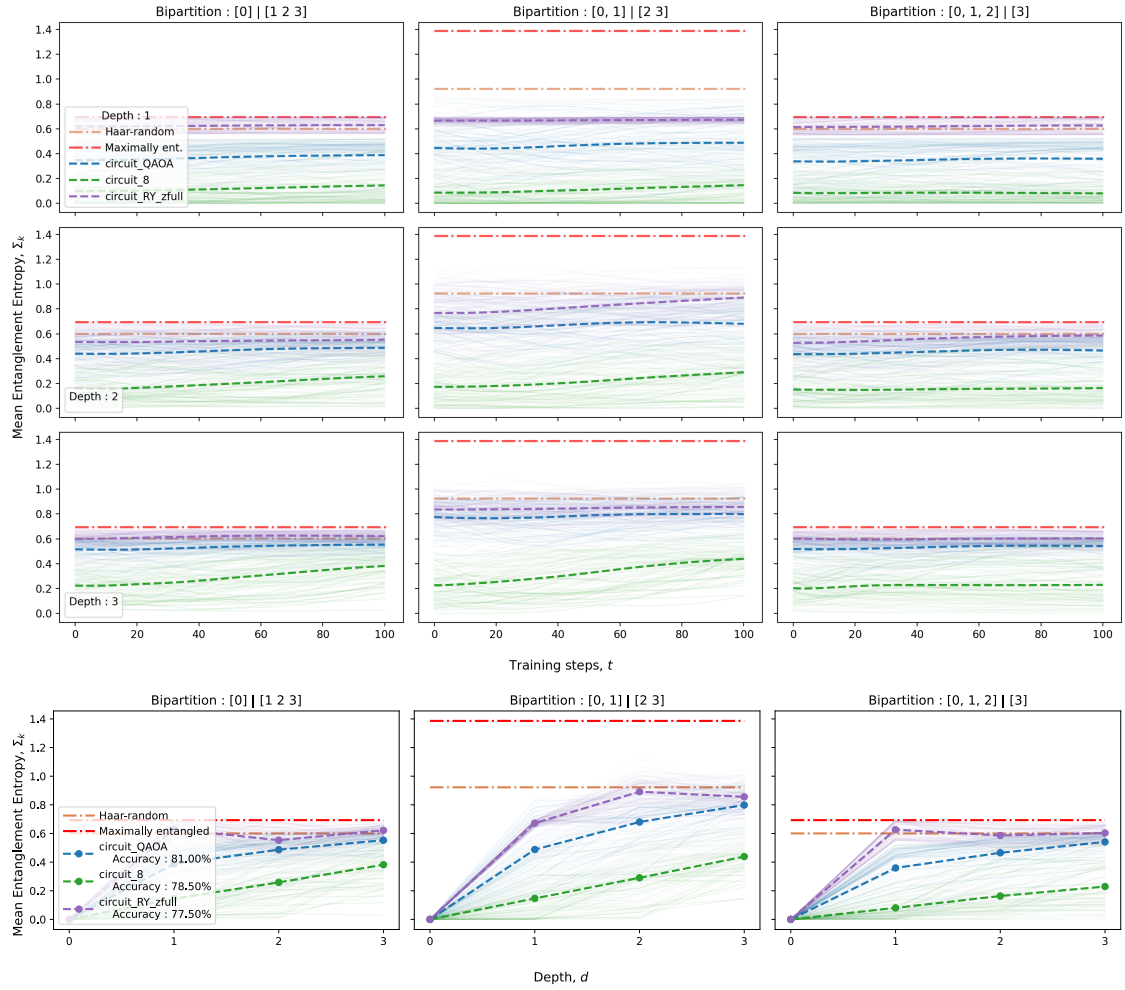


Figure 4.5: Entropy generation is examined across three distinct models for the cases of $n = 4$ and $L = 3$, considering various circuit designs. The upper section illustrates the progression of each entropy among the considered bipartitions throughout the training phase. Conversely, the lower section showcases the ultimate entropy values attained after each iteration.

4.2.2 Parameters initialization

Parameter initialization has been identified as a contributing factor to the presence of barren plateaus [51]. Specifically, when setting the initial parameters prior to training using a Gaussian distribution centered around zero and with an appropriate standard deviation, the circuit demonstrates a reduced level of entanglement.

Operationally, this approach enables the model to be initialized in a region of the Hilbert space that does not exhibit any plateau. As a result, the optimization through gradient descent in the initial stages of training becomes more impactful.

Similarly to the preceding section, a comparison between two models is analyzed. Each of these models comprises 3 repetitions of the QAOA block, serving as an Ansatz for 4 qubits. However, they are initialized using two distinct Gaussian distributions: one with a smaller standard deviation ($\sigma = 0.3$), and the other with a larger standard deviation ($\sigma = 3$), close to being a uniform distribution.

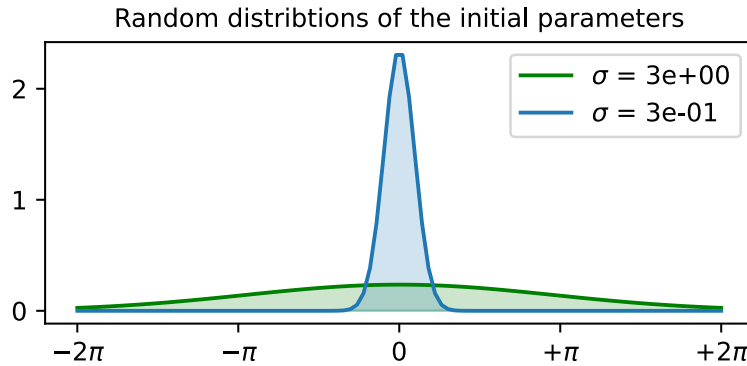


Figure 4.6: Drawing distribution of the initial parameters for a model with low variance (in blue) and for a model with high variance (in green, with a nearly uniform distribution).

Figure 4.7 depicts the entropy generation of these two models. The upper graph illustrates the evolution across various bipartitions and depths. As previously mentioned, the primary distinguishing factor between the two models lies in the initial training phase, where the entropies of the low standard deviation model start below those of the other model. However, after a few epochs, the entropies of the low standard deviation model begin to converge with those of the higher standard deviation model.

4.2. ENTROPY PRODUCTIONS

The lower set of plots displays the entropies of the fully trained models as the wavefunctions progress through the circuit. On the whole, the average entropies of the low standard deviation model remain slightly lower than those of the other model. Moreover as expected, the accuracy of the first model is marginally higher than the other, surpassing it by a 1.3 percentage point difference.

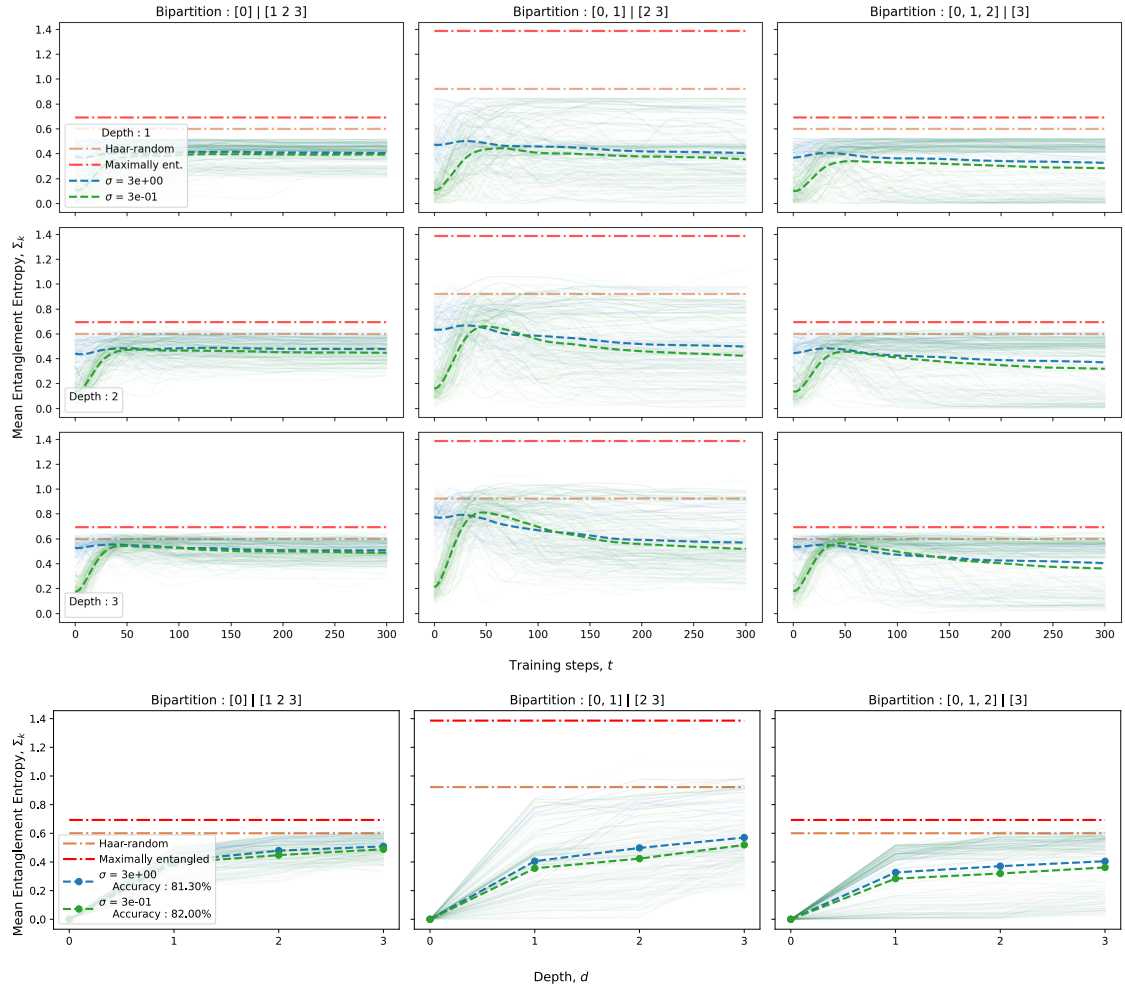


Figure 4.7: Entropy production examined for different parameters initialization for a $L = 3$ and $n = 4$, QAOA circuit.

4.2.3 Cost function

The final factor under study contributing to the occurrence of barren plateaus during training is the type of cost function. This term pertains to the number of qubits that are measured at the end of the circuit to generate predictions.

The cost function can be categorized as either *global*, involving the measurement of all qubits to assess the output, or *local*, where the output is derived solely from the measurement of a subset of all qubits, as depicted in Figure 4.8.

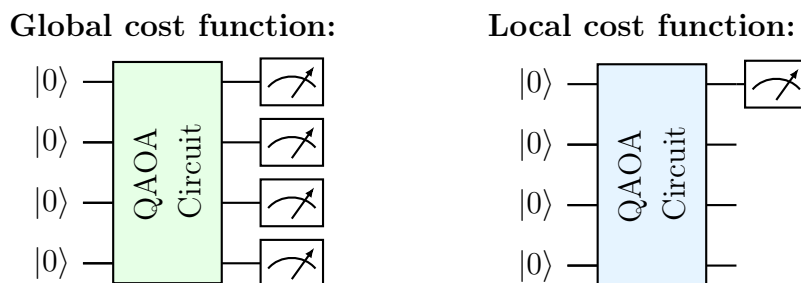


Figure 4.8: For a circuit with a global cost function (left), the output is derived from the information encompassing all qubits. For a circuit featuring a local cost function (right), the output is acquired through the measurement of only a small subset of qubits.

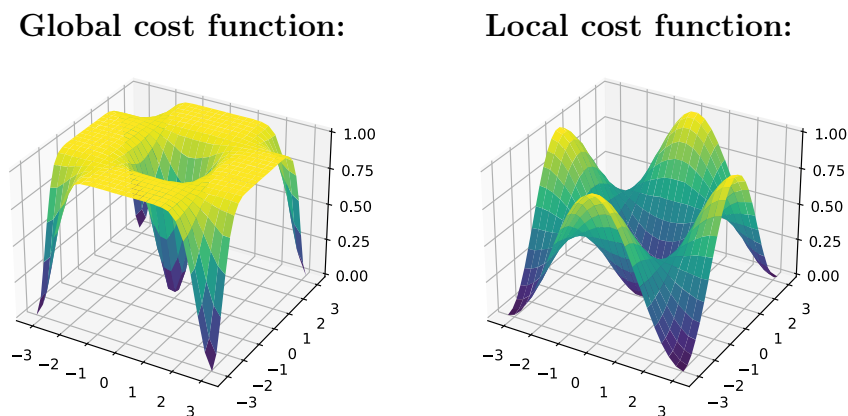


Figure 4.9: Change in the loss landscape for the same model with a Global Cost function and a Local Cost function. [55]

A local cost function has been already established to enhance the training process of a model by creating a more gradient-friendly loss function [55, 56], as illustrated

4.2. ENTROPY PRODUCTIONS

through an instance in Figure 4.9. This attribute justifies to the popularity of the QCNN architecture, because its intrinsic design ensures a localized measurement approach [50].

In this last section, the examination focuses on the entropy generation of the QAOA Circuit with parameters $N = 4$ and $L = 3$. One model is trained using a global cost function, while the other with a local cost function as in Figure 4.9.

Figure 4.10 depicts the entropy production in the two configurations. Across different depths and training steps denoted as t , the model with a local cost function consistently displays lower entropy levels.

The model with a local cost function shows minor improvements in accuracy, although it is expected that further benefits of employing a local cost function can be harnessed by tailoring the circuit Ansatz to connect specific qubits according to those being measured as well as creating the circuit dependent on the specific classification task.

Overall here were presented some factors that alleviated the barren plateau problem, showing how it is related to entropy production. While opting for the improved hyper-parameters discussed - including a balanced Ansatz, Gaussian initialization, and a local cost function - does yield improved models, it does not ensure flawlessly trouble-free training due to the high expressivity. To gain a substantial advantage, innovative techniques need to be developed that go over the mere selection of hyper-parameters. This holds particular significance when entering the era of deep quantum neural networks.

CHAPTER 4. BENCHMARKING OF ENTROPY PRODUCTION IN A QUANTUM CLASSIFIER

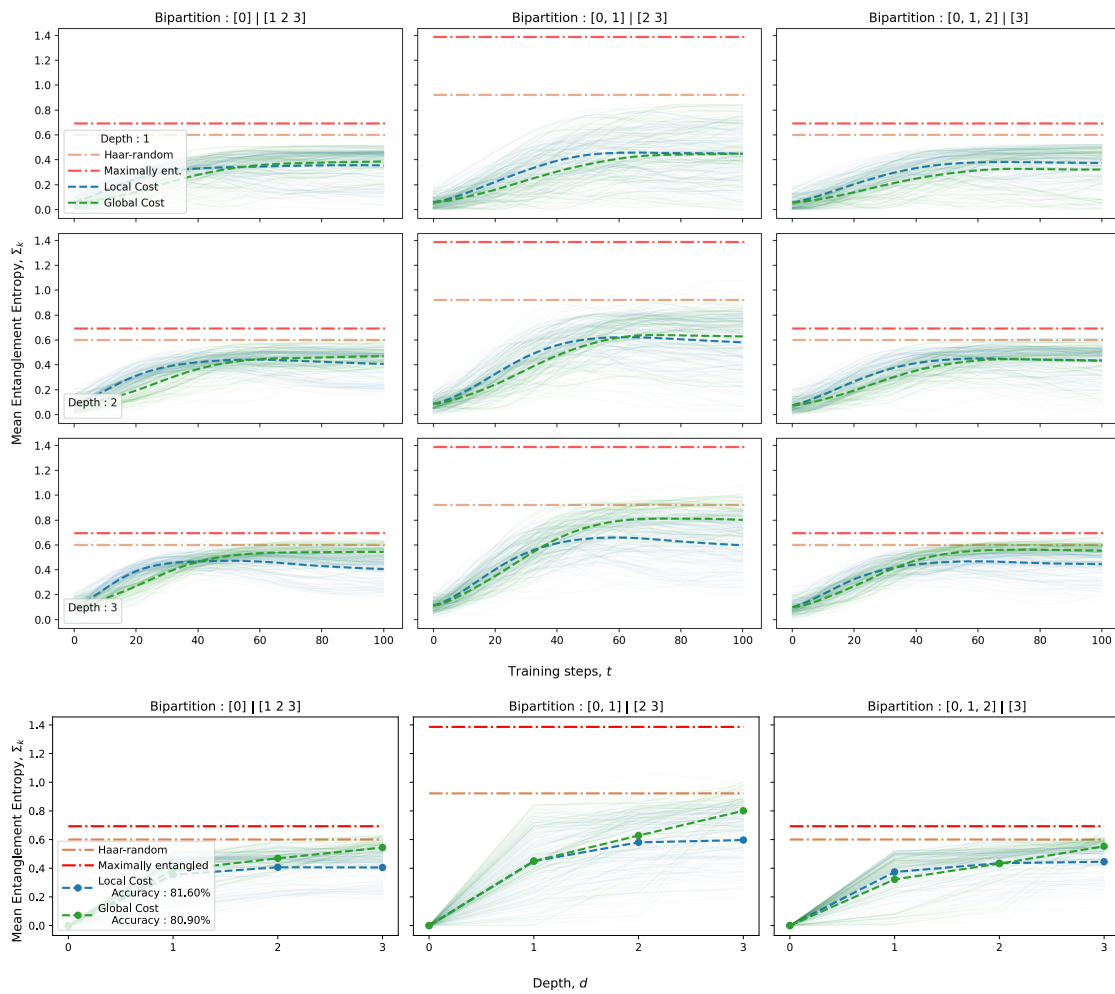


Figure 4.10: Entropy production examined with a local and global cost function for a $L = 3$ and $n = 4$, QAOA circuit.

Entanglement-based models for b - and c -jet tagging

Until now, the focus of the study has been on entropies within regular quantum classifiers, treating the examination of entropy production as an additional element. In the upcoming section, a reversed approach is adopted, where models are constructed based on the concept of entropies. As a result, the entropies of the models developed correspond to meaningful quantities, leading to unique models that differ from those observed previously.

The model presented in this chapter retains its focus on distinguishing between bottom and charm jets and it draws inspiration from the work titled "*Quantum-inspired machine learning on high-energy physics data*" [57]. In that study, a Tree Tensor Network Approach was devised for performing b - versus \bar{b} -jet classification. However, the data set used in the cited paper differs from the one employed in this work (see 3.1).

5.1 Characteristic circuits

The primary goal of this model is to optimize the parameters of a circuit $U(\vec{\theta})$ in such a way that the resulting evolution from the initial state $|0\rangle$, denoted as $U(\vec{\theta})|0\rangle$, is a wavefunction that represents the data on which the model has been trained.

In the specific case of this study, which involves the classification of b - versus c -jets,

5.1. CHARACTERISTIC CIRCUITS

the idea is to separately train two circuits, namely $U_B(\vec{\theta}_B)$ and $U_C(\vec{\theta}_C)$. The resulting circuits applied to the initial state $|0\rangle$, will output two wavefunctions that optimally capture the characteristics of the two classes, b and c .

5.1.1 Training of the circuits

As just mentioned, the training process for both circuits involves making the resulting wavefunctions represent the occurrences of their respective classes, also ensuring in the meantime that they differ significantly from the events belonging to the opposing class.

The concept of quantifying the distance between the output of the circuit, the representative wavefunction, and the input events is accomplished using the *inner product* between two wavefunctions.

Inner product between two wavefunctions: The inner product between two quantum states is a mathematical operation that measures the degree of overlap or similarity. Given two quantum states represented as ket vectors, $|\psi\rangle$ and $|\phi\rangle$, the inner product $\langle\psi|\phi\rangle$ is defined as:

$$\langle\psi|\phi\rangle = \sum_i \psi_i^* \phi_i \quad (5.1)$$

where ψ_i^* is the i -th element the complex conjugate state $|\psi\rangle$, and ϕ_i denotes the i -th component of state $|\phi\rangle$.

If the inner product between two states is 0, it indicates that they are orthogonal or completely dissimilar. On the other hand, a non-zero inner product indicates that there is some degree of similarity or overlap between the states.

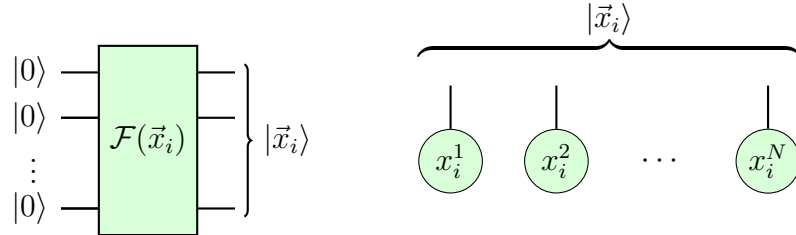


Figure 5.1: Embedded input state using a Quantum Circuit and as Tensor Network.

While training a characteristic Circuit, the inner product is used between the output of the circuit and the wavefunction obtained from embedding an input event (as in Figure 5.1). A notable distinction from the previous models in this study is that the input event contributes to the loss function computation not by directly loading the data into the circuit before the variational unitaries (namely through *embedding*), but rather as an independent instance used in the loss for computing the overlaps with the characteristic wavefunction.

Quantitatively, the following objectives are aimed to be achieved for each characteristic function with the label K :

- Ensure a high overlap between the characteristic function and the events sharing the same label: $\langle \vec{x}_i | \psi_K(\vec{\theta}) \rangle \sim 1$, where $label(\vec{x}_i) = K$.
- Ensure a low overlap (orthogonal) between the characteristic function and the events with a different label: $\langle \vec{x}_i | \psi_K(\vec{\theta}) \rangle \sim 0$, where $label(\vec{x}_i) \neq K$.

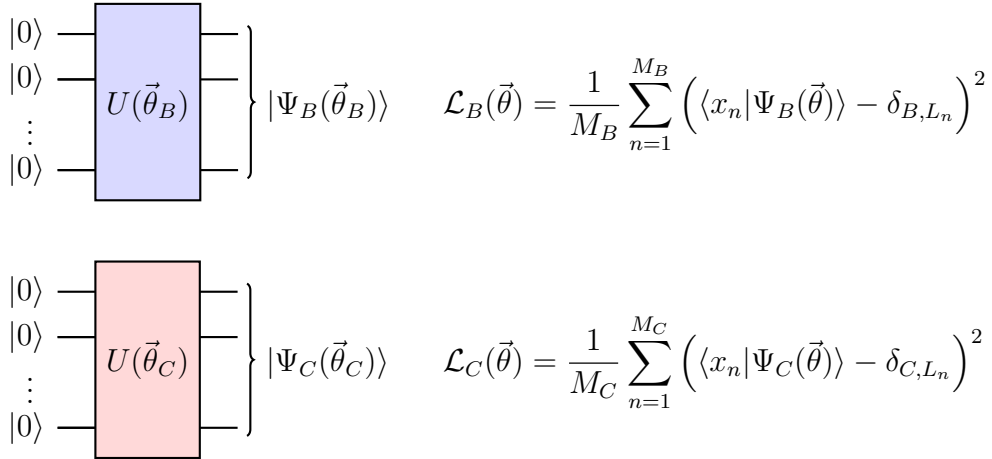


Figure 5.2: Characteristic circuits for the class of b and c jets and their respective Loss functions.

The loss functions used for the two models are illustrated in Figure 5.2. These loss functions are specifically crafted to achieve both aforementioned objectives.

The term δ_{K,L_n} corresponds to the Kronecker delta, where L_n denotes the label of the event x_n . When the label differs from the characteristic circuit's label, the delta is 0. In this case, the optimization aims to make the overlap between the

5.1. CHARACTERISTIC CIRCUITS

characteristic function and the event close to 0. On the other hand, if the event shares the same label as the characteristic function, the Kronecker delta is 1. Consequently, to minimize this quantity, the overlap needs to be as close to 1 as possible.

To some extent, this loss is a mean squared error (as seen in section 1.1), in which the target value changes arbitrarily: the target is either 1 if the classes of the input event and of the wavefunction match (high overlap), or 0 if the two classes do not match (low overlap).

Figure 5.3 illustrates the training process of the two characteristic circuits. The left plot depicts the curve of the loss function $\mathcal{C}(\vec{\theta}_B, \vec{\theta}_C)$, which is the aggregate of the individual losses from the two circuits: $\mathcal{C}(\vec{\theta}_B, \vec{\theta}_C) = \mathcal{L}_B(\vec{\theta}_B) + \mathcal{L}_C(\vec{\theta}_C)$. The right plot showcases the progression of the overlap between the outputs of the two circuits $\langle \Psi_B | \Psi_C \rangle$. While this measure does not inherently provide information about the optimization quality, it offers valuable insights. Specifically, if the overlap between the two characteristic functions were to approach either 1 or -1, it would signify a high correlation between the two functions. Consequently, distinguishing events in one of the two classes would become exceedingly challenging.

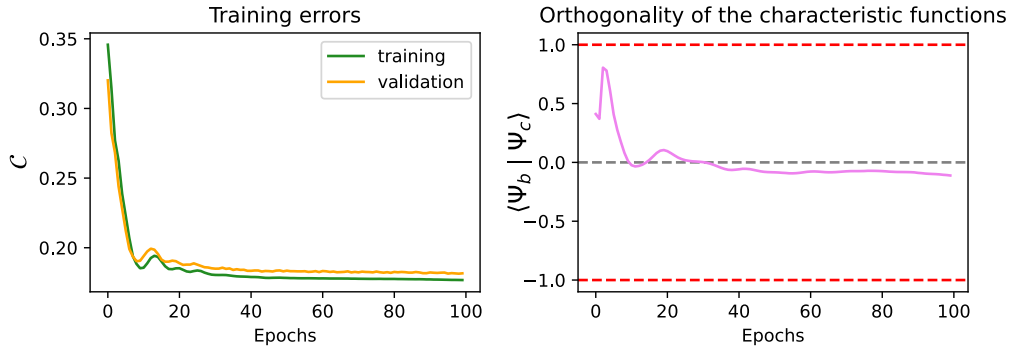


Figure 5.3: Training of the two characteristic circuits: on the left, the error curves for the training and validation set, on the right the value of overlap between the two characteristic wavefunctions.

5.1.2 Classification phase

Following such training, the resulting characteristic wavefunctions will exhibit a favorable response to events with labels that match the characteristic circuit, resulting in a high overlap. Conversely, they will not respond effectively to events with labels

different from the circuit, leading to a low value of overlap.

Therefore, one possible method for performing classification in this framework is to assign the label of the circuit that produces the highest overlap as shown in Figure 5.4. The prediction metric employed for an event $|\vec{x}_i\rangle$ is determined by the difference in the overlaps: $\langle \vec{x}_i | \Psi_C(\vec{\theta}_C) \rangle - \langle \vec{x}_i | \Psi_B(\vec{\theta}_B) \rangle$. If this value is negative, it indicates that circuit B exhibits a more favorable response, consequently leading to its classification as a b-jet. Conversely, if this value is positive, the label is chosen as C .

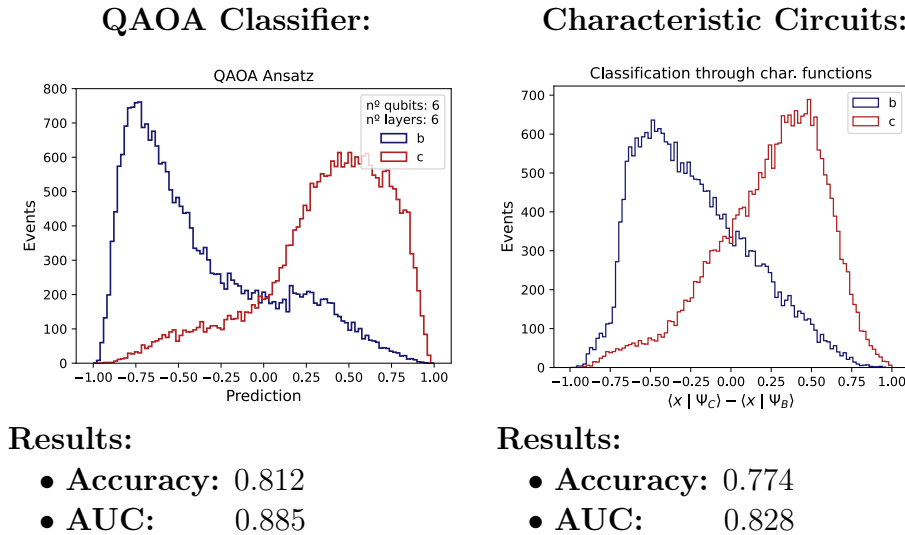


Figure 5.4: Comparison of the prediction distributions for a QAOA classifier (left) and a classifier through the characteristic circuits.

Figure 5.4 depicts a comparison of the classifications using a QAOA classifier and a classification using the studied model in this section. To ensure a fair comparison, both models were trained using the same numbers of qubits $N = 6$ and the same number of iterations of the circuit blocks $L = 6$. The classification through characteristic circuits demonstrates lower effectiveness, yielding a lower accuracy of and an AUC score than the ones obtained with the dedicated Quantum classifier. However, the primary objective of this model is not centered around classification, but rather aimed at extracting insights from the training data obtaining intricate underlying data structures that a conventional classifier might not be able to catch.

The resulting characteristic wavefunctions carry valuable information in their entanglement. A crucial aspect of this process is the use of Angle Embedding to compute

the wavefunction of the training events as in Figure 5.1. During training with data, each wire of the characteristic circuit represents a specific feature. For example, if the feature $\forall t x_m$ is embedded on the first wire, the first qubit of the trained characteristic circuit will represent that particular feature.

The entropies between subsystems of wires that represent features naturally provide insights into the significance of those specific features. In the following two sections, the Feature Importance and Correlations between the features is showcased.

5.2 Feature importance

Feature importance within the context of ML involves an assessment of the data set to determine the relative significance of individual features for the classification or regression task. This technique is employed to reduce the dimensionality of the problem, reducing the computation time without any major drop in accuracy and avoiding over-fitting.

Within the domain of QML, the efficient reduction of data set dimensions holds utmost significance. This is due to current quantum circuit models that are constrained in their capacity to accommodate only a limited number of features. For instance, an average computer could simulate up to 10 qubits, whereas a supercomputer equipped with multiple GPUs might achieve a simulation of up to 20 qubits. Beyond these numbers, the employment of actual quantum hardware becomes indispensable. This stands in contrast to classical models, which are capable of handling much larger feature sizes, potentially reaching hundreds of thousands in the case of image data sets [58].

Classical Feature Importance: There are many approaches available for assessing the feature importance for a ML model. Among these, one of the most adaptable and straightforward methods is known as *permutation importance*.

The core concept behind permutation importance is to assess the impact on performance when a particular feature loses its significance. This is achieved by first training the model, evaluating its accuracy, and subsequently randomizing (by shuffling) only the values associated with a specific feature and then measuring the subsequent drop in performance. The extent of this drop determines the importance of the feature.

The algorithm works as follows [59]:

1. Train a model and evaluate its performance.
2. For each feature f :
 - 2.1 Repeat the following process k times (for statistical robustness):
 - 2.1.1 Shuffle the values corresponding to the column of feature f .
 - 2.1.2 Calculate the resulting accuracy.
 - 2.2 Compute the average reduction in accuracy.
3. Normalize the average accuracy reduction values.

One of the primary advantages of this method is that it can be applied to any ML model. In the ensuing comparison, the classical model utilized for determining feature importance is the BDT, introduced in Section 3.2.

Quantum Feature Importance: The feature importance derived from a trained characteristic function is determined by computing the Von Neumann entropy of the bipartition between the wire representing that feature and all the other qubits in the characteristic function:

$$\text{Importance}(A) = \frac{S(\rho_A)}{\sum_F S(\rho_F)} \quad (5.2)$$

where $S(\rho_A)$ is the value of entropy between the subsystems A and B as in Figure 5.5, and the denominator is the sum of all the importance of the feature for normalization.

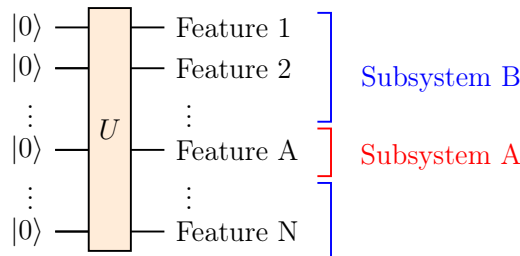


Figure 5.5: Computation of $S(\rho_A)$.

The motivation behind considering these quantities as feature importance is best explained through the Tensor Network equivalence [57]. Specifically, the Von Neumann

5.2. FEATURE IMPORTANCE

Entropy between these two bipartitions is equivalent to the bond dimension of the corresponding Matrix Product State.

In that context, the bond dimension represents the amount of information shared during the classification contraction as in Figure 5.6 (which is equivalent to the computation of the overlap). This can be readily interpreted as the feature importance, where a higher bond dimension indicates that the corresponding feature contributes more significantly to the classification process.

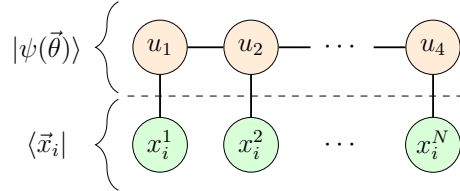


Figure 5.6: Overlap computed on a Tensor Network model.

Both classical and quantum approaches for the feature importance analysis have been performed on the first six features of the data set in Figure 5.7. Due to the absence of an analytical method for an exact feature importance computation, direct comparison with an absolute ground truth is not possible. However, the general agreement between the two methods implies that both models are capturing meaningful importance values.

Furthermore, certain insights can be drawn based on the underlying physics phenomena. For instance, in the tagging between b - and c -jets, the corrected mass of the SV is highly significant. This is because of the distinct masses of the decaying b and c hadrons that generate the SV. Indeed, both quantum and classical methods assign the highest degree of importance to the corrected mass feature. This also is in agreement with the considerations made from the distribution of the features in Figure 3.2.

The analyzed features are `vtx_mCor`, `vtx_ipChi2Sum`, `vtx_tau`, `vtx_ptSvrJet`, `vtx_m`, and `vtx_pt`. Figure 5.8 illustrates the impact of feature selection in a resource-limited quantum classifier. The two exhibited classifiers were respectively trained utilizing the foremost four crucial features, namely `vtx_mCor`, `vtx_ipChi2Sum`, `vtx_tau`, and `vtx_ptSvrJet`, and the least four significant features, which include `vtx_tau`, `vtx_ptSvrJet`, `vtx_m`, and `vtx_pt`.

The classifier trained with the most important features showcases a discernible advantage, evident in both accuracy and AUC metrics, which surpass the corresponding values achieved by the alternate model.

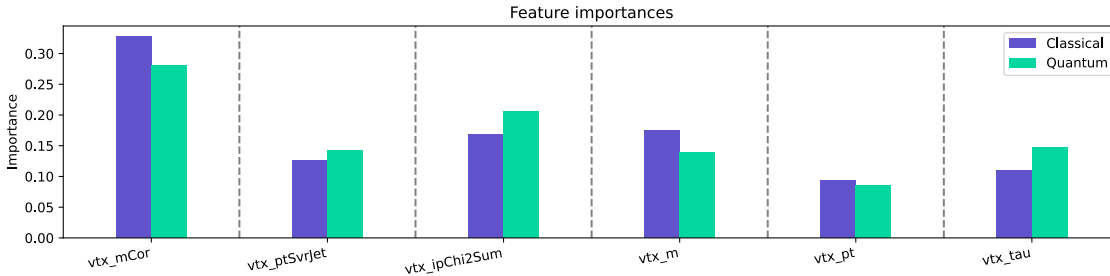


Figure 5.7: Feature importance of 6 features of the SV data set computed with the Quantum method and classical method.

5.3 Correlation from entanglement

Similar to the process employed for obtaining feature importance, the correlation between two variables can be evaluated using derived quantities from the characteristic wavefunctions. To facilitate this analysis, a distinct set of features has been selected to accentuate the assessment of correlations. This choice was made to enhance the detection of correlations, given that the prior feature set exhibited minimal feature correlations.

The updated set of features for this section is: `vtx_mCor`, `vtx_ipChi2Sum`, `vtx_nTrk`, `vtx_nTrkJet`, `vtx_ptSvrJet`, and `vtx_pt`.

As affirmed by this analysis, there exists a correlation between `vtx_nTrk` and `vtx_nTrkJet`, as well as between `vtx_ptSvrJet` and `vtx_pt`. All other features demonstrate a general lack of correlation.

Correlation between features has been computed with two standard methods: Pearson correlation and Classical mutual information, and a quantum method derived from the characteristic functions.

Pearson correlation coefficient: The Pearson correlation coefficient r is a statistical measure that quantifies the strength of a linear relationship between two variables.

The Pearson correlation coefficient ranges from -1 to 1, where:

- if $r = -1$ as one variable increases, the other decreases by a consistent predictable amount. It indicates a perfect *anticorrelation*;

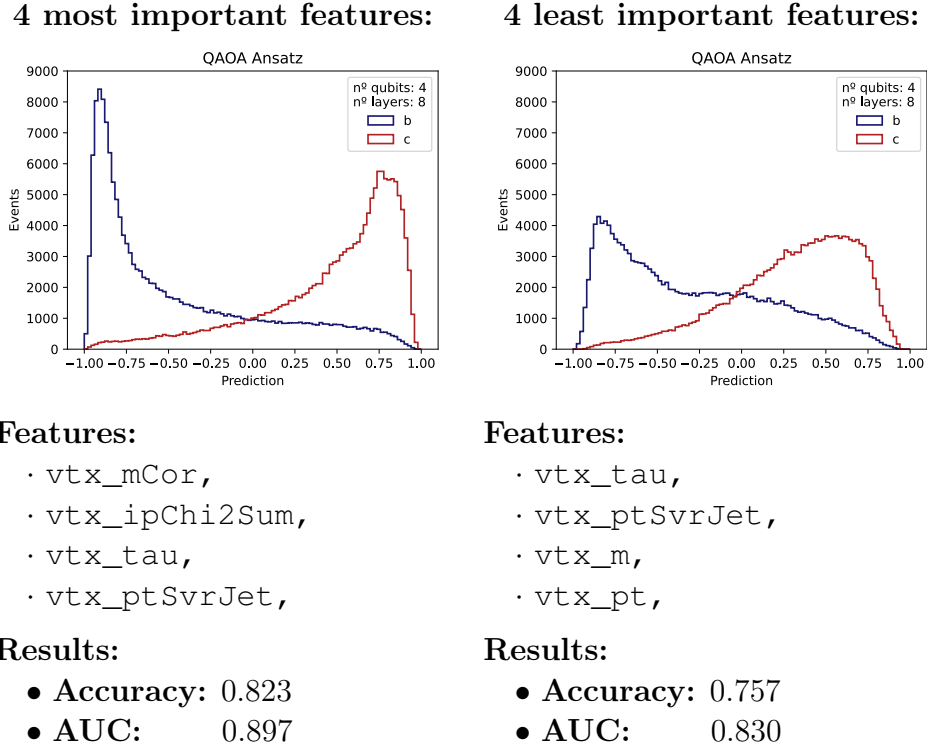


Figure 5.8: Comparison of two 4-qubits classifiers trained with the 4 most important features (left) and 4 least important features (right) among the six analyzed.

- if $r = 0$ it indicates no linear correlation: There is no discernible pattern between the variables' variations.
- if $r = 1$ the two features have perfect linear *correlation*: As one variable increases, the other also increases by a consistent predictable amount.

To assess the correlations between two SV features, one considers all values of the two features as two separate probability distributions J and K . The correlation between these features, denoted as r_{JK} , is then computed using the formula:

$$r_{JK} = \frac{\sum (J_i - \bar{J})(K_i - \bar{K})}{\sqrt{\sum (J_i - \bar{J})^2 \sum (K_i - \bar{K})^2}} \quad (5.3)$$

where:

- J_i and K_i are the data points of the feature distributions J and K
- \bar{J} and \bar{K} are the respective average values.

Classical Mutual Information: Another classical method inspected to compute the correlation among features is through the computation of classical mutual information. It comes from information theory, and it is a quantity that tells how much knowing the value of one variable reduces the uncertainty about the other variable. Similarly to the Pearson coefficient, it is computed assuming the values of the two features as two probability distributions and then using the following formula:

$$I(J, K) = \sum_i \sum_l p(J_i, K_l) \log \left(\frac{p(J_i, K_l)}{p(J_i) \cdot p(K_l)} \right) \quad (5.4)$$

where:

- $I(J, K)$ represents the mutual information between features J and K .
- $p(J_i, K_l)$ is the joint probability of observing values J_i and K_l for features J and K .
- $p(J_i)$ and $p(K_l)$ are the marginal probabilities of observing values J_i and K_l for features J and K , respectively.

Quantum Mutual information: The quantum approach for calculating correlation involves utilizing the quantum version of mutual information. Up to this point, entropy has been utilized as a measure of the level of entanglement between two subsystems that together constitute the entire quantum system. When assessing the extent of entanglement specifically between two qubits within a larger quantum circuit, one can employ the concept of quantum mutual information.

The mutual information between two qubits, denoted as qubit A and qubit B , is a measure of the amount of information they share and is defined using the Von Neumann entropy.

It is expressed as:

$$MI(A : B) = S(\rho_A) + S(\rho_B) - S(\rho_{AB}) \quad (5.5)$$

where:

- $S(\rho_A)$ is the entropy between the qubit A and the complementary subsystem;
- $S(\rho_B)$ is the entropy between the qubit B and the complementary subsystem;
- $S(\rho_{AB})$ is the entropy between the system containing the qubits A and B , and the complementary subsystem

5.3. CORRELATION FROM ENTANGLEMENT

In the context of quantum information, this formula can be interpreted as follows: the information shared between the two qubits, is the information shared between qubit A and the rest of the system, in addition to the information shared between qubit B and the rest, subtracting the information shared between the two qubits along with the complementary subsystem. This concept follows a similar idea to the formula for the probability of the union between two sets: $P(A \cup B) = P(A) + P(B) - P(A \cap B)$.

The mutual information allows us to quantify the correlations and entanglement between the two qubits in the larger quantum circuit. Positive mutual information indicates correlated or entangled qubits, while zero mutual information implies uncorrelated qubits.

In the context of the characteristic wavefunction, where each qubit represents a feature, computing the correlation between qubits using mutual information allows us to infer the correlation between features.

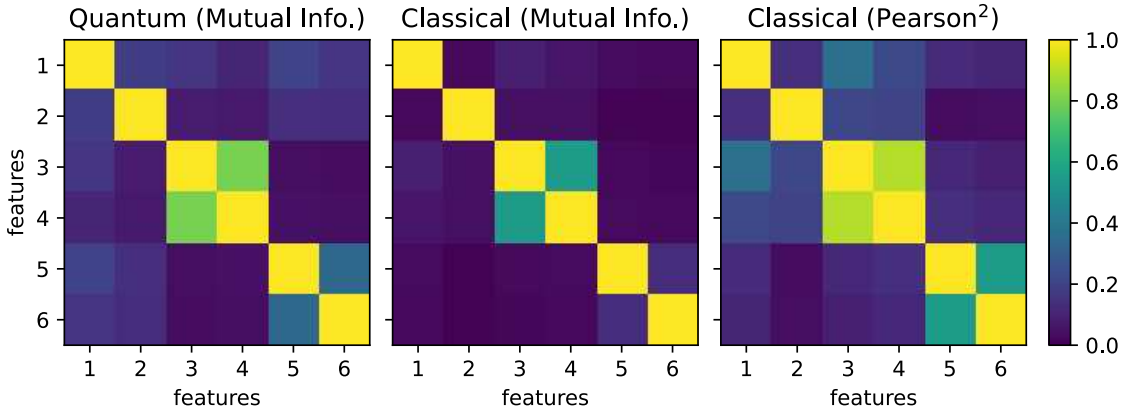
Figure 5.9 displays the application of the three metrics explained above to examine data correlations. Both quantum and classical mutual information have been normalized to span the range from 0 to 1. As for the Pearson coefficient, its squared value has been taken to fall within the same 0 to 1 range as the other two metrics, for an easy comparison.

All the models seem to have successfully captured the expected correlations discussed at the beginning of this section. Particularly, a strong correlation between `vtx_nTrk` and `vtx_nTrkJet`, as well as between `vtx_ptSvrJet` and `vtx_pt`, have been identified. However, the classical mutual information encountered difficulty in identifying the correlation between the latter pair of features, while the other model effectively managed it.

Among the three, the quantum approach appears to be the most balanced. It manages to capture all the correlations anticipated based on physics insights, while also mitigating potential weak false correlations between other pairs of features which could possibly arise due to the data set having a limited number of samples.

The model however did not identify any novel correlations that differ from those detectable by classical approaches.

While it remains possible that a slightly altered implementation or a different data



Features:

- | | |
|-------------------------|------------------------|
| 1: vtx_mCor | 4: vtx_nTrkJet |
| 2: vtx_ipChi2Sum | 5: vtx_ptSvrJet |
| 3: vtx_nTrk | 6: vtx_pt |

Figure 5.9: Correlation of the features using the fully quantum method (left), classical mutual information (middle), and the Pearson coefficient (right).

set might achieve that result, a significant advancement in this framework could be achieved through the future integration of quantum sensing and quantum memories. This advancement could facilitate the analysis of an event without first converting it into classical data. This approach could potentially set a standard for the analysis of complex quantum correlations for subsequent implementations that handle true quantum data.

The application of the characteristic wavefunctions methodology might be able to unlock new applications beyond just the computation of the feature importance and correlations. One example can be the generation of a tailored Ansatz for the characteristic circuits with the assistance of mutual information. This entails the application of multi-qubit gates to qubits (features) exhibiting strong correlations while restricting the application of these gates to uncorrelated qubits. As discussed in Chapter 4, limiting the number of multi-qubit gates narrows down the space of the problem leading to a better convergence and training efficiency overall. Moreover, the scalability of calculating quantum correlations extends easily to the computation of 3-point correlations. This could potentially unveil novel data patterns and provide fresh perspectives on the formation of jets, which is yet not fully understood.

Conclusions

HEP stands as one of the best-proving grounds for testing the capabilities of Quantum Computation and in particular QML [6]. Within this thesis, QML models have been employed on simulated data for the task of *jet tagging* heavy jets. First, a quantum classifier has been implemented, inspecting how different choices of hyperparameters affect the classification accuracy. Then novel models have been developed that leverage the entanglement between qubits to gather useful information for the input data.

The outcomes can be categorized into three distinct parts:

- **Quantum Classifier for Jet Tagging:** A quantum model able to distinguish between b - and c -jets has been developed.

Multiple evaluations of the performances have been conducted by varying the number of loaded features and by altering the number of trainable parameters. The developed quantum classifiers have been compared to the state-of-the-art classical classifiers for jet tagging at LHCb (the BDTs), revealing just slightly lower performances in terms of classification capability.

While it is true that the quantum models had lower performances than their classical counterparts, it is important to acknowledge that they were considerably limited by the simulator employed. With advancements in optimization strategies and the implementation of the model on real quantum hardware, featuring 20 or more qubits, it is expected a remarkable enhancement in the performance. This progress is projected to eventually surpass the capabilities of the current BDT.

- **Study of Entropy productions:** One of the main challenges to overcome

in the field of QML is the presence of the *Barren Plateaus*. In the thesis, this phenomenon has been linked to the capability of a model to build up entanglement. By monitoring the production of entanglement in quantum models dedicated to *jet tagging*, it has been possible to study more accurately the current strategies to limit the presence of these Plateaus, leading to slightly better values of accuracy.

- **Entropy-based models:** From the study of entropy production, a model has been developed that yields meaningful information in the amount of entanglement between the qubits. Through these values, it has successfully computed the feature importance and the correlation among the SV features with a full-quantum approach. These two methodologies produced results comparable to established classical counterparts.

A consideration must be made about its functioning. This model depends on the ability to compute the entropies, whose values can be obtained through the computation of the wavefunction. However, for larger scales than implemented and on real quantum hardware, the access of the wavefunction becomes unfeasible. Improvements to this model can be made to overcome this limitation, such as the C-Swap test [60], which can make possible the computation of the entropy without direct access to the wavefunctions.

These entropy-based models are also expected to yield more compelling results when applied to real quantum data. This can be realized with the future development of quantum sensors and quantum memories.

C.1 Future developments in heavy jet tagging

The SV data set presents a very challenging set to investigate, as it is the result of complex physics phenomena and multiple steps of processing.

One of the objectives set for this thesis was to investigate whether a quantum model would be able to extract some hidden information from the SV data invisible to a classical model. In the study of the SV data set in Chapter 5, no significant discrepancies were observed from a classical analysis concerning the feature importance and correlations.

However, it is important to note that this analysis has been conducted on a simulated data set. Monte Carlo events might not reproduce faithfully all the intricacies involved in the real events due to the still limited knowledge behind these complex

physical phenomena.

One hypothesis is that the study on the data set of real events might potentially reveal more pronounced divergences between classical and quantum models which can be attributed to hidden correlations visible only by the quantum model.

In the domain of jet tagging, the integration of quantum models into the jet reconstruction process has the potential to enhance tagging accuracy. Firstly, concerning particles identification and SV detection, as elucidated in Chapter 2. The incorporation of quantum models could offer advantages in finding complex patterns within the data set by harnessing their high expressive capabilities.

Likewise, accurate jet clustering significantly impacts the resulting tagging power of the classifier. Instances where algorithms encounter difficulties during the pre-classification stages lead to slight deviations in the data from their actual values. In this regard, quantum models could be employed to either replace or complement the existing anti- k_t clustering algorithm.

A well-executed implementation of a quantum pipeline is expected to offer a degree of advantage. This can be verified quantitatively by evaluating the tagging performance using the same classifier since better data pre-processing leads to the production of higher quality data, thereby resulting in a better tagging efficacy of subsequent models.

One of the first outcomes of developing enhanced heavy jets classifiers is a better understanding of Higgs boson decays ($H \rightarrow b\bar{b}$ and $H \rightarrow c\bar{c}$). The same models can be employed to improve the identification of these decays, providing increased statistics and insights. Additionally, one important aspect not to overlook is the ability of the of quantum models of being more interpretable. This arises from their construction, as each circuit (unitary) is reversible, a property that comes from the foundations of quantum mechanics.

C.1. FUTURE DEVELOPMENTS IN HEAVY JET TAGGING

Bibliography

- [1] *LHC Run 3: An opportunity to expand the Large Hadron Collider's physics programme*. URL: <https://home.cern/press/2022/run-3>.
- [2] URL: <https://home.cern/science/accelerators/high-luminosity-lhc>.
- [3] Donatella Lucchesi, Carlos Cocha, and Lorenzo Sestini. “Study of b-and c-jets identification with quantum machine learning algorithms and application to the Higgs reconstruction.” In: ().
- [4] Alessio Gianelle et al. “Quantum Machine Learning for b-jet charge identification”. In: *arXiv preprint arXiv:2202.13943* (2022).
- [5] Wen Guan et al. “Quantum machine learning in high energy physics”. In: *Machine Learning: Science and Technology* 2.1 (2021), p. 011003.
- [6] Alberto Di Meglio et al. “Quantum Computing for High-Energy Physics: State of the Art and Challenges. Summary of the QC4HEP Working Group”. In: *arXiv preprint arXiv:2307.03236* (2023).
- [7] Mengnan Du, Ninghao Liu, and Xia Hu. “Techniques for interpretable machine learning”. In: *Communications of the ACM* 63.1 (2019), pp. 68–77.
- [8] Tatiana Likhomanenko et al. “LHCb topological trigger reoptimization”. In: *Journal of Physics: Conference Series*. Vol. 664. 8. IOP Publishing, 2015, p. 082025.
- [9] Vinicius Mikuni and Florencia Canelli. “Unsupervised clustering for collider physics”. In: *Physical Review D* 103.9 (2021), p. 092007.

BIBLIOGRAPHY

- [10] Viktoria Chekalina et al. “Generative models for fast calorimeter simulation: the LHCb case”. In: *EPJ Web of Conferences*. Vol. 214. EDP Sciences. 2019, p. 02034.
- [11] Stefano Carrazza and Frédéric A Dreyer. “Jet grooming through reinforcement learning”. In: *Physical Review D* 100.1 (2019), p. 014014.
- [12] Maria Schuld and Francesco Petruccione. *Supervised learning with quantum computers*. Vol. 17. Springer, 2018.
- [13] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).
- [14] Yunchao Liu, Srinivasan Arunachalam, and Kristan Temme. “A rigorous and robust quantum speed-up in supervised machine learning”. In: *Nature Physics* 17.9 (2021), pp. 1013–1017.
- [15] John Von Neumann. *Mathematische grundlagen der quantenmechanik*. Vol. 38. Springer-Verlag, 2013.
- [16] Claude Elwood Shannon. “A mathematical theory of communication”. In: *The Bell system technical journal* 27.3 (1948), pp. 379–423.
- [17] Pasquale Arpaia et al. “Machine learning for beam dynamics studies at the CERN Large Hadron Collider”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 985 (2021), p. 164652.
- [18] URL: <https://quantum-computing.ibm.com/composer/docs/iqx/guide/shors-algorithm>.
- [19] Christian L Degen, Friedemann Reinhard, and Paola Cappellaro. “Quantum sensing”. In: *Reviews of modern physics* 89.3 (2017), p. 035002.
- [20] Roel Aaij et al. “First measurement of the charge asymmetry in beauty-quark pair production”. In: *Physical Review Letters* 113.8 (2014), p. 082003.
- [21] Lyndon Evans and Philip Bryant. “LHC machine”. In: *Journal of instrumentation* 3.08 (2008), S08001.
- [22] URL: <https://home.cern/science/accelerators/large-hadron-collider>.
- [23] Georges Aad et al. “The ATLAS experiment at the CERN large hadron collider”. In: (2008).

BIBLIOGRAPHY

- [24] Roman Adolphi et al. “The CMS experiment at the CERN LHC”. In: *Jinst* 803 (2008), S08004.
- [25] Kenneth Aamodt et al. “The ALICE experiment at the CERN LHC”. In: *Journal of Instrumentation* 3.08 (2008), S08002.
- [26] Ewa Lopienska. *The CERN accelerator complex, layout in 2022*. Tech. rep. 2022.
- [27] A Augusto Alves Jr et al. “The LHCb detector at the LHC”. In: *Journal of instrumentation* 3.08 (2008), S08005.
- [28] URL: <https://lhcb-outreach.web.cern.ch/detector/calorimeters/>.
- [29] Eric M. Metodiev. *The fractal lives of jets*. Apr. 2020. URL: <https://www.ericmetodiev.com/post/jetformation/>.
- [30] Gavin P Salam. “Towards jetography”. In: *The European Physical Journal C* 67 (2010), pp. 637–686.
- [31] URL: <https://twiki.cern.ch/twiki/bin/view/CMSPublic/SWGuideParticleFlow>.
- [32] ATLAS collaboration et al. *Secondary vertex finding for jet flavour identification with the ATLAS detector*. Tech. rep. ATL-PHYS-PUB-2017-011, 2017.
- [33] Jonathan Shlomi et al. “Secondary vertex finding in jets with neural networks”. In: *The European Physical Journal C* 81.6 (2021), p. 540.
- [34] Torbjörn Sjöstrand, Stephen Mrenna, and Peter Skands. “A brief introduction to PYTHIA 8.1”. In: *Computer Physics Communications* 178.11 (2008), pp. 852–867.
- [35] Anders Ryd et al. “EvtGen: a Monte Carlo generator for B-physics”. In: *BAD* 522 (2005), p. v6.
- [36] Sea Agostinelli et al. “GEANT4—a simulation toolkit”. In: *Nuclear instruments and methods in physics research section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 506.3 (2003), pp. 250–303.
- [37] Song Cheng et al. “Tree tensor networks for generative modeling”. In: *Physical Review B* 99.15 (2019), p. 155131.
- [38] LHCb Collaboration et al. “Identification of beauty and charm quark jets at LHCb”. In: *Journal of Instrumentation* 10.06 (2015), P06013.

- [39] Haowen Deng et al. “Ensemble learning for the early prediction of neonatal jaundice with genetic features”. In: *BMC medical informatics and decision making* 21 (2021), pp. 1–11.
- [40] Ville Bergholm et al. “Pennylane: Automatic differentiation of hybrid quantum-classical computations”. In: *arXiv preprint arXiv:1811.04968* (2018).
- [41] James Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13. 2018. URL: <http://github.com/google/jax>.
- [42] Nikolaj Moll et al. “Quantum optimization using variational algorithms on near-term quantum devices”. In: *Quantum Science and Technology* 3.3 (2018), p. 030503.
- [43] Adrián Pérez-Salinas et al. “Data re-uploading for a universal quantum classifier”. In: *Quantum* 4 (2020), p. 226.
- [44] In-Kwon Yeo and Richard A Johnson. “A new family of power transformations to improve normality or symmetry”. In: *Biometrika* 87.4 (2000), pp. 954–959.
- [45] James P Egan, Gordon Z Greenberg, and Arthur I Schulman. “Operating characteristics, signal detectability, and the method of free response”. In: *The Journal of the Acoustical Society of America* 33.8 (1961), pp. 993–1007.
- [46] Wikimedia Commons. *roc curve*. 2007. URL: https://upload.wikimedia.org/wikipedia/commons/1/13/Roc_curve.svg.
- [47] Sukin Sim, Peter D. Johnson, and Alán Aspuru-Guzik. “Expressibility and Entangling Capability of Parameterized Quantum Circuits for Hybrid Quantum-Classical Algorithms”. In: *Advanced Quantum Technologies* 2.12 (Oct. 2019), p. 1900070. DOI: 10.1002/qute.201900070. URL: <https://doi.org/10.1002/qute.201900070>.
- [48] Jarrod R McClean et al. “Barren plateaus in quantum neural network training landscapes”. In: *Nature communications* 9.1 (2018), p. 4812.
- [49] Iris Cong, Soonwon Choi, and Mikhail D Lukin. “Quantum convolutional neural networks”. In: *Nature Physics* 15.12 (2019), pp. 1273–1278.
- [50] Arthur Pesah et al. “Absence of barren plateaus in quantum convolutional neural networks”. In: *Physical Review X* 11.4 (2021), p. 041011.
- [51] Kaining Zhang et al. *Escaping from the Barren Plateau via Gaussian Initializations in Deep Variational Quantum Circuits*. 2022. arXiv: 2203.09376 [quant-ph].

- [52] James C Spall. “A one-measurement form of simultaneous perturbation stochastic approximation”. In: *Automatica* 33.1 (1997), pp. 109–112.
- [53] Marco Ballarín et al. “Entanglement entropy production in Quantum Neural Networks”. In: *arXiv preprint arXiv:2206.02474* (2022).
- [54] Alfred Haar. “Der Massbegriff in der Theorie der kontinuierlichen Gruppen”. In: *Annals of mathematics* (1933), pp. 147–169.
- [55] Thomas Storwick. *Alleviating barren plateaus with local cost functions*. Jan. 2021. URL: https://pennylane.ai/qml/demos/tutorial_local_cost_functions.
- [56] M. Cerezo et al. “Cost function dependent barren plateaus in shallow parametrized quantum circuits”. In: *Nature Communications* 12.1 (Mar. 2021). DOI: 10.1038/s41467-021-21728-w. URL: <https://doi.org/10.1038/s41467-021-21728-w>.
- [57] Timo Felser et al. “Quantum-inspired machine learning on high-energy physics data”. In: *npj Quantum Information* 7.1 (2021), p. 111.
- [58] Jia Deng et al. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.
- [59] Leo Breiman. “Random forests”. In: *Machine learning* 45 (2001), pp. 5–32.
- [60] Steph Foulds, Viv Kendon, and Tim Spiller. “The controlled SWAP test for determining quantum entanglement”. In: *Quantum Science and Technology* 6.3 (2021), p. 035002.
- [61] Duarte Magano et al. “Quantum speedup for track reconstruction in particle accelerators”. In: *Physical Review D* 105.7 (2022), p. 076012.
- [62] E Knuth Donald et al. “The art of computer programming”. In: *Sorting and searching* 3.426-458 (1999), p. 4.
- [63] Matthias Troyer. *Quantum Advantage: Hope and hype*. May 2023. URL: <https://cloudblogs.microsoft.com/quantum/2023/05/01/quantum-advantage-hope-and-hype/>.
- [64] John Von Neumann. *Mathematical foundations of quantum mechanics: New edition*. Vol. 53. Princeton university press, 2018.
- [65] Erik Gustafson. “Noise improvements in quantum simulations of sqed using qutrits”. In: *arXiv preprint arXiv:2201.04546* (2022).

BIBLIOGRAPHY

Chapter 6

Appendix

A.1 Quantum Computing

Quantum computation is a new interdisciplinary field that combines Computer Science and Quantum Mechanics. At its core, Quantum Computation aims to leverage the precise evolution of quantum systems, utilizing multiple quantum bits (qubits) to gain a computational advantage by harnessing quantum mechanical phenomena, such as quantum entanglement, to our benefit.

This following additional chapter will delve into the fundamental principles of Quantum Computation, starting from the basic unit of information, *the qubit* and progressing towards the construction of the first quantum circuits.

A.1.1 Quantum Advantage

In 1995, the field of Quantum Computation received a significant boost when P. Shor formulated a quantum algorithm capable of solving the factorization problem, resulting in faster executing times for large numbers than for any classical algorithms. Subsequently, in 1996, L. Grover introduced a quantum algorithm that offered a significant *speedup* for unstructured searches in databases. These two fundamental algorithms garnered substantial interest within the scientific community. Shor's factorization algorithm exhibited the potential to factorize large numbers exponentially

faster than any known classical methods, thereby challenging the security of systems reliant on the difficulty of factoring large numbers. On the other hand, Grover's algorithm found applications in various tasks, such as pattern recognition, cryptographic key breaking, and optimization problem-solving. As an example, the High Energy Physics community has been exploring the adaptation of Grover's algorithm for the task of track reconstruction [61]. The remarkable progress made by these algorithms has led to a surge of research interest and exploration of their applications across different domains.

Complexity of an algorithm

The main advantage of quantum computers over classical computers lies in their ability to harness quantum properties to execute exclusive *quantum* algorithms which might be able to solve given tasks faster than their classical counterparts. To evaluate the effectiveness of these algorithms, *complexity* measures are employed for comparison.

Computational complexity of an algorithm describes the amount of time an algorithm requires to solve a particular task. It is typically evaluated by counting the number of elementary operations involved. If an algorithm demands a constant number of operations irrespective of the input size, its complexity is considered *constant*. On the other hand, if the time taken by the algorithm increases proportionally with the input size, the complexity can be classified as *linear*, *linearithmic*, *polynomial*, *exponential*, or even worse, depending on the rate at which the time taken (number of operations) scales, as shown in Figure 6.1 and Table 6.1.

As two examples:

- An algorithm that outputs whether a given integer number is either *even* or *odd*, can work at a constant time no matter the number of digits that the input number has, since it only has to check whether the last digit is even or odd.
- An algorithm that outputs whether a given integer number contains the digit 0 in it, works in linear time since in the worst case scenario, it has to check every digit, up until the last one.

Additionally, the same task can be performed in different computational times given the algorithm used. As an example, there is a wide selections of different *sorting*

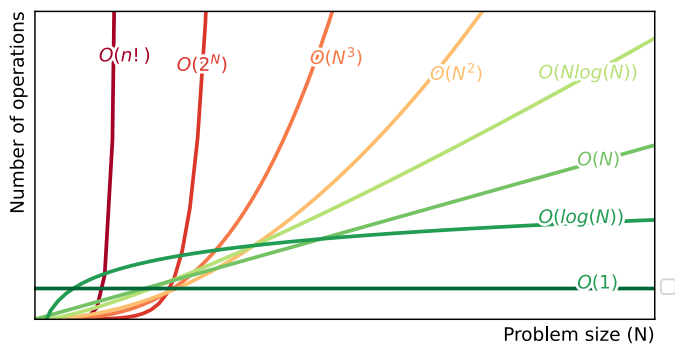


Figure 6.1: Most common classes of algorithms with Big O notation.

Big O Notation	Name
$O(1)$	Constant
$O(\log(N))$	Logarithmic
$O(N)$	Linear
$O(N \log(N))$	Linearithmic
$O(N^m)$	Polynomial
$O(k^N)$	Exponential
$O(N!)$	Factorial

Table 6.1: Big O notation and names of the most common classes of algorithms.

algorithms, the most basic implementation *the exchange sort* scales as $O(n^2)$, while more advanced implementations manage to reduce the scaling to $O(n \log(n))$ [62].

Quantum Computation introduces a distinct framework for constructing novel algorithms, rooted in the principles of Quantum Mechanics. As demonstrated by Shor and Grover, there are situations where a Quantum Algorithm may outperform its classical counterpart for specific tasks. However, not all tasks will necessarily benefit from a Quantum Algorithm over classical approaches.

The Advantage of Quantum Algorithms over existing classical counterparts lies in their better scaling for tasks where they are expected to outperform.

For instance, for the case of factorization algorithms, when transitioning from classical to Quantum Algorithms, the scaling reduces to *polynomial* ($const * d^3$) instead of the previous exponential scaling ($\exp(const * d^{\frac{1}{2}})$) considering the best classical

A.1. QUANTUM COMPUTING

factorization algorithm, as shown in Figure 6.3.

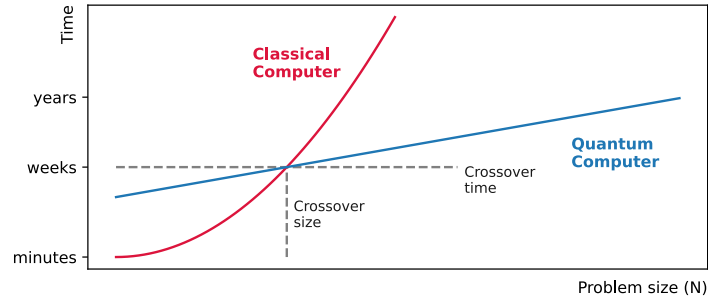


Figure 6.2: Advantage for Quantum Algorithms, after the problem size is bigger than the *Crossover size*, quantum algorithms become advantageous. [63]

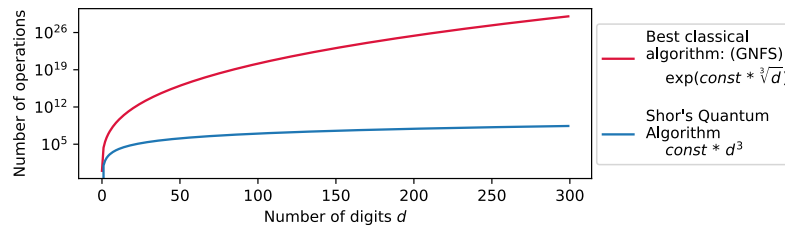


Figure 6.3: Complexity of Shor's algorithm and its best classical counterpart. [18]

A.1.2 The Qubit

In Classical Information theory, the unit of information is the *bit* which represents a logic state that can have only two possible values: 0 (or *false*, *off*, etc.) or 1 (or *true*, *on*, etc.). Single bits are concatenated into strings to be able to represent an exponentially larger number of different states: an n -bits string can represent a maximum of 2^n different configurations, e.g the task of counting integers from 0 to 1000 can be accomplished using at least 10 bits since $2^{10} = 1024 > 1000$.

Similarly in Quantum Computing, the elementary information unit is called *qubit*, a two-level quantum system that can be measured in two states $|0\rangle$ (ground-state) and $|1\rangle$ (first excited state), which form an orthonormal basis, called *computational*

basis, of the 2-dimensional Hilbert space \mathcal{H} .

A generic state $|\psi\rangle$ of a single qubit can be written as:

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle \quad (6.1)$$

where $\alpha_0, \alpha_1 \in \mathbb{C}$, satisfying the normalization condition $|\alpha_0|^2 + |\alpha_1|^2 = 1$.

The main difference with classical computation is that at any given time, a qubit does not have to be in either one of the two states ($|0\rangle$ or $|1\rangle$) but it can be in any proportions of both states at once. This behaviour is exclusive to quantum systems and it is called *superposition*. However, in the instance the qubits are measured, they collapse to either one of the two states with probabilities depending on the coefficients α .

The 2-dimensional Hilbert space \mathcal{H} is isomorphic to the \mathbb{C}^2 vector space, so we can conveniently associate the canonical basis of \mathbb{C}^2 to the computational basis of \mathcal{H} :

$$|0\rangle \in \mathcal{H} \rightarrow \begin{pmatrix} 1 \\ 0 \end{pmatrix} \in \mathbb{C}^2 \quad (6.2)$$

$$|1\rangle \in \mathcal{H} \rightarrow \begin{pmatrix} 0 \\ 1 \end{pmatrix} \in \mathbb{C}^2 \quad (6.3)$$

This isomorphism allows us to represent quantum states as complex (normalized) vectors and linear operators as complex matrices.

The equation 6.1 hints at the potential of manipulating information using qubits over classical bits, as the former can carry a higher amount of information through its superposition properties. The potential of qubits becomes even more significant when multiple qubits interact, surpassing the standard classical bit concatenation.

Bloch sphere: A convenient way to graphically represent the 1-qubit state is through the Bloch sphere: the expression 6.1 can be rewritten as:

$$|\psi\rangle = \cos \frac{\theta}{2} |0\rangle + e^{i\phi} \sin \frac{\theta}{2} |1\rangle \quad (6.4)$$

where θ and ϕ are real number with $0 \leq \theta \leq \pi$ and $0 \leq \phi \leq 2\pi$. In a 3D sphere the angle θ corresponds to latitude and angle ϕ corresponds to a phase factor representing the longitude.

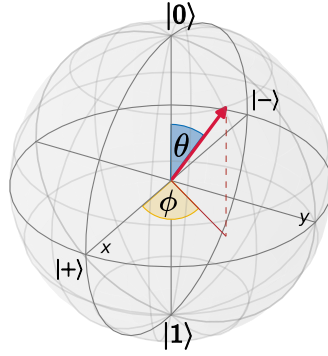


Figure 6.4: Bloch sphere.

With reference to the Figure 6.4, some noteworthy examples can be made:

- for $\theta = 0$, $\psi = |0\rangle$ regardless of the phase ϕ
- on the contrary, for $\theta = \pi$, $\psi = |1\rangle$ regardless of the phase ϕ
- while for $\theta = \pi/2$, there are different scenarios depending on the value of the phase ϕ :

- for $\phi = 0$, $|\psi\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ which is the state denoted as $|+\rangle$.
- for $\phi = \pi$, $|\psi\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$ which is the state denoted as $|-\rangle$

Representing the states

Density matrix and mixed states: The quantity obtained by the outer product of a state-vector $|\psi\rangle$ is called *density matrix* of the pure state (ρ_{pure}) and represents a projector operator:

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle \quad \rho_{\text{pure}} = |\psi\rangle\langle\psi| = \begin{pmatrix} |\alpha_0|^2 & \alpha_0\alpha_1^* \\ \alpha_0^*\alpha_1 & |\alpha_1|^2 \end{pmatrix} \quad (6.5)$$

ρ_{pure} has the following properties:

- $\rho_{\text{pure}} = \rho_{\text{pure}}^\dagger$ (ρ is Hermitian)
- $\forall|\phi\rangle : \langle\phi|\rho_{\text{pure}}|\phi\rangle \geq 0$ (ρ is positive semi-defined)

- $\text{Tr}(\rho_{\text{pure}}) = 1$ (the diagonal terms $|\alpha_i|^2$ of ρ_{pure} are the probabilities of measuring $|\psi\rangle$ in the respective basis state and, being probabilities, they sum up to 1)
- $\rho_{\text{pure}}^2 = \rho_{\text{pure}}$ (ρ is a projector)

The density matrix is particularly useful when dealing with quantum states that are not in pure but rather in mixed states which is a statistical mixture of different pure states. Additionally, density matrix essential for computing the *amount of entanglement* among qubits in that given state [64].

Multi-qubit states: The description of a multi-qubit quantum system can be easily gained using the concept of *tensor product* between Hilbert spaces. An n -qubit quantum system is described by the tensor product of n single qubit states:

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \dots \otimes |\psi_n\rangle \quad |\psi_1\rangle, |\psi_2\rangle, \dots, |\psi_n\rangle \in \quad |\psi\rangle \in \bigotimes_{i=1}^n \mathcal{H} \quad (6.6)$$

A completely generic n -qubits state is described by 2^n complex coefficients:

$$|\psi\rangle = \alpha_1|0\dots00\rangle + \alpha_2|0\dots01\rangle + \alpha_3|0\dots10\rangle + \dots + \alpha_{2^n}|1\dots11\rangle \quad \alpha_i \in \mathbb{C} \quad (6.7)$$

Where each coefficient is related to the probability of measuring the state in that particular basis state: $|\alpha_i|^2 \equiv P[\text{state } i]$.

Generalizing density matrices of pure and mixed states is straightforward: let $|\psi\rangle$ be an n -qubits state, then:

$$|\psi\rangle = \sum_{i=1}^{2^n} \alpha_i |i\rangle \quad \rho_{\text{pure}} = |\psi\rangle\langle\psi| \quad \rho_{\text{mix}} = \sum_{i,j} \alpha_{ij} |i\rangle\langle j| \quad \alpha_j, \alpha_{ij} \in \mathbb{C} \quad (6.8)$$

Quantum measurement: The state of a quantum system can be accessed only through measurements operated on the qubits of the system. Measurements does collapse the wavefunction in one of the eigenvectors with a probability given by its relative coefficient. For the sake of simplicity, let us assume to deal with a generic 1-qubit state as in the expression 6.1. In this case, the two possible outcomes of a measurement on the computational basis (0 or 1) are associated to the two projectors on the eigenspaces:

$$P_0 = |0\rangle\langle 0| \qquad P_1 = |1\rangle\langle 1|$$

Therefore, the probabilities associated to the two measurement outcomes are:

$$p(0) = \langle \psi | P_0 | \psi \rangle = |\alpha_0|^2 \qquad p(1) = \langle \psi | P_1 | \psi \rangle = |\alpha_1|^2$$

which again, they correspond to the modulus squared of the coefficients a_i .

So, the final quantum state gets destroyed upon measurement and it has to be rebuilt if repeated assessments are needed. This type of measurements on a system of qubits are called *projective measurements* on the computational basis.

The Qutrit

Other than qubits, additional units of information are considered in the advancement of Quantum Computing algorithms and hardware. In a practical implementation of a quantum computer, the states $|0\rangle$ and $|1\rangle$ represent distinct energy levels within the system (refer to A.1.4). Theoretically, there is no limitation in utilizing higher excited states alongside these (as in figure 6.5). The introduction of qutrits serves as an initial example of extending beyond the use of only the first two states.



$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle \Rightarrow |\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle + \alpha_2|2\rangle$$

Figure 6.5: Energy levels and general wavefunction of a qubit (on the left) and a qutrit (on the right).

The utilization of qutrits provides the benefit of accommodating a greater number of state combinations using an equivalent number of information units.

The advantage of employing qutrits becomes apparent when considering the increased capacity to represent an even greater number of combinations with the same number

of information units. This distinction becomes evident even when comparing just 2 qutrits over 2 qubits.

Qubits:	$ 00\rangle, 01\rangle, 10\rangle, 11\rangle$	2^2 (4) states
Qutrits:	$ 00\rangle, 01\rangle, 02\rangle, 10\rangle,$ $ 11\rangle, 12\rangle, 20\rangle, 21\rangle, 22\rangle$	3^2 (9) states

Table 6.2: Number of states for a pair of two qubits and two qutrits

Qutrits have been demonstrated to exhibit greater resilience to errors [65]. However, their implementation and manipulation pose significant challenges. Nevertheless, in the context of emerging Quantum Computer designs, the utilization of qutrits could potentially yield a even higher advantage.

A.1.3 Quantum Circuits

Single qubit gates

Gates acting on a single qubit are described by 2×2 complex unitary matrices acting on the \mathbb{C}^2 vector resulting in a rotation of the vector state in the Bloch sphere:

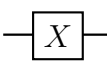
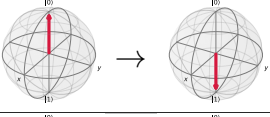
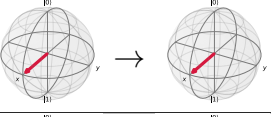
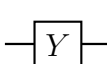
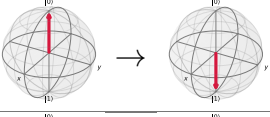
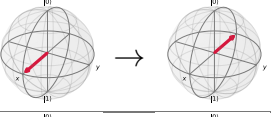
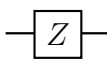
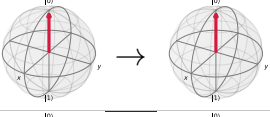
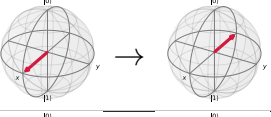
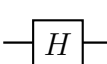
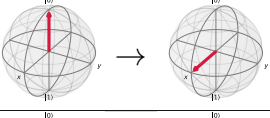
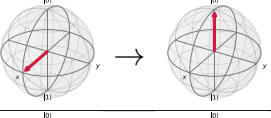
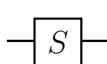
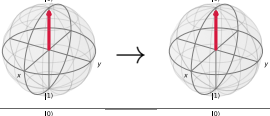
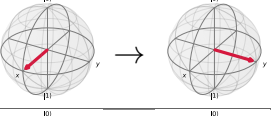
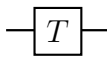
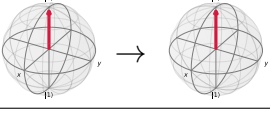
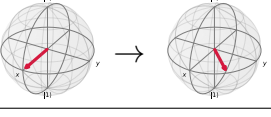
Name	Symbol	Matrix	$U 0\rangle$	$U +\rangle$
Pauli X		$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$		
Pauli Y		$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$		
Pauli Z		$\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$		
Hadamard		$\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$		
Phase gate		$\begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$		
$\pi/8$		$\begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$		

Table 6.3: Main single qubit gates. From left to right: the name, their symbolic representation in a quantum circuit, the matricial representation of the operator, the action of the gate on the state $|0\rangle$, the action of the gate on the state $|+\rangle$.

A generalization of the Pauli Gates above (Pauli X, Pauli Y and Pauli Z) are the parametrized rotations $RX(\theta)$, $RY(\theta)$, and $RZ(\theta)$ which rotate the state vector by an angle $\theta/2$ around the axes x, y and z respectively.

$$RX(\theta) = e^{-i\theta X/2} = \cos \frac{\theta}{2} \mathbb{I} - i \sin \frac{\theta}{2} X = \begin{pmatrix} \cos \frac{\theta}{2} & -i \sin \frac{\theta}{2} \\ -i \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{pmatrix} \quad (6.9)$$

$$RY(\theta) = e^{-i\theta Y/2} = \cos \frac{\theta}{2} \mathbb{I} - i \sin \frac{\theta}{2} Y = \begin{pmatrix} \cos \frac{\theta}{2} & -\sin \frac{\theta}{2} \\ \sin \frac{\theta}{2} & \cos \frac{\theta}{2} \end{pmatrix} \quad (6.10)$$

$$RZ(\theta) = e^{-i\theta Z/2} = \cos \frac{\theta}{2} \mathbb{I} - i \sin \frac{\theta}{2} Z = \begin{pmatrix} e^{-i(\theta/2)} & 0 \\ 0 & e^{i(\theta/2)} \end{pmatrix} \quad (6.11)$$

The three parametrized gates above represents the building blocks for Quantum Machine Learning. Infact just like classical Machine Learning, in Quantum Machine Learning one must find through an optimization procedure, the most optimal parameters set of parameters (weights of a Neural Network for classical Machine Learning, and parameters of these gates for Quantum Machine Learning) such that the output of the circuit is the most optimal one, given its input.

Other than the parametrized gates, one other crucial component for QML are the *Multi qubit gates*, which are gates that act on multiple qubits at once, building entanglement among the involved qubits. Entanglement makes the qubits achieve complex cooperative behaviours, which is a required phenomena for solving non-trivial problems.

Multi qubit gates

The interplay between qubits is created via multi-qubit gates. A two-qubit gate operates on both a *control qubit* and a *target qubit*.

The roles of control and target qubits reference the behaviour of the gates for the trivial cases with no superposition. Consider the truth table of the Controlled-NOT (CNOT) gate for example:

Input $ \text{control}\rangle \otimes \text{target}\rangle$	Output $ \text{control}\rangle \otimes \text{target}\rangle$
$ 0\rangle \otimes 0\rangle$	$ 0\rangle \otimes 0\rangle$
$ 0\rangle \otimes 1\rangle$	$ 0\rangle \otimes 1\rangle$
$ 1\rangle \otimes 0\rangle$	$ 1\rangle \otimes 1\rangle$
$ 1\rangle \otimes 1\rangle$	$ 1\rangle \otimes 0\rangle$

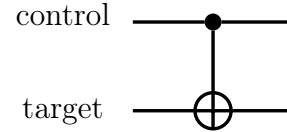


Table 6.4: Truth table of the CNOT gate.

In this case with no superposition, this gate applies a Pauli X (*bit-flip*) operation on the target gate if the control is $|1\rangle$, while leaving the control gate unchanged.

The Controlled-Not (CNOT) gate and the Controlled-Z (CZ) gate are the most common used multi qubit gates. The symbolic and matrix representations are shown in Table 6.5

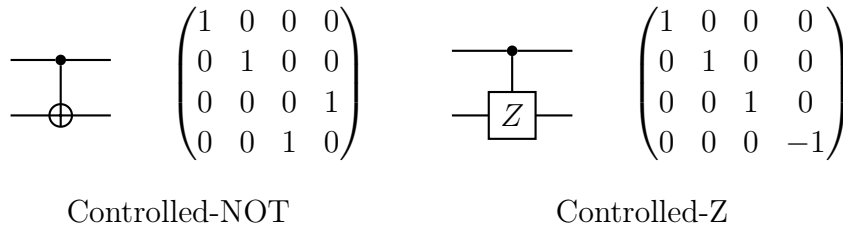


Table 6.5: The two most common controlled gates: on the left the Controlled-NOT (CNOT), on the right the Controlled-Z (CZ).

Universality

To construct a Quantum Computer capable of performing any conceivable operation, it is not necessary to include every gate discussed in the preceding sections. A set

of gates that allows for the (approximate) expression of any unitary operation is referred to as *universal*.

Some universal quantum gate sets are:

- The parametrized gates $RX(\theta)$, $RY(\theta)$, $RZ(\theta)$, the phase-shift gate $P(\varphi)$ and CNOT;
- The Hadamard Gate H , Phase Gate S , T-gate and CNOT;

Single qubit gate on multiple qubits

As shown in the preceding sections, a single qubit gate is represented by a 2×2 matrix due to its application on a two-component vector representing the qubit's state.

For example:

$$|0\rangle \text{ --- } \boxed{X} \text{ --- } |1\rangle \quad \rightarrow \quad X|0\rangle = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad (6.12)$$

Applying a single qubit gate to a specific qubit becomes ambiguous when multiple qubits are involved. The procedure to build the correct operator is quite similar for concatenating multiple quantum states as seen in the section A.1.2, some examples are:

$$\begin{array}{l} \text{---} \\ \boxed{X} \\ \text{---} \end{array} \quad \mathbb{I} \otimes X = \begin{bmatrix} 1 \times \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} & 0 \times \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \\ 0 \times \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} & 1 \times \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\begin{array}{l} \boxed{X} \\ \text{---} \\ \text{---} \end{array} \quad X \otimes \mathbb{I} = \begin{bmatrix} 0 \times \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & 1 \times \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \\ 1 \times \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} & 0 \times \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\begin{array}{l} \boxed{H} \\ \boxed{X} \\ \text{---} \end{array} \quad X \otimes H = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 \times \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} & 1 \times \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \\ 1 \times \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} & 0 \times \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \\ 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \end{bmatrix}$$

Example: Bell Circuit

With the tools shown in the last sections, it is possible to construct the Quantum Circuit that can evolve the starting state $|00\rangle$ into the Bell State $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. The following circuit is analyzed in the section 1.3 to explain the formation of entanglement from a Information Theory perspective.

$$\begin{array}{c} |0\rangle \\ |0\rangle \end{array} \text{---} \boxed{\text{Bell Circuit}} \text{---} \frac{|00\rangle + |11\rangle}{\sqrt{2}} \quad \hat{B}|00\rangle = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \quad (6.13)$$

The circuit capable of performing this operation is the following:

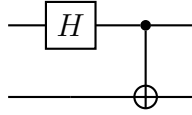


Figure 6.6: Circuit representation of the Bell circuit.

Which is matricial form is:

$$\begin{aligned}
 \hat{B} &= (CNOT)(H \otimes \mathbb{I}) \\
 &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & 1 \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ 1 \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & -1 \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{bmatrix} \\
 &= \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \end{bmatrix} \quad (6.14)
 \end{aligned}$$

Which applied to the state $|00\rangle$, it outputs the desired state:

$$\hat{B}|00\rangle = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \frac{|00\rangle + |11\rangle}{\sqrt{2}} \quad (6.15)$$

A.1.4 Types of Quantum Computers

Fundamentally, any quantum two-level system can be classified as a quantum computer. Currently, there are several distinct variations of quantum computers, each manipulating different quantum mechanical phenomena. As of the present moment, the main types of qubits are as follows:

- **Superconducting** The basic building block of a superconducting qubit is a Josephson junction. It consists of two superconducting materials at low temperature separated by a thin insulating barrier. The two distinct energy levels $|0\rangle$ and $|1\rangle$ correspond to the flow of current in the junction. Superposition is obtained by applying external control micro-wave signals and the gates are simulated by modifying the amplitude of the states with those microwave signals.
- **Photonic** The state of a qubit is encoded in the properties of a single photon, such as its polarization, and gates are simulated by optical devices.
- **Neutral atoms** The internal energy levels of neutral atoms serve as the qubits. Typically, two energy levels are chosen to represent the qubit states, such as the ground state and an excited state and they are manipulated by applying laser beams with specific frequencies and intensities.
- **Trapped ions** Qubits are implemented using the internal energy levels of individual ions that are trapped via electromagnetic fields and laser are used to carefully manipulate their state.
- **Quantum dots** Qubits are implemented using quantum dots, which are small semiconductor structures that can confine and manipulate electrons. The states are represented by the energy levels of electron that are confined within the quantum dot. This confinement creates discrete energy levels.

Currently, there is no universally superior implementation of quantum computers. Each type of quantum computer has its own set of advantages and disadvantages. For instance, superconducting quantum computers show promise in terms of scalability but necessitate extremely low temperatures (~ 1 mK) and are more susceptible to decoherence. On the other hand, photonic quantum computers do not require low temperatures, significantly reducing operational costs. However, establishing entanglement between photons has proven to be a challenging task.

Given the many types of quantum computer available The DiVincenzo criteria were

formulated to outline the key requirements for building a practical and functional quantum device.

DiVincenzo's criteria

DiVincenzo's criteria, proposed by David P. DiVincenzo in 2000, form a set of requirements that must be satisfied for the successful realization of a quantum computer. They consist of five essential rules:

1. A scalable physical system with well-characterized qubit.
2. The ability to initialize the state of the qubits to a simple fiducial state: ensuring the consistent behavior of a model heavily relies on the reliable production of its initial state. Typically, the desired starting state is the state in which each qubit is $|0\rangle$, which is achieved by allowing the system to anneal to its ground state.
3. Long relevant decoherence times: due to their inherent fragility, quantum states are highly susceptible to decoherence, resulting from interactions with the surrounding environment, collapsing the state and losing any "quantumness" to it. To effectively work with a quantum system, it is crucial to maintain its coherence for as long as possible, minimizing the detrimental effects of decoherence.
4. A "universal" set of quantum gates: as discussed before, an universal set of gates is required to freely manipulate the state.
5. A qubit-specific measurement capability.