Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias Físico Matemáticas

Quantum generative adversarial networks for data augmentation

Tesis presentada al

Colegio de Física

como requisito parcial para la obtención del grado de

Licenciado en Física

por

Joshua Devin Torres Tapia

Asesorado por

María Isabel Pedraza Morales
Enrique Varela Carlos

Puebla Pue.
December 12, 2025

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias Físico Matemáticas

Quantum generative adversarial networks for data augmentation

Tesis presentada al

Colegio de Física

como requisito parcial para la obtención del grado de

Licenciado en Física

por

Joshua Devin Torres Tapia

Asesorado por

María Isabel Pedraza Morales
Enrique Varela Carlos

Puebla Pue.
December 12, 2025

Título: Quantum generative adversarial networks for data augmentation
Estudiante: Joshua Devin Torres Tapia

COMITÉ

Humberto Antonio Salazar Ibargüen
Presidente

Haydee Hernández Arellano
Secretario

Jorge Velázquez Castro
Vocal

Ana Aurelia Avilez López
Suplente

María Isabel Pedraza Morales
Asesor

Enrique Varela Carlos
Asesor

Acknowledgments

First, I would like to thank my family, all my uncles, aunts and cousins who always helped me with whatever they could and motivated me to finish my bachelor's degree. Specials thanks to my mother, for whom I know how much sacrifices she had to make as a single mom. To my brother who did all he could to help while I was out of the country. To my first college friends Victor, Salvador, Rubén, the one I wish I knew sooner, Josué and the one who I wish we could had share a class, Ibáñez, you guys made my years in college more memorable.

To Dr. Pedraza, thank you for your patience, your guidance and for accepting me under your mentorship even though my circumstances and that you did not know me from any project or class before. I sincerely think you are one of the most kind and empathetic teachers I've had in my life and I hope to continue working with you in the future. Also thanks to her team, Dr. Haydee and all the persons who helped me correct mistakes I may have had or suggested different approaches to solve this problem.

I would also like to give special thanks to Natasha, who was always there to cheer me up and support me unconditionally. I will always cherish you and the impact you had in me, it's for you that I am a better man.
To Natasha's family who always made me feel welcomed. Also special mention to Manchitas and Lula who always made me company in those early morning meetings.

To my grandmother, who could not see me become a physicist.
To every single person who helped me become who I am today, thank you.

Abstract

In this thesis, we address a high-energy physics problem involving two types of events, signal and background, with 50 samples for each where signal events are gluons and background events quarks. Each sample has five features that define the type of event. A quantum generative adversarial network model is proposed, which is trained separately on each type of data. The goal is to replicate the features of each sample, thereby expanding the dataset where sample scarcity was a problem. For this purpose, we propose two very similar codes; the difference between them is that in one, only the samples classified as background are loaded, while in the other, only those classified as signal are loaded, the rest of the architecture is the same. The model is expected to successfully generate samples that imitate the features. If the model is capable of such a thing, then this means it is indeed possible to train a quantum generative adversarial networks model with a small amount of samples to generate realistic data. Another thing that is expected to be proven or refuted with this thesis is that, for this task, a pure quantum model (quantum generator and discriminator) is better than a hybrid one (quantum generator and classic discriminator). With this approach, we aim to demonstrate the usefulness of a quantum architecture in problems where data augmentation is required due to the limited amount of available data. The approach uses parameterized quantum circuits implemented with `PennyLane` and `PyTorch`, where both the generator and the discriminator are modeled using variational quantum circuits composed of `StronglyEntanglingLayers`.

Contents

List of Figures

Chapter 1

Introduction

Conducting experiments in fields such as particle physics entails grappling with vast and intricate datasets, presenting unique challenges in data handling and analysis.
In the last decade, machine learning has emerged as a pivotal tool in physical sciences, with deep learning increasingly applied to diverse issues in particle physics and astrophysics. In addition to conventional classical methods like boosted decision trees (BDT) and support vector machines (SVM), cutting-edge deep learning techniques including convolutional neural networks, recurrent models, and geometric deep learning are making significant strides across various tasks.
But there still are some scenarios where these algorithms are not capable to overcome the problem at hand, some of this common situations when the algorithms fail are when the dataset available is very small, because there is not enough statistics for the algorithm to extract information to work with, or the computational resources are just not enough, making the runtime very long and just not suitable for conducting experiments. Looking at the challenges that lie ahead of the scientific community, the ambitious High Luminosity LHC (HL-LHC) program over the next couple of decades and beyond will require immense computational resources, this prompts the inquiry into whether innovative technologies like quantum machine learning could potentially surmount this computational hurdle. The advent of quantum computing platforms and accessible simulators has catalyzed research on quantum algorithms and applications which have been proposed recently to address the computational complexities inherent in particle physics data processing and analysis. Moreover, quantum machine learning offers means to acquire quantum algorithms tailored for specific tasks, akin to classical machine learning, which means that it is possible to create quantum deep learning models that could be as or even more useful as their classic counterparts, this is the motivation behind this thesis, we will explore if a quantum algorithm is able to work with a small dataset and still obtain good results and discuss how can we optimize the computational resources if we switch to a quantum architecture.

1.1 General objective

This thesis aims to demonstrate the feasibility of implementing a quantum generative adversarial network model that can be trained with a small number of samples and still generate additional artificial data with features that mimic the real ones for both signal and background events.

1.2 Specific objective

- Understand the structure of the data.
- Organize signal and background events for use in network training.

- Develop a quantum generative adversarial network model that can be trained with both types of events.
- To determine whether a fully quantum architecture (quantum generator and discriminator) performs better or worse than a hybrid one (quantum generator and classical discriminator).
- Reproduce at least two features of each sample.

Chapter 2

Artificial Intelligence

Artificial Intelligence (AI) has revolutionized the way we approach complex problems in science, industry, and society. At its core, AI refers to the design of algorithms capable of mimicking or surpassing human cognitive abilities such as learning, reasoning, and decision-making [30]. Among the various subfields of AI, machine learning (ML) has emerged as the most impactful, enabling systems to improve performance through experience without being explicitly programmed.
In the context of high-dimensional and structured data, such as those arising in high energy physics, generative models have demonstrated strong capabilities in data synthesis, anomaly detection, and distribution modeling. Among them, Generative Adversarial Networks (GANs) have gained particular attention due to their ability to generate highly realistic data samples [11]. However, classical GANs face challenges when dealing with limited data, high computational costs, and non-Gaussian multimodal distributions, all of which motivate the exploration of hybrid quantum-classical architectures such as Quantum Generative Adversarial Networks (QGANs).

2.1 Machine learning

Machine Learning (ML) is a branch of Artificial Intelligence (AI) that focuses on creating algorithms and models enabling machines to learn from data and improve their performance over time without explicit programming [20]. It relies on statistical methods to identify patterns, make predictions, and automate decision-making.
Its applications span various fields, including image recognition, natural language processing, recommendation systems, financial modeling, and autonomous vehicles, making it a key driver of innovation across industries.
Machine learning entails programming computers to optimize performance criteria using example data or prior experience. It involves defining a model with specific parameters, and learning involves iteratively optimizing these parameters using training data or past experiences. Models can be predictive, facilitating future predictions, or descriptive, extracting insights from data.

Machine learning models classification

Machine learning models are categorized according to the information they have or the information they lack. The most general classification of machine learning has 3 categories:

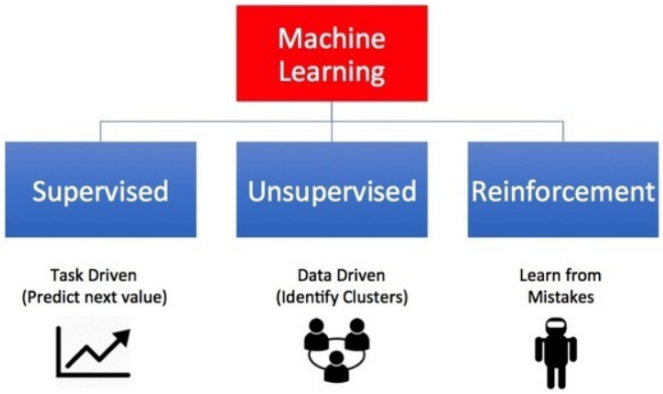


Figure 2.1: ML categories [33]

Supervised learning Supervised learning algorithms consist of a mathematical model that already has both the inputs and the desired outputs, meaning that it deals with labeled data. The data, known as training data, has a set of training examples and the desired outputs, also known as a supervisory signal. Some types of supervised-learning algorithms are classification and regression [12]. For example, in figure 2.2 we give the labeled data, which in this case are figures like rectangle or circle, and the labels for each figure to our ML model, then when we run the algorithm enough times the model "learns" the patterns and can now classify which figures are being introduced.

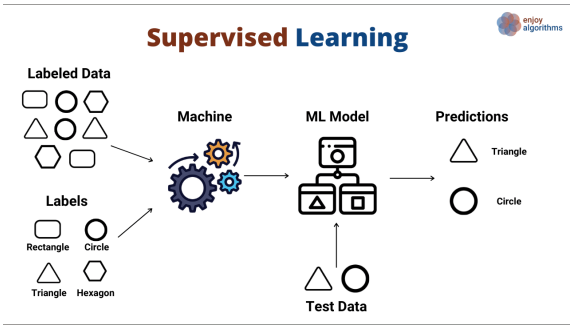


Figure 2.2: Supervised learning workflow [23]

Unsupervised learning Contrasting supervised learning, unsupervised learning algorithms find structures in data that do not have labels, classes or categories. Instead, the algorithms identify commonalities in the data and react based on the presence or absence of patterns. Some applications of these algorithms are density estimation, clustering and dimensionality reduction. A special type of unsupervised learning called self-supervised learning consists of training a model that generates the supervisory signal from the data itself instead of receiving it directly [5]. In figure 2.3 the input data for our model are apples and watermelons, but the model does not now either of this labels, because of this instead of grouping the data by labels it has to rely on other properties like statistics patterns, magnitude of the vectors, etc. to form groups that shares this traits. When this model is done, the apples have been correctly grouped only with other apples and the similar occurs with the watermelons.

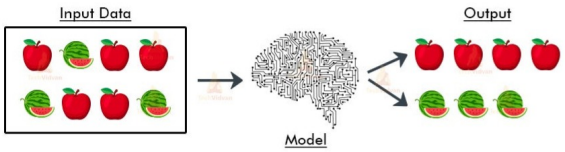


Figure 2.3: Unsupervised learning workflow [15]

Reinforcement learning This is an area of machine learning that makes software agents take actions in an environment with the objective to maximize some notion of cumulative reward. Due to its generality, it is studied in many disciplines, such as game theory, information theory, statistics, etc [32].

2.2 Data cleaning

Once the dataset is well understood, the next step is **data cleaning**, which ensures the dataset's quality before it is fed into any model. The specific cleaning procedures depend on the type and nature of the data, but for tabular data, the common tasks include:

- Handling missing values (NaN values), which may involve removing them or imputing them using statistical methods such as replacing them with the mean, median, or mode of the respective feature.
- Identifying and managing outliers to ensure they do not distort the model's learning. Outlier detection can use statistical measures like percentiles or interquartile range (IQR).
- Correcting data entry errors or inconsistencies to avoid any biases introduced by flawed data collection methods.

2.3 Deep learning

Deep Learning is a subfield of machine learning that leverages artificial neural networks (ANNs) with multiple layers to model complex patterns and hierarchical data representations. With the advent of powerful hardware (notably GPUs) and the availability of large datasets, deep learning has led to breakthroughs in computer vision, natural language processing, generative modeling, and scientific applications such as high energy physics and quantum computing [17].

2.3.1 Perceptron

The perceptron, introduced by Frank Rosenblatt in 1958 [28], is one of the earliest models of an artificial neuron. It serves as a linear binary classifier, learning a decision boundary that separates input data into two classes. The output of a perceptron is defined by the following equation:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

Each component of this expression plays a distinct role in the computation:

- x_i – **Input features:** These are the components of the input vector fed into the perceptron, representing the data to be classified.

- w_i – **Weights:** Each input x_i is associated with a weight w_i that quantifies its influence on the final output. The perceptron learns the optimal values of these weights during training to correctly classify the input data.
- b – **Bias term:** The bias allows the decision boundary to be shifted away from the origin, enabling the model to better fit the data by increasing its flexibility.
- $\sum_{i=1}^{n} w_i x_i + b$ – **Linear combination:** This expression computes the weighted sum of inputs plus the bias, representing the total signal received by the neuron before activation.
- $f(\cdot)$ – **Activation function:** This function determines the perceptron's output based on the linear combination. In the original model, a step function was used, returning 1 if the input is positive and 0 otherwise. Modern neural networks typically use smoother functions such as the sigmoid or ReLU to allow for gradient-based optimization.
- y – **Output:** The result after applying the activation function, representing the predicted class.

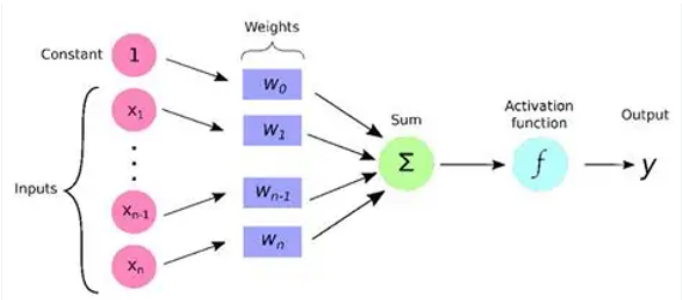


Figure 2.4: Perceptron structure

Activation functions

Activation functions introduce non-linearity into neural networks, allowing them to learn complex patterns and approximate non-linear decision boundaries. Without them, a neural network composed solely of linear operations would be equivalent to a single-layer model [10].
Activation functions are applied element-wise after the linear transformation in each neuron. The three most common activation functions in modern deep learning architectures are:

- **Sigmoid:**

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

  The sigmoid function maps any real-valued input into the interval $(0, 1)$. It was historically popular for binary classification but suffers from vanishing gradients for large input magnitudes.

- **Hyperbolic tangent (tanh):**

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

  This function maps inputs to the interval $(-1, 1)$ and is zero-centered, which often leads to better convergence than sigmoid. However, it can still suffer from vanishing gradients.

- **ReLU (Rectified Linear Unit):**

$$\text{ReLU}(x) = \max(0, x)$$

  ReLU is the most commonly used activation function due to its simplicity and computational efficiency. It avoids the vanishing gradient problem for positive values and introduces sparsity in the network. However, it can suffer from the "dying ReLU" problem when too many neurons output zero [21].

The choice of activation function can significantly influence the training dynamics and final performance of a model. Often, ReLU or its variants (e.g., Leaky ReLU, ELU) are used in hidden layers, while sigmoid or softmax functions are employed in output layers depending on the task (binary vs. multi-class classification) [9].

2.4 Artificial neural networks

To overcome the limitations of linear separability, multilayer perceptrons (MLPs) were developed. These consist of multiple layers of interconnected neurons, typically including:

- Input layer.
- One or more hidden layers.
- Output layer.

Each neuron applies a nonlinear activation function, such as the sigmoid, tanh, or ReLU (Rectified Linear Unit), to the weighted sum of its inputs. This allows the network to approximate complex, nonlinear functions.
Training a neural network involves minimizing a loss function (e.g., mean squared error or cross-entropy) using optimization algorithms like stochastic gradient descent (SGD). The backpropagation algorithm is used to compute gradients efficiently and update the weights during training [29].
Multilayer networks are capable of universal approximation, meaning they can approximate any continuous function given enough neurons and layers [13]. This expressiveness makes them suitable for a wide range of tasks, from classification to function regression and signal processing.

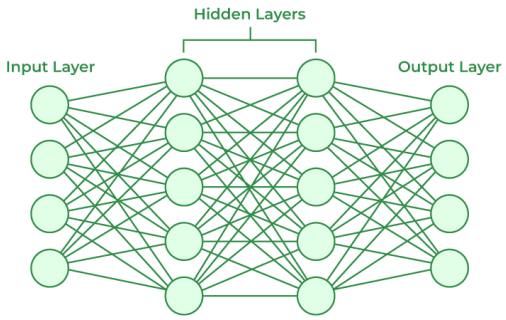


Figure 2.5: The three types of layers in an ANN

2.5 The learning process in artificial neural networks

The learning process of an artificial neural network (ANN) refers to the procedure by which the model adjusts its internal parameters (weights and biases) to minimize the difference between its

predicted outputs and the true labels. This process is iterative and composed of several key steps, described as follows.

Forward propagation

In forward propagation, the input vector $\mathbf{x}$ is passed through the layers of the network. Each neuron in a layer performs a weighted sum of its inputs followed by an activation function:

$$z^{(l)} = \mathbf{w}^{(l)} \cdot \mathbf{a}^{(l-1)} + b^{(l)} \tag{2.1}$$

$$\mathbf{a}^{(l)} = f\left(z^{(l)}\right) \tag{2.2}$$

where:

- $\mathbf{w}^{(l)}$ are the weights at layer l.
- $b^{(l)}$ is the bias term.
- $\mathbf{a}^{(l-1)}$ is the activation from the previous layer.
- $f(\cdot)$ is a non-linear activation function (e.g., ReLU, sigmoid, tanh).

This process continues until the output layer produces a prediction $\hat{y}$.

Loss function

The loss function $\mathcal{L}(y, \hat{y})$ quantifies the discrepancy between the predicted output $\hat{y}$ and the true target value y. It serves as a guiding signal during training, allowing the optimization algorithm to adjust the model's parameters in a direction that reduces the prediction error. Common choices of loss functions include:

- **Cross-entropy loss:** Often used for classification tasks, especially when the outputs represent class probabilities. It measures the difference between the true class distribution and the predicted probability distribution.
- **Mean Squared Error (MSE):** Commonly used in regression tasks.

Minimizing the chosen loss function is the main objective during training, as it directly drives the model to improve its predictions over time.

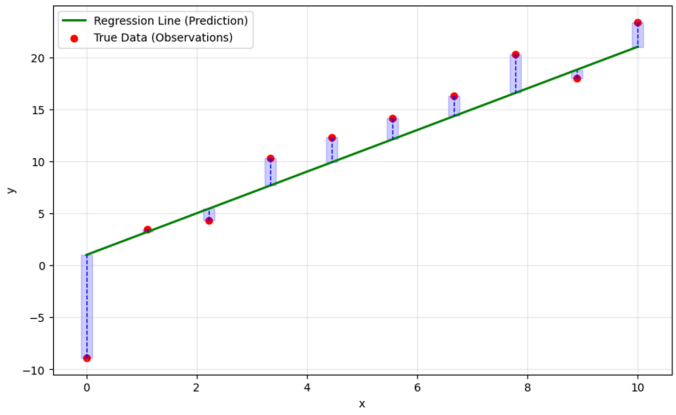


Figure 2.6: Visualization of Mean Squared Error

This figure illustrates the concept of the Mean Squared Error (MSE) loss function. Each point represents a data sample, and the vertical lines connecting the predicted and true values indicate the magnitude of the prediction error for that sample. The MSE is calculated as the average of the squares of these differences:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

By squaring the differences we obtain a positive number but also enables MSE to penalize larger errors more severely, encouraging the model to make predictions that are consistently close to the true values across the dataset.

Binary cross-entropy

In this thesis, the loss function used is **binary cross-entropy**, as the classification problem is binary, signal vs. background.
Binary cross-entropy is defined as:

$$\mathcal{L}_{\text{BCE}}(y, \hat{y}) = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right], \tag{2.3}$$

where $y \in \{0, 1\}$ is the true label (0 for background, 1 for signal), and $\hat{y} \in (0, 1)$ is the model's predicted probability that the input belongs to class 0 or 1 depending on the case.
This loss function penalizes confident but incorrect predictions more severely than uncertain or correct ones. If the predicted probability $\hat{y}$ is close to the true label y, the loss is small; if $\hat{y}$ is far from y, the loss is large. When $y = 1$, the second term vanishes, and the function focuses on making $\hat{y}$ close to 1, and vice versa when $y = 0$ the the first term vanishes. This encourages the model to produce well-calibrated probability estimates.
Binary cross-entropy is particularly well-suited for tasks involving generative models and binary discriminators, such as in Quantum Generative Adversarial Networks (QGANs), where the discriminator must distinguish between real and generated (fake) samples belonging to a single target class [11].

Backward propagation

After computing the loss, the network uses the backpropagation algorithm to calculate the gradient of the loss with respect to each parameter (weights and biases). This is done using the chain rule of calculus:

$$\frac{\partial \mathcal{L}}{\partial w_{ij}^{(l)}}, \quad \frac{\partial \mathcal{L}}{\partial b_i^{(l)}} \tag{2.4}$$

Gradients are propagated backward from the output layer to the input layer, allowing the network to understand how to modify its parameters to reduce the loss.

Optimization and weight update

Using the gradients computed in backpropagation, the parameters are updated using an optimization algorithm that adjust the model's parameters (weights and biases) to minimize the loss function during training. The most basic form is stochastic gradient descent (SGD):

$$w_{ij}^{\prime(l)} \leftarrow w_{ij}^{(l)} - \eta \cdot \frac{\partial \mathcal{L}}{\partial w_{ij}^{(l)}} \tag{2.5}$$

$$b_i'^{(l)} \leftarrow b_i^{(l)} - \eta \cdot \frac{\partial \mathcal{L}}{\partial b_i^{(l)}} \tag{2.6}$$

where η is the learning rate, which controls the size of the update steps.
Other optimization algorithms, such as Adam, RMSProp, and Momentum, improve convergence speed and stability using different strategies.

Optimization: the ADAM optimizer

Among several optimization methods, ADAM (Adaptive Moment Estimation) is widely used due to its efficiency and robustness [16]. Adam combines two key ideas:

- **Momentum:** Accumulates an exponentially decaying average of past gradients to smooth updates.
- **Adaptive learning rate:** Adjusts the learning rate for each parameter individually, based on the magnitude of past gradients, which helps us have a more stable training and faster convergence.

The Adam update rule for each parameter θ is:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\nabla_\theta L_t, \tag{2.7}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)(\nabla_\theta L_t)^2, \tag{2.8}$$

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \tag{2.9}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}, \tag{2.10}$$

$$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}. \tag{2.11}$$

Where:

- α is the learning rate.
- β_1 and β_2 are decay rates for the first and second moment estimates.
- ϵ is a small constant to prevent division by zero.
- m_t and v_t are the first and second moment estimates of the gradient.

The bias-corrected terms $\hat{m}_t$ and $\hat{v}_t$ compensate for the initialization of the moment estimates at zero, ensuring that early updates are not underestimated. In practice, the default hyperparameter values $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ are commonly used and work well across a wide range of applications.
Adam's ability to adaptively tune learning rates and handle noisy or sparse gradients makes it especially well-suited for training deep neural networks, including the quantum models explored in this thesis. Although Adam generally converges faster than stochastic gradient descent (SGD), some studies suggest that SGD with momentum may achieve better generalization in certain tasks.

Epochs and convergence

The entire dataset is passed through the network multiple times in cycles called *epochs*. With each epoch, the network's parameters ideally become better adjusted, reducing the training loss. Training continues until a stopping criteria is met, such as:

- The loss stops decreasing (convergence).
- A maximum number of epochs is reached.
- Performance on a validation set stops improving (early stopping).

Regularization and Generalization

To prevent overfitting, the following regularization techniques may be employed:

- **Dropout:** randomly deactivating a subset of neurons during training.
- **L2 regularization:** penalizing large weights.
- **Early stopping:** halting training when validation performance declines.

These techniques help the model generalize better to unseen data.

2.6 Generative adversarial networks (GANs)

Generative Adversarial Networks (GANs) are a class of deep learning models introduced by Ian Goodfellow in 2014 [11]. GANs are designed to learn the underlying distribution of data and generate new samples that resemble the training data. Unlike traditional neural networks, GANs consist of two competing neural networks trained simultaneously through an adversarial process.

Architecture of GANs

A typical GAN consists of two main components:

- **Generator (G):** A neural network that takes a random noise vector $\mathbf{z}$, sampled from a prior distribution (e.g., uniform or Gaussian), and generates synthetic data samples $G(\mathbf{z})$.
- **Discriminator (D):** A neural network that takes a data sample as input and outputs a probability indicating whether the sample is real (from the training data) or fake (produced by the generator).

The two networks are trained simultaneously in a zero-sum game where:

- The **generator** aims to produce samples that are indistinguishable from real data.
- The **discriminator** aims to correctly classify real and generated data.

2.6.1 Adversarial training process

The learning process of GANs involves the following steps:

1. **Training the discriminator:** Given a batch of real data samples $\mathbf{x}$ and generated samples $G(\mathbf{z})$, the discriminator is trained to maximize the probability of assigning the correct label (real or fake).
2. **Training the generator:** The generator is trained to minimize the discriminator's ability to distinguish fake samples from real ones, effectively "fooling" the discriminator.

The training objective can be formulated as the following minimax optimization problem:

$$\min_G \max_D V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log(1 - D(G(\mathbf{z})))] \tag{2.12}$$

Where:

- p_{data} is the distribution of the real data.
- $p_{\mathbf{z}}$ is the prior distribution of the noise vector.
- $D(\mathbf{x})$ represents the discriminator's output for real data.
- $D(G(\mathbf{z}))$ represents the discriminator's output for generated data.

Challenges in training GANs

Despite their conceptual simplicity, GANs are notoriously difficult to train. Some of the common challenges include:

- **Mode collapse:** The generator produces limited variations of samples, failing to capture the full diversity of the data distribution.
- **Non-convergence:** The adversarial process may not reach a stable equilibrium, leading to oscillations or divergence.
- **Vanishing gradients:** The discriminator may become too strong, providing no useful gradients to improve the generator.

Numerous variants of GANs have been proposed to address these issues, including Wasserstein GAN (WGAN), Least Squares GAN (LSGAN), and Deep Convolutional GAN (DCGAN).

Applications of GANs

GANs have found success in a variety of fields, including:

- **Image synthesis:** Generating highly realistic images of faces, landscapes, and objects.
- **Data augmentation:** Creating synthetic data to augment small datasets and improve model generalization.
- **Domain adaptation:** Translating data from one domain to another, such as converting sketches to photographs or daytime images to nighttime scenes.
- **Anomaly detection:** Learning the distribution of normal data to identify outliers.
- **Scientific modeling:** Simulating complex distributions in fields such as particle physics, chemistry, and medicine.

2.6.2 Limitations of classical GANs and the quantum perspective

Despite their powerful generative capabilities, classical GANs face limitations in contexts where:

- Data is scarce or expensive to obtain.
- Distributions are highly non-Gaussian, discrete, or multimodal.
- Computational resources are constrained.

These challenges have motivated the exploration of quantum machine learning, where quantum circuits exploit principles of superposition and entanglement to model complex probability distributions. Quantum Generative Adversarial Networks (QGANs) are a hybrid quantum-classical extension of GANs, leveraging quantum processors to improve generative modeling in such difficult regimes.

Summary

The learning process in an artificial neural network consists of 5 key steps:

1. Forward propagation to generate predictions.
2. Loss computation so we can measure the error.
3. Backpropagation to calculate gradients.
4. Optimization to update parameters.
5. Iteration over multiple epochs to converge to an optimal value.

This process is what makes neural networks such a successful algorithm in today's machine learning paradigm, they are able to improve their performance without being explicitly programmed, learn patterns from data and generalize them to new inputs, forming the basis of modern deep learning systems. By leveraging this process and combining it with their adversarial training model, GANs can provide a powerful framework for learning complex data distributions and generating realistic synthetic samples, and with the correct balance between the discriminator and generator, the results can be so promising that this type of architecture continue to be an active area of research in machine learning.

Chapter 3

Quantum Computing

3.1 Introduction

Quantum computing is an emerging field of computation that leverages the principles of quantum mechanics to perform calculations that are intractable for classical computers. Unlike classical computing, which relies on bits as the fundamental unit of information, quantum computing uses *qubits*, which can exist in a superposition of states.

Computational basis states

In quantum computing there are two states that arise very often:

$|0\rangle$: This special state $|0\rangle$ is called a computational basis state.

$$|0\rangle \coloneqq \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$|1\rangle$: This special state $|1\rangle$ is called a computational basis state.

$$|1\rangle \coloneqq \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

Quantum bit

Just like a bit, a qubit has a state. But whereas the state of a bit is a number (0 or 1), the state of a qubit is a two-dimensional complex vector. It must satisfy the normalization constraint: $|\alpha|^2 + |\beta|^2 = 1$ [24].

Quantum computing elements

Superposition A quantum system can exist in multiple states simultaneously. For a qubit, this means that it can be in a linear combination of the basis states $|0\rangle$ and $|1\rangle$:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{3.1}$$

where α and β are complex amplitudes such that $|\alpha|^2 + |\beta|^2 = 1$.

Coherence Coherence refers to the preservation of the relative phase between quantum states. It is essential for maintaining superposition and is typically lost due to interaction with the environment (decoherence).

Entanglement Entanglement is a phenomenon where the states of two or more qubits become correlated such that the state of each qubit cannot be described independently of the others [14].

Measurement Measurement collapses the quantum state into one of its basis states. For a single qubit, measurement results in either $|0\rangle$ or $|1\rangle$ with probabilities $|\alpha|^2$ and $|\beta|^2$, respectively.

3.2 Two-Level quantum systems

Qubits

A qubit is a two-level quantum system. Unlike a classical bit, a qubit can be in any superposition of the states $|0\rangle$ and $|1\rangle$.

Dirac notation

Quantum states are represented using Dirac notation (or bra-ket notation). The states $|0\rangle$ and $|1\rangle$ form an orthonormal basis:

$$\langle 0|0\rangle = 1, \quad \langle 1|1\rangle = 1, \quad \langle 0|1\rangle = 0 \tag{3.2}$$

Inner products and measurement probabilities

The probability of measuring a quantum state $|\psi\rangle$ in the state $|\phi\rangle$ is given by:

$$P = |\langle \phi|\psi\rangle|^2$$

Tensor product for multi-qubit systems

Multi-qubit systems are represented using tensor products. For example, the combined state of two qubits is:

$$|\psi\rangle \otimes |\phi\rangle \tag{3.3}$$

The tensor product allows the description of entangled states [24].

3.3 Quantum gates and circuits

Quantum gates are fundamental building blocks of quantum computation. Much like classical logic gates operate on bits to perform logical operations, quantum gates manipulate qubits through well-defined mathematical transformations. However, while classical gates are typically irreversible Boolean functions, quantum gates are reversible operations represented by unitary matrices [8]. This reversibility stems from the foundational principles of quantum mechanics, specifically the requirement that the evolution of a closed quantum system be governed by a unitary operator, preserving the total probability of the system's state.

Comparison with classical logic gates

In classical computing, gates such as AND, OR, and NOT take input bits (0 or 1) and produce output bits based on deterministic rules. These gates typically operate on a finite set of states, and many of them are irreversible. For example, the classical AND gate cannot be reversed: given an output of 0, one cannot unambiguously determine the input values.
Quantum gates, by contrast, operate on qubits that can exist in superposition states—linear combinations of the basis states $|0\rangle$ and $|1\rangle$. Therefore, quantum gates must be capable of transforming such superpositions while preserving the probabilities encoded in the system. Mathematically, this is achieved using unitary matrices, which by definition satisfy

$$U^\dagger U = I, \tag{3.4}$$

where $U^\dagger$ is the conjugate transpose of U and I is the identity matrix.
For example, the quantum analogue of the classical NOT gate is the **Pauli-X gate**, which flips the state of a qubit:

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}. \tag{3.5}$$

When applied to $|0\rangle$, it produces $|1\rangle$, and vice versa, mimicking the behavior of a classical NOT operation. However, unlike its classical counterpart, the Pauli-X gate also acts on superposition states, rotating them accordingly on the Bloch sphere.

3.4 The Bloch sphere representation

To better visualize how quantum gates transform qubit states, it is useful to represent a qubit geometrically using the **Bloch sphere**. The Bloch sphere is a unit sphere where any single-qubit pure state can be represented as a point on its surface. A general qubit state $|\psi\rangle$ can be expressed in terms of spherical coordinates (θ, ϕ) as:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle. \tag{3.6}$$

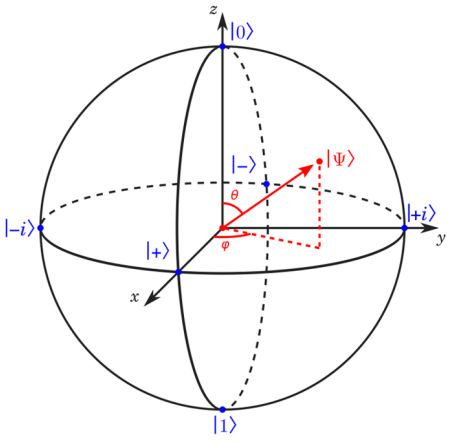


Figure 3.1: Representation of the three main basis on the Bloch sphere: $|0\rangle$ and $|1\rangle$, $|+\rangle$ and $|-\rangle$, and $|i\rangle$ and $|-i\rangle$ [22].

Here, $\theta \in [0, \pi]$ and $\phi \in [0, 2\pi)$ determine the position of the qubit on the sphere. The north and south poles correspond to the computational basis states $|0\rangle$ and $|1\rangle$, respectively. States on the

equator, such as $|+\rangle = (|0\rangle + |1\rangle)/\sqrt{2}$ and $|-\rangle = (|0\rangle - |1\rangle)/\sqrt{2}$, represent equal superpositions with different phases.
Quantum gates act as rotations on this sphere. For instance, the Pauli-X gate performs a π-rotation around the X-axis, flipping $|0\rangle$ and. The Pauli-Y and Pauli-Z gates rotate around their respective axes, altering the relative phases of the qubit state. Likewise, the Hadamard gate corresponds to a rotation that moves a pole state to a point on the equator, placing the qubit into a balanced superposition.
Rotation gates, discussed in the next section, allow continuous adjustments to the qubit state, enabling arbitrary rotations around the X, Y, or Z axes of the Bloch sphere. This geometric visualization is fundamental for understanding how quantum algorithms manipulate qubits through sequences of such rotations.
The Bloch sphere is thus a powerful tool for intuitively understanding how quantum gates transform qubit states. While classical bits can only occupy discrete values (0 or 1), a qubit's state spans a continuous surface, highlighting the vastly richer structure of quantum state space [25].
Quantum gates can act on individual qubits or multiple qubits simultaneously. Single-qubit gates modify the state of a single qubit and play a crucial role in creating and manipulating superpositions and phases.

Single-qubit gates

One important family of gates is the **Pauli matrices**, which form the foundation of many quantum algorithms. The **Pauli-X gate**, as mentioned, flips the qubit state. The **Pauli-Y gate** introduces a phase shift combined with a state flip and is represented by the matrix:

$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}. \tag{3.7}$$

The **Pauli-Z gate** leaves the $|0\rangle$ state unchanged while applying a phase flip to the $|1\rangle$ state:

$$Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{3.8}$$

Another critical single-qubit gate is the **Hadamard gate (H)**. Unlike the Pauli gates, which primarily flip states or introduce phase shifts, the Hadamard gate transforms a basis state into a superposition of $|0\rangle$ and $|1\rangle$:

$$H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{3.9}$$

For example, applying H to $|0\rangle$ yields the state $(|0\rangle + |1\rangle)/\sqrt{2}$, which means that the qubit now has equal probabilities of being measured in either of the basis state. This ability to create superposition is essential for quantum parallelism, a phenomenon where a quantum computer can process multiple computational paths simultaneously.
In addition to these, **rotation gates** provide fine control over the state of a qubit on the Bloch sphere. These gates rotate the qubit around a specific axis by an angle θ. The general form of rotation gates is expressed as exponential operators of Pauli matrices:

$$R_X(\theta) = e^{-i\frac{\theta}{2}X}, \quad R_Y(\theta) = e^{-i\frac{\theta}{2}Y}, \quad R_Z(\theta) = e^{-i\frac{\theta}{2}Z}. \tag{3.10}$$

These gates are indispensable in quantum algorithms and error correction protocols because they allow continuous transformations rather than discrete flips.

Universality of quantum gates

Quantum computation relies on the principle of **gate universality**, which states that any quantum computation can be decomposed into a sequence of single-qubit and two-qubit gates [2]. Specifically, any unitary operation acting on multiple qubits can be approximated to arbitrary precision by composing basic gates such as the Hadamard, Pauli, phase, and controlled-NOT gates. This universality is similar to the role played by NAND gates in classical computing, where a single type of gate suffices to build any logical circuit.

Physical implementation

Unlike classical gates, which are made with transistors, quantum gates are physically implemented using a variety of technologies, such as superconducting circuits, trapped ions, or photonic systems. In each of these platforms, external controls (e.g., microwave pulses, laser beams) manipulate the quantum state in a manner corresponding to the desired unitary transformation.

The NISQ Era

The current stage of quantum computing is commonly referred to as the Noisy Intermediate-Scale Quantum (NISQ) era. This term, introduced by Preskill (2018), characterizes devices that contain tens to a few hundred qubits, which are not yet fault-tolerant and remain highly sensitive to noise and decoherence. Despite these limitations, NISQ devices provide a valuable platform for exploring quantum algorithms, including variational methods and quantum generative adversarial networks (QGANs). Their intermediate scale makes them particularly suitable for hybrid quantum-classical approaches, where classical optimization techniques are combined with quantum circuits to exploit potential quantum advantages while mitigating hardware imperfections.

3.5 Quantum neural networks

Quantum Neural Networks (QNNs) are hybrid models that integrate the principles of quantum computing into the architecture of neural networks. The aim of QNNs is to leverage the advantages of quantum mechanics—such as superposition and entanglement—to enhance the representational power and computational efficiency of classical neural networks, especially in contexts where quantum data or high-dimensional Hilbert spaces are involved [31].
In a typical QNN, the classical layers of a neural network are partially or fully replaced by parameterized quantum circuits (PQCs) [4]. These quantum circuits are composed of quantum gates whose parameters (e.g., rotation angles) are trainable and act on qubits that encode input information. The architecture generally follows a hybrid quantum-classical pipeline:

1. **Data encoding:** Classical input data is embedded in quantum states using encoding techniques such as angle encoding or amplitude encoding.

2. **Parameterized quantum circuit (PQC):** A quantum circuit is constructed with gates that depend on trainable parameters. This circuit processes the encoded input and transforms it within the quantum state space.

3. **Measurement:** After running the PQC, the expectation values of certain observables (e.g., Pauli operators) are measured, yielding classical outputs that can be used in architectures like generative models.

4. **Loss computation and optimization:** The outputs are used to calculate a loss function, which is then minimized using gradient-based optimization methods that modify the trainable parameters in order to minimize it.

QNNs are particularly useful for quantum machine learning tasks where quantum data is present, but they have also shown promise in classical domains with limited classical data.

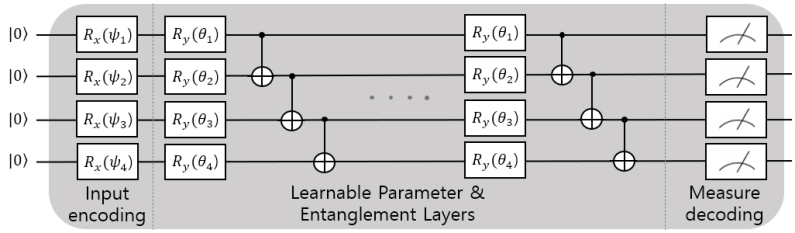


Figure 3.2: Conceptual architecture of a Quantum Neural Network (QNN).

3.6 Quantum generative adversarial networks

Parametric quantum circuits can be integrated into a generative adversarial network to obtain a quantum generative adversarial network (QGAN), combining the expressive power of quantum circuits with the optimization strategies of classic neural networks [7]. Since a GAN can be explained as two neural networks [11], then all the points mentioned in this thesis are valid for them too. We can choose to have a pure quantum architecture, meaning that the generator and discriminator are quantum, or a hybrid architecture, this is a quantum generator and classic discriminator.

Summary

In summary, quantum gates extend the concept of logical manipulation from deterministic binary operations to probabilistic transformations on superposed states. Their ability to maintain coherence and entanglement during computation underlies the exponential advantage that quantum algorithms can provide over their classical counterparts in certain tasks. QNNs represent a promising frontier in machine learning, although the field is still in active development, early results suggest potential quantum advantages in various fields. We may be still far from the end of the NISQ era but in one way or another, just as AI is changing the paradigms, it seems quantum computing is going to be the next technological revolution.

Chapter 4

Classical computing limitations and quantum computing as a solution

One of the major challenges in modern experimental science is the rapidly increasing demand for computational resources. Large-scale experiments, such as those conducted at the Large Hadron Collider (LHC), rely heavily on simulations, data analysis, and machine learning models that require enormous processing power. Classical computers, even with the current trends in hardware and software optimization, are struggling to keep up with these demands. The rate of development in classical computing is simply not fast enough to match the challenges that lie ahead of the scientific community.
This is why many researchers are now turning to quantum computing, a paradigm that promises to overcome some of the fundamental bottlenecks of classical approaches. If quantum devices can perform the same amount of operations as classical computers using a fraction of the resources, the total available computational power would effectively be multiplied, providing an advantage for large-scale scientific projects.

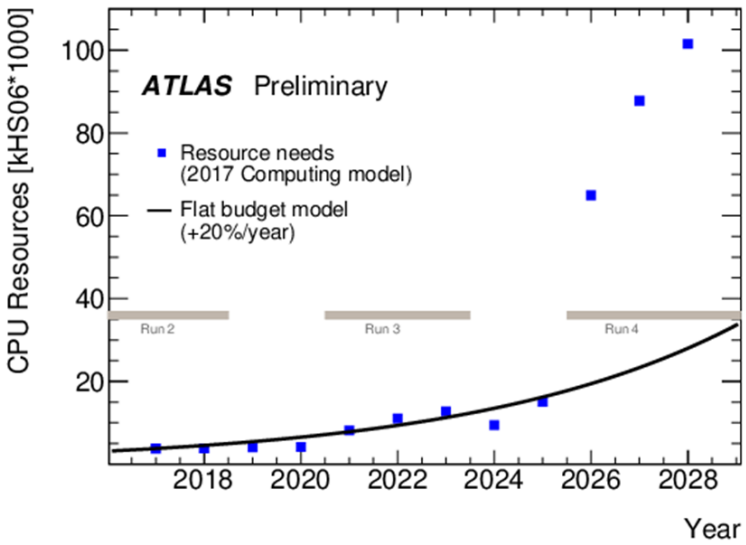


Figure 4.1: ATLAS estimated computational resources. [1].

In 4.1 the blue points correspond to an estimation of the computational resources ATLAS will need in future years for both data and simulation processing using the current software performance and the ATLAS computing model parameters from 2017. The solid line shows the expected increase of available resources under a flat funding assumption (20% per year), based on current technology trends.

4.1 Current bottlenecks in high-energy physics computing

As illustrated in Fig. 4.1, there exists a clear gap between the projected computational needs of experiments such as ATLAS and the resources expected to be available with classical technologies. This mismatch is especially critical when considering tasks like Monte Carlo (MC) event generation and detector simulations, which are both computationally intensive and indispensable for precision physics analysis.

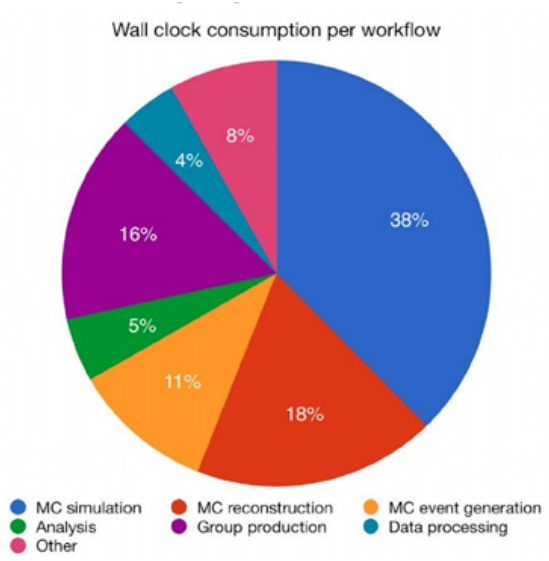


Figure 4.2: ATLAS CPU hours used by various activities in 2018 [19].

As shown in Fig. 4.2, Monte Carlo event generation and simulation represent nearly half (49%) of the total CPU usage in ATLAS computing. This implies that almost half of the available resources are dedicated solely to producing and simulating events, leaving fewer resources for other essential tasks such as data reconstruction and analysis. Consequently, improving the efficiency of generation and simulation represents a major step toward achieving better CPU utilization.

Quantum computing: a promising strategy

Quantum computing provides a fundamentally different framework for performing computations, exploiting quantum mechanical properties such as superposition and entanglement to achieve exponential speedups for certain tasks. While the technology is still in its early stages, promising directions have already been identified in domains directly relevant to high-energy physics:

- **Quantum simulation:** Quantum devices can naturally represent complex quantum systems, offering the potential to simulate particle interactions or detector effects more efficiently than classical algorithms [26].

- **Quantum machine learning:** Hybrid quantum-classical algorithms, such as quantum generative adversarial networks (QGANs), could provide more efficient generative models for data augmentation, reducing the burden of classical Monte Carlo generation [18, 3].
- **Resource reallocation:** By offloading generative and simulation processes to quantum devices, even partially, significant amounts of classical CPU hours could be freed and redirected towards other tasks such as event reconstruction and real-time analysis.

But the benefits of the quantum computers do not stop with their speed of performing a task, but also the optimization of the resources, for example, if we had a problem that a classic computer can do faster than a quantum one, the quantum computer may still be useful because it can be able to solve the same problem but with less computational resources, so we may still want to use the quantum version even if it is slower.

Summary

The roadmap of the high-energy physics community already anticipates the need for disruptive computing technologies, with quantum computing identified as one of the most promising candidates. Although quantum hardware remains limited in terms of qubit number and error rates (the so-called *Noisy Intermediate-Scale Quantum* or NISQ era), the steady progress in both algorithms and devices suggests that quantum solutions may become practical within the coming decades.
It is unlikely that quantum computers will replace classical machines in the short term. Instead, hybrid approaches—where classical and quantum resources complement each other—are the most realistic path forward. In this scenario, quantum algorithms could take over specific bottleneck tasks (such as data generation, simulation, or high-dimensional optimization), thereby alleviating pressure on classical infrastructures.

Chapter 5

Implementation of a quantum generative adversarial network for a small dataset problem

5.1 Characteristics of a particle collision

Background and Signal Events

To understand the problem at hand, first we must know what does background and signal actually mean. In high-energy physics experiments, the data collected from particle collisions consist of numerous events that can be broadly classified into two categories: *background* and *signal* events. **Background events** are those that originate from known and well-understood physical processes that are not of direct interest to the specific analysis being conducted; in this thesis they are quarks. These events typically represent noise or common occurrences that can obscure the detection of rare phenomena. Background events are abundant and form the basis for the experimental data.

Signal events, on the other hand, correspond to the rare or new phenomena that the experiment aims to detect or study, such as the presence of a new particle or interaction. These events exhibit distinctive characteristics or signatures in the measured features that differentiate them from the background. For this thesis, the signal events correspond to gluons.
The separation between background and signal is fundamental in data analysis, as it allows physicists to identify and study rare processes amidst a large amount of unrelated data.

Particle Jets

Talking about signal and background events, this events are detected through *jets*, they are collimated structures of particles that emerge when a quark or gluon, produced in a high-energy collision, cannot exist as a free particle due to color confinement. Instead, these partons undergo a process known as hadronization, in which they transform into a collection of hadrons that propagate approximately in the same direction as the original parton. The experimentally observable result in the detector is a "spray" of particles, commonly referred to as a *jet*.

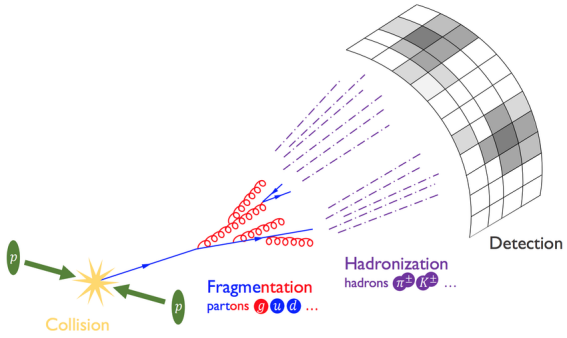


Figure 5.1: Proton-proton collision produce particles that go through a hadronization process during their trajectory towards the detector [6].

The study of jets is crucial because it provides indirect information about the underlying quarks and gluons, which cannot be observed directly. Moreover, jets are among the most frequent signatures in experiments such as ATLAS and CMS at the LHC, where they are abundantly produced in proton-proton collisions.

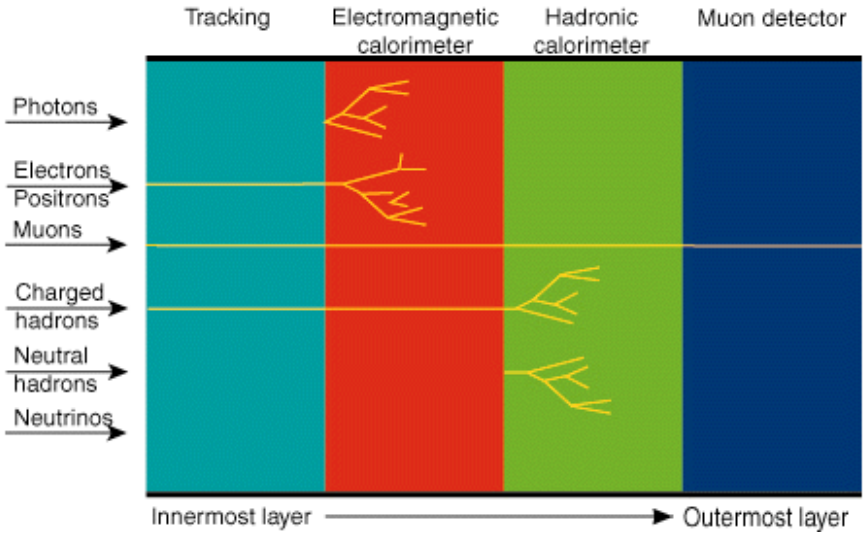


Figure 5.2: Distance that each type of particle travels through the detectors [27]

Jet Detection

The detection and reconstruction of jets are carried out using the different subdetectors that compose modern high-energy physics experiments. In general terms, the procedure involves the following:

- **Calorimeters**: Most of the energy deposited by the hadronic and electromagnetic particles forming the jet is absorbed in the calorimeters of the detector. The spatial distribution of this deposited energy is used to reconstruct the jet profile.
- **Tracking detectors**: Charged particles within the jet leave tracks in the inner tracking detectors, providing complementary information on momentum and charge in addition to the energy measurements.

- **Reconstruction algorithms**: Based on the measured energy deposits and tracks, clustering algorithms are applied to group the measured particles into a physical object defined as a jet. These algorithms allow the total energy and direction of the jet to be associated with the initial parton that originated it.

Although quarks and gluons cannot be directly detected, jet analysis provides an indirect yet robust way to study the fundamental interactions of quantum chromodynamics (QCD).

5.2 Problem

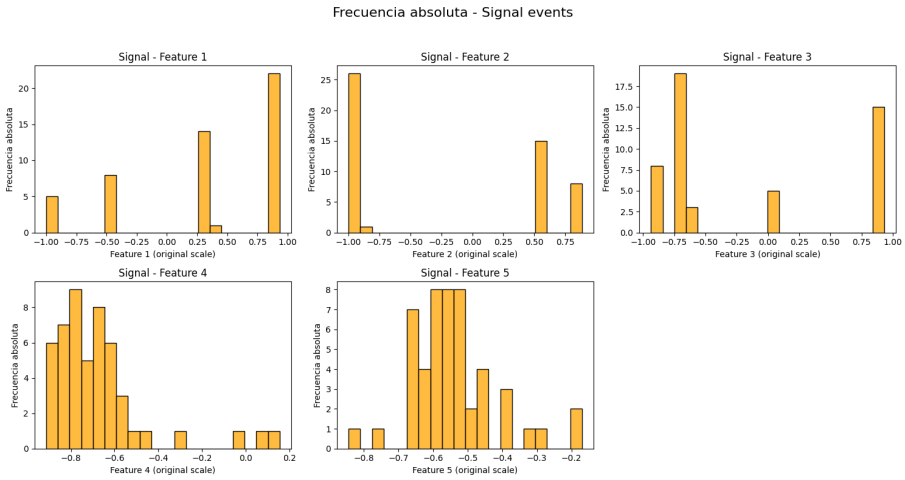
The problem involves a small dataset generated with Delphes that tests the limits of what a quantum enhanced generative model can achieve. We have two types of data, signal events (gluons) and background events (quarks), and as mentioned in past sections, we propose a QGAN capable of learning the underlying structure of the data in order to generate synthetic samples and improve classification performance when compared to classic networks.
The dataset contains:

- **100 training samples**,
- **100 testing samples**.

Each data point is labeled as either:

- **1 (signal event)**
- **0 (background event)**.

Each event has five features which determine if it is signal or background, the goal is to learn the distribution of signal and background events in order generate features that mimic the real ones for both type of events. The small sample size poses challenges for classic neural networks in terms of model generalization and robustness, therefore why it is interesting the results a QGAN may have.



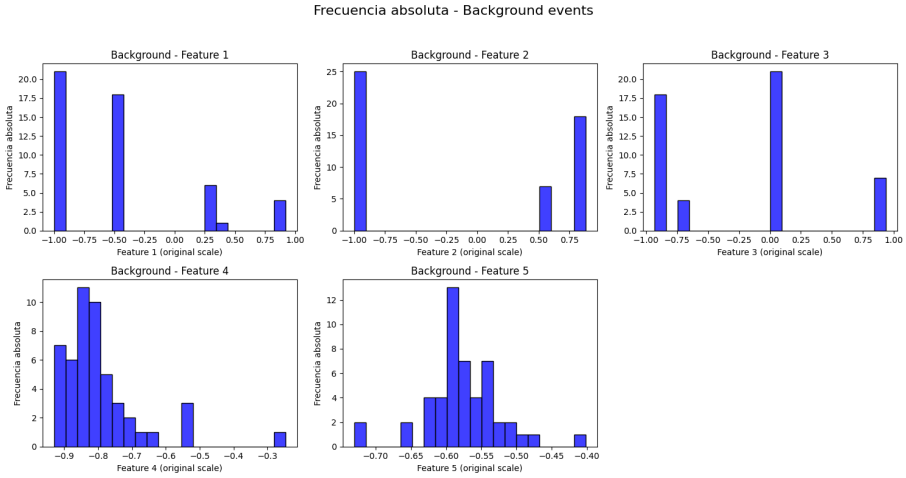


Figure 5.3: Feature distribution: the first three present a discrete distribution while the last two have a continuous gaussian behavior

We focused our work exclusively on the fourth and fifth features of each event vector. These two features were selected for modeling because the first three features exhibit discrete and highly non-continuous structures that proved challenging to reproduce accurately using the current quantum circuit architecture. In contrast, features 4 and 5 display more continuous and statistically tractable behavior, making them better suited for modeling with a variational quantum circuit. As such, the generator is trained solely using the data from these two features in signal and background events.

5.3 Implementation Details and Architecture

The framework implemented for this QGAN is based on **Pennylane** for quantum circuit construction and **Pytorch** for hybrid classical-quantum training. These tools enable the seamless integration of quantum circuits into Pytorch computational graph, allowing for gradient-based optimization of parameterized quantum circuits (PCA) which we use to model the generator and discriminator.

Quantum Generator

The generator is implemented as a parametrized quantum circuit which takes as input a latent vector z sampled from a standard normal distribution. The input is encoded using `Angle Embedding` and passed through a series of `Strongly Entangling Layers` on 2 qubits, corresponding to the two selected features, which apply RX, RY, and RZ rotations while also entangling the qubits with CNOT gates. Then we obtain the expectation value of the Pauli-Z observable on each qubit, forming a 2-dimensional output.
To overcome the range limitations of Pauli Z„ which is $[-1, 1]$, we add learnable classical scaling and shifting parameters that allow the model to adjust the generated outputs to better match the distribution of real samples. These parameters are initialized from the statistics (mean and standard deviation) of the original data. The generated outputs are computed as:

$$x_{\text{gen}} = G(z) = \text{scale} \cdot \text{output}_{\text{qnode}}(z) + \text{shift}$$

Quantum Discriminator

The discriminator is also modeled as a quantum neural network using 2 qubits. It takes as input a 2D real or generated sample, encodes it via `Angle Embedding`, and processes it with `strongly`

`entangling layers`. The circuit outputs the expectation value of a Pauli-Z observable on the first qubit, which is squashed through a sigmoid function to obtain a binary probability representing the likelihood that the input is real.
An important thing to note about the discriminator is that for each type of architecture that was tested, there was only one type of event, what this means is that, if we wanted to test an hybrid architecture for signal events, then the only data uploaded into the algorithm were the signal events. By doing this, the discriminator does not have to classify between signal or background and can concentrate into only classifying between real or generated features.

5.4 Quantum vs Hybrid architecture

Another problem that needed to be addressed was the type of architecture that was better suited for this task, quantum generator and discriminator or a quantum generator and classic discriminator? For the signal data set, it was immediately clear that the quantum architecture had the upper hand, as for the background data set, it was not clear which option was best.

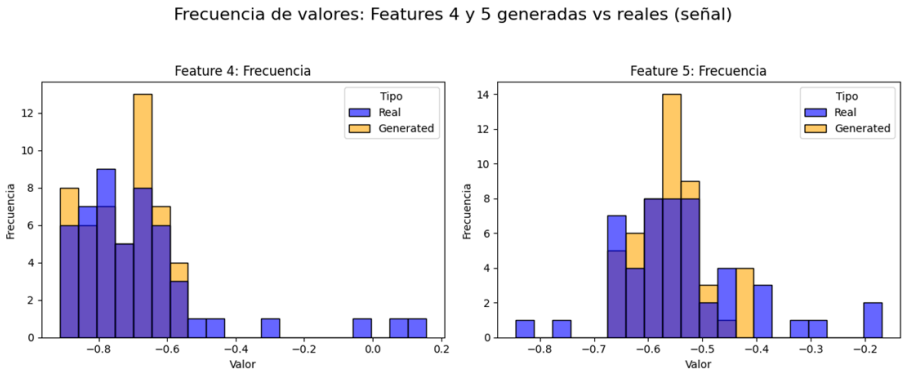


Figure 5.4: Quantum generator and discriminator (signal/gluons), We can notice a significant amount of intersection between the generated and real features but some peaks do not match.

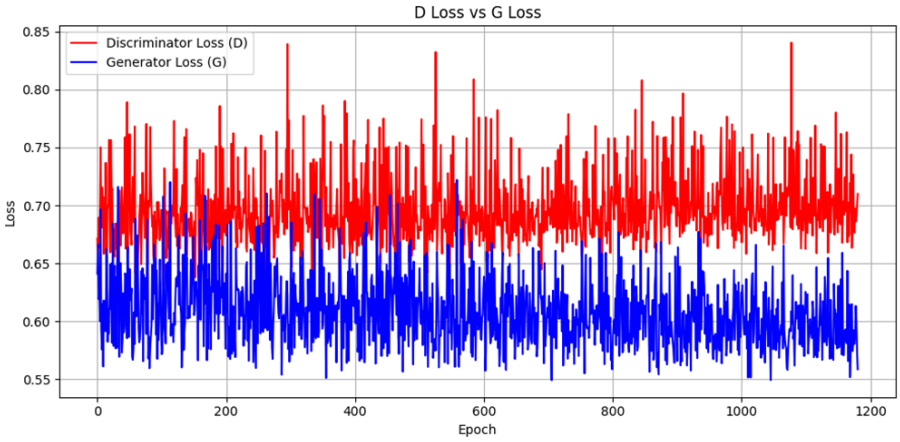


Figure 5.5: Quantum generator and discriminator loss (signal/gluons), both loss function graphs do not seem to approximate a Nash equilibrium despite that, we obtained good results.

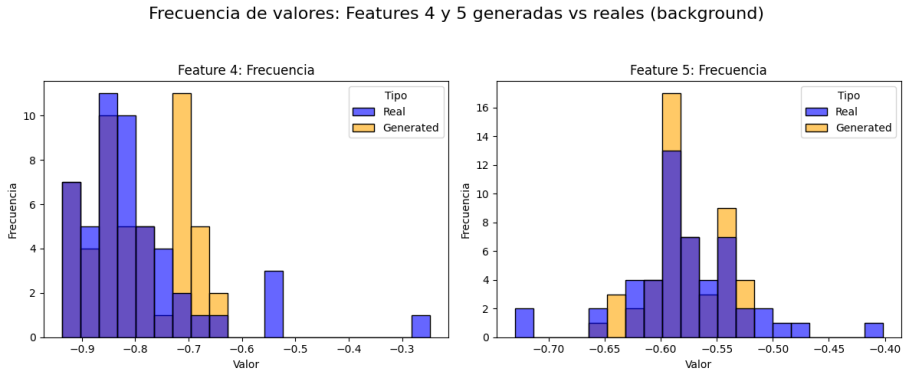


Figure 5.6: Quantum generator and discriminator (background/quarks), the feature 5 seems to be well replicated but the feature 4 has a generated peak that does not match any real data.

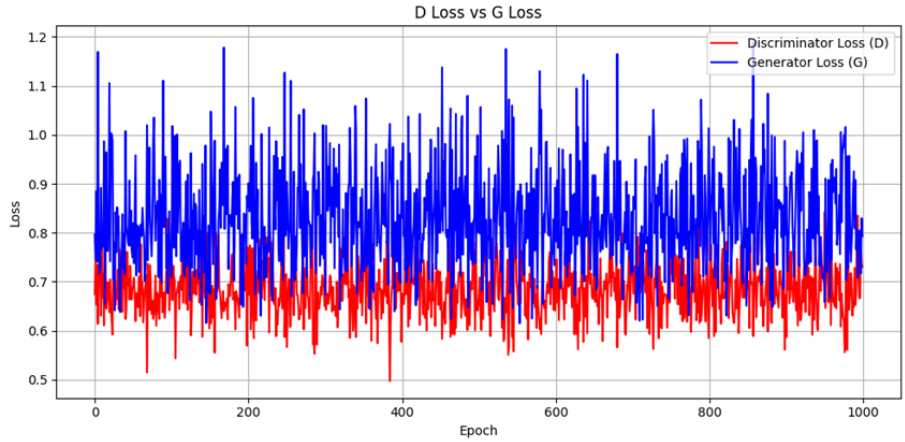


Figure 5.7: Quantum generator and discriminator loss (background/quarks).
Both loss functions for quark data have a higher value than those for the gluon data, which makes sense since we have less area of intersection between the generated and real features, and even though they seem to have reached a limit in the training where they do not improve across more epochs this does not mean we have reached a proper Nash equilibrium, but that this architecture hit it's limit.

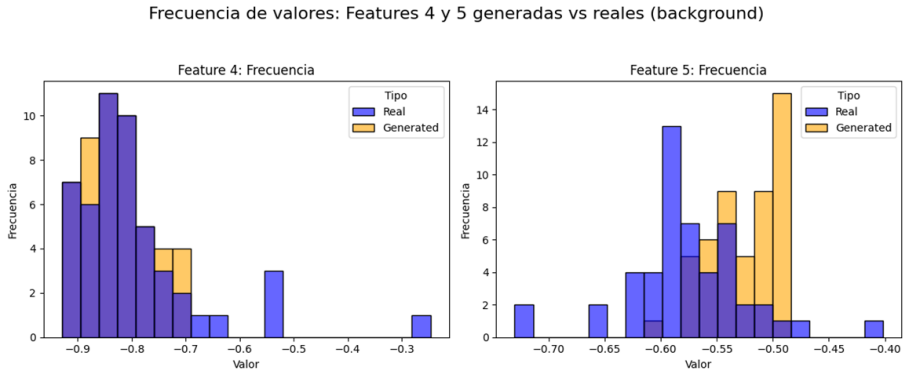


Figure 5.8: Quantum generator and classic discriminator (background/quarks). The feature 4 is very well replicated but with some room for improvement. Meanwhile the feature 5 is poorly generated, it even appears that in the training process the generator "ignored" the qubit for the feature 5. This behavior will be discussed later in this section.

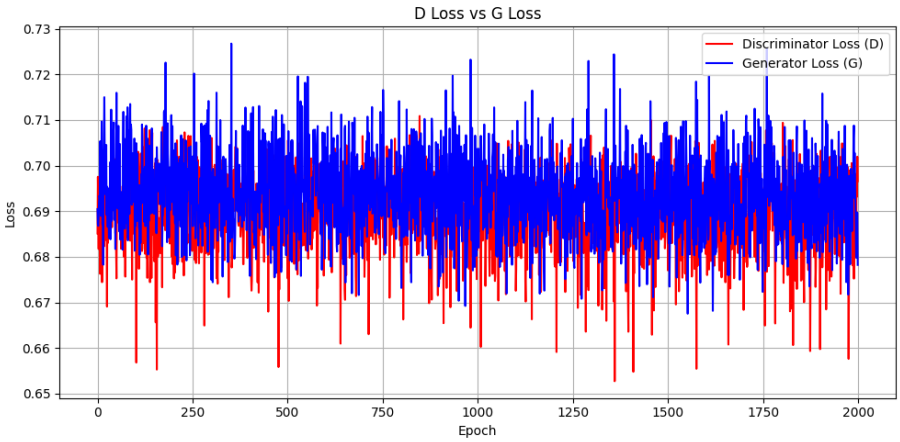


Figure 5.9: Quantum generator and classic discriminator loss (background/quarks). Even with the feature 5 being poorly generated, the loss functions reached a Nash equilibrium.

Nash equilibrium

An interesting thing to note here is that for most of the runs of both events, even if the loss metric showed that a nash equilibrium has been achieved, this was not the case. Some runs did not achieve a Nash equilibrium at all but were able to generate features similar to the real ones and some other runs had a very good Nash equilibrium, when looking at the loss function, but the features generated by this ones were not close to the reality. Therefore, for the majority of the runs, the loss function metric was not as determinant as it was expected to be, so we could only use it as an early stage indicator of a good training but was not enough to conclude that a training was successful.

Runtime

For the signal events, the runtime for 1200 epochs was approximately 138 minutes which is 2 hours and 18 minutes.

For the background events the runtime was approximately 165 minutes, which is 2 hours and 45 minutes for 2000 epochs. The difference in the epochs between the quantum and hybrid architecture for background events is due to the time it took each architecture to run. for each 1000 epochs of the quantum architecture, the hybrid one could do 2000 approximately.

Training Strategy

Both the generator and discriminator are trained adversarially using mini-batch stochastic gradient descent with Adam optimizers. The training is done using a small batch size (2) to match the qubits and features involved, maximize gradient signal and promote stable training.
Key strategies used during training include:

- **Feature Matching:** We add a regularization term based on the absolute difference between the mean discriminator outputs for real and generated samples.
- **Progressive Parameter Clipping:** To ensure numerical stability, the scaling and shifting parameters of the generator are clamped within reasonable ranges, (0.5, 2) for scaling and (-1.5, 1.5) for shifting.

5.5 Evaluation Metrics

In order to evaluate the performance of the Quantum Generative Adversarial Network (QGAN), we focus on metrics that are particularly relevant to generative models. Unlike traditional classifiers that are assessed via accuracy or AUC, generative models are evaluated based on how well the generated samples match the true data distribution.

Evaluation and Visualization

The model is evaluated using:

- **Visual Histogram Overlap:** After training, we plot histograms of the absolute frequency of the real and generated data distributions for both Feature 4 and Feature 5, providing a visual assessment of how closely the model has learned the signal distribution. This reveals whether the generator has learned the correct modes and shape of the target distribution. A high overlap indicates successful modeling of the data structure, while a poor match can mean mode collapse or insufficient capacity in the quantum circuit.
- **Loss:** This metric is only useful in the early stages of the training, if the losses are far too separate, then we can try different learning rates or different number of layers, but for mid and late stages of the training this metric does not give useful information.

Overall, this implementation shows how a fully quantum QGAN, when properly regularized and trained, can learn to reproduce the underlying structure of a Gaussian distribution from limited samples. This is a fundamental step toward scalable quantum generative models for data augmentation.

5.6 Measuring only one qubit

For all the networks run until this point, we have only been measuring the qubit with the information of feature 4. This is a detail that was not noted until very deep into this thesis but a reasonable conclusion is that this is the reason why the hybrid architecture seemed like it forgot about feature 5, because the qubit corresponding with that feature was not being measured. This

does not have the same effect on the pure quantum architecture since it was capable of generating real-like data for both features 4 and 5. This can be explained due to the quantum entanglement, meaning that even if we do not directly measure qubit 1 (feature 5) we still can obtain information about it by indirectly measuring through qubit 0 (feature 4) because the strongly entangling layers entangled them, and for some reason, perhaps because this architecture has a major quantum part and preserves it better, the entanglement appears to be stronger in the pure quantum architecture.

Pure quantum architecture impediment

Due to the nature of an adversarial model, the discriminator's output has to be a single number that represents the probability that the feature is real or generated. Because of this, if we want to measure both qubits then we have to use a classic neural network in the discriminator to a greater or leaser extent. We can choose between a purely classic discriminator or one with quantum input layers, but it is imperative that the last layers have to be classic in order to convert the values of both qubits into a single real number.

Measuring both qubits

The type of architectures that were tested measuring both qubits are the following:

- Signal events fully classic discriminator.
- Signal events hybrid discriminator.
- Background events fully classic discriminator.
- Background events hybrid discriminator.

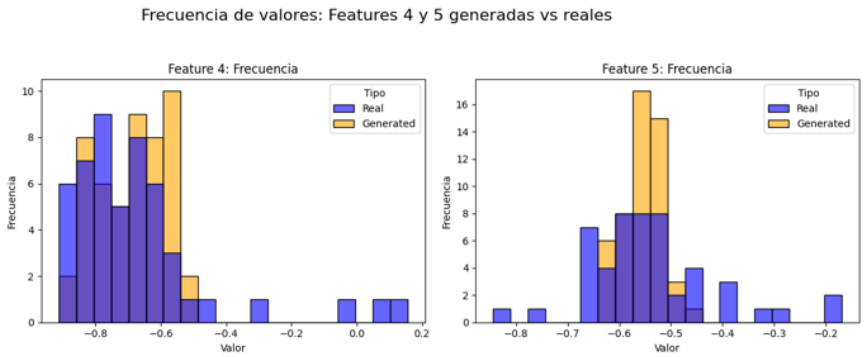


Figure 5.10: Two-qubit measurement for signal events, fully classic discriminator. Feature 4 is better replicated than in the hybrid discriminator but still needs some tuning and feature 5 only replicates the data in the center with some generated peaks that do not match any real data.

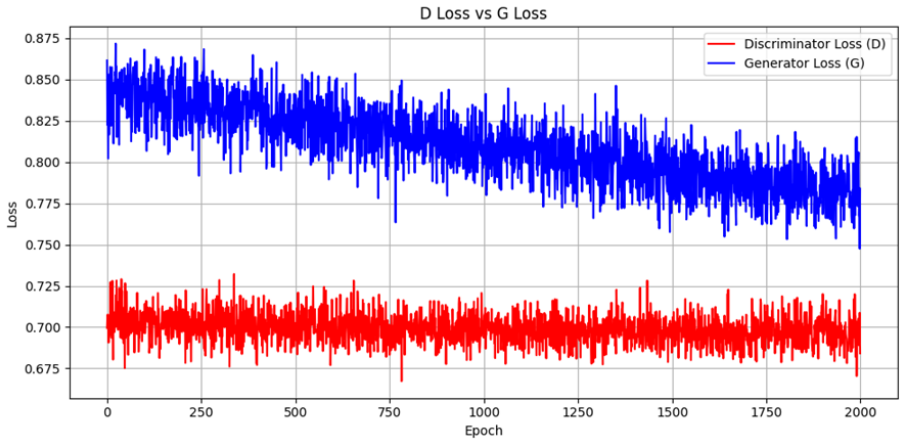


Figure 5.11: Two-qubit measurement for signal events, fully classic discriminator loss. The graphs seem to tend to a Nash equilibrium very slowly but the training was stop before it could be achieved

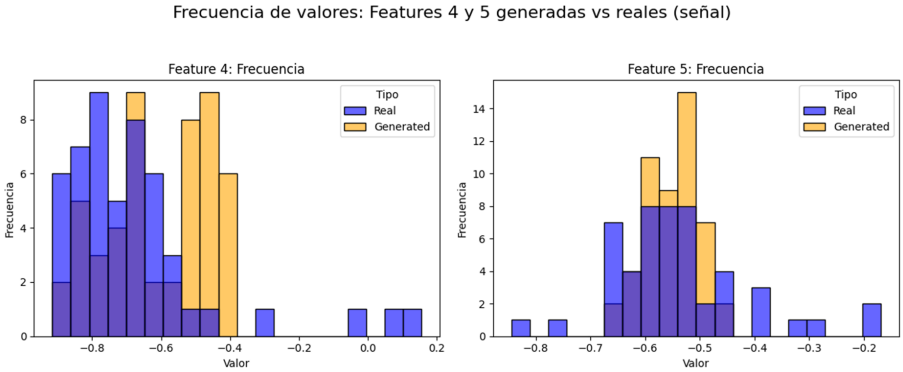


Figure 5.12: Two-qubit measurement for signal events, hybrid discriminator. The feature 4 is partially replicated but has a generated peak that does not correspond to any real data. The feature 5 is better replicated than feature 4 but can still be improved.

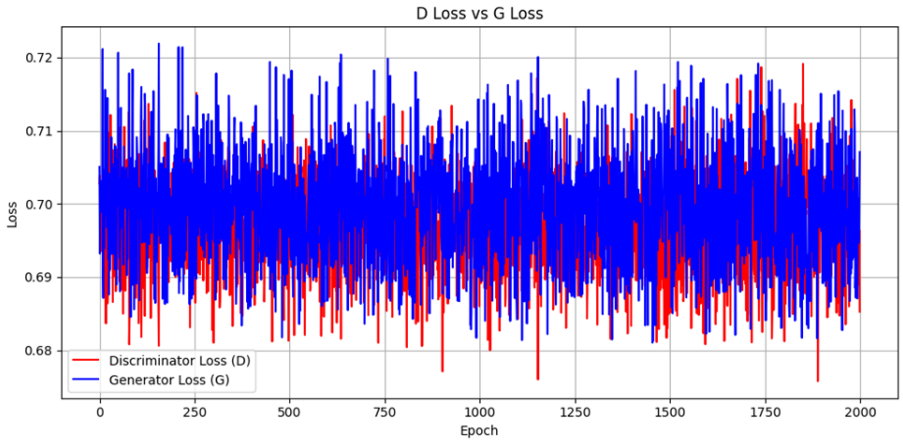


Figure 5.13: Two-qubit measurement for signal events, hybrid discriminator

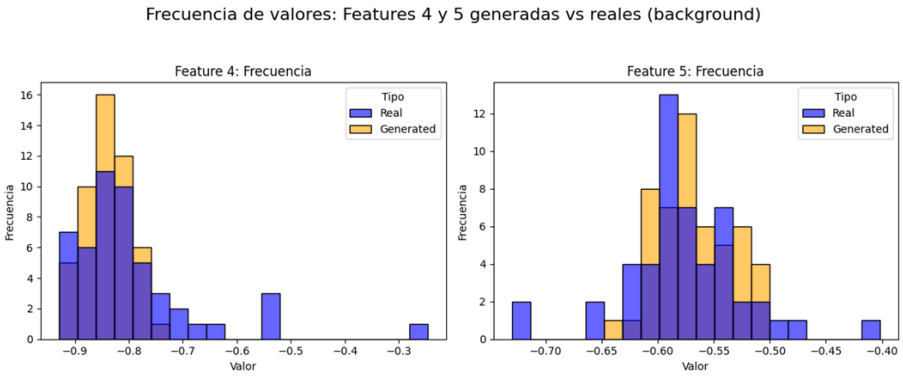


Figure 5.14: Two-qubit measurement for background events, fully classic discriminator. The feature 4 is partially replicated but feature 5 is poorly generated.

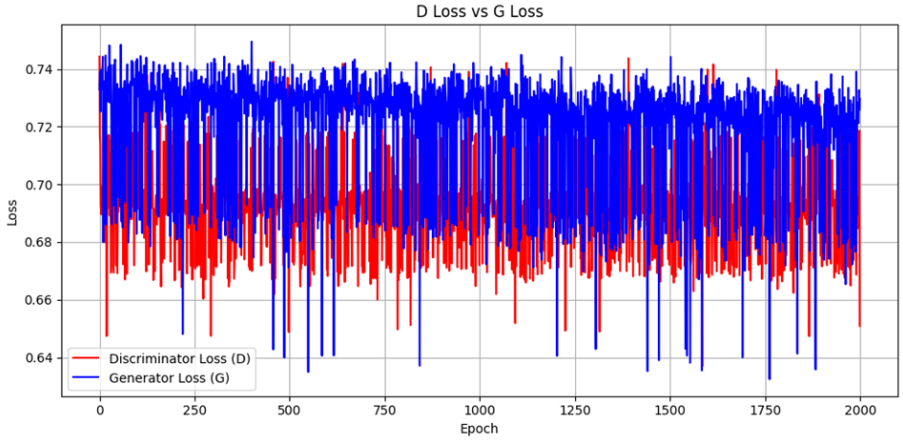


Figure 5.15: Two-qubit measurement for background events with a fully classic discriminator loss. If we see the y axis values we can note that the loss functions oscillations are very small, thus we have a Nash equilibrium.

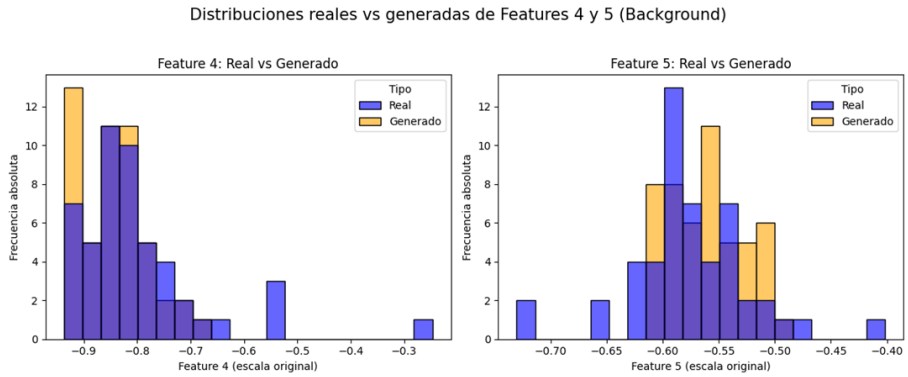


Figure 5.16: Two-qubit measurement for background events, hybrid discriminator

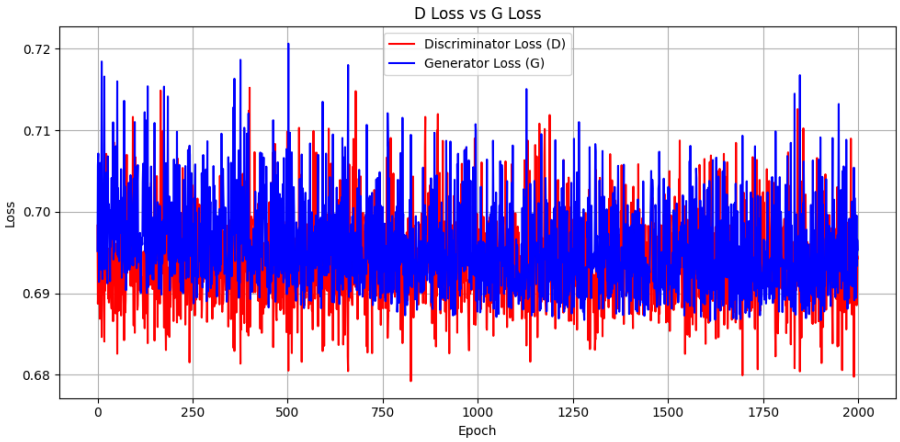


Figure 5.17: Two-qubit measurement for background events with hybrid discriminator loss. Though a Nash equilibrium can be observed, only the feature 4 was well replicated and still with room for improvement, the feature 5 was poorly generated.

Runtime

All the following runs were stopped at 2000 epochs:

- Signal events, hybrid discriminator 130 minutes which is 2 hours and 10 minutes.
- Signal events, fully classic discriminator: 58 minutes.
- Background events and fully classic discriminator: 61 minutes.
- Background events and hybrid discriminator: 2 hours and 2 minutes.

As it was expected, the more quantum the architecture the longer the runtime will be, if the generator is the only quantum component of the network then the runtime will be halved.

5.7 Best architecture

The architecture with the best results for both signal and background events is undoubtedly the pure quantum one that only measures one qubit for both signal and background events. Even without measuring both qubits the pure quantum architecture was still capable of replicating both features better than any other architecture, this can be explained due to the level of entanglement, even if the hybrid architecture could replicate the feature 4 far better, since only the generator was quantum, it did not possess the capability to keep the quantum entanglement that was needed to preserve the information of qubit 5. On the other hand, the purely quantum one could obtain information of feature 5 indirectly due to quantum entanglement and this did not stop in the generator because the discriminator was also quantum, keeping more information of the entangled state until the very last moment before the measurement. Surprisingly, the network that measures both qubits did not meet the expected results but, if this type of network would be better if it was purely quantum is still an open question since this was the only architecture that was not tested but a way to implement it will be discussed in the last section of this thesis.

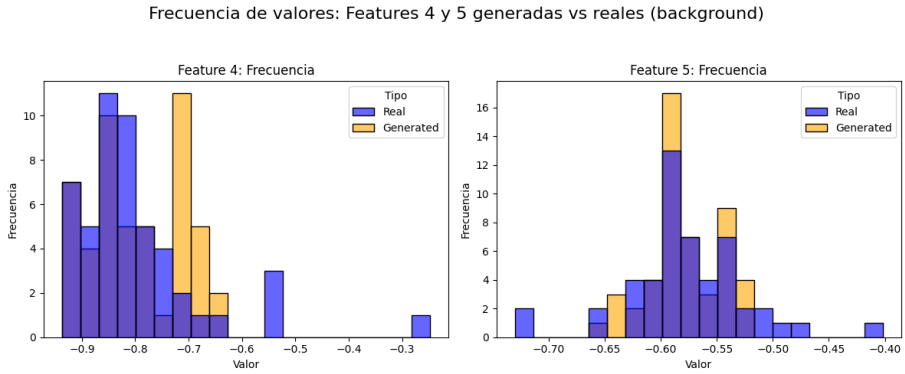


Figure 5.18: Quantum generator and discriminator (background/quarks), the architecture with best results for background events.

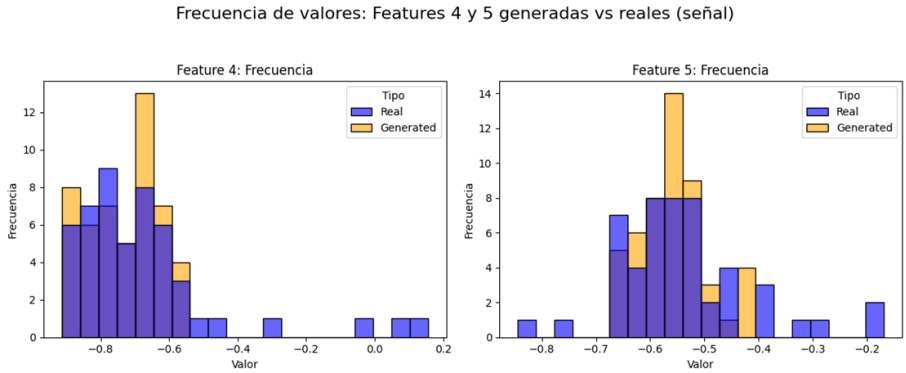


Figure 5.19: Quantum generator and discriminator (signal/gluons), was the architecture with best results for signal events.

Chapter 6

Summary and conclusion

Although the current model has certain limitations, as it is capable of replicating only two out of the five features, it demonstrates that a targeted quantum generative approach can be effective for specific subsets of the data. One possible extension is to implement an additional QGAN architecture specifically designed for the remaining three features, which exhibit non-continuous and potentially multimodal distributions. By training each QGAN on the feature set for which it is most suitable, and subsequently combining their outputs, it would be possible to replicate the full five-dimensional feature space with higher fidelity. This modular strategy has the advantage of independent optimization for each model, potentially improving convergence stability and the quality of the generated samples.

To overcome the problem that does not let the discriminator be purely quantum when measuring both qubits from the generator, we suggest that the swap test could be useful. The architecture could be arranged in a way that we will have 3 qubits, two with the probability that the features are real and one auxiliary qubit that will only activate when the other two are equal or at least very similar (meaning that both features are probably real). Doing the measurement this way, we will make sure that the only features being passed are those with a real-like distribution, but it could also be challenging to balance the threshold at which the qubit will activate; Being to flexible with let poorly generated features go through and a very strict threshold will be too hard on the generator and nothing will come out as the output. This may not be a solution but the proposal is open and it ensures that the architecture will remain fully quantum for further experiments.
Finally, he proposed model of a QGAN did achieve the goal it was created for, it is possible to train a QGAN model with a small dataset that can generate data that imitate the real one. This brings to light the utility of the QGAN architectures for data augmentation in problems with a limited amount of data, and even though in this thesis the runtime was very long (more than 2 hours), if we were to use a real quantum computer this time will become very small compared to the simulator. It was also proven that, at least for this task, a pure quantum architecture is better than a hybrid one, but this should not be the only thing to consider when choosing an architecture since other problems can arise. The cons of using a pure quantum architecture are that quantum computers are very hard (and expensive) to use, so for problems with similar results between the architectures, it can be enough to use a hybrid architecture. But for problems where the pure quantum architecture advantage is clear, such as in the signal events, a quantum computer might be needed.
Beyond this specific implementation, the broader potential of quantum generative adversarial networks for data augmentation is significant, improvements in quantum circuit desgin, noise mitigation techniques and hybrid optimization strategies could allow QGANs to become capable of handling larger, higher-dimensional datasets with more complex correlations than those feasible on current devices, accurately reproduce non-Gaussian distributions or multi-modal distributions

that remain challenging for classical GANs and even newly QGANs (such as the first three features mentioned in this thesis). For now, there is still much room for quantum technologies to improve and overcome the NISQ era but eventually, as quantum hardware matures, its integration into the computing model of large-scale experiments may become essential to sustain the pace of discovery not only in particle physics but in the majority of fields in science.

Bibliography

[1] Johannes Albrecht, Antonio Augusto Alves, and et. Al. Amadio. "A Roadmap for HEP Software and Computing Ramp;D for the 2020s". In: *Computing and Software for Big Science* 3.1 (Mar. 2019). ISSN: 2510-2044. DOI: 10.1007/s41781-018-0018-8. URL: http://dx.doi.org/10.1007/s41781-018-0018-8.

[2] Adriano Barenco et al. "Elementary gates for quantum computation". In: *Physical Review A* 52.5 (1995), p. 3457.

[3] Marcello Benedetti et al. "Adversarial quantum circuit learning for pure state approximation". In: *New Journal of Physics* 21.4 (2019), p. 043023. DOI: 10.1088/1367-2630/ab14b5.

[4] Marcello Benedetti et al. "Parameterized quantum circuits as machine learning models". In: *Quantum Science and Technology* 4.4 (2019), p. 043001. DOI: 10.1088/2058-9565/ab4eb5.

[5] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. ISBN: 9780387310732.

[6] Carlos Cocha. "Study of b- and c-jets identification with quantum machine learning algorithms and application to the Higgs reconstruction". Tesis doctoral. PhD thesis. 2022.

[7] Pierre-Luc Dallaire-Demers and Nathan Killoran. "Quantum generative adversarial networks". In: *Physical Review A* 98.1 (2018), p. 012324. DOI: 10.1103/PhysRevA.98.012324.

[8] David Deutsch. "Quantum computational networks". In: *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences* 425.1868 (1989), pp. 73–90.

[9] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep sparse rectifier neural networks". In: *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (AISTATS)*. Vol. 15. 2011, pp. 315–323.

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. Cambridge, MA: MIT Press, 2016.

[11] Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2014, pp. 2672–2680.

[12] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. 2nd ed. Springer, 2009. ISBN: 9780387848570.

[13] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. "Multilayer feedforward networks are universal approximators". In: *Neural Networks* 2.5 (1989), pp. 359–366.

[14] Ryszard Horodecki et al. "Quantum entanglement". In: *Reviews of Modern Physics* 81.2 (2009), p. 865.

[15] InspiredPencil. *Unsupervised Learning — imágenes 2023*. Sitio web. Image retrieved from ar.inspiredpencil.com: https://ar.inspiredpencil.com/pictures-2023/unsupervised-learning. 2023. URL: https://ar.inspiredpencil.com/pictures-2023/unsupervised-learning.

[16] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[17] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (2015), pp. 436–444.

[18] Seth Lloyd, Christian Weedbrook, and Maarten van den Nest. "Quantum Generative Adversarial Learning". In: *Phys. Rev. Lett.* 121.4 (2018), p. 040502. DOI: 10.1103/PhysRevLett.121.040502.

[19] J. Marks et al. *Models and Requirements for LHC Computing 2020–2030*. CERN report. 2020.

[20] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997. ISBN: 9780070428072.

[21] Vinod Nair and Geoffrey E. Hinton. "Rectified Linear Units improve Restricted Boltzmann Machines". In: (2010), pp. 807–814.

[22] Marcus R.A. Newman. *Bloch sphere*. https://prefetch.eu/know/concept/bloch-sphere/. 2025. URL: https://prefetch.eu/know/concept/bloch-sphere/.

[23] Kishore Ng. *Supervised Learning*. LinkedIn Pulse article. Image retrieved from LinkedIn. URL: https://www.linkedin.com/pulse/supervised-learning-kishore-ng/. n.d. URL: https://www.linkedin.com/pulse/supervised-learning-kishore-ng/.

[24] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. 10th Anniversary Edition. Cambridge University Press, 2010.

[25] John Preskill. *Lecture Notes for Physics 229: Quantum Information and Computation*. http://theory.caltech.edu/~preskill/ph229/. 1998.

[26] John Preskill. "Quantum Computing in the NISQ era and beyond". In: *Quantum* 2 (2018), p. 79.

[27] Chiara Rizzi. "LHC and ATLAS". In: *Searches for Supersymmetric Particles in Final States with Multiple Top and Bottom Quarks with the Atlas Detector*. Springer Theses. Springer, 2020, pp. 33–67. DOI: 10.1007/978-3-030-52877-5_3. URL: https://link.springer.com/chapter/10.1007/978-3-030-52877-5_3.

[28] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain". In: *Psychological Review* 65.6 (1958), pp. 386–408.

[29] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (1986), pp. 533–536.

[30] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. 3rd ed. Pearson, 2016. ISBN: 9780136042594.

[31] Maria Schuld and Nathan Killoran. "Quantum machine learning in feature Hilbert spaces". In: *Physical Review Letters* 122.4 (2019), p. 040504. DOI: 10.1103/PhysRevLett.122.040504.

[32] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. 2nd ed. MIT Press, 2018. ISBN: 9780262039246.

[33] Techlech. *What are the three types of machine learning? Supervised learning, unsupervised learning and reinforcement learning*. https://techlech.com/what-are-the-three-types-of-machine-learning-supervised-learning-unsupervised-learning-and-reinforcement-learning/. 2021.