Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias Físico Matemáticas

Quantum Generative Adversarial Networks for High Energy Physics

Tesis presentada al

Colegio de Física

como requisito parcial para la obtención del grado de

Licenciado en Física

por

Lázaro Raúl Díaz Lievano

Asesorado por

Dra. María Isabel Pedraza Morales
Dr. Enrique Varela Carlos

Puebla Pue.
April 1, 2025

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias Físico Matemáticas

Quantum Generative Adversarial Networks for High Energy Physics

Tesis presentada al

Colegio de Física

como requisito parcial para la obtención del grado de

Licenciado en Física

por

Lázaro Raúl Díaz Lievano

Asesorado por

Dra. María Isabel Pedraza Morales
Dr. Enrique Varela Carlos

Puebla Pue.
April 1, 2025

Título: Quantum Generative Adversarial Networks for High Energy Physics
Estudiante: Lázaro Raúl Díaz Lievano

COMITÉ

Dr. Iván Fuentecilla Cárcamo
Presidente

Dr. Humberto Antonio Salazar Ibargüen
Secretario

Dr. Jorge Velázquez Castro
Vocal

Dr. Sergei Gleyzer
Suplente

Dra. María Isabel Pedraza Morales
Asesor

Dr. Enrique Varela Carlos
Asesor

Dedications

A mis padres, a mi hermano y a Dannae, cuyo apoyo incondicional y cariño han sido mi mayor fortaleza. Sin ustedes, este logro no habría sido posible.

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my parents. Thank you for allowing me to follow my dreams. Your unconditional support and trust in me have been fundamental to achieving this milestone. To my brother, thank you for being my friend and always standing by my side. To Angeles and Mytzi, I am deeply grateful for your friendship and for making my university years memorable.

I would also like to express my sincere gratitude to Dannae, who believed in me and gave me her unconditional support. I greatly appreciate every moment she was there for me, listening and encouraging me to continue moving forward. I will always cherish her contribution to my personal and academic growth.

I give special thanks to Dr. Isabel Pedraza; your patience, confidence in me, and guidance profoundly shaped my academic and professional development. Working under your mentorship was undoubtedly one of the most enriching experiences of my life, as your guidance deepened my understanding of physics and inspired me to pursue my goals with confidence and determination.

To all my professors, Google Summer of Code mentors, and friends who have supported me throughout this journey, thank you for sharing your passions and knowledge.

Abstract

The increasing computational demands of the High-Luminosity Large Hadron Collider (HL-LHC) program have positioned quantum computing as a promising emerging technology for advancing high-energy physics (HEP) research. This thesis investigates the application of Quantum Generative Adversarial Networks (QGANs) to address two challenges in HEP: particle identification and data generation. A modified quantum model of GANs offers a novel framework for simultaneously tackling data simulation and classification tasks by combining the principles of quantum computing and generative adversarial networks.

Using simulated data from Delphes, we developed a two-headed QGAN model designed to perform two key tasks: discriminating real data from generated data and classifying events into signal and background jet categories. This architecture incorporates quantum generators and discriminators and is implemented using TensorFlow Quantum and Google Cirq, seamlessly integrating quantum circuits with classical computation. While quantum computing remains in its early stages, this work explores the potential of QGANs as a complementary tool in computationally intensive tasks in HEP.

The model's results demonstrate a classification accuracy of 90%, indicating its capacity to mimic real quantum data distributions while addressing classification tasks. However, we emphasize that no evidence of quantum advantage is presented in this work, and further research is required to evaluate the scalability and efficiency of QGANs compared to classical approaches. Future work could explore the incorporation of image-based data, alternative quantum circuit designs, and extensions to larger, more complex datasets. By framing QGANs as a tool to complement classical methods, this study contributes to the growing intersection of quantum computing and high-energy physics, providing a foundation for future research in the era of the HL-LHC.

Contents

List of Figures

Chapter 1

Introduction

In recent decades, the amount of data generated across many fields, including High Energy Physics (HEP), has grown exponentially. This has rise to the need for more advanced techniques and models to analyze information and develop accurate predictive models. This demand has driven the fast advancement of Artificial Intelligence (AI), a field that offers powerful tools to tackle these challenges.

At the same time, Quantum Computing has emerged as an exciting area of innovation. Although still in its early stages, this technology has generated significant interest due to its potential to revolutionize computation across various industries. By leveraging quantum phenomena such as superposition and entanglement, Quantum Computing promises to solve certain problems much more efficiently than classical approaches.

The combination of Machine Learning and Quantum Computing has opened up exciting new possibilities. This thesis focuses on one of these possibilities: using Quantum Generative Adversarial Networks (QGANs) to address some of the challenges in analyzing the complex data produced in HEP. By bringing together the flexibility of ML and the unique capabilities of Quantum Computing, QGANs could offer a new way to work with the large and detailed datasets in this field.

1.1 General Objective

This thesis aims to demonstrate the application of quantum computing in High Energy Physics by developing a QGAN capable of learning and replicating particle collision data distributions.

1.2 Specific Objectives

1. Design and implement a QGAN model using hybrid quantum-classical platforms such as TensorFlow Quantum and Cirq.
2. Train the QGAN on simulated datasets to produce quantum states that accurately replicate the multi-dimensional features of particle jets.
3. Evaluate the performance of the QGAN model, focusing on its ability to classify signal and background events, and analyze its results using accuracy metrics.

1.3 Chapter Overview

The thesis is organized as follows: Chapter 2 reviews the basics of machine learning, with an emphasis on Artificial Neural Networks and GANs. Chapter 3 introduces the fundamentals of

quantum computing, including quantum gates, circuits, and the basis of quantum neural networks. Chapter 4 provides an overview of the Large Hadron Collider (LHC), focusing on the CMS detector and how machine learning is used in particle physics experiments. Chapter 5 presents the design, implementation, and training of the QGAN model, along with an analysis of its performance. Finally, the conclusion summarizes the findings and discusses the broader implications of using quantum computing for advancing data analysis in HEP.

Chapter 2

Artificial Intelligence

The last three decades have seen an unprecedented increase in our ability to generate and analyze large datasets. This "big data" revolution has been spurred by an exponential increase in computing power and memory, commonly known as Moore's law. Computations that were unthinkable a few decades ago can now be routinely performed on laptops. Specialized computing machines (such as GPU-based machines) continue this trend towards cheap, large-scale computation, suggesting that the"big data" revolution is here to stay. This increase in our computational ability has been accompanied by new techniques for analyzing and learning from large datasets. These techniques draw heavily from ideas in statistics, computational neuroscience, computer science, and physics.

The European Commission and the OECD define Artificial Intelligence (AI) as systems capable of displaying intelligent behavior by analyzing their environment and taking autonomous actions – with some degree of autonomy – to achieve specific goals. [21]

A key example of AI is the technology behind self-driving cars. These vehicles are not programmed to handle every possible scenario they may encounter or are instructed on the exact actions to take in each situation (such as turning, accelerating, or braking). Instead, they autonomously process real-time data from their environment, making decisions on their own, similar to how a human would. This capacity for independent decision-making, based on learned information and without needing explicit commands for every situation, distinguishes AI from traditional programming approaches.

Machine learning is a subfield of artificial intelligence that develops algorithms that automatically learn from data. In particular, an artificially intelligent agent needs to recognize objects in its surroundings and predict the behavior of its environment to make informed choices. Therefore, ML techniques tend to focus more on prediction than estimation.

2.1 Machine Learning

Machine Learning has become one of the most influential branches of artificial intelligence. ML focuses on developing algorithms that allow machines to learn from data and improve their performance on specific tasks without being explicitly programmed. This approach has applications in various fields, from image processing and pattern recognition to predicting complex events, enabling significant advances in physics, medicine, and the tech industry.

Arthur Samuel, a pioneering figure in computer gaming and artificial intelligence, introduced

the term "Machine Learning" in 1959 during his time at IBM. He defined machine learning as *"the field of study that empowers computers to learn without explicit programming"*. Despite its foundational significance, machine learning lacks a universally accepted definition, with varying interpretations by different authors. Here is a more complete definition:

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.

Many problems in ML and data science start with the same ingredients. The first ingredient is the dataset $D = (X, y)$ X is a matrix of independent variables, and y is a vector of dependent variables. The second is the model, $f(x; \theta)$, which is a function, : $x \longrightarrow y$, of the parameters θ. That is, f is a function used to predict an output from a vector of input variables. The final ingredient is the cost function $C(y, f(X; \theta))$ that allows us to judge how well the model performs on the observations, y. The model is fit by finding the value of θ that minimizes the cost function.

2.2 Classification of Machine Learning models

Machine learning algorithms are categorized based on their intended outcomes, forming a taxonomy to guide their application. While various classifications exist, we will explore the primary ones.

2.2.1 Supervised learning

In supervised learning, the algorithm generates a function that maps the inputs to the desired outputs. The key difference is that this kind of learning deals with labeled data or already knows the result. Common supervised learning tasks include *classification and regression.*

2.2.2 Unsupervised learning

Unsupervised learning models a set of inputs without labeled examples. Instead, the algorithm identifies patterns or structures within the data, such as *clustering similar data points together, reducing dimensionality, or generative modeling.*

2.2.3 Reinforcement learning

In reinforcement learning, the algorithm learns a policy for decision-making based on interactions with an environment. An agent learns by interacting with an environment and changing behavior to maximize reward.
In this thesis, we will focus on Supervised and Unsupervised learning.

2.3 Data Processing

Data is a critical component in any machine learning project and can generally be categorized into two types: **structured data** and **unstructured data**. Structured data is highly organized and typically comes in a tabular format with rows and columns, while unstructured data, such as images, text, or audio, lacks this clear structure.

The first step in any data analysis workflow is to **understand and explore** the dataset. This involves reading the documentation (if available) to gain insights into the data's format, features, and potential issues. Understanding the data's characteristics is essential for making informed decisions during the cleaning and preprocessing. By examining the data's initial state, you can identify anomalies, missing values, and patterns that will guide your next steps.

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 or imputing them using statistical methods such as replacing them with the respective feature's mean, median, or mode.
- 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.

After cleaning, the data often requires additional **data processing** to prepare it for model training. This can involve transforming the data into more convenient formats (e.g., converting raw data into images or reshaping data for specific algorithms) or creating new features that may improve the model's performance. In classification tasks, it's important to check for data imbalance, where one class may dominate over others, which can bias the model. Techniques like oversampling, undersampling, or using weighted loss functions can help mitigate this issue.

One crucial preprocessing step is normalization or scaling, ensuring all features are on a similar scale. This is especially important for machine learning models, as it promotes faster convergence during training and helps avoid issues where features with more extensive numerical ranges might dominate models.

Figure 2.1: Data processing workflow.

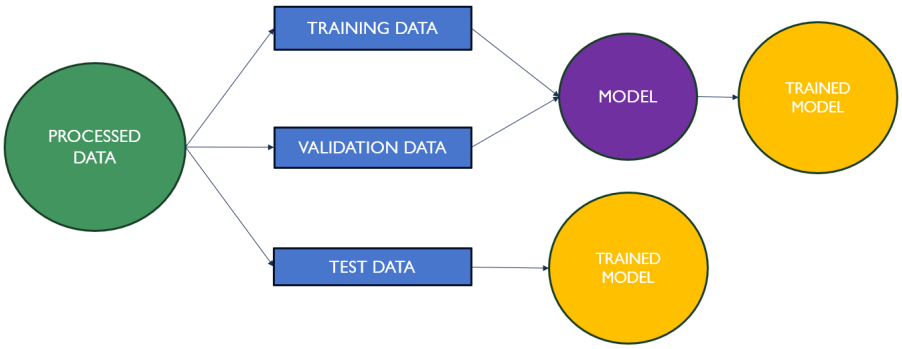


Figure 2.2: Data workflow for Machine Learning.

Finally, to effectively train machine learning models, it is necessary to split the data into training, validation, and test sets. This division is essential for evaluating the model's generalization capability. The training set fits the model, while the validation set helps fine-tune hyperparameters and prevent overfitting. The test set is kept completely unseen during training and is used to evaluate the final performance of the model on new, unseen data. This separation helps ensure that the model's performance is not overly optimistic and reflects how well it will perform in real-world scenarios.

2.4 Deep Learning

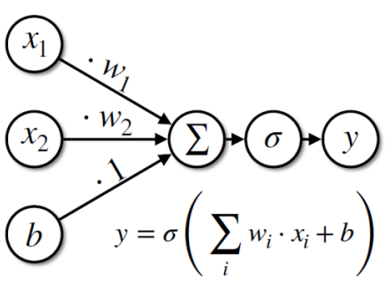
Deep Learning is a subfield of Machine Learning based on **Artificial Neural Networks** (ANNs), a class of algorithms loosely inspired by the structure and functioning of the human brain. ANNs comprise layers of interconnected nodes (called **neurons**).
This section delves into the fundamental concepts of deep learning, beginning with a detailed overview of artificial neural networks, their components, and their role in the learning process. We will then explore a well-known application of deep learning in image classification using the MNIST dataset, followed by a discussion of deep learning's broader applications in various domains.[8]

2.4.1 Perceptron

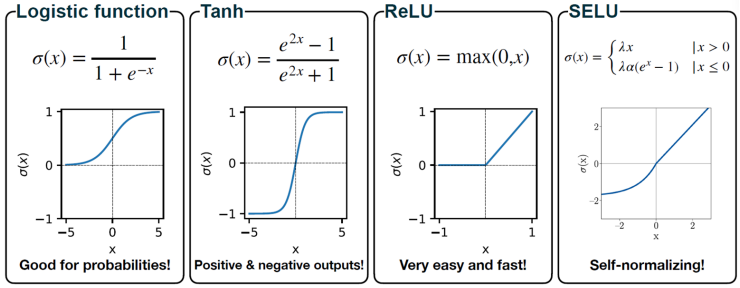
A perceptron is the fundamental building block of Artificial Neural Networks. It represents a simple computational model that consists of two key parameters: **weights** (W) and **bias** (b). These parameters and an **activation function** determine whether the perceptron "fires" or activates, producing an output. This idea was presented in 1958 by Frank Rosenblatt in [41]
The perceptron follows a straightforward process:

1. Input data, denoted as x, is fed into the perceptron. Each input is associated with a weight w, which adjusts the influence of the input on the model.
2. The perceptron computes the dot product between the input data and the corresponding weights, given by $w^T \cdot x$.
3. A bias term b is added to the dot product to introduce flexibility in shifting the activation threshold. The sum is computed as $\sum w^T \cdot x + b$.
4. This result is passed through an activation function σ, which determines the final output of the perceptron:
$$y = \sigma\left(\sum w^T \cdot x + b\right).$$

The activation function is crucial for introducing non-linearity to a model, enabling it to solve complex problems beyond simple linear decision boundaries. Common activation functions include the sigmoid (Logistic), hyperbolic tangent (tanh), Rectified Linear Unit (ReLU), and Self Linear Unit (SELU). Each function has its advantages depending on the problem and data at hand. The sigmoid function is often used in binary classification but can suffer from vanishing gradients, while Tanh improves on this but may still face gradient issues in deep networks.
ReLU is widely used in deep learning due to its simplicity and effectiveness in promoting sparsity. However, if many neurons output zeros, it can cause dead neurons. The SELU function, an extension of ReLU, ensures self-normalization, which can be helpful in intense networks. [44]



(a) Perceptron structure.



(b) Most common activation functions in deep learning.

2.4.2 Artificial Neural Networks (ANNs)

Artificial Neural Networks (ANNs) are at the core of deep learning models, designed to mimic the functioning of neural networks in the human brain. These networks are composed of interconnected neurons organized in layers, each receiving inputs from the neurons in the previous layer. ANNs learn by adjusting the connections (weights and biases) between neurons based on the data, improving their ability to make accurate predictions or classifications over time. [31] [2]
The process begins with the **input data**, which can take various forms such as images, text, or audio. For instance, the input would consist of pixel values in an image recognition task. This input data is typically represented as a vector and then passed into the network. Each connection between the neurons carries an associated **weight**, which reflects the strength of the connection and determines how much influence the input data has on the neuron's output. Neurons also have a **bias** that allows them to adjust the output independently of the input data. Together, these weights and biases form the parameters of the network. In this context, learning involves optimizing these parameters to reduce the error between the network's predictions and the actual outcomes. [4]
At the heart of each neuron's processing is the computation of the **dot product** between the input vector and the neuron's weights, followed by the addition of the bias. This mathematical operation generates a weighted sum of the inputs, which is represented as:

$$z = \mathbf{w}^T \mathbf{x} + b$$

The next critical step in a neuron's operation involves the application of an **activation function**. This function introduces non-linearity into the network, which is essential because the network would only be capable of learning linear relationships without it. In the real world, most problems are non-linear, so the activation function allows the network to model complex data patterns effectively.
The **output** of the neural network varies depending on the task it is designed to solve. For classification tasks, the output layer typically uses a softmax function to convert the network's raw outputs into probabilities for different classes. For regression tasks, the output is often a continuous value. The structure of the output layer and the choice of loss function are crucial for the network's performance.
Neurons, the computational units within the network, are organized into **layers**. The network generally consists of three types of layers: *the input layer, hidden layers, and the output layer*. The input layer receives the raw data, the hidden layers perform intermediate computations, and the output layer produces the final result, whether a classification label or a continuous value. A key

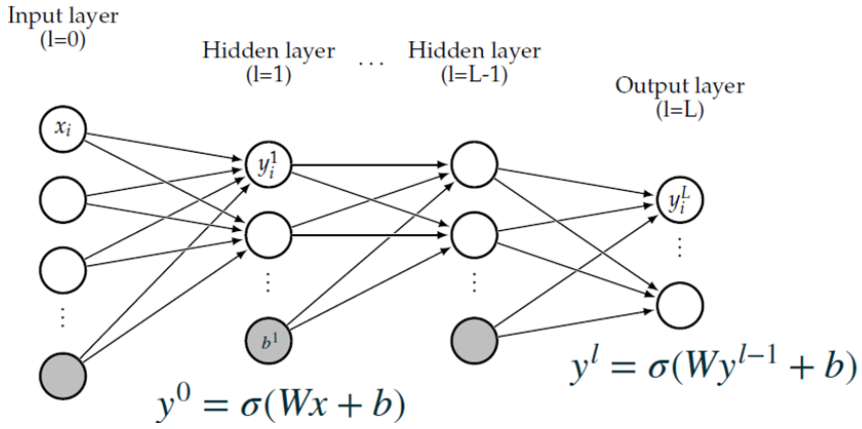


Figure 2.4: Deep Neural Networks model.

advantage of deep learning models is their multiple hidden layers, allowing the network to capture intricate patterns in the data. Earlier layers might detect simple patterns like edges in an image in a deep network, while later layers can identify more complex structures like objects or faces.

2.4.3 Learning

The learning process in deep learning involves adjusting the weights of a neural network through a technique called backpropagation. This process helps the network minimize the difference between its predicted outputs and target values. The goal is to progressively reduce errors and improve performance through multiple passes over the data, which leads to the network learning meaningful patterns. A good place to learn more about Neural Networks and Deep Learning is the platform of Michael Nielsen. [36]

Loss function
The loss function quantifies the difference between the network's predictions and the actual target values during training. It measures how well the model is performing and provides the necessary feedback for weight adjustments. Two of the most common loss functions are:

- Mean Squared Error (MSE): Typically used for regression tasks, it calculates the average of the squared differences between predicted and actual values.

$$\mathcal{L}_{\text{MSE}} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.1}$$

  where:

  - N is the number of samples.
  - y_i is the true value (ground truth).
  - $\hat{y}_i$ is the predicted value.

- Cross-Entropy Loss: Commonly used in classification tasks, this loss function measures the dissimilarity between the predicted probability distribution and the true distribution.

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right] \tag{2.2}$$

  where:

  - N is the number of samples.
  - $y_i \in \{0, 1\}$ is the true binary label.
  - $\hat{y}_i \in (0, 1)$ is the predicted probability.

The choice of the loss function is crucial, as it directly affects how the network learns and what it prioritizes during training.

Optimizers
Optimizers are algorithms or methods used to update the neural network weights to minimize the loss function. They play a key role in the efficiency and convergence of the learning process. Popular optimizers include:

- Stochastic Gradient Descent (SGD): One of the most basic optimizers, SGD updates the network's weights by computing the gradient of the loss concerning the weights and taking

small steps in the direction that minimizes the loss. However, vanilla SGD can converge slowly and sometimes get stuck in local minima.

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta C(\theta_t) \tag{2.3}$$

where:

- θ_t represents the parameters at iteration t.
- η is the learning rate.
- $\nabla_\theta C(\theta_t)$ is the gradient of the loss function concerning the parameters.

- Adam (Adaptive Moment Estimation): An advanced optimizer that adapts the learning rate for each parameter by estimating the gradients' first and second moments. Adam tends to perform well in practice and is widely used for training deep learning models due to its faster convergence. [26]

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \nabla_\theta C(\theta_t) \tag{2.4}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) \nabla_\theta C(\theta_t)^2 \tag{2.5}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.6}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \tag{2.7}$$

where:

- m_t and v_t are the gradient's first and second-moment estimates.
- β_1, β_2 are the decay rates (typically 0.9 and 0.999, respectively).
- $\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected versions of m_t and v_t.

- RMSProp: Another adaptive learning rate method, particularly effective in dealing with non-stationary objectives (i.e., those that change over time), is useful in handling large, noisy datasets.

$$v_t = \beta v_{t-1} + (1 - \beta) \nabla_\theta C(\theta_t)^2 \tag{2.8}$$

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t + \epsilon}} \nabla_\theta C(\theta_t) \tag{2.9}$$

where:

- v_t is the moving average of the squared gradients.
- β is the decay factor (typically 0.9).
- ϵ is a small term to prevent division by zero.

Training over epochs

Training a neural network is an iterative process, and the entire dataset is usually passed through the network multiple times in cycles called **epochs**. Each epoch consists of several iterations, where a batch of data is fed into the network, and the weights are updated according to the optimizer's rules.

The number of epochs determines how often the model will process the entire training dataset. Significant weight adjustments are made early in training, but as the training progresses and the loss decreases, the updates become smaller. Typically, as more epochs are completed, the model

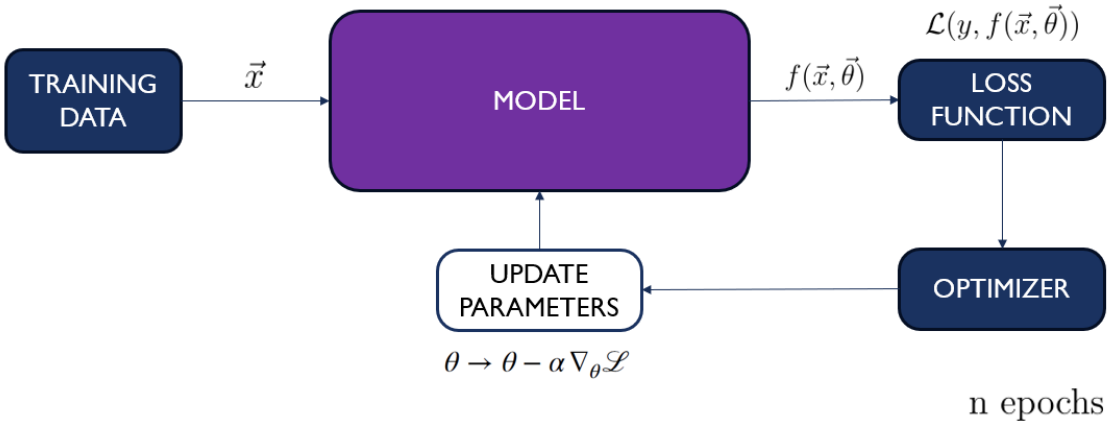


Figure 2.5: Overview of the training process in deep learning models.

continues to improve, but **overfitting** can occur if the network is trained for too long. Monitoring the performance on a validation set helps determine the optimal number of epochs.

Hyperparameters
Hyperparameters are predefined values that significantly influence the performance and efficiency of a neural network. Unlike the model's weights, which are learned during training, hyperparameters must be set prior to training and require careful tuning to achieve optimal results. Key hyperparameters in deep learning include the **learning rate, batch size, network architecture, and regularization techniques.**
The learning rate determines the step size for updating weights during training, and setting this parameter too high risks overshooting the optimal solution. At the same time, a learning rate that is too low may result in slow convergence. *The batch size*, which specifies the number of training examples processed in a single iteration, balances computational resources and training stability. Smaller batch sizes produce noisier gradient updates and consume less memory, whereas larger batch sizes generate smoother gradients but demand greater computational power.
The network architecture, defined by the number of layers and the number of neurons per layer, dictates the model's capacity to capture complex patterns in the data. Deeper networks can learn intricate relationships but are more susceptible to overfitting. Techniques like dropout are employed to mitigate overfitting, where a fraction of neurons are randomly deactivated during training. This forces the model to learn more generalized and robust features.
Choosing the appropriate hyperparameters is essential to the neural network's performance. Hyperparameter tuning often involves trial and error, guided by experience and experimentation.

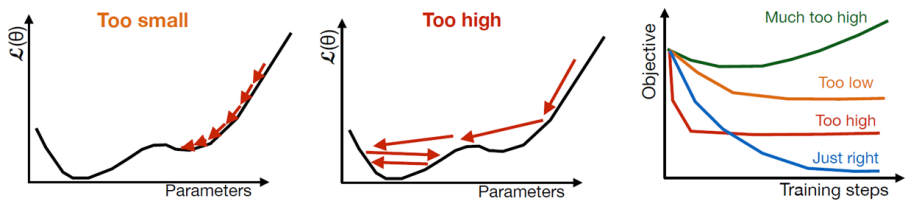


Figure 2.6: Different values for learning rate.

2.5 Generative Adversarial Networks

Ian Goodfellow and his collaborators introduced generative adversarial networks (GANs) in 2014 as a novel framework for training generative models. citegoodfellow2014gan A GAN consists of two neural networks: a textbfgenerator and a textbfdiscriminator, which are taught simultaneously through adversarial processes. The generator aims to produce data that mimic the true distribution of the training set, while the discriminator distinguishes between real data and the generator's synthetic outputs. The interplay between these networks allows GANs to produce highly realistic data samples, making them valuable tools in various fields, including image generation and data augmentation.

2.5.1 Generator

The generator is a neural network that learns to map a random input, typically from a latent space, to the target data distribution. Its primary goal is to produce synthetic samples that are as close to real data as possible. During training, the generator aims to "fool" the discriminator by generating data that the discriminator classifies as real. Mathematically, the generator optimizes a loss function that encourages it to minimize the discriminator's ability to differentiate between real and generated data. The architecture of the generator can vary depending on the complexity of the data, often consisting of fully connected layers or convolutional layers for image data.

2.5.2 Discriminator

The discriminator acts as a binary classifier that distinguishes real data from the synthetic data generated by the generator. It receives real data from the training set and fake data from the generator as inputs. The discriminator is trained to assign a high probability to real data and a low likelihood to fake data. This network's optimization process is critical to the adversarial nature of GANs, as it indirectly improves the generator's performance by forcing it to produce more realistic samples. The discriminator's architecture typically mirrors a standard neural network classifier, using convolutional layers for image-based data or fully connected layers for other data types.

2.5.3 Training

Training a GAN involves alternating between updating the generator and the discriminator in a process often referred to as adversarial training. The training procedure follows a min-max opti-

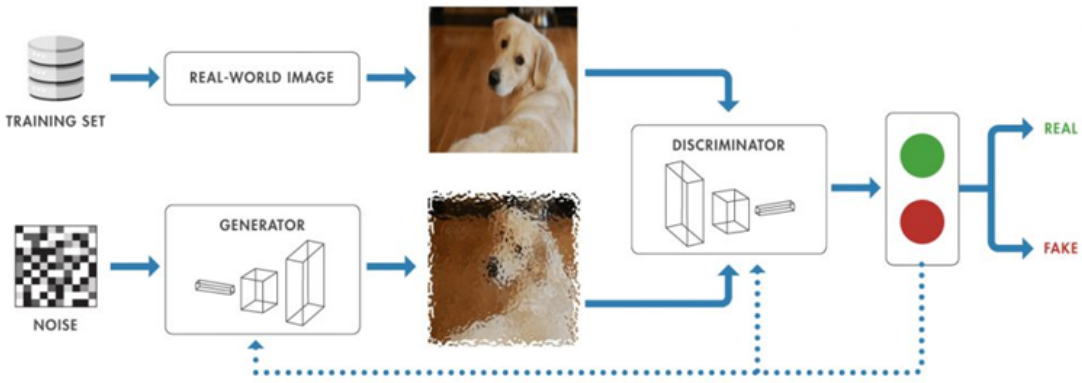


Figure 2.7: Generative Adversarial Networks overview.

mization, where the generator minimizes the likelihood of the discriminator correctly identifying its outputs. In contrast, the discriminator maximizes its ability to distinguish between real and fake data. The combined cost function can be expressed as:

$$\min_G \max_D \mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z)))] \tag{2.10}$$

Where $G(z)$ represents the output of the generator given input z sampled from a prior distribution, and $D(x)$ is the probability assigned by the discriminator to a real data point x. Training can be challenging due to issues like mode collapse, where the generator produces limited variations of data, and instability, which may occur if the discriminator overpowers the generator or vice versa. Various techniques, such as feature matching, batch normalization, and more advanced architectures (e.g., Wasserstein GAN), have been introduced to stabilize training.

2.5.4 Evaluation

Evaluating GANs is particularly challenging due to lacking a straightforward objective metric. Traditional methods, such as calculating the loss of the generator and discriminator, may not directly correlate with the quality of the generated data. Instead, several evaluation metrics have been proposed to assess the performance of GANs:

- **Inception Score (IS):** The Inception Score evaluates the quality and diversity of generated samples using a pre-trained Inception network. This metric computes the classification confidence of the generated samples by analyzing the probabilities of the predicted class labels. Specifically, it considers two aspects:
  - *Quality:* High confidence in class predictions (sharp distributions) indicates that the generated samples resemble realistic, well-defined categories.
  - *Diversity:* A diverse set of generated samples will cover a wide range of class labels, ensuring output variability.

  A high IS suggests that the generated data is diverse and high-quality.

- **Frechet Inception Distance (FID):** The Frechet Inception Distance measures the similarity between real and generated data distributions. It uses a pre-trained Inception network to extract feature vectors for real and generated samples, effectively mapping them into a high-dimensional feature space. The FID then computes the Wasserstein-2 distance (also called Frechet distance) between the feature distributions of the two datasets, characterized by their means and covariances. Mathematically, it is given by:

  $$\text{FID} = ||\mu_r - \mu_g||^2 + \text{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}),$$

  Where μ_r, μ_g are the means and Σ_r, Σ_g are the covariance matrices of the actual and generated data distributions, respectively. A lower FID score indicates that the generated data closely matches the real data in terms of both content and diversity, making it a reliable metric for evaluating the quality of generative models.

For specific applications in high-energy physics, the evaluation of GANs might also involve domain-specific metrics, such as comparing generated data to simulated or real experimental data based on physics-driven characteristics, including energy distributions or particle momenta.

Chapter 3

Quantum Computing

3.1 Introduction

Quantum computing represents a significant paradigm shift in computational theory, one that exploits the principles of quantum mechanics to perform certain types of calculations more efficiently than classical computers. With potential cryptography, optimization, and quantum simulation applications, quantum computing has gained substantial interest from academia and industry. This chapter provides an overview of the fundamental principles, hardware, and algorithms of quantum computing, along with the current challenges and applications in the near term.

3.2 Fundamentals of Quantum Computing

Richard Feynman once said, "Nature isn't classical, and if you want to simulate nature, you would better make it quantum mechanical." This statement encapsulates the core idea behind quantum computing, using quantum mechanics as a basis for computation to model physical processes that classical computers struggle to simulate effectively. Quantum computing leverages fundamental quantum properties, which are outlined below. [37]

3.2.1 Quantum Principles

The key points of quantum computing are that exploit the following properties:

- **Superposition**: Unlike classical bits, which exist in a state of either 0 or 1, quantum bits or qubits can exist in a combination of states, known as Superposition. Mathematically, a qubit's state can be represented as $\psi = \alpha|0\rangle + \beta|1\rangle$, where α and β are complex amplitudes with $|\alpha|^2 + |\beta|^2 = 1$, this normalization condition is due to its probability representation.
- **Coherence**: Coherence refers to the persistence of the quantum state over time. Quantum information systems must maintain Coherence long enough to perform complex computations, but interactions with the external environment often cause decoherence, posing a significant challenge.
- **Entanglement**: When two qubits become entangled, the state of one qubit is directly related to the state of the other, regardless of the distance between them. Entanglement is crucial for many quantum algorithms as it allows for complex correlations that classical systems cannot replicate.

- **Measurement**: Measurement collapses the qubit's state into one of its basis states, 0 or 1, based on the probability amplitudes. This introduces a probabilistic nature to quantum computing, as the outcome of each Measurement is not deterministic.

This work implements quantum computing concepts using the PennyLane and Cirq frameworks. Developed by Xanadu, PennyLane was initially designed to differentiate hybrid quantum-classical computations automatically. Over time, it has evolved into a versatile tool capable of executing a wide range of quantum algorithms, including those for quantum machine learning and quantum simulations [9]. Cirq, developed by Google, complements this by providing a robust framework tailored explicitly for designing, simulating, and running quantum circuits on near-term quantum hardware.

3.2.2 Quantum Computing: Hardware

The physical realization of quantum computers requires specialized hardware to manipulate qubits while preseCoherenceerence. Some of the most prominent hardware platforms include:
Superconducting Qubits: Relying on superconducting circuits cooled to near absolute zero, these qubits are manipulated using microwave pulses. Companies like IBM and Google favor this approach.
Ion Traps: Trapped ions are used as qubits and manipulated using laser beams. Ion trap systems are characterized by long coherence times and high-fidelity operations and are championed by companies like IonQ.
Photonic Quantum Computing: Photonic systems use photons as qubits and are manipulated with beam splitters, phase shifters, and mirrors. Photonic qubits offer the advantage of room-temperature operation and natural scalability for networking.

3.3 Two-Level Systems: Qubits

A two-level system is the most straightforward representation of a quantum system and forms the basis of quantum computing, where qubits replace classical bits. This approach is essential as the unit of information in a quantum system is not restricted to discrete values of 0 and 1 but can exist in a combination of both states due to quantum Superposition. In Dirac notation:

$$\psi = \alpha|0\rangle + \beta|1\rangle$$

Now, we are describing a helpful concept that will help visualize a one-qubit state.

3.3.1 Bloch Sphere Representation

The **Bloch sphere** is a geometric representation of a qubit in three-dimensional space, which allows for the visualization of pure quantum states. On this sphere, any qubit state can be represented as a point on its surface using a unit vector:

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

where θ and ϕ are angles that define the position on the sphere, with $|0\rangle$ and $|1\rangle$ representing the qubit's basis states. This formalism allows visualizing the action of quantum gates as rotations on the sphere.

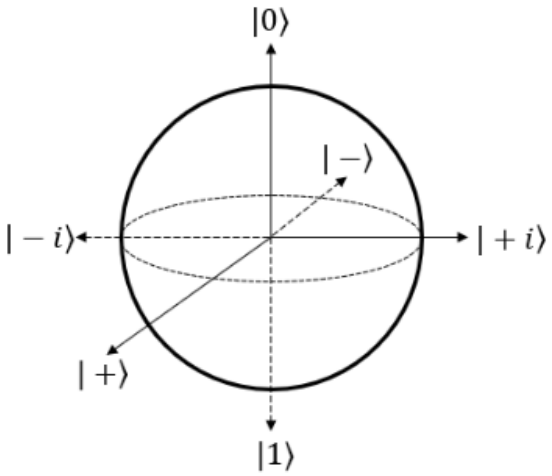


Figure 3.1: The Bloch sphere represents the state of a single qubit, with the three main bases shown: the computational basis $|0\rangle$ and $|1\rangle$ along the Z-axis, the $|+\rangle$ and $|-\rangle$ states along the X-axis, and the $|i\rangle$ and $|-i\rangle$ states along the Y-axis.

3.3.2 Dirac Notation

In quantum mechanics and quantum computing, **Dirac notation** (or bra-ket notation) is an essential tool to describe quantum states and their operations within a Hilbert space. This complex vector space accommodates quantum states. Dirac notation provides a concise, structured way to handle vectors and inner products central to quantum mechanics.

Representing the Qubit States in Dirac Notation

A qubit exists in a two-dimensional complex vector space, commonly represented as $\mathbb{C}^2$. A single qubit's two primary basis states, analogous to the classical binary states 0 and 1, are denoted as $|0\rangle$ and $|1\rangle$ in Dirac notation. These states are represented as column vectors in matrix notation:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Usually, $|0\rangle$ and $|1\rangle$ are used as the basis. However, it's common to find the following basis too:

$$|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \quad \text{and} \quad |-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) \qquad \text{or}$$
$$|i\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle) \quad \text{and} \quad |-i\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle)$$

A general qubit state $|\psi\rangle$ can be expressed as a linear combination (or Superposition) of the standard basis states:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

where α and β are complex numbers that satisfy the normalization condition $|\alpha|^2 + |\beta|^2 = 1$. In matrix form, this state can be written as:

$$|\psi\rangle = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}.$$

Bra and Ket Notation

Dirac notation separates vectors into two types:

- **Kets**: Represent column vectors, denoted as $|\psi\rangle$.
- **Bras**: Represent row vectors (conjugate transposes of kets), denoted as $\langle\psi|$.

For a state $|\psi\rangle = \begin{pmatrix}\alpha\\ \beta\end{pmatrix}$, the corresponding bra is written as:

$$\langle\psi| = \begin{pmatrix}\alpha^* & \beta^*\end{pmatrix},$$

where α^* and β^* are the complex conjugates of α and β, respectively.

Inner Products and Measurement Probabilities

The **inner product** (or dot product) between two states, say $|\psi_1\rangle$ and $|\psi_2\rangle$, is represented as $\langle\psi_1|\psi_2\rangle$ in Dirac notation. This inner product is calculated as:

$$\langle\psi_1|\psi_2\rangle = \begin{pmatrix}\alpha_1^* & \beta_1^*\end{pmatrix}\begin{pmatrix}\alpha_2\\ \beta_2\end{pmatrix} = \alpha_1^*\alpha_2 + \beta_1^*\beta_2.$$

The inner product has significant physical meaning in quantum mechanics, particularly Measurement. If $|\psi_1\rangle$ and $|\psi_2\rangle$ are two quantum states, the magnitude squared of the inner product $|\langle\psi_1|\psi_2\rangle|^2$ gives the probability of finding the system in state $|\psi_2\rangle$ if it was initially prepared in state $|\psi_1\rangle$.

Tensor Product for Multi-Qubit Systems

When dealing with multiple qubits, Dirac notation, and the tensor product represent the system's combined state. If we have two qubits, each in states $|\psi_1\rangle = \alpha_1|0\rangle + \beta_1|1\rangle$ and $|\psi_2\rangle = \alpha_2|0\rangle + \beta_2|1\rangle$, their joint state $|\psi_1\rangle \otimes |\psi_2\rangle$ in the larger, four-dimensional space $\mathbb{C}^4$ is written as:

$$|\psi_1\rangle \otimes |\psi_2\rangle = (\alpha_1|0\rangle + \beta_1|1\rangle) \otimes (\alpha_2|0\rangle + \beta_2|1\rangle).$$

Using the distributive property of the tensor product, this expands to:

$$|\psi_1\rangle \otimes |\psi_2\rangle = \alpha_1\alpha_2|00\rangle + \alpha_1\beta_2|01\rangle + \beta_1\alpha_2|10\rangle + \beta_1\beta_2|11\rangle.$$

In matrix notation, each of the basis states for the two-qubit system is represented as:

$$|00\rangle = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}, \quad |01\rangle = \begin{pmatrix}0\\1\\0\\0\end{pmatrix}, \quad |10\rangle = \begin{pmatrix}0\\0\\1\\0\end{pmatrix}, \quad |11\rangle = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}.$$

You can notice that the number of quantum states increases as 2^n, where n is the number of qubits in the quantum circuit.

3.4 Quantum Gates and Circuits

Quantum gates are the equivalent of classical logic gates used to manipulate quantum states. They are represented by unitary matrices that act on the state vectors of qubits in Hilbert space. When combined, these gates form **quantum circuits**, the foundation of quantum computations.
On the other hand, a **quantum circuit** is a sequence of quantum gates applied to qubits to accomplish a computational task. Quantum circuits are represented sequentially, where qubit agates manipulate the qubit's initial state to a final state by circuit representation, including collisions, which collapse the quantum state into classical information. [7]

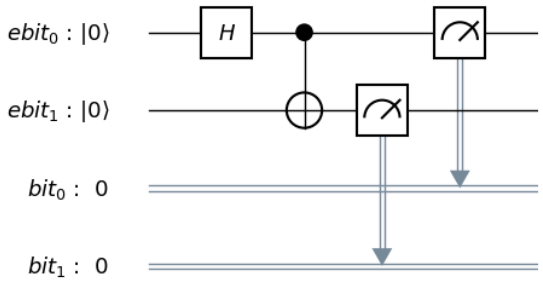


Figure 3.2: Example of a Quantum Circuit: This circuit generates a Bell state, resulting only in the states $|00\rangle$ or $|11\rangle$.

3.4.1 Single-Qubit Gates

Single-qubit gates are fundamental unitary operations that manipulate the state of an individual qubit. The following are key single-qubit gates, each with its matrix representation and action on basis states in Dirac notation:

- **Pauli-X Gate (NOT Gate)**: The Pauli-X gate acts as a quantum NOT Gate, flipping the state of the qubiGateom $|0\rangle$ to $|1\rangle$ and vice versa. In matrix form, it is represented as:

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

  Its action in Dirac notation is:

$$X|0\rangle = |1\rangle, \quad X|1\rangle = |0\rangle.$$

- **Pauli-Y Gate**: The Pauli-Y gate introduces a π rotation around the Y-axis of the Bloch sphere. In matrix form:

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

  In Dirac notation:

$$Y|0\rangle = i|1\rangle, \quad Y|1\rangle = -i|0\rangle.$$

- **Pauli-Z Gate**: The Pauli-Z gate performs a π rotation around the Z-axis of the Bloch sphere. It leaves $|0\rangle$ unchanged and applies a phase shift to $|1\rangle$. Its matrix representation is:

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

  In Dirac notation:

$$Z|0\rangle = |0\rangle, \quad Z|1\rangle = -|1\rangle.$$

- **Hadamard Gate (H)**: The Hadamard gate creates a superposition by transforming $|0\rangle$ to an equal superposition of $|0\rangle$ and $|1\rangle$, and similarly for $|1\rangle$. It is represented by:

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

  In Dirac notation:

$$H|0\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}} = |+\rangle, \quad H|1\rangle = \frac{|0\rangle - |1\rangle}{\sqrt{2}} = |-\rangle.$$

- **Rotation Gates (RX, RY, RZ)**: These gates provide continuous rotations around the X, Y, and Z axes, parameterized by an angle θ. For example, the $RZ(\theta)$ gate rotates a qubit around the Z-axis by an angle θ:

$$RZ(\theta) = \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix}.$$

  In Dirac notation:

$$RZ(\theta)|0\rangle = e^{-i\theta/2}|0\rangle, \quad RZ(\theta)|1\rangle = e^{i\theta/2}|1\rangle.$$

These gates form the building blocks of quantum circuits, allowing for various operations necessary for quantum algorithms.

3.4.2 Multi-Qubit Gates

Multi-qubit gates enable operations on multiple qubits simultaneously and are essential for implementing **entanglement**, a phenomenon unique to quantum systems and critical for many quantum algorithms.

3.4.3 Separable and Non-Separable Gates

In multi-qubit systems, it is crucial to distinguish between **separable** and **non-separable** gates. Separable gates act independently on qubits, meaning they do not create entanglement. Non-separable gates, such as the **CNOT gate**, act on two qubits in a way that links their states, which is essential for creating entangled states.

CNOT Gate (Controlled-NOT)

The CNOT (Controlled-NOT) gate is a two-qubit gate where one qubit functions as the control and the other as the target. If the control qubit is in state $|1\rangle$, the CNOT gate flips the state of the target qubit; otherwise, the target qubit remains unchanged. In matrix form:

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

Its action in Dirac notation is:

$$\text{CNOT}|00\rangle = |00\rangle, \quad \text{CNOT}|01\rangle = |01\rangle, \quad \text{CNOT}|10\rangle = |11\rangle, \quad \text{CNOT}|11\rangle = |10\rangle.$$

Toffoli Gate (CCNOT)

The Toffoli gate, or CCNOT, is a three-qubit gate that acts as an extension of the CNOT gate. It flips the state of the third qubit (target) only if the first two qubits (controls) are both in the state $|1\rangle$. The matrix representation of the Toffoli gate is:

$$\text{Toffoli} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}.$$

In Dirac notation:

$$\text{Toffoli}|110\rangle = |111\rangle, \quad \text{Toffoli}|111\rangle = |110\rangle,$$

With other states remaining unchanged.

3.4.4 Universal Quantum Gates

Universal quantum gates can approximate any unitary operation in a quantum system to arbitrary precision. Specifically, any quantum operation can be implemented using single-qubit gates combined with the CNOT gate. These universal gate sets are essential for constructing quantum algorithms, as they provide a foundation for any operation within a quantum system.

$$|\psi\rangle = \mathcal{U}|\psi_0\rangle,$$

where $\mathcal{U}$ represents a unitary operator composed of a sequence of quantum gates acting on the initial quantum state $|\psi_0\rangle$. This operation transforms $|\psi_0\rangle$ into the final state $|\psi\rangle$, encoding the desired computation or transformation applied by the quantum circuit.

3.4.5 Challenges of Quantum Computing

The journey towards a scalable, fault-tolerant quantum computer is fraught with technical challenges. A significant hurdle is building quantum hardware that can operate with high fidelity while preseCoherenceeerence. Additionally, the susceptibility to quantum noise requires advanced error correction methods. Potential applications, such as cryptographic security, combinatorial optimization, and machine learning, could be transformative. Yet, these applications depend on advancements in quantum hardware and algorithmic innovation to handle noise and decoherence. In this project, we will focus on near-term quantum algorithms, a method for running algorithms in current quantum and vicestakestakesg advantage of variational quantum algorithms.

3.5 Near Term Quantum Algorithms

Quantum computing is recognized as a transformative technology across various industries, including finance, pharmaceuticals, materials science, and cryptography. Despite the rapid progress in recent years, we are still in the Noisy Intermediate-Scale Quantum (NISQ) era, where current devices contain tens to thousands of qubits prone to quantum noise. In this section, we explore techniques tailored for NISQ devices, which aim to harness these devices' potential while mitigating the effects of noise. [25]
Several approaches have been developed to utilize NISQ devices effectively. Among them are variational quantum algorithms, error mitigation strategies, quantum circuit compilation, and benchmarking protocols. These techniques play an essential role in enhancing the robustness of quantum computations against noise and pushing the limits of what NISQ devices can achieve. Additionally, classical simulations are integral to validating these algorithms, providing a basis for further advancements in quantum computing.

3.5.1 Variational Quantum Algorithms

Variational Quantum Algorithms (VQAs) combine classical optimization with quantum circuits to solve complex problems in chemistry, optimization, and machine learning. These algorithms operate by preparing a quantum state with trainable parameters and optimizing these parameters based on a classical cost function, creating a hybrid approach that leverages the strengths of both quantum and classical resources.

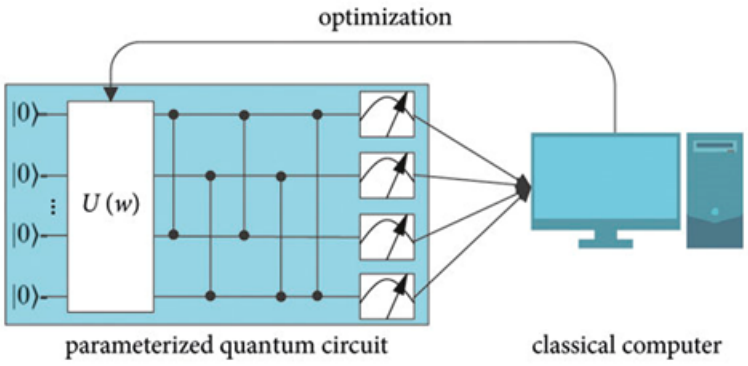


Figure 3.3: Parameterized quantum circuits are trained using classical computers.

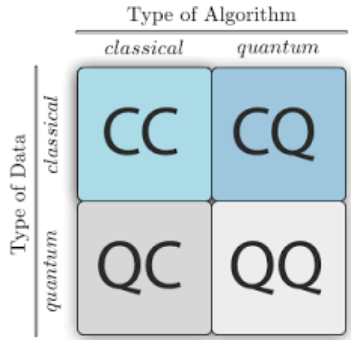


Figure 3.4: Possible approaches in computational tasks.

3.6 Quantum Machine Learning

Quantum machine learning (QML) explores the fusion of quantum mechanics with machine learning to develop models that are either fully quantum or hybrid in nature. This field can utilize both classical and quantum data to implement quantum algorithms. The core aspect of QML involves Variational quantum circuits with adjustable parameters, which enable quantum systems to emulate neural networks when used alongside classical optimizers and loss functions. [6] [5]

3.6.1 Quantum Neural Network

Quantum Neural Networks (QNNs) represent the quantum counterpart to classical neural networks. Instead of traditional neurons, QNNs use qubits and quantum gates to perform computations, enabling them to model and learn complex functions in a way that potentially leverages quantum advantages. By exploiting quantum Superposition, entanglement, and interference, QNNs hold promise for handling complex, high-dimensional data in ways that classical networks may find computationally challenging or intractable. [46] [11] [3]
The main components of a QNN include the following:

- Data Embedding: A classical input is mapped into a quantum state through a quantum feature map $\phi : X \to H$, where H is a Hilbert space and $x \to |\phi(x)\rangle$ represents a classical input transformed into a quantum state via a unitary operation $U_\phi(x)$.
- Ansatz (Variational Quantum Circuit): A variational quantum circuit consists of quantum gates with trainable parameters that are adjusted during training to optimize the model.

Typically, these circuits use rotation gates that apply tunable rotations to qubits.

$$|\psi(\theta, x)\rangle = \mathcal{U}(\theta)|\psi(x)\rangle$$

- Measurement: Once the quantum state is prepared, one or more qubits are measured to obtain the output. The Measurement is typically performed concerning the Pauli-Z observable, yielding expectation values contributing to the final prediction.

$$y(\theta, x) = \langle\psi(\theta, x)|O|\psi(\theta, x)\rangle$$

Data Embedding

When working with classical data, the first step in constructing a QNN is embedding classical data into a quantum circuit. This process requires transforming classical data into quantum states using specific quantum gates, such as rotational gates, that encode the data into the amplitudes or angles of qubit states. The choice of embedding method can significantly impact the model's performance and efficiency. [39] [7]

- **Amplitude Embedding:** This technique encodes data by embedding each entry into the amplitude of a quantum state. Amplitude embedding is highly efficient because it allows encoding 2^n data points into a system with only n qubits, leveraging quantum parallelism. However, this method often requires normalization of the data and is generally used for datasets with large numbers of features.
- **Angle Embedding:** Here, each data point is embedded as a parameter (angle) of a quantum gate, typically a rotational gate such as the RZ gate. Angle embedding enables the encoding of n data points into a quantum system with n qubits and is well-suited for applications where interpretability and simplicity in embedding are needed.
- **Data Re-uploading:** This method presented by Adrián Pérez-Salinas et Al. [40] extends angle embedding by applying the same rotational gates multiple times within the quantum circuit. By re-uploading the data several times, it increases the expressivity of the circuit and enables the encoding of complex features using a relatively simple quantum architecture. Data re-uploading allows encoding n entries in a quantum system with n qubits while enhancing the model's capacity to learn complex functions.

The selection of a data embedding method depends on the nature and dimensionality of the data. Amplitude embedding, for instance, is often preferred for high-dimensional data (e.g., images) due to its efficiency. Once data is embedded, the next step is constructing a parametrized quantum circuit, also known as a variational quantum circuit.

Variational Quantum Circuit

A variational quantum circuit (VQC) is a quantum circuit with trainable parameters, often represented by quantum rotational gates that act on the qubits. Each parameterized Gate in the circuit is controlled by a variate parameter, denoted as θ, which can be adjusted during training. These adjustable gates form the unitary operation $U(\theta)$, allowing the circuit to learn data representations. VQCs are optimized by adjusting these parameters to minimize a cost function, similar to the optimization process in classical neural networks. [18]
Variational quantum circuits are key to QNNs as they introduce nonlinearity and complexity. They allow the model to approximate intricate functions and relationships in the data. Training a VQC typically involves a classical optimizer that iteratively updates the parameters based on measurements from the quantum circuit. [16] [34]

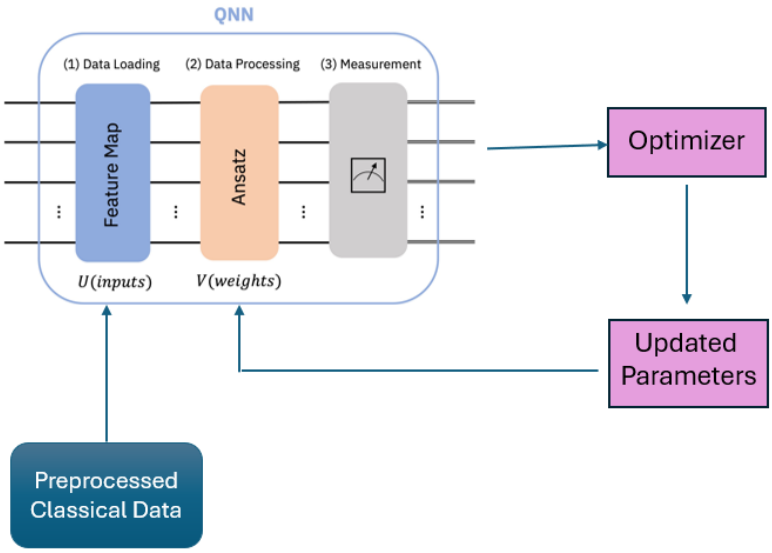


Figure 3.5: Overview of Quantum Neural Network Architecture.

Measurement

The final step in a QNN is Measurement, where we extract information from the quantum system. Measurement collapses the quantum state into a classical value, providing the final output of the QNN. This output is typically a probability distribution over different classes or a set of values corresponding to regression targets in supervised learning tasks. [6]
Measurement in Quantum Neural Networks Measurements are central in Quantum Neural Networks (QNNs), as they determine how quantum information is extracted and used to solve classical tasks. Different outputs can be utilized in the context of QNNs, each offering unique advantages depending on the problem. Below, we explore the three main types of measurements commonly used in QNNs:

- Measurement of Observables
  One of the most common types of quantum measurements involves measuring an observable, such as the Pauli-Z operator. This Measurement returns values in the $[1, 1]$ range, making it a natural choice for binary classification tasks. By interpreting the result of the Pauli-Z measurement, a simple threshold-based decision can be made: if the outcome is positive, classify it as one class; if negative, classify it as the other. For example, in Pennylane, this can be achieved using the $qml.expval(qml.PauliZ(wires = i))$ function, where i specifies the measured qubit.

  This method is particularly advantageous for problems requiring quick and efficient binary decisions. The output directly corresponds to the expected value of the observable for the quantum state prepared, and its simplicity aligns well with optimization algorithms for training.

- Probabilities of Quantum States
  Another useful type of output involves measuring the probabilities of each of the 2^n possible quantum states, where n is the number of qubits. These probabilities provide a richer representation of the quantum system and are useful for multiclass classification tasks or problems requiring more detailed information about the quantum state. In Pennylane, this

can be achieved using the $qml.probs(wires = range(n))$ function returns a probability distribution over all possible states.

For example, in a QNN designed for a multiclass classification task, the probabilities can be interpreted as confidence levels for each class, allowing the network to output nuanced predictions beyond simple binary decisions.

- Accessing the Quantum State Directly
  In some cases, obtaining the full quantum state without performing a collapsing measurement is useful. This can be done in Pennylane using the $qml.State()$ function. This output provides the amplitudes of the current quantum state on a computational basis. While this method is not a physical measurement (since it does not project the state onto a classical outcome), it is invaluable for debugging, analyzing quantum algorithms, or feeding quantum states into other quantum circuits.

  However, using $qml.state()$ is primarily restricted to simulations on classical hardware. Accessing the entire quantum state directly is generally infeasible on actual quantum devices due to the exponential growth in amplitudes with the number of qubits.

After Measurement, the results are used to calculate the cost function, quantifying the difference between the predicted and actual values. The estimated cost is then fed back into the classical optimizer to adjust the parameters θ in the variational circuit, iterating until convergence. [20] [42]
Through this combination of quantum data embedding, parametrized circuits, and Measurement, QNNs offer a promising approach to leveraging quantum resources for tasks in complex data analysis. Future research will explore these architectures in more depth, especially in applications like quantum simulation, where QNNs have the potential to provide insights into physical systems that are challenging to simulate classically. Furthermore, these concepts about QNN can be extended to build complex quantum models like Quantum Convolutional Neural Networks and be used to classify classical data. [29] [38] [16]

Chapter 4

High Energy Physics

In the past, atoms were believed to be the smallest indivisible units of matter. This perspective began to change with the discovery of subatomic particles in the 20th century. J.J. Thomson's identification of the electron in 1897 marked the first insight into the structure of atoms, revealing that they were not indivisible. Later, Ernest Rutherford's experiments in 1911 showed that atoms consist of a dense nucleus surrounded by electrons, and James Chadwick's discovery of the neutron in 1932 completed the picture of the atomic nucleus.

As experimental techniques advanced, it became clear that protons and neutrons, once considered fundamental, are composed of even smaller particles called quarks. In the 1960s, Murray Gell-Mann and George Zweig independently proposed the quark model, a groundbreaking idea that explained the properties of protons, neutrons, and other hadrons. This theoretical framework was later confirmed experimentally in 1968 through deep inelastic scattering experiments at the Stanford Linear Accelerator Center (SLAC), solidifying the quark model as a cornerstone of particle physics.

The study of fundamental forces also saw significant advancements. The strong nuclear force, which binds quarks together within protons and neutrons, was found to be mediated by particles called gluons. This discovery, alongside the development of quantum chromodynamics (QCD) in the 1970s, provided a deeper understanding of how quarks interact. Meanwhile, the weak nuclear force, responsible for phenomena like beta decay, was unified with the electromagnetic force in the electroweak theory developed by Sheldon Glashow, Abdus Salam, and Steven Weinberg. This theory predicted the existence of the W and Z bosons, mediators of the weak force, which were experimentally observed at CERN in 1983 under the leadership of Carlo Rubbia and Simon van der

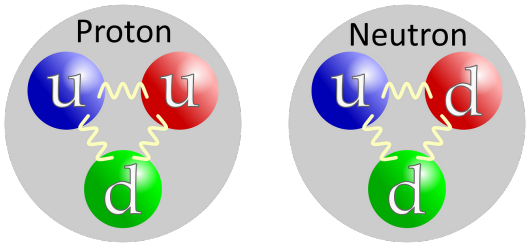


Figure 4.1: Composition of a proton and a neutron, illustrating their quark structure. The proton consists of two up quarks and one down quark (uud), while the neutron consists of one up quark and two down quarks (udd), bound together by gluons.

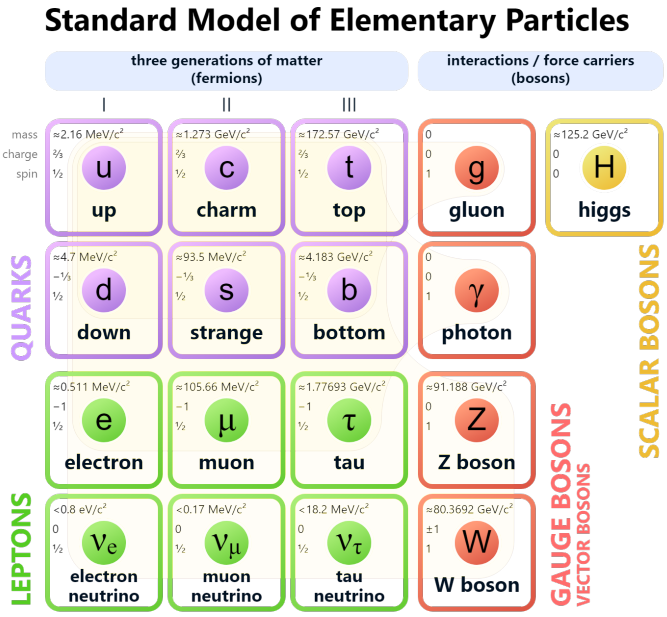


Figure 4.2: Standard Model of Elementary Particles.

Meer. These discoveries validated the electroweak theory and represented a monumental achievement in the Standard Model of particle physics.

Perhaps the most celebrated milestone in recent particle physics was the discovery of the Higgs boson in 2012 by the ATLAS and CMS experiments at the Large Hadron Collider (LHC). This particle, first theorized by Peter Higgs, François Englert, and others in 1964, is the key to understanding how particles acquire mass through interaction with the Higgs field. Its discovery confirmed one of the last missing pieces of the Standard Model and provided profound insights into the universe's structure.

The Standard Model organizes the fundamental particles into two categories: fermions and bosons. Fermions, which make up all matter, are divided into two families: quarks and leptons. Quarks come in six flavors—up (up), down (d), charm (c), strange (s), top (t), and bottom (b) —and combine to form protons, neutrons, and other hadrons. Leptons include the electron (e), muon (μ), tau (tau), and their associated neutrinos (ν_e, ν_μ, ν_τ). Bosons, on the other hand, mediate the fundamental forces. The photon (γ) mediates the electromagnetic force, the gluon (g) mediates the strong nuclear force, and the W and Z bosons mediate the weak nuclear force. Finally, the Higgs boson (H) provides mass to particles via the Higgs field. Each of these particles plays a specific role in governing the interactions and structure of the universe.

While the Standard Model has proven remarkably successful in describing the known particles and their interactions, it is incomplete. It does not incorporate gravity, explain the nature of dark matter, or address the imbalance between matter and antimatter. To tackle these challenges, experiments like those conducted at the LHC continue to investigate the limits of the Standard Model and search for new physics beyond its framework.

4.1 The Large Hadron Collider (LHC)

The Large Hadron Collider (LHC) is the world's most powerful particle accelerator at CERN near Geneva, Switzerland. It consists of a 26.7 km circular tunnel where protons and heavier hadrons are accelerated in opposite directions before colliding at designated interaction points. With a *center-of-mass energy* of up to $\sqrt{s} = 13.6\,\text{TeV}$, the LHC allows scientists to probe fundamental physics at unprecedented energy scales. These collisions enable the study of phenomena predicted by the Standard Model (SM) and searching for new physics, such as supersymmetry or dark matter candidates.

The LHC hosts four main experiments, each equipped with sophisticated detectors designed for specific physics goals: ALICE, ATLAS, CMS, and LHCb. ALICE (A Large Ion Collider Experiment) studies quark-gluon plasma, a state of matter present shortly after the Big Bang. ATLAS (A Toroidal LHC Apparatus) and CMS (Compact Muon Solenoid) are general-purpose detectors tasked with exploring a wide range of phenomena, including the search for new particles, tests of the Standard Model, and studies of dark matter candidates. LHCb (Large Hadron Collider beauty) specializes in precision measurements of CP violation and the properties of heavy quarks. In this work, we will focus on the CMS detector. This versatile and high-performance experiment has played a central role in key discoveries, such as the Higgs boson, and continues to probe the fundamental nature of the universe. [14]

At the Large Hadron Collider (LHC), protons are accelerated to speeds close to the speed of light, where relativistic effects become critical to understanding their behavior. In relativistic physics, the total energy of a particle is given by $E^2 = (pc)^2 + (mc^2)^2$, where p is the momentum, m is the rest mass, and c is the speed of light. For particles like protons in the LHC, their momentum dominates over their rest mass due to their extreme velocities, meaning their total energy is primarily determined by pc.
The "center-of-mass energy," 13.6 TeV at the LHC, refers to the total energy available for particle interactions when measured in a frame where the two colliding protons have equal and opposite momenta. Powerful magnetic fields accelerate protons in opposite directions in the LHC's 26.7 km ring to achieve this. Each proton individually reaches an energy of about 6.8 TeV, but in the center-of-mass frame, these energies add due to the relativistic relationship between energy, momentum, and mass.

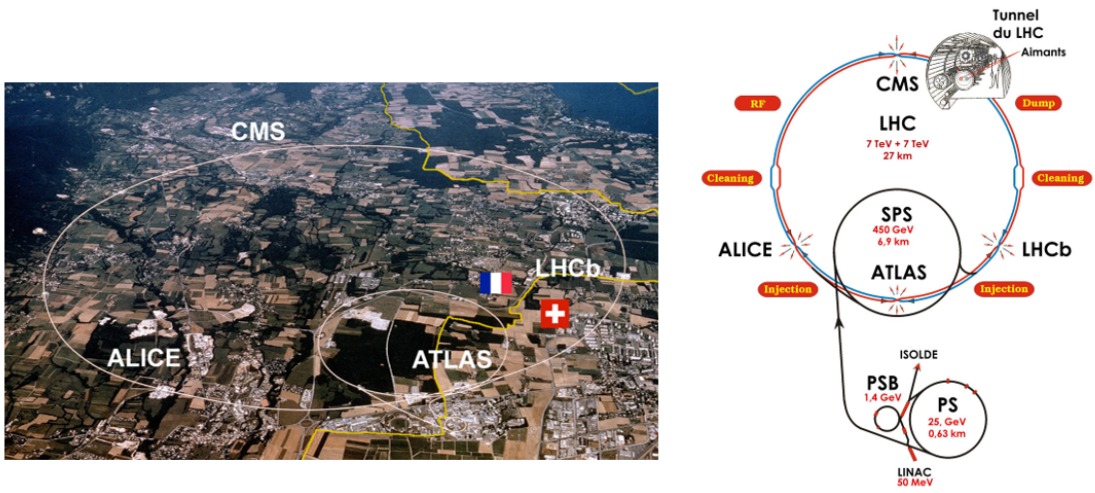


Figure 4.3: Map of the LHC Showing the Four Main Detectors: ATLAS, CMS, ALICE, and LHCb.

This extraordinarily high energy is crucial for probing the smallest scales of nature. It allows the production of massive particles, like the Higgs boson, and the exploration of interactions at energy scales far beyond what occurs naturally. By recreating these extreme conditions, the LHC provides a unique opportunity to test the fundamental laws of physics and search for phenomena beyond the Standard Model.

4.2 The Compact Muon Solenoid (CMS) Detector

The Compact Muon Solenoid (CMS) is a general-purpose detector designed to study the products of proton-proton collisions at the LHC. It has a cylindrical geometry with a total length of 21.6 m and a diameter of 15 m. The CMS detector consists of several sub-detectors arranged in concentric layers around the interaction point, each serving a specific purpose in particle identification and energy measurement. [13]
The CMS's layered structure enables the precise detection and characterization of various particle types through their unique interactions with the sub-detectors.
Once the collision occurs at the center of the detector, the process begins in the **tracker**, the innermost subdetector. Charged particles produced in the collisions traverse this region under a strong magnetic field, which bends their trajectories. By recording these curved paths with high precision, the tracker determines the momentum and charge of the particles.
Next, particles reach the **electromagnetic calorimeter (ECAL)**, designed to absorb the energy of photons and electrons completely. When these particles interact with the dense, scintillating material of the ECAL, they deposit all their energy, allowing precise measurement of their energy and spatial position.
Hadronic particles, such as protons, neutrons, and pions, pass through the ECAL but lose only part of their energy. These particles continue into the **hadronic calorimeter (HCAL)**, interacting with the dense absorber material. Through a cascade of nuclear and electromagnetic interactions, hadrons deposit most of their remaining energy in the HCAL. By the time they leave the HCAL, most hadrons have decayed or are fully absorbed. [27]
Beyond the HCAL lies the **superconducting magnet**, which generates the powerful magnetic field essential for bending the trajectories of charged particles in the tracker and identifying their momenta. The magnet also confines the magnetic flux, ensuring precision in particle tracking.
The only charged particles that survive beyond the magnet and calorimeters are muons. Due to their relatively high mass and energy, muons are minimally affected by the materials in the tracker,

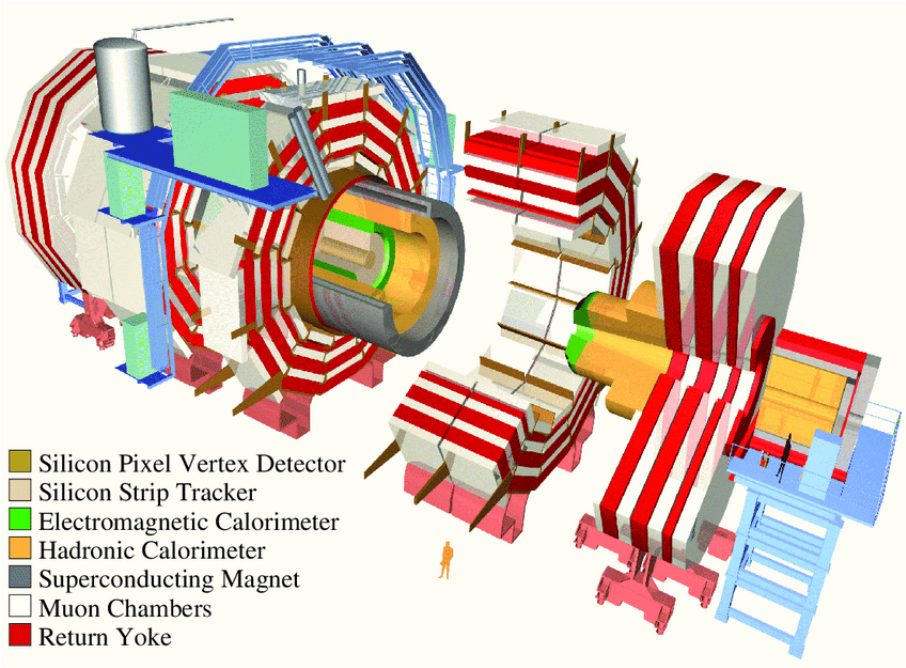


Figure 4.4: Structure of the Compact Muon Solenoid Detector.

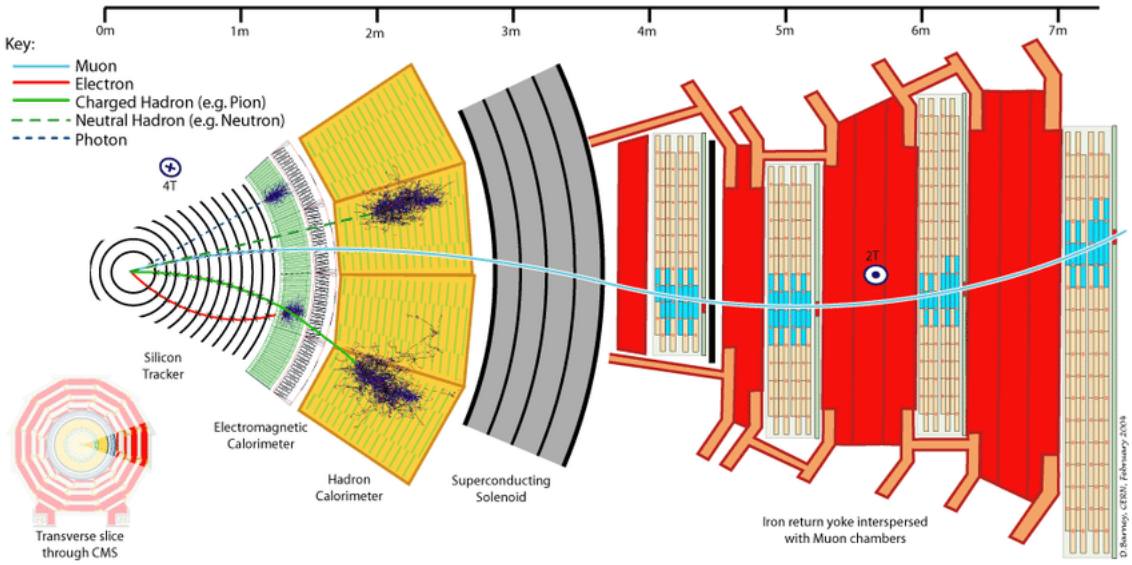


Figure 4.5: Particle Identification in the Compact Muon Solenoid Detector.

ECAL, and HCAL. These particles penetrate the entire detector and are subsequently identified in the **muon system**, the outermost layer of CMS. This system consists of layers of muon detectors interleaved with thick slabs of iron, known as the **return yoke**, which completes the solenoid's magnetic circuit and provides structural support. The muon detectors, combined with the return yoke, allow for precise measurement of the trajectory and momentum of muons.
Finally, neutrinos produced in the collisions escape detection entirely. These particles interact so weakly with matter that they pass through all the detector layers without a trace. Their presence is inferred indirectly by measuring the imbalance of momentum and energy in the detector, reported as missing transverse energy (E_T^{miss}).
Each particle's identification relies on its unique interaction pattern as it traverses the CMS detector. For instance, electrons leave a curved trajectory in the tracker due to their charge and deposit all their energy in the ECAL, producing electromagnetic showers. Electrically neutral photons bypass the tracker and are identified exclusively in the ECAL, where they also produce electromagnetic showers. Hadrons, such as pions or protons, interact with the tracker if charged, depositing energy partially in the ECAL and predominantly in the HCAL, generating hadronic showers. As massive and weakly interacting, Muons leave tracks in the tracker, pass through the ECAL and HCAL with minimal energy loss, and are detected in the muon system. Neutrinos, with no charge and extremely weak interactions, do not interact directly with any part of the detector and are identified indirectly through missing energy and momentum measurements. [17]
This ability to differentiate particles based on their interaction signatures in the subdetectors is a cornerstone of the CMS experiment. It enables precise reconstruction and identification of the diverse particles produced in high-energy collisions.

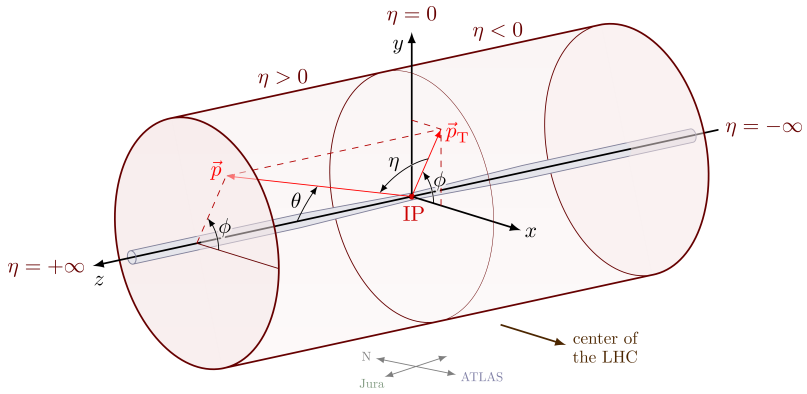


Figure 4.6: Coordinate System of the CMS Detector.

4.2.1 Coordinate System

The CMS coordinate system is defined as follows:

- The z-axis aligns with the beam direction.
- The x-axis points towards the center of the LHC ring.
- The y-axis is perpendicular to both, towards the surface, forming a right-handed Cartesian coordinate system.

Additionally, cylindrical coordinates are employed to describe particle trajectories. The azimuthal angle ϕ is measured in the transverse plane, while the polar angle θ determines the direction relative to the beam axis.
For a particle four-momentum $\vec{p} = (E, p_x, p_y, p_z)$, ϕ can be calculated as:

$$\tan(\phi) = \frac{p_y}{p_x} \longrightarrow \phi = \tan^{-1}\left(\frac{p_y}{p_x}\right)$$

where is limited to $-\pi \leq phi \leq \pi$. Another common quantity is transverse momentum, which is defined as:

$$p_t = \sqrt{p_x^2 + p_y^2}$$

This quantity is the component of momentum in the transverse plane X-Y. The transverse mass is another Lorentz invariant quantity defined as $m_t = \sqrt{m^2 + p_t^2}$.
Defining another variable in accelerator physics is useful instead of using the polar angle θ. This new variable is the rapidity, y, which is in terms of a particle momentum p:

$$y = \frac{1}{2}\left(\frac{E + p_z}{E - p_z}\right) \tag{4.1}$$

However, rapidity is hard to measure for particles with high energy $p >> m$. Therefore, another quantity must be introduced, which can be derived from the previous equation:

$$y = \frac{1}{2}\left(\frac{E + p_z}{E - p_z}\right) = \frac{1}{2}\ln\left(\frac{\sqrt{p^2 + m^2} + p\cos\theta}{\sqrt{p^2 + m^2} - p\cos\theta}\right) \tag{4.2}$$

$$\approx \frac{1}{2}\ln\left(\frac{p + p\cos\theta}{p - p\cos\theta}\right) = \ln\sqrt{\frac{1 + \cos\theta}{1 - \cos\theta}} = -\ln\tan\left(\frac{\theta}{2}\right) = \eta \tag{4.3}$$

Where θ is the angle between the particle's three momentum and the positive direction of the Z-axis, η is called the pseudorapidity of the particle.

4.2.2 CMS Subdetectors

Magnetic System

At the heart of the CMS detector lies a powerful superconducting magnet that generates a uniform magnetic field of 4 Tesla. This field bends the trajectories of charged particles, enabling precise measurement of their momentum. The magnetic system consists of a 13-meter-long superconducting coil with a 5.9-meter diameter, which confines the magnetic flux within the detector. Surrounding the coil is a return yoke of thick iron layers, which completes the magnetic circuit and provides structural support. The yoke is interleaved with muon detectors, allowing efficient tracking of muons as they penetrate the outermost layers of the detector.

Silicon Tracker

The silicon tracker is the innermost subdetector, designed to reconstruct the trajectories of charged particles with exceptional precision. It consists of two main components. The pixel vertex detector, located closest to the collision point, accurately measures particle trajectories. Composed of multiple layers of compact silicon pixel detectors, it excels in withstanding high particle flux and identifying the primary and secondary vertices of particle decays. Surrounding the pixel detector, the silicon strip tracker extends the tracking coverage and measures the momentum of charged particles. This subsystem, which employs long and narrow silicon strips to detect particle passage through ionization signals, is divided into the barrel region, which provides tracking for particles with low pseudorapidity ($|\eta| < 2.5$), and the endcap region, which extends coverage to particles at wider angles.

Electromagnetic Calorimeter (ECAL)

The ECAL measures the energy of electrons and photons by absorbing their energy and generating electromagnetic showers. Constructed with dense, radiation-resistant lead tungstate ($PbWO_4$) crystals, the ECAL ensures precise energy measurements. It consists of two sections: the barrel ECAL (EB), which covers the central region of the detector ($|\eta| < 1.479$) and is composed of thousands of crystals arranged cylindrically, and the endcap ECAL (EE), which extends coverage to higher pseudorapidities ($1.479 < |\eta| < 3.0$). The crystals in the endcap are slightly larger to optimize performance in the forward region, ensuring excellent resolution across the detector.

Hadronic Calorimeter (HCAL)

The HCAL is responsible for measuring the energy of hadrons and contributes to the detection of missing transverse energy (E_T^{miss}), which is often indicative of neutrinos or other non-interacting particles. It consists of several sections: the barrel HCAL (HB), which covers $|\eta| < 1.3$ and uses dense absorber materials like brass and scintillators to detect hadronic showers; the endcap HCAL (HE), which extends the coverage to pseudorapidities between 1.3 and 3.0; and the forward HCAL (HF), positioned in the very forward regions ($3.0 < |\eta| < 5.0$), designed to detect forward jets and improve E_T^{miss} measurements.

Muon System

The muon system, located in the outermost layer of the CMS detector, is specifically designed to identify and measure muons, which can penetrate the inner subdetectors. This system is interleaved with the return yoke and comprises three types of detectors. Drift tubes (DTs), located in the barrel region ($|\eta| < 1.3$), provide high-precision measurements of the position and timing of muons. Cathode strip chambers (CSCs), situated in the endcap regions ($1.3 < |\eta| < 2.4$), are designed to withstand high radiation levels and particle flux, offering spatial and timing information for forward muons. Resistive plate chambers (RPCs) are present in both the barrel and endcap regions, delivering fast timing signals critical for the CMS trigger system and complementing the information provided by DTs and CSCs to enhance muon identification efficiency.

Trigger System

The high collision rate at the LHC, reaching 40 MHz, makes it impossible to record all events. The CMS trigger system addresses this challenge by filtering data to select the most relevant events for storage and analysis. It operates in two stages. The Level-1 Trigger (L1), a hardware-based system, uses information from the calorimeters and muon detectors to make rapid initial event selections, reducing the event rate to 100 kHz with latencies of just a few microseconds. The High-Level Trigger (HLT), a software-based system, performs more refined event selections using complete detector information, reducing the event rate to approximately 1 kHz. This reduction enables storage and offline analysis while employing sophisticated algorithms to reconstruct particles and identify physics signatures such as Higgs boson decays or rare particle interactions.

4.2.3 Data Storage and Global Analysis

The filtered data are transferred to the Data Acquisition (DAQ) system and distributed to a global network of computing centers. These facilities conduct detailed analyses, including reconstructing particle trajectories and energies, identifying physics objects such as jets, leptons, and photons, and calculating observables like invariant masses and angular correlations. This distributed analysis ensures that the vast volumes of data generated by the CMS detector are efficiently processed and utilized for scientific discovery.

4.3 Theoretical and Experimental Cycles in Particle Physics

Particle physics advances through two complementary and interconnected approaches: from theory to experiment and from experiment to theory. These cycles are not isolated but form a feedback loop that drives progress in understanding the fundamental nature of the universe.

4.3.1 From Theory to Experiment

This approach begins with theoretical predictions derived from fundamental principles or extensions of existing frameworks, such as the Standard Model. The process involves multiple steps to test whether the predictions are consistent with experimental observations:

1. **Event Generation:** Theoretical models are translated into observable phenomena through Monte Carlo event generators, such as PYTHIA, MADGRAPH, or DELPHES. These programs simulate the particle interactions predicted by the theory, including the production of new particles, decay chains, and background processes. Event generators rely on parton distribution functions (PDFs) to describe the momentum distribution of the quarks and gluons within the protons involved in the collisions. [43][22]

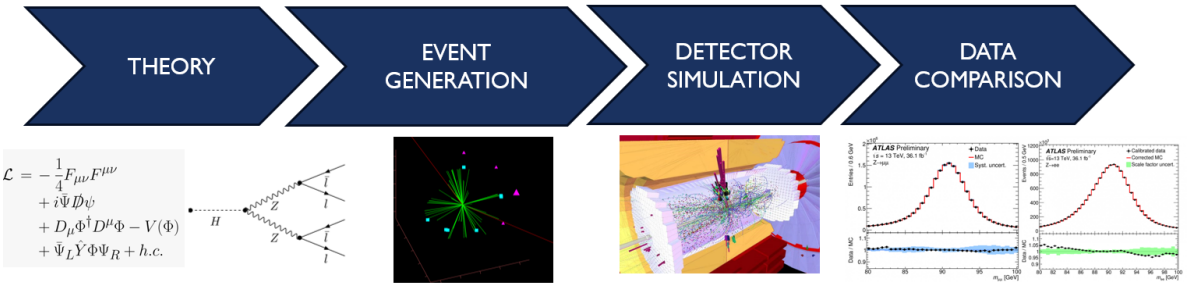


Figure 4.7: Theoretical to Experimental Workflow in Particle Physics.

2. **Detector Simulation:** The generated events are processed through detailed detector response simulations using tools like GEANT4. These simulations model how particles interact with the various subdetectors of CMS, accounting for effects such as energy deposition, track reconstruction, and detector noise. This step produces simulated data in a format directly comparable to experimental data.[1]

3. **Data Comparison:** Simulated data are compared to actual experimental results. This comparison involves applying the same reconstruction algorithms and analysis techniques to both datasets, ensuring a consistent methodology. Discrepancies between the two can indicate the need for refinement in the theoretical model or the simulation parameters.

The success of this approach hinges on the accuracy of both the theoretical models and the detector simulations. The theory gains credibility if the predictions match the data within statistical and systematic uncertainties. If significant discrepancies arise, they can point to the need for adjustments in the model or signal the presence of new physics.

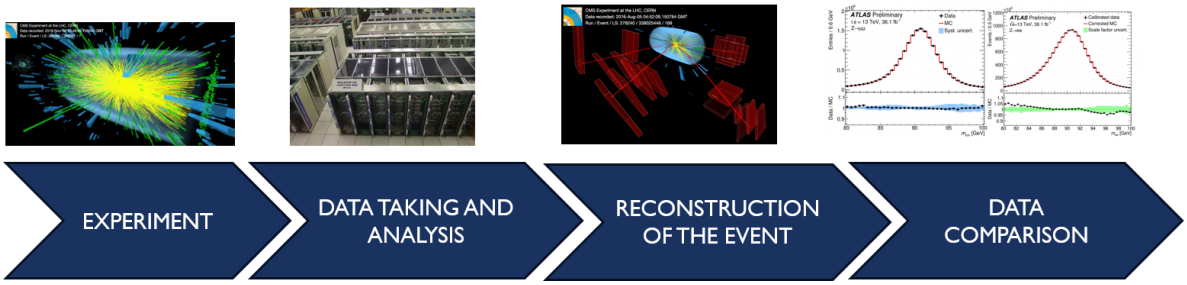


Figure 4.8: Experimental to Theoretical Workflow in Particle Physics.

4.3.2 From Experiment to Theory

In this reverse cycle, experimental data take center stage, guiding the refinement or extension of theoretical models. The process unfolds as follows:

1. **Event Analysis:** Data collected from high-energy collisions are processed to identify specific events or processes of interest. Advanced reconstruction algorithms identify particles such as electrons, muons, jets, and missing transverse energy. The focus is often on rare events or anomalies that could indicate new phenomena.

2. **Theoretical Refinement:** Once specific processes are identified, theorists use the experimental results to refine existing models or propose new ones. This refinement can involve adjusting parameters within the Standard Model, such as coupling constants or mass values, or formulating entirely new theories, such as those involving supersymmetry or extra dimensions.

3. **Validation:** The refined or new theories are validated by generating simulated data and comparing them with experimental results. The validation process requires the simulated data to reproduce the experimental observations with high confidence. The benchmark for such agreement is often a statistical significance of less than 5σ, ensuring that the alignment is not due to random fluctuations.

This approach enables the discovery of patterns or anomalies in the data that cannot be explained by existing theories, providing clues to phenomena beyond the Standard Model. For instance, the discovery of the Higgs boson involved detecting its decay signatures in the data and the theoretical interpretation of these observations within the Higgs mechanism framework.

4.4 The Interplay Between Theory and Experiment

The advancement of particle physics relies on the dynamic interaction between theory and experiment. Theories provide a framework to interpret experimental data, while experiments challenge these frameworks and inspire new theoretical developments. For instance, the prediction and discovery of the Higgs boson exemplify decades of collaboration between theory and experimentation. Similarly, searches for dark matter candidates, like WIMPs, and anomalies, such as the muon's magnetic moment or lepton universality discrepancies, demonstrate how experimental results drive theoretical innovation. This iterative cycle ensures continuous refinement, expanding our understanding of the fundamental laws of nature.

4.5 Some Applications of ML to HEP

Machine learning (ML) has become an indispensable tool in high-energy physics (HEP), addressing the unique challenges posed by massive datasets, complex detectors, and intricate physics processes. Below, we outline some key areas where ML is transforming HEP, with examples and future perspectives. [33] [12] [24]

4.5.1 Signal vs Background Classification

One of the most common applications of ML in HEP is distinguishing rare signal events from overwhelming background noise. [30] [28]
In particle physics, a signal refers to events corresponding to a specific physical process or phenomenon of interest, such as the production of a particular particle or the manifestation of a hypothesized interaction. These events are characterized by unique properties or observables that distinguish them from other processes. Conversely, the background consists of events arising from other physical processes, often well-described by the Standard Model, which mimic or obscure the signal due to similar signatures in the detector.
Separating the signal from the background is a critical task in data analysis. It requires sophisticated statistical and computational techniques to maximize signal sensitivity while minimizing background contamination. This distinction enables precise measurements and reliable discoveries in high-energy physics experiments.

In collider experiments, processes like the production of Higgs bosons or potential new particles occur amid a sea of more mundane interactions. ML models classify events based on reconstructed observables, such as particle momentum, pseudorapidity, and energy. For instance, Machine learning algorithms, such as decision trees and neural networks, are trained using simulated data to distinguish subtle patterns in the data that separate signal events from the more abundant background. These methods have been instrumental in enhancing the efficiency of event selection and enabling precision measurements in collider experiments, paving the way for breakthroughs in understanding fundamental physics.

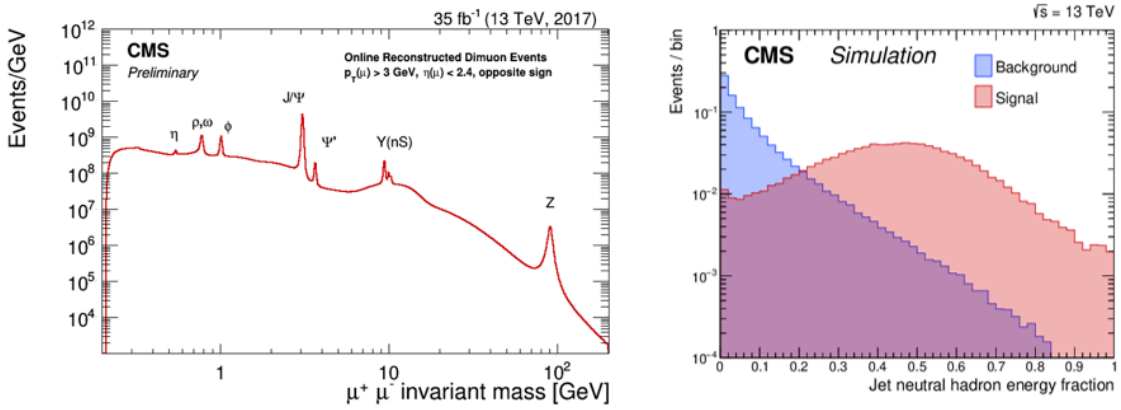


Figure 4.9: Example of Signal vs Background Classification.

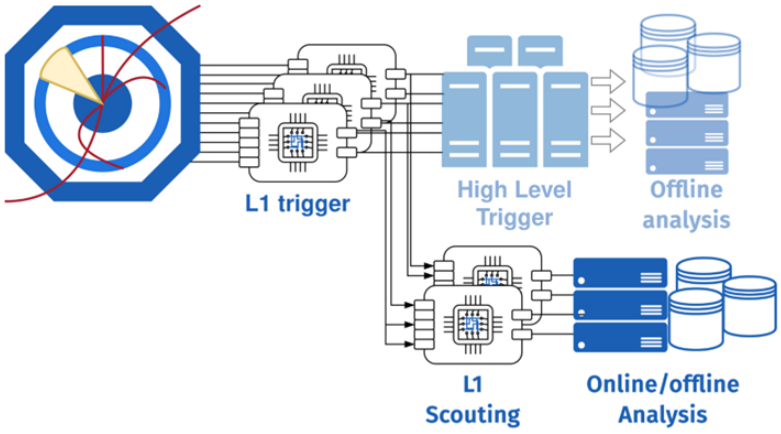


Figure 4.10: Trigger system in the CMS detector.

4.5.2 Particle Reconstruction

Reconstructing and identifying particles from raw detector data is a cornerstone of high-energy physics (HEP) experiments. Machine learning (ML) algorithms are pivotal in identifying particle types, such as electrons, muons, and jets, based on their energy deposition patterns and interaction signatures across the tracker, calorimeters, and muon systems. They also reconstruct particle trajectories and momenta with high precision, often replacing traditional pattern recognition algorithms with advanced approaches like convolutional or graph neural networks. Additionally, ML techniques are instrumental in separating overlapping particle signals in dense environments, such as those produced in high-luminosity LHC collisions. For instance, graph neural networks have shown significant promise in tracking systems by modeling particle trajectories as connections between detector hits.

4.5.3 Trigger Optimization

The trigger system of detectors like CMS and ATLAS must process collision data at up to 40 MHz, selecting only a fraction of events for further analysis. Machine learning significantly enhances trigger efficiency and flexibility by enabling real-time decision-making with lightweight neural networks that select events containing high-energy leptons, jets, or missing transverse energy. ML models also optimize resource allocation by reducing data throughput without compromising sensitivity to rare or exotic events. Advanced techniques like reinforcement learning are being explored to dynamically adapt trigger thresholds based on real-time detector conditions and beam luminosity variations, further improving performance. Future developments in this area include integrating ML directly into hardware, allowing for faster event filtering and implementing more sophisticated models, ensuring scalability as collision rates increase with future LHC upgrades.

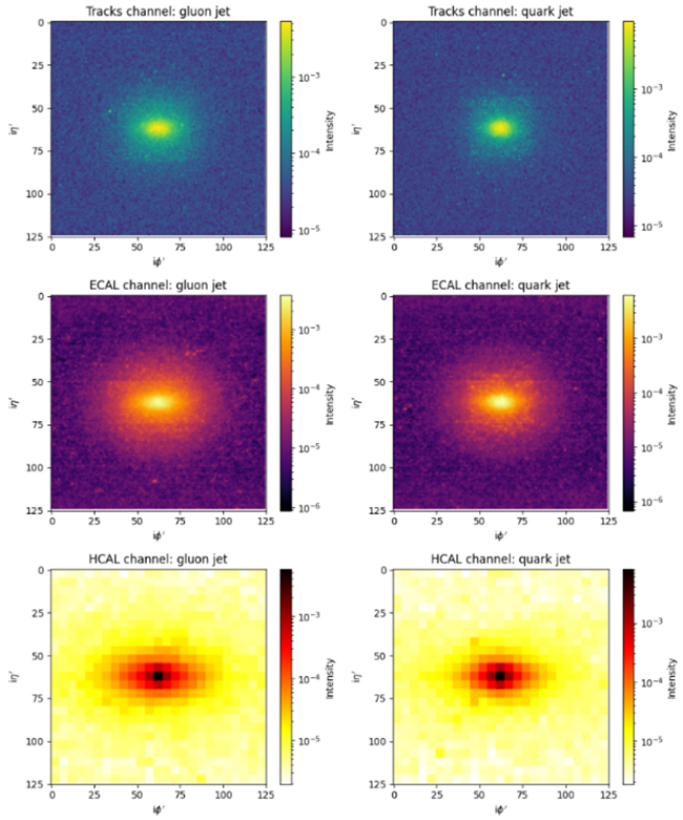


Figure 4.11: Examples of Heatmaps from Calorimeters in the CMS Detector.

4.5.4 Image Analysis: Calorimeter Heatmaps

Calorimeters in HEP detectors produce spatially resolved energy deposition maps, often resembling images. Machine learning models, particularly convolutional neural networks (CNNs), excel in interpreting these heatmaps. CNNs are widely used to identify particle showers, distinguishing electromagnetic showers, such as those produced by photons or electrons, from hadronic showers generated by jets or pions. These models also directly reconstruct high-level features, such as the total energy, shower shape, and event origin, from raw calorimeter data. Furthermore, heatmap analysis aids anomaly detection, identifying patterns that may indicate new physics. This approach leverages advancements in image recognition from computer vision, adapting them to meet the unique challenges of HEP.

4.5.5 Data Generation

Simulating particle collisions and their detector responses is computationally expensive, often requiring large-scale computing resources. Machine learning accelerates this process through generative models like Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs), which produce realistic simulated events at speeds far exceeding traditional Monte Carlo methods. Hybrid approaches that combine physics-based simulations with ML-based fast approximations maintain accuracy while significantly reducing computational costs. These innovations enable more efficient use of computing resources and facilitate rapid prototyping for new detector designs or experimental configurations.

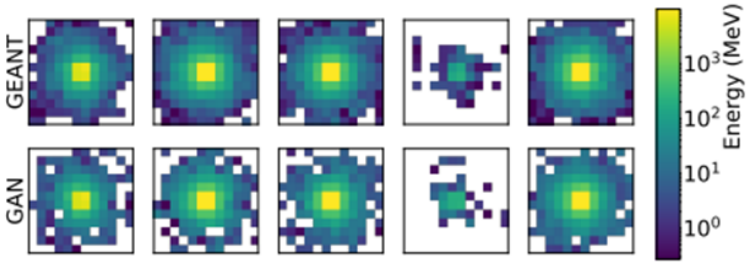


Figure 4.12: Example of a Generative Adversarial Network for Simulating GEANT4-like Data in Particle Physics.

4.5.6 Detector Optimization and Data Quality Monitoring

Ensuring optimal detector performance and collecting high-quality data are critical for successful experiments. Machine learning assists in predicting and correcting detector misalignments, noise, or malfunctioning components by analyzing sensor data in real time. Anomalies in data streams can be identified promptly, allowing issues to be flagged and addressed quickly, thereby improving data-taking efficiency. ML techniques are also used to simulate and optimize detector designs, employing reinforcement learning or surrogate modeling. For example, anomaly detection algorithms monitor calorimeter signals to detect radiation damage or electronics failures during operation, ensuring the continued reliability of detector systems.

4.5.7 Future Directions: Physics-Informed Neural Networks

Integrating physical principles into ML models represents an exciting frontier in HEP. Physics-informed neural networks (PINNs) embed conservation laws, symmetries, or other theoretical constraints directly into their architectures or loss functions. By incorporating principles such as energy and momentum conservation, Lorentz invariance, or unitarity, PINNs ensure that ML models adhere to fundamental physical laws. These models enhance interpretability and reduce the need for extensive training datasets, as prior knowledge narrows the solution space. Applications of PINNs include improving particle track reconstruction, enabling more accurate detector simulations, and guiding model-agnostic searches for new physics. As HEP experiments are complex, PINNs offer a promising approach to combining data-driven methods with deep theoretical insights.

4.6 Potential Applications of QML to HEP

Quantum Machine Learning (QML) has the potential to revolutionize high-energy physics (HEP) by addressing complex challenges in experiments such as CMS and ATLAS at the LHC. Traditional machine learning (ML) has proven indispensable in tasks like signal vs. background classification, particle reconstruction, and detector optimization. QML builds upon these foundations, offering the prospect of leveraging quantum computers to tackle similar problems with potential advantages in speed, scalability, and handling complex quantum data structures. [23] [32]
In HEP, QML could enhance several key areas already benefiting from classical ML: Quantum models like Quantum Convolutional Neural Networks (QCNNs) have been employed to classify signal and background events, harnessing quantum encoding and the parallelism of quantum computation for efficient feature extraction and classification. [**QML-LHCb**] [15]

Quantum Generative Adversarial Networks (QGANs) represent a groundbreaking method for simulating particle collision datasets, offering a faster alternative to Monte Carlo methods while preserving accuracy. [35] [10] [45] Additionally, QML algorithms hold promise for real-time monitoring of detector conditions, analyzing the high-dimensional data from tracking systems and calorimeters. Quantum models could also improve trigger systems by optimizing event selection criteria to efficiently identify rare signals amidst the overwhelming data produced by LHC collisions.

This thesis focuses on applying QGANs for signal and background classification using simulated datasets generated with Delphes, a widely utilized framework for fast detector simulations. The research has two primary objectives: First, leverage QGANs to generate realistic signal and background events that complement traditional Monte Carlo simulations. This method aims to accelerate the creation of high-dimensional datasets while maintaining fidelity to experimental data. Second, to employ quantum classifiers, such as Quantum Convolutional Neural Networks, to analyze Delphes-simulated data and achieve efficient and accurate discrimination between signal and background events.
By exploring these applications, this work contributes to the growing integration of QML into HEP. It highlights its potential to address the computational challenges of the High-Luminance LHC era and beyond.

CERN Quantum Technology Initiative (CERN QTI)

The CERN QTI leads efforts to integrate quantum technologies into high-energy physics research, establishing a comprehensive roadmap and research program in collaboration with its advisory board, the high-energy physics community, and the quantum-technology research community. This initiative seeks to foster joint research efforts that explore the synergy between quantum technologies and particle physics. It also provides educational and training opportunities to prepare the next generation of researchers in quantum computing and its applications to HEP. In addition, CERN QTI aims to establish the necessary quantum computing infrastructure to support this emerging field, facilitating innovation and knowledge exchange through dedicated mechanisms for collaboration.
As part of this broader effort, applying quantum machine learning (QML) to HEP can potentially drive groundbreaking discoveries. By addressing the computational challenges posed by the increasing complexity of modern experiments, QML offers novel tools to expand the frontiers of particle physics. This thesis contributes to this vision by demonstrating the utility of Quantum Generative Adversarial Networks (QGANs) for simulating and classifying Delphes-generated signal and background events, exemplifying how quantum technologies can transform high-energy physics research. [32]

Chapter 5

Development of the QGAN Model

With the imminent demand for extensive computing resources in the HL-LHC program, quantum computing's potential as a solution has garnered attention, particularly in particle physics contexts like the Large Hadron Collider (LHC). This chapter outlines the design and implementation of a Quantum Generative Adversarial Network (QGAN) tailored for high-energy physics (HEP) applications, specifically the classification and generation of signal and background jet events simulated with Delphes.

5.1 Introduction to QGANs in HEP

Quantum Generative Adversarial Networks (QGANs) are hybrid quantum-classical models that combine the representational power of quantum circuits with classical optimization techniques. They are particularly well-suited for high-energy physics applications, where the datasets are high-dimensional and the computational requirements are immense. By leveraging quantum computing's inherent parallelism, QGANs aim to:

- Distinguish between signal and background events.
- Generate data that closely resembles real data.

This study demonstrates the feasibility of applying QGANs to HEP problems, employing Google Cirq and TensorFlow Quantum for implementation.
The code used in this project can be found in the GitHub repository. [19]

5.2 Dataset and Preprocessing

The dataset consists of simulated events generated using Delphes, a fast detector simulation framework. It includes:

- 100 training and 100 testing samples, labeled as signal (1) and background (0).
- Features encoded using quantum rotation gates to map the input data onto a quantum state.

5.3 QGAN Architecture

The general architecture of the QGAN model comprises a quantum generator and a quantum discriminator designed to work cohesively within a quantum framework. Real classical data is transformed into quantum data through angle-based embedding, where each feature is mapped to

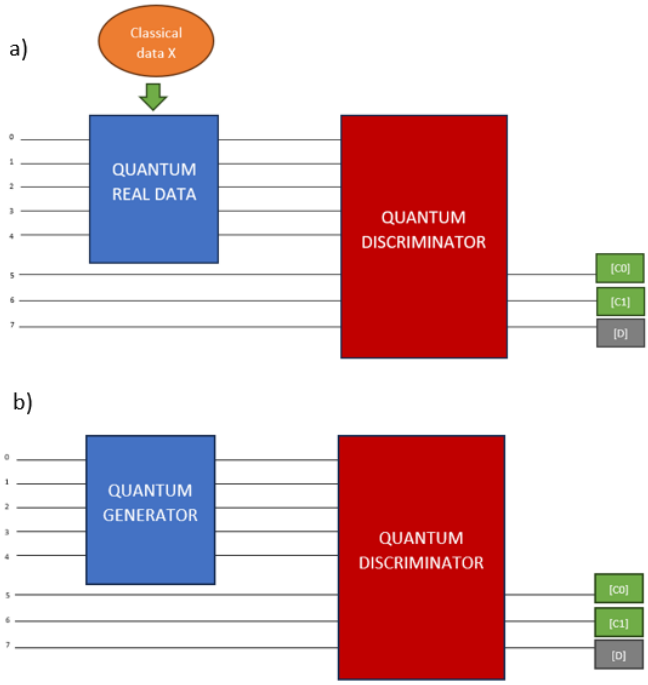


Figure 5.1: Architecture of the QGAN model: (a) Quantum discriminator classifying real quantum data encoded from classical inputs. (b) Quantum generator creates synthetic quantum data, which the discriminator evaluates to distinguish between real and generated data.

the argument of a rotational gate. This method encodes classical information into quantum states and facilitates a direct connection between the quantum generator and discriminator.
While this work focuses on low-dimensional data, the approach is inherently scalable. Leveraging the exponential encoding capacity of quantum systems allows for representing 2^n features using only n qubits. Subsequent sections will delve into the specifics of these quantum models, detailing their architecture, features, and the training process employed to achieve convergence.

5.3.1 Quantum Generator

The quantum generator is implemented as a Parameterized Quantum Circuit (PQC) operating on five qubits. Each qubit performs the following operations:

- Initial Random Rotations: RY gates apply random rotations, with angles sampled from a normal distribution of mean zero and variance $\frac{\pi}{3}$.
- Trainable Rotations: RX, RY, and RZ gates perform parameterized rotations, which are optimized during training.
- Entanglement: CNOT gates introduce entanglement between the qubits, ensuring a non-trivial quantum state.

These operations can be repeated over multiple layers to enhance the quantum generator's expressiveness. At the final stage, a one-qubit unitary operation is applied to each qubit, completing the circuit.
The primary objective during training is to optimize the quantum generator to produce synthetic quantum states that are indistinguishable from real quantum states by the quantum discriminator.

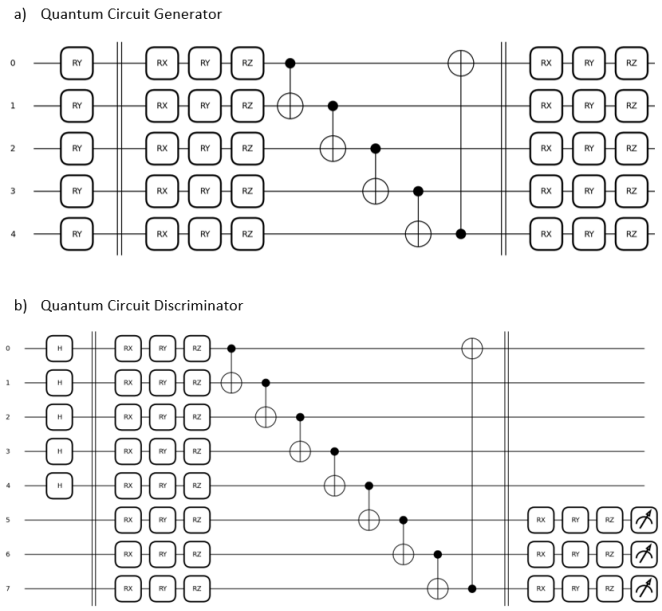


Figure 5.2: Quantum circuits used in the QGAN model: (a) Quantum generator circuit with parameterized rotations and entanglement gates to create synthetic data. (b) A quantum discriminator circuit with a similar structure includes measurements for classification.

Notably, no measurement is performed within the generator itself, as its purpose is to create quantum states directly analogous to the target quantum states.

5.3.2 Quantum Discriminator

The Quantum Discriminator is composed of 8 qubits. The first five qubits correspond to the quantum data originating from the generator or real quantum data. The remaining three qubits serve as the output for predicting classes. Among these, two qubits are dedicated to classifying signal and background events, represented as $[C_0, C_1]$:

- $[1, -1]$ indicates label 0 (background)
- $[-1, 1]$ indicates label 1 (signal)

Meanwhile, the third qubit is employed to classify real and fake data $[D]$:

- $[-1]$ signifies fake data
- $[1]$ signifies real data

The Quantum Discriminator consists of Hadamard and unitary gates with trainable parameters across L layers. Ultimately, measurements are taken from the last three qubits to inform the classification outcome.

5.4 Loss Functions

Training is crucial in this model as it requires a delicate balance between the learning of the generator and the discriminator. For this quantum approach and its two classification tasks, we need to divide the training into the following steps:

1. Train the generator-discriminator system.
2. Train the discriminator with real and fake data to classify events.

In step 1, similar to the original Generative Adversarial Networks, we first train one model and freeze the weights of the other. This principle remains valid for this quantum version.
The loss function we aim to minimize combines two classification tasks. Our model should distinguish between real and fake data and signal and background events.

5.4.1 Discriminator Loss

The discriminator loss function (L_D) is used for the real/fake data classification task and is defined as:

$$L_D = \frac{1}{m_{\text{fake}} + m_{\text{real}}} \left(\sum_{i=1}^{m_{\text{fake}}} \log\left(1 - D(x^i_{\text{fake}})\right) + \sum_{j=1}^{m_{\text{real}}} \log D(x^j_{\text{real}}) \right) \tag{5.1}$$

Where:

- m_{fake} and m_{real} are the number of samples in the fake (generated) data and real data sets, respectively.
- x^i_{fake} and x^j_{real} represent individual samples in the fake (generated) and real data sets, respectively.
- $D(x)$ is the output of the discriminator for a given input sample x. Specifically, $D(x^i_{\text{fake}})$ is the discriminator's output for the i-th fake data sample, and $D(x^j_{\text{real}})$ is its output for the j-th real data sample.
- $\log\left(1 - D(x^i_{\text{fake}})\right)$ and $\log D(x^j_{\text{real}})$ represent the logarithm of the discriminator's predicted probabilities for fake and real data samples, respectively. The logarithmic scaling is used to improve the stability and convergence of optimization.

This loss function aims to minimize the negative log probabilities the discriminator assigns to the fake data samples and maximize the log probabilities assigned to the real data samples, ensuring the discriminator distinguishes real and fake data effectively.

Classifier Loss Function
For the classification task of distinguishing signal and background events, we define a classification loss function L_C, aimed at minimizing classification errors. The Categorical Cross-Entropy Loss Function (CE) is used:

$$L_C = CE(y_{\text{true}}, y_{\text{predict}})$$

Where:

- y_{predict} is the prediction of the class made by the quantum discriminator. The class labels are represented as vectors: $[1, -1]$ for background and $[-1, 1]$ for signal.
- y_{true} is the ground truth class label for the event.

This loss function measures the discrepancy between the predicted and actual class labels, guiding the training to improve the discriminator's classification accuracy for signal and background events.

General Loss Function

Combining these two loss functions, we can assign a weight to each loss depending on the classification task's importance:

$$\mathcal{L} = \alpha_C L_C + (1 - \alpha_C) L_D \tag{5.2}$$

Where:

- α_C is a constant value that sets the weight of the loss function of the classifier for signal/background events.
- L_C is the loss function aiming to minimize signal/background data classification errors.
- L_D is the loss function that aims to minimize errors in discriminating real/fake data.

Note: The constant α_C lets us specify the importance of correctly classifying signal/background events. For example:

- If $\alpha_C = 0.0$, our loss function is only $\mathcal{L} = L_D$, meaning we are solely interested in discriminating real/fake data.
- If $\alpha_C = 0.25$, our loss function is $\mathcal{L} = 0.25 L_C + 0.75 L_D$, indicating consideration for both loss functions, with a higher priority on correctly classifying real/fake data.
- If $\alpha_C = 0.50$, both loss functions are given equal weight.
- If $\alpha_C = 1$, our loss function is only $\mathcal{L} = L_C$, indicating full focus on correctly classifying signal/background events.

With this in mind, first, we train our Quantum Discriminator model with $\alpha_C = 0.50$ to better classify real/fake data.

5.4.2 Generator Loss

The generator aims to maximize the probability of the discriminator misclassifying fake data as real:

$$L_G = -\frac{1}{m} \sum_{i=1}^{m} \log D(x^i_{fake}) \tag{5.3}$$

5.5 Training Procedure

The training involves two phases:

1. Training the generator-discriminator system with combined loss L_{CD}.
2. Training the discriminator independently for real/fake and signal/background classification.

The Adam optimizer is used for both components, considering a learning rate of $lr = 0.001$. Separate learning rates are used for stability. The batch size is 10 for the generator and 32 for the discriminator.

5.6 Results

The QGAN model effectively generates synthetic quantum data and classifies real quantum states as signal or background. This section provides a detailed analysis of the results, including insights from Q-sphere and Bloch sphere visualizations and metrics such as discriminator accuracy and the convergence of generator and discriminator loss curves for different hyperparameter values α.

5.6.1 Loss and Accuracy Analysis

Figure 3 presents the loss curves and discriminator accuracy for different values of α, which controls the priority of the classification task (signal vs. background) during training. The following observations can be made:

- $\alpha = 0.50$**:** The generator loss decreases steadily while the discriminator loss stabilizes, indicating the successful adaptation of the generator to the discriminator's feedback. The accuracy of the discriminator fluctuates around 75%, reflecting acceptable performance in distinguishing real and synthetic data.
- $\alpha = 0.75$**:** A similar trend is observed, with the discriminator maintaining stable accuracy around 82%. The loss curves show consistent behavior, with a slight reduction in discriminator dominance, suggesting improved collaboration between the two models.
- $\alpha = 1.00$**:** The generator loss is significantly reduced, while the discriminator loss remains relatively constant. The discriminator accuracy tends to be 90%, suggesting a more significant commitment to classifying signal and background jet events.

These results illustrate how α influences the balance between generator and discriminator training dynamics, enabling the model to be fine-tuned for various objectives, such as enhancing data realism or improving classification accuracy.

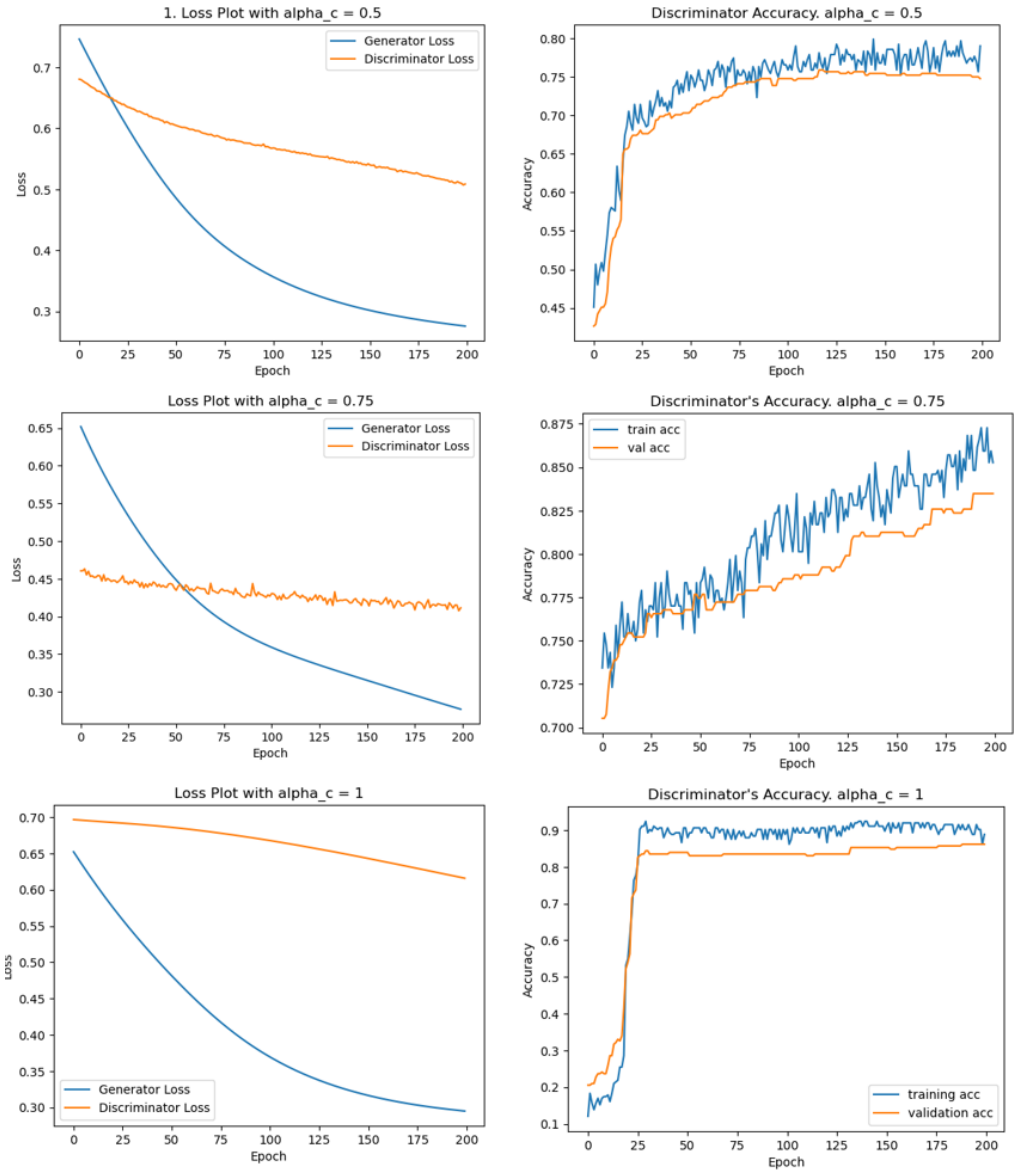


Figure 5.3: Loss and accuracy plots for generator and discriminator using different values of α.

5.6.2 Q-Sphere Visualization

The Q-sphere visualizations in Figure 5.4 highlight the fidelity of the generated quantum states compared to the real quantum data. The generated quantum states (left Q-sphere) exhibit a similar distribution of points to the real quantum states (right Q-sphere), with comparable amplitudes and phases. The color represents the phase of each computational basis state, while the amplitude is reflected in the size of the points on the Q-sphere. This close resemblance indicates that the quantum generator successfully captures the underlying structure of the real data.

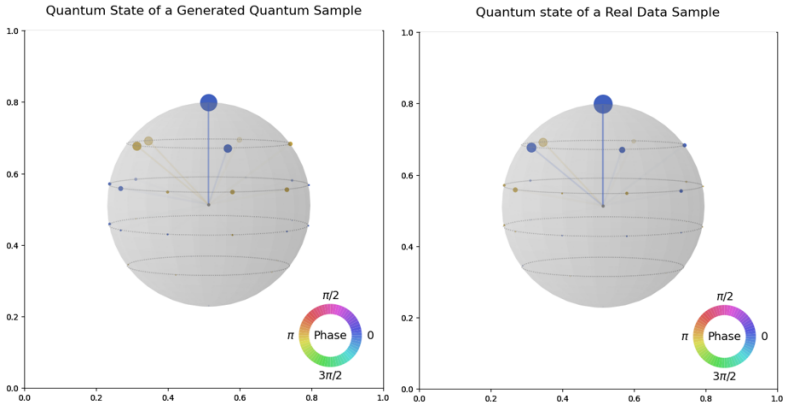


Figure 5.4: Generated and real quantum data comparison using Q spheres.

5.6.3 Bloch Sphere Analysis

Individual qubits are visualized using Bloch spheres in Figure 5.5 to further analyze the generated and real quantum states.
The Bloch spheres of the generated quantum states (Figure 5.5a) display diverse quantum states with coherence and entanglement patterns introduced by the generator. Comparing them to the real quantum states (Figure 5.5b) reveals similar state vector distributions, indicating that the QGAN model effectively learns the quantum data distribution.

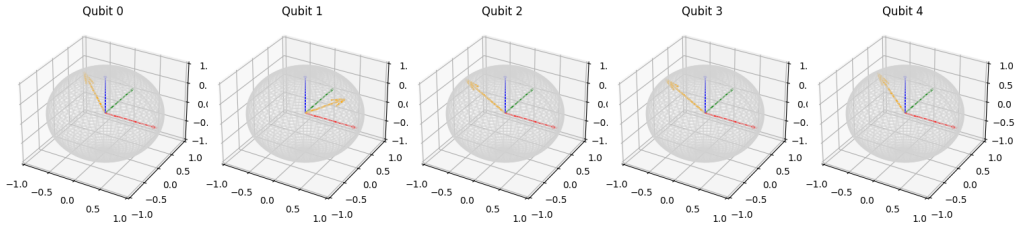
5.6.4 Discriminator Performance

The discriminator achieves an average accuracy of approximately 90%, as shown in the accuracy plots in Figure **??**. The consistent performance across different α values suggests that the discriminator is well-trained to differentiate between real and synthetic data, even as the generator becomes more proficient.

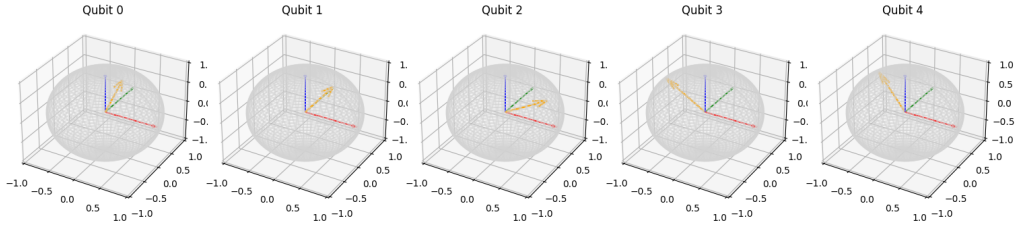
5.6.5 Conclusion of Results

The QGAN model demonstrates its capability to:

- Generate synthetic quantum data that closely resembles real quantum data, as evidenced by the Q-sphere and Bloch sphere comparisons.
- Maintain stable performance in classifying signal and background quantum states, with a discriminator accuracy of around 90%
- Balance the training dynamics between generator and discriminator, as shown by the convergence of loss curves and the impact of α on the training process.



(a) Bloch spheres for generated quantum states.



(b) Bloch spheres for real quantum states.

Figure 5.5: Comparison of Bloch spheres for generated and real quantum states.

These results indicate the feasibility of QGANs for quantum data generation and classification tasks. They provide a robust framework for future quantum machine learning and high-energy physics applications.

5.7 Discussion and Future Work

The QGAN model demonstrated significant potential for quantum-classical hybrid applications in high-energy physics, but it is evident that there is still ample room for improvement and exploration. Beyond the visual comparisons provided by Q-spheres and Bloch spheres, more rigorous and quantitative methods can be employed to evaluate the similarity between real and generated quantum states. Metrics such as fidelity and measures derived from the Haar distance offer a more precise understanding of how closely the generated states replicate the real ones. This model's flexibility also opens the door to generating more complex data types, such as high-dimensional vectors or even quantum representations of classical data like images. This versatility makes QGANs a promising tool for high-energy physics and applications in quantum imaging and optimization.
Another exciting avenue for future work involves integrating classical discriminators into the QGAN architecture. To achieve this, the quantum generator would need to output classical data through various measurement strategies, such as extracting probabilities of computational basis states or performing measurements relative to specific observables. These classical outputs could then be compared to real data using advanced metrics like the Frechet distance or the Wasserstein distance, which are widely used in evaluating generative models. Combining quantum and classical components could significantly enhance the model's adaptability and utility.
The scalability of the QGAN model is another critical area to address. Expanding the architecture to accommodate larger datasets and higher-dimensional quantum systems would involve increasing the number of qubits and circuit layers, enabling the model to handle more complex data distributions. Furthermore, applying the QGAN framework to real experimental data, such as those collected from the Large Hadron Collider (LHC), presents an exciting opportunity to test the model's robustness and practicality under realistic conditions. This would also require the development of strategies to manage noisy quantum data and integrate it with experimental workflows.

Improving the training process is another key consideration. Like classical GANs, QGANs can suffer from instability during training. Addressing this challenge might involve exploring alternative loss functions, introducing regularization techniques, or leveraging advanced optimizers to improve convergence and stability. Additionally, the model's performance could be evaluated more comprehensively by combining quantum metrics, such as fidelity, with classical metrics like precision, recall, and the area under the curve (AUC). Such a hybrid approach would provide a more complete picture of the model's strengths and weaknesses.

This work represents an essential foundational step in exploring the potential of QGANs, but much remains to be done to unlock their capabilities thoroughly. With further research into advanced quantum metrics, the integration of classical components, and the extension to more complex datasets, QGANs have the potential to become a powerful tool for generating and classifying quantum data. As quantum computing hardware continues to improve, the practical applicability of QGANs to real-world problems will only increase, offering exciting possibilities for high-energy physics and beyond.

Chapter 6

Summary, Outlook and Conclusion

Quantum Generative Adversarial Networks (QGANs) represent a cutting-edge approach for generating synthetic quantum data while simultaneously enabling classification tasks, making them highly relevant for applications in quantum machine learning and high-energy physics. In this thesis, we explored the implementation and evaluation of a QGAN model capable of generating quantum states resembling real quantum data, as well as classifying events such as signal and background in the context of high-energy physics.

The QGAN architecture developed for this work integrates a quantum generator and discriminator, utilizing parameterized quantum circuits to create and assess quantum states. The generator employs techniques such as angle-based embedding to encode input data into quantum states, which are processed and refined through training to closely approximate the structure of real quantum datasets. Meanwhile, the discriminator evaluates the generated states, helping to distinguish between real and synthetic data, and iteratively improving the generator through adversarial training.

Results from the QGAN model demonstrated the feasibility of using quantum machine learning to generate and classify quantum states. Visualizations such as Q-spheres and Bloch spheres confirmed that the generator could produce synthetic quantum states closely resembling the target quantum states. Additionally, metrics such as fidelity, Wasserstein distance, and Frechet distance could provide further quantitative measures of similarity between generated and real data. Moreover, the loss curves and discriminator accuracy highlighted the training dynamics and the impact of hyperparameters like α, which modulates the balance between the classification and data generation objectives. The discriminator achieved stable performance with an accuracy of around 75%, showcasing the model's capability to distinguish between real and generated data.

The QGAN framework developed in this thesis is flexible and scalable, opening the door to numerous future applications. For example, it can be extended to generate more complex datasets such as high-dimensional quantum states or even quantum representations of classical images. Additionally, incorporating classical discriminators into the QGAN workflow could facilitate the use of more robust evaluation metrics, such as probabilistic measurements or observable-based outputs, to bridge the gap between quantum and classical systems. The exploration of alternative quantum encodings, such as amplitude or tensor-based methods, could further enhance the model's expressiveness and scalability.

While this work has laid the groundwork for QGANs in quantum data generation and classification, much remains to be explored. Challenges such as scaling the model to larger datasets, integrating real experimental data, and improving training stability are areas ripe for future research. The

application of QGANs to real-world scenarios, such as data collected from the Large Hadron Collider (LHC), would test their robustness in practical settings. Furthermore, improving training techniques, such as exploring alternative loss functions and optimizing hyperparameters, could address stability issues inherent in adversarial models.

The implications of this research extend beyond high-energy physics. QGANs could find applications in fields like quantum chemistry, optimization problems, and financial modeling, where the ability to generate and analyze quantum data is increasingly valuable. This work demonstrates the potential of QGANs to bridge the gap between quantum computing and machine learning, paving the way for further advancements in quantum data generation and classification.

In conclusion, this thesis contributes to the growing field of quantum machine learning by developing and analyzing a QGAN model tailored to generate and classify quantum data. While challenges remain, the flexibility and scalability of the QGAN framework offer exciting opportunities for future exploration and application in various domains. With ongoing advancements in quantum hardware and algorithms, QGANs are poised to become a cornerstone of quantum data science.

Bibliography

[1] S. 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. DOI: 10.1016/S0168-9002(03)01368-8. URL: https://doi.org/10.1016/S0168-9002(03)01368-8.

[2] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[3] Arthur Pesah et al. "Absence of Barren Plateaus in Quantum Convolutional Neural Networks". In: *Physical Review X* 11.4 (2021). DOI: 10.1103/physrevx.11.041011. URL: https://doi.org/10.1103%2Fphysrevx.11.041011.

[4] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[5] Martin Larocca et al. "Group-Invariant Quantum Machine Learning". In: (2022). DOI: 10.48550/ARXIV.2205.02261. URL: https://arxiv.org/abs/2205.02261.

[6] Matthias C. Caro et al. "Generalization in quantum machine learning from few training data". In: (2021). DOI: 10.48550/ARXIV.2111.05292. URL: https://arxiv.org/abs/2111.05292.

[7] Md Sajid Anis et al. *Qiskit: Basic Syntax*. 2022. URL: https://qiskit.org/textbook/chappendix/qiskit.html.

[8] Alexander Amini. *MIT Introduction to Deep Learning 6.S191: Lecture 3*. 2020.

[9] Ville Bergholm et al. "PennyLane: Automatic Differentiation of Hybrid Quantum-Classical Computations". In: *arXiv preprint* 2004.06768 (2020). URL: https://arxiv.org/abs/2004.06768.

[10] Amey Bhatuse. *Quantum Generative Adversarial Neural Networks for High Energy Physics Analysis at the LHC*. Sept. 2022. URL: https://github.com/ML4SCI/QMLHEP/tree/main/Quantum_GAN_for_HEP_Amey_Bhatuse/QGANSHEP.

[11] Joseph Bowles, Shahnawaz Ahmed, and Maria Schuld. *Better than classical? The subtle art of benchmarking quantum machine learning models*. 2024. eprint: arXiv:2403.07059.

[12] Paolo Calafiura, David Rousseau, and Kazuhiro Terao. "FRONT MATTER". In: *Artificial Intelligence for High Energy Physics*, pp. i–xii. DOI: 10.1142/9789811234033_fmatter. eprint: https://www.worldscientific.com/doi/pdf/10.1142/9789811234033_fmatter. URL: https://www.worldscientific.com/doi/abs/10.1142/9789811234033_fmatter.

[13] CERN. *Detector: CMS Experiment*. Accessed: 2023-11-22. 2023. URL: https://cms.cern/detector.

[14] CERN. *The Large Hadron Collider*. Accessed: 2023-11-22. 2019. URL: https://home.cern/science/accelerators/large-hadron-collider.

[15] L. Cervantes Guevara. *Quantum Convolutional Neural Networks for High Energy Physics*. Undergraduate Thesis. Accessed: 2023-11-22. Sept. 2022. URL: https://hdl.handle.net/20.500.12371/16897.

[16] Samuel Yen-Chi Chen et al. *Quantum Convolutional Neural Networks for High Energy Physics Data Analysis*. 2020. eprint: arXiv:2012.12177.

[17] CMS Collaboration. "Particle flow reconstruction and global event description with the CMS detector". In: *Journal of Instrumentation* 12.10 (2017), P10003–P10003. DOI: 10.1088/1748-0221/12/10/P10003.

[18] Gavin E. Crooks. "Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition". In: (2019). DOI: 10.48550/ARXIV.1905.13311. URL: https://arxiv.org/abs/1905.13311.

[19] Lazaro Diaz. *GitHub Repository for Thesis*. Accessed: 2024-12-01. 2024. URL: https://github.com/LazaroR-u.

[20] Gilles Brassard Esma Aïmeur and Sébastien Gambs. "Machine Learning in a Quantum World". In: *Advances in Artificial Intelligence*. Ed. by Luc Lamontagne and Mario Marchand. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 431–442. ISBN: 978-3-540-34630-2.

[21] European Commission. *Artificial Intelligence for Europe*. Accessed: 2023-11-22. 2018. URL: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=COM%3A2018%3A237%3AFIN.

[22] J. de Favereau et al. "DELPHES 3: A Modular Framework for Fast Simulation of a Generic Collider Experiment". In: *Journal of High Energy Physics* 2014.2 (Feb. 2014), p. 57. DOI: 10.1007/jhep02(2014)057. URL: https://doi.org/10.1007/jhep02(2014)057.

[23] Wen Guan et al. "Quantum machine learning in high energy physics". In: *Machine Learning: Science and Technology* 2.1 (Mar. 2021), p. 011003. DOI: 10.1088/2632-2153/abc17d. URL: https://dx.doi.org/10.1088/2632-2153/abc17d.

[24] Dan Guest, Kyle Cranmer, and Daniel Whiteson. "Deep Learning and its Application to LHC Physics". In: (2018). DOI: 10.1146/annurev-nucl-101917-021019. eprint: arXiv:1806.11484.

[25] He-Liang Huang et al. "Near-Term Quantum Computing Techniques: Variational Quantum Algorithms, Error Mitigation, Circuit Compilation, Benchmarking and Classical Simulation". In: (2022). DOI: 10.1007/s11433-022-2057-y. eprint: arXiv:2211.08737.

[26] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: (2014). DOI: 10.48550/ARXIV.1412.6980. URL: https://arxiv.org/abs/1412.6980.

[27] K. Lassila-Perini. *Overview of CMS Physics Goals and Detector*. Accessed: 2023-11-22. 2015. URL: https://twiki.cern.ch/twiki/bin/view/CMSPublic/WorkBookCMSExperiment.

[28] Diana León Silverio. *Search for Monotop Production in Events with One Lepton, Missing Transverse Energy, and Jets*. Undergraduate Thesis. Nov. 2018.

[29] Weikang Li, Zhide Lu, and Dong-Ling Deng. "Quantum Neural Network Classifiers: A Tutorial". In: *SciPost Phys. Lect. Notes* (2022), p. 61. DOI: 10.21468/SciPostPhysLectNotes.61. URL: https://scipost.org/10.21468/SciPostPhysLectNotes.61.

[30] Joshua Lin et al. "Boosting $H \to b\bar{b}$ with Machine Learning". In: (2018). DOI: 10.1007/JHEP10(2018)101. eprint: arXiv:1807.10768.

[31] Warren S. McCulloch and Walter Pitts. "A Logical Calculus of the Ideas Immanent in Nervous Activity". In: *The Bulletin of Mathematical Biophysics* 5 (1943), pp. 115–133. DOI: 10.1007/BF02478259.

[32] Alberto Di Meglio et al. *Quantum Computing for High-Energy Physics: State of the Art and Challenges. Summary of the QC4HEP Working Group*. 2023. eprint: arXiv:2307.03236.

[33] Pankaj Mehta et al. "A high-bias, low-variance introduction to Machine Learning for physicists". In: *Physics Reports* 810 (2019), pp. 1–124. DOI: 10.1016/j.physrep.2019.03.001.

[34] Satoshi Morita and Hidetoshi Nishimori. "Mathematical foundation of quantum annealing". In: *Journal of Mathematical Physics* 49.12 (2008), p. 125210.

[35] Kouhei Nakaji and Naoki Yamamoto. "Quantum semi-supervised generative adversarial network for enhanced data classification". In: (2020). DOI: 10.1038/s41598-021-98933-6. eprint: arXiv:2010.13727.

[36] Michael A. Nielsen. *Neural Networks and Deep Learning*. http://neuralnetworksanddeeplearning.com/. Accessed: 2023-11-22. 2018.

[37] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. 10th Anniversary Edition. Cambridge, UK: Cambridge University Press, 2000. ISBN: 978-1-107-00217-3. URL: https://doi.org/10.1017/CBO9780511976667.

[38] Seunghyeok Oh, Jaeho Choi, and Joongheon Kim. *A Tutorial on Quantum Convolutional Neural Networks (QCNN)*. 2020. eprint: arXiv:2009.09423.

[39] PennyLane. *PennyLane: Learn Quantum Programming*. Accessed: 2023-11-22. 2023. URL: https://pennylane.ai/qml.

[40] Adrián Pérez-Salinas et al. "Data re-uploading for a universal quantum classifier". In: (2019). DOI: 10.22331/q-2020-02-06-226. eprint: arXiv:1907.02085.

[41] Frank Rosenblatt. "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain". In: *Psychological Review* 65.6 (1958), pp. 386–408. DOI: 10.1037/h0042519.

[42] Maria Schuld and Francesco Petruccione. *Supervised Learning with Quantum Computers*. Springer Cham, 2018. DOI: 10.1007/978-3-319-96424-9.

[43] Torbjörn Sjöstrand et al. "An Introduction to PYTHIA 8.2". In: *Computer Physics Communications* 191 (June 2015), pp. 159–177. DOI: 10.1016/j.cpc.2015.01.024. URL: https://doi.org/10.1016/j.cpc.2015.01.024.

[44] Sandro Skansi. *Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence*. Springer, 2018.

[45] S. Vallecorsa. "Generative models for fast simulation". In: *Journal of Physics: Conference Series* 1085.2 (Sept. 2018), p. 022005. DOI: 10.1088/1742-6596/1085/2/022005. URL: https://dx.doi.org/10.1088/1742-6596/1085/2/022005.

[46] Yunfei Wang and Junyu Liu. "A comprehensive review of Quantum Machine Learning: from NISQ to Fault Tolerance". In: (2024). DOI: 10.1088/1361-6633/ad7f69. eprint: arXiv:2401.11351.