國立臺灣大學理學院物理學研究所

博士論文

Department of Physics

College of Science

National Taiwan University

Doctoral Dissertation

量子完全圖神經網路在高能物理噴流辨識的應用及量子優勢之探討

Quantum Complete Graph Neural Network in the Jet Discrimination of High-Energy Physics and Discussion on its Quantum Advantage

陳奕安

Yi-An Chen

指導教授: 陳凱風 教授

Advisor: Prof. Kai-Feng Chen

中華民國 114 年 5 月

May, 2025

doi:10.6342/NTU202501658

國立臺灣大學博士學位論文
口試委員會審定書
PhD DISSERTATION ACCEPTANCE CERTIFICATE
NATIONAL TAIWAN UNIVERSITY

量子完全圖神經網路在高能物理噴流辨識的應用及量子優勢之探討

Quantum Complete Graph Neural Network in the Jet Discrimination of

High-Energy Physics and Discussion on its Quantum Advantage

本論文係 **陳奕安 F08222011** 在國立臺灣大學物理學系完成之博士學位論文，於民國 114 年 05 月 27 日承下列考試委員審查通過及口試及格，特此證明。

The undersigned, appointed by the Department of Physics on 27 / 05 / 2025 have examined a PhD dissertation entitled above presented by **Yi-An Chen F08222011** candidate and hereby certify that it is worthy of acceptance.

口試委員 Oral examination committee:

陳彩風	吳肇祥	管希聖
（指導教授 Advisor）		
吳正庸	林俊達	
許琇娟		

Acknowledgements

One day in September 2023, I was in Taitung for the Moonlight-Sea Concert (月光・海音樂會). Unfortunately, Typhoon Haikui arrived, and everyone was stuck at the Gift Box Hostel. Since I couldn't go anywhere, I picked up a pen and some paper and started doing some calculations. About an hour later, the prototype of QCGNN was born...

Back in my undergraduate years, I never imagined I would become a programmer. In my fourth year, I took the course *"Introduction to Numerical Analysis"* just to accompany some friends. That's when I met my advisor, **Kai-Feng Chen (Jack)**, who later became the MVP of my PhD journey. After that course, I became obsessed with coding and deeply interested in machine learning. Jack's warm smile in every lecture inspired me to become his student. I'm truly grateful to Jack for giving me freedom in research and always encouraging me when I felt lost in the world of quantum machine learning. Thanks to his support, I was able to find a research direction, write a paper, handle replies to referees and editors, and eventually publish it in *Physical Review D*. He also encouraged and supported me to attend summer schools and international conferences. Among all the choices I've made in life, choosing Jack as my advisor was definitely one of the best.

In the academic community, I've received a lot of help from many people. Though not officially, Professor **Jiunn-Wei Chen** has acted like my co-advisor. We used to have regular meetings discussing machine learning and quantum computing. I also had the great pleasure to take his course on particle physics, which further deepen my understanding about quantum theories. He always provided sharp insights, got straight to the point, and had deep understanding of the details. He's a humble person who treats students with respect. From him, I learned not only how to think and study, but also how to be a researcher with the right attitude. I also want to thank Professor **Hsi-Sheng Goan** for writing recommendation letters for me multiple times in the field of quantum computing, even though I wasn't his student and we didn't have meetings. I'm really thankful for his kindness. In addition, I would like to thank Professors **Rong-Shyang Lu**, **Cheng-Wei Chiang**, **Guin-Dar Lin**, and **Hsiu-Chuan Hsu** for serving on my oral defense committee and for the insightful discussions on machine learning and quantum computing.

doi:10.6342/NTU202501658

I especially want to thank my girlfriend, **Yen-Ju Wei**. As a pioneer working alone in the lab, I often felt frustrated and depressed. Even though you couldn't give suggestions on my research, just staying by my side and studying together at coffee shops gave me a lot of support. When I had the opportunity to go to Europe for the CHEP conference, we worked hard to plan the trip and finally went on our first Europe journey together.

Thanks also to my friends from NTUPHYS and NTUCYLS for being there at the beginning of my PhD and for playing badminton together. Special thanks to **Andrew Wu**, not only for joining us in Europe, but also for providing accommodations and planning trips when we visited California during your time at UC Davis.

In the last years of my PhD, I moved to room 922 in the Astronomy-Mathematics Building. Deciding what to eat for lunch and dinner was often harder than doing research, especially with some foodies around. It was a good time spent drinking beers and cocktails, and playing games like League of Legends and MapleStory. We also traveled together during TPS and TIDC meetings, and it was always a lot of fun. Chatting with each other when we got bored or frustrated with research was one of the most relaxing things, and it really helped keep things chill.

In 2024, I joined AEPSHEP in Thailand, a 14-day summer school covering various topics in high-energy physics. I was the only Taiwanese participant, and I really appreciate my friends from China for chatting with me in Chinese, and my friends from Japan for letting me practice Japanese. It was a wonderful experience meeting people from around the world, like bumping into my Norwegian friends at CERN, and meeting friends from Korea, Thailand, India, and more during the 2025 LHCP in Taiwan.

Last but not least, I want to thank my parents for always supporting me. At every stage of my student life, you respected my decisions and never interfered. I also want to thank my grandparents for their support, whether it was food or financial help. Special thanks to my dad, my grandma, and the Taipei City Government for letting me drive a car for free, including fuel and parking.

doi:10.6342/NTU202501658

摘要

機器學習技術，尤其是深度神經網路，近年來逐漸成為高能物理領域當中分析的重要工具。這些方法已在多項應用中展現出驚人的效果且強大的潛力。隨著機器學習與量子運算的整合，一個特別的新研究領域「量子機器學習」逐漸成形，預期能在處理更複雜的資料時提供一定的優勢。

在本論文中，我們提出一種名為「量子完全圖神經網路（QCGNN）」的方法，它是一種建立在變分量子電路所設計的演算法，目標是處理完全圖上的學習任務。該模型透過參數化的量子邏輯閘達成資料編碼，並利用量子平行性的特性，期許在一些特別場景中能比古典方法帶來更有效率的運算。

為了比較，我們將此模型應用於大型強子對撞機（LHC）中的質子－質子碰撞事件，處理噴流分類的任務。在這個情境下，噴流是由高能夸克或膠子所產生的粒子集合，並可建構為完全圖的形式。由於噴流分類對理解粒子交互作用中有著關鍵性的影響，所以更快速且精準的模型來辨識不同來源的噴流可以有更好的分析結果。QCGNN 展現出能有效處理基於圖形結構的噴流資料的能力，有望在未來高亮度 LHC 實驗中發揮其應用潛力。

此外，我們也與古典深度學習模型進行比較分析，以評估 QCGNN 的效能表現。我們也在 IBM 的量子硬體上實作並測試此模型，以探討其在真實量子設備上的可行性。整體而言，QCGNN 不僅對圖形方法中常見的運算挑戰提出了解法，也展現出在量子運算與高能物理交叉領域中具有的研究潛力。

關鍵字：機器學習、量子機器學習、量子電腦、噴流辨識、高能物理

doi:10.6342/NTU202501658

doi:10.6342/NTU202501658

Abstract

Machine learning techniques, especially deep neural networks, have gradually become important tools in the field of high-energy physics. These methods have shown encouraging results in various applications. In recent years, the combination of machine learning and quantum computing has led to the development of a new field of research field called the “Quantum Machine Learning”, which is expected to offer advantages in handling increasingly complex data.

This dissertation proposes the Quantum Complete Graph Neural Network (QCGNN) designed for learning tasks on fully connected graphs, which is a VQC-based algorithm, where VQC stands for “Variational Quantum Circuit”. The QCGNN makes use of parameterized quantum circuits and aims to benefit from quantum parallelism, potentially providing computational advantages compared to classical approaches.

We apply the QCGNN to the task of jet discrimination in proton-proton collisions at the Large Hadron Collider (LHC). In this context, jets, which are collections of particles originating from energetic partons, are represented as complete graphs. Since jet classification plays a crucial role in understanding particle interactions, an effective model is necessary to distinguish between different jet types. The QCGNN shows the ability to process graph-based jet data efficiently, which may be helpful for future analyses at the high-luminosity LHC.

In addition, we compare the QCGNN with classical deep learning models to evaluate its performance. We also test the implementation of QCGNN on IBM quantum hardware to examine its feasibility on real devices. The results suggest that the QCGNN has poten-

doi:10.6342/NTU202501658

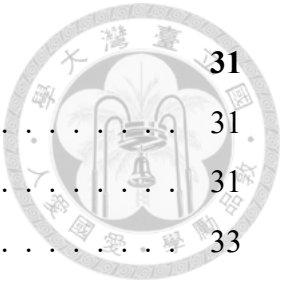
tial for further exploration in both quantum computing and high-energy physics research.

Keywords: Machine Learnning, Quantum Machine Learning, Quantum Computation, Jet Discrimination, High-Energy Physics

doi:10.6342/NTU202501658

Contents

doi:10.6342/NTU202501658



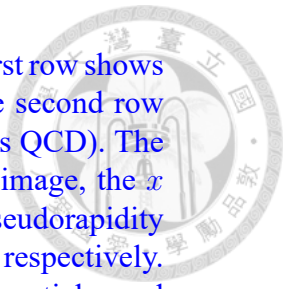
doi:10.6342/NTU202501658

doi:10.6342/NTU202501658

doi:10.6342/NTU202501658

List of Figures

doi:10.6342/NTU202501658



doi:10.6342/NTU202501658

doi:10.6342/NTU202501658

doi:10.6342/NTU202501658

doi:10.6342/NTU202501658

doi:10.6342/NTU202501658

List of Tables

doi:10.6342/NTU202501658

doi:10.6342/NTU202501658

Denotation

HEP	High Energy Physics
LHC	Large Hadron Collider
DNN	Deep Neural Network
MLP	Multi-Layer Perceptron
QML	Quantum Machine Learning
IR	Index Register
NR	Network Register
VQC	Variational Quantum Circuit
GNN	Graph Neural Network
MPGNN	Message-Passing Graph Neural Network
QCGNN	Quantum Complete Graph Neural Network
IBMQ	IBM Quantum Platform
PFN	Particle Flow Network
PNet	Particle Net
ParT	Particle Transformer
USO	Uniform State Oracle

doi:10.6342/NTU202501658

doi:10.6342/NTU202501658

Chapter 1

Introduction

1.1 High-Energy Physics (HEP)

Throughout the development of physics, scientists have continuously sought to explore the fundamental properties of the universe. In the early period, classical mechanics provided a basis to model physical systems using deterministic rules. Nevertheless, it was eventually recognized that such theories could not fully account for particle behaviour at microscopic scales. This drawback led to the rise of quantum mechanics, which brought novel ideas including the uncertainty principle and the wave-particle duality.

Later on, quantum mechanics was extended by incorporating special relativity, leading to the framework known as quantum field theory (QFT). In QFT, each elementary particle has a corresponding field, and the particles are recognized as the excitations of fields. The interactions are described through field theory and symmetry principles, i.e., gauge symmetries. Using this approach, the Standard Model (SM) was formulated. It offers a unified explanation of three out of the four fundamental interactions: strong, weak, and electromagnetic forces. The SM also predicted the existence of the Higgs boson, which is the excitation of the Higgs field, and was experimentally confirmed in 2012 [1, 2].

To verify the predictions of the SM and explore potential new physics, experiments operating at extremely high energies are necessary. In CERN (the European Organization for Nuclear Research), the Large Hadron Collider (LHC) collides protons at velocities approximately the speed of light, LHC enables the recreation of conditions similar to those just after the Big Bang. This capability makes it possible for physicists to investigate elementary particles and their fundamental interactions in great depth.

Although the SM is highly successful, it still cannot explain several important issues, such as dark matter, which could be a possible candidate to explain the origin of neutrino mass,

doi:10.6342/NTU202501658

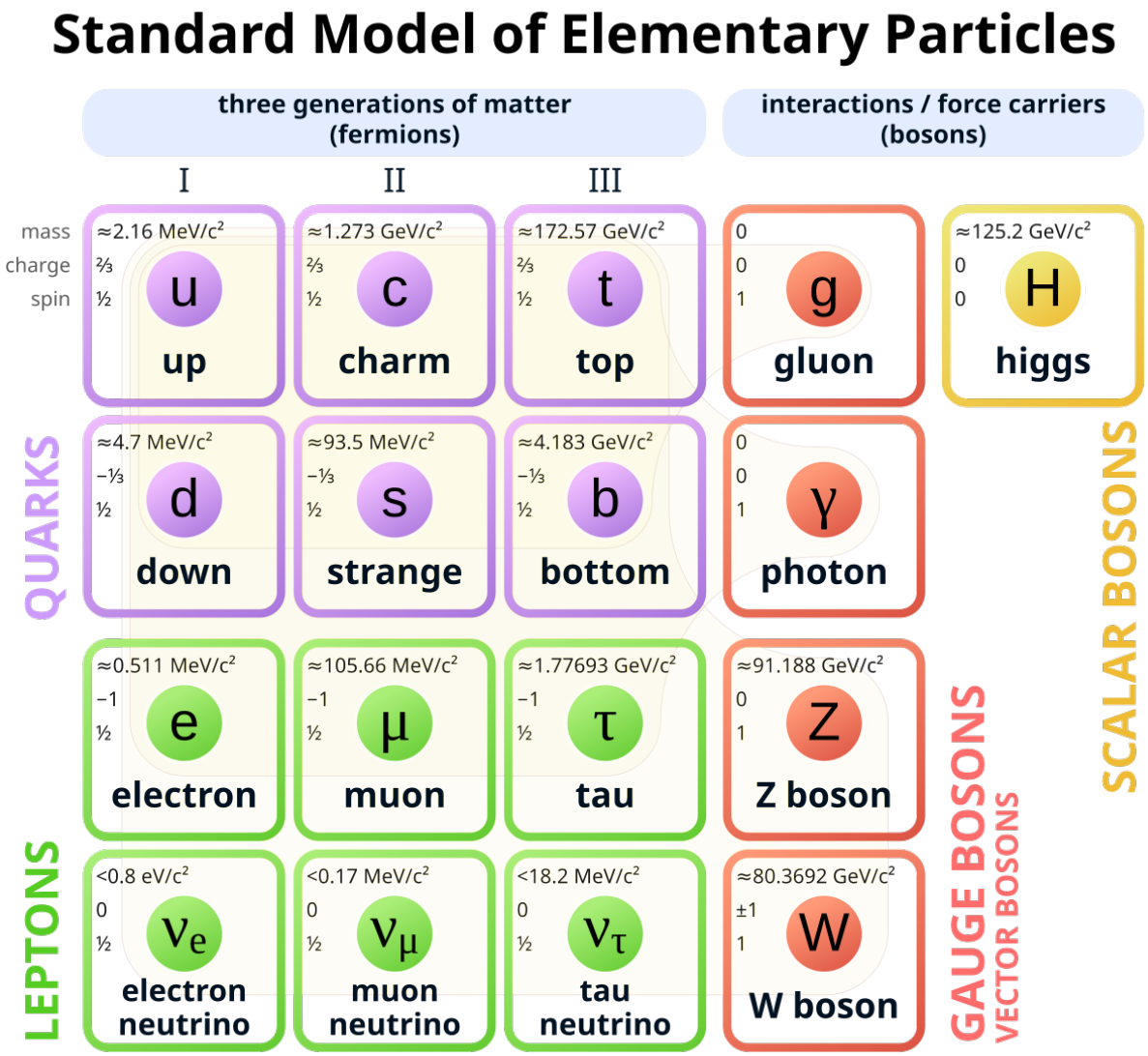


Figure 1.1: The elementary particles of Standard Model presented as a table. Matter particles, known as fermions, consist of quarks and leptons, and are grouped into three distinct generations. The force carriers, or bosons, which are responsible for mediating interactions, include the gluon (g), photon (γ), Z and $W^{\pm}$ bosons, as well as the Higgs boson (H). Figure adapted from [3].

doi:10.6342/NTU202501658

and the incorporation of gravity. To explore these unanswered questions, the High-Luminosity LHC (HL-LHC) project has been proposed for the search for new physics. This upgrade will significantly increase the luminosity of collisions, allowing more data to be collected. With this improvement, researchers hope to find evidence of new particles or interactions that go beyond the SM.

In summary, HEP aims to answer fundamental questions about the universe. With the development of advanced theoretical frameworks and powerful experimental facilities like the LHC, scientists continue to push the boundaries of our knowledge.

1.2 Machine Learning (ML)

Machine learning, which aims to find efficient and automatic methods capable of learning complex or intrinsic relations from data and achieves great success in different applications, is a growing field of artificial intelligence and computer science. In the early stages, many ML models were based on well-defined mathematical rules and decision processes. Examples include the binary decision tree (BDT), which splits data into different branches based on feature thresholds, and the support vector machine (SVM), which finds a hyperplane to classify points in the high-dimensional latent space. These methods were widely used for tasks such as regression and classification due to their simplicity and interpretability.

As the demand for more powerful models increased, researchers began exploring neural networks, which are motivated by neurons in our brains. A neural network is constructed by several layers of neurons that process input data through complicated computations with tunable parameters and non-linear operations. Neural networks can represent complex functions and are able to discover intrinsic features from data, especially when the network is deep and constructed with many layers.

The progress in hardware, especially with the emergence of graphics processing units (GPUs), has accelerated the development of artificial neural networks. Owing to the property of parallel computation, GPUs can handle massive datasets efficiently, thereby allowing it feasible to train deep neural networks that contain millions or even billions of parameters. This advancement has significantly contributed to the growth in scale of deep learning models.

A notable milestone in modern ML is the introduction of large language models (LLMs), which are built upon the transformer framework [4]. Originally designed for sequence-to-sequence applications, the transformer architecture makes use of self-attention mechanisms as its core computational strategy, allowing it to capture global dependencies between tokens in long-sequential data. Thanks to their straightforward structure and strong scalability with in-

doi:10.6342/NTU202501658

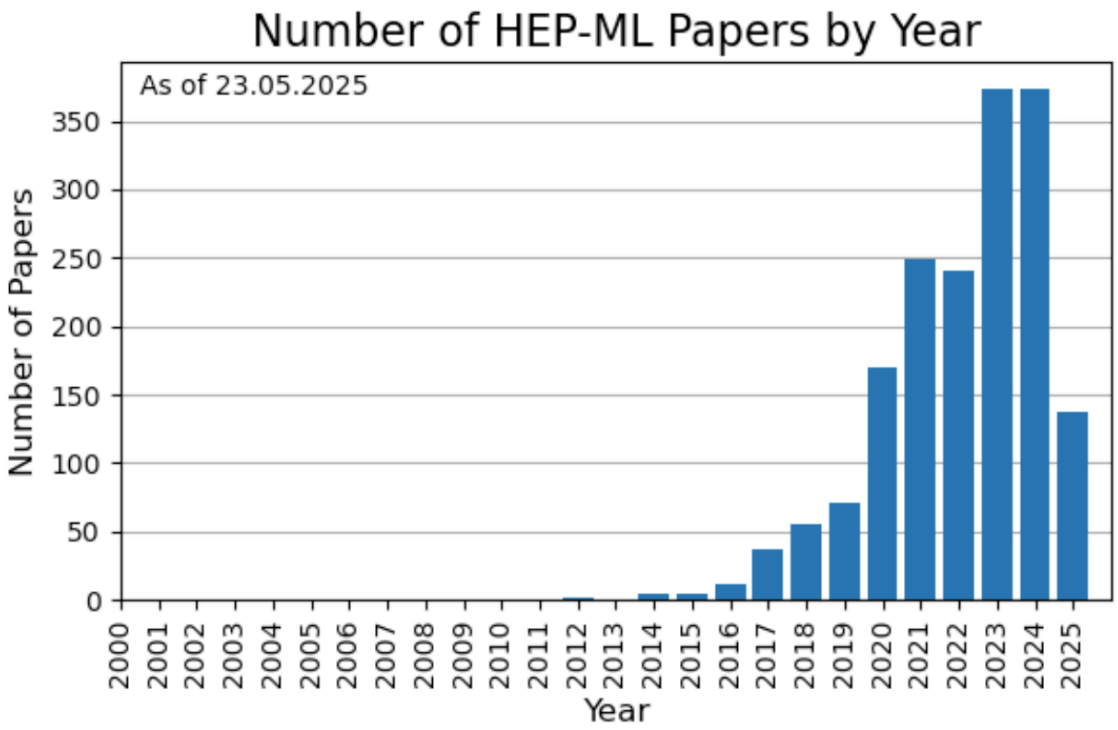


Figure 1.2: Annual number of papers at the intersection of high-energy physics and machine learning, from 2000 to 2025 (as of May 23, 2025). This plot is adopted from [5].

creasing data and model size, transformers have been chosen as the backbone of state-of-the-art LLMs. Examples such as ChatGPT, DeepSeek, Claude, and Gemini have shown successful performance in different tasks by leveraging extensive parameter counts and vast training datasets.

In conclusion, ML has evolved from rule-based models to highly complex neural architectures, powered by advances in computational resources. These developments continue to transform various scientific and industrial fields.

1.3 Jet Discrimination with ML in HEP

Jets arising from energetic proton-proton interactions at the LHC appear as bunches of particles, typically emerging from the hadronization process of high-energy partons, which consist of gluons and quarks. Accurately identifying and categorizing these jets is vital for probing the underlying particle interactions and for uncovering potential evidence of new physics. The term *jet discrimination* encompasses a range of techniques aimed at tracing the origin of jets and distinguishing those produced by different initiating particles.

Deep neural networks (DNNs) have gained significant traction in HEP [5–7], and have proven effective in addressing jet classification challenges in recent years. These models are

doi:10.6342/NTU202501658

capable of learning complex features and intrinsic correlations within the data, thereby improving the accuracy of jet tagging based on their internal properties. One widely adopted approach is to represent jets as graphs, where individual particles serve as nodes and their interactions are encoded as edges [8–21]. This graphical formulation allows both local and global structural information to be captured, making it particularly advantageous for jet-related learning tasks.

However, as the number of particles within jets grows, the computational cost of graph-based algorithms grows roughly quadratically. This scalability issue becomes especially prominent in the high-luminosity upgrade of the LHC, where both the total dataset size and the number of particles are projected to rise substantially.

Quantum machine learning (QML) [22–26] integrates machine learning and quantum computation. It presents a potentially powerful strategy for tackling the growing complexity of data in HEP. By using properties such as superposition and entanglement, QML facilitates computations that would otherwise be challenging or infeasible using conventional computing architectures.

This dissertation introduces and provides a detailed implementation of the Quantum Complete Graph Neural Network (QCGNN) [27], a new QML framework tailored for jet classification using fully connected graph representations. Although inspired by the design of classical graph neural networks (GNNs), the QCGNN incorporates quantum algorithms to improve both computational performance and efficiency. When analyzing jets containing N particles, standard GNNs often require a computational cost scaling as $O(N^2)$. In contrast, the QCGNN achieves a more favourable scaling of $O(N)$ by taking advantage of quantum computational principles. This efficiency gain allows QCGNN to handle more extensive datasets and capture finer jet substructures, making it a compelling candidate for applications in upcoming HEP experiments.

This dissertation is arranged as the following structure: Chapter 2 gives a comprehensive introduction to jets in HEP, including their generation mechanisms, properties, and the common observables used for jet discrimination. Chapter 3 examines traditional ML approaches for jet classification, with special attention paid to GNNs and their variants. In Chapter 4, we transition into QML, beginning with foundational concepts in quantum computing and gradually moving toward the study of variational quantum circuits [28–30]. Chapter 5 is devoted to the development of the QCGNN, where we detail its architecture and evaluate its computational advantages in the context of quantum acceleration. Chapter 7 presents a performance comparison between QCGNN and conventional ML-based jet discrimination techniques. The results show the feasibility of QCGNN, and underscore its potential in advancing research within the HEP domain. The dissertation concludes by summarizing key insights and proposing future directions for the development and application of QCGNN in experimental studies.

doi:10.6342/NTU202501658

doi:10.6342/NTU202501658

Chapter 2

Jet in High-Energy Physics

In this chapter, we will discuss the hadronization process and the concept of jet discrimination in high-energy physics. We will begin by introducing the hadronization process, which is crucial for understanding how jets are formed from partons in high-energy collisions. We will then explore the cylindrical coordinate system used in particle detectors, followed by a detailed explanation of the anti-k_T algorithm, which is widely used for jet clustering. Finally, we will introduce particle flow features that are essential for characterizing jets and their constituents.

2.1 Hadronization and Jet Discrimination

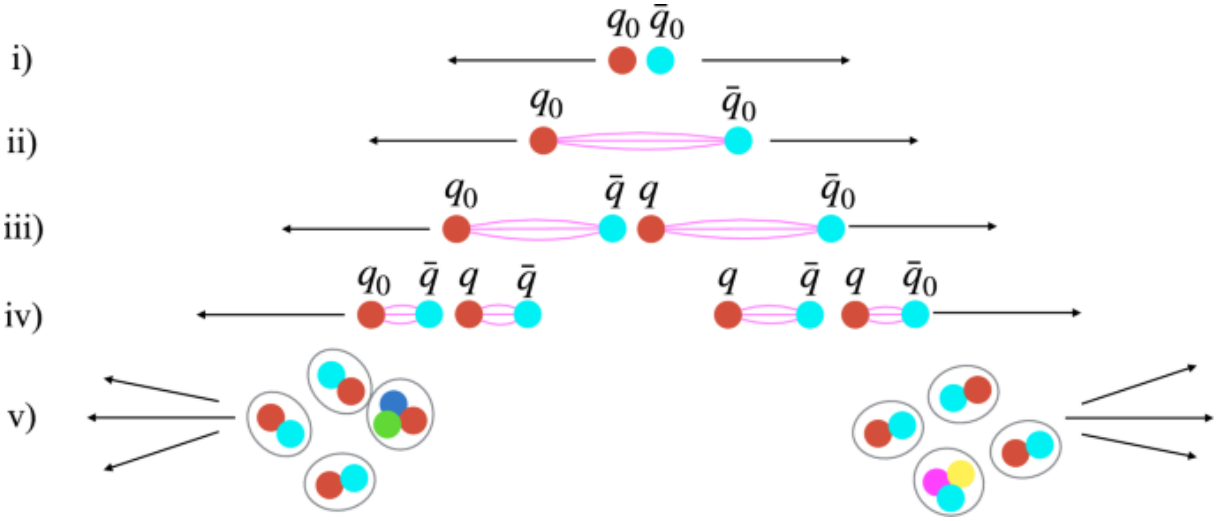


Figure 2.1: A schematic diagram illustrating the hadronization process, as shown in [31]. In step (i), a quark (q_0) and an antiquark ($\bar{q}_0$) are created from a hard scattering event. In step (ii), the quark and antiquark move apart, and as the distance between them increases, another quark-antiquark pair is generated from the vacuum, as shown in step (iii). This process continues, as depicted in step (iv), and eventually, color-neutral hadrons are formed in step (v).

doi:10.6342/NTU202501658

In the LHC, the high-energy proton-proton collisions create sprays of particles called jets, where energetic quarks and gluons go through a process known as hadronization and produce narrow sprays of particles. According to quantum chromodynamics (QCD), partons that carry color charge, namely quarks and gluons, cannot exist freely at long distances. This behavior, known as color confinement, ensures that only color-neutral combinations, or singlet states, appear as observable hadrons at the end of the process.

When two protons collide at relativistic energies, partons within the protons interact and undergo hard scatterings, often imparting large transverse momenta to the outgoing partons. These energetic quarks or gluons then initiate parton showers through successive emissions of gluons and quark-antiquark pairs, a process that is governed by QCD splitting functions and is well-modeled using perturbative techniques up to a certain energy scale.

As the virtuality of the partons decreases and the coupling strength α_s becomes large, the perturbative description breaks down. At this non-perturbative stage, the partons undergo hadronization, a complex transition into colorless hadronic states. During hadronization, the potential energy between quarks grows linearly with distance, as described by the confining potential:

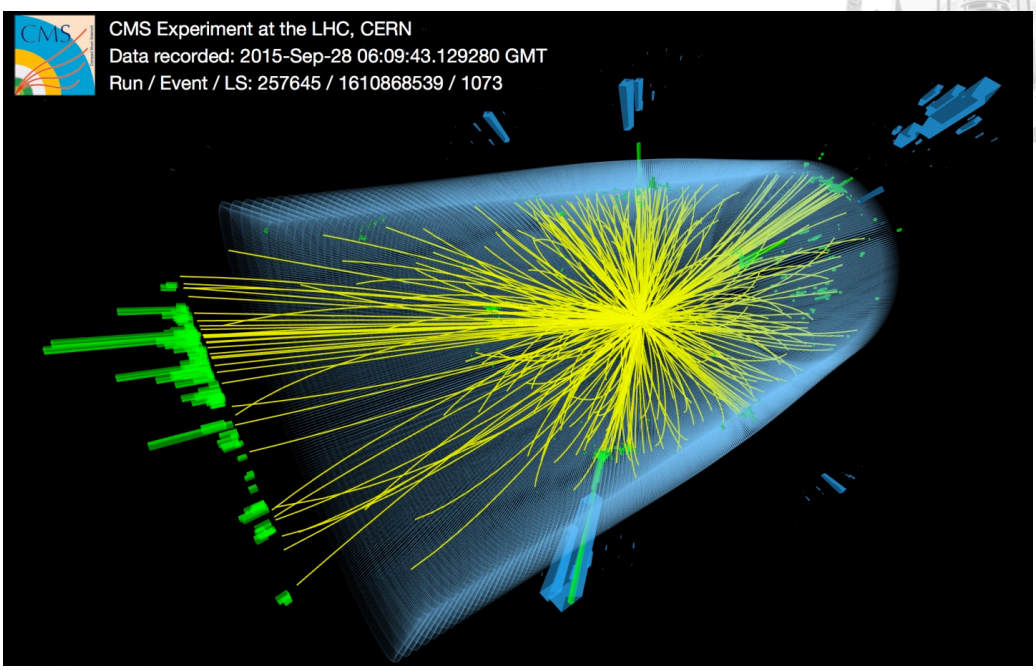
$$V(r) = \sigma r, \tag{2.1}$$

where σ is the string tension and r is the distance between the quarks. As partons move apart, the energy in the string between them becomes large enough to produce quark-antiquark pairs from the vacuum, leading to the formation of new color-neutral hadrons. Figure 2.1 illustrates this process, where a quark and an antiquark are created from a hard scattering event. The hadronization process results in a spray of hadrons that are spatially clustered around the original parton's direction of motion.

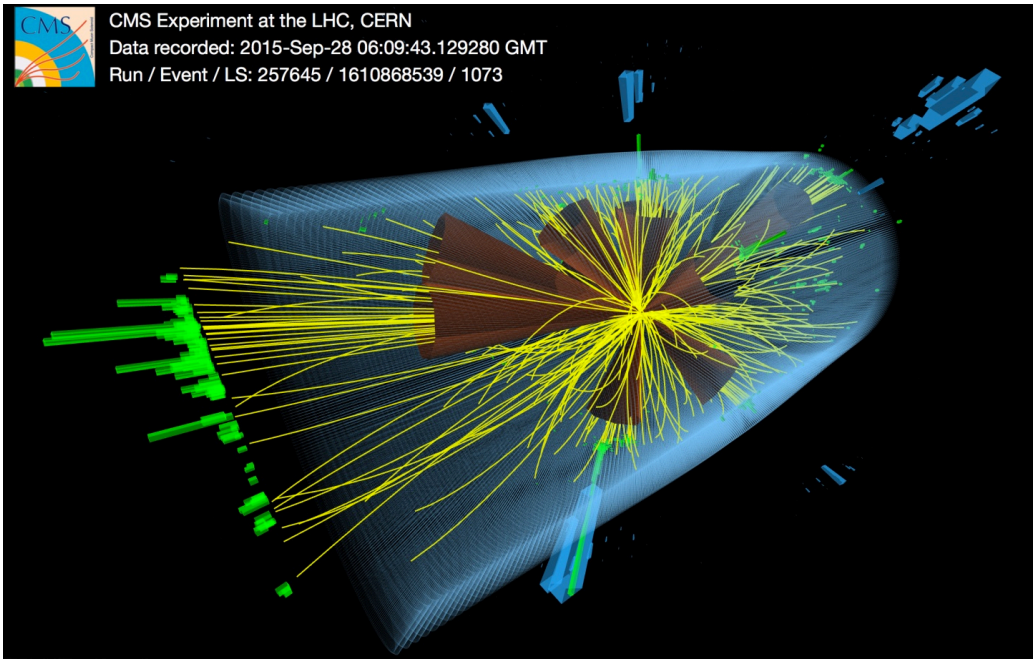
The resulting structure, observed in detectors as a localized clustering of energy and particle tracks, is identified as a jet. The identification and reconstruction of jets are essential for probing the dynamics of the initial partonic interaction, inferring the properties of the initiating partons, and testing predictions of QCD. Jet clustering algorithms, such as anti-k_T, are employed to cluster final-state particles into jets in a theoretically and experimentally consistent manner. Figure 2.2a shows a real event captured by the CMS detector, and Figure 2.2b shows the jets reconstructed through some jet clustering algorithm.

At the LHC, a variety of particles can produce jets (see Figure 2.3), including light quarks (u, d, s), gluons, heavier quarks such as bottom (b), charm (c), and top (t), as well as gauge bosons (W/Z) and Higgs bosons. Jet discrimination, also known as jet identification or jet tagging, encompasses techniques used to determine the origin of jets and distinguish those produced by different particle species.

doi:10.6342/NTU202501658



(a) An event collected by the CMS detector.



(b) Jets clustered with brown cones.

Figure 2.2: A real event captured by the CMS detector, adopted from [32]. Figure 2.2a shows the event display of a proton-proton collision at the Large Hadron Collider, captured by the CMS detector. The brown cones in Figure 2.2b represent the jets reconstructed through a jet clustering algorithm.

doi:10.6342/NTU202501658

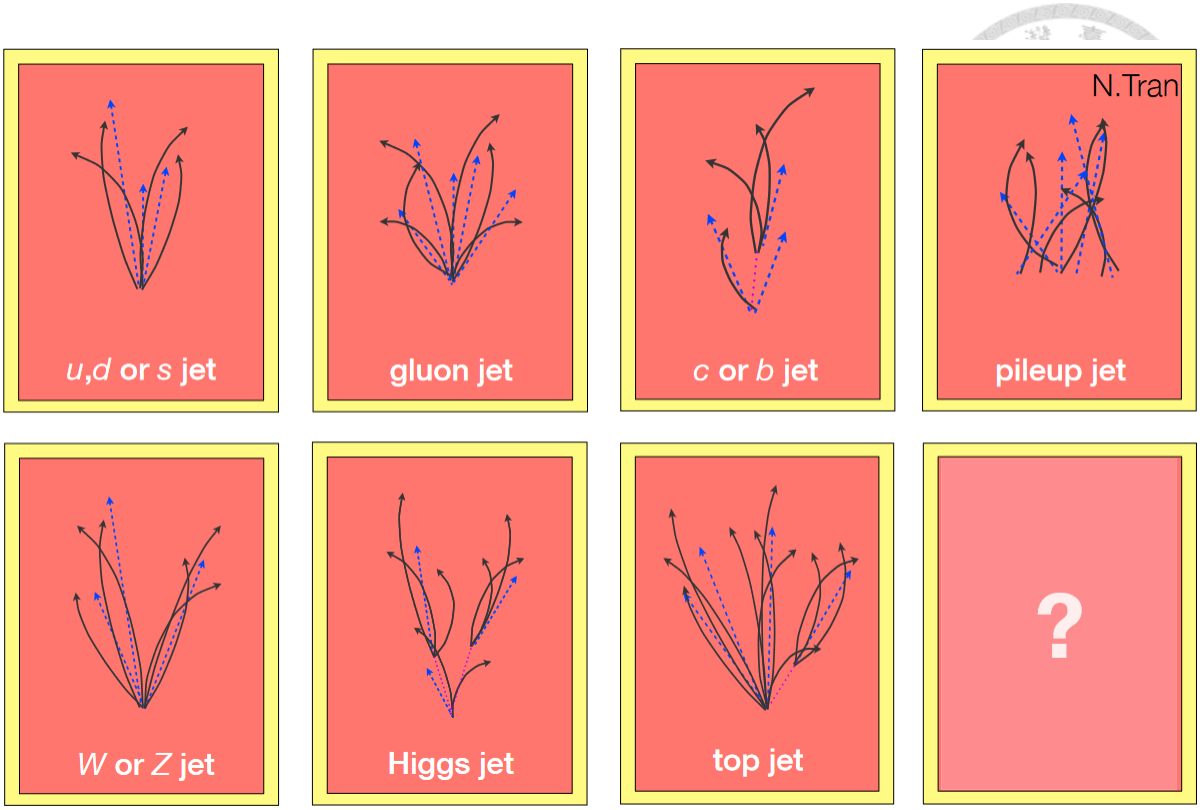


Figure 2.3: An illustration of different types of jets produced in high-energy collisions, adopted from [33]. The figure shows the different types of jets, including gluon jets, heavy quark jets, and light quark jets.

2.2 Cylindrical Coordinate System

In high-energy collider experiments, the momentum of a particle is commonly expressed in Cartesian coordinates as $\mathbf{p} = (p_x, p_y, p_z)$, where p_x, p_y, and p_z denote the components along the orthogonal axes. While Cartesian coordinates are suitable for theoretical formulations, they are often suboptimal for experimental applications due to the geometry of detectors, such as those at the LHC, which exhibit cylindrical symmetry about the beamline.

To better exploit the symmetry and practical layout of modern detectors, a cylindrical coordinate system is employed, characterized by the pseudorapidity η, the azimuthal angle ϕ, and the transverse momentum p_T, defined by:

$$\eta = -\ln\left(\frac{|\mathbf{p}| + p_z}{|\mathbf{p}| - p_z}\right), \tag{2.2}$$

$$\phi = \arctan\left(\frac{p_y}{p_x}\right), \tag{2.3}$$

$$p_T = \sqrt{p_x^2 + p_y^2}, \tag{2.4}$$

doi:10.6342/NTU202501658

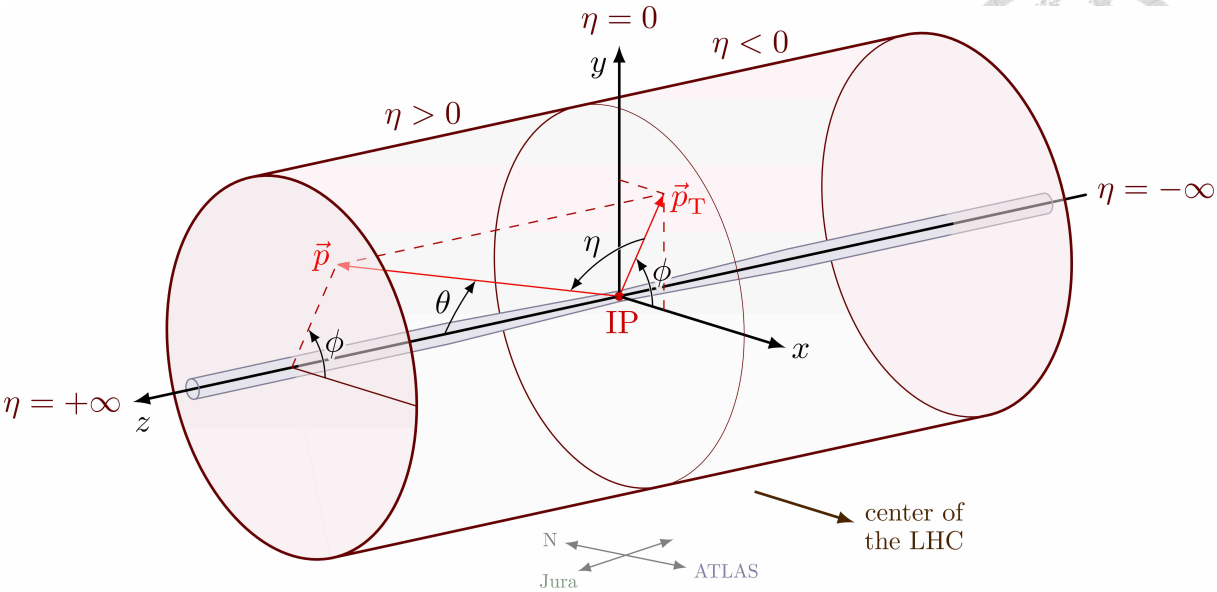


Figure 2.4: Illustration of the cylindrical coordinate system used in the CMS detector, adapted from [34]. The longitudinal axis of the is set to be the direction of the beam axis, which is aligned with the z-axis. The polar angle θ is defined as the difference between the particle track and the positive z-direction. The pseudorapidity η is defined as $\eta = -\ln(\tan(\theta/2))$, providing a measure of the angle relative to the beam axis. The IP in the center denotes the interaction point. The azimuthal angle ϕ is defined in the transverse plane, where the x and y direction can be set arbitrarily.

where $|\mathbf{p}| = \sqrt{p_x^2 + p_y^2 + p_z^2}$ is the magnitude of the momentum vector, with the z-direction defined as the beam axis. The transverse momentum p_T represents the momentum component perpendicular to the beam axis and is crucial in identifying events with missing energy or transverse imbalance. The angle ϕ describes the particle's azimuthal orientation in the transverse plane, while η serves as a logarithmic measure of the particle's angle relative to the beamline.

Importantly, these cylindrical variables exhibit distinct transformation behaviors: p_T remains invariant under both rotations about and boosts along the z-axis; ϕ is invariant under longitudinal boosts but changes under rotations respect to the beam axis; and η is invariant under azimuthal rotations but not under longitudinal boosts.

Figure 2.4 depicts the cylindrical coordinate system centered at the interaction point, defined as the spatial location where the proton-proton collision is presumed to occur. The polar angle θ is measured from the positive z-axis and the corresponding particle track. The xy-plane lies transverse to the beam, and the beam axis is in the z-direction. The pseudorapidity can be related to θ through the identity:

$$\eta = -\ln\left(\frac{|\mathbf{p}| + p_z}{|\mathbf{p}| - p_z}\right) = -\ln\left(\frac{1+\cos\theta}{1-\cos\theta}\right) = -\ln\left[\tan\left(\frac{\theta}{2}\right)\right]. \tag{2.5}$$

In summary, the adoption of cylindrical coordinates in collider physics is a reflection of

doi:10.6342/NTU202501658

both the symmetry of the detector geometry and the need for variables that behave predictably under Lorentz transformations. These coordinates provide a natural and powerful framework for characterizing the kinematics of particles produced in high-energy collisions.

2.3 Anti-k_T Algorithm

In particle detectors, such as the CMS experiment, only particle-level information is directly accessible. Consequently, algorithms are required to cluster individual particles into jets. The anti-k_T algorithm [35] is one of the most widely used jet clustering algorithms in high-energy physics, particularly at the LHC. Its primary function is to efficiently identify and reconstruct jets from the particles produced in high-energy collisions.

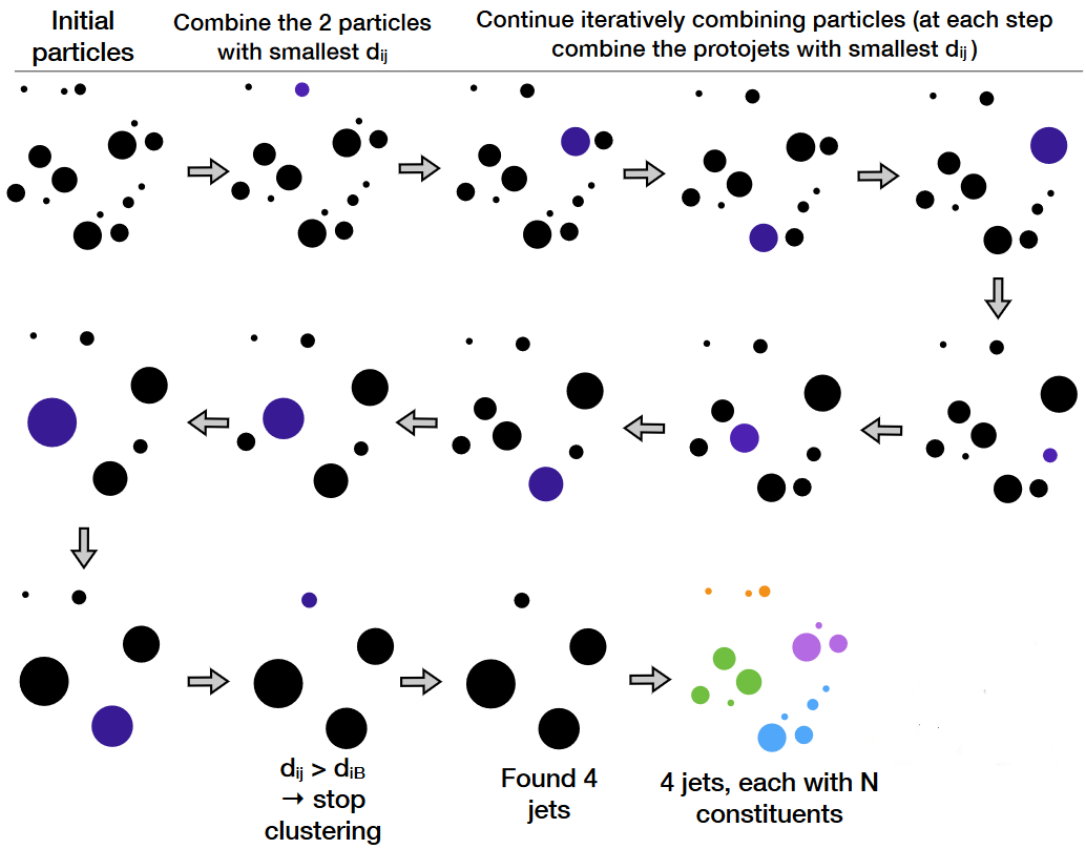


Figure 2.5: This figure illustrates the jet clustering process, adapted from [36]. Particles are clustered into jets iteratively, with the algorithm successively merging the closest particles or pseudo-jets (intermediate jet candidates) based on a distance measure. This process continues until all particles are assigned to jets.

The algorithm clusters particles into jets by comparing pairwise distances between particles (i, j) as well as the distance between each particle and the beam axis (denoted by B). In the

doi:10.6342/NTU202501658

anti-k_T algorithm, these distance measures are:

$$d_{ij} = \min\left(\frac{1}{p_{T,i}^2}, \frac{1}{p_{T,j}^2}\right)\frac{\Delta_{ij}^2}{R^2}, \tag{2.6}$$

$$d_{iB} = \frac{1}{p_{T,i}^2}, \tag{2.7}$$

where $p_{T,i}$ represents the transverse momentum of the i-th particle, and

$$\Delta_{ij} = \sqrt{(\phi_i - \phi_j)^2 + (\eta_i - \eta_j)^2} \tag{2.8}$$

is the distance between two particles in (η, ϕ) space, where the particle indices are denoted with i and j. The parameter R is the clustering radius, where smaller values of R lead to more compact jets, and larger values of R result in jets with a greater number of constituent particles. In the CMS experiment, typical values for R are 0.4 and 0.8 for analyses conducted at center-of-mass energy of $\sqrt{s} = 13$ TeV.

The anti-k_T algorithm operates iteratively, as depicted in Figure 2.5. Initially, each particle is treated as an individual jet candidate. The distances between all particle pairs and between each particle and the beam axis are calculated. The two closest objects, whether particles or pseudo-jets, are then merged into a single jet. If the smallest distance corresponds to a particle and the beam axis, the particle is labeled as a finalized isolated jet. This iterative process continues until a satisfactory jet configuration is achieved. The distance measure ensures that particles with higher transverse momentum are clustered first, promoting the formation of more physically meaningful jets.

2.4 Particle Flow Features

Once the particles produced in a collision are grouped together to form jets via a jet clustering algorithm, the momentum of each jet is computed by the summation of the 4-momentum of all the constituent particles. The properties of jets can thus be described in cylindrical coordinates, including the azimuthal angle ϕ_{jet}, pseudorapidity η_{jet}, and transverse momentum p_T^{jet}. These features provide an efficient means of characterizing the jet in HEP experiments.

In such experiments, the interaction point on the beam axis in the lab frame, where the particles collide, does not necessarily happen to be in the center of mass (COM) frame of the parton-level interaction. This discrepancy arises from the stochastic nature of parton momentum fractions, as described by parton distribution functions (PDFs). Since the protons are collided at high energies, the laboratory frame can be Lorentz-boosted along the z-direction to transform

doi:10.6342/NTU202501658

into the COM frame. While the azimuthal angle and the transverse momentum of the jet are invariant under this Lorentz z-boost, the pseudorapidity can be significantly affected by the frame mismatch.

On the other hand, rapidity, another quantity used to describe the motion of particles in high-energy collisions. A particle with energy E has the rapidity:

$$y = \frac{1}{2} \ln \left(\frac{E + p_z}{E - p_z} \right). \tag{2.9}$$

Although rapidity is not invariant under a Lorentz z-boost, the difference between the rapidities of two objects is invariant under such a boost, i.e.,

$$y_i - y_j = y_i' - y_j' \tag{2.10}$$

in two frames related by a z-direction Lorentz boost. This property ensures that jet analyses remain consistent across different reference frames, which is essential in high-energy collisions, where the center-of-mass frame may not align with the laboratory frame.

In practice, pseudorapidity η, which is defined as

$$\eta = \frac{1}{2} \ln \left(\frac{|\mathbf{p}| + p_z}{|\mathbf{p}| - p_z} \right) = - \ln \left(\tan \frac{\theta}{2} \right), \tag{2.11}$$

, where θ is the angle between the particle tracks and the beam axis, is often preferred over rapidity y in HEP experiments. This preference arises because the energy E of particles is not always directly measured with high precision. Additionally, pseudorapidity can be approximated as rapidity in the ultra-relativistic limit, where the particle's mass is negligible compared to its momentum. This approximation holds for most particles produced in high-energy collisions, where their energies are significantly larger than their rest masses. Under this approximation ($E \approx |\mathbf{p}|$), the difference in pseudorapidity is Lorentz invariant under z-direction boosts.

Within jet analyses, individual particle contributions to a jet are characterized using particle flow features, which provide insights into the distribution and behavior of particles within the jet. These features are defined as follows:

$$\begin{aligned} z_i &= \frac{p_{T,i}}{p_T^{\text{jet}}}, \\ \Delta\eta_i &= \eta_i - \eta_{\text{jet}}, \\ \Delta\phi_i &= \phi_i - \phi_{\text{jet}}. \end{aligned} \tag{2.12}$$

Here, z_i represents the fraction of the transverse momentum of jet carried by the i-th particle. This feature is essential for understanding how each particle contributes to the overall energy

doi:10.6342/NTU202501658

flow within the jet. The quantities $\Delta\eta_i$ and $\Delta\phi_i$ represent the differences in pseudorapidity and azimuthal angle, respectively, between axis of jet and the i-th particle. Specifically, the differences $\Delta\eta_i$ and $\Delta\phi_i$ quantify how far each particle is from the jet's central direction in the η and ϕ planes. Figure 2.6 shows samples of jets represented in the 2D $(\Delta\eta, \Delta\phi)$ plane.

These particle flow features provide physical understanding into the structure of jets, facilitating the discrimination between different types of jets (e.g., light quark jets, gluon jets, and jets originating from heavy flavor particles like b and c quarks). Furthermore, these features play a crucial role in jet tagging algorithms, which are used to determine the partonic origin of jet. The invariance of these quantities under specific Lorentz transformations enhances their applicability in various experimental analyses, ensuring consistency and reliability when characterizing jets in particle detectors, e.g., CMS and ATLAS at the LHC.

In conclusion, particle flow features such as z_i, $\Delta\eta_i$, and $\Delta\phi_i$ are essential tools in the analysis of jet structure and dynamics. By incorporating these features, we ensure that our jet reconstruction and analysis methods are independent of reference frames, enabling us to distinguish between different types of jets based on their underlying partonic origins.

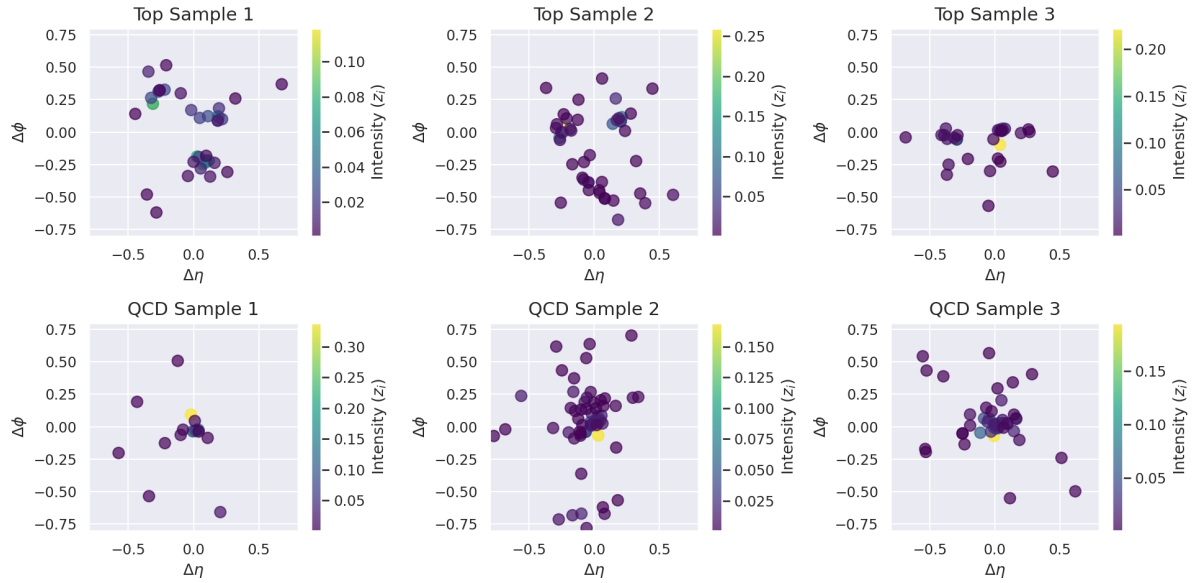


Figure 2.6: Samples of jets represented in the 2D $(\Delta\eta, \Delta\phi)$ plane. The first row shows three samples of top quark jets (denoted as Top), while the second row presents three samples light quark and gluon jets (denoted as QCD). The samples are selected from the public dataset [37]. In each image, the x and y-axis represent the differences to the jet of particles in pseudorapidity $(\Delta\eta)$ and azimuthal angle $(\Delta\phi)$, as defined in Equation 2.12, respectively. The color of each point corresponds to the ratio of the p_T of particles and their corresponding jet described in Equation 2.12, denoted as z_i. This representation provides a visualization of the spatial distribution of jets in the $(\Delta\eta, \Delta\phi)$ plane, highlighting their momentum characteristics and spatial arrangement.

doi:10.6342/NTU202501658

doi:10.6342/NTU202501658

Chapter 3

Classical Machine Learning

To compare with quantum models, we provide an overview on applications of deep learning in jet discrimination in this chapter, focusing on the challenges posed by the undetermistic number and types of particles inside jets. We discuss different representations of jets, including image-based, sequence-based, tree-based, graph-based, and set-based approaches. Each representation has its own advantages and limitations, and we highlight the importance of selecting a suitable representation based on the specific task at hand.

We discuss the classical models based on graph or set representations that are widely used in jet discrimination tasks, and introduce the general architecture of graph neural networks, which are designed to learn from set-based data while preserving permutation invariance. Furthermore, we then present the Particle Flow Network (PFN), Particle Net (PNet), and Particle Transformer (ParT) architectures, which leverage graph representations and message-passing mechanisms to capture the relationships among jet constituents. These models serve as classical benchmarks for comparison with our quantum model.

3.1 An Overview of Applications in Jet Discrimination

Deep neural networks (DNNs) have gained widespread attention due to their flexibility in architecture and remarkable capability for capturing intricate patterns, making them increasingly prominent in high-energy physics [5–7]. DNNs have demonstrated substantial success in addressing challenges such as particle classification, event reconstruction, and jet discrimination. Nevertheless, designing an efficient DNN model specifically tailored for jet discrimination remains challenging. This difficulty arises primarily from the inherently variable number and types of constituent particles within jets, as jets do not exhibit uniform structures, and each constituent particle contributes uniquely to the overall jet characteristics. To address these com-

doi:10.6342/NTU202501658

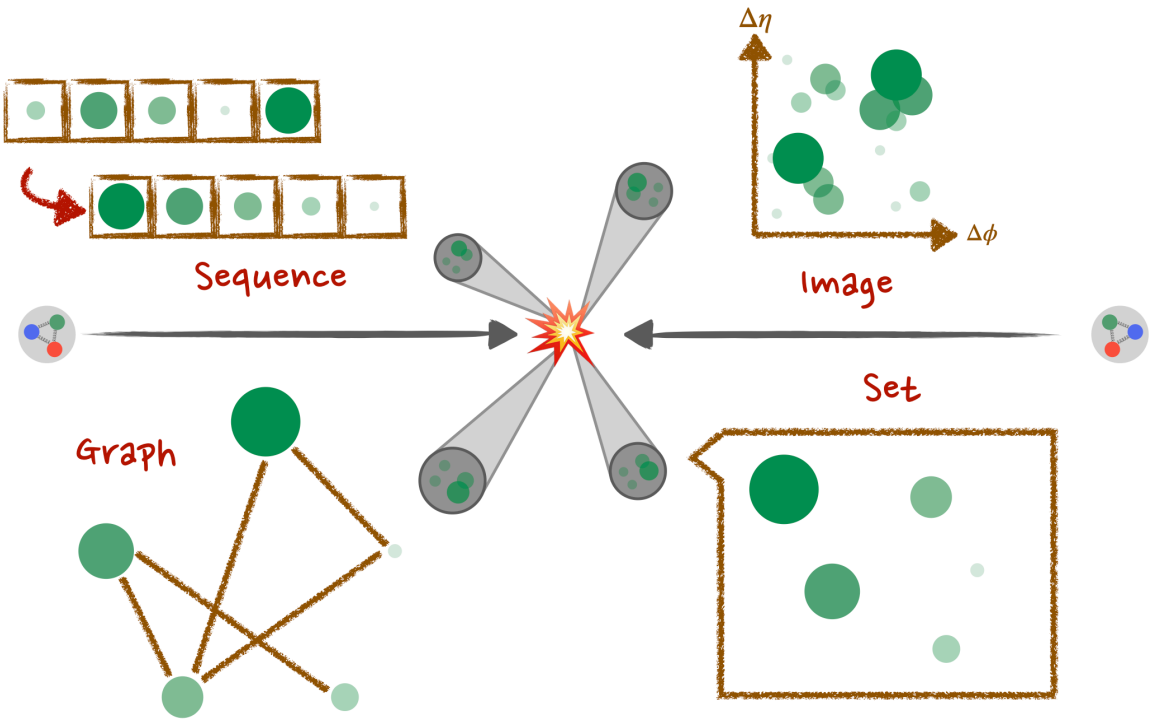


Figure 3.1: The representations that frequently used for jets, including sequences (top-left), images (top-right), graphs (bottom-left), and sets (bottom-right).

plexities, multiple data representations and corresponding deep learning architectures have been proposed, with the aim of effectively encoding jet information into forms suitable for DNN inputs. Figure 3.1 shows different representations commonly chosen for jets.

One prevalent method represents jets as two-dimensional images, where jet constituents are projected onto a grid whose axes typically correspond to physically meaningful quantities, such as pseudorapidity η and azimuthal angle ϕ. Each grid cell encodes particle properties such as energy or transverse momentum within its corresponding spatial region. This approach leverages well-established convolutional neural network (CNN) architectures that have proven highly effective at extracting spatial features from image-based data [38–45]. Despite notable successes, jet images have limitations, including potential loss of precise particle momentum and positional information. Additionally, image representations for jets also face the problem of sparsity when increasing the resolution of the grid, as many pixels may remain empty due to the limited particle number inside jets.

Alternative representations utilize sequences [46–52] or trees [53, 54], ordering jet constituents according to specific criteria. In sequential representations, particles are often arranged by decreasing transverse momentum, allowing the model to capture dominant particles first. Tree-based structures, conversely, aim to encode the hierarchical relationships, with each node

doi:10.6342/NTU202501658

representing individual particles and edges denoting parent-child connections. While sequence and tree structures facilitate capturing certain relational aspects among particles, they inherently introduce ordering biases instead of retaining the property of permutation invariance, and limiting generalizability across jets with varying particle arrangements.

Graph-based approaches have increasingly been adopted due to their capability in capturing particle relationships while maintaining permutation invariance [8–21]. Here, jet constituents are represented as nodes in a graph, and edges encode relational properties such as spatial distances, energy correlations, or other physically relevant metrics. This method offers significant flexibility in modeling jet substructure, enabling sophisticated representation of particle correlations. Graph neural networks (GNNs) are designed for learning this representation, and have shown particular promise in effectively learning complex particle interdependencies. Nonetheless, graph representations introduce computational challenges, especially in scenarios involving large particle multiplicities and complicated relational structures, potentially requiring advanced network architectures and optimization strategies. Also, defining the edges between particles can be nontrivial, as it may depend on the specific physics context or the task at hand.

Another prominent representation treats jets as sets of particles, viewing jet constituents as unordered collections of particle properties [55–64]. Set-based representations naturally exhibit permutation invariance, a desirable characteristic for modeling jets with variable particle numbers and arrangements. By encoding particle information without imposing explicit ordering constraints, set-based models can flexibly learn particle correlations and effectively manage jets of varying sizes. Similar to graph representations, however, set-based models also necessitate complex architectures and high computational costs, particularly when dealing with large particle multiplicities.

Despite extensive advancements, each jet representation brings different challenges and trade-offs. Jet images, while benefiting from straightforward application of CNNs, risk information loss due to fixed discretization and inherent symmetries. Sequence and tree-based methods impose artificial ordering constraints, potentially obscuring essential physical properties due to loss of permutation invariance. On the other hand, graph and set-based approaches provide enhanced flexibility in capturing detailed particle correlations and remaining permutation-invariant, but they introduce additional complexity in terms of computational resources and model design. Consequently, selecting a suitable representation and corresponding model requires careful consideration of task-specific demands, available computational resources, and desired balance between representational accuracy and model complexity.

In Chapter 5, we give an introduction to a brand new quantum neural network architecture, QCGNN, that employs a complete graph representation, where all particles within a jet are fully connected with each other. Accordingly, this chapter focuses on the application of graph-based

doi:10.6342/NTU202501658

representations and various GNN architectures used for benchmarks. In addition, we discuss the necessary and sufficient conditions for a model to have the property of permutation-invariance over a set.

3.2 Graph Representation

Graphs provide a powerful and flexible abstraction for modeling systems characterized by complex pairwise relationships. They appear ubiquitously across diverse scientific and practical fields, such as social networks, biological systems, knowledge graphs, and recommendation systems. The intrinsic structure of graph data enables representation of entities as nodes and their mutual interactions as edges. Extracting information from the graph and designing an efficient algorithm are fundamental challenges in GNNs.

Formally, a graph G consists of an ordered pair $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents the collection of nodes (also called vertices), and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ specifies the edge set. Let $N = |\mathcal{V}|$ represent the number of nodes. An edge connecting two nodes i and j is denoted by E_{ij}. Typically, an undirected graph implies symmetry in the edge set, i.e., $E_{ij} = E_{ji}$. Moreover, many practical scenarios assume graphs to be unweighted, assigning equal importance to each edge by neglecting numerical weights or attributes. In this dissertation, we focus on undirected, unweighted graphs if not specified otherwise.

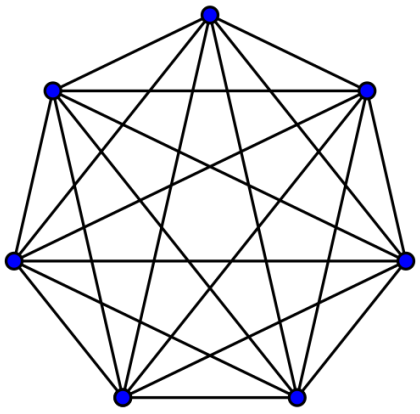


Figure 3.2: Illustration of a complete graph with 7 nodes, where each node is directly connected to every other node. Adapted from [65].

A graph is said to be *complete* if its every distinct node pair is connected by an edge, formally expressed as $E_{ij} \in \mathcal{E}$ for all $i, j \in \mathcal{V}$ with $i \neq j$. An example of a complete graph is shown in Figure 3.2. For undirected, unweighted complete graphs, the graph structure is fully characterized by nodes and is mathematically equivalent to a set. In this special case, the complete graph is used to emphasize the pairwise relationships among all elements.

doi:10.6342/NTU202501658

Each node $i \in \mathcal{V}$ is has a vector $\mathbf{x}_i \in \mathbb{R}^d$ that corresponds to their feature, where d denotes the size of the feature space. These vectors contain information that describes the corresponding entities. In machine learning applications, node features act as the initial representations used by models to carry out various downstream tasks, including classifying nodes, predicting connections, and making decisions at the level of entire graphs.

Understanding and effectively utilizing the structural properties of graphs, such as node features and connectivity patterns, remains a key area of ongoing research. The combination of discrete structure and continuous feature representations makes graph learning both conceptually rich and technically demanding, requiring continued development of innovative models and computational techniques.

3.3 The Deep Set Theorem

In many machine learning tasks, particularly those involving sets or graphs, the input data commonly manifests as unordered collections of elements. In these scenarios, models must inherently respect the symmetry of the data, ensuring that their outputs are invariant under permutations of the input elements. This property, known as *permutation invariance*, is crucial for neural networks designed to handle sets or graphs since reordering the elements should not affect the predictions or representations produced by the network. This concept extends naturally from sets to graphs, where node permutations should leave the essential properties and structures unchanged.

The *Deep Set Theorem* [66] provides a rigorous mathematical foundation for permutation-invariant functions defined over sets, thus guiding the construction of deep learning models suitable for set-based data (the neighbors of a graph's node also form a set). This theorem has widespread applicability across inputs inherently lack meaningful ordering.

Formally, consider a set $X = \{x_1, x_2, \ldots, x_N\}$, where each element $x_i \in X$ denotes an individual item, and N represents the set's cardinality. Such a set X could represent diverse contexts, including the neighborhood of a node within a graph, where node features are treated as elements of the set. A function f is considered permutation-invariant with respect to X if it satisfies the condition:

$$f(\{x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(N)}\}) = f(\{x_1, x_2, \ldots, x_N\}), \tag{3.1}$$

for any permutation π of the indices $\{1, 2, \ldots, N\}$. This criterion implies that the output should be unaffected by arbitrary rearrangement of the elements, ensuring the model's output depends solely on the intrinsic properties of the set rather than on any imposed ordering.

doi:10.6342/NTU202501658

The Deep Set Theorem characterizes permutation-invariant functions by stating that a function f acting on sets could be written into a canonical form:

$$f(X) = g\left(\sum_{x \in X} h(x)\right), \tag{3.2}$$

with some appropriate transformations or neural network modules denoted as g and h. Specifically, the function h independently transforms each element $x \in X$, after which the resulting representations are summed, generating a global set embedding. Subsequently, the function g acts upon this aggregated embedding to yield the final output. This formulation inherently ensures permutation invariance, as the summation operation is commutative, and the subsequent transformation via g does not impose any ordering constraints.

The Deep Set Theorem serves as a foundational principle in the design of neural network architectures for processing set-based or graph-based data, ensuring that learned representations inherently respect permutation invariance. An illustrative application is jet discrimination in particle physics, where the input data is a set of particles characterized by various features. Since the physical identity and jets' properties do not depend on any order in which its constituent particles are presented, the classification must also be insensitive to such permutations. Thus, employing models inspired by the Deep Set Theorem is essential to accurately capture invariant properties and produce robust predictions.

In summary, the Deep Set Theorem provides a critical theoretical framework enabling the construction of neural networks tailored for unordered data, guaranteeing that learned models maintain the essential property of permutation invariance required across many practical applications.

doi:10.6342/NTU202501658

3.4 Message-Passing Graph Neural Network

Motivated by the Deep Set Theorem, Message-Passing Graph Neural Networks (MPGNNs) represent a general and flexible neural network architecture that preserves permutation invariance. Formally, MPGNNs can be mathematically formulated as follows:

$$\mathbf{x}_i^{(l)} = \gamma^{(l)} \left[\mathbf{x}_i^{(l-1)}, \bigoplus_{j \in \mathcal{N}(i)} \Phi^{(l)} \left(\mathbf{x}_i^{(l-1)}, \mathbf{x}_j^{(l-1)} \right) \right], \tag{3.3}$$

where $\Phi^{(l)}$ aggregates information from neighboring nodes $\mathcal{N}(i)$, and $\gamma^{(l)}$ updates node features at iteration l. Including the original **SUM**, MPGNN generalizes the $\sum$ in Equation 3.2 by employing various aggregation methods (denoted as $\oplus$), such as, **MEAN**, **MAX**, and **MIN**. These aggregation methods can be formally expressed through suitable transformations which can be absorbed into γ and Φ. For example, the **MAX** function can be represented using the limit of the infinite norm:

$$\mathbf{MAX}(X) = \lim_{p \to \infty} \left(\sum_{x \in X} x^p \right)^{\frac{1}{p}}. \tag{3.4}$$

Here, the set X is assumed to be positive real numbers. This assumption does not lose generality, as the **MAX** function can be transformed into a positive domain by simply a shift, or through a suitable transformation such as $x \to e^x$, which is a monotonic function. Either way can be absorbed into γ and Φ. Similarly, **MIN** can be expressed with an additional transformation $x \to \frac{1}{x}$, which can also be absorbed into γ and Φ.

In short, the MPGNN has four steps:

1. Message passing: computes the pairwise information between neighbor particles.
2. Node aggregation: aggregates the correlations from neighbor particles.
3. Repeat steps 1 & 2 (optional).
4. Graph aggregation: final aggregation over nodes in the graph.

The procedures above are also depicted in Figure 3.3.

In the following subsections, we discuss prominent variants of MPGNN architectures widely employed in high-energy physics, particularly jet discrimination tasks, including the Particle Flow Network [55], Particle Net [8], and Particle Transformer [56]. These models serve as classical benchmarks in comparison to the QCGNN approach discussed later in Chapter 6.

doi:10.6342/NTU202501658

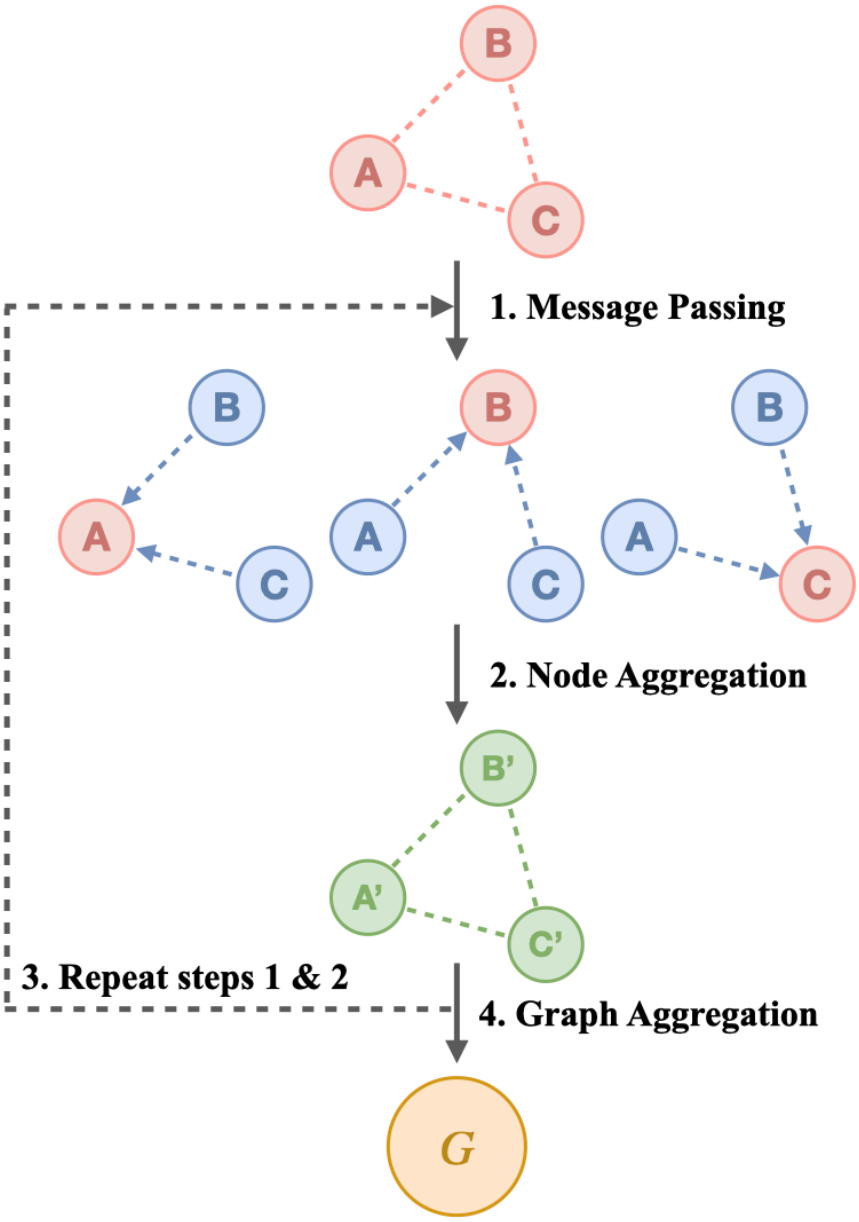


Figure 3.3: A high-level overview of the procedures of MPGNN with an example of 3-particle jet represented as a complete graph. Including (1) Message passing: computes the pairwise information between neighbor particles. (2) Node aggregation: aggregates the correlations from neighbor particles. (3) Repeat steps 1 & 2 (optional). (4) Graph aggregation: final aggregation over nodes in the graph.

doi:10.6342/NTU202501658

3.4.1 Particle Flow Network (PFN)

The Particle Flow Network (PFN) [55], illustrated in Figure 3.4, leverages the set representation of jet constituents. Each particle's feature vector is individually processed through a transformation Φ, aggregated using a summation operation (**SUM**), and finally passed through another transformation γ to yield the classification score F. In practice, both Φ and γ are implemented with neural networks, typically consisting of multiple fully connected layers. Mathematically, we can express it in the form of:

$$F = \gamma \left[\sum_{\mathbf{x}_i \in G} \Phi(\mathbf{x}_i) \right] . \tag{3.5}$$

The PFN architecture is designed to be permutation-invariant, as the summation operation ensures that the order of particles will not affect the model output.

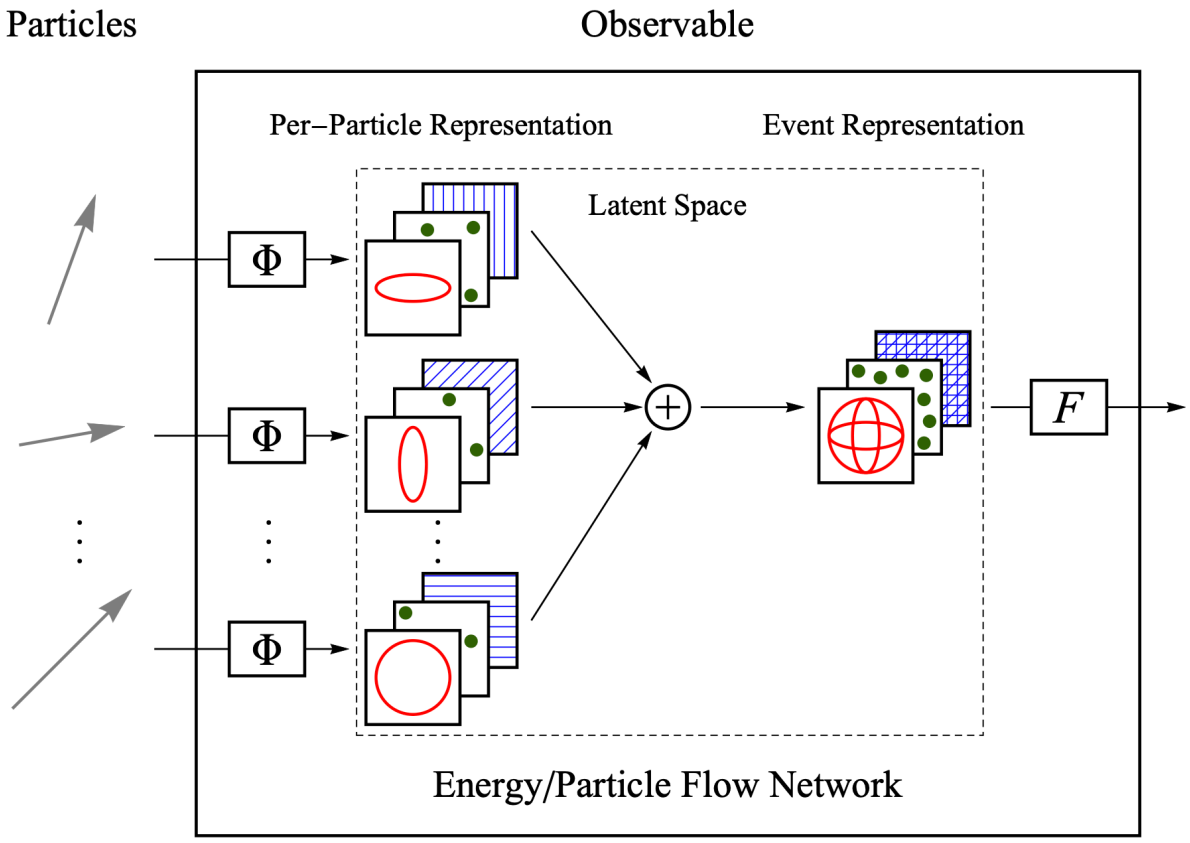


Figure 3.4: Architecture of the Particle Flow Network (PFN), adapted from [55].

doi:10.6342/NTU202501658

3.4.2 Particle Net (PNet)

The Particle Net (PNet) [8], inspired by the dynamical GNN proposed in [67], utilizes EdgeConv blocks to manage computational complexity associated with graph representations of jets. These blocks define fixed-size graphs using the k nearest neighbors (k-NN), as depicted in Figure 3.5b, where k is the number of maximum edges to be constructed for each node. The higher the k value, the more edges are created, leading to a denser graph and more computational costs. The initial metric used for calculating the distances between nodes are computed based on coordinates in (η, ϕ) space:

$$D_{ij} = \sqrt{(\Delta\phi_i - \Delta\phi_j)^2 + (\Delta\eta_i - \Delta\eta_j)^2}. \tag{3.6}$$

After the first iteration, the subsequent metrics used in latent feature space is the Euclidean norm:

$$D_{ij} = ||\mathbf{x}_i^{(l)} - \mathbf{x}_j^{(l)}||_2. \tag{3.7}$$

In the EdgeConv formulation, feature updates at each layer follow:

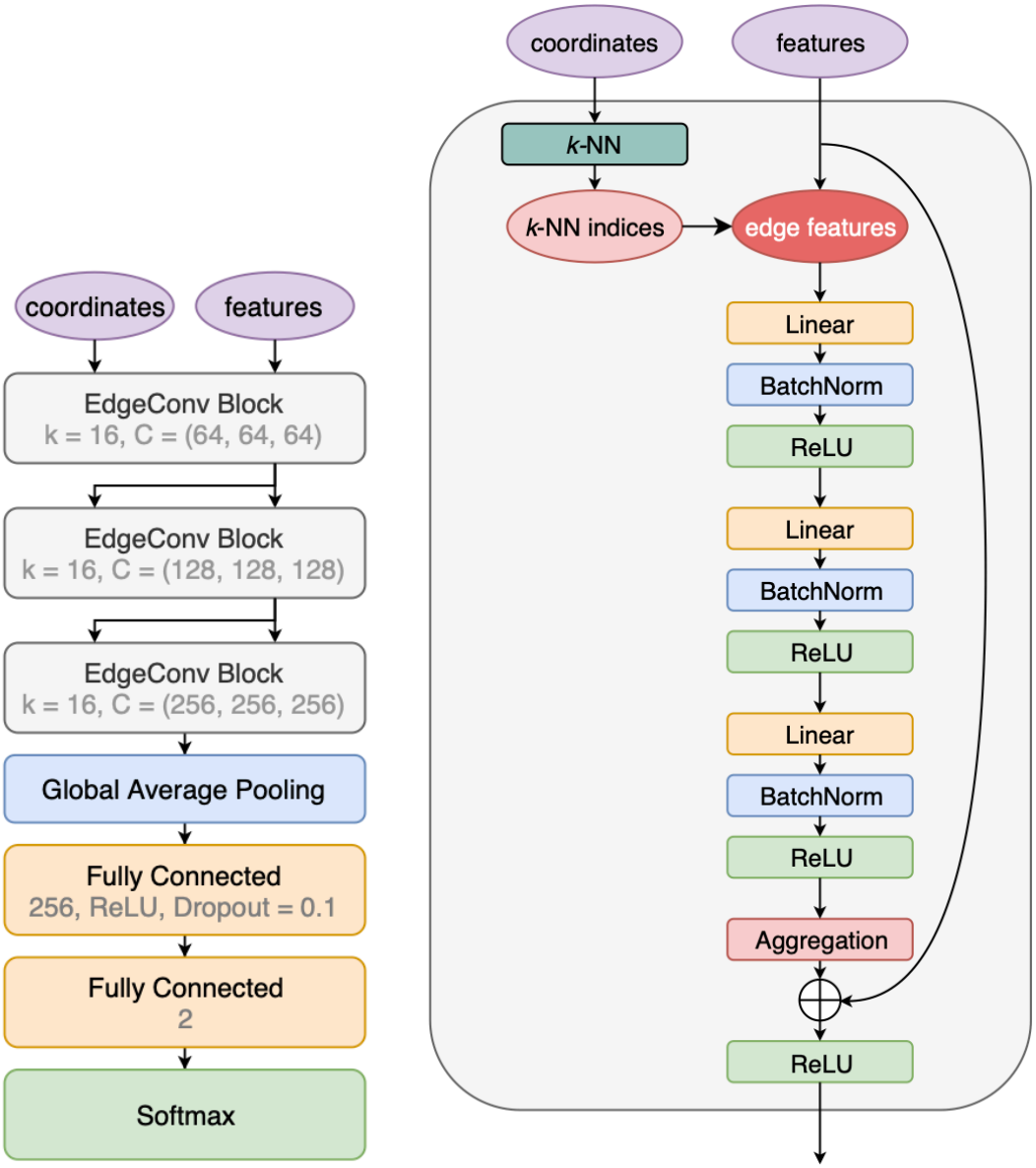
$$\mathbf{x}_i^{(l)} = \gamma^{(l)} \left[\bigoplus_{j \in \mathcal{N}_k^l(i)} \Phi^{(l)} \left(\mathbf{x}_i^{(l-1)}, \mathbf{x}_i^{(l-1)} - \mathbf{x}_j^{(l-1)} \right) \right], \tag{3.8}$$

where $\mathcal{N}_k^l(i)$ denotes the k nearest neighbors of i-th node at l-th layer. Typically, both Φ and γ are implemented with fully connected layers.

The EdgeConv block captures local features by aggregating information from neighboring nodes, effectively modeling the relationships among particles within a jet. Specifically, the dynamical graphs in PNet are directed, as the corresponding k nearest neighbors depend on nodes.

PNet aggregates node features via global mean pooling and then employs fully connected layers to produce the final prediction, as illustrated in Figure 3.5a. The dynamic k-NN and global pooling ensure permutation invariance within this model.

doi:10.6342/NTU202501658



(a) PNet Structure (b) EdgeConv Block

Figure 3.5: Particle Net (PNet) architecture, adapted from [8].

doi:10.6342/NTU202501658

3.4.3 Particle Transformer (ParT)

Particle Transformer (ParT) [56] employs Transformer architectures, which is the foundation that widely recognized in large language models nowadays. Without positional embeddings, the attention blocks in the Transformers inherently operate as permutation-invariant models resembling complete graphs. The attention mechanism computes relationships among all node pairs through query (Q), key (K), and value (V) transformations (typically via linear matrices):

$$\text{Attention}(Q, K, V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.9}$$

Here, Q, K, and V result from applying linear mappings to the input features, and d_k denotes the dimensionality of the projected latent space. The attention mechanism captures all pairwise relationships among particles, including local and global, allowing the model to learn complex interactions.

Since the attention mechanism computes interactions between all pairs of nodes, it inherently represents a complete graph structure. Moreover, the Softmax operation constitutes a valid aggregation function within this framework, as the exponential transformation and subsequent normalization can be integrated into the functions Φ and γ defined in Equation 3.3. The attention mechanism is rather complicated such that it is difficult to express it in the form of Equation 3.3. Although the attention mechanism is relatively complex and does not readily fit the formulation of Equation 3.3, it can still be viewed as a variant of message passing, for which each node exchanges mutual-information with every other node. This interpretation aligns closely with the fundamental concept of MPGNNs.

Specifically, ParT realizes the global pooling via a designated class token, then passing the aggregated information through multi-layer perceptrons and a activation function via Softmax to obtain the final classification score.

In summary, the attention mechanism and global pooling provide inherent permutation invariance to the ParT architecture, and the pairwise computations among all particle pairs allow capturing complex particle interactions.

doi:10.6342/NTU202501658

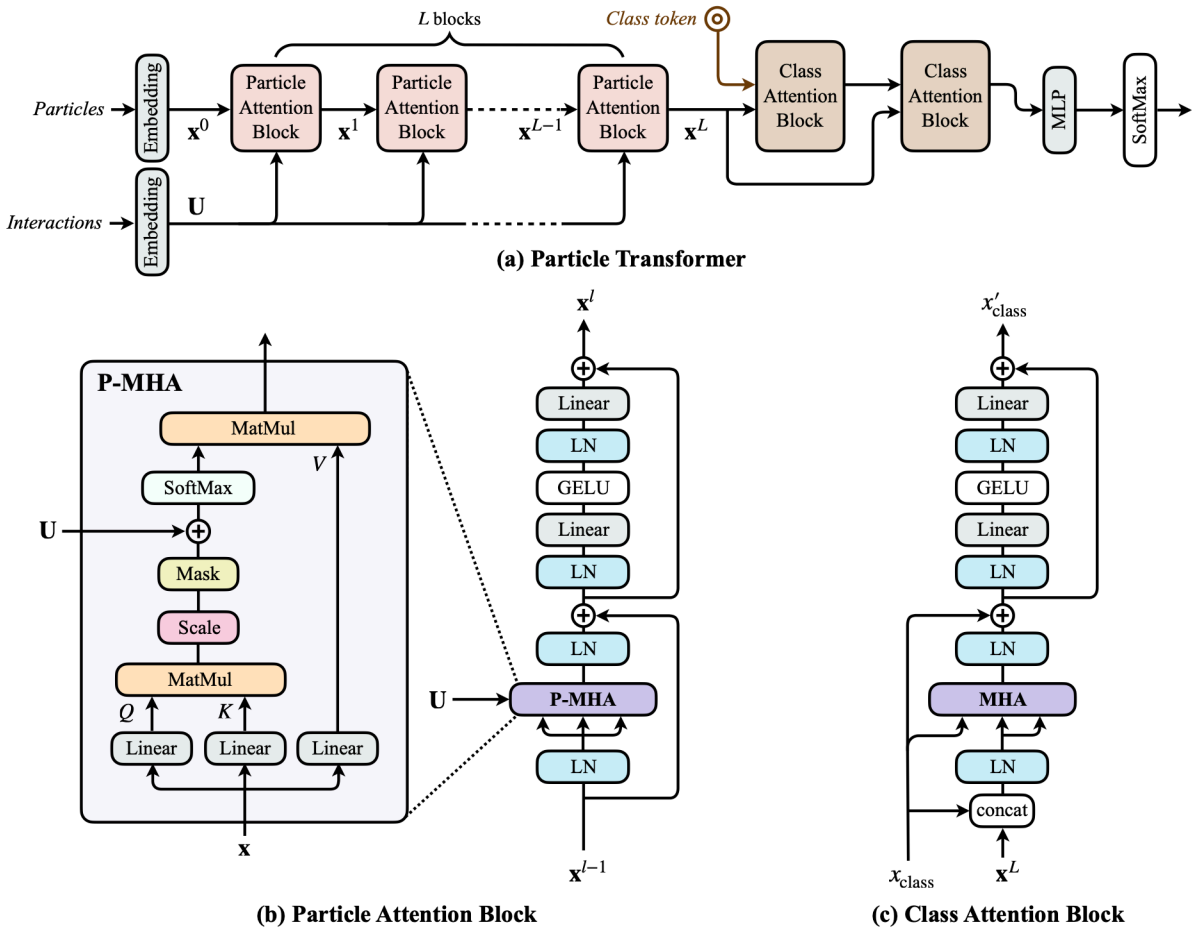


Figure 3.6: Particle Transformer (ParT) architecture, adapted from [56].

doi:10.6342/NTU202501658

doi:10.6342/NTU202501658

Chapter 4

Quantum Machine Learning

To properly introduce our model, QCGNN, it is essential first to establish foundational knowledge concerning quantum computing and quantum machine learning. This chapter aims to review fundamental principles of quantum computation, beginning with an overview of quantum bits (qubits), quantum operations and quantum gates. These concepts are the basis for understanding quantum information processing and computing. Additionally, we briefly cover the overview on the hardware of IBM quantum computers.

We subsequently focus on the Variational Quantum Circuit (VQC) framework, a hybrid quantum-classical approach for quantum neural networks. In this framework, classical optimization routines iteratively update quantum circuit parameters to minimize a predefined cost function, effectively integrating quantum computational power with classical optimization techniques. At the end of this chapter, we discuss the techniques and the concepts frequently used in constructing VQCs, which are also foundations of our QCGNN.

4.1 A Review on Fundamentals of Quantum Computating

4.1.1 Qubits and Quantum Entanglement

In quantum computation, the quantum bit, or qubit, serves as the elementary unit of quantum information. Unlike classical bits that only have two distinct states, 0 or 1, a qubit is able to lie in a linear superposition of those two states. Mathematically, the expression of the quantum state of a single qubit can be described as:

$$|\psi\rangle = a\,|0\rangle + b\,|1\rangle = a\begin{bmatrix}1\\0\end{bmatrix} + b\begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}a\\b\end{bmatrix}, \tag{4.1}$$

doi:10.6342/NTU202501658

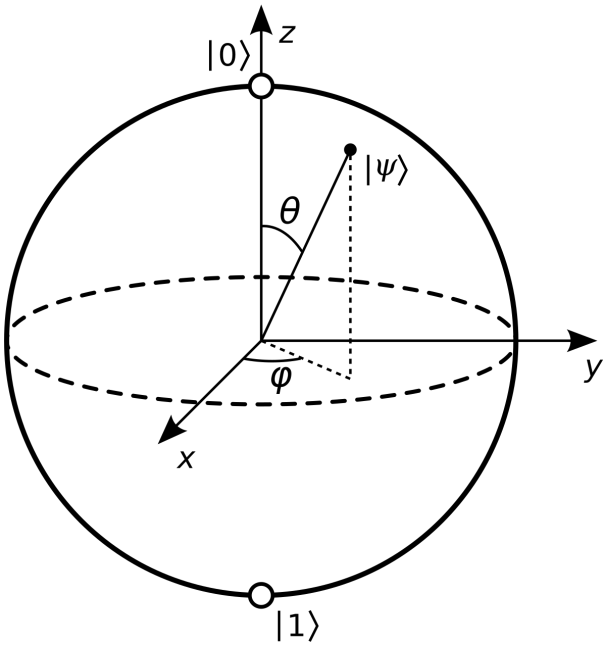


Figure 4.1: A visualization of a qubit on the Bloch sphere. The angles θ and ϕ define the qubit's position on the sphere. This figure is adapted from [68].

where $|0\rangle$ and $|1\rangle$ represent the computational states, and a and b are complex coefficients that determine the probability amplitudes for measuring the qubit in each of these states. The unitarity of the quantum state requires $|a|^2 + |b|^2 = 1$ to ensure that the sum of the probabilities of all measuring outcome will add up to one.

Figure 4.1 provides a visualization of a qubit, which the jargon calls it the Bloch sphere. In this representation, the qubit's state is characterized by two arguments, which are the azimuthal angle ϕ and the polar angle θ. The state of the qubit is usually written as:

$$|\psi\rangle = \cos\left(\frac{\theta}{2}\right)|0\rangle + e^{i\phi}\sin\left(\frac{\theta}{2}\right)|1\rangle\,, \tag{4.2}$$

where the global phase $e^{i\alpha}$ is omitted, as it does not influence the physical properties of the state.

For multi-qubit systems, the total state of the qubits can be expressed in the form of product of tensors that described each qubit state. For instance, consider two qubits in arbitrary states:

$$\begin{aligned} |\psi_1\rangle &= a_1|0\rangle + b_1|1\rangle\,, \\ |\psi_2\rangle &= a_2|0\rangle + b_2|1\rangle\,. \end{aligned} \tag{4.3}$$

The combined state of the two qubits is given by the tensor product:

$$\begin{aligned} |\psi\rangle &= |\psi_1\rangle \otimes |\psi_2\rangle \\ &= (a_1|0\rangle + b_1|1\rangle) \otimes (a_2|0\rangle + b_2|1\rangle) \\ &= a_1a_2|00\rangle + a_1b_2|01\rangle + b_1a_2|10\rangle + b_1b_2|11\rangle\,. \end{aligned} \tag{4.4}$$

doi:10.6342/NTU202501658

Here, $|ij\rangle$ denotes the tensor product $|i\rangle \otimes |j\rangle$. A general quantum state of two qubits can thus be expressed as a linear combination of the four computational basis:

$$|\psi\rangle = c_{00}\,|00\rangle + c_{01}\,|01\rangle + c_{10}\,|10\rangle + c_{11}\,|11\rangle\,, \tag{4.5}$$

where c_{ij} are coefficients with complex numbers corresponding the probability amplitudes for each of the basis states. A two-qubit system is considered entangled if the state can not be decomposed into a simple tensor product of single qubit states. E.g., $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ is an entangled state, whereas $|01\rangle = |0\rangle \otimes |1\rangle$ and $|10\rangle = |1\rangle \otimes |0\rangle$ are not entangled.

4.1.2 Unitary Transformations and the Quantum Gates

The time evolution of a quantum state, as stated by the principles of quantum mechanics, follows unitary transformations. These operations are described using unitary matrices, which ensure that the inner product is preserved, thereby maintaining the normalization of the quantum states. A unitary matrix U satisfies the relation:

$$U^\dagger U = I, \tag{4.6}$$

where $U^\dagger$ denotes the Hermitian conjugate (complex conjugate transpose) of U, and I stands for the identity matrix. When a unitary operator acts on a quantum state $|\psi\rangle$, the resulting expression becomes:

$$|\psi'\rangle = U\,|\psi\rangle\,, \tag{4.7}$$

where $|\psi'\rangle$ is the transformed state. Just like classical logic gates, quantum gates serve as essential building elements in quantum circuits. These gates modify qubit states and are mathematically expressed as unitary matrices operating on quantum state vectors.

In quantum computers, the unitary transformations are implemented through quantum gates, which are the basic components of quantum circuits. Quantum gate consists of single-qubit gates and multi-qubit gates. Single qubit gate is applied on one qubit at a time, while multi qubit gates operate on two or more qubits simultaneously. The quantum circuits are drawn as sequences of quantum gates operated on qubits. The quantum circuit diagram consists of horizontal lines representing qubits, and quantum gates are represented as boxes. The final state of the qubits is obtained via implementing the gates in sequence and is traditionally depicted to be operated from left to right. Here, we show some of the quantum gates that will be frequently used later.

- **Hadamard Gate (H):** The Hadamard gate creates superpositions of computational basis.

doi:10.6342/NTU202501658

Figure 4.2: The Hadamard gate depicted on the quantum circuit.

It is represented by the matrix:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{4.8}$$

The Hadamard gates transform the computational basis $|0\rangle$ and $|1\rangle$ as follows:

$$\begin{aligned} H|0\rangle &= \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), \\ H|1\rangle &= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle). \end{aligned} \tag{4.9}$$

- **SWAP Gate**: The SWAP gate is a two-qubit gate that swaps the states of two qubits. It

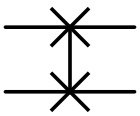


Figure 4.3: The SWAP gate depicted between two qubits.

is represented by the matrix:

$$\text{SWAP} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.10}$$

The SWAP gate transforms the basis states as follows:

$$\begin{aligned} \text{SWAP}|00\rangle &= |00\rangle, \\ \text{SWAP}|01\rangle &= |10\rangle, \\ \text{SWAP}|10\rangle &= |01\rangle, \\ \text{SWAP}|11\rangle &= |11\rangle. \end{aligned} \tag{4.11}$$

- **CNOT Gate (Controlled-NOT):** The CNOT gate is a two-qubit gate that is crucial for

doi:10.6342/NTU202501658

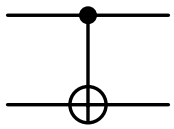


Figure 4.4: The CNOT gate, where the control qubit and the target qubit are depicted as the first and second qubit, respectively.

creating entanglement. It is represented by the matrix:

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{4.12}$$

The CNOT gate flips the state of the target qubit if the control qubit is in the state $|1\rangle$:

$$\begin{aligned} \text{CNOT}\,|00\rangle &= |00\rangle\,, \\ \text{CNOT}\,|01\rangle &= |01\rangle\,, \\ \text{CNOT}\,|10\rangle &= |11\rangle\,, \\ \text{CNOT}\,|11\rangle &= |10\rangle\,, \end{aligned} \tag{4.13}$$

where the first qubit is the control qubit and the second qubit is the target qubit in conventional notation. To get the representation where the first qubit is the target qubit and

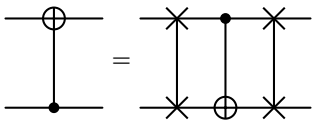


Figure 4.5: The CNOT gate, where the control qubit and the target qubit are depicted as the second and first qubit, respectively.

the second qubit is the control qubit, we can use the following transformation:

$$\text{CNOT} \rightarrow \text{SWAP} \cdot \text{CNOT} \cdot \text{SWAP}, \tag{4.14}$$

such that the CNOT gate undergoes a unitary transformation by the SWAP gates.

- **Rotation Gates (R_x, R_y, and R_z):** The rotation gates are single-qubit gates that apply a

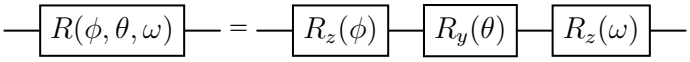


Figure 4.6: General single-qubit gate.

doi:10.6342/NTU202501658

rotation with the $SU(2)$ generators, also known as the Pauli matrices. The Pauli matrices are defined as:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \tag{4.15}$$

which are known as Pauli matrices. These rotation gates are single-qubit gates and are represented as follows:

$$\begin{aligned} R_x(\theta) \equiv e^{-i\theta X/2} &= \cos\left(\frac{\theta}{2}\right) I - i \sin\left(\frac{\theta}{2}\right) X = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) \\ -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix}, \\ R_y(\theta) \equiv e^{-i\theta Y/2} &= \cos\left(\frac{\theta}{2}\right) I - i \sin\left(\frac{\theta}{2}\right) Y = \begin{bmatrix} \cos\left(\frac{\theta}{2}\right) & -\sin\left(\frac{\theta}{2}\right) \\ \sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) \end{bmatrix}, \\ R_z(\theta) \equiv e^{-i\theta Z/2} &= \cos\left(\frac{\theta}{2}\right) I - i \sin\left(\frac{\theta}{2}\right) Z = \begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}. \end{aligned} \tag{4.16}$$

These rotation gates rotate the qubit state vector by an angle θ around the respective axis (X, Y, or Z) of the Bloch sphere. Any single-qubit gate can be expressed as a combination of these rotation gates with an additional global phase, which is not relevant to the final computation. Ignoring the global phase, any single-qubit gate can be decomposed as:

$$R(\phi, \theta, \omega) = R_z(\omega) R_y(\theta) R_z(\phi) = \begin{bmatrix} e^{-i(\phi+\omega)/2} \cos\left(\frac{\theta}{2}\right) & -e^{i(\phi-\omega)/2} \sin\left(\frac{\theta}{2}\right) \\ e^{-i(\phi-\omega)/2} \sin\left(\frac{\theta}{2}\right) & e^{i(\phi+\omega)/2} \cos\left(\frac{\theta}{2}\right) \end{bmatrix}. \tag{4.17}$$

4.1.3 Measurements and Observables

In quantum mechanics, measurement is the special operation of obtaining information from a quantum system. When measuring a qubit, it turns out to be one of its basis states of the observable with a probability corresponded to the coefficients of the basis, where the qubit is said to be collapsed. For example, if a qubit is in the state $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$ and measured with a Z observable, the probabilities of measuring $|0\rangle$ and $|1\rangle$ are given by $|\alpha|^2$ and $|\beta|^2$, respectively. Given an observable O, the expectation value of in the state $|\psi\rangle$ corresponding to observable O is given by:

$$\langle O \rangle = \langle \psi | O | \psi \rangle . \tag{4.18}$$

An observable must be Hermitian, meaning it has real eigenvalues and orthogonal eigen-

doi:10.6342/NTU202501658

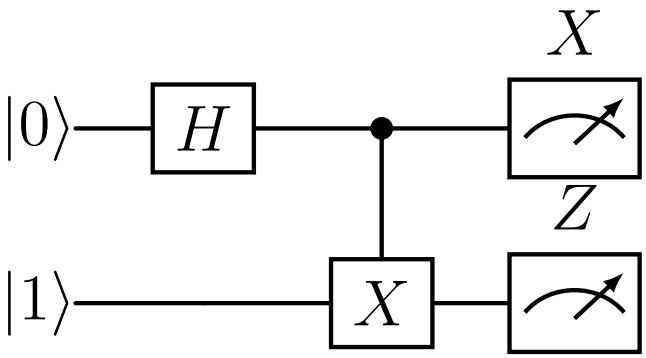


Figure 4.7: A Bell state $|\psi\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$ where the first qubit and the second qubit are measured in the X-basis and Z-basis, respectively. The calorimeter symbol stands for the measurement.

states. According to the spectral theorem, any Hermitian operator can be decomposed as linear combinations of its eigenstates:

$$O = \sum_i \lambda_i |\psi_i\rangle \langle\psi_i| , \tag{4.19}$$

where λ_i are the eigenvalues and $|\psi_i\rangle$ are the corresponding eigenstates. For example, the commonly used Pauli matrices can be decomposed as:

$$\begin{aligned} X &= \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = |1\rangle \langle 0| + |0\rangle \langle 1| , \\ Y &= \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = i |1\rangle \langle 0| - i |0\rangle \langle 1| , \\ Z &= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = |0\rangle \langle 0| - |1\rangle \langle 1| . \end{aligned} \tag{4.20}$$

Figure 4.7 shows an example of a two-qubit quantum circuit, where the final quantum state is a Bell state $|\psi\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$. The first qubit is measured in the X-basis, and the second qubit is measured in the Z-basis.

4.2 IBM Quantum Computers

Several physical platforms have been explored for the realization of quantum computation, each leveraging different physical phenomena to implement and manipulate qubits. These include trapped ions, which use the internal electronic states of ions confined by electromagnetic

doi:10.6342/NTU202501658

fields; neutral atoms in optical lattices; nitrogen-vacancy centers in diamond; nuclear magnetic resonance (NMR); and photonic systems using single photons and linear optics. Each of these platforms has distinct advantages and challenges in terms of scalability, coherence times, gate fidelity, and readout mechanisms.

IBM Quantum Computers [69] are based on superconducting qubits, which are macroscopic quantum systems engineered from electrical circuits. These systems operate at cryogenic temperatures and leverage the principles of circuit quantum electrodynamics (cQED) to control, manipulate, and measure quantum information. In this subsection, we give a brief introduction of the underlying physical architecture, Hamiltonian formulation, gate implementation, and readout mechanisms that enable quantum computation on the IBM quantum platform.

4.2.1 Superconducting Qubits and Circuit Hamiltonian

IBM's qubits, also known as the transmon qubits, are realized using nonlinear superconducting circuits, most notably based on variations of the LC (inductor-capacitor) oscillator. In the classical regime, the Hamiltonian of an ideal linear LC oscillator is given by:

$$H = \frac{1}{2}LI^2 + \frac{1}{2}CV^2, \tag{4.21}$$

where V is the voltage between the plates of the capacitor, and I is the current through the inductor. Using the relations $Q = CV$ and $I = \frac{dQ}{dt}$, where Q is the charge stored on the capacitor, we can express the Hamiltonian as:

$$H = \frac{1}{2}L\dot{Q}^2 + \frac{Q^2}{2C}. \tag{4.22}$$

This leads to the classical Lagrangian

$$\mathcal{L}[Q, \dot{Q}] = \frac{1}{2}L\dot{Q}^2 - \frac{Q^2}{2C}. \tag{4.23}$$

The canonical momentum conjugate to Q is given by:

$$\Phi := \frac{\partial \mathcal{L}}{\partial \dot{Q}} = L\dot{Q}, \tag{4.24}$$

where Φ is identified as the magnetic flux. Therefore, the Hamiltonian expressed in canonical variables becomes:

$$H = \frac{\Phi^2}{2L} + \frac{Q^2}{2C}. \tag{4.25}$$

doi:10.6342/NTU202501658

The Hamiltonian is structurally identical to the form of a harmonic oscillator:

$$H = \frac{p^2}{2m} + \frac{1}{2}m\omega^2 x^2, \tag{4.26}$$

with the correspondences $Q \leftrightarrow x$, $\Phi \leftrightarrow p$, $L \leftrightarrow m$, and $\omega = \frac{1}{\sqrt{LC}}$. Upon quantization, Q and Φ become operators satisfying the canonical commutation relation:

$$[\hat{Q}, \hat{\Phi}] = i\hbar. \tag{4.27}$$

The energy spectrum then follows:

$$E_n = \hbar\omega\left(n + \frac{1}{2}\right), \quad n = 0, 1, 2, \ldots \tag{4.28}$$

However, the equidistant spacing of energy levels in a harmonic oscillator prevents its direct use as a qubit. An externally applied microwave pulse resonant with the $|0\rangle \rightarrow |1\rangle$ transition could also unintentionally excite the $|1\rangle \rightarrow |2\rangle$ transition, leading to leakage out of the computational subspace.

To address this issue, the linear inductor is replaced with a nonlinear element known as the *Josephson junction*. A Josephson junction is constructed with two superconductors that seperated by a thin insulating barrier, and its current–phase relation is given by [71]:

$$I = I_c \sin\left(\frac{2\pi}{\Phi_0}\Phi(t)\right), \tag{4.29}$$

where the critical current is denoted as I_c, and $\Phi_0 = \frac{h}{2e}$ is the magnetic flux quantum. In the junction, the corresponding potential energy stored is obtained via:

$$U(\phi) = -\int I(\Phi)\, d\Phi = -E_J \cos\left(\frac{2\pi}{\Phi_0}\Phi\right), \quad \text{with} \quad E_J = \frac{\Phi_0 I_c}{2\pi}, \tag{4.30}$$

where E_J is referred to as the Josephson energy.

If the inductor inside the LC oscillator is replaced by a Josephson junction, the total Hamiltonian becomes:

$$H = \frac{Q^2}{2C} - E_J \cos(\phi), \tag{4.31}$$

where $\phi = \frac{2\pi}{\Phi_0}\Phi$. This is the Hamiltonian of a nonlinear oscillator, commonly referred to as a Josephson junction qubit. Specifically, the transmon qubit operates in the regime where $E_J \gg E_C$, with $E_C = \frac{e^2}{2C}$ being the charging energy [72]. In this limit, the system behaves like a weakly anharmonic oscillator, where the energy levels remain nearly harmonic but exhibit sufficient nonlinearity to distinguish the states $|0\rangle$ and $|1\rangle$ from higher levels.

The cosine potential introduces the required anharmonicity, enabling reliable qubit opera-

doi:10.6342/NTU202501658

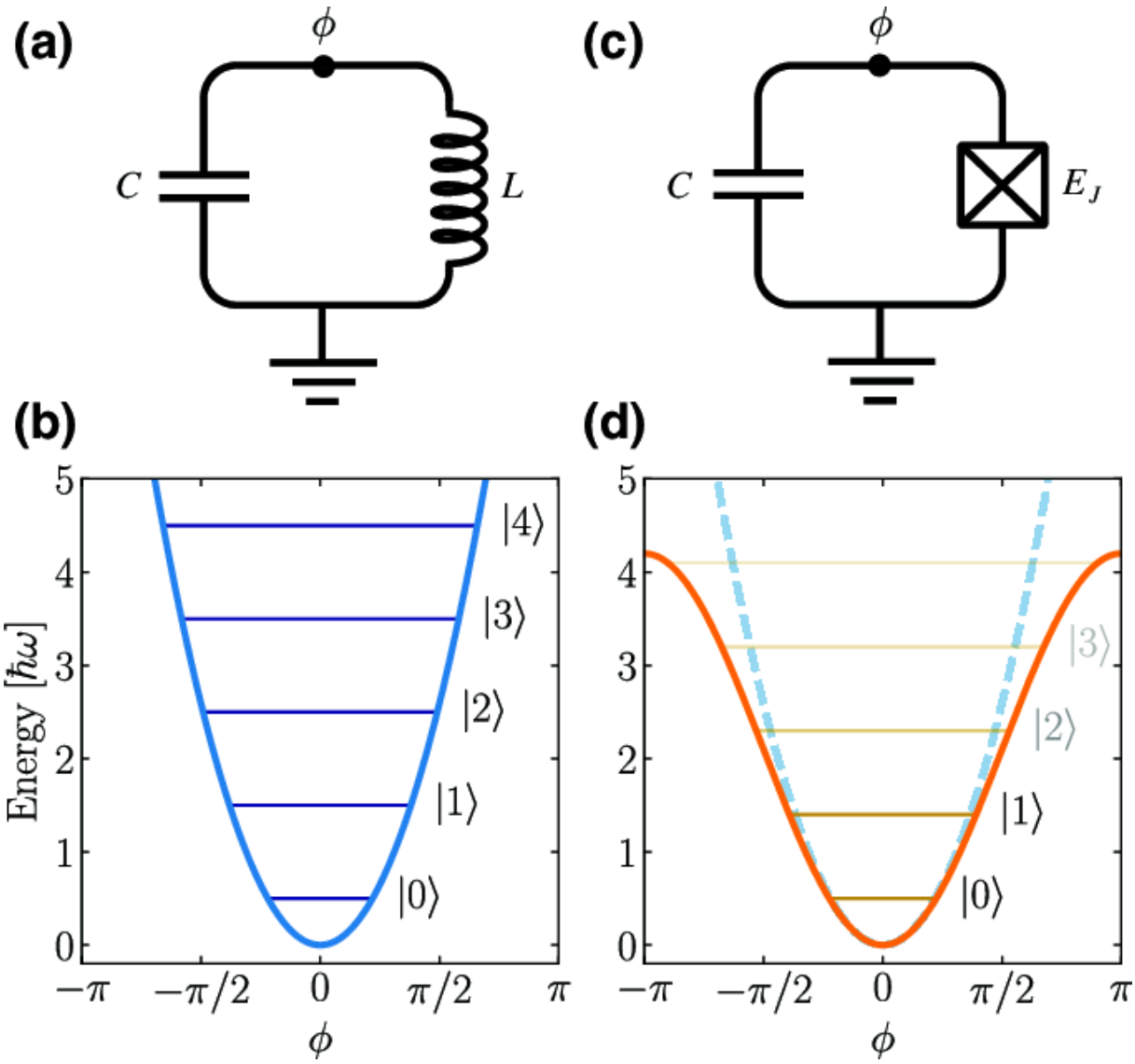


Figure 4.8: Image adopted from [70]. (a) LC circuit. (b) The energy spectrum of quantum harmonic oscillators. (c) LC circuit replaced by a Josephson junction. (d) The energy levels of the quantum anharmonic oscillator, where the Josephson junction introduces the anharmonicity.

doi:10.6342/NTU202501658

tions restricted to the two-level subspace. This design prevents leakage and becomes foundations for several superconducting quantum computing architectures, such as those deployed by IBM.

4.2.2 Qubit Control via Microwave Drive

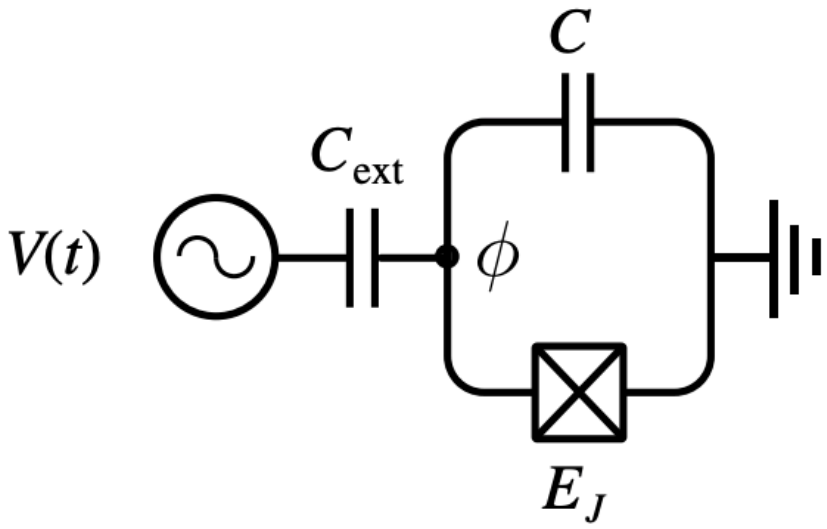


Figure 4.9: A single transmon qubit coupled to an external driving microwave field. Figure adapted from [71].

The transmon qubit is manipulated through the application of externally driven microwave pulses. This control is implemented by coupling the superconducting circuit to a voltage source via an external capacitor. The presence of this capacitor modifies the system' s Lagrangian, originally given in Eq. 4.23, to the following form [71]:

$$\mathcal{L}[\phi, \dot{\phi}] = \frac{C}{2}\dot{\phi}^2 + E_J \cos\phi + \frac{C_{\text{ext}}}{2}\left(V(t) - \dot{\phi}\right)^2, \tag{4.32}$$

where C_{ext} is the external coupling capacitance, and $V(t)$ is the time-dependent driving voltage. The total system now behaves as a weakly anharmonic oscillator subject to an external drive, allowing selective control of transitions between energy levels. The effective interaction Hamiltonian induced by the microwave drive can be expressed as:

$$H_{\text{int}}^{\text{ext}} \propto V(t)\left[\sin(\omega t)\hat{\sigma_X} - \cos(\omega t)\hat{\sigma_Y}\right], \tag{4.33}$$

where ω is the drive frequency, and $\hat{\sigma_X}$ and $\hat{\sigma_Y}$ are Pauli operators acting on the qubit. By selecting a suitable voltage waveform $V(t)$ (e.g., a sinusoidal), one can implement arbitrary rotations around axes in the Bloch sphere.

To implement entangling gates between qubits, direct interaction between multiple transmons must be introduced. A common coupling mechanism is through a shared capacitor or bus

doi:10.6342/NTU202501658

resonator. Consider two transmon circuits coupled with capacitors, the total Hamiltonian can be written as:

$$H = H_1 + H_2 + H_{\text{int}}, \tag{4.34}$$

where H_1 and H_2 are the individual transmon Hamiltonians, and the interaction term is:

$$H_{\text{int}} = C_g V_1 V_2, \tag{4.35}$$

with C_g representing the mutual coupling capacitance, and V_1, V_2 the voltage operators associated with the two qubits.

In the qubit basis, and under appropriate approximations, this capacitive interaction induces terms such as $\sigma_1^x \sigma_2^x$ or $\sigma_1^z \sigma_2^z$, which form the basis of commonly used entangling quantum gates, e.g., the controlled-Z (CZ) or iSWAP. These gates are realized by tuning the drive frequency near the avoided crossing of specific multi-qubit energy levels or by parametrically modulating the interaction strength.

These entangling operations are important for universal quantum gates. The specific coupling architecture determines the available gate set, the strength and type of interactions, and the gate speed. Careful calibration and error mitigation techniques are used to ensure high-fidelity two-qubit operations, with IBM Quantum systems currently achieving two-qubit gate fidelities above 99% in several devices.

4.2.3 Cryogenic Environment and Dilution Refrigeration

Quantum computers based on superconducting circuits require extremely low temperatures to operate correctly. IBM Quantum systems achieve this by placing the quantum processor inside a specialized cryogenic system known as a *dilution refrigerator*, often informally referred to as the ”big fridge”. This refrigerator cools the qubit chip down to temperatures as low as $10-15$ millikelvin, and operational temperatures are typically around 4 millikelvin, which is just a fraction of a degree above absolute zero.

The dilution refrigerator works by taking advantage of the special features of a mixture of two helium isotopes: ^{4}He and ^{3}He [74]. Maintaining this ultra-cold environment is critical for the proper operation of superconducting qubits. There are three main reasons:

1. **Superconductivity:** The materials used in the qubits must be in a superconducting state to function. Superconductivity only occurs below certain critical temperatures. At millikelvin temperatures, superconductivity is stable and reliable.

2. **Reducing thermal noise:** At higher temperatures, thermal energy can excite the qubit

doi:10.6342/NTU202501658

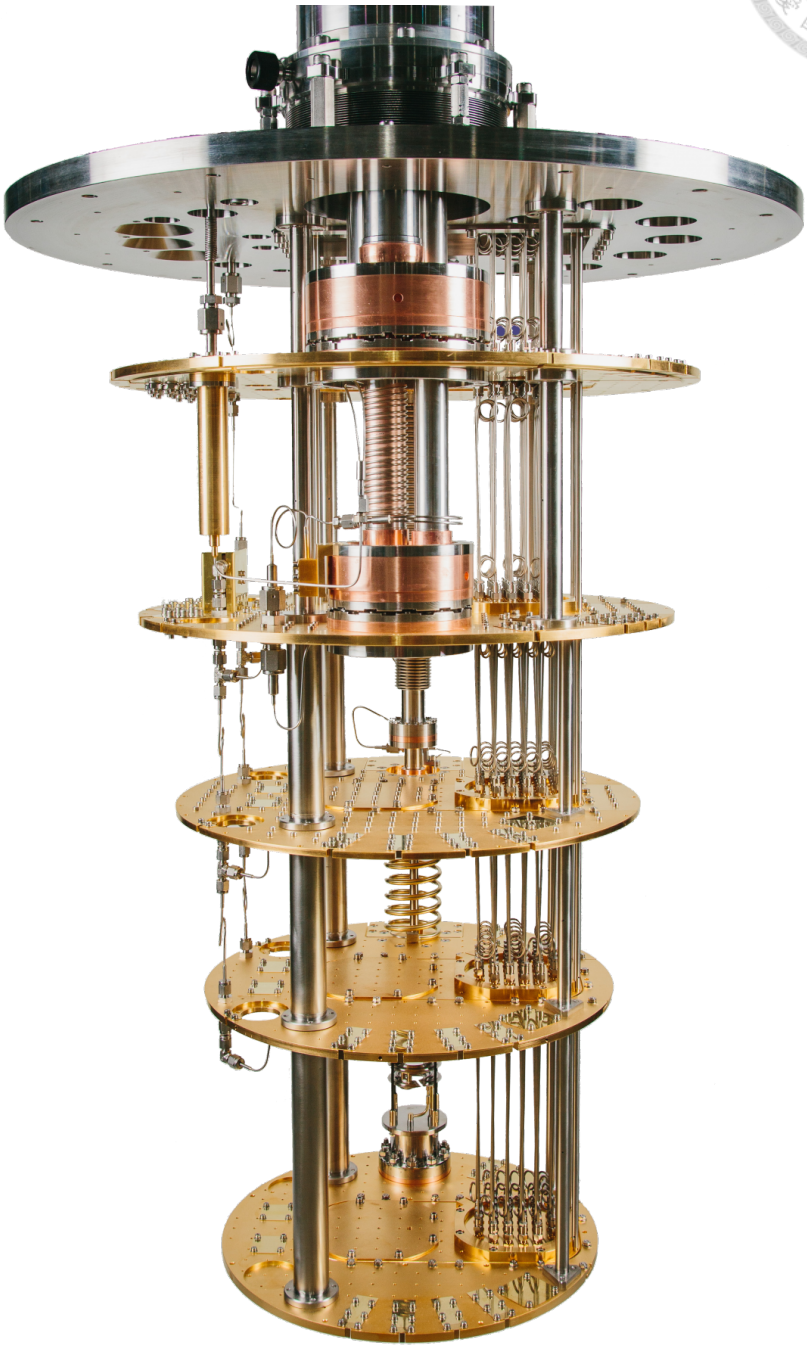


Figure 4.10: A dilution refrigerator used to cool quantum processors to millikelvin temperatures. Multiple temperature stages are stacked vertically, with the quantum processor mounted at the bottom-most plate. Figure adapted from [73].

doi:10.6342/NTU202501658

from the ground state to higher states, causing decoherence or unintended operations. By cooling the system to 4 mK, the thermal energy $k_B T$ becomes much smaller than the qubit energy gap $\hbar\omega_q$, suppressing these excitations.

3. **Improving coherence times:** Low temperatures reduce the interactions between the surrounding environments (such as phonons or stray photons) and qubits. This leads to longer coherence times, allowing more quantum operations to be performed before errors occur.

Figure 4.10 shows an illustration with the similar structure of an IBM dilution refrigerator. The quantum chip is mounted at the bottom stage, where the temperature is lowest. Higher temperature stages are used to shield and support the system, ensuring minimal thermal load reaches the qubits. This cryogenic environment is essential to implement superconducting circuits and protect the qubits, allowing IBM' s quantum computers to perform coherent quantum operations with high fidelity.

4.3 Variational Quantum Circuit (VQC)

4.3.1 The Ansatz of Variational Quantum Circuit

The Variational Quantum Circuit (VQC) [28–30] is a hybrid approach that integrates quantum and classical components to address optimization problems in quantum computing. This framework utilizes quantum features such as the entanglement and the superposition, while employing classical techniques for optimization. In a VQC, the quantum circuit is governed by tunable classical parameters. These parameters are refined through repeated application of classical optimization routines, with the goal of reducing a predefined cost function.

As shown in Figure 4.11, a VQC typically comprises two main components: data encoding and introduction of tunable parameters, with the corresponding unitary transformations denoted as U_{ENC} and U_{PARAM}, respectively. For an n-qubit VQC, the circuit's output can be expressed as:

$$f(\mathbf{x}, \boldsymbol{\theta}) = \langle 0|^{\otimes n} U^{\dagger}(\mathbf{x}, \boldsymbol{\theta}) P U(\mathbf{x}, \boldsymbol{\theta}) |0\rangle^{\otimes n}, \tag{4.36}$$

where P is a Pauli string representing a measurement observable, and U is a unitary operator. The quantum state begins in the n-qubit $|0\rangle$ state, denoted $|0\rangle^{\otimes n}$. The unitary gate U will encode classical data $\mathbf{x}$ on the qubit while also introducing tunable parameters $\boldsymbol{\theta}$.

A Pauli string refers to a product of Pauli operators acting on multiple qubits, typically structured as a tensor product. For instance, $X_1 \otimes Z_2$ represents a configuration where the 1st qubit is associated with an X measurement and the 2nd qubit with a Z measurement. In actual

doi:10.6342/NTU202501658

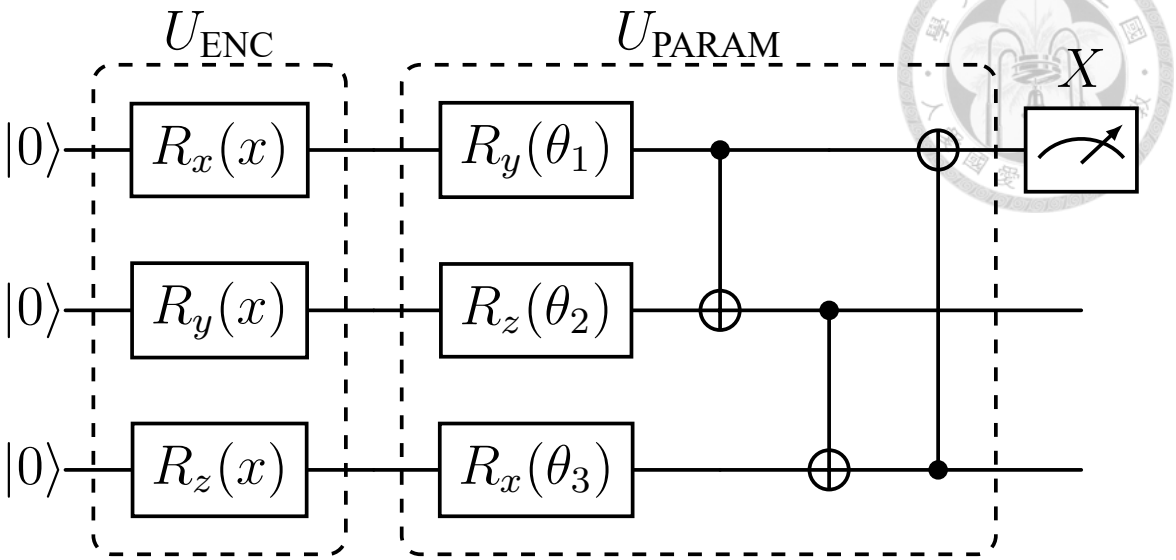


Figure 4.11: An example of a VQC circuit with three qubits, with all qubits initialize to $|0\rangle$. The data $x \in \mathbb{R}$ is encoded through the rotation gates, and the tunable parameters θ_i are introduced through the rotation gates, then the circuit is applied with a series of controlled gates to produce the entanglement. The final output is the expectation value of the X-basis measurement in the first qubit.

implementations, quantum hardware is generally limited to measurements defined by specific sets of observables. To address this, many observables are rewritten as sums over multiple Pauli strings. As a result, in the context of variational quantum circuits (VQCs), the observable P is often selected as a Pauli string or as a sum over such strings, allowing it to be represented in terms of Pauli matrices.

In most cases, both data encoding and parameterized unitary operations are implemented using general rotation gates as described in Equation 4.17. Consequently, the data $\mathbf{x}$ is usually normalized to a specific range, typically either $[0, 1]$ or $[-\pi, \pi]$, to ensure that the parameters remain within the operational range of the rotation gates.

The process follows a structure similar to classical machine learning workflows. After the quantum circuit is evaluated, the expectation values of the observable P are either passed to another classical neural network or used directly as output. The loss function is then defined based on these values, and the gradients of the tunable parameters are computed. The gradients are collected to iteratively tune the parameters until the loss function converges, ensuring the optimal quantum circuit configuration.

doi:10.6342/NTU202501658

4.3.2 The Parameter-Shift Rule

Unlike classical computing, where outputs are deterministic, quantum computation is inherently probabilistic, causing expectation values to be affected by noises from statistical fluctuations. Consequently, traditional finite difference methods become impractical for computing gradients of VQCs. To circumvent this challenge, the PARAMETER-SHIFT RULE (PSR) [75–77] provides an analytical method to compute these gradients precisely, requiring only minor modifications of the original VQC. Here, we derive the PSR specifically for single-qubit rotation gates generated by Pauli matrices. A comprehensive discussion on more general single-qubit gates is discussed in [75–77].

Consider a single-qubit rotation gate parameterized by an angle θ and generated by a Pauli matrices $\sigma \in \{X, Y, Z\}$ (defined explicitly in Equation 4.20). The corresponding unitary operator can be expressed as:

$$U_\sigma(\theta) = e^{-i\frac{\theta}{2}\sigma} = \cos\left(\frac{\theta}{2}\right) I - i \sin\left(\frac{\theta}{2}\right) \sigma. \tag{4.37}$$

For an n-qubit VQC initialized to $|0\rangle^{\otimes n}$, with unitary operations

$$U_1, \ldots, U_i, U_\sigma(\theta), U_{i+1}, \ldots, U_N, \tag{4.38}$$

followed by measurement of an observable P, the output function $f(\theta)$ is given by:

$$f(\theta) = \langle 0|^{\otimes n} U_1^\dagger \ldots U_i^\dagger U_\sigma(\theta)^\dagger U_{i+1}^\dagger \ldots U_N^\dagger P U_N \ldots U_{i+1} U_\sigma(\theta) U_i \ldots U_1 |0\rangle^{\otimes n} \tag{4.39}$$

$$= \langle\psi| U_\sigma(\theta)^\dagger A U_\sigma(\theta) |\psi\rangle, \tag{4.40}$$

where $|\psi\rangle = U_i \ldots U_1 |0\rangle^{\otimes n}$ and $A = U_{i+1}^\dagger \ldots U_N^\dagger P U_N \ldots U_{i+1}$. The derivative of the expectation value, i.e., the model output $f(\theta)$, with respect to θ can be computed as:

$$\begin{aligned}
\frac{df(\theta)}{d\theta} &= \langle\psi| [\frac{i\sigma}{2} U_\sigma^\dagger(\theta)] A U_\sigma(\theta) |\psi\rangle + \langle\psi| U_\sigma^\dagger(\theta) A [\frac{-i\sigma}{2} U_\sigma(\theta)] |\psi\rangle \\
&= \langle\psi| [\frac{i\sigma}{\sqrt{2}} U_\sigma^\dagger(\theta)] A [\frac{1}{\sqrt{2}} U_\sigma(\theta)] |\psi\rangle \langle\psi| [\frac{1}{\sqrt{2}} U_\sigma^\dagger(\theta)] A [\frac{-i\sigma}{\sqrt{2}} U_\sigma(\theta)] |\psi\rangle \\
&= \langle\psi| [\sin(\frac{\pi}{4}) i\sigma U_\sigma^\dagger(\theta)] A [\cos(\frac{\pi}{4}) U_\sigma(\theta)] |\psi\rangle \\
&\quad + \langle\psi| [\cos(\frac{\pi}{4}) U_\sigma^\dagger(\theta)] A [-\sin(\frac{\pi}{4}) i\sigma U_\sigma(\theta)] |\psi\rangle,
\end{aligned} \tag{4.41}$$

where the property of Hermicity of Pauli matrices, i.e., $\sigma = \sigma^\dagger$, are used. By exploiting the periodic structure of Pauli rotations and using trigonometric identities, the gradient can be rep-

doi:10.6342/NTU202501658

resented in terms of function evaluations at shifted parameter values:

$$\begin{aligned} f(\theta+\frac{\pi}{2}) &= \langle\psi|\, U_\sigma^\dagger(\theta+\frac{\pi}{2}) A U_\sigma(\theta+\frac{\pi}{2})\,|\psi\rangle \\ &= \langle\psi|\left[(\cos(\frac{\pi}{4})+i\sin(\frac{\pi}{4})\sigma) U_\sigma^\dagger(\theta)\right] A U_\sigma(\theta)\,|\psi\rangle \\ &\quad + \langle\psi|\, U_\sigma^\dagger(\theta) A \left[(\cos(\frac{\pi}{4})-i\sin(\frac{\pi}{4})\sigma) U_\sigma(\theta)\right]|\psi\rangle\,, \end{aligned} \tag{4.42}$$

and

$$\begin{aligned} f(\theta-\frac{\pi}{2}) &= \langle\psi|\, U_\sigma^\dagger(\theta-\frac{\pi}{2}) A U_\sigma(\theta\frac{\pi}{2})\,|\psi\rangle \\ &= \langle\psi|\left[(\cos(\frac{\pi}{4})-i\sin(\frac{\pi}{4})\sigma) U_\sigma^\dagger(\theta)\right] A U_\sigma(\theta)\,|\psi\rangle \\ &\quad + \langle\psi|\, U_\sigma^\dagger(\theta) A \left[(\cos(\frac{\pi}{4})+i\sin(\frac{\pi}{4})\sigma) U_\sigma(\theta)\right]|\psi\rangle\,. \end{aligned} \tag{4.43}$$

Through algebraic simplification, one arrives at the explicit form of the PSR for gradients respect to rotation gates generated by Pauli matrices:

$$\frac{df(\theta)}{d\theta} = \frac{1}{2}\left[f\left(\theta+\frac{\pi}{2}\right) - f\left(\theta-\frac{\pi}{2}\right)\right]. \tag{4.44}$$

Figure 4.12 demonstrates an illustration of how PSR implemented respect to a single qubit, typically a rotation gate, with tunable parameters ω, ϕ, θ.

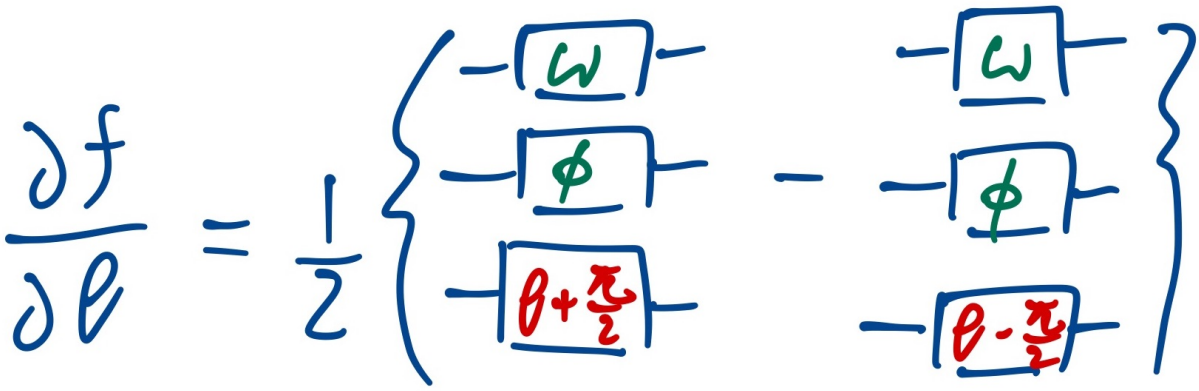


Figure 4.12: An illustration of parameter-shift rule respect to a single qubit with parameter θ. The other parameters ω and ϕ stay constant when calculating the partial derivative respect to θ.

Equation 4.44 enables efficient and exact computation of gradients within VQCs, solving the issues from the inaccuracies caused by finite difference approximations. This approach is particularly advantageous in large-scale quantum circuits, significantly improving scalability and precision. Furthermore, the PSR naturally extends to multi-qubit gates by applying the rule individually to each parameterized gate acting on the constituent qubits of the circuit.

doi:10.6342/NTU202501658

4.3.3 Data-Reuploading Technique

To enhance the expressiveness of VQCs, the data-reuploading technique [78] is frequently utilized. This method involves repeatedly encoding the input data into the quantum circuit, eventually allowing the circuit to capture more complicated relationships of the data. By introducing multiple repetitions of the data encoding step interleaved with parameterized gates, each reuploading introduces distinct tunable parameters, thus significantly augmenting the model's capacity to represent complex data relationships.

Typically, as illustrated in Figure 4.13, the data-reuploading approach alternates between data encoding gates (U_{ENC}) and parameterized gates (U_{PARAM}). For practicality, the data encoding ansatz usually remains fixed across reuploading steps, while the parameters of the gates differ each time to enhance model expressiveness.

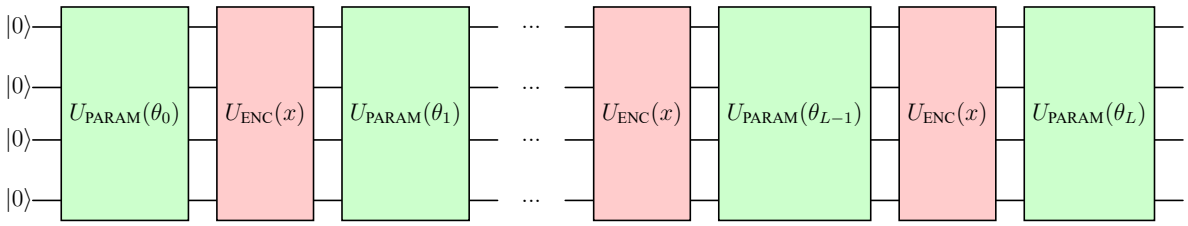


Figure 4.13: A VQC example that employs the data-reuploading technique. The data encoding gates U_{ENC} (red) and parameterized gates U_{PARAM} (green) alternate and are applied L times (or $L+1$ times with an additional U_{PARAM} at the beginning). While the encoding ansatz remains the same across repetitions, distinct parameters θ are utilized in each repetition, significantly increasing circuit expressiveness.

A formal mathematical justification of the increased expressiveness provided by data-reuploading is presented in [79]. Here we outline a concise yet rigorous argument from the literature. Without loss of generality, we assume the Hamiltonian associated with the data encoding gate (U_{ENC}) is diagonal. Specifically, any Hermitian matrices can be diagonalized via unitary transformations, with these transformations can be absorbed into the parameterized gates (U_{PARAM}). Denoting the eigenvalues of the encoding Hamiltonian as $\{\lambda_1, \ldots, \lambda_d\}$, with d representing its dimensionality, one defines:

$$\Lambda_{\mathbf{j}} = \lambda_{j_1} + \cdots + \lambda_{j_L}, \quad \text{with} \quad \mathbf{j} \in [d]^L, \tag{4.45}$$

which are sums of all d^L combinations of eigenvalue sums. These sums determine a real-valued set:

$$\Omega = \{\Lambda_{\mathbf{k}} - \Lambda_{\mathbf{j}} \quad \text{with} \quad \mathbf{k}, \mathbf{j} \in [d]^L\}. \tag{4.46}$$

As demonstrated in [79], the output function of the VQC given input data x is a summation of

doi:10.6342/NTU202501658

Fourier series, and can be written as

$$\mathrm{VQC}(x) = \sum_{\omega \in \Omega} a_\omega e^{ix\omega}, \tag{4.47}$$

where a_ω denotes Fourier coefficients corresponding to frequencies $\omega \in \Omega$. As the number of reuploading L increases, the set Ω expands due to the combinatorial increase in eigenvalue combinations. Consequently, the circuit can approximate increasingly intricate functions even with a fixed encoding ansatz. Thus, the data-reuploading technique substantially enhances the expressiveness of VQCs, facilitating their scalability and capability to model complex tasks.

doi:10.6342/NTU202501658

doi:10.6342/NTU202501658

Chapter 5

Quantum Complete Graph Neural Network (QCGNN)

For this chapter, we give an detailed explanation of the Quantum Complete-Graph Neural Network (QCGNN). In short, the QCGNN is a quantum variant of the graph neural network and specifically designed for complete graphs. This architecture is particularly suitable for jet discrimination tasks in HEP, where each particle within a jet is treated as a node in a fully connected graph.

The QCGNN is constructed to be permutation invariant and supports both classification and regression tasks. It leverages quantum entanglement and parallelism by employing VQCs with carefully chosen observables. We also analyze the computational complexity of QCGNN and demonstrate its potential efficiency advantage over classical models. Furthermore, we explore its connection to quantum kernel methods, which is known as a well-studied approach in theoretical machine learning, and discuss extensions of the QCGNN to sequential data and general graph structures.

5.1 Model Architecture of QCGNN

The QCGNN employs two distinct quantum registers, where the 1st quantum register is called *index register* (IR). The "index" stands for the reason of relations to indices of data. On the other hand, the 2nd quantum register is named *network register* (NR) for reason of containing trainable parameters. For a complete graph with N particles (N nodes), the IR encodes particle indices, while the NR is used for the quantum neural network (i.e., the VQC ansatz). Although the particles are indexed during preprocessing (seemingly breaking permutation invariance), we show that with a properly designed measurement observable, the final output remains permutation invariant.

doi:10.6342/NTU202501658

The IR consists of $n_I = \lceil \log_2 N \rceil$ qubits, sufficient to represent N basis states. For example, five particles are encoded as $|000\rangle$ to $|100\rangle$ in binary. The NR consists of n_Q qubits, a tunable hyperparameter analogous to the number of hidden neurons in classical MLPs. Increasing n_Q enhances the model's expressive power. A long-standing hypothesis in quantum computing is that deeper and wider VQCs can learn richer representations than classical models, potentially enabling quantum advantage.

Step 1: Uniform Superposition Initialization

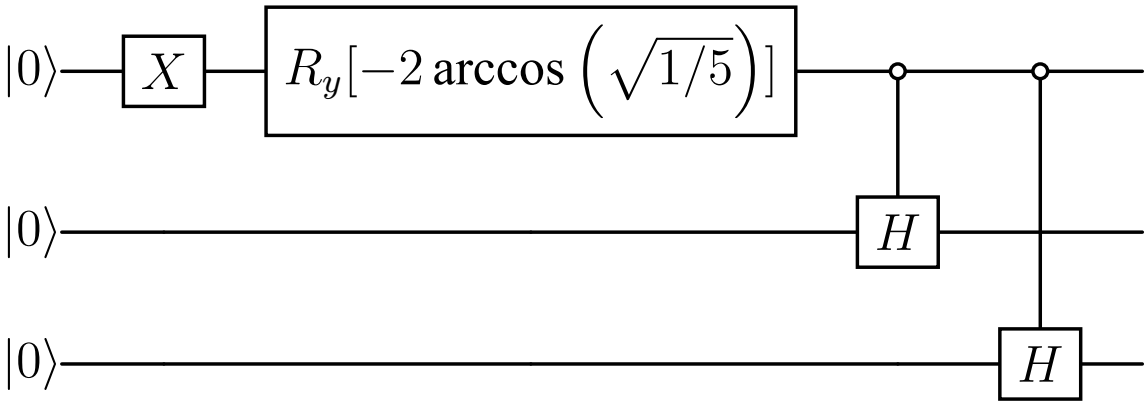


Figure 5.1: A quantum circuit of uniform state oracle that prepares a uniform superposition over $N = 5$ states.

We initialize the IR into a uniform superposition of N basis states:

$$|\psi_0\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle \, |0\rangle^{\otimes n_Q} , \tag{5.1}$$

where $|i\rangle$ spans the computational basis of the IR and the NR is initialized to $|0\rangle^{\otimes n_Q}$. When N is not a power of two, this state cannot be prepared using Hadamard gates alone. Instead, specialized circuits known as Uniform State Oracles (USOs) are used to construct this superposition [80, 81]. Figure 5.1 illustrates such a circuit for $N = 5$. Section 5.1.1gives a detailed implementation on USO.

doi:10.6342/NTU202501658

Step 2: Particle Information Encoding

For i-th particle, the corresponding feature vector $\mathbf{x}_i$ will be encoded on the NR through a unitary transformation $U_{\text{ENC}}(\mathbf{x}_i)$, conditioned on its corresponding IR basis state:

$$C_i^{n_I} U_{\text{ENC}}(\mathbf{x}_i) \left[|j\rangle |0\rangle^{\otimes n_Q} \right] = \begin{cases} |i\rangle U_{\text{ENC}}(\mathbf{x}_i) |0\rangle^{\otimes n_Q} & \text{if } j = i, \\ |j\rangle |0\rangle^{\otimes n_Q} & \text{otherwise.} \end{cases} \tag{5.2}$$

Applying these controlled gates for all i results in the state:

$$|\psi_1\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle U_{\text{ENC}}(\mathbf{x}_i) |0\rangle^{\otimes n_Q} . \tag{5.3}$$

While the construction appears dependent on particle ordering, we will demonstrate that the output remains permutation invariant after measurement with a suitable observable. Details on decomposing the multi-controlled unitaries into elementary gates are presented in Section 5.1.2.

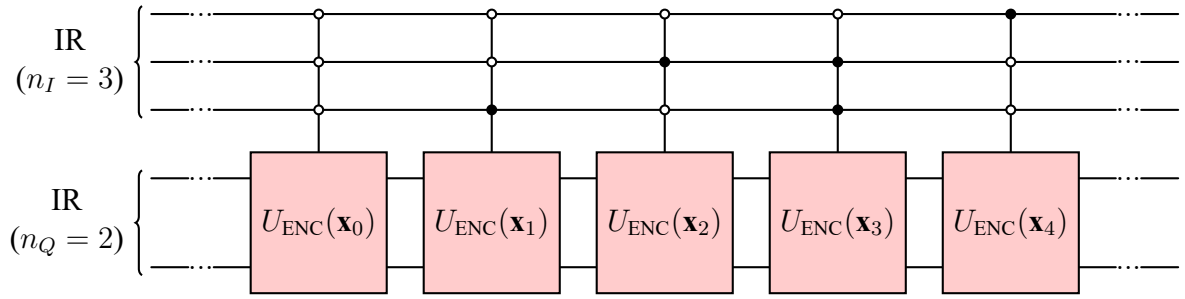


Figure 5.2: Quantum circuit encoding five particle features via multi-controlled gates. White and black circles denote control values of 0 and 1, respectively. The NR has $n_Q = 2$ qubits.

Step 3: Parameterized Ansatz

We introduce trainable parameters via a standard variational ansatz $U_{\text{PARAM}}(\boldsymbol{\theta})$ acting solely on NR:

$$|\psi_2\rangle = U_{\text{PARAM}}(\boldsymbol{\theta}) |\psi_1\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle U_{\text{PARAM}}(\boldsymbol{\theta}) U_{\text{ENC}}(\mathbf{x}_i) |0\rangle^{\otimes n_Q} . \tag{5.4}$$

Each $\mathbf{x}_i$ is transformed identically under $U_{\text{PARAM}}(\boldsymbol{\theta})$, analogous to the shared Φ function of classical MPGNNs in Equation 3.3. To improve expressiveness, we use a data-reuploading technique, alternating U_{ENC} and U_{PARAM} layers across R repetitions:

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle \left[\prod_{r=1}^{R} U_{\text{PARAM}}(\boldsymbol{\theta}^{(r)}) U_{\text{ENC}}(\mathbf{x}_i) \right] |0\rangle^{\otimes n_Q} . \tag{5.5}$$

doi:10.6342/NTU202501658

For simplicity, we denote

$$|\mathbf{x}_i, \boldsymbol{\theta}\rangle \equiv \left[\prod_{r=1}^{R} U_{\text{PARAM}}(\boldsymbol{\theta}^{(r)}) U_{\text{ENC}}(\mathbf{x}_i)\right] |0\rangle^{\otimes n_Q} . \tag{5.6}$$

Step 4: Measurement with $J \otimes P$

Let P be a Pauli observable on the NR. Measuring the state yields:

$$\langle\psi|P|\psi\rangle = \frac{1}{N}\sum_{i=0}^{N-1} \langle \mathbf{x}_i, \boldsymbol{\theta}| \, P \, |\mathbf{x}_i, \boldsymbol{\theta}\rangle . \tag{5.7}$$

To achieve permutation invariance, we consider a Hermitian operator on the IR, denoted as J and defined as the all-ones matrix. Measuring with $J \otimes P$ yields:

$$\begin{aligned} \langle\psi|J \otimes P|\psi\rangle &= \frac{1}{N}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \langle \mathbf{x}_i, \boldsymbol{\theta}| \, P \, |\mathbf{x}_j, \boldsymbol{\theta}\rangle , \\ &= \frac{1}{N}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} h(\mathbf{x}_i, \mathbf{x}_j; P), \end{aligned} \tag{5.8}$$

where

$$h(\mathbf{x}_i, \mathbf{x}_j; P) = \frac{1}{2}\left[\langle \mathbf{x}_i, \boldsymbol{\theta}| \, P \, |\mathbf{x}_j, \boldsymbol{\theta}\rangle + \langle \mathbf{x}_j, \boldsymbol{\theta}| \, P \, |\mathbf{x}_i, \boldsymbol{\theta}\rangle\right], \tag{5.9}$$

ensuring symmetry. This measurement aggregates over all pairwise interactions and is therefore permutation invariant.

In practice, the observable J is constructed as:

$$J = \bigotimes_{q=1}^{n_I} (I_q + X_q), \tag{5.10}$$

where I_q and X_q denote the identity and X-Pauli operators on the q-th IR qubit, respectively. To exclude self-interactions, we subtract the identity contribution:

$$J \to J - \bigotimes_{q=1}^{n_I} I_q. \tag{5.11}$$

An example of the affects of different Pauli strings on the IR qubits is shown in Figure 5.3.

In summary, Figure 5.4 illustrates the QCGNN circuit, which consists of a USO, followed by R repetitions of U_{ENC} and U_{PARAM} layers. The final measurement is performed with the observable $J \otimes P$, yielding a permutation invariant output.

doi:10.6342/NTU202501658

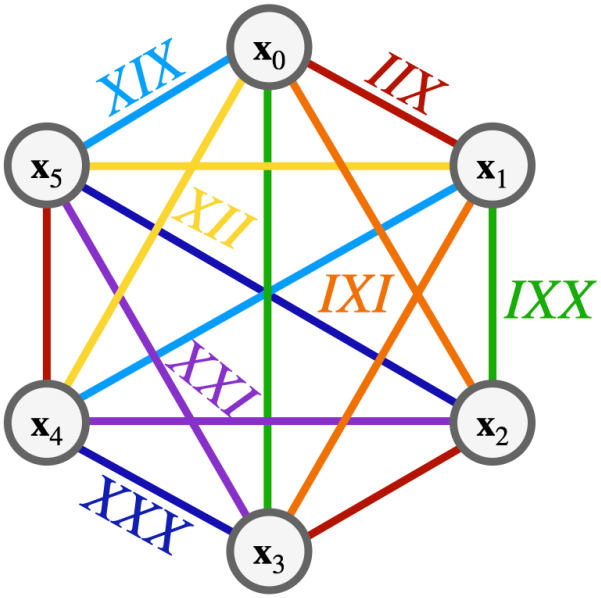


Figure 5.3: Illustration of how Pauli X observable affects the IR qubits. This figure shows a jet with six particles. The edges show the different combinations of Pauli X observables acting on the IR qubits. The edges with same color represent the pair correlations of the same Pauli string.

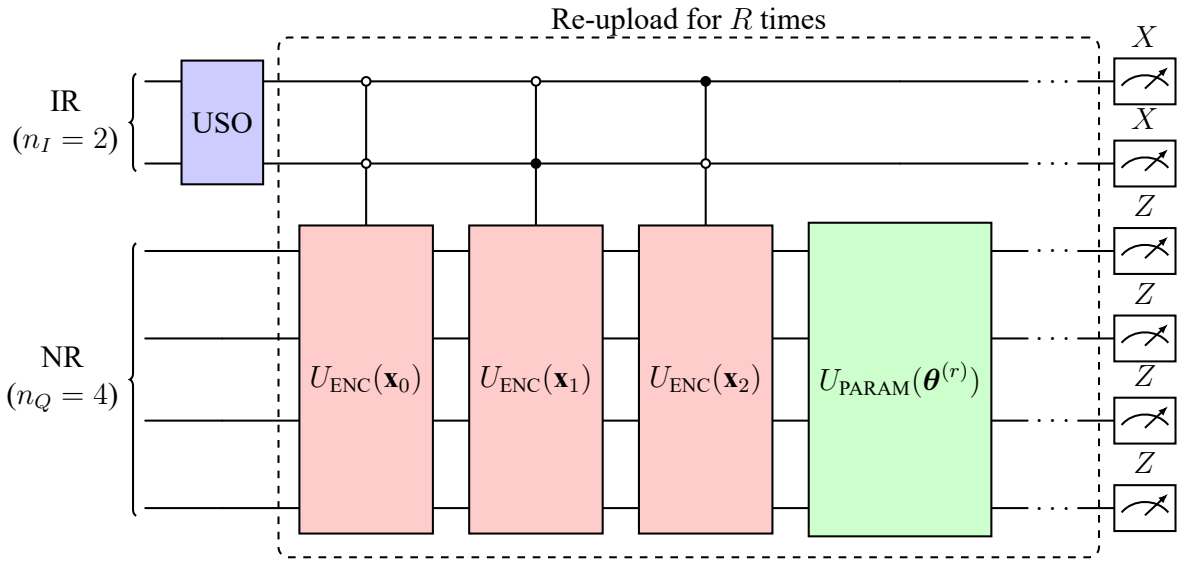


Figure 5.4: An example quantum circuit for the QCGNN with $N = 3$ particles, using $n_I = 2$ and $n_Q = 4$ qubits. The circuit starts with a USO (purple box), followed by R alternating layers of encoding ansatz (red boxes) and parameterized gates (green boxes). Measurements are performed on IR in the X basis. For NR, we arbitrarily chosen measurements in the Z basis. The final output is computed by evaluating the expectation value of $J \otimes P$, as defined in Equations 5.8 and 5.10.

doi:10.6342/NTU202501658

5.1.1 Uniform State Oracle

The Uniform State Oracle (USO) is a quantum algorithm designed to prepare a uniform superposition of N basis states, starting from an initial state in the $|0\rangle$ superposition. One of the USO ansatz can be found in [80], which requires only $O(\log_2 N)$ two-qubit gates. Here, we provide a high-level motivation, while leaving the derivation in the original work [80].

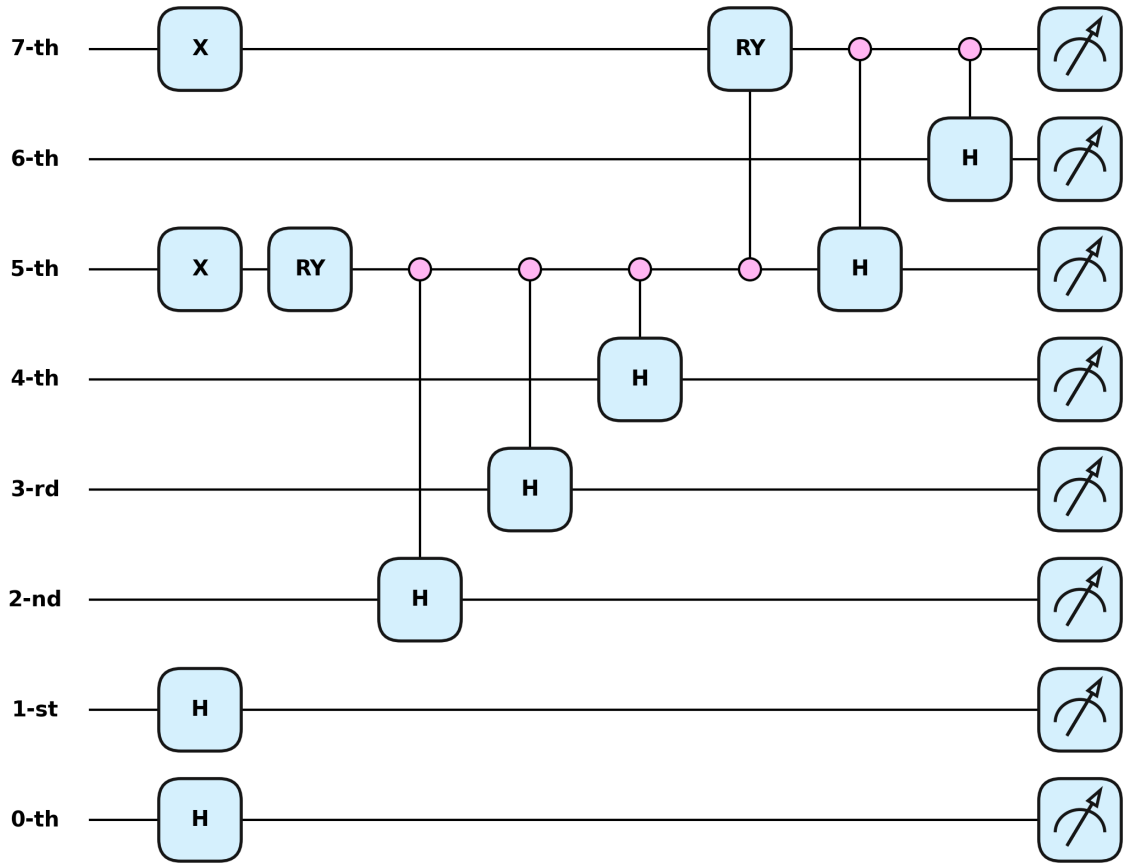


Figure 5.5: A USO quantum circuit for $N = 164$, drawn with *PennyLane* package [82]. The qubits in the top and the bottom are the most significant and least significant qubits, respectively. The qubits are labeled from 0 to 7. The 0-th qubit is the least significant qubit and the most significant qubit is the 7-th qubit.

Consider an even number $N = 164$. Its corresponding binary representation is 10100100, where the rightmost qubit (the 0-th qubit) stands for the least significant bit, and the leftmost qubit (the 7-th qubit) represents the most significant bit. The corresponding USO quantum circuit is shown in Figure 5.5. The circuit is designed to prepare a uniform superposition of N states, beginning with the initial state $|0\rangle^{\otimes 8}$, which is the 8-qubit state $|00000000\rangle$. The core

doi:10.6342/NTU202501658

objective is to create a uniform superposition state as follows:

$$|\psi\rangle = \frac{1}{\sqrt{164}} \sum_{i=0}^{163} |i\rangle = \frac{1}{\sqrt{164}} \left[|00000000\rangle + |00000001\rangle + \cdots + |10100011\rangle\right]. \tag{5.12}$$

Here, we denote $|h\rangle = |0\rangle + |1\rangle$, where we intentionally omit the normalization factor $\frac{1}{\sqrt{2}}$, as it will be useful later for canceling other factors. We can decompose $|\psi\rangle$ into groups as follows:

$$|\psi\rangle = \frac{1}{\sqrt{164}} \left[|0hhhhhhh\rangle + |100hhhhh\rangle + |101000hh\rangle\right]. \tag{5.13}$$

This decomposition is justified by observing that the last state, $|10100011\rangle$, falls into the group $|101000hh\rangle$. Notably, the groups $|0hhhhhhh\rangle$, $|100hhhhh\rangle$, and $|101000hh\rangle$ correspond to 2^7, 2^5, and 2^2 states, respectively, which sum to $164 = 2^7 + 2^5 + 2^2$.

Starting from the state $|\psi\rangle = |0\rangle^{\otimes 8}$, we first apply X-gates to the 5-th and 7-th qubits, while the 2-nd qubit remains in the $|0\rangle$ state. Subsequently, we further use the Hadamard gates to all qubits on the right-hand side of 2-nd qubit, such that the result is:

$$|\psi\rangle = |00000000\rangle \xrightarrow{X\text{-gates}} |10100000\rangle \xrightarrow{H\text{-gates}} \sqrt{\frac{1}{2^2}} |101000hh\rangle. \tag{5.14}$$

To create the group $|100hhhhh\rangle$, we need to rotate the 5-th qubit into a superposition of $|0\rangle$ and $|1\rangle$, with some suitable coefficients. Since the final coefficient is $\sqrt{\frac{1}{164}} = \sqrt{\frac{1}{2^7+2^5+2^2}}$, we apply an R_y gate to 5-th qubit via an angle $\theta = -2\arccos\left(\sqrt{\frac{2^2}{2^7+2^5+2^2}}\right)$:

$$R_y \left[-2\arccos\left(\sqrt{\frac{2^2}{2^7+2^5+2^2}}\right)\right] = \begin{bmatrix} \sqrt{\frac{2^2}{2^7+2^5+2^2}} & \sqrt{\frac{2^7+2^5}{2^7+2^5+2^2}} \\ -\sqrt{\frac{2^7+2^5}{2^7+2^5+2^2}} & \sqrt{\frac{2^2}{2^7+2^5+2^2}} \end{bmatrix}, \tag{5.15}$$

which ensures that the coefficient of each state in $|101000hh\rangle$ is $\sqrt{\frac{1}{2^7+2^5+2^2}}$. The quantum state then becomes:

$$|\psi\rangle = \sqrt{\frac{1}{2^2} \cdot \frac{2^7+2^5}{2^7+2^5+2^2}} |100000hh\rangle + \sqrt{\frac{1}{2^7+2^5+2^2}} |101000hh\rangle. \tag{5.16}$$

Next, we apply controlled Hadamard gates to the 2-nd, 3-rd, and 4-th qubits, with the condition that the 5-th qubit being in the $|0\rangle$ state, yielding:

$$|\psi\rangle \xrightarrow{CH\text{-gates}} \sqrt{\frac{1}{2^2} \cdot \frac{2^7+2^5}{2^7+2^5+2^2} \cdot \frac{1}{2^3}} |100hhhhh\rangle + \sqrt{\frac{1}{2^7+2^5+2^2}} |101000hh\rangle. \tag{5.17}$$

To generate the group $|0hhhhhhh\rangle$, we need to rotate the 7-th qubit. This requires a con-

doi:10.6342/NTU202501658

trolled R_y gate, as we must ensure the group $|101000hh\rangle$ is unaffected. We apply the controlled R_y gate, with the condition that the 5-th qubit being in the $|0\rangle$ state, with an angle $\theta = -2\arccos\left(\sqrt{\frac{2^5}{2^7+2^5}}\right)$. The corresponding R_y gate is:

$$R_y\left[-2\arccos\left(\sqrt{\frac{2^5}{2^7+2^5}}\right)\right] = \begin{bmatrix} \sqrt{\frac{2^5}{2^7+2^5}} & \sqrt{\frac{2^7}{2^7+2^5}} \\ -\sqrt{\frac{2^7}{2^7+2^5}} & \sqrt{\frac{2^5}{2^7+2^5}} \end{bmatrix}, \tag{5.18}$$

which results in the quantum state:

$$|\psi\rangle = \sqrt{\frac{2^2}{2^7+2^5+2^2}}\,|000hhhhh\rangle + \sqrt{\frac{1}{2^7+2^5+2^2}}\,(|100hhhhh\rangle + |101000hh\rangle)\,. \tag{5.19}$$

Note the angle is chosen such that the coefficients of the group $|100hhhhh\rangle$ result in $\sqrt{\frac{1}{2^7+2^5+2^2}}$.

Finally, by applying controlled Hadamard gates to the 5-th and 6-th qubits, with the condition that the 7-th qubit is the $|0\rangle$ state, we arrive at the final quantum state:

$$\begin{aligned} |\psi\rangle \xrightarrow{CH\text{-gates}} & \sqrt{\frac{2^2}{2^7+2^5+2^2}\cdot\frac{1}{2^2}}\,|0hhhhhhh\rangle + \sqrt{\frac{1}{2^7+2^5+2^2}}\,(|100hhhhh\rangle + |101000hh\rangle) \\ = & \sqrt{\frac{1}{2^7+2^5+2^2}}\,[|0hhhhhhh\rangle + |100hhhhh\rangle + |101000hh\rangle]\,. \end{aligned} \tag{5.20}$$

For the case where N is an odd number, say $N = 163$, with the binary representation 10100011, the final state should be:

$$|\psi\rangle = \frac{1}{\sqrt{163}}\,[|0hhhhhhh\rangle + |100hhhhh\rangle + |1010000h\rangle + |10100010\rangle]\,. \tag{5.21}$$

The procedure is similar to the even case, i.e., we first apply X-gates to the 7-th, 5-th, and 1-st qubits. But this time, we skip applying Hadamard gates subsequently, since the maximum basis is $|10100010\rangle$. Instead, we directly apply rotation gates, and the rest of the procedure follows similarly.

The full USO algorithm is shown in Figure 5.6. The key distinction between the even and odd cases lies in the application of Hadamard gates after the X-gates (as depicted in lines 6 and 7).

In summary, this algorithm employs only two-qubit controlled gates (controlled R_y and controlled H gates) with at most $O(\log_2 N)$ operations. These gates can be easily decomposed into $CNOT$ or CZ gates, with additional single-qubit rotation gates.

doi:10.6342/NTU202501658

Algorithm 1: A quantum algorithm for the preparation of uniform superposition state $|\Psi\rangle = \frac{1}{\sqrt{M}}\sum_{j=0}^{M-1}|j\rangle$.

Input: A positive integer M with $2 < M < 2^n$ and $M \neq 2^r$ for any $r \in \mathbb{N}$.

Output: A quantum state $U_M|0\rangle^{\otimes n} = \frac{1}{\sqrt{M}}\sum_{j=0}^{M-1}|j\rangle$, that is in a uniform superposition of M distinct states.

Function Uniform (M)

/* $l_0, l_1, \ldots, l_k$ is an ordered sequence of numbers representing the locations of 1 in the reverse binary representation of M. */

Compute $l_0, l_1, \ldots, l_k$, where $M = \sum_{j=0}^{k} 2^{l_j}$ with $0 \leq l_0 < l_1 < \ldots < l_{k-1} < l_k \leq n-1$.

Initialize $|\Psi\rangle = |q_{n-1}\rangle \otimes |q_{n-2}\rangle \otimes \cdots\cdots \otimes |q_1\rangle \otimes |q_0\rangle = |0\rangle^{\otimes n}$.

Apply X gate on $|q_i\rangle$ for $i = l_1, l_2, \ldots, l_k$. // Apply X gates on qubits at positions l_1, l_2, ..., l_k.

Set $M_0 = 2^{l_0}$.

/* If M is an even number, then apply Hadamard gates on the rightmost l_0 qubits. */

if $l_0 > 0$ **then**

  Apply H gate on $|q_i\rangle$ for $i = 0, 1, \ldots, l_0 - 1$.

Apply the rotation gate $R_Y(\theta_0)$ on $|q_{l_1}\rangle$, where $\theta_0 = -2\arccos\left(\sqrt{\frac{M_0}{M}}\right)$.

Apply a controlled Hadamard (H) gate on $|q_i\rangle$ for $i = l_0, l_0+1, \ldots, l_1 - 1$ conditioned on q_{l_1} being equal to 0.

for $m = 1$ to $k-1$ **do**

  Apply a controlled $R_Y(\theta_m)$ gate, with $\theta_m = -2\arccos\left(\sqrt{\frac{2^{l_m}}{M - M_{m-1}}}\right)$, on $|q_{l_{m+1}}\rangle$ conditioned on q_{l_m} being 0.

  Apply a controlled Hadamard (H) gate on $|q_i\rangle$ for $i = l_m, l_m+1, \ldots, l_{m+1} - 1$ conditioned on $q_{l_{m+1}}$ being equal to 0.

  Set $M_m = M_{m-1} + 2^{l_m}$.

return $|\Psi\rangle$

Figure 5.6: The USO algorithm adapted from [80].

doi:10.6342/NTU202501658

5.1.2 Multi-Controlled Quantum Gates

For the data encoding of particle features, we introduced the multi-controlled quantum gates, which are essential for the QCGNN architecture. This subsection elaborates on the construction and implementation of such gates using elementary quantum operations.

We assume the set of fundamental gates comprises the single qubit rotation gate, along with the controlled-NOT (CNOT) gate. In practice, only two of the three rotation gates are required, as the third can be constructed from the others. For instance, the $R_z(\theta)$ gate can be expressed as:

$$R_z(\theta) = R_y\left(-\frac{\pi}{2}\right) R_x(\theta) R_y\left(\frac{\pi}{2}\right). \tag{5.22}$$

All multi-controlled gates presented here are constructed from these elementary operations.

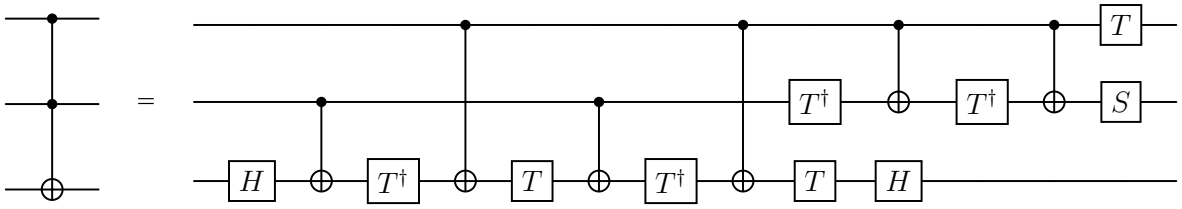


Figure 5.7: Decomposition of the Toffoli gate using H, T, S, and CNOT gates. A NOT operation is applied to the target qubit by the Toffoli gate if and only if the two control qubits being simultaneously in the $|1\rangle$ state. Adapted from [83].

The rotation gates also encompass common single-qubit gates, e.g., the Hadamard and Pauli gates. For instance, the Hadamard gate may be decomposed as:

$$H = i\, R_x(\pi) R_y\left(\frac{\pi}{2}\right), \tag{5.23}$$

up to some global phase factor $e^{i\pi/2}$. Similarly, the Pauli gates can be represented as:

$$X = i\, R_x(\pi), \quad Y = i\, R_y(\pi), \quad Z = i\, R_z(\pi), \tag{5.24}$$

again up to a global phase. In addition, two frequently used single-qubit gates are defined as follows:

$$S = e^{i\frac{\pi}{4}} R_z\left(\frac{\pi}{2}\right) = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \quad T = e^{i\frac{\pi}{8}} R_z\left(\frac{\pi}{4}\right) = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\frac{\pi}{4}} \end{bmatrix}, \tag{5.25}$$

where S is also referred to as the phase gate and T as the $\pi/8$ gate.

We now focus on the three-qubit controlled-controlled-NOT gate, also known as the Toffoli gate. Its action is to flip the target qubit when both control qubits are in the $|1\rangle$ state. Figure 5.7 presents a circuit-level implementation via only single qubit rotations and CNOT gates.

doi:10.6342/NTU202501658

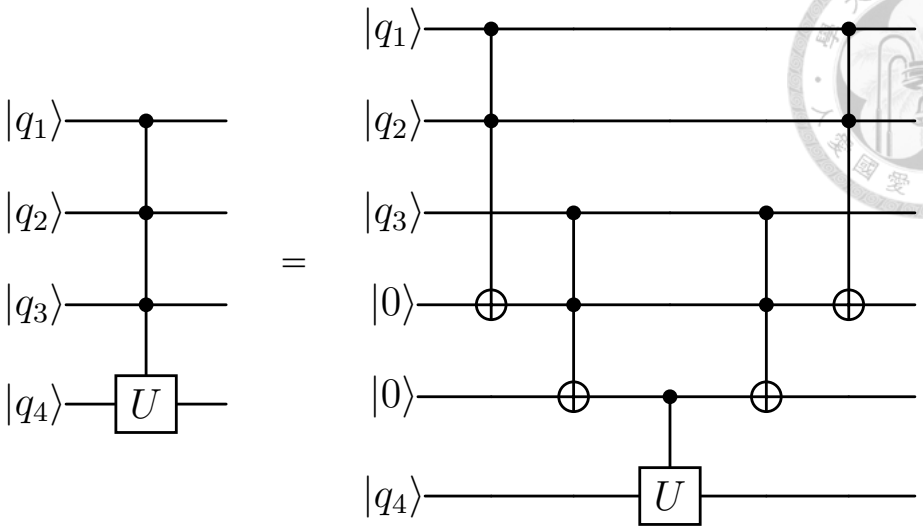


Figure 5.8: Circuit for implementing a controlled-U gate with three control qubits and one target qubit. Two ancillary qubits are used and initialized in the $|0\rangle$ state. Adapted from [83].

For controlled gates more than two control qubits, we introduce ancilla qubits (or usually known as work qubits) to store intermediate logical conditions. These ancilla qubits are initialized in the $|0\rangle$ state and are discarded after computation. The idea is to propagate the control values of control qubits on the ancilla qubits recursively using Toffoli gates. For example, the condition from q_1 and q_2 is encoded into the first ancilla qubit, then combined with q_3 to update the second ancilla qubit, and so forth. For example, Figure 5.8 illustrates the circuit for a controlled-U gate with three control qubits and one target qubit.

Consider a QCGNN with N particles, we have $n_I = \lceil \log_2 N \rceil$. The number of ancilla qubits is $n_I - 1$. Hence, we need $O(\log_2 N)$ ancilla qubits, and additional $O(N \log_2 N)$ Toffoli gates.

If the control condition is $|0\rangle$ (represented as open circles in circuit diagrams), we can apply X-Pauli gates before and after control gate to invert the control condition, as illustrated in Figure 5.9.

We now discuss the implementation of general controlled single-qubit rotations, as it is the case for QCGNN encoding ansatz. For an arbitrary single qubit gate U, it can be decomposed in the form:

$$U = e^{i\alpha} AXBXC, \quad \text{where} \quad ABC = I. \tag{5.26}$$

In this form, a controlled-U gate is able to be carried out by using two CNOT gates and three single-qubit gates, as depicted in Figure 5.10. For QCGNN cases, the multi-controlled rotation

doi:10.6342/NTU202501658

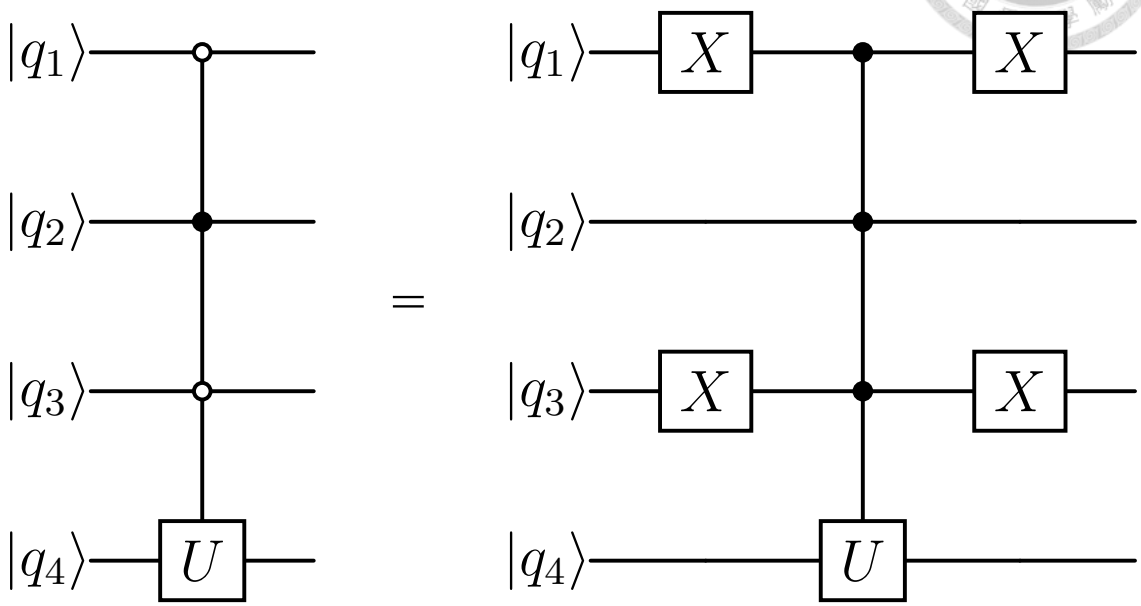


Figure 5.9: Conversion of a control condition from $|0\rangle$ to $|1\rangle$ via Pauli X-gates.

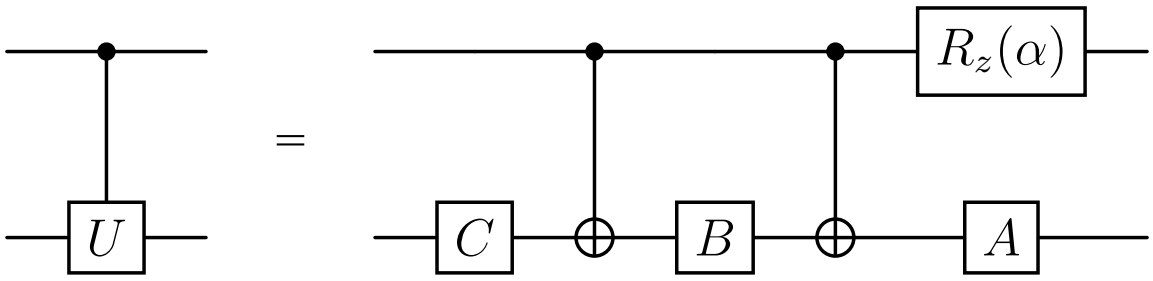


Figure 5.10: Circuit for implementing a controlled-U gate with $U = e^{i\alpha}AXBXC$ and $ABC = I$. The global phase $e^{i\alpha/2}$ is omitted. Adapted from [83].

doi:10.6342/NTU202501658

gates are used for encoding, and the corresponding U can be expressed as:

$$R_x(\theta) = R_z(-\frac{\pi}{2})R_y(\frac{\theta}{2})XR_y(-\frac{\theta}{2})XR_z(\frac{\pi}{2}) \tag{5.27}$$

$$R_y(\theta) = R_y(\frac{\theta}{2})XR_y(-\frac{\theta}{2})XI \tag{5.28}$$

$$R_z(\theta) = R_z(\theta)XR_z(-\frac{\theta}{2})XR_z(\frac{\theta}{2}). \tag{5.29}$$

In conclusion, the implementation of multi-controlled rotation gates, as required by the QCGNN encoding ansatz, is achievable using only Toffoli gates and ancilla qubits. Although further optimization may reduce gate counts, the current method is straightforward and can be done using the Toffoli gate and ancilla qubits. We leave the optimization of the QCGNN circuit for future work.

5.2 Formulating QCGNN within the MPGNN Framework

The QCGNN architecture can be mathematically expressed within the MPGNN framework. Recall that the standard MPGNN formulation consists of two key operations: node-level aggregation from neighboring nodes, and a graph-level aggregation. Analogously, the QCGNN performs double summations over particle pairs, which can be interpreted as quantum analogs of these classical operations.

For the i-th particle, the QCGNN evaluates correlations with all other particles in the graph. This pairwise interaction is captured through inner products of variational quantum states in the NR. Thus, the node-level aggregation in QCGNN corresponds to:

$$\bigoplus_{j\in\mathcal{N}(i)} \Phi\left(\mathbf{x}_i, \mathbf{x}_j\right) = \sum_{j=0}^{N-1} \left\langle \mathbf{x}_i, \boldsymbol{\theta}\right| P \left|\mathbf{x}_j, \boldsymbol{\theta}\right\rangle, \tag{5.30}$$

where the $\Phi(\mathbf{x}_i, \mathbf{x}_j)$ in Equation 3.3 is implicitly represented through the quantum state overlap under observable P, and $\mathcal{N}(i)$ includes all nodes (since the graph is complete).

The transformation γ in Equation 3.3, which maps aggregated messages to updated node embeddings, can be implemented either via classical neural networks or through quantum circuits such as variational quantum circuits.

In practice, the output of a QCGNN layer, i.e., the aggregated quantum features, can serve as input to subsequent QCGNN layers, enabling iterative message passing and capturing higher-order correlations among particles. These iterative operations mimic the classical MPGNN ar-

doi:10.6342/NTU202501658

chitecture, where successive rounds of iterations allow for more expressiveness.

Finally, a global graph-level aggregation is performed to yield a graph representation:

$$\sum_{i=0}^{N-1} \bigoplus_{j\in\mathcal{N}(i)} \Phi\left(\mathbf{x}_i, \mathbf{x}_j\right) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} \langle \mathbf{x}_i, \boldsymbol{\theta}| P |\mathbf{x}_j, \boldsymbol{\theta}\rangle = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} h(\mathbf{x}_i, \mathbf{x}_j; P), \tag{5.31}$$

which corresponds to Equation 5.8 in the QCGNN formalism. This graph-level feature can then be fed into classical post-processing networks such as fully-connected neural networks for downstream tasks including classification or regression.

5.3 Connections Between QCGNN and Kernel Methods

Kernel methods are foundational tools in theoretical machine learning, originating from the development of Support Vector Machines (SVMs). The core idea lies in the *kernel trick*, which allows algorithms to compute without explicitly computing the transformation in latent spaces that have high dimensionality. Instead, the similarity between two data $\mathbf{X}_i$ and $\mathbf{X}_j$ is captured via a kernel function $K(\mathbf{X}_i, \mathbf{X}_j)$, which computes the inner products in the implicitly defined latent space. Here, each data point $\mathbf{X}_i$ may represent either a fixed-size feature vector or a structured object such as a graph, for instance, a set of particles within a jet.

Quantum machine learning introduces an analogous idea through *quantum kernels* [84, 85], which are typically defined as the inner product of quantum states constructed by a parameterized quantum circuit, e.g., :

$$|\mathbf{X}, \boldsymbol{\theta}\rangle = U(\mathbf{X}, \boldsymbol{\theta})\,|0\rangle\,. \tag{5.32}$$

The corresponding quantum kernel is given by

$$K_Q(\mathbf{X}_i, \mathbf{X}_j) = |\langle \mathbf{X}_i, \boldsymbol{\theta} | \mathbf{X}_j, \boldsymbol{\theta}\rangle|^2 = \left|\langle 0| U^\dagger(\mathbf{X}_i, \boldsymbol{\theta}) U(\mathbf{X}_j, \boldsymbol{\theta})\,|0\rangle\right|^2. \tag{5.33}$$

This kernel can be efficiently estimated on quantum hardware, and has been shown to outperform classical kernels in certain tasks under appropriate assumptions on data distributions or complexity classes.

In contrast to SVMs, neural networks (including quantum neural networks, such as QCGNN) do not define an explicit kernel function during training. This distinction arises because SVMs are not trained through parameter tuning; once a kernel and associated hyperparameters (e.g., regularization) are specified, the optimal hyperplane is determined by solving a convex quadratic program. In neural networks, however, model behavior emerges through iterative parameter updates, making the implicit kernel difficult to identify.

doi:10.6342/NTU202501658

Despite this difference, recent theoretical work has sought to relate neural networks to kernel methods. In particular, the concept of the *Neural Tangent Kernel* introduced in [86], and follow-up work such as *Path Kernel* [87] extended this idea. According to [87], models trained using gradient-based optimization can, under certain conditions, be described by equivalent kernel methods:

$$K^g(\mathbf{X}_i, \mathbf{X}_j) = \nabla_w f_w(\mathbf{X}_i) \cdot \nabla_w f_w(\mathbf{X}_j), \tag{5.34}$$

where f_w is the model prediction, and the the tunable parameters are denoted as w.

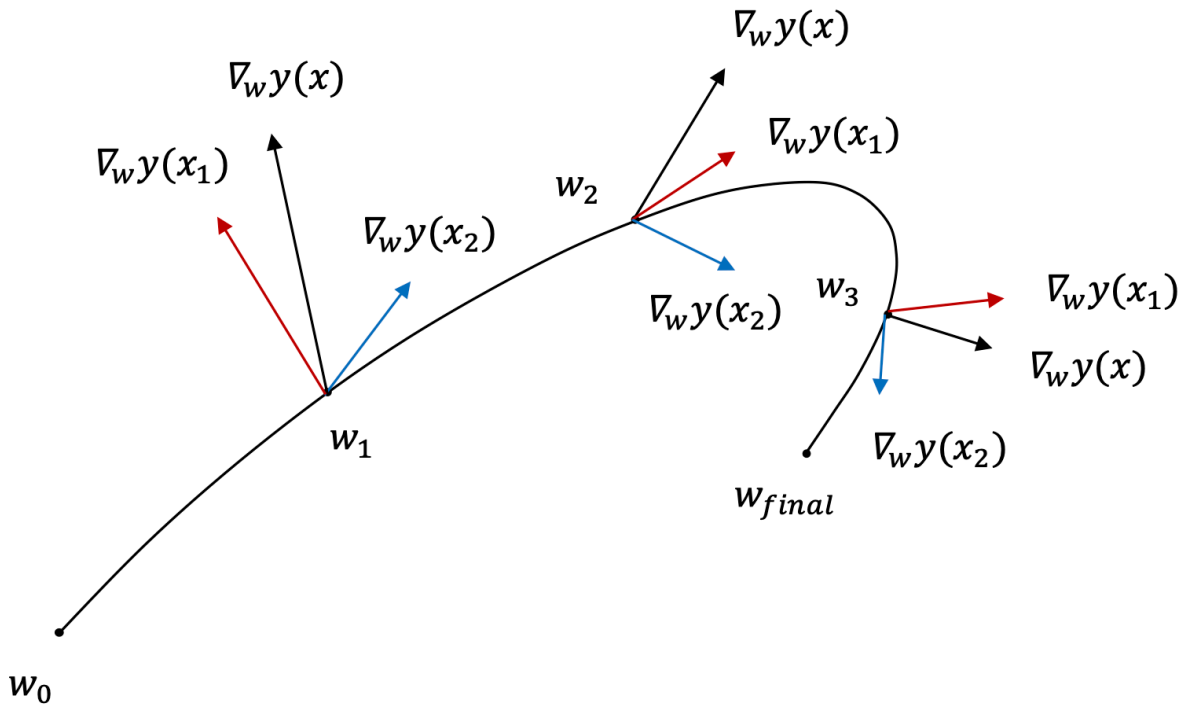


Figure 5.11: A schematic diagram illustrating the path kernel, adopted from [87]. The tangent kernel is evaluated at each point along the curve $c(t)$ in the parameter space, where t represents the optimization steps. The path kernel is defined as the integral of the tangent kernel along this trajectory. The $\nabla_w y(x)$ notation is equivalent to $\nabla_w f_w(x)$.

During the training of the model, the trajectory of w in parameter space can be viewed as a curve $c(t)$, with t representing optimization steps, as shown in Figure 5.11. The *path kernel* is then defined as an integral of the tangent kernel over the training path:

$$K^p(\mathbf{X}_i, \mathbf{X}_j) = \int_{c(t)} K^g(\mathbf{X}_i, \mathbf{X}_j) dt. \tag{5.35}$$

Within this framework, the model output y can be approximated by:

$$\lim_{\epsilon \to 0} y = y_0 - \int_{c(t)} \sum_{i=1}^{N} \frac{\partial L(y_i^*, y_i)}{\partial y_i} K^g(\mathbf{X}, \mathbf{X}_i) dt, \tag{5.36}$$

doi:10.6342/NTU202501658

To relate this to the path kernel, we define

$$\overline{L'}(y_i^*, y_i) = \frac{\int_{c(t)} K^g(\mathbf{X}, \mathbf{X}_i) \frac{\partial L(y_i^*, y_i)}{\partial y_i} dt}{\int_{c(t)} K^g(\mathbf{X}, \mathbf{X}_i) dt}, \tag{5.37}$$

we can rewrite the model output as:

$$\lim_{\epsilon \to 0} y = \sum_{i=1}^{N} \overline{L'}(y_i^*, y_i) K^p(\mathbf{X}, \mathbf{X}_i), \tag{5.38}$$

for which L is the cost function and y_i^* denotes the truth label. In particular, this approximation holds in the limit of small learning rates ϵ.

These developments provide a theoretical perspective to analyze the neural networks, including both classical and quantum models. This connection suggests a pathway for investigating the quantum advantage of QCGNN in high-energy physics tasks. Future work may leverage the kernel tricks, following the direction proposed in [88], to analyze the potential quantum advantage of QCGNN with a particular dataset. Such an analysis could shed light on when and why QCGNNs might outperform classical neural networks in tasks such as jet classification or event reconstruction.

5.4 Computational Complexity Analysis of QCGNN

Here, we conduct an analysis on the computational complexity of both QCGNN and MPGNN, for which both models represent jets through complete graphs. Consider the most trivial case that each model maps a set of vectors with d dimensions to single-dimension output, namely, $\mathbb{R}^d \to \mathbb{R}$. Furthermore, consider the case that the cost for a classical model to compute single-dimension output output is approximately in the same order to the cost for a quantum model to measure an observable, e.g., a Pauli string. Under this assumption, we demonstrate that the computational complexity of QCGNN scales as $O(N)$, whereas that of MPGNN scales as $O(N^2)$. Although this equivalence in cost is not exact, assuming a linear relation remains intuitive and reasonable, and eventually leads to the same result. Thus, after the discussion of this section, we argue that QCGNN provides a polynomial speedup over MPGNN.

Consider a jet consisting of N particles. As shown in Section 5.1.1, the uniform state preparation in QCGNN requires $O(\log_2 N)$ quantum gates. For particle encoding, an additional $O(\log_2 N)$ ancillary qubits are needed. Each particle involves multi-controlled gates, which in turn require $O(\log_2 N)$ Toffoli gates per particle, as discussed in Section 5.1.2. Consequently, the total cost of encoding all N particles is $O(N \log_2 N)$.

doi:10.6342/NTU202501658

Assume the parameterized quantum circuit employed in QCGNN has gate complexity $O(q(N))$ for some function $q(N)$. To aggregate the final result, QCGNN measures $O(2^{n_I}) \approx O(N)$ Pauli string observables, as indicated in Equation 5.10. Therefore, the overall computational cost of QCGNN is:

$$\mathcal{O}_Q = O(N) \cdot [O(N \log_2 N) + O(q(N))] \,. \tag{5.39}$$

For classical models such as MPGNN, the jet is represented as a complete graph, and obtaining all pairwise interactions involves $O(N^2)$ operations, with each pair evaluated by a neural network. Assuming a cost function $c(N)$ for obtaining a scalar output, the total computational complexity of MPGNN becomes:

$$\mathcal{O}_C = O(N^2) \cdot O(c(N)). \tag{5.40}$$

Suppose both classical and quantum models are deep networks, with MPGNN utilizing many neurons and layers, and QCGNN using deep variational circuits with large n_Q and multiple data re-uploading layers. In those cases, it is reasonable to make the assumption that

$$c(N) = \Omega(N \log_2 N), \quad q(N) = \Omega(N \log_2 N), \quad \text{and} \quad c(N) = \Theta(q(N)), \tag{5.41}$$

where Ω is the opposite of the big O notation, typically can be understood as the lower bound asymptotics, and Θ denotes asymptotic equivalence in scaling. The assumption that $c(N) = \Theta(q(N))$ is simply set for fair comparison. Under this assumption, the complexities reduce to:

$$\mathcal{O}_Q = O(N \cdot q(N)), \tag{5.42}$$

$$\mathcal{O}_C = O(N^2 \cdot c(N)), \tag{5.43}$$

and thus,

$$\mathcal{O}_Q = O(N \cdot q(N)) < \mathcal{O}_C = O(N^2 \cdot c(N)), \tag{5.44}$$

demonstrating that QCGNN is polynomially faster than MPGNN.

This assumption is justified in practical scenarios, where the network size is typically determined by the maximum particle multiplicity in a jet. For example, the number of neurons may scale as $\lceil 10 N_{\text{MAX}} \rceil$ and the number of layers as $\lceil N_{\text{MAX}}/100 \rceil$, yielding an overall complexity of $O(N_{\text{MAX}}^2)$.

In summary, assuming QCGNN is trained using deep variational quantum circuits, its computational complexity is $O(N \cdot q(N))$, compared to $O(N^2 \cdot c(N))$ for MPGNN. This indicates a polynomial speedup of QCGNN over MPGNN under reasonable scaling conditions.

doi:10.6342/NTU202501658

5.5 Extending QCGNN for Sequential and Temporal Data

The QCGNN framework can be naturally extended to handle sequential or temporal data, such as time series or event streams. A straightforward strategy is to concatenate the sequential information into a single feature vector, which can then be encoded into the NR. Here, we offer a more physical approach that leverages the inherent structure of sequential data.

A more physical and efficient approach is inspired by Recurrent Neural Networks (RNNs), in which temporal data are processed in a step-wise, recurrent manner. Recall that QCGNN employs the data-reuploading technique, whereby the input features are encoded repeatedly on the quantum states. Let $\mathbf{x}_i^{(t)}$ denote the feature vector of the i-th data at time step t, where $1 \leq t \leq T$. For instance, t may correspond to layers of detector, e.g., the hadronic calorimeter (HCAL) or electromagnetic calorimeter (ECAL), or to sequential points along a particle track within a tracking system.

To incorporate sequential data, we first generalize the standard data-reuploading formulation in Equation 5.5 as follows:

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle \left[\prod_{t=1}^{T} U_{\text{PARAM}}(\boldsymbol{\theta}^{(t)}) U_{\text{ENC}}(\mathbf{x}_i^{(t)}) \right] |0\rangle^{\otimes n_Q} . \tag{5.45}$$

At first glance, it may seem that this temporal encoding reduces model expressiveness, since it replaces the full feature reuploading with time-indexed sequential data. However, this is not necessarily a limitation. In fact, the original data-reuploading technique can be recovered by treating the entire sequence as a single unit and applying reuploading over the joint temporal structure, i.e.,

$$|\psi\rangle = \frac{1}{\sqrt{N}} \sum_{i=0}^{N-1} |i\rangle \left[\prod_{r=1}^{R} \prod_{t=1}^{T} U_{\text{PARAM}}(\boldsymbol{\theta}^{(r,t)}) U_{\text{ENC}}(\mathbf{x}_i^{(t)}) \right] |0\rangle^{\otimes n_Q} . \tag{5.46}$$

This formulation maintains the expressive capacity of QCGNN while adapting to temporally structured data. This allows QCGNN to act as a quantum analog of RNNs or even Transformer-like sequential architectures, with potential benefits where temporal or layered structure is physically meaningful.

5.6 Generalizing QCGNN to Arbitrary Graph Topologies

The QCGNN framework can be generalized to support graphs with weights. As an example, consider an undirected graph where the adjacency matrix A is defined as the Kronecker prod-

doi:10.6342/NTU202501658

uct of some weight vector $|w\rangle = \sum_i w_i |i\rangle$, resulting in edge weights of the form $A_{ij} = w_i w_j$. Note this assumption is only valid for specific situations, and the more general cases will be discussed later. In this setup, the initial quantum state in the IR can be prepared in a non-uniform superposition:

$$\sum_{i=0}^{N-1} \frac{1}{\sqrt{N}} |i\rangle |0\rangle^{\otimes n_Q} \quad \longrightarrow \quad \sum_{i=0}^{N-1} \frac{w_i}{\sqrt{\langle w|w\rangle}} |i\rangle |0\rangle^{\otimes n_Q}, \tag{5.47}$$

where the normalization factor $\langle w|w\rangle$ guarantees that the quantum states are normalized properly. As a result, the expression for the expectation value in Equation 5.8 is adjusted to:

$$\langle\psi|J \otimes P|\psi\rangle = \frac{1}{N} h(\mathbf{x}_i, \mathbf{x}_j; P) \quad \longrightarrow \quad \frac{w_i w_j}{\langle w|w\rangle} h(\mathbf{x}_i, \mathbf{x}_j; P) = \frac{A_{ij}}{\mathrm{Tr}(A)} h(\mathbf{x}_i, \mathbf{x}_j; P). \tag{5.48}$$

This updated formulation incorporates edge weight information into the quantum state by adjusting the probability amplitudes in the IR using Amplitude Embedding. In this encoding, the amplitudes reflect the structure encoded in the weight vector w.

For more general graph topologies, including directed and weighted graphs, the adjacency matrix A may be non-Hermitian. Any square matrix $A \in \mathbb{C}^{N\times N}$ can be decomposed into a Hermitian part H and a skew-Hermitian part S as follows:

$$H = \frac{1}{2}(A + A^\dagger), \quad S' = \frac{1}{2}(A - A^\dagger), \tag{5.49}$$

such that $A = H + S'$. The Hermitian part H has real eigenvalues, while S' is skew-Hermitian and has purely imaginary eigenvalues. We can further extract the imaginary i out of S' to obtain a Hermitian matrix $S = -iS'$ with real eigenvalues. According to the spectral theorem, both H and S are normal matrices and can be diagonalized via unitary transformations. Thus, the matrix A can be expanded as:

$$A = H + iS = \sum_{k=1}^{N} \lambda_k |u_k\rangle \langle u_k| + i \sum_{k=1}^{N} \mu_k |v_k\rangle \langle v_k|, \tag{5.50}$$

where $\lambda_k \in \mathbb{R}$ and $\mu_k \in \mathbb{R}$ represent the eigenvalues of H and S, respectively.

To simplify the implementation, one can choose to retain only a subset of dominant eigenbasis, e.g., those with the largest absolute eigenvalues, and subsequently discard the rest. Each surviving term corresponds to a rank-one matrix of the form $|u_k\rangle \langle u_k|$ or $|v_k\rangle \langle v_k|$, reducing the problem to the previously discussed outer-product case where the weight vector $|w\rangle = \sum_i w_i |i\rangle$. The full result can then be reconstructed by applying QCGNN separately to each eigenbasis component and summing the contributions weighted by their respective eigenvalues.

However, this approach introduces additional computational overhead, particularly from diagonalizing A and discarding a subset of eigenbasis, which may result in information loss.

doi:10.6342/NTU202501658

Moreover, preparing amplitude-encoded states typically demands substantial quantum resources, for example, extra ancilla qubits and numerous quantum gate operations, which may negate the practical benefits of using QCGNN when the number of significant eigenvalues is large.

In conclusion, while QCGNN can, in principle, be extended to general directed and weighted graphs, the practical feasibility of such implementations may be constrained by the cost of state preparation and diagonalization. Careful trade-off analysis is required to determine whether the QCGNN is suitable for general graph topologies, especially when the adjacency matrices are too complicated.

doi:10.6342/NTU202501658

Chapter 6

Benchmark on Jet Discrimination

To show the performance of QCGNN, we employ jet classification from two public datasets simulated with Monte Carlo in this chapter. The pretrained QCGNNs are then evaluated on real quantum devices provided by IBM quantum platform (IBMQ). The outcomes that obtained from classical models are treated as comparisons, including MPGNN and popular models such as ParT, PNet, and PFN.

The full code for the benchmark in this chapter, including the training and evaluation scripts, is provided on the github repository: `https://github.com/NTUHEP-QML/QCGNN`. The code is primarily implemented using the *PennyLane* [82] and *PyTorch* [89] libraries.

6.1 Monte Carlo Simulated Jet Datasets

To conduct an experiment on the performance of the QCGNN, we apply it on two public datasets with simulated events for jet classification via Monte Carlo simulations: the JetNet [90] and TopQCD dataset [37]. For these two cases, jets are constructed via the anti-k_T algorithm [35, 91], where we have set the distance parameter to be $R = 0.8$, as described in Section 2.3.

The TopQCD dataset [37] is used for binary classification, for which the main task to distinguish signals originating from top quarks (denoted as Top) from backgrounds produced in gluon and quark interactions (denoted as QCD). Those events are generated using Pythia8 [92] with its default tune at a center-of-mass energy of 14 TeV, without modeling multiple interactions or pile-up. For detector simulation, Delphes [93, 94] is used with the standard ATLAS detector card. Only the leading jet in each event is retained, and the transverse momentum of the jet is selected to lie within the range $[550, 650]$ GeV. For each jet, the top 200 constituent particles (sorted by p_T) are saved. Jets with particles less than two-hundred constituents are zero-padded by the original dataset authors. The dataset includes 1.2M training events, and for both the

doi:10.6342/NTU202501658

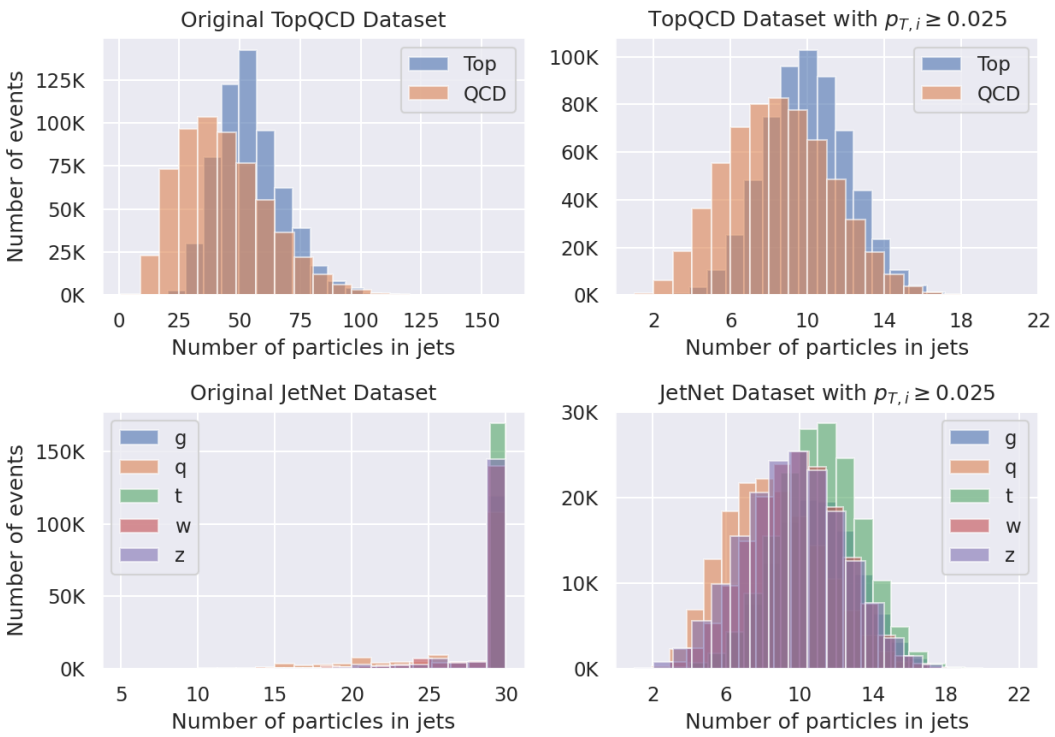


Figure 6.1: The histogram of the number of particles in the jets of the TopQCD dataset and the JetNet dataset. The momentum threshold for each particle is set to $p_{T,i} \geq 0.025 p_T^{\text{jet}}$, where p_T^{jet} is the jet's transverse momentum.

validation and testing events, each contains 400K samples. Several samples of jets are shown in Figure 6.2, while jet and particle histograms of the full dataset are shown in Figures 6.3 and 6.4. Further details can be found in [95].

The JetNet dataset [90] is designed for multi-class classification, with jets generating from gluons (g), light quarks (q), top quarks (t), Z bosons (z), and W bosons (w). Events are generated at leading order using MadGraph5 [96], with parton showering and hadronization performed by Pythia8 [92]. Each jet class consists of approximately 170 thousand events with transverse momenta centered around 1 TeV. In this study, we select events with jet p_T within the range $[800, 1200]$ GeV. Only the thirty particles in each jet that have the highest p_T are used. Some events are displayed in Figure 6.5, with histograms of the full dataset given in Figures 6.6 and 6.7. Additional information is provided in [97].

To focus on evaluating QCGNN, we restrict our analysis to the particle flow observables defined in Section 2.4. Additional features such as electric charge, mass, or particle identification are not utilized. The jets are represented with fully connected graphs, for which every node is treated as a particle, and edges exist between all pairs. To suppress p_T^{jet} dependence, we uniformly sample jets from ten equal-width bins within the $[550, 650]$ GeV range for TopQCD and $[800, 1200]$ GeV for JetNet.

doi:10.6342/NTU202501658

Since our models are graph-based, zero-padding is no longer necessary. However, because QCGNN is simulated on classical hardware, we apply an additional transverse momentum threshold for particles within jets to reduce computational cost. Specifically, we remove particles with $p_{T,i} < 0.025\, p_T^{\text{jet}}$. This significantly reduces the number of particles per jet to approximately 6–14, as shown in Figure 6.1, enabling faster simulation and training while preserving most of the relevant physical information. The transverse momentum threshold will be justified in Section 6.3.1 after we discuss the models for benchmark.

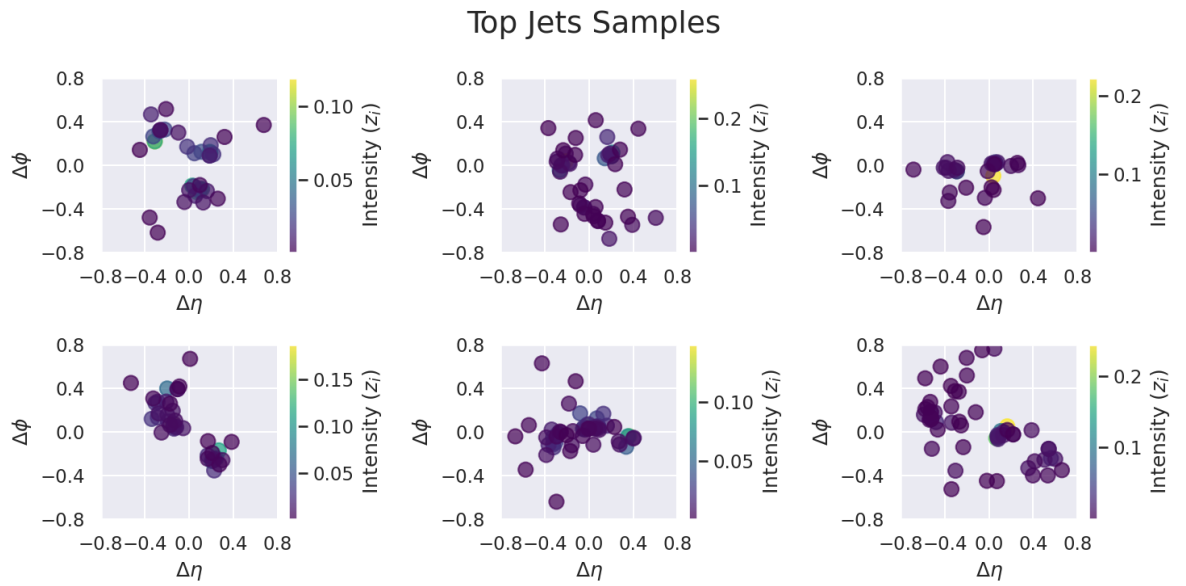


Figure 6.2: Samples in TopQCD dataset: (a) Top jets. The figure continues with the next figure.

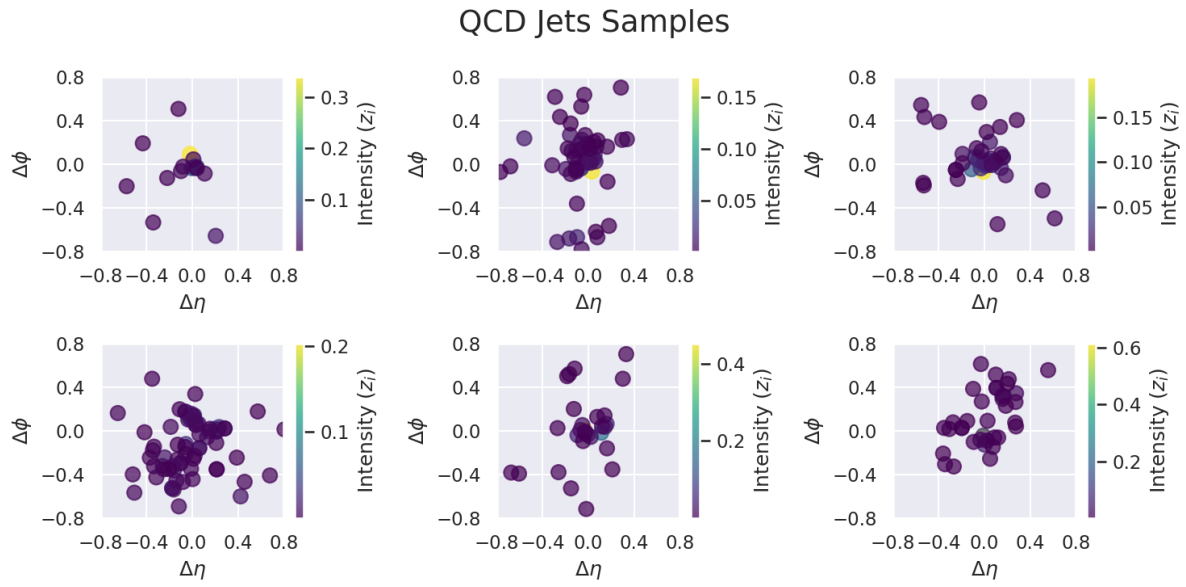


Figure 6.2: (b) QCD jets, from light quarks and gluons. The axes ranges are $[-0.8, 0.8]$, since $R = 0.8$ was used for jet clustering.

doi:10.6342/NTU202501658

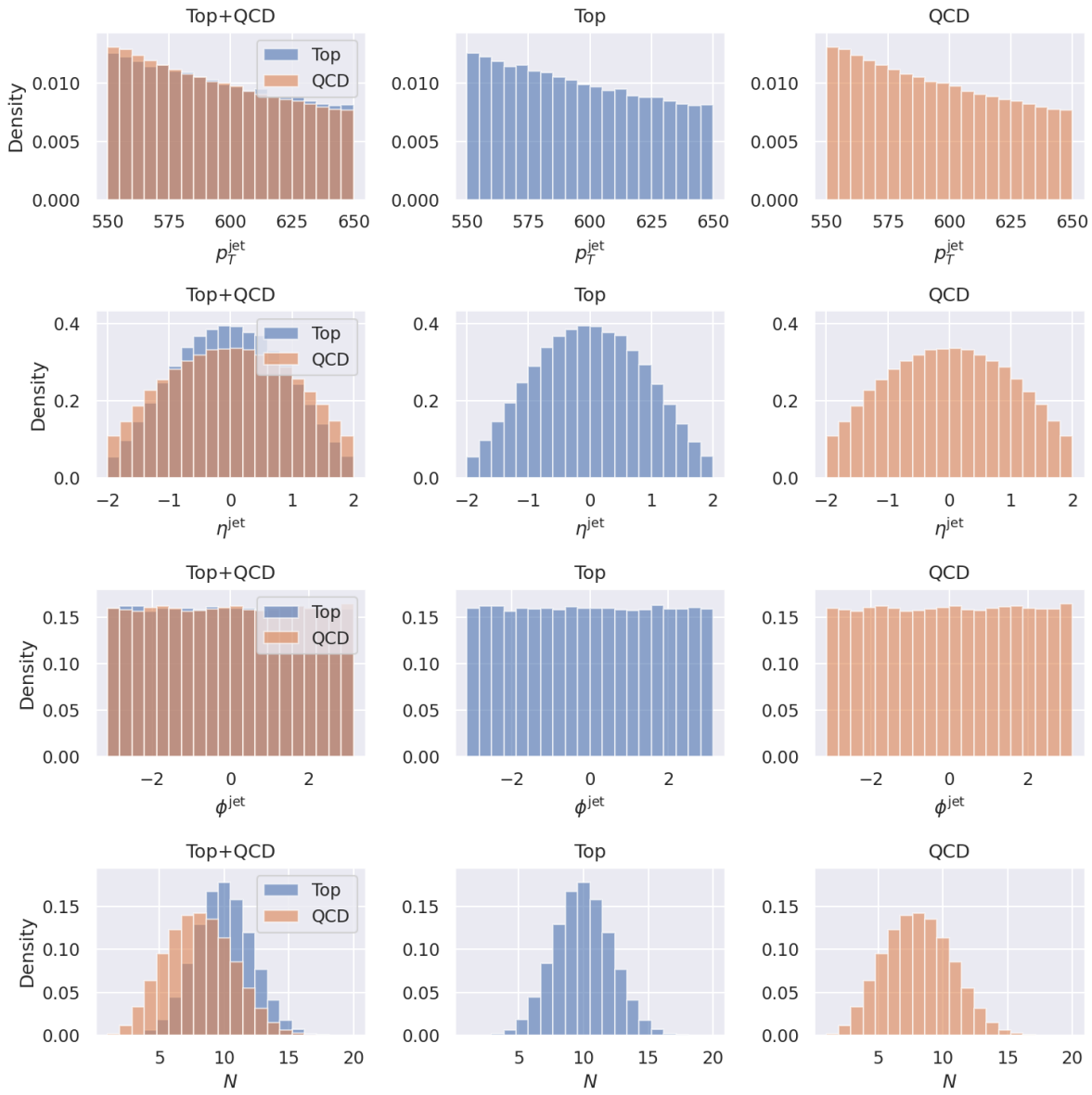


Figure 6.3: Histograms of the particle flow information of jets in TopQCD dataset. The jet's azimuthal angle, pseudorapidity, and transverse momentum are denoted by ϕ^{jet}, η^{jet}, and p_T^{jet}, respectively. The N is used for denoting number of particles in the jet. The histograms are shown in density, namely, the area under the histogram is equal to 1.

doi:10.6342/NTU202501658

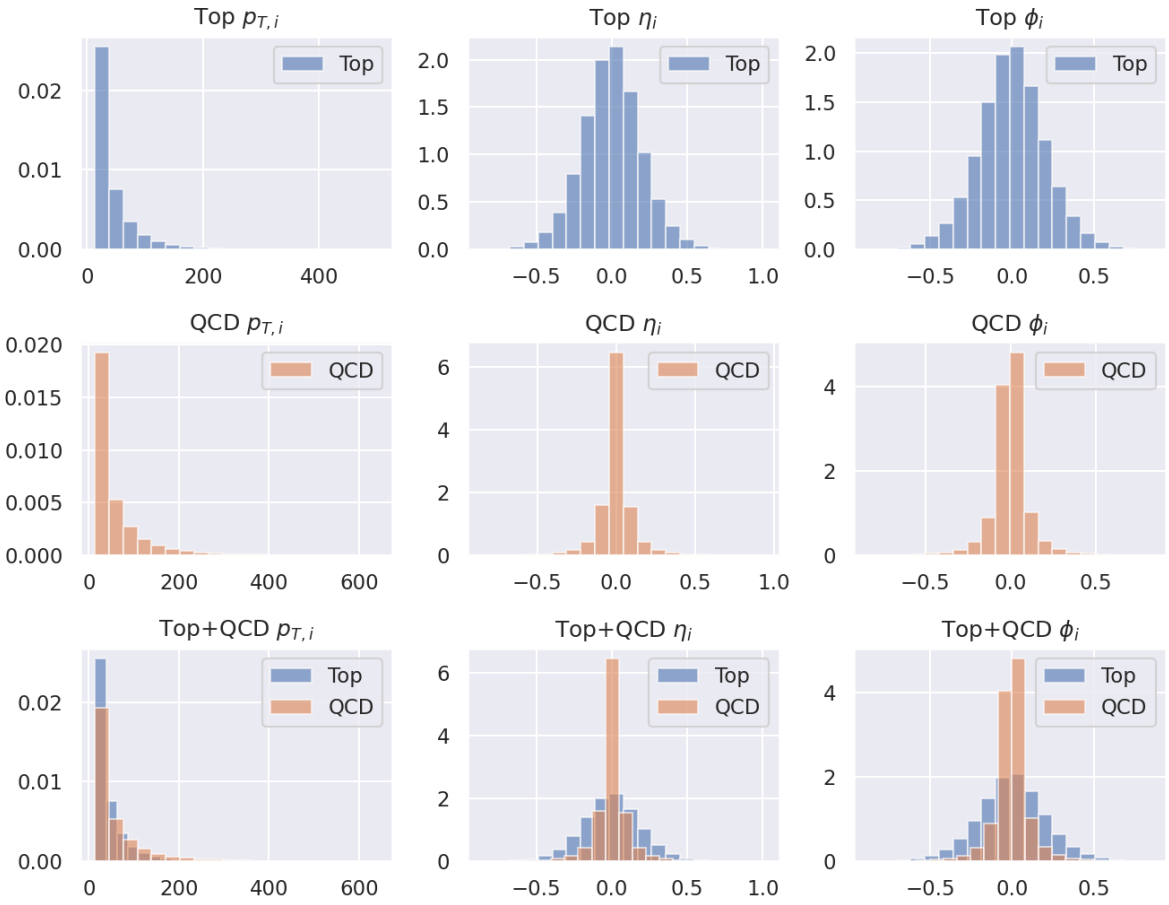


Figure 6.4: Histograms of the particle flow information of particles within jets in TopQCD dataset. The η_i, ϕ_i, and $p_{T,i}$ are the pseudorapidity, azimuthal angle, and transverse momentum of the particles, respectively. The signal and background are denoted as Top and QCD, respectively. The histograms are shown in density, namely, the area under the histogram is equal to 1.

doi:10.6342/NTU202501658

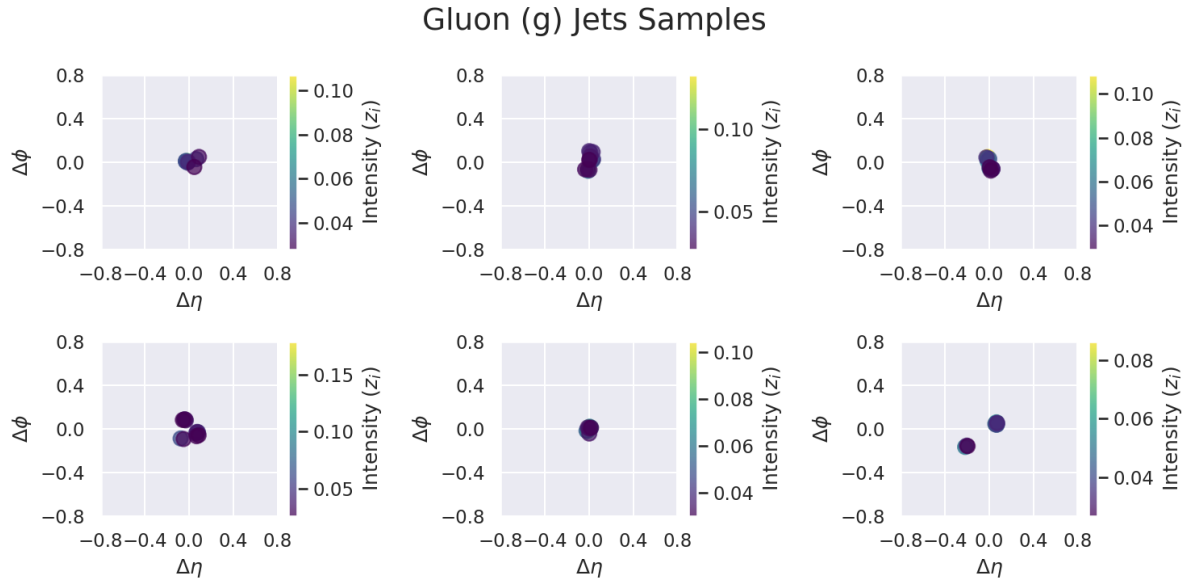


Figure 6.5: Samples of the JetNet dataset: (a) Gluon jets. The figure continues on the next figure.

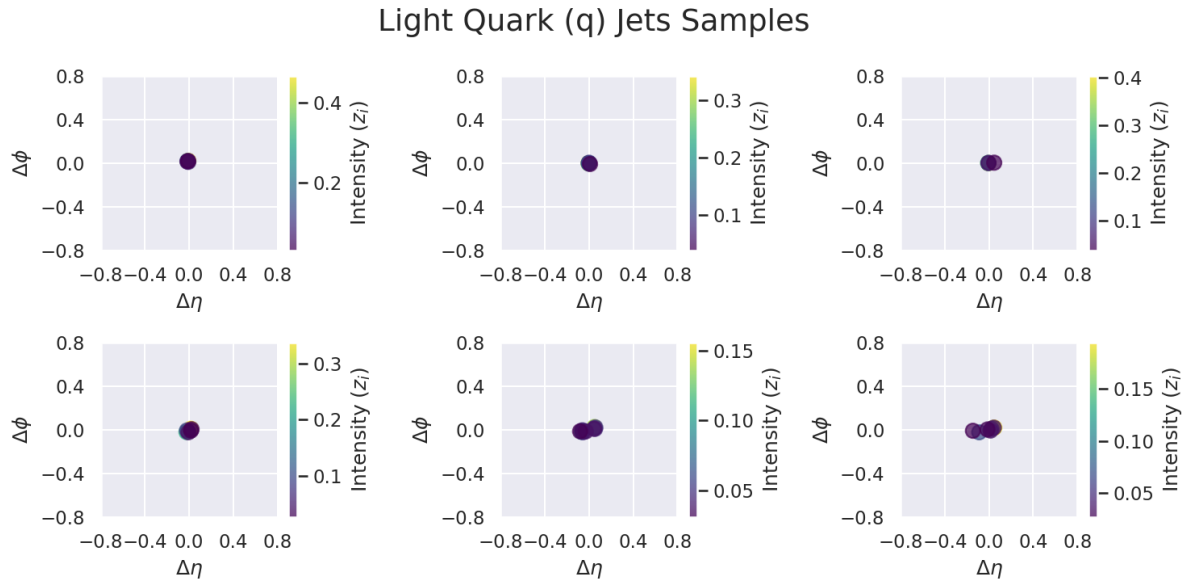


Figure 6.5: (b) Light quark jets. The figure continues on the next page.

doi:10.6342/NTU202501658

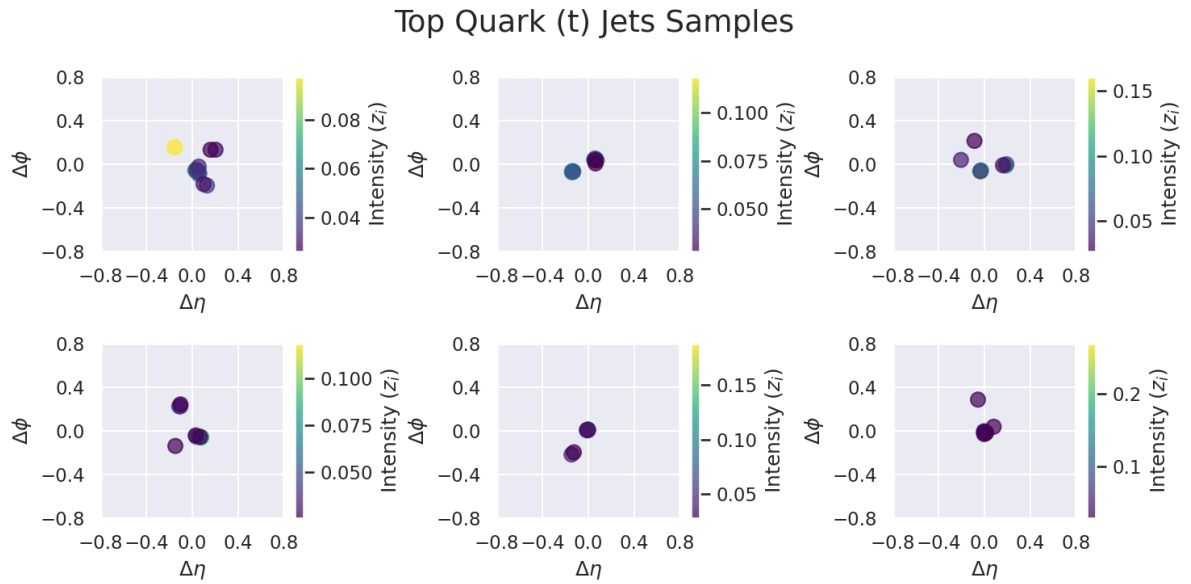


Figure 6.5: (c) Top quark jets. The figure continues on the next figure.

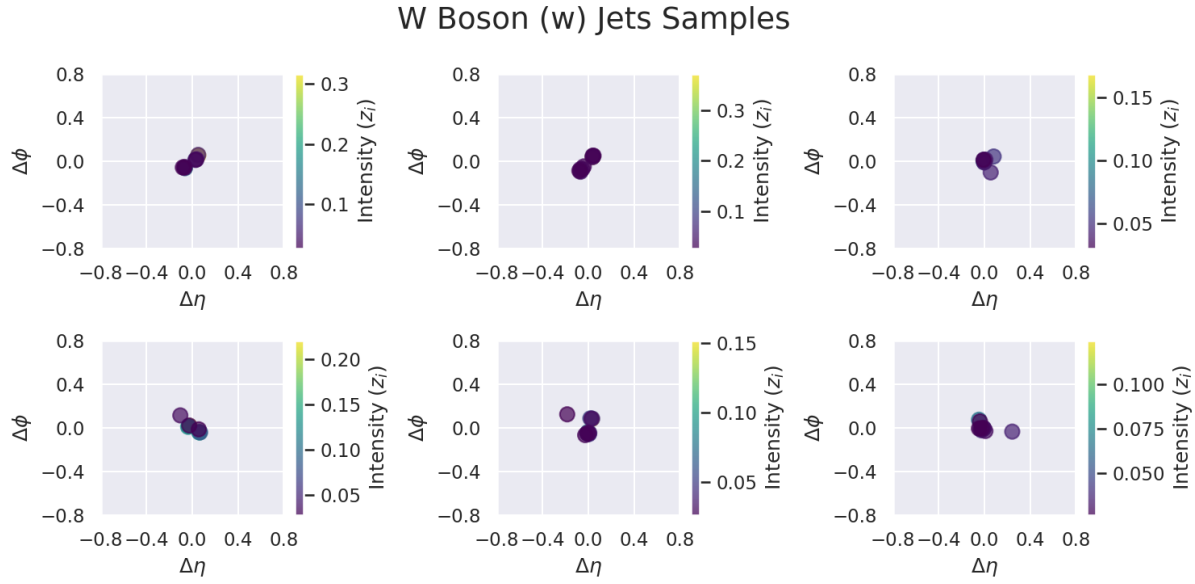


Figure 6.5: (d) W-boson jets. The figure continues on the next page.

doi:10.6342/NTU202501658

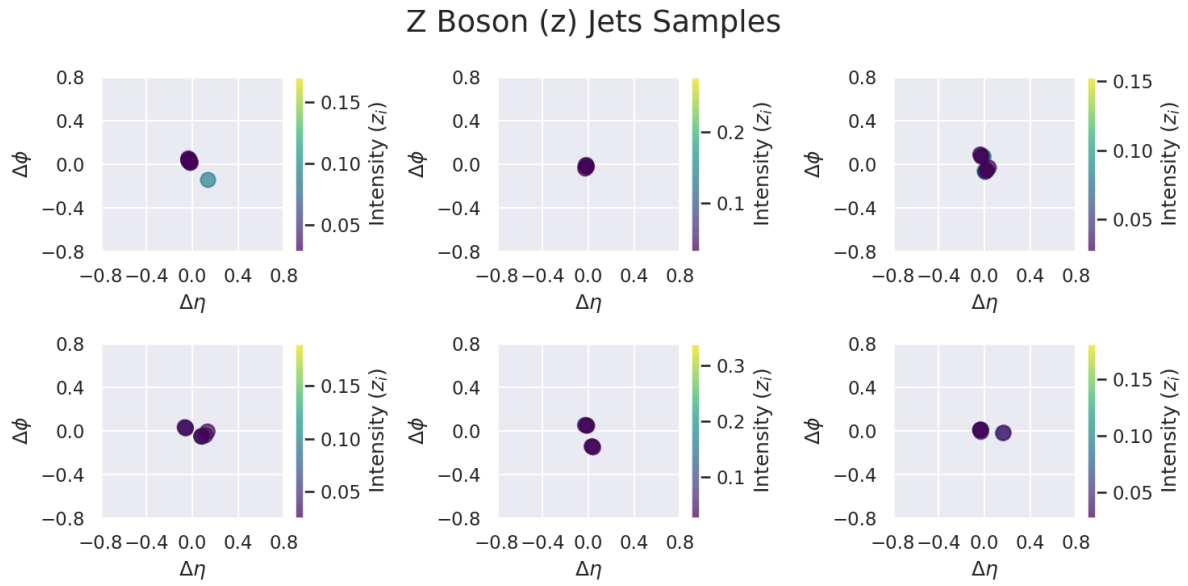


Figure 6.5: (e) Z-boson jets. The axes ranges are $[-0.8, 0.8]$, since $R = 0.8$ was used for jet clustering.

doi:10.6342/NTU202501658

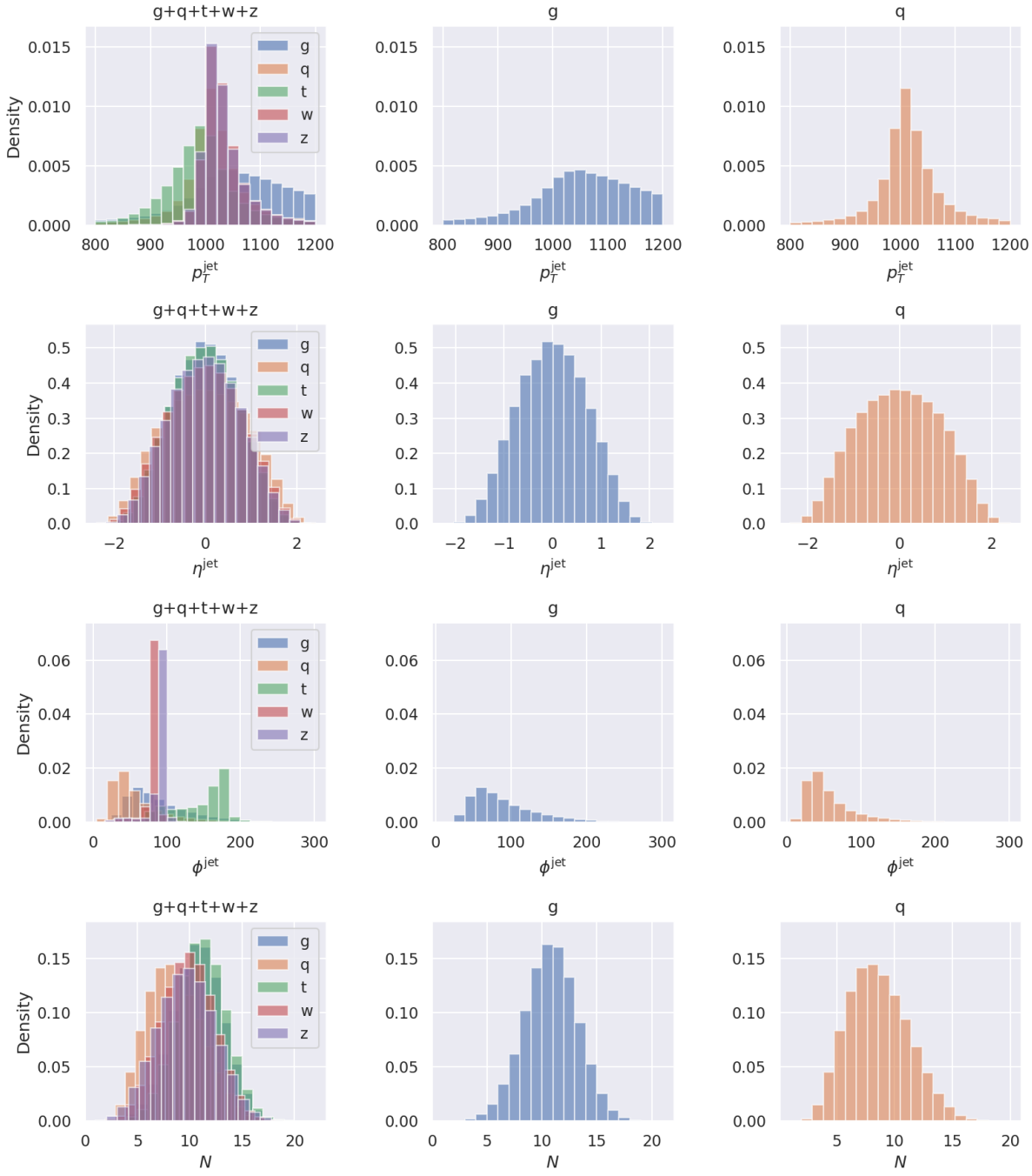


Figure 6.6: (a) Histograms of the particle flow information of jets in JetNet dataset, generated from gluons (g) and light quarks (q). The figure continues on the next page.

doi:10.6342/NTU202501658

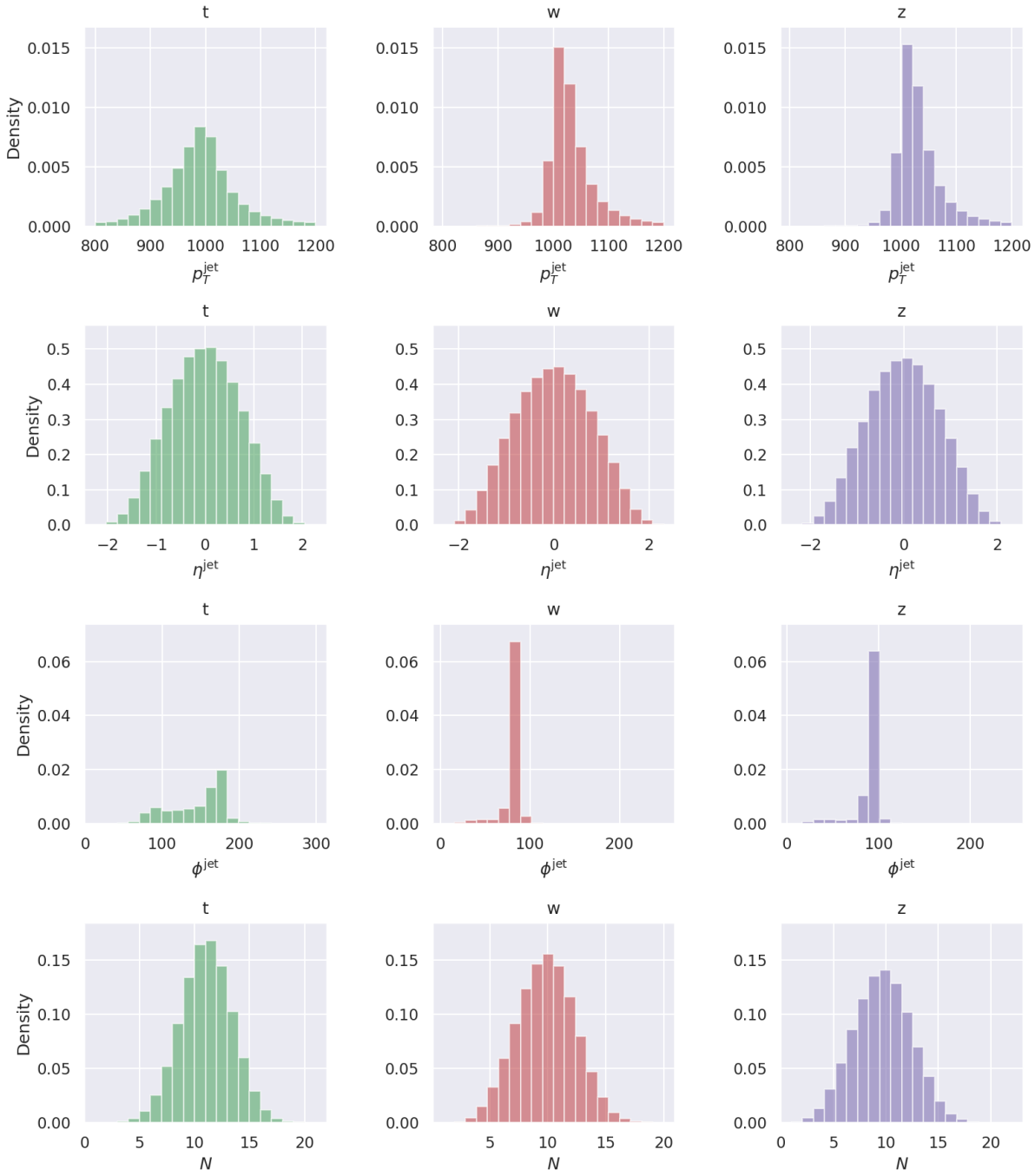


Figure 6.6: (b) Histograms of the particle flow information of jets in JetNet dataset, generated from top quarks (t), Z-bosons (z), and W-bosons (w). The jet's azimuthal angle, pseudorapidity, and transverse momentum are denoted as ϕ^{jet}, η^{jet}, and p_T^{jet}, respectively. The N is used for denoting the number of particles in the jet. The histograms are shown in density, namely, the area under the histogram is equal to 1.

doi:10.6342/NTU202501658

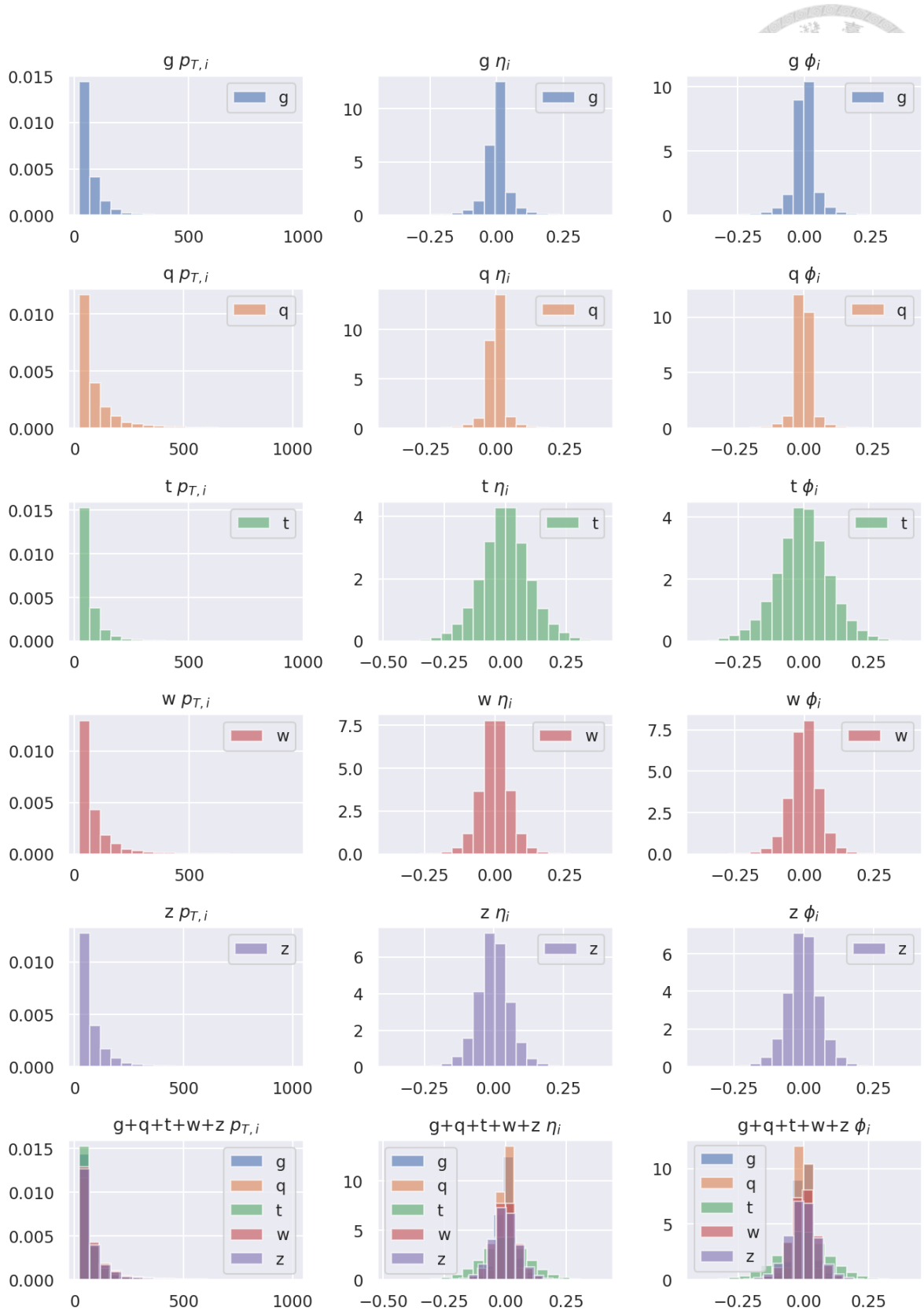


Figure 6.7: Histograms of the particle flow information of particles within jets in JetNet dataset. The η_i, ϕ_i, and $p_{T,i}$ are the pseudorapidity, azimuthal angle, and transverse momentum of the particles, respectively. The five classes are denoted as z (Z-bosons), q (light quarks), t (top quarks), g (gluons), and w (W-bosons). The histograms are shown in density, namely, the area under the histogram is equal to 1.

doi:10.6342/NTU202501658

6.2 Classical and Quantum Models

The MPGNN is chosen as the benchmark classical model for the QCGNN, since their formulation is close and simple. In this section, we first discuss the setup of the MPGNN and the QCGNN, and how the hyperparameters are determined. Then to demonstrate how the classical models nowadays perform on the same datasets, we discuss the setup for the models introduced in Section 3.4, including Particle Transformer (ParT), Particle Net (PNet), and Particle Flow Network (PFN).

6.2.1 QCGNN and MPGNN Setup

In this subsection, we give the detail implementation on the structure of both QCGNN and MPGNN in detail and highlight the structural similarities between them. To ensure a meaningful comparison, we configure the hyperparameters of each model so that their parameter counts are on a similar scale.

QCGNN Structure

Since the number of particles per jet is reduced to approximately $6-14$, we restrict our dataset to jets with $4 \leq N \leq 16$ particles. Accordingly, the index register (IR) requires $n_I = \lceil \log_2 16 \rceil = 4$ qubits. For the network register (NR), we consider two configurations: $n_Q = 3$ and $n_Q = 6$.

The encoding ansatz employs R_x and R_y rotation gates, where the mathematical expression can be found in Equation 4.16. Figure 6.8 shows an example of this encoding for $n_Q = 4$, where the rotation angles are functions of the input features.

For the parameterized ansatz, we adopt the strongly entangling layers proposed in [98], illustrated in Figure 6.9. Each layer consists of general rotation gates $R(\phi, \theta, \omega)$ as defined in Equation 4.17, along with a full entanglement pattern using CNOT gates. The strongly entangling ansatz is typically repeated multiple times to increase expressiveness.

The complete QCGNN architecture is constructed with the components below:

1. A linear layer mapping the 3-dimensional particle flow features to a $3n_Q$-dimensional vector. An additional arctan activation function is applied for ensuring the values lie between $[-\frac{\pi}{2}, \frac{\pi}{2}]$.

2. The QCGNN module, which encodes the transformed features using the ansatz structure

doi:10.6342/NTU202501658

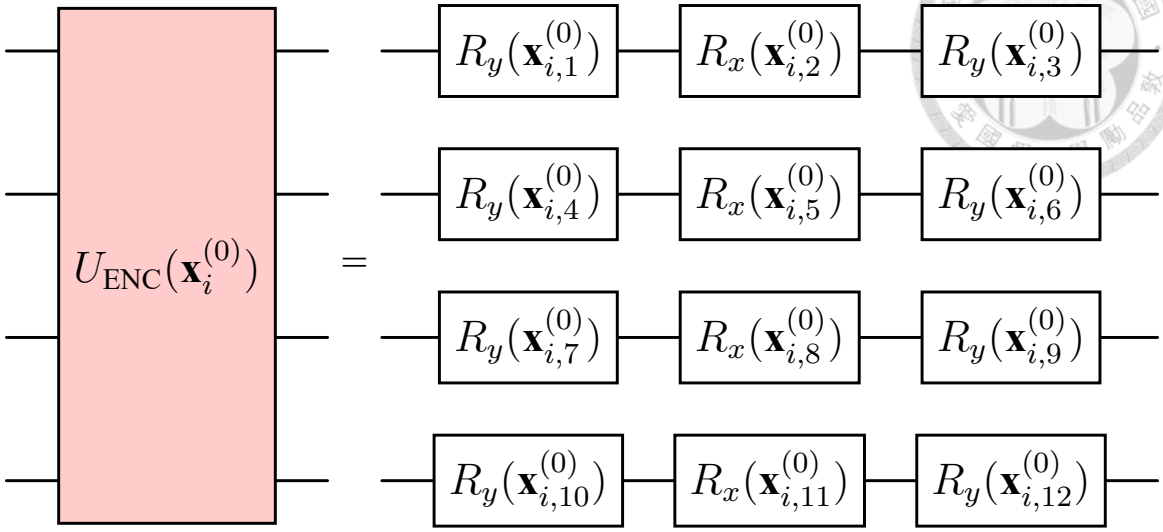


Figure 6.8: An illustration of the ansatz for QCGNN data encoding with $n_Q = 4$ qubits. Classical features are embedded into quantum states using rotation gates described in Equation 4.16. The $\mathbf{x}_{i,j}^{(0)}$ is taken as the angle for rotation gates, where the (i, j) denotes the indices for the particle and the corresponding feature, respectively, where the input features can either be raw particle flow variables or representations transformed by prior linear layers.

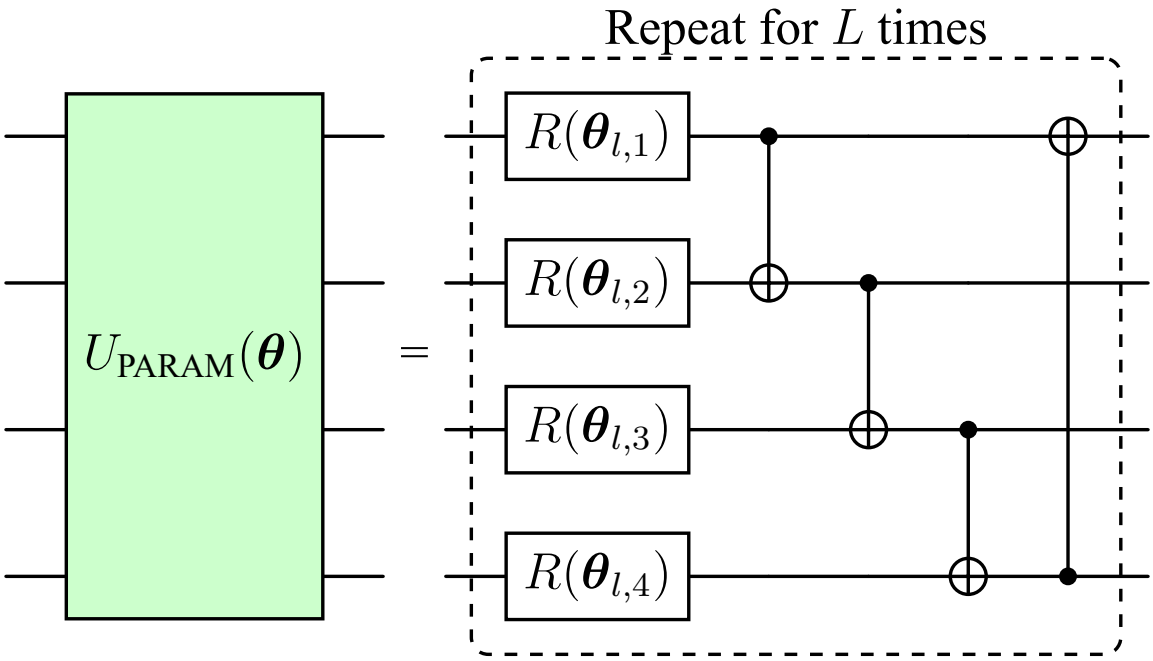


Figure 6.9: Strongly entangling ansatz [98] used in QCGNN with $n_Q = 4$ qubits. Each qubit is acted upon by the three-parameter rotation gate $R(\phi, \theta, \omega)$ defined in Equation 4.17. Trainable parameters $\boldsymbol{\theta}_{l,j}$ vary with repetition index l and qubit index j, where $1 \leq j \leq 4$, and each $\boldsymbol{\theta}_{l,j}$ contains the three angles for the R gate. The ansatz scales naturally for $n_Q \geq 3$; for smaller systems, alternative circuit designs are recommended to maintain sufficient entanglement.

doi:10.6342/NTU202501658

shown in Figure 6.8. The variational quantum circuit uses the strongly entangling ansatz shown in Figure 6.9.

3. The measurement observables for IR and NR are chosen as J and Pauli-Z, respectively. The J observable is decomposed into Pauli-X, as described in Equation 5.10.

4. Two additional dense layers are used: the first contains 16 hidden units, and the second has n_C-dimension output, which is the number of classes.

To be more specific, each Pauli-Z observable is treated as an output feature. For a QCGNN with n_Q network register (NR) qubits, the output is represented as $\mathbf{x}^Q \in \mathbb{R}^{n_Q}$, where the q-th component is computed as:

$$\begin{aligned} \mathbf{x}_q^Q &= N\left[\langle\psi|J \otimes Z_q^{\mathrm{NR}}|\psi\rangle - \langle\psi|Z_q^{\mathrm{NR}}|\psi\rangle\right] \\ &= \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} h(\mathbf{x}_i^{(0)}, \mathbf{x}_j^{(0)}; Z_q^{\mathrm{NR}}) - \sum_{i=0}^{N-1} h(\mathbf{x}_i^{(0)}, \mathbf{x}_i^{(0)}; Z_q^{\mathrm{NR}}), \end{aligned} \tag{6.1}$$

where Z_q^{NR} refers to the Pauli-Z operator applied specifically to the q-th qubit in the NR.

This setup effectively mirrors a classical feedforward neural network, where the final layer consists of n_Q output neurons. In the next subsection, we demonstrate how Equation 6.1 closely resembles the classical expression in Equation 6.2, thereby highlighting the permutation invariance property of the QCGNN output.

MPGNN Structure

The MPGNN architecture follows Equation 3.3. For each particle pair (i, j), the features are concatenated and passed through fully-connected layers with ReLU activation functions [99]. The γ transformation is chosen to be the identity function, i.e., no further transformation is applied after node aggregation. We adopt global sum pooling for graph aggregation, followed by two additional linear layers, consistent with the QCGNN architecture.

The overall feature of the graph denoted as $\mathbf{x}^C$ (the output of MPGNN graph aggregation),

doi:10.6342/NTU202501658

is calculated by:

$$\begin{aligned}\mathbf{x}^C &= \sum_{i=0}^{N-1} \mathbf{x}_i^{(1)} \\ &= \sum_{i=0}^{N-1} \sum_{j \neq i} \Phi(\mathbf{x}_i^{(0)}, \mathbf{x}_j^{(0)}) \\ &= \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \Phi(\mathbf{x}_i^{(0)}, \mathbf{x}_j^{(0)}) - \sum_{i=0}^{N-1} \Phi(\mathbf{x}_i^{(0)}, \mathbf{x}_i^{(0)}).\end{aligned} \tag{6.2}$$

As described above, the Φ function corresponds to the fully-connected layers with ReLU activation functions. The inputs of Φ is trivially constructed by concatenating $\mathbf{x}_j^{(0)}$ and $\mathbf{x}_i^{(0)}$, where in the Φ we only need 6 neurons for the first linear layer (each particle has 3 particle flow features).

The structure of MPGNN resembles the PFN model, which will be introduced in the next subsection. The only distinction is that PFN transforms each particle features individually, whereas MPGNN concatenates the particle features and calculates the pairwise information $\Phi(\mathbf{x}_i^{(0)}, \mathbf{x}_j^{(0)})$ between particles.

6.2.2 Number of Parameters in QCGNN and MPGNN

Assume that the parameterized quantum circuit U_{PARAM} consists of L_Q repetitions of the strongly entangling ansatz shown in Figure 6.9, and the number of data-reuploading (including the initial encoding) is L_R. Including the parameters from the initial linear transformation layer, the overall amount of trainable parameters in QCGNN can be computed by:

$$N_Q = 3n_Q L_R L_Q + 3(n_Q + 1), \tag{6.3}$$

where the first term accounts for the variational quantum circuit, and the second term corresponds to the initial linear layer that transforms input features.

For MPGNN, assume each hidden layer contains n_M neurons, and there are L_C hidden layers in total. The number of trainable parameters is:

$$\begin{aligned}N_C &= 6(n_M + 1) + n_M L_C (n_M + 1) \\ &= L_C n_M^2 + (6 + L_C) n_M + 6.\end{aligned} \tag{6.4}$$

To make sure the comparison is fair to some degree, we set the output dimensions of QCGNN and MPGNN, and use identical final linear layers. $n_Q = n_M = D$ is chosen and constrain n_Q to be multiples of 3 such that $L_Q = D/3$. Substituting these into above expres-

doi:10.6342/NTU202501658

sions gives:

$$N_C = L_C D^2 + (6 + L_C)D + 6, \tag{6.5}$$

$$N_Q = L_R D^2 + 3D + 3. \tag{6.6}$$

If we further assume $L_C = L_R = L$, then both QCGNN and MPGNN have comparable complexity, scaling as $O(LD^2)$.

In this study, we choose $L = 2$ and $D \in \{3, 6\}$. This choice reflects a balance between training efficiency and classification performance, especially given that QCGNN training is conducted on classical simulators, which are computationally expensive.

In summary, Figures 6.10 and 6.11 illustrate the model structures and hyperparameter settings of QCGNN and MPGNN, respectively.

6.2.3 State-of-the-art Classical Models

To estimate the upper bound of neural network performance on the jet datasets, we benchmark three advanced classical models: PFN (Section 3.4.1), PNet (Section 3.4.2), and ParT (Section 3.4.3). The architectures are adapted from their original papers, with minor modifications to accommodate the specific jet datasets used in this work. In this section, we describe the modifications made to each model and outline their configurations.

- **Structure of Particle Flow Network (PFN)**

  Figure 3.4 depicts the overall design of PFN. The input features have dimension 3, corresponding to the particle flow observables. A linear layer with 100 neurons is applied, followed by another linear layer that maps to a 256-dimensional latent space. The particle-wise features are aggregated using a **SUM** pooling operation. The aggregated graph feature is subsequently passed through a multilayer perceptron, where the first hidden layer contains 100 neurons and the final layer outputs a vector of dimension n_C, corresponding to the number of classes. ReLU activation functions [99] are used between all linear layers.

- **Structure of ParticleNet (PNet)**

  Figure 3.5a shows the detailed configuration of PNet. The only modification is that, given the relatively small amount of particles per jet (at most 16), the number of nearest neighbors for dynamic graph convolution is reduced to $k = 3$.

- **Structure of Particle Transformer (ParT)**

doi:10.6342/NTU202501658

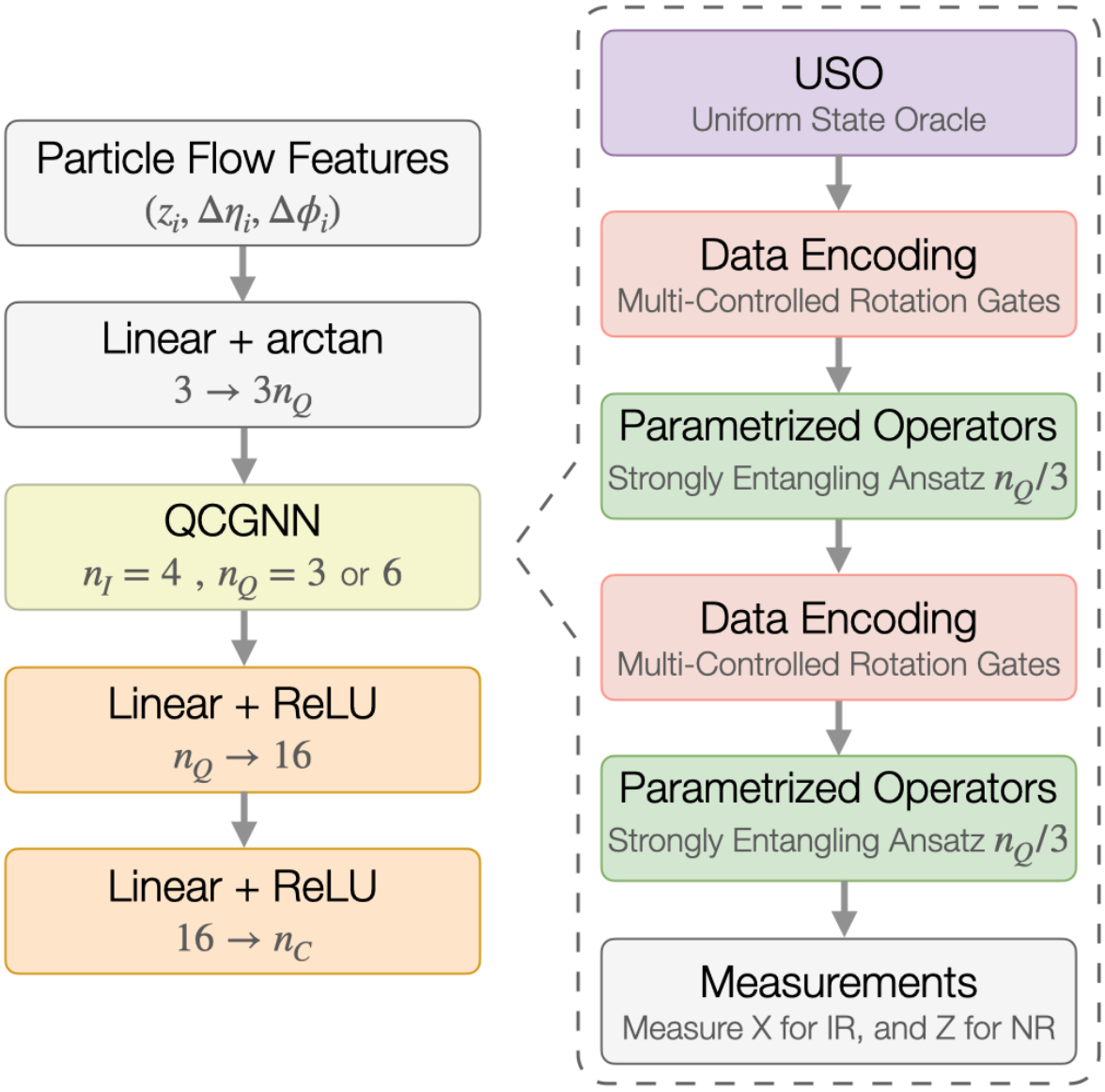


Figure 6.10: Data reuploading is applied twice (including the initial encoding), and the strongly entangling ansatz is repeated $n_Q/3$ times. Two additional linear layers are appended, with the final layer maps to n_C classes.

doi:10.6342/NTU202501658

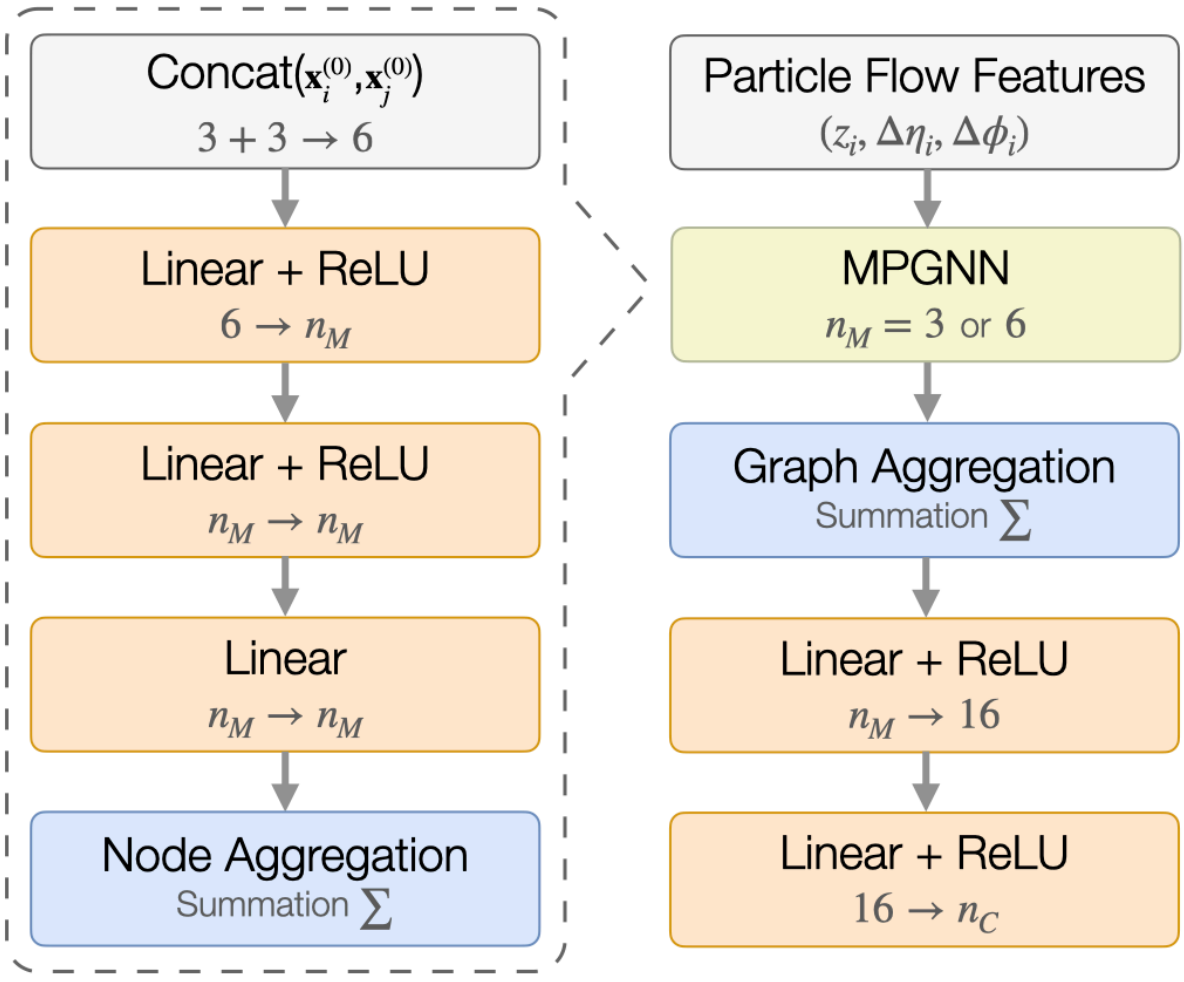


Figure 6.11: For each particle pair, their features are concatenated and passed through a multilayer perceptron. For both models, Two additional linear layers are appended, with the final layer maps to n_C classes.

doi:10.6342/NTU202501658

The architecture of ParT is depicted in Figure 3.6. Following the original design in [56], an interaction matrix is constructed, where for each particle pair (i, j), additional features are defined as:

$$\Delta_{i,j} = \sqrt{(\Delta\phi_i - \Delta\phi_j)^2 + (\Delta\eta_i - \Delta\eta_j)^2}, \tag{6.7}$$

$$k_{T,i,j} = \min(p_{T,i}, p_{T,j})\,\Delta_{i,j}, \tag{6.8}$$

$$z_{i,j} = \frac{\min(p_{T,i}, p_{T,j})}{p_{T,i} + p_{T,j}}, \tag{6.9}$$

$$m^2_{i,j} = (E_i + E_j)^2 - \|\mathbf{p}_i - \mathbf{p}_j\|^2, \tag{6.10}$$

where p_T, $\Delta\eta$, and $\Delta\phi$ are the particle flow features, and E is the particle energy. Since our datasets do not provide energy information, the $m^2_{i,j}$ term is omitted from the interaction matrix. Apart from this adjustment, the rest of the model follows the original ParT implementation.

6.3 Training Results of Jet Discrimination

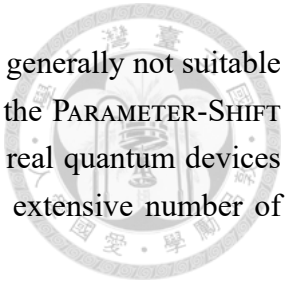
We trained the models using five different random seeds, with each training run lasting 30 epochs. For data preparation, we selected 25,000 samples per class for training and 2,500 samples per class for both validation and testing. The TopQCD dataset is formulated as a binary classification task involving two classes, whereas the JetNet dataset is structured as a classification task with five categories. The number of samples per class was chosen to balance computational cost and demonstrate the capabilities of QCGNN, as detailed in Section 6.3.1.

For the model outputs, binary classification on TopQCD employs a single output unit activated by the Sigmoid layer, namely, $S(x) = \frac{1}{1+e^{-x}}$. Eventually, the loss function is defined and optimized using binary cross-entropy loss. In contrast, the JetNet model uses an output layer with five units and applies the Softmax function, with training guided by the multi-class cross-entropy loss.

We implement the classical models in the framework of *PyTorch* [89], and some GNNs are specifically using *PyTorch Geometric* [100]. For the simulations of QCGNN were performed through the QML package: *PennyLane* [82]. For all models, the Adam optimizer [101] was optimized by a constant learning rate of 0.001. The size of each training batch was chosen to be 64, mainly limited by memory requirements of quantum circuit simulations.

Computing gradients in quantum neural networks differs substantially from their classical

doi:10.6342/NTU202501658



counterparts. Standard techniques, such as finite difference methods, are generally not suitable for quantum hardware. Instead, gradient estimation is performed using the PARAMETER-SHIFT RULE (PSR), as explained in Section 4.3.2. However, applying PSR on real quantum devices remains challenging due to hardware noise, long queue times, and the extensive number of measurements required for stable gradient estimates.

Given these constraints in the NISQ era [102], QCGNN models were trained on classical hardware using noiseless quantum simulators provided by *PennyLane*. While GPU acceleration is supported, noticeable performance improvements typically occur only when simulating more than around 20 qubits. Since our models operate on fewer qubits, all simulations were run on CPUs.

Despite using classical resources, quantum simulations remain computationally demanding. As an illustration, training a QCGNN with a batch size of 64 and 10 qubits $(n_I, n_Q) = (4, 6)$ on ten thousands samples required roughly one thousand seconds per epoch. Training across thirty epochs with 5 different random seeds thus costed to almost 30 days of computation time.

6.3.1 Justification of the Transverse Momentum Threshold

In this subsection, we argue that the current dataset size provides a reasonable basis for assessing the performance of each model. In our study, we trained several advanced classical models, including ParT, PNet, PFN, and MPGNN, all configured with 64 hidden units denoted as n_M.

To evaluate the models, we used the TOPQCD and JETNET datasets, where each model was trained using various numbers of training examples per class and repeated with five different random seeds. The corresponding results are shown in Figures 6.12 and 6.13. We only considered events containing 4 to 16 particles, except in the 'Full-100K' setting, which includes all particle information without applying a transverse momentum cut.

We observed that model performance tends to stabilize when the number of training samples per class is between 25K and 50K. This suggests that choosing 25K samples per class, as done in Section 6.3, offers a good balance between training efficiency and performance. For datasets that retain all particles without p_T filtering, we used 100K samples per class for training. According to our results, the simplest model, MPGNN, achieved the best performance when fewer particles are used for training. On the other hand, when all available particle information was included, the ParT model consistently delivered the highest accuracy.

doi:10.6342/NTU202501658

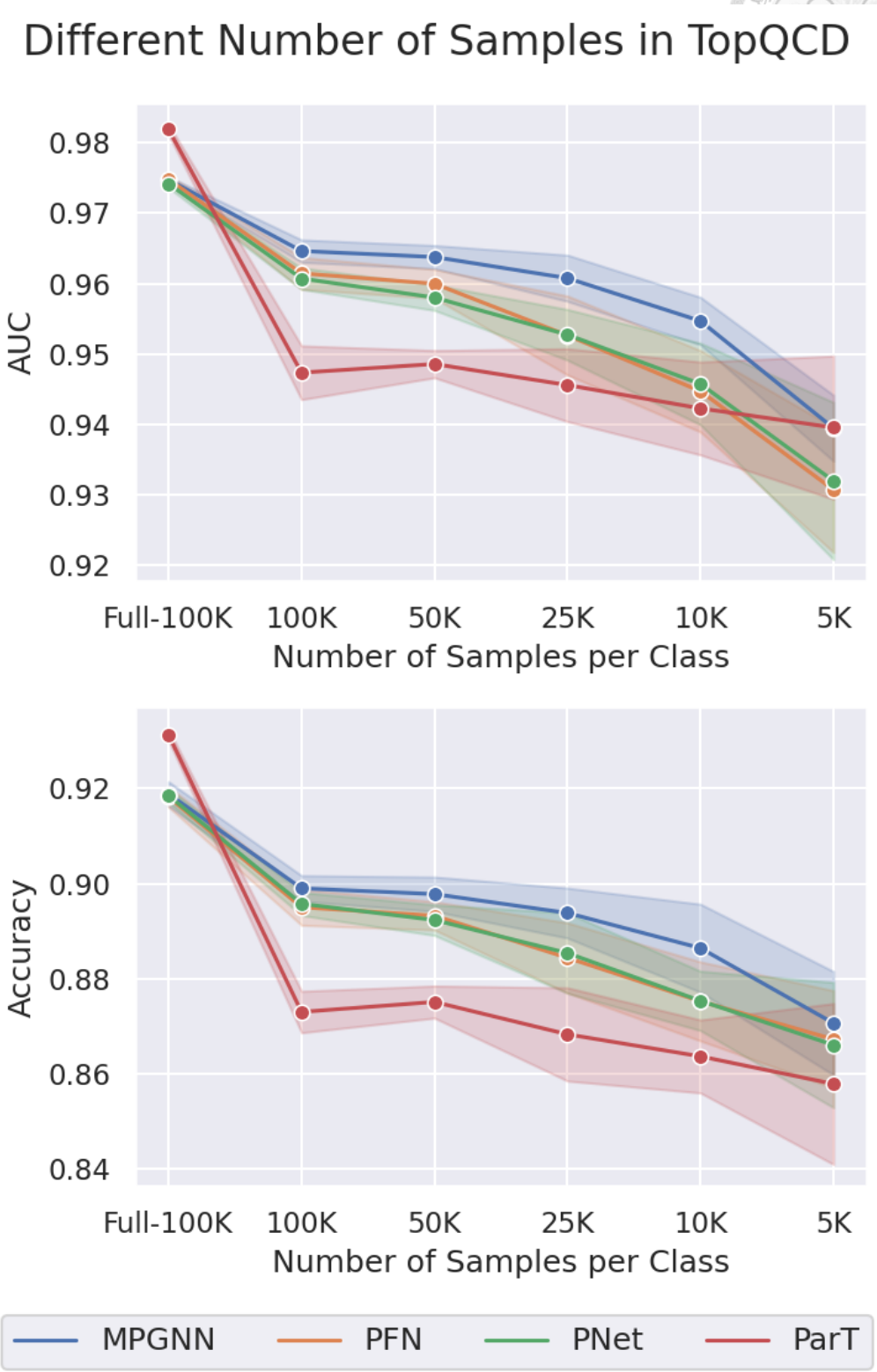


Figure 6.12: AUC and accuracy of classical models with different numbers of data in TopQCD dataset. The legend 'Full-100K' denotes using all available particle information that do not applied with additional selection about the transverse momentum.

doi:10.6342/NTU202501658

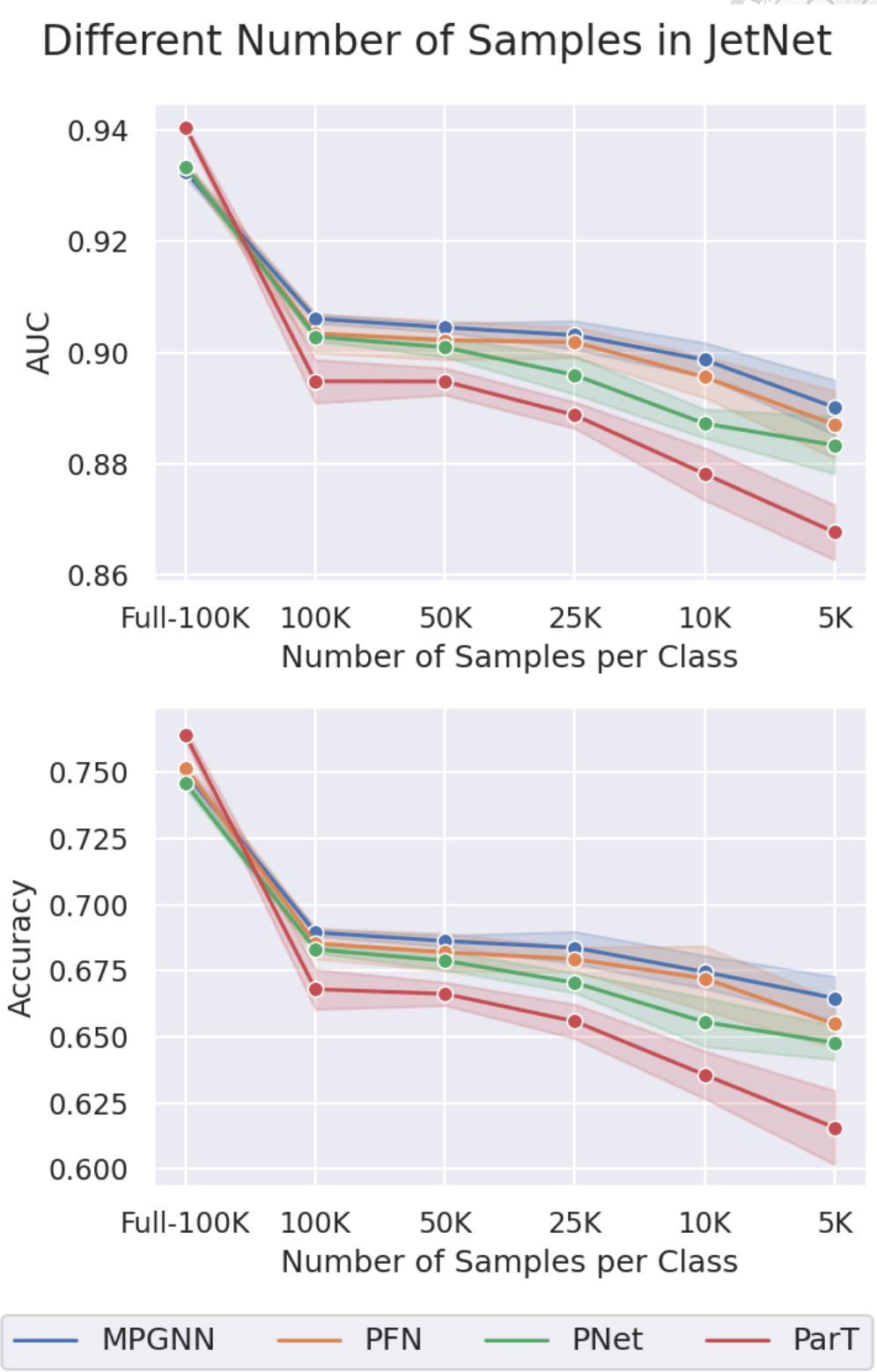


Figure 6.13: AUC and accuracy of classical models with different numbers of data in JetNet dataset. The legend 'Full-100K' denotes using all available particle information that do not applied with additional selection about the transverse momentum.

doi:10.6342/NTU202501658

6.3.2 Classical Models and QCGNN on Simulators

Model	JetNet			TopQCD		
	Accuracy	AUC	# params	Accuracy	AUC	# params
ParT	65.6±0.6%	88.9±0.2%	2.2×10^6	86.8±0.9%	94.6±0.5%	2.2×10^6
PNet	66.9±0.4%	89.6±0.3%	1.8×10^5	88.5±0.6%	95.3±0.3%	1.8×10^5
PFN	67.5±0.5%	90.0±0.3%	7.3×10^4	88.5±0.5%	95.4±0.4%	7.3×10^4
MPGNN - $n_M = 64$	68.3±0.7%	90.3±0.2%	1.3×10^4	89.6±0.3%	96.1±0.3%	1.3×10^4
MPGNN - $n_M = 3$	47.5±14.1%	75.7±11.0%	1.8×10^2	86.4±0.6%	92.2±0.5%	1.3×10^2
MPGNN - $n_M = 6$	61.5±1.0%	86.5±0.4%	3.2×10^2	86.6±0.6%	92.4±0.6%	2.6×10^2
QCGNN - $n_Q = 3$	50.5±1.4%	79.6±0.9%	1.7×10^2	86.4±0.5%	91.9±0.6%	1.0×10^2
QCGNN - $n_Q = 6$	54.3±0.6%	82.2±0.3%	2.7×10^2	86.8±0.5%	93.2±0.4%	2.0×10^2

Table 6.1: The table compares the metrics of various models conducted with the datasets JetNet and TopQCD. The n_Q refers to the number of qubits in the NR, i.e., 2nd register of QCGNN. The number of neurons in hidden layers is denoted as n_M. For each model, the classification accuracy, AUC, and number of parameters are reported. The AUC is calculated as the mean over all possible class pairs, while the accuracy reflects the overall classification performance across all classes. Each result represents the average over five independent runs, with the standard deviation indicated.

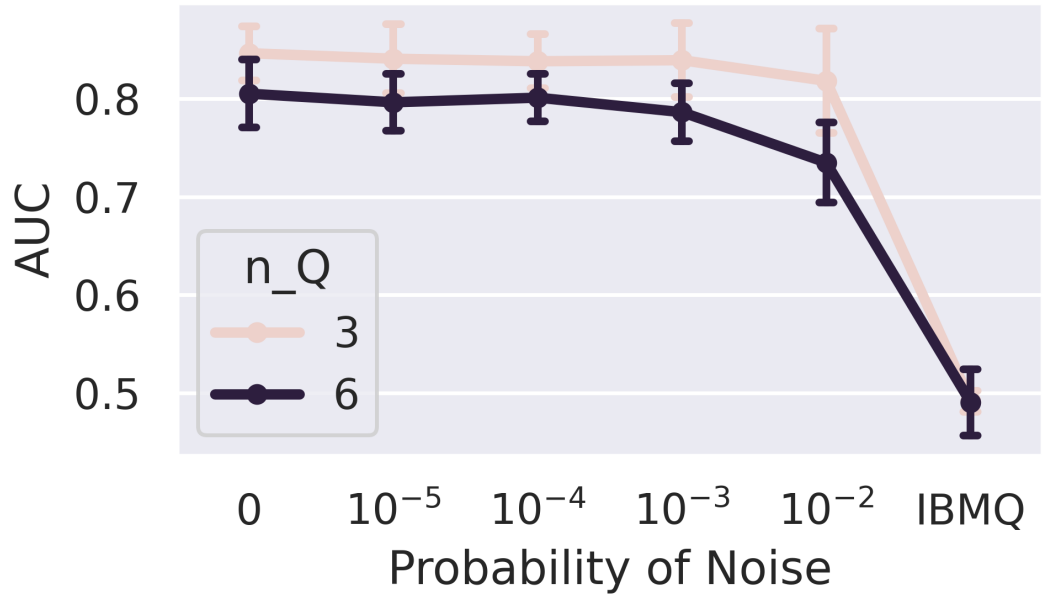
Table 6.1 shows the training metrics of the performance between classical and quantum models on the TopQCD and JetNet datasets. When the amount of parameters is of similar magnitude, the results of inference about the MPGNN and QCGNN models are comparable. As more qubits are introduced, we anticipate that QCGNN will more closely match the behavior of MPGNN. On the JetNet dataset, which involves multi-class classification with limited parameters, QCGNN exhibits greater training stability, whereas MPGNN shows larger fluctuations across different runs.

Among the various advanced models we evaluated, MPGNN with $n_M = 64$ yields the best inference performance. This observation can be explained by the effect of the preprocessing procedure, where only 4 to 16 particles are kept for each jet, potentially discarding some soft particle details. When training is performed using the complete particle information from the original jet datasets, some of the other competitive models show comparable, or even superior, performance to MPGNN, as further discussed in Section 6.3.1.

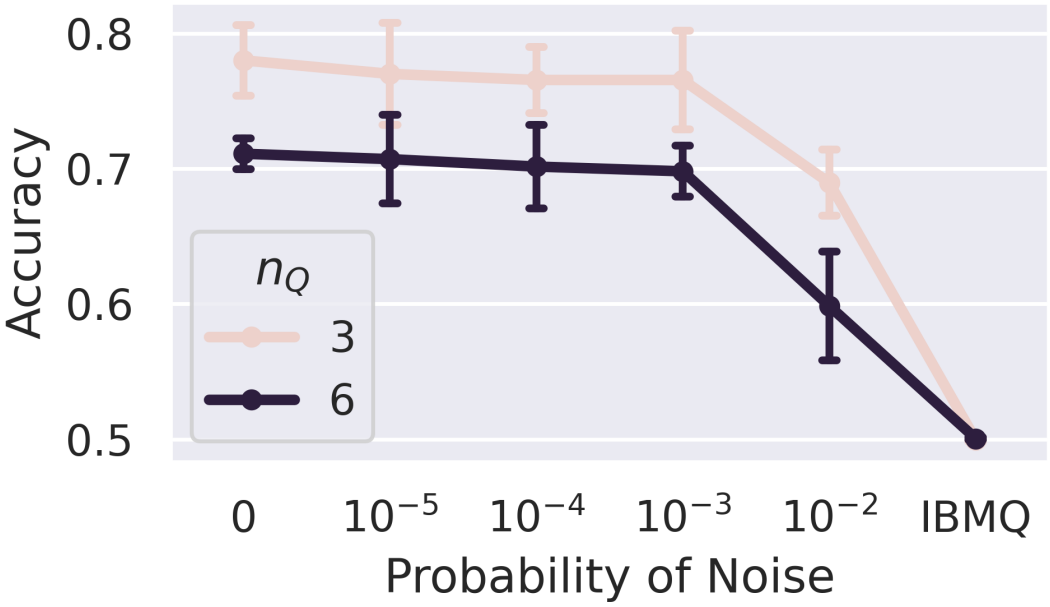
6.3.3 Pre-trained QCGNN on IBMQ

Despite that executing QML remains infeasible on current quantum hardware due to the limitations of the NISQ era [102], it is still possible to benchmark the inference capabilities of pretrained QCGNNs on real devices provided by the IBM Quantum platform (IBMQ) [69]. To mitigate the impact of hardware-induced noise, the evaluation is restricted to events containing

doi:10.6342/NTU202501658



(a) AUC with noise extrapolation.



(b) Accuracy with noise extrapolation.

Figure 6.14: Performance of pretrained QCGNNs under different levels of simulated quantum noise. The horizontal axis indicates the probability of amplitude damping and depolarizing errors applied after each gate. The point labeled "IBMQ" corresponds to execution on the real quantum device *ibm_brussels*, while the ideal (noise-free) scenario is shown at zero. Results are based on binary classification using 400 jet events, each with four particles, corresponding to $n_I = 2$ qubits in the input register. The vertical axis reports both AUC and classification accuracy, with error bars denoting the standard deviation across five independent runs.

doi:10.6342/NTU202501658

only four particles from the TopQCD dataset. This configuration requires only $n_I = 2$ qubits in the input register, significantly reducing circuit depth for both preparing the initial quantum state and encoding the data. Under these simplifications, the initialization of quantum state can be implemented simply by applying Hadamard gates to all qubits in IR. Since IBMQ hardware supports only 1-qubit and 2-qubit gates, the quantum gates multi-controlled conditions used for encoding particle information in IR are realized according to the methods outlined in Section 5.1.2.

The hardware tests were performed using the *ibm_brussels* backend, and the number of shots is set to 1,024. However, due to the high noise levels inherent in contemporary quantum devices, the QCGNN outputs, namely, the expectation values, on real hardware was approximately random, yielding an AUC and classification accuracy near 0.5 in the binary classification task. To further investigate the performance of QCGNN inference with different noise level, we conducted noise extrapolation studies using PennyLane simulators. In these simulations, depolarizing and amplitude damping errors were applied after each quantum gate.

We now describe the implementation of the simulated noise models. Consider the quantum system expressed via a density matrix ρ of n qubits. Noise affecting the q-th qubit is introduced via Kraus operators through the following operation:

$$\begin{aligned}\mathcal{E}^{(q)}_{\text{dep}}(\rho) = \sum_{j=0}^{3} &\left(I_0 \otimes \cdots \otimes I_{q-1} \otimes K_j \otimes I_{q+1} \otimes \cdots \otimes I_{n-1}\right) \rho \\ &\times \left(I_0 \otimes \cdots \otimes I_{q-1} \otimes K_j \otimes I_{q+1} \otimes \cdots \otimes I_{n-1}\right)^\dagger .\end{aligned} \tag{6.11}$$

The Kraus operators for each type of noise are defined as follows:

- **Depolarizing Noise**: The depolarizing channel is characterized by the Kraus operators:

$$\begin{aligned} K_0 &= \sqrt{1-p}\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}, \quad K_1 = \sqrt{\frac{p}{3}}\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}, \\ K_2 &= \sqrt{\frac{p}{3}}\begin{bmatrix}0 & -i\\ i & 0\end{bmatrix}, \quad K_3 = \sqrt{\frac{p}{3}}\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}. \end{aligned} \tag{6.12}$$

- **Amplitude Damping Noise**: The amplitude damping channel is described by the Kraus

doi:10.6342/NTU202501658

operators:

$$K_0 = \sqrt{\gamma}\begin{bmatrix}1 & 0\\ 0 & \sqrt{1-p}\end{bmatrix}, \quad K_1 = \sqrt{\gamma}\begin{bmatrix}0 & \sqrt{p}\\ 0 & 0\end{bmatrix}, \tag{6.13}$$

$$K_2 = \sqrt{1-\gamma}\begin{bmatrix}\sqrt{1-p} & 0\\ 0 & 1\end{bmatrix}, \quad K_3 = \sqrt{1-\gamma}\begin{bmatrix}0 & 0\\ \sqrt{p} & 0\end{bmatrix}.$$

In this work, the damping parameter γ is set to 0.5.

Here, p denotes the probability that an error occurs. In our simulations, noise is applied after the USO, after the data encoding operations for each particle within jets, and after the parameterized variational ansatz. For each noise insertion, each qubit is randomly assigned either a depolarizing error or an amplitude damping error.

As shown in Fig. 6.14, maintaining inference performance significantly better than random guessing requires reducing the noise probability to below $p \approx 10^{-3}$.

6.3.4 QCGNN Quantum Gate Runtime Analysis

IBMQ Backend	N	T_{ENC}	T_{PARAM}
ibm_nazca	2	2.567	0.209
	4	5.352	0.197
	8	10.551	0.219
ibm_strasbourg	2	2.595	0.217
	4	5.416	0.197
	8	11.085	0.211

Table 6.2: Runtime analysis of encoding gate operating time and parameterized layers on various IBMQ backends. The encoding runtime T_{ENC} and parameterized-layer runtime T_{PARAM} are defined in Equations 6.14 and 6.15, respectively. All gate operating times are reported in seconds, and N denotes the number of particles.

To empirically test the analysis of time complexity discussed in Section 5.4, we executed untrained QCGNN circuits on several IBMQ quantum backends: *ibm_strasbourg* and *ibm_nazca*. All experiments were performed using $n_Q = 100$ qubits and 1,024 shots.

We tested QCGNN on input graphs with 2, 4, and 8 nodes, such that the uniform state oracle (the initial state preparation) only requires Hadamard gates. The measurement of runtime was structured in three stages:

doi:10.6342/NTU202501658

1. First, we executed QCGNN without any encoding or parameterized gates to obtain the baseline runtime T_0, representing the cost of quantum state initialization (including the uniform state oracle) and measurement.

2. Second, we conducted encoding operations with ten rounds of data re-uploading and recorded the runtime T_1.

3. Third, we appended parameterized operations, consisting of ten strong-entanglement ansatz per reuploading and ten total reuploading steps, yielding a hundred strong-entanglement ansatz in total. The resulting runtime was denoted T_2.

Each runtime measurement was averaged over 10 executions to mitigate stochastic fluctuations.

The runtime per encoding step was estimated as:

$$T_{\text{ENC}} = \frac{T_1 - T_0}{10}, \tag{6.14}$$

and the operating time per strongly entangling layer was computed as:

$$T_{\text{PARAM}} = \frac{T_2 - T_1}{100}. \tag{6.15}$$

The summary is shown in Table 6.2. Agreeing to our expectation, the encoding time T_{ENC} scales approximately linearly with particle number per jet, consistent with theoretical analysis. In contrast, the time per parameterized layer T_{PARAM} remains roughly constant, indicating that once the circuit depth becomes sufficiently large, the overall runtime is dominated by the parameterized layers.

This confirms the key conclusion discussed in Section 5.4 that when training with deep quantum circuits, the cost of data encoding becomes negligible relative to the cost of parameterized operations.

doi:10.6342/NTU202501658

doi:10.6342/NTU202501658

Chapter 7

Conclusion and Future Prospect

7.1 Summary about QCGNN

Representing jets as graphs has become a common strategy in particle physics, primarily due to the intrinsic permutation invariance of particle orderings. Graph-based representations allow learning algorithms to process sets of particles in a physically meaningful way without being biased by arbitrary ordering. Despite their popularity, however, defining graph structures that accurately reflect the underlying physics of jets remains a nontrivial and largely unresolved problem. In the absence of rigorous theoretical guidance for edge construction, we adopt a simple yet general approach by treating each jet as a complete graph, using undirected and unweighted edges. This assumption ensures full pairwise interaction between particles, which aligns naturally with the symmetric processing afforded by quantum circuits.

Building on this complete-graph formulation, we give an detailed implementation on the QCGNN, which is a QML architecture tailored for permutation-invariant learning over jet data. In QCGNN, aggregation of pairwise information is performed using symmetric functions such as **SUM** or **MEAN**, which are realized through quantum observable averaging. When operating on N particles of a jet, the computational complexity of QCGNN scales as $O(N)$ under the assumption of sufficiently deep parameterized operators. This linear scaling stands in contrast to classical graph neural networks such as MPGNNs, which typically requires $O(N^2)$ complexity due to the need to process all pairwise combinations, and suggests the potential for polynomial speedup using quantum resources.

To conduct an experiment on the feasibility of QCGNN, we apply it to the task of jet classification. As shown in Section 6.3.2, QCGNN achieves performance comparable to its classical counterpart, i.e., MPGNN, when constrained to a similar number of trainable parameters. Notably, QCGNN also exhibits more consistent behavior across different random seeds, suggesting

doi:10.6342/NTU202501658

greater training stability in some regimes.

To further examine the feasibility of applying our method in practice, we test QCGNN on actual quantum hardware made available by IBMQ. While some initial experiments using pre-trained QCGNNs have already been carried out on real quantum processors, the current presence of quantum noise still limits the reliability of inference. To assess how this noise might impact performance in the NISQ regime, we adopt extrapolation techniques over noise through quantum simulators, as described in Section 6.3.3. Moreover, we evaluate the runtime of QCGNN on real quantum devices, as explained in Section 6.3.4, and observe that the data encoding cost increases with the number of particles in a jet roughly with a linear scale. In contrast, the computational cost of parameterized layers stays relatively stable, which is consistent with our theoretical findings in Section 5.4.

In summary, QCGNN offers a promising and resource-efficient method for learning from unstructured jet data using quantum machine learning. Its design leverages the symmetry of complete graphs, supports efficient observable-based aggregation, and remains scalable in practice. Importantly, the overhead introduced by quantum state initialization and encoding becomes negligible as the depth of variational circuits increases. Despite these advantages, whether QCGNN (or QML more broadly) can deliver a provable quantum advantage in high-energy physics remains an open and fundamental question. Furthermore, the development of more expressive encoding methods of HEP data on quantum circuits continues to represent a rich direction for future research, particularly in bridging the gap between theoretical models and experimental feasibility.

7.2 Future Work

One of the technical challenges in realizing the practical deployment of QCGNN is to mitigate the effects of quantum noise. As discussed in Section 6.3.3, the current noise levels in near-term quantum devices significantly affect the performance and stability of quantum neural networks, particularly when deeper circuits are involved. These limitations result in the demands on robust error mitigation strategies. Techniques such as zero-noise extrapolation have shown promise in recent studies and should be further investigated within the QCGNN framework. In the longer term, the integration of quantum error correction codes may offer a path toward fault-tolerant quantum computation, although such schemes currently impose substantial overhead and hardware requirements.

Another crucial aspect is the optimization of quantum circuit design for QCGNN. Our current implementation, though conceptually straightforward, employs parameterized quantum

doi:10.6342/NTU202501658

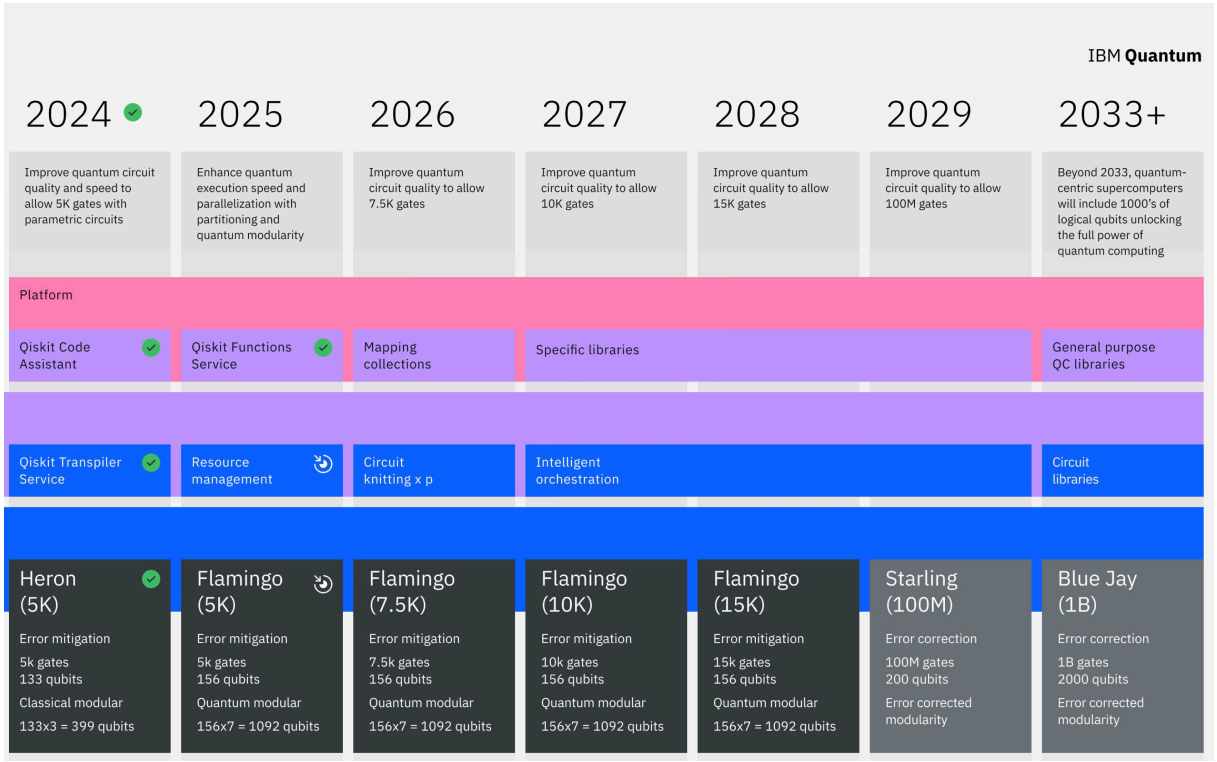


Figure 7.1: IBM Quantum development roadmap from 2024 through 2033 and beyond. The roadmap outlines yearly goals in improving quantum circuit quality, increasing gate depth (up to 1 billion), and advancing hardware platforms from Heron to Blue Jay. It also highlights planned milestones in quantum software services, such as Qiskit platforms, circuit orchestration, and library development. The roadmap envisions a transition from modular error mitigation to fully error-corrected quantum computing with thousands of logical qubits by 2033+. Figure adapted from IBM Quantum.

doi:10.6342/NTU202501658

circuits that are resource-intensive regarding to both circuit depth and gate count. This design, while sufficient for proof-of-concept experiments, may not scale efficiently with larger inputs or more complex tasks. Therefore, developing more efficient quantum data encoding schemes is an essential direction for future research. The goal is to balance expressive power with circuit simplicity to preserve the potential quantum advantage of QCGNN.

Moreover, there remains a significant opportunity to deepen the theoretical understanding of quantum models, especially quantum neural networks, e.g., VQCs. In particular, it is important to investigate under what conditions QCGNN can outperform classical counterparts not only in terms of computational complexity but also in learning capacity and generalization. Exploring these questions will require a combination of analytical tools from quantum information theory, learning theory, and graph theory.

In conclusion, while QCGNN presents a compelling approach to quantum machine learning for jet physics, substantial work remains to bring this conceptual work into practice. Reducing quantum noise, optimizing circuit design, and expanding theoretical foundations are all vital components of the path forward. The intersection of quantum computing and high-energy physics offers potential for innovation, and future advancements in hardware and algorithms will determine the ultimate viability of quantum machine learning in real-world scientific applications.

doi:10.6342/NTU202501658

Bibliography

[1] S. Chatrchyan et al. "Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC". In: *Physics Letters B* 716.1 (2012), pp. 30–61. ISSN: 0370-2693. DOI: https://doi.org/10.1016/j.physletb.2012.08.021. URL: https://www.sciencedirect.com/science/article/pii/S0370269312008581.

[2] G. Aad et al. "Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC". In: *Physics Letters B* 716.1 (2012), pp. 1–29. ISSN: 0370-2693. DOI: https://doi.org/10.1016/j.physletb.2012.08.020. URL: https://www.sciencedirect.com/science/article/pii/S037026931200857X.

[3] Wikipedia contributors. *Standard Model*. Accessed: April 6, 2025. 2025. URL: https://en.wikipedia.org/wiki/Standard_Model.

[4] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[5] Matthew Feickert and Benjamin Nachman. *A Living Review of Machine Learning for Particle Physics*. 2021. arXiv: 2102.02770 [hep-ph].

[6] HEP ML Community. *A Living Review of Machine Learning for Particle Physics*. URL: https://iml-wg.github.io/HEPML-LivingReview/.

[7] Alexander Radovic et al. "Machine learning at the energy and intensity frontiers of particle physics". In: *Nature* 560.7716 (2018), pp. 41–48. DOI: 10.1038/s41586-018-0361-2. URL: https://doi.org/10.1038/s41586-018-0361-2.

[8] Huilin Qu and Loukas Gouskos. "Jet tagging via particle clouds". In: *Phys. Rev. D* 101 (5 Mar. 2020), p. 056019. DOI: 10.1103/PhysRevD.101.056019. URL: https://link.aps.org/doi/10.1103/PhysRevD.101.056019.

[9] Isaac Henrion et al. "Neural Message Passing for Jet Physics". In: 2017. URL: https://api.semanticscholar.org/CorpusID:39724044.

[10] Eric A. Moreno et al. "JEDI-net: a jet identification algorithm based on interaction networks". In: *The European Physical Journal C* 80.1 (2020), p. 58. DOI: 10.1140/epjc/s10052-020-7608-4. URL: https://doi.org/10.1140/epjc/s10052-020-7608-4.

[11] Amit Chakraborty, Sung Hak Lim, and Mihoko M. Nojiri. "Interpretable deep learning for two-prong jet classification with jet spectra". In: *Journal of High Energy Physics* 2019.7 (2019), p. 135. DOI: 10.1007/JHEP07(2019)135. URL: https://doi.org/10.1007/JHEP07(2019)135.

doi:10.6342/NTU202501658

[12] Amit Chakraborty et al. “Neural network-based top tagger with two-point energy correlations and geometry of soft emissions”. In: *Journal of High Energy Physics* 2020.7 (2020), p. 111. DOI: 10.1007/JHEP07(2020)111. URL: https://doi.org/10.1007/JHEP07(2020)111.

[13] Jonathan Shlomi, Peter Battaglia, and Jean-Roch Vlimant. “Graph neural networks in particle physics”. In: *Machine Learning: Science and Technology* 2.2 (Dec. 2020), p. 021001. DOI: 10.1088/2632-2153/abbf9a. URL: https://dx.doi.org/10.1088/2632-2153/abbf9a.

[14] Xiangyang Ju and Benjamin Nachman. “Supervised jet clustering with graph neural networks for Lorentz boosted bosons”. In: *Phys. Rev. D* 102 (7 Oct. 2020), p. 075014. DOI: 10.1103/PhysRevD.102.075014. URL: https://link.aps.org/doi/10.1103/PhysRevD.102.075014.

[15] Frédéric A. Dreyer and Huilin Qu. *Jet tagging in the Lund plane with graph networks*. 2021. arXiv: 2012.08526 [hep-ph]. URL: https://arxiv.org/abs/2012.08526.

[16] Shiqi Gong et al. “An efficient Lorentz equivariant graph neural network for jet tagging”. In: *Journal of High Energy Physics* 2022.7 (2022), p. 30. DOI: 10.1007/JHEP07(2022)030. URL: https://doi.org/10.1007/JHEP07(2022)030.

[17] Fei Ma, Feiyi Liu, and Wei Li. “Jet tagging algorithm of graph network with Haar pooling message passing”. In: *Phys. Rev. D* 108 (7 Oct. 2023), p. 072007. DOI: 10.1103/PhysRevD.108.072007. URL: https://link.aps.org/doi/10.1103/PhysRevD.108.072007.

[18] Farouk Mokhtar, Raghav Kansal, and Javier Duarte. *Do graph neural networks learn traditional jet substructure?* 2022. arXiv: 2211.09912 [hep-ex]. URL: https://arxiv.org/abs/2211.09912.

[19] Daniel Murnane. *Graph Structure from Point Clouds: Geometric Attention is All You Need*. 2023. arXiv: 2307.16662 [cs.LG]. URL: https://arxiv.org/abs/2307.16662.

[20] Savannah Thais et al. *Graph Neural Networks in Particle Physics: Implementations, Innovations, and Challenges*. 2022. arXiv: 2203.12852 [hep-ex].

[21] Jun Guo et al. “Boosted Higgs boson jet reconstruction via a graph neural network”. In: *Phys. Rev. D* 103 (11 June 2021), p. 116025. DOI: 10.1103/PhysRevD.103.116025. URL: https://link.aps.org/doi/10.1103/PhysRevD.103.116025.

[22] Jacob Biamonte et al. “Quantum machine learning”. In: *Nature* 549.7671 (2017), pp. 195–202. DOI: 10.1038/nature23474. URL: https://doi.org/10.1038/nature23474.

[23] Amine Zeguendry, Zahi Jarir, and Mohamed Quafafou. “Quantum Machine Learning: A Review and Case Studies”. In: *Entropy* 25.2 (2023). ISSN: 1099-4300. DOI: 10.3390/e25020287. URL: https://www.mdpi.com/1099-4300/25/2/287.

[24] David Peral García, Juan Cruz-Benito, and Francisco José García-Peñalvo. *Systematic Literature Review: Quantum Machine Learning and its applications*. 2022. arXiv: 2201.04093 [quant-ph].

[25] Kyriaki A. Tychola, Theofanis Kalampokas, and George A. Papakostas. “Quantum Machine Learning mdash;An Overview”. In: *Electronics* 12.11 (2023). ISSN: 2079-9292. DOI: 10.3390/electronics12112379. URL: https://www.mdpi.com/2079-9292/12/11/2379.

doi:10.6342/NTU202501658

[26] Maria Schuld and Francesco Petruccione. *Machine Learning with Quantum Computers*. Jan. 2021. ISBN: 978-3-030-83097-7. DOI: 10.1007/978-3-030-83098-4.

[27] Yi-An Chen and Kai-Feng Chen. "Jet discrimination with a quantum complete graph neural network". In: *Phys. Rev. D* 111 (1 Jan. 2025), p. 016020. DOI: 10.1103/PhysRevD.111.016020. URL: https://link.aps.org/doi/10.1103/PhysRevD.111.016020.

[28] M. Cerezo et al. "Variational quantum algorithms". In: *Nature Reviews Physics* 3.9 (2021), pp. 625–644. DOI: 10.1038/s42254-021-00348-9. URL: https://doi.org/10.1038/s42254-021-00348-9.

[29] Alberto Peruzzo et al. "A variational eigenvalue solver on a photonic quantum processor". In: *Nature Communications* 5.1 (2014), p. 4213. DOI: 10.1038/ncomms5213. URL: https://doi.org/10.1038/ncomms5213.

[30] Jarrod R McClean et al. "The theory of variational hybrid quantum-classical algorithms". In: *New Journal of Physics* 18.2 (Feb. 2016), p. 023023. DOI: 10.1088/1367-2630/18/2/023023. URL: https://dx.doi.org/10.1088/1367-2630/18/2/023023.

[31] J. Altmann et al. "Towards the understanding of heavy quarks hadronization: from leptonic to heavy-ion collisions". In: *The European Physical Journal C* 85.1 (2025), p. 16. DOI: 10.1140/epjc/s10052-024-13641-5. URL: https://doi.org/10.1140/epjc/s10052-024-13641-5.

[32] CMS Collaboration and Thomas Mc Cauley. *Multi-jet event recorded by the CMS detector (Run 2, 13 TeV)*. CERN Document Server. Accessed: 2025-04-04. 2015. URL: https://cds.cern.ch/record/2114784.

[33] CMS Collaborations and N.Tran. *Jet Types*. Accessed: 2025-04-05. URL: https://cms-opendata-workshop.github.io/workshop2022-lesson-physics-objects/06-substructure/index.html.

[34] Izaak Neutelings. *CMS coordinate system*. TikZ.net. Accessed: 2025-04-05. URL: https://tikz.net/axis3d_cms/.

[35] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. "The anti-kt jet clustering algorithm". In: *Journal of High Energy Physics* 2008.04 (Apr. 2008), p. 063. DOI: 10.1088/1126-6708/2008/04/063. URL: https://dx.doi.org/10.1088/1126-6708/2008/04/063.

[36] CMS Collaborations. *Jet Clustering Algorithm*. Accessed: 2025-04-05. URL: https://cms-opendata-workshop.github.io/workshop2023-lesson-physics-objects/04-jetmet/index.html.

[37] Gregor Kasieczka et al. *Top Quark Tagging Reference Dataset*. Version v0 (2018_03_27). Zenodo, Mar. 2019. DOI: 10.5281/zenodo.2603256. URL: https://doi.org/10.5281/zenodo.2603256.

[38] Kai-Feng Chen and Yang-Ting Chien. "Deep learning jet substructure from two-particle correlations". In: *Phys. Rev. D* 101 (11 June 2020), p. 114025. DOI: 10.1103/PhysRevD.101.114025. URL: https://link.aps.org/doi/10.1103/PhysRevD.101.114025.

[39] Hamza Kheddar et al. *Image Classification in High-Energy Physics: A Comprehensive Survey of Applications to Jet Analysis*. 2024. arXiv: 2403.11934 [hep-ph]. URL: https://arxiv.org/abs/2403.11934.

doi:10.6342/NTU202501658

[40] Jason Sang Hun Lee et al. "Quark-Gluon Jet Discrimination Using Convolutional Neural Networks". In: *Journal of the Korean Physical Society* 74.3 (2019), pp. 219–223. DOI: 10.3938/jkps.74.219. URL: https://doi.org/10.3938/jkps.74.219.

[41] Jing Li and Hao Sun. *An Attention Based Neural Network for Jet Tagging*. 2020. arXiv: 2009.00170 [hep-ph]. URL: https://arxiv.org/abs/2009.00170.

[42] Sang Kwan Choi et al. "Automatic detection of boosted Higgs boson and top quark jets in an event image". In: *Phys. Rev. D* 108 (11 Dec. 2023), p. 116002. DOI: 10.1103/PhysRevD.108.116002. URL: https://link.aps.org/doi/10.1103/PhysRevD.108.116002.

[43] Gregor Kasieczka et al. "Deep-learning top taggers or the end of QCD?" In: *Journal of High Energy Physics* 2017.5 (2017), p. 6. DOI: 10.1007/JHEP05(2017)006. URL: https://doi.org/10.1007/JHEP05(2017)006.

[44] Patrick T. Komiske, Eric M. Metodiev, and Matthew D. Schwartz. "Deep learning in color: towards automated quark/gluon jet discrimination". In: *Journal of High Energy Physics* 2017.1 (2017), p. 110. DOI: 10.1007/JHEP01(2017)110. URL: https://doi.org/10.1007/JHEP01(2017)110.

[45] Pierre Baldi et al. "Jet substructure classification in high-energy physics with deep neural networks". In: *Phys. Rev. D* 93 (9 May 2016), p. 094034. DOI: 10.1103/PhysRevD.93.094034. URL: https://link.aps.org/doi/10.1103/PhysRevD.93.094034.

[46] *Identification of Jets Containing b-Hadrons with Recurrent Neural Networks at the ATLAS Experiment*. Tech. rep. All figures including auxiliary figures are available at https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PUBNOTES/ATL-PHYS-PUB-2017-003. Geneva: CERN, 2017. URL: https://cds.cern.ch/record/2255226.

[47] Rafael Teixeira de Lima. *Sequence-based Machine Learning Models in Jet Physics*. 2021. arXiv: 2102.06128 [physics.data-an]. URL: https://arxiv.org/abs/2102.06128.

[48] E. Bols et al. "Jet flavour classification using DeepJet". In: *Journal of Instrumentation* 15.12 (Dec. 2020), P12012. DOI: 10.1088/1748-0221/15/12/P12012. URL: https://dx.doi.org/10.1088/1748-0221/15/12/P12012.

[49] Daniel Guest et al. "Jet flavor classification in high-energy physics with deep neural networks". In: *Phys. Rev. D* 94 (11 Dec. 2016), p. 112002. DOI: 10.1103/PhysRevD.94.112002. URL: https://link.aps.org/doi/10.1103/PhysRevD.94.112002.

[50] Jason Sang Hun Lee et al. "Quark Gluon Jet Discrimination with Weakly Supervised Learning". In: *Journal of the Korean Physical Society* 75.9 (2019), pp. 652–659. DOI: 10.3938/jkps.75.652. URL: https://doi.org/10.3938/jkps.75.652.

[51] Shannon Egan et al. *Long Short-Term Memory (LSTM) networks with jet constituents for boosted top tagging at the LHC*. 2017. arXiv: 1711.09059 [hep-ex].

[52] Jannicke Pearkes et al. *Jet Constituents for Deep Neural Network Based Top Quark Tagging*. 2017. arXiv: 1704.02124 [hep-ex].

[53] Taoli Cheng. "Recursive Neural Networks in Quark/Gluon Tagging". In: *Computing and Software for Big Science* 2.1 (2018), p. 3. DOI: 10.1007/s41781-018-0007-y. URL: https://doi.org/10.1007/s41781-018-0007-y.

doi:10.6342/NTU202501658

[54] Gilles Louppe et al. "QCD-aware recursive neural networks for jet physics". In: *Journal of High Energy Physics* 2019.1 (2019), p. 57. DOI: 10.1007/JHEP01(2019)057. URL: https://doi.org/10.1007/JHEP01(2019)057.

[55] Patrick T. Komiske, Eric M. Metodiev, and Jesse Thaler. "Energy flow networks: deep sets for particle jets". In: *Journal of High Energy Physics* 2019.1 (2019), p. 121. DOI: 10.1007/JHEP01(2019)121. URL: https://doi.org/10.1007/JHEP01(2019)121.

[56] Huilin Qu, Congqiao Li, and Sitian Qian. *Particle Transformer for Jet Tagging*. 2024. arXiv: 2202.03772 [hep-ph]. URL: https://arxiv.org/abs/2202.03772.

[57] Matthew J. Dolan and Ayodele Ore. "Equivariant energy flow networks for jet tagging". In: *Phys. Rev. D* 103 (7 Apr. 2021), p. 074022. DOI: 10.1103/PhysRevD.103.074022. URL: https://link.aps.org/doi/10.1103/PhysRevD.103.074022.

[58] *Deep Sets based Neural Networks for Impact Parameter Flavour Tagging in ATLAS*. Tech. rep. All figures including auxiliary figures are available at https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PUBNOTES/ATL-PHYS-PUB-2020-014. Geneva: CERN, 2020. URL: https://cds.cern.ch/record/2718948.

[59] Benno Käch, Dirk Krücker, and Isabell Melzer-Pellmann. *Point Cloud Generation using Transformer Encoders and Normalising Flows*. 2022. arXiv: 2211.13623 [hep-ex]. URL: https://arxiv.org/abs/2211.13623.

[60] Dimitrios Athanasakos et al. *Is infrared-collinear safe information all you need for jet classification?* 2023. arXiv: 2305.08979 [hep-ph]. URL: https://arxiv.org/abs/2305.08979.

[61] Benno Käch and Isabell Melzer-Pellmann. *Attention to Mean-Fields for Particle Cloud Generation*. 2023. arXiv: 2305.15254 [hep-ex]. URL: https://arxiv.org/abs/2305.15254.

[62] Spandan Mondal, Gaetano Barone, and Alexander Schmidt. *PAIReD jet: A multi-pronged resonance tagging strategy across all Lorentz boosts*. 2023. arXiv: 2311.11011 [hep-ex]. URL: https://arxiv.org/abs/2311.11011.

[63] Patrick Odagiu et al. *Sets are all you need: Ultrafast jet classification on FPGAs for HL-LHC*. 2024. arXiv: 2402.01876 [hep-ex]. URL: https://arxiv.org/abs/2402.01876.

[64] Rikab Gambhir, Athis Osathapan, and Jesse Thaler. *Moments of Clarity: Streamlining Latent Spaces in Machine Learning using Moment Pooling*. 2024. arXiv: 2403.08854 [hep-ph]. URL: https://arxiv.org/abs/2403.08854.

[65] Wikipedia contributors. *Complete graph*. Accessed: April 6, 2025. 2025. URL: https://en.wikipedia.org/wiki/Complete_graph.

[66] Manzil Zaheer et al. *Deep Sets*. 2018. arXiv: 1703.06114 [cs.LG]. URL: https://arxiv.org/abs/1703.06114.

[67] Yue Wang et al. *Dynamic Graph CNN for Learning on Point Clouds*. 2019. arXiv: 1801.07829 [cs.CV]. URL: https://arxiv.org/abs/1801.07829.

[68] Wikipedia contributors. *Bloch sphere*. Accessed: April 6, 2025. 2025. URL: https://en.wikipedia.org/wiki/Bloch_sphere.

[69] Ali Javadi-Abhari et al. *Quantum computing with Qiskit*. 2024. DOI: 10.48550/arXiv.2405.08810. arXiv: 2405.08810 [quant-ph].

doi:10.6342/NTU202501658

[70] S.E. Rasmussen et al. “Superconducting Circuit Companion—an Introduction with Worked Examples”. In: *PRX Quantum* 2 (Dec. 2021). DOI: 10.1103/PRXQuantum.2.040204.

[71] S.E. Rasmussen et al. “Superconducting Circuit Companion—an Introduction with Worked Examples”. In: *PRX Quantum* 2 (4 Dec. 2021), p. 040204. DOI: 10.1103/PRXQuantum.2.040204. URL: https://link.aps.org/doi/10.1103/PRXQuantum.2.040204.

[72] Thomas E. Roth, Ruichao Ma, and Weng C. Chew. “The Transmon Qubit for Electromagnetics Engineers: An introduction”. In: *IEEE Antennas and Propagation Magazine* 65.2 (2023), pp. 8–20. DOI: 10.1109/MAP.2022.3176593.

[73] Oxford. *Oxford instruments - nanoscience*. Jun 4, 2025. URL: https://nanoscience.oxinst.com/products/proteoxmx.

[74] Jerry Chow Pat Gumann. *IBM scientists cool down the world' s largest quantum-ready cryogenic concept system*. 8 Sep 2022. URL: https://www.ibm.com/quantum/blog/goldeneye-cryogenic-concept-system.

[75] Maria Schuld et al. “Evaluating analytic gradients on quantum hardware”. In: *Phys. Rev. A* 99 (3 Mar. 2019), p. 032331. DOI: 10.1103/PhysRevA.99.032331. URL: https://link.aps.org/doi/10.1103/PhysRevA.99.032331.

[76] David Wierichs et al. “General parameter-shift rules for quantum gradients”. In: *Quantum* 6 (Mar. 2022), p. 677. ISSN: 2521-327X. DOI: 10.22331/q-2022-03-30-677. URL: https://doi.org/10.22331/q-2022-03-30-677.

[77] Gavin E. Crooks. *Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition*. 2019. arXiv: 1905.13311 [quant-ph].

[78] Adrián Pérez-Salinas et al. “Data re-uploading for a universal quantum classifier”. In: *Quantum* 4 (Feb. 2020), p. 226. ISSN: 2521-327X. DOI: 10.22331/q-2020-02-06-226. URL: https://doi.org/10.22331/q-2020-02-06-226.

[79] Maria Schuld, Ryan Sweke, and Johannes Jakob Meyer. “Effect of data encoding on the expressive power of variational quantum-machine-learning models”. In: *Phys. Rev. A* 103 (3 Mar. 2021), p. 032430. DOI: 10.1103/PhysRevA.103.032430. URL: https://link.aps.org/doi/10.1103/PhysRevA.103.032430.

[80] Alok Shukla and Prakash Vedula. “An efficient quantum algorithm for preparation of uniform quantum superposition states”. In: *Quantum Information Processing* 23.2 (Jan. 2024). ISSN: 1573-1332. DOI: 10.1007/s11128-024-04258-4. URL: http://dx.doi.org/10.1007/s11128-024-04258-4.

[81] Ryan Babbush et al. “Encoding Electronic Spectra in Quantum Circuits with Linear T Complexity”. In: *Physical Review X* 8.4 (Oct. 2018). ISSN: 2160-3308. DOI: 10.1103/physrevx.8.041015. URL: http://dx.doi.org/10.1103/PhysRevX.8.041015.

[82] Ville Bergholm et al. *PennyLane: Automatic differentiation of hybrid quantum-classical computations*. 2022. arXiv: 1811.04968 [quant-ph].

[83] Michael A Nielsen and Isaac L Chuang. “Controlled operations”. In: *Quantum Computation and Quantum Information*. Cambridge University Press, 2007. Chap. 4.

[84] Vojtěch Havlíček et al. “Supervised learning with quantum-enhanced feature spaces”. In: *Nature* 567.7747 (2019), pp. 209–212. DOI: 10.1038/s41586-019-0980-2. URL: https://doi.org/10.1038/s41586-019-0980-2.

doi:10.6342/NTU202501658

[85] Maria Schuld. *Supervised quantum machine learning models are kernel methods*. 2021. arXiv: 2101.11020 [quant-ph]. URL: https://arxiv.org/abs/2101.11020.

[86] Arthur Jacot, Franck Gabriel, and Clément Hongler. *Neural Tangent Kernel: Convergence and Generalization in Neural Networks*. 2020. arXiv: 1806.07572 [cs.LG]. URL: https://arxiv.org/abs/1806.07572.

[87] Pedro Domingos. *Every Model Learned by Gradient Descent Is Approximately a Kernel Machine*. 2020. arXiv: 2012.00152 [cs.LG]. URL: https://arxiv.org/abs/2012.00152.

[88] Hsin-Yuan Huang et al. "Power of data in quantum machine learning". In: *Nature Communications* 12.1 (2021), p. 2631. DOI: 10.1038/s41467-021-22539-9. URL: https://doi.org/10.1038/s41467-021-22539-9.

[89] PyTorch Team. *PyTorch*. URL: https://pytorch.org.

[90] Raghav Kansal et al. *JetNet*. Version 2. Zenodo, Aug. 2022. DOI: 10.5281/zenodo.6975118. URL: https://doi.org/10.5281/zenodo.6975118.

[91] Matteo Cacciari, Gavin P. Salam, and Gregory Soyez. "FastJet user manual". In: *The European Physical Journal C* 72.3 (2012), p. 1896. DOI: 10.1140/epjc/s10052-012-1896-2. URL: https://doi.org/10.1140/epjc/s10052-012-1896-2.

[92] Christian Bierlich et al. *A comprehensive guide to the physics and usage of PYTHIA 8.3*. 2022. arXiv: 2203.11601 [hep-ph].

[93] S. Ovyn, X. Rouby, and V. Lemaitre. *Delphes, a framework for fast simulation of a generic collider experiment*. 2010. arXiv: 0903.2225 [hep-ph].

[94] J. de Favereau et al. "DELPHES 3, A modular framework for fast simulation of a generic collider experiment". In: *JHEP* 02 (2014), p. 057. DOI: 10.1007/JHEP02(2014)057. arXiv: 1307.6346 [hep-ex].

[95] Gregor Kasieczka et al. "The Machine Learning landscape of top taggers". In: *SciPost Physics* 7.1 (July 2019). ISSN: 2542-4653. DOI: 10.21468/scipostphys.7.1.014. URL: http://dx.doi.org/10.21468/SciPostPhys.7.1.014.

[96] J. Alwall et al. "The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations". In: *JHEP* 07 (2014), p. 079. DOI: 10.1007/JHEP07(2014)079. arXiv: 1405.0301 [hep-ph].

[97] Raghav Kansal et al. *Particle Cloud Generation with Message Passing Generative Adversarial Networks*. 2022. arXiv: 2106.11535 [cs.LG]. URL: https://arxiv.org/abs/2106.11535.

[98] Maria Schuld et al. "Circuit-centric quantum classifiers". In: *Phys. Rev. A* 101 (3 Mar. 2020), p. 032308. DOI: 10.1103/PhysRevA.101.032308. URL: https://link.aps.org/doi/10.1103/PhysRevA.101.032308.

[99] Abien Fred Agarap. *Deep Learning using Rectified Linear Units (ReLU)*. 2019. arXiv: 1803.08375 [cs.NE]. URL: https://arxiv.org/abs/1803.08375.

[100] Matthias Fey and Jan E. Lenssen. "Fast Graph Representation Learning with PyTorch Geometric". In: *ICLR Workshop on Representation Learning on Graphs and Manifolds*. 2019.

doi:10.6342/NTU202501658

[101] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG]. URL: https://arxiv.org/abs/1412.6980.

[102] John Preskill. "Quantum Computing in the NISQ era and beyond". In: *Quantum* 2 (Aug. 2018), p. 79. ISSN: 2521-327X. DOI: 10.22331/q-2018-08-06-79. URL: https://doi.org/10.22331/q-2018-08-06-79.

doi:10.6342/NTU202501658