Machine Learning in Particle Physics: Graph Neural Networks for Jet Clustering at the Future Circular Collider (FCC-ee)

Candidate no. 1048263

Thesis word count: 18956

A thesis submitted for the degree of
Master of Computer Science

Trinity 2021

Abstract

We present an interdisciplinary project in Particle Physics and Computer Science by applying novel data processing techniques, modern data analysis and integrating them with advanced machine learning. Our objective is to develop a novel methodology to perform final state particle classification, based on its origin of decay in the event of two leptons(electron and positron) colliding at the FCC-ee detector, a future circular detector, proposed at CERN in the coming decades. Based on the classification of the final state, as decaying from the Higgs Boson, or Z Boson, we group the particles of the same class together: jet clustering. Traditional techniques use sequential recombination methods for jet clustering. A wide variety of graph machine learning architectures, using diverse graph representations, specifically targeting generated level simulations, have been recently explored in the literature and have offered promising results. However, there has been limited exploration of the application of graph neural networks in jet clustering, as most relevant work has been carried out in pile-up mitigation, reconstruction of the calorimeter, and jet tagging. Our research project is the first to build a supervised node classification for jet clustering in an electron-positron collision within an inductive learning setup on a simulation dataset. Each event collision is represented as a graph, whereas the final state particles (observed at the detector) are represented as nodes. We aim to develop three new graph representations based on existing literature reviews, implement a baseline for comparison among the eight different graph neural network models, and evaluate the best model parameters through multiple empirical experiments. Our key findings report an increased performance against the established baseline, the most suitable network depth, the best performing GNN architectures, graph processing scheme, and edge-generation scheme. Our work highlights the way forward for further exploration and research to enhance jet clustering algorithms for the future circular collider.

Machine Learning in Particle Physics: Graph Neural Networks for Jet Clustering at the Future Circular Collider (FCC-ee)

Candidate no. 1048263

Word count: 18956

A thesis submitted for the degree of

Master of Computer Science

☐ I hereby certify that this is entirely my own work unless otherwise stated.

Trinity 2021

Abstract

We present an interdisciplinary project in Particle Physics and Computer Science by applying novel data processing techniques, modern data analysis and integrating them with advanced machine learning. Our objective is to develop a novel methodology to perform final state particle classification, based on its origin of decay in the event of two leptons(electron and positron) colliding at the FCC-ee detector, a future circular detector, proposed at CERN in the coming decades. Based on the classification of the final state, as decaying from the Higgs Boson, or Z Boson, we group the particles of the same class together: jet clustering. Traditional techniques use sequential recombination methods for jet clustering. A wide variety of graph machine learning architectures, using diverse graph representations, specifically targeting generated level simulations, have been recently explored in the literature and have offered promising results. However, there has been limited exploration of the application of graph neural networks in jet clustering, as most relevant work has been carried out in pile-up mitigation, reconstruction of the calorimeter, and jet tagging. Our research project is the first to build a supervised node classification for jet clustering in an electron-positron collision within an inductive learning setup on a simulation dataset. Each event collision is represented as a graph, whereas the final state particles (observed at the detector) are represented as nodes. We aim to develop three new graph representations based on existing literature reviews, implement a baseline for comparison among the eight different graph neural network models, and evaluate the best model parameters through multiple empirical experiments. Our key findings report an increased performance against the established baseline, the most suitable network depth, the best performing GNN architectures, graph processing scheme, and edge-generation scheme. Our work highlights the way forward for further exploration and research to enhance jet clustering algorithms for the future circular collider.

Contents

List of Figures **ix**

List of Abbreviations **x**

1 Introduction **1**
1.1 Particle Physics . . . 1
1.1.1 The Standard Model . . . 2
1.1.2 Limitations of the Standard Model . . . 2
1.1.3 Objective of Experimental Particle Physics . . . 2
1.2 Computing in Particle Physics . . . 3
1.3 Higgs Boson . . . 3
1.3.1 Significance of the Higgs Boson . . . 3
1.3.2 Future Experiments Studying the Higgs Boson . . . 4
1.4 Graph Neural Networks in Particle Physics . . . 4
1.5 Motivation: Problem Statement . . . 5
1.6 Proposed Solution . . . 5
1.7 Overview of the Chapters . . . 6
1.8 Key Contributions . . . 7

2 Background **9**
2.1 Introduction . . . 10
2.2 The Standard Model (SM) . . . 10
2.2.1 Fermions . . . 10
2.2.2 Force Carriers . . . 13
2.2.3 Standard Units of Measurements . . . 14
2.2.4 Limitations of SM . . . 14
2.3 Experimental Particle Physics . . . 15
2.3.1 Introduction to Particle Accelerators and Colliders . . . 15
2.3.2 Experimental Dataset . . . 17
2.3.3 Simulations . . . 18
2.3.4 Distributed Computing: . . . 19
2.3.5 Accelerators of the Future . . . 19

2.4 Existing Experiments . . . 19
2.5 Future experiments . . . 20
2.5.1 How is the Higgs Boson produced at FCC-ee? . . . 21
2.5.2 Higgs Boson Production in HZ mode: . . . 21
2.5.3 Higgs Boson Decay: . . . 22
2.5.4 Event Decay Chain . . . 22
2.6 Role of Engineering and Computer Science in Particle Physics . . . 23
2.6.1 CERN's Infrastructure . . . 23
2.6.2 Future of Computing in HEP . . . 23
2.6.3 Challenges in HEP: Physics and Computing . . . 23
2.7 Graph Neural Networks in Particle Physics . . . 24
2.7.1 Introduction . . . 24
2.7.2 Related Work . . . 25
2.7.3 Jet Classification: Jet Tagging . . . 25
2.7.4 Jet Clustering . . . 26
2.8 Graph Neural Networks in Jet Clustering . . . 27
2.8.1 Research Questions . . . 28

3 Graph Machine Learning **29**
3.1 What is a graph? . . . 30
3.1.1 What are the types of graphs . . . 30
3.1.2 Mathematical Representations of Graph . . . 31
3.1.3 Properties: Graph Metrics and Node Metrics . . . 31
3.1.4 Graph Network Embedding . . . 33
3.1.5 Graph Representation Learning . . . 34
3.1.6 Machine Learning in Network Science . . . 35
3.2 Message Passing Networks . . . 35
3.2.1 K-Hop Neighbourhood . . . 36
3.2.2 Node Embeddings . . . 37
3.2.3 Message Passing with Self Loops . . . 37
3.2.4 Basic Graph Neural Networks . . . 37
3.3 Graph Convolutional Networks . . . 38
3.3.1 Symmetric Normalisation . . . 38
3.3.2 Graph Convolutional Networks . . . 38
3.4 Set Aggregations . . . 38
3.4.1 Neighbourhood Attention Mechanisms . . . 39
3.5 Generalised Update Functions . . . 39
3.5.1 Over-Smoothing . . . 40
3.5.2 Skip Connections . . . 40

3.5.3 Jumping Knowledge Connections 41
3.6 Overview of Graph Neural Networks 41
3.6.1 History of Graph Neural Networks 41
3.6.2 Types of Graph Neural Networks 42
3.6.3 Training Settings for Graph Neural Networks 42
3.6.4 Computational Modules in Graph Neural Networks 43
3.6.5 Recent advances in Graph Neural Networks 44
3.6.6 Complex Real World Graphs in Network Science 45
3.6.7 Graph Transfer Learning 45
3.6.8 Explainability in Graph Neural Networks 45
3.6.9 Computational Efficiency 46
3.7 Strengths and Weaknesses of Graph Neural Networks 47
3.7.1 Strengths and Advantages 47
3.7.2 Limitations and their Proposed Solutions 48

4 Methodology **51**
4.1 Introduction 52
4.2 Particle Physics Data 52
4.2.1 Event Data 53
4.2.2 Simulated Dataset 53
4.3 Data Generation and Conversion 54
4.3.1 Data handling framework 54
4.3.2 Simulation Frameworks 54
4.3.3 Data Conversion to Compatible Formats 54
4.4 Data Description 55
4.4.1 Dataset Version 1 55
4.4.2 Dataset Version 2 56
4.4.3 Data Analysis and Preprocessing 57
4.5 An overview of the GNN Pipeline 57
4.5.1 Practical Implementation 57
4.5.2 Introduction to Graph Generation in Particle Physics 57
4.5.3 Construction of a Graph 57
4.5.4 Role of Edges in a Graph 58
4.5.5 Size of a Graph 58
4.6 Definition of the Machine Learning Task 59
4.6.1 What are the entities and relations that could also be represented as nodes and edges, respectively? 59
4.6.2 What is the desired output, such as predictions at the edge, node, or graph level? 59

4.6.3 Is it necessary to have a global output network to create graph-level outputs? . . . 59
4.6.4 How many message-passing steps should be utilised to spread information among the graph's remote nodes? . . . 59
4.7 Edge-generation strategies . . . 60
4.7.1 edge-KNN . . . 60
4.7.2 edge-radius . . . 60
4.7.3 edge-label . . . 60
4.8 Graph Processing . . . 61
4.8.1 Fixed-size Graph dataset . . . 61
4.8.2 Variable-size Graph dataset . . . 61
4.8.3 Edges in Fixed and Variable size dataset . . . 61
4.9 Model Evaluation Metrics . . . 61
4.9.1 Accuracy . . . 62
4.9.2 Node Accuracy . . . 62
4.9.3 Event Accuracy . . . 62
4.10 Model Methodology . . . 63
4.10.1 Graph Network Blocks . . . 63
4.10.2 Attention mechanism . . . 63
4.10.3 Stacking Graph Network Blocks . . . 64
4.10.4 GNN architectures . . . 64
4.11 Selecting the Graph Network Blocks . . . 64
4.11.1 GCNConv . . . 64
4.11.2 ChebConv . . . 65
4.11.3 TAGConv . . . 66
4.11.4 Jumping Knowledge . . . 66
4.11.5 SAGEConv . . . 67
4.11.6 GINConv . . . 67
4.11.7 GATConv . . . 67
4.11.8 SuperGATConv . . . 68
4.11.9 Conclusion . . . 69

5 Results **70**
5.1 Introduction . . . 71
5.1.1 Event wise classification . . . 71
5.1.2 Multiclass-classifier: . . . 71
5.1.3 Default Case: . . . 72
5.2 MLP: Establishing a Baseline with neural networks . . . 72
5.2.1 Objective: MLP-2 . . . 72

5.2.2 Results: . . . 72
5.3 MLP: Network Depth . . . 73
5.3.1 Objective: Network Depth of 2, 4, 8 . . . 73
5.3.2 Results: . . . 73
5.4 Graph Convolutional Network: An in-depth exploration . . . 73
5.4.1 Objective: GCN-2 . . . 73
5.4.2 Results: . . . 73
5.5 GCN: Most Representative Dataset . . . 74
5.5.1 Objective: GCN-2 on edge-radius, edge-KNN, edge-label . . 74
5.5.2 Results: . . . 75
5.6 GCN: Network Depth Experiment . . . 75
5.6.1 Objective: Network Depth of 2, 4, 8, 16 . . . 75
5.6.2 Results: . . . 76
5.7 GCN: Accuracy with Different Dataset Sizes . . . 80
5.7.1 Objective: Dataset split into 0.2, 0.4, 0.6, 0.8 . . . 80
5.7.2 Results: . . . 80
5.8 GCN: Hyperparameter Search . . . 81
5.8.1 Objective: Find optimal hyperparameters for GCN-2 on edge-KNN . . . 81
5.8.2 Results: . . . 81
5.9 GNN-variants: Model Evaluation and Comparison . . . 81
5.9.1 Objective: MLP, GCN, ChebNet, SAGE, GAT, GIN, JKNet, TAGCN and SuperGAT . . . 81
5.9.2 Results: . . . 83
5.10 Conclusion . . . 85
5.10.1 Could machine learning techniques be used to predict the sets of final state particles originating from the Higgs Boson and Z boson in the HZ production mode at the future lepton collider? . . . 85
5.10.2 How can graph neural networks be used to model event collisions as graphs and particles as nodes for the supervised task of node classification? . . . 85
5.10.3 How can we generate relationships or links between final state particles? What is the most optimal type of graph data representation of the stable particles detected at a collider? . 86
5.10.4 What is the most promising graph neural architecture to correctly predict all the nodes' labels in a graph? . . . 86

6 Conclusions **87**
6.1 Overview . . . 87
6.2 Limitations of Our Work . . . 88
6.3 Key Contributions and Results . . . 89
6.4 Future work and Directions . . . 89
6.4.1 Higgs Production Modes . . . 89
6.4.2 Data representation . . . 90
6.4.3 Lepton Collider Datasets . . . 91
6.4.4 Extending Model Designs . . . 91
6.4.5 Exhaustive evaluation on Random Seeds . . . 91
6.4.6 Fast Inference . . . 91
6.4.7 Larger Datasets . . . 92
6.5 Conclusion . . . 92

Bibliography **93**

List of Figures

2.1 The Standard Model of Particle Physics 11
2.2 An example of an event collision at the LHC: ATLAS[147] 17
2.3 Supervised ML algorithms are used to invert the detector simulation to infer something about the underlying physics and are then applied to real data. 18
2.4 Future Circular Collider[145] 20
2.5 Reference Frame: [146] 21
2.6 Higgsstrahlung Process [30] 22

3.1 Embeddings of a Graph[148] 34
3.2 Message Passing Networks: Aggregation from a node's local neighbourhood[148] 35
3.3 Comparison of Euclidean and Non-Euclidean Data 42
3.4 Computational Modules of a Graph Neural Network[29] 44

4.1 Graph Neural Networks Pipeline [29] 58

5.1 Training and Validation Accuracy Curves on GCN-2: edge-KNN . . 74
5.2 Training and Validation Loss Curves on GCN-2: edge-KNN 74
5.3 Networkx Visualisation of a sample event: edge-KNN 75
5.4 Networkx Visualisation of a sample event: edge-radius 76
5.5 Node Metrics for a sample event: edge-KNN 77
5.6 Graph Global Metrics for a sample of 100 event:edge-KNN 78
5.7 Visualisation of Subgraph to Explain GCN-2 predictions for node at indices(L-R): 0, 11, 12, 21 using [91]: edge-KNN 79

List of Abbreviations

Anti-kt . . . . Anti-kt clustering algorithm

ATLAS . . . . A Toroidal LHC Apparatus

AWAKE . . . Advanced Proton Driven Plasma Wakefield Acceleration Experiment

BDT . . . . . Boosted Decision Trees

C-A . . . . . Cambridge-Aachen

CEPC . . . . . Circular Electron Positron Collider

CERN . . . . . European Organisation for Nuclear Research

ChebNet . . . Graph Convolutional Network with Chebyshev filter

CLIC . . . . . Compact Linear Collider

CMS . . . . . Compact Muon Solenoid

CNN . . . . . Convolutional Neural Networks

CPU . . . . . Central Processing Unit

DAG . . . . . Directed Acyclic Graph

DNN . . . . . Deep Neural Network

e- or e . . . . . Electron

e+ . . . . . Positron

ESA . . . . . European Space Agency

eV . . . . . Electron-Volt

FCC . . . . . Future Circular Collider

FCC-ee . . . . Future Circular Lepton Collider

FPGA . . . . . Field Programmable Gate Arrays

GAT . . . . . Graph Attention Network

GCN . . . . . Graph Convolutional Network

GeV . . . . . Giga-electron Volt

GIN . . . . . Graph Isomorphism Network

GNN . . . . . Graph Neural Networks

H . . . . . Higgs Boson

HEP . . . . . High Energy Physics

HL-LHC . . . High-Luminosity LHC

ILC . . . . . International Linear Collider

JKNet . . . . . Jumping Knowledge Network

KNN . . . . . K Neareast Neighbours

LEP . . . . . Large Electron–Positron Collider

LHC . . . . . Large Hardon Collider

LR . . . . . Learning Rate

MC . . . . . Monte Carlo

ML . . . . . Machine Learning

MLP . . . . . Multi Layer Perceptron (Neural Network)

MVA . . . . . Multivariate Analysis

NN . . . . . Neural Network

PB . . . . . Peta Byte

pp . . . . . Proton-Proton

QCD . . . . . Quantum Chromodynamics

QFT . . . . . Quantum Field Theory

RF . . . . . Radio-Frequency

RL . . . . . Re-inforcement Learning

RNN . . . . . Recurrent Neural Networks

SAGE . . . . . Sample and Aggregate

SGD . . . . . Stochastic Gradient Descent

SM . . . . . Standard Model

SuperGAT . . Self-supervised Graph Attention Network

TAGCN . . . . Topology Aware Graph Convolutional Network

TB . . . . . Tera Byte

W- . . . . . Gauge Boson and carrier of weak force

W+ . . . . . Gauge Boson and carrier of weak force

WD . . . . . Weight Decay

WLCG . . . . Worldwide LHC Computing Grid

WWW . . . . World Wide Web

Z . . . . . Z Boson

1 Introduction

Contents

1.1 Particle Physics 1
1.1.1 The Standard Model 2
1.1.2 Limitations of the Standard Model 2
1.1.3 Objective of Experimental Particle Physics 2
1.2 Computing in Particle Physics 3
1.3 Higgs Boson 3
1.3.1 Significance of the Higgs Boson 3
1.3.2 Future Experiments Studying the Higgs Boson 4
1.4 Graph Neural Networks in Particle Physics 4
1.5 Motivation: Problem Statement 5
1.6 Proposed Solution 5
1.7 Overview of the Chapters 6
1.8 Key Contributions 7

1.1 Particle Physics

Particle physics is a field of science devoted to the development of the fundamental laws of matter and forces. It is a scientific discipline that investigates the nature of our Universe's constituents. Its objective is to understand and uncover all characteristics of elementary particles, the smallest irreducible subset of constituents, and how they interact and behave fundamentally. As a result of this endeavour,

many important theoretical and experimental discoveries have been made in the last few decades, with the effort spanning across centuries.

1.1.1 The Standard Model

Particle physics focuses on studying fundamental particles in controlled environments such as in collider physics or nature. The standard model is a theory of elementary particles such as quarks and leptons and the strong, weak, and electromagnetic forces. The Standard Model is currently our best explanation of the elements of matter [1]. It was built with the help of interrelated quantum field theories majorly created in the second half of the twentieth century.

1.1.2 Limitations of the Standard Model

While the scope and precision of the theoretical framework are phenomenal, the standard model still fails to describe some key aspects of our Universe, including gravity, dark matter and energy, the fine-tuned hierarchy of effects, the origin of neutrino's mass, and the matter-antimatter asymmetry. The main challenge of modern physics is to broaden its scope to include these critical components, resulting in the birth of theories such as Supersymmetry, String Theory, and Grand Unified Theory.

1.1.3 Objective of Experimental Particle Physics

Experimental particle physics aims to overcome the existing predicament by exposing evidence of physics outside the Standard Model or occurrences where its predictions differ from observations. Physicists explore the subatomic realm by colliding large numbers of particles and examining the decayed particles formed due to the collisions. Massive detectors constructed around the collision or interaction point record a wide variety of data from these events. The validity of theoretical predictions of the SM can be tested against observations by analysing the data acquired so far.

1.2 Computing in Particle Physics

The largest machine and most powerful accelerator, the Large Hadron Collider, was built at CERN in 2008. On a typical day of operation, the Large Hadron Collider's different detectors produce approximately 1 PB of collision data per second and capture about 100 TB per day [2, 3]. As a result, computational efficiency is a crucial component of modern particle physics. The experimental design requires computational approaches across data, software, hardware, and infrastructure regimes for resource sharing, compression, storage, event filtering, simulation, reconstruction, and physics analysis.

1.3 Higgs Boson

The Higgs boson discovery, a fundamental scalar boson with a mass of 125 GeV, in 2012 [4, 5, 6, 20] at the LHC was the Standard Model's most recent spectacular triumph. The Higgs boson's couplings to other elementary particles, precise measurements of the Higgs boson's characteristics, an early examination of the Higgs boson's self-interaction and form of the Higgs potential have all been the focus of studies at the LHC so far.

1.3.1 Significance of the Higgs Boson

The 125 GeV mass of the Higgs boson is a remarkable value, implying that the fundamental condition of the Universe, the vacuum, is extremely close to the stable-metastable boundary, suggesting deeper physics beyond the standard model. [7] The Higgs potential also influences theories about the cosmological constant, dark energy, which drives the Universe's accelerating expansion, dark matter, which constitutes about 80 percent of the Universe's matter, and a possible phase transition in the early Universe that could be responsible for baryogenesis.

1.3.2 Future Experiments Studying the Higgs Boson

According to the latest European Particle Physics Strategy update in 2020 [10,11], precision studies of the Higgs boson and its interactions are a prime objective for the next high-energy collider.

Future colliders at CERN could include the Future Circular Collider(FCC-ee) [12,13] or the Compact Linear Collider (CLIC) [14,15]. Other additional electron-positron (e+e) collider options being discussed in the context of CERN's future could be the International Linear Collider (ILC) [16] and the Circular Electron Positron Collider (CEPC) in Japan and China, respectively [17].

Although electron-positron colliders cannot attain the same high centre-of-mass energies as proton-proton colliders, they have a considerably cleaner initial collision state, having fundamental particles with well-defined energies entering the interactions. Therefore, 'Higgs production could be measured inclusively from its presence as a recoil to the Z in e+e HZ events in electron-positron colliders, allowing the absolute measurement of the Higgs boson's coupling to the Z boson.' [7]

1.4 Graph Neural Networks in Particle Physics

Graph neural networks are trainable functions that operate on graphs—sets of objects and their pairwise relationship. GNNs are a key component of geometric deep learning. They are extremely expressive, and they have outperformed other traditional deep learning techniques in a multitude of domains. Particle physics data is frequently represented by sets and graphs; therefore, graph neural networks could provide significant benefits. [8]

Several graph representational techniques recently proposed in the literature for jet tagging have been successful. [18, 19, 21] Traditional jet clustering methods [9] are still prevalent and an important preprocessing step in the jet-tagging pipeline. Supervised jet clustering using graph neural networks on the edge level has been explored recently for proton-proton collisions at the LHC [31]. Therefore, it allows us to examine if graph neural networks can generate representative node embeddings

for jet clustering on the electron-positron collisions for the Future Circular Collider. We aim to investigate if the graph neural network algorithms can show promising results on the simulation dataset for supervised jet clustering at the node level.

1.5 Motivation: Problem Statement

Jets are a collimated spray of stable particles, which are directly observed in an experiment. Jets are first clustered and then classified into specific types (jet tagging or jet classification) to understand and identify the origin of the stable observed particles. Our purpose is to cluster particles and specify which set of final state (stable) particles were generated from the decay of the Higgs Boson, Z boson or other types of particles. To narrow our search, we aim to study whether graph representation learning algorithms could be added to the toolbox of experimental physicists of the FCC collaboration at CERN.

1.6 Proposed Solution

Our approach differs from existing methods as it questions if only final state particles can be used to discriminate based on the origin of decay. We first create the labels directly from the raw features of the event for all the particles in an event decay. Then we selectively filter the final state particles of the event. We then use the particle features of the final state particles and generate edges between them to build a graph dataset that is eventually used to create node embeddings through multiple layers of graph convolutions. In the end, we make predictions on the node and classify it into different categories: as particle decaying from Higgs, as a particle decaying from Z and as a particle decaying from others.

We process a simulated dataset: a node is represented as a particle observed at the detector, and all the particles in a single event collision are represented as a graph. We generate three different graphical representations of the final state particles by connecting them with edges and evaluate the node classification

accuracy of 10,000 events over various GNN architectures. Therefore, we develop a classifier that distinguishes among stable particles for a future lepton collider.

Deriving an accurate graph representation of events and developing a graph-machine-learning algorithm to discriminate particles based on their origin of decay in the FCC-ee experiment could prove extremely useful in its quest to unravel the laws of our Universe.

1.7 Overview of the Chapters

- Chapter 2 discusses the background physics and graph neural networks in-depth, the key research areas and methodologies, and highlights the current state of the art of jet clustering and node classification techniques in HEP. These concepts are aggregated and re-framed into a concrete and cohesive set of research questions for our research project.

- Chapter 3 discusses the mathematical definition, properties and background of graphs. It highlights the message passing network in graph neural networks and explains the most prominent architectures in depth. Lastly, it covers recent advances in the domain, followed by a set of advantages and disadvantages of using graph neural networks.

- Chapter 4 is dedicated to the project's overall methodology, and it focuses on processing data, generating and analysing graph datasets, and developing models suitable for the supervised learning task of node classification. A simulation dataset was collected and processed to meet requirements to analyse the effectiveness and characteristics of the investigated approach. We build three types of datasets: edge-KNN, edge-Radius, edge-Label based on edge generation recommendations by the authors in [8]. Since there are no benchmarks on the supervised node classification of the jet clustering task, we first establish a baseline using an MLP and GCN model. We present the foundations for graph neural networks, explain the model design, development and implementation. The chapter highlights seven types of recently developed

GNN architectures that are further implemented to improve the baseline results.

- Chapter 5 highlights the results of the implemented data and model methodology. We build upon the baseline models and evaluate the performance of the state of the art GNN architectures. We discover the most suitable dataset generated for our task and present the findings of the model performance using evaluation metrics of node classification accuracy. Further, we augment our work by introducing event accuracy, the number of events classified with 100 per cent node accuracy divided by the number of events in the dataset. Finally, all the models are examined, and their results are summarised for future work.

- This dissertation draws to a close with Chapter 6, which summarises the key findings from each research stage, limitations of the current work and future avenues for growth.

1.8 Key Contributions

The key contributions of the project are

- Edge Generation of three types of datasets:
  edge-KNN, edge-Radius, edge-Label.

- Implementation of a baseline for supervised node classification:
  Neural Network, Graph Convolutional Network.

- Model Exploration, Implementation and Optimisation of 8 GNN architectures.
  GCN, SAGE, GAT, GIN, JKNet, ChebNet, TAGCN, SuperGAT.

- Model evaluation and comparison of the top-performing GNN architectures.

Our research places a strong emphasis on the multi-disciplinary nature of the project: particle physics and machine learning. This dissertation aims to examine aspects of both disciplines and present a novel discriminator for the

clustering of the final state particles originating from H, Z or other particles at the future FCC-ee experiment.

2
Background

Contents

2.1 Introduction 10
2.2 The Standard Model (SM) 10
2.2.1 Fermions 10
2.2.2 Force Carriers 13
2.2.3 Standard Units of Measurements 14
2.2.4 Limitations of SM 14
2.3 Experimental Particle Physics 15
2.3.1 Introduction to Particle Accelerators and Colliders . . . 15
2.3.2 Experimental Dataset 17
2.3.3 Simulations 18
2.3.4 Distributed Computing: 19
2.3.5 Accelerators of the Future 19
2.4 Existing Experiments 19
2.5 Future experiments 20
2.5.1 How is the Higgs Boson produced at FCC-ee? 21
2.5.2 Higgs Boson Production in HZ mode: 21
2.5.3 Higgs Boson Decay: 22
2.5.4 Event Decay Chain 22
2.6 Role of Engineering and Computer Science in Particle Physics 23
2.6.1 CERN's Infrastructure 23
2.6.2 Future of Computing in HEP 23
2.6.3 Challenges in HEP: Physics and Computing 23
2.7 Graph Neural Networks in Particle Physics 24
2.7.1 Introduction 24
2.7.2 Related Work 25
2.7.3 Jet Classification: Jet Tagging 25
2.7.4 Jet Clustering 26

2.8 Graph Neural Networks in Jet Clustering 27
2.8.1 Research Questions 28

2.1 Introduction

We introduce high-energy physics (HEP) and computer science methodologies to accomplish the goals of the interdisciplinary project. We provide a summary of the physics background by outlining the major principles and underlying theoretical knowledge. We discuss the experimental setup at a new electron-lepton collider at the Future Circular Collider. Further, we introduce the Higgs production and decay and the motivation for specifically targeting the decay of particles from the Higgs Boson and Z bosons. We describe event decay chains and highlight how traditional clustering methods operate in particle physics. We present our aim to build an algorithm to cluster all particles originating from the same source together using graph neural networks and highlight the research questions for the project.

2.2 The Standard Model (SM)

The *Standard Model (SM)* has so far been the most successful theory for explaining matter's constituents and dynamics [4]. It is often regarded as science's most successful theory [10], capable of accurately predicting and modelling some of the Universe's mechanisms to a high degree of precision. The Standard Model is a set of fields and their dynamics: bosons which explain how matter interacts with other matter in electromagnetic, weak, and strong interactions and fermions, which describe the matter itself.

2.2.1 Fermions

- Fermions are classified according to how they interact or equivalently, by what charges they carry.

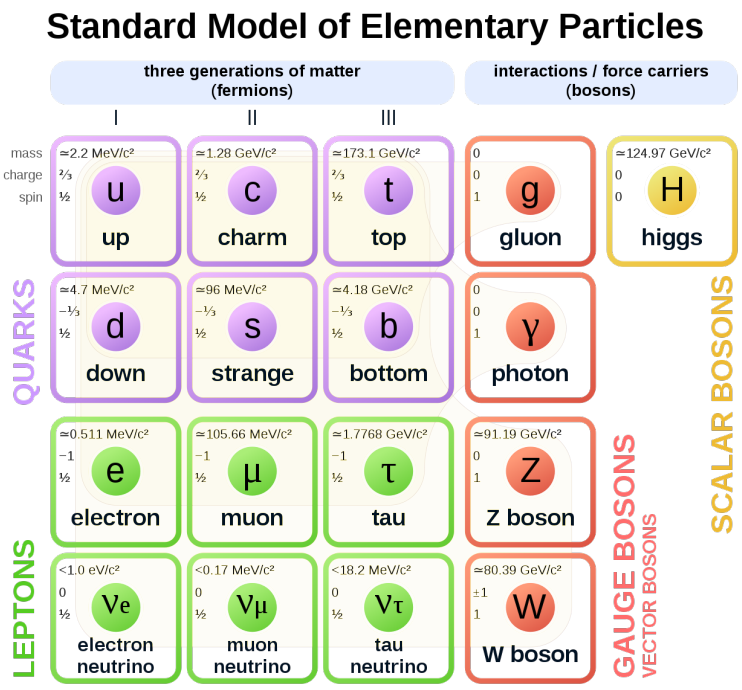

Figure 2.1: The Standard Model of Particle Physics

- Quarks and leptons are the two fundamental types of these particles. Each group is made up of six particles that are linked together in pairs, or generations. The first generation has the lightest and most stable particles, whereas the second and third generations include the heavier and less stable particles. All stable matter in the universe comprises first-generation particles; heavier particles rapidly decay to more stable ones. The six quarks are coupled in three generations: first, up quark and down quark, second, charm quark and strange quark, and finally, the top quark and bottom (or beauty) quark [32].

- Quarks come in three distinct 'colours' and can only be mixed in certain ways to produce colourless objects. These can be up, down, bottom, top, charm and strange quarks. Quantum Field Theory (QFT) of Quantum Chromodynamics describes this process (QCD).

- Colour denotes the fundamental strong charge and is available in three colours: 'blue', 'red', and 'green'. The gluon is the mediator of this interaction, is

colour-charged, with two charges, and massless, which has a significant impact on quark and gluon behaviour known as colour-confinement.

- Quarks are can never be found in isolation due to a mechanism known as colour confinement. Quarks can only be found within hadrons, comprising baryons (such as protons and neutrons) and mesons, or quark-gluon plasmas [138].
- Hadrons are composite particles and are made up of quarks: the most stable of which are protons and neutrons (building blocks of atomic nuclei)
- Quarks experience all four fundamental interactions, also known as fundamental forces: electromagnetism, gravitation, strong interaction, and weak interaction. Quarks are the only fundamental fermions sensitive to strong interaction forces. They are also the only particles having electric charges that are not integer multiple of the elementary charge.
- The six leptons – electron and electron neutrino, muon and muon neutrino, and tau and tau neutrino– are similarly organised in three generations.
- Leptons can be of two types: charged leptons, also known as electron-like leptons or muons, and neutral leptons, better known as neutrinos. The electron, muon, and tau all have an electric charge and a significant mass, but neutrinos are electrically neutral and have a little mass. For example, an electron is a lepton with an electric charge e = -1ev, and the anti-electron referred to as positron is depicted as e+ with a positive charge of +1ev.
- Charged leptons may mix with other particles to create composite particles like atoms and positronium, while neutrinos seldom interact with anything and are rarely seen.
- Leptons, unlike quarks, are not affected by the strong interaction, but they are affected by the other three basic interactions: gravity, the weak interaction, and electromagnetism, the latter of which is proportional to charge and therefore 0 for electrically neutral neutrinos.

2.2.2 Force Carriers

- The Universe is governed by four fundamental forces. They operate in a variety of ranges and have varying strengths. Gravity is the weakest of the forces, yet it has an unlimited range. The electromagnetic force has an unlimited range, but it is several orders of magnitude stronger than gravity. The weak and strong forces are only powerful over a very small distance and dominate solely at the subatomic particle level. The weak force, despite its name, is considerably stronger than gravity, yet it is the weakest of the three. As the name suggests, the strong force is the most powerful among the four basic interactions [32].

- Three of the basic forces result from the exchange of force-carrier particles, which are classified as bosons in a larger context. Particles of matter exchange discrete quantities of energy by exchanging bosons. Each fundamental force has its associated boson: electromagnetic force is carried by the photon, the weak force is carried by the W and Z bosons, and the strong force is carried by the gluon. Although yet to be discovered, the graviton should be the equivalent force-carrying particle of gravity.

- Based on their spins, elementary bosons are further categorised as gauge bosons and scalar bosons. Spin is defined as a type of angular momentum that is carried by elementary particles.

- Gauge bosons are force carriers that mediate strong, weak, and electromagnetic interactions: gluons, W^+, W^-, and Z and photons, respectively.

- Scalar bosons, like the Higgs Boson, are particles with no inherent spin. The Higgs boson is a fundamental particle attributed with the Higgs field, which gives mass to other fundamental particles like electrons and quarks [33][139].

2.2.3 Standard Units of Measurements

The masses of the particles, by convention, are in GeV, a specific unit of energy. ($10^9 eV$ where 1 eV is the energy of an electrical charge equal to that of the electron accelerating from rest in a potential of 1 volt). The renowned equation $E = mc^2$, with c the speed of light in vacuum, links the mass m and energy in the rest frame E. Both momentum and mass are frequently represented in GeV units, with the appropriate factor of c added back in as necessary. A mass represented in GeV can be re-written in the correct unit using GeV/c^2.

2.2.4 Limitations of SM

- As noted in the introduction, the SM is an incomplete model, missing an explanation for certain fundamental properties of the Universe, such as gravity, the mass of neutrinos, dark matter, imbalance of matter and anti-matter, and dark energy.

- The SM cannot be the ultimate theory for a range of reasons. It currently does not incorporate gravity, for example. It is difficult to connect this interaction with the others since it is extremely weak at the quantum level. It only becomes significant at larger scales. Fortunately for particle physics, the impact of gravity on the minuscule scale of particles is so small that it is insignificant. Only when matter is in mass, such as at the size of the human body or the planets, can gravity take over. So, despite its omission of one of the basic forces, the Standard Model nevertheless functions effectively [32].

- The SM predicts, just like the photons, neutrinos have no mass. However, scientists have discovered that the three neutrinos oscillate as they move, which is only feasible because neutrinos are not massless. It is unclear to what extent did the standard model made its predictions on neutrinos incorrectly. After all, neutrinos have extremely small masses [34].

- When scientists discovered that galaxies, based on the gravitational attraction of their visible matter, were spinning far faster than they should be, they concluded they were missing something. Galaxies should have ripped themselves apart since they were spinning so rapidly. Galaxies must be receiving extra mass—and therefore gravitational pull—from something we cannot directly observe, called dark matter. However, the Standard Model does not include dark matter, which accounts for 27 per cent of the Universe's total mass [34].

- The Standard Model cannot explain the imbalance between matter and anti-matter in the Universe. When equal amounts of matter and anti-matter interact, they annihilate each other. Scientists claim that matter and anti-matter should have been generated in equal amounts when the Universe was formed in the Big Bang. However, some mechanisms prevented matter and anti-matter from destroying one other in their expected pattern, and matter now dominates the Universe around us [34].

- According to the latest Hubble Space Telescope and ESA's Gaia(observatory) data, galaxies are speeding away from us at a rapid rate of 45 miles per second. The rate is thought to be caused by dark energy, an unknown feature of space-time that is pulling the Universe apart. Scientists estimate that the dark energy accounts for approximately 68 per cent of the energy in the universe[34].

2.3 Experimental Particle Physics

2.3.1 Introduction to Particle Accelerators and Colliders

- Planck-Einstein relation: According to the prominent physicist Albert Einstein, energy and length are inversely related in physics by the formula $E = \frac{hc}{\lambda}$ where h, is known as the Planck's constant, c is the speed of light and λ is the wavelength. To observe and precisely measure a particle, one must first obtain a wavelength and energy corresponding to the particle's size of interest. The smaller the particle, the higher energy would be used.

- Beam: A beam is a focused unidirectional stream of particles or radiation and is accelerated using particle accelerators. [35]

- Purpose of accelerators: Accelerators are designed to enhance the kinetic energy of charged particles (such as protons or electrons) before they release their inherent content through collisions (interactions). Charged particles are smashed onto a target or collide against other particles circulating in the opposite direction. Physicists probe into the sub-atomic realms by studying these particle collisions.[36] [35]

- Energy-Mass Relationship: When the particle's gains sufficient energy and collides with another particle, the energy of the collision is transformed into matter, and new particles are formed, the biggest of which existed in the early Universe. The phenomenon is described by Einstein's famous equation $E = mc^2$, according to which matter and energy are interchangeable.

- Collision: A collision occurs when two or more particles collide closely enough for physical quantities like energy, momentum, and charge to be exchanged. The mechanisms dictating physics at a particular level of energy are revealed by the decay of the original particle, which generates new particles.

- Accelerator Shape: Accelerators usually employ a circular shape to keep charged particle beams, typically gathered into bunches, in a closed trajectory. However, there are many linear accelerators as well. The particle bunches' speed and kinetic energy increase with each cycle via radio-frequency cavities (RF-cavities). The RF-cavities create an oscillating electromagnetic field that is synchronised with the evolving revolution frequency of the bunches to accelerate the particle bunches.

- Magnets: Powerful magnetic components constrain the beams to their intended trajectory so that they are collimated within a specific beamline radius. The particle bunches are monitored, controlled, and shaped using various magnets such as dipoles and quadrupoles.

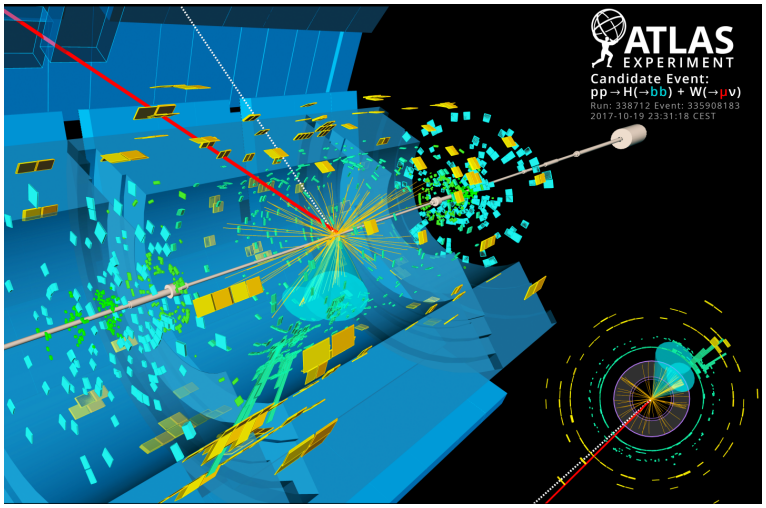

Figure 2.2: An example of an event collision at the LHC: ATLAS[147]

- Beamline: A beamline refers to all of the equipment used to control, monitor, and create a beam with specific characteristics. Magnets, intensity monitors, beam position monitors, and collimators are all typical components of a beamline.

- Interaction Point: Two distinct beams can intersect at a controlled Interaction Point along their path (IP).

- Event: Event refers to all information and processes resulting from two-particle bunches colliding.

2.3.2 Experimental Dataset

- Data Transmission: In the collider's operation, the high rate of collisions at the detector's interaction point produces a data transmission bottleneck. A set of filters should be used to limit data processing to events of interest. These filters are known as triggers, and they limit the rate of events by using a hierarchical setup of electronic devices at the lower level and numerical methods at the higher level.

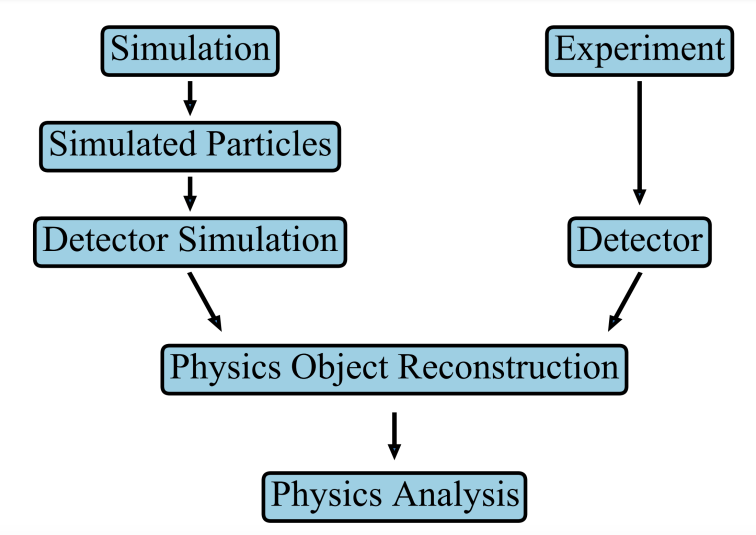

Figure 2.3: Supervised ML algorithms are used to invert the detector simulation to infer something about the underlying physics and are then applied to real data.

- Data Processing: Raw data that passes through all triggers could then be stored in massive databases. Higher-level variables are reconstructed from low-level information using several stages of processing.

- Data Analysis: Physicists can then access a hierarchy of increasingly processed databases for analysing various signal regions.

2.3.3 Simulations

Monte Carlo Techniques are stochastic, and thus, are the appropriate modelling tool for generating physical events: this is referred to as the generated level [19]. The simulated events would then be passed through another simulation layer to incorporate detection, bringing them to the reconstructed level. Many of the reconstructed events are then grouped into distributions, which are then compared to the measured events to see whether there is any agreement or divergence with respect to theoretical predictions. Recently, many variants of deep generative models have been considered for the same. There has been an evident growth of VAE(variational autoencoders) and adversarial networks for simulations in particle physics [117].

2.3.4 Distributed Computing:

The quest to undertake precise tests of the Standard Model and find evidence of new physics drives the massive resources committed to these enormous experiments. Event analysis, reconstruction, transfer and storage are computationally intensive tasks that eventually lead to resource pooling into a global computer grid. Currently, the Worldwide-LHC-Grid is in operation for resource sharing, monitoring and maintenance of simulation, experimental and processed datasets [39].

2.3.5 Accelerators of the Future

It takes decades to imagine, design, and construct an accelerator. For example, CERN scientists were already planning to replace the old LEP electron-positron accelerator with a more powerful one before it had started operating. That was twenty-four years before the LHC began, in 1984. Scientists have been working on the successor to the LHC, the High-Luminosity LHC, since 2010. The CERN Council approved the second-generation LHC in 2016, and it is scheduled to begin operations around 2025. CERN scientists are also researching accelerators that will be operational beyond 2040, such as the Future Circular Collider(FCC) or the Compact Linear Collider(CLIC). Alternative acceleration methods are also being investigated, such as the AWAKE experiment, a linear accelerator [36].

2.4 Existing Experiments

We focus on particle physics experiments that study the collision of particles at high energies to identify and accurately measure Higgs Boson's properties. The CMS and ATLAS experiments detected the Higgs Boson in 2012 at the LHC at 13 TeV [6,20]. One of the major priorities of the LHC physics program, after the Higgs discovery, is to measure as precisely as possible the Higgs Boson's properties.

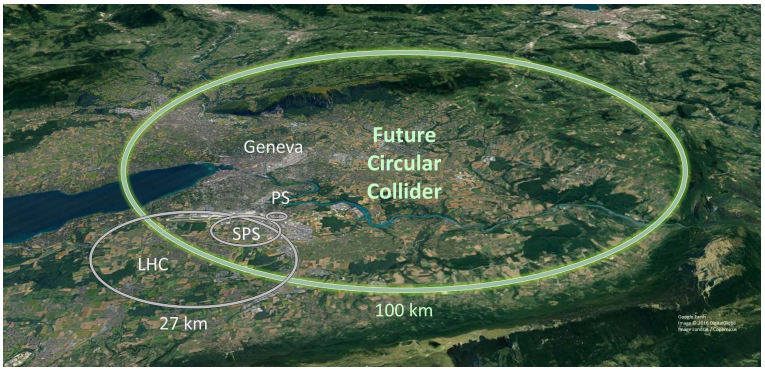

Figure 2.4: Future Circular Collider[145]

2.5 Future experiments

The FCC's mission is to push particle colliders' energy and intensity boundaries in the quest for novel physics to achieve collision energies of 100 TeV. The FCC considers scenarios for three kinds of particle collisions: hadron (proton-proton and heavy-ion) collisions, similar to those seen at the LHC (FCC-hh), and electron-positron collisions (FCC-ee), similar to those seen at the old LEP. Proton–electron collisions and proton–heavy-ion collisions are two more possibilities.

Currently, experiments are being designed to improve the measurements of the Higgs Boson and its properties. More precise experiments will require a new collider, which led to the proposal of Future Circular Lepton Collider (FCC-ee), that collides the electrons and positrons, and will be sensitive to rare phenomena [30][36].

The FCC-ee is a precision machine, colliding point-like particles: electrons and positrons. Such a collider will be precise and sensitive to rare phenomena. The future circular collider e+e will operate at energies below the LHC from 45 GeV to 175 GeV per electron and positron beam. Because electrons and positrons have no internal structure, a lepton collider provides a highly clean experimental environment with minimal backgrounds, allowing accurate observations and precise measurements. The FCC-ee beams will include billions of electrons, compressed to approximately 50 nm before colliding to accomplish the maximum number of collisions[37].

The reference frame used in this research is described with a polar angle, θ, and an azimuthal angle, ϕ, as the two angles. The $x - axis$ is aligned with the

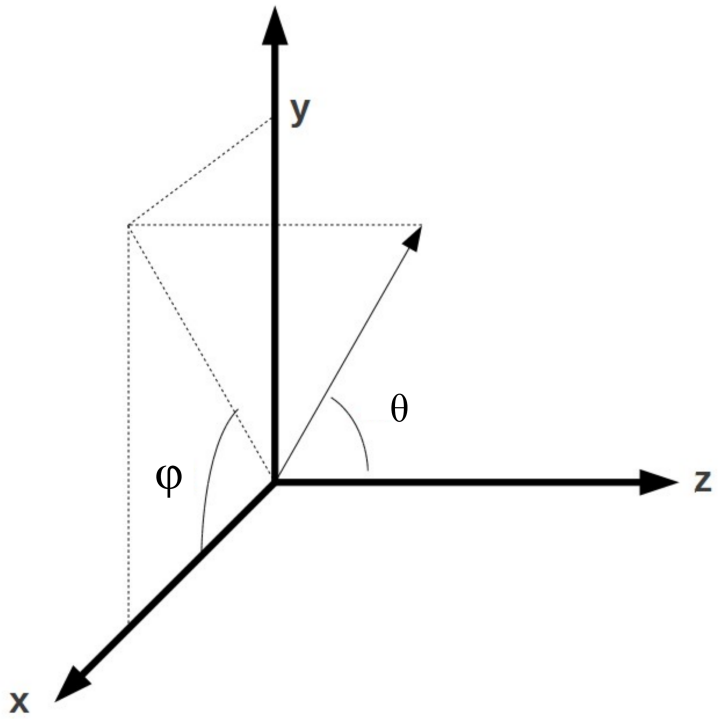

Figure 2.5: Reference Frame: [146]

FCC ring's centre, the $z-axis$ follows the beamline, and the $y-axis$ closes the basis, making the system $x-y-z$ right-handed. Transversal variables exist in the $x-y$ plane and have the subscript T appended to them. (E_T is the energy projected in the transverse plane, and $\vec{p_T}$ is the momentum projected in the x-y plane with a momentum of $\vec{p}$). Instead of the polar angle θ, the pseudorapidity $\eta = -\log\left(\tan\frac{\theta}{2}\right)$ is frequently used in HEP, which maps θ in the range of $[0,\pi]$ to $[-\infty,\infty]$. There is a limit to the range of η, since if θ is near 0 or π, the particle would still be inside the beam pipe and would not be detected.

2.5.1 How is the Higgs Boson produced at FCC-ee?

At an $e+e$ collider, the two main Higgs production mechanisms are Higgs–strahlung and W-fusion Higgs production [13]

2.5.2 Higgs Boson Production in HZ mode:

The Higgs Boson is produced as a result of many-particle interactions, the main production mode concerned with this study is the Higgs Strahlung process, or the

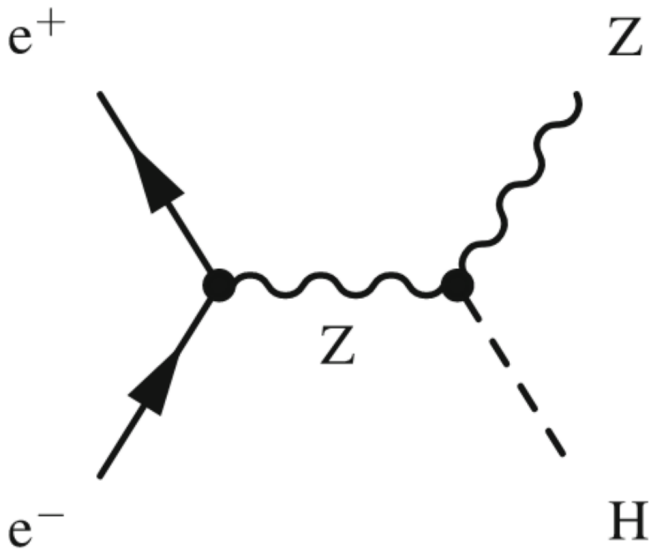

Figure 2.6: Higgsstrahlung Process [30]

HZ mode, where the electron and positron collision decays to HZ, where H is Higgs Boson, and Z is Z boson. The energy regime for this decay is between 240 and 250 GeV [30].

2.5.3 Higgs Boson Decay:

The Higgs Boson and Z boson further decays into a stable particle and has multiple decay modes. The chain of decay particles is often non-deterministic.

2.5.4 Event Decay Chain

When leptons(electrons, positrons) collide at specific energies, they create different particles. After the event collision, in our simulation dataset, the electron-positron collision produces Higgs and Z boson. These intermediate particles further decay into more stable particles, and the final states of the collision are called stable particles. The chain of particle in the decay chain forms an event decay chain. Such a collision generates a cascade of combinations and decays that form a dense cone of particles surrounding the initial decaying particle's track. Due to the law of energy-momentum conservation, the initial particle's momentum is shared among the elements of the end state. The particles are then clustered together based on

their origin of decay. Traditionally, the particles can be clustered by using cone clustering or sequential clustering algorithms [38].

2.6 Role of Engineering and Computer Science in Particle Physics

2.6.1 CERN's Infrastructure

CERN is one of the most challenging computer environments in the world of science. The World Wide Web(WWW) was invented at CERN to satisfy the need for automated information exchange among scientists at universities and institutions worldwide. Computing stands at the core of CERN's infrastructure: software development, data acquisition, processing and storage, networks, support, automation and controls, as well as services for the accelerator complex.The Worldwide LHC Computing Grid (WLCG) — a tier-based distributed computing infrastructure – provides near-real-time access to LHC data to a community of thousands of scientists. The CERN data centre is the hub of the WLCG, serving as the initial point of interaction between LHC experimental data and the grid [39].

2.6.2 Future of Computing in HEP

In 2017, the High Energy Physics community published a roadmap Community White Paper that identifies software research and development investments necessary for HEP experiments for the next decade. First, to achieve improvements in software efficiency, scalability, and performance and to take advantage of advances in computational, storage, and network technologies. Second, to allow novel computer and software methods that may dramatically expand the physics reach of the detectors. Lastly, to ensure data and knowledge preservation and to guarantee long-term software sustainability over the lifespan of the experiment[40].

2.6.3 Challenges in HEP: Physics and Computing

The improvements necessary for the achievement of long-term particle physics goals include: [41] First, a larger number of events are proposed to be collected

to decrease statistical uncertainty in measurements. Second, different ranges of energies should be explored and used to expand the spectrum of possible collision events. Lastly, the introduction of more precise methodologies to enhance the quality of datasets, reconstruct a larger number of events, identify and implement better filters to minimise data transfer bottleneck, and improve the accuracy by finer track reconstruction. Further, as per the European Strategy for Particle Physics report in 2020 [11], the physics community must pursue collaborations with computer science, data science and information technology to develop software and computing infrastructure to exploit the recent advancements in the field.

2.7 Graph Neural Networks in Particle Physics

2.7.1 Introduction

In High Energy Physics, machine learning has a long history, beginning with Multivariate Analysis (MVA) in the early 2000s [42] and continuing to current deep learning applications. Machine learning is frequently used in HEP experiments to train complex inverse functions to infer something about the underpinning physics processes from the data collected in the detector.

Many challenges require data that can be readily expressed as graphs and represented as unordered sets of components with extensive relations and interactions. Unless there is a specific tree structure, they are not easy to express as vectors, grids, or sequences — the format needed by CNNs, and RNNs, respectively — unless there is a specific tree structure [43, 44]. The expressive power of graphs in deep learning has shown significant advancement in social networks, biomedical networks, knowledge graphs and other research areas. Therefore, it is natural to extend successful techniques into particle physics.

Representation of data in particle physics measurements is typically performed in large accelerator facilities such as CERN and Fermilab, with detectors measuring tens of metres in size that record millions of high-dimensional measurements every second.

These detectors are made up of numerous sub-detectors: tracking detectors, calorimeters, muon detectors, and so on — each measuring the trail of particles using

a different sets of techniques. As a result, data in particle physics is heterogeneous. Machine learning could be used at multiple stages of the event, including triggering, reconstruction, simulation, and distinguishing signals from noise in physics analysis.

2.7.2 Related Work

- Reconstruction of calorimeter: A calorimeter is a detector whose objective is to contain and measure a system's total energy. The reconstruction of the incoming particle's energy requires calibration and clustering of the signal from various cells. A graph network-based technique to clustering and assigning the signal in a high granularity calorimeter to two incoming particles is presented in [46]. Two methods are suggested for graph connectedness: one, GravNet, utilises nearest neighbours in a latent space, while the other, GarNet, uses a fixed number of additional nodes in the graph.

- Pileup Mitigation: Misleading, less interesting interactions (pileup) is regarded as noise in the study, and such prevention of pileup is critical for physics analysis at colliders. In particle flow reconstruction [47], state of the art is to compute a pileup weight per particle [48] and utilise it for mitigation. The authors use the gated graph network design [49] in [50] to predict a per particle likelihood of belonging to the pileup part of the collision event. The network has one node for each charged and neutral particle in the event, and the connectivity is imposed to $\Delta R \equiv \sqrt{\delta\phi^2 + \delta\eta^2} < 0.3$ in the azimuth-pseudorapidity plane. The authors of [18] use the graph attention network from [52] to forecast a per-particle pileup probability.

2.7.3 Jet Classification: Jet Tagging

Jets are primarily represented in three ways;

- First, a physical object with observable properties such as several components that led to approaches such as DNN and BDT, widely used in the HEP community[142].

- Second, a jet can be represented as an image, which led to CNNs being used on calorimetry data, often in combination with tracking data. Only a small portion of the particle physics data could be represented as images, which resulted in increased usage of computer vision techniques and improved performances [140, 141]. On the other hand, image representations are limited by the uneven geometry of detectors and the sparsity of the projections used. Image representations may restrict the amount of information retrieved from data due to the inherent loss of information [8].

- Lastly, jets can be represented as a series of branchings that can be ordered (such as the factorisation tree) or be unordered (such as the particle cloud). Such relations led to the application of RNN and GNNs on particle physics data. The term 'particle cloud' refers to a point cloud which has been transformed to consider a jet as an unordered collection of stable or final state particles. When paired with DGCNN to perform complex convolution operations on the graph (made of final state particles), the approach can result in an performance, specifically for jet tagging applications [53]. Many jet classification (tagging) techniques have been improved using graph neural networks, such as ParticleNet, JediNet and ABCNet [18, 19, 21].

2.7.4 Jet Clustering

All of the methods for jet tagging share one thing in common: they all start with a set of constituents chosen using a jet clustering approach. They still rely on non-machine learning techniques to cluster stable particles—for example, all of the methods mentioned above cluster particles using the anti-kt algorithm.

Jet clustering algorithms are divided into two categories: cone algorithms that define jets based on cones in phase space containing the bulk of an event's energy flow and sequential recombination algorithms which merge pairs of particles until a stopping condition is reached [51].

Several researchers have explored how to improve the jet clustering method by considering a range of aspects [56, 57, 58]. While essential for convergence

in the conventional paradigm, these methods are fundamentally constrained by the discreteness of algorithm types and the flexibility provided by a particular algorithm's adjustable parameters.

'As there is no unique method to link hadronic final states with the quark and gluon degrees of freedom that produced them, jet clustering is typically an unsupervised learning problem.[55]' It implies there are no per-particle labels that could be employed to create the jets. However, for uncoloured particles like the W, Z, and Higgs bosons, it is possible to both generate labels and is to feasible to predict the ancestors of the final state particles in an approximate way. One recent work [55] applies graph neural networks for supervised jet clustering for proton-proton collisions at the LHC.

Therefore, it is reasonable to question whether a supervised method for clustering jets could be devised for electron-positron collisions.[55]

Based on the particle's kinematic characteristics, positional information as well as the connection with other particles in the event, a model could be trained to identify individual particles as coming from an ancestor or not. While this method would forego the calculability provided by algorithms such as anti-kt, however, it may be a better method in cases when calculability is not needed.[55] An architecture that can handle variable length sets as input would be required to build such a supervised jet clustering method. Therefore, we turn to graph neural networks to return node-level predictions for the supervised jet clustering.

2.8 Graph Neural Networks in Jet Clustering

Overall, the project aims to use machine learning techniques, particularly graph networks, in a new sub-domain of jet clustering for electron-positron collisions at the future circular collider. Our objective is to build graph neural nets that can exploit both physical features and spatio-temporal relationships between final state particles to classify them based on their ancestor (source of origin), to eventually cluster them together.

2.8.1 Research Questions

- Could machine learning techniques be used to predict the sets of final state particles originating from the Higgs Boson and Z boson in the HZ production mode at the future lepton collider?
- How can graph neural networks be used to model event collisions as graphs and particles as nodes for the supervised task of node classification? We proceed with the assumption that the predicted labels of the nodes would be used to cluster particles coming from the same source together.
- What is the most optimal type of graph data representation of the stable particles detected at a collider? How can we generate relationships or links between final state particles?
- What is the most promising graph neural architecture to correctly predict all the node's labels in a graph?

Our research is a collaborative and interdisciplinary project between the Department of Computer Science and Particle Physics at the University of Oxford. Our research aims to provide a foundation for future physics experiments by investigating the potential of new and specialised computational tools and algorithms based on advanced graph machine learning.

3
Graph Machine Learning

Contents

3.1 What is a graph? 30
3.1.1 What are the types of graphs 30
3.1.2 Mathematical Representations of Graph 31
3.1.3 Properties: Graph Metrics and Node Metrics 31
3.1.4 Graph Network Embedding 33
3.1.5 Graph Representation Learning 34
3.1.6 Machine Learning in Network Science 35
3.2 Message Passing Networks 35
3.2.1 K-Hop Neighbourhood 36
3.2.2 Node Embeddings 37
3.2.3 Message Passing with Self Loops 37
3.2.4 Basic Graph Neural Networks 37
3.3 Graph Convolutional Networks 38
3.3.1 Symmetric Normalisation 38
3.3.2 Graph Convolutional Networks 38
3.4 Set Aggregations 38
3.4.1 Neighbourhood Attention Mechanisms 39
3.5 Generalised Update Functions 39
3.5.1 Over-Smoothing 40
3.5.2 Skip Connections 40
3.5.3 Jumping Knowledge Connections 41
3.6 Overview of Graph Neural Networks 41
3.6.1 History of Graph Neural Networks 41
3.6.2 Types of Graph Neural Networks 42
3.6.3 Training Settings for Graph Neural Networks 42
3.6.4 Computational Modules in Graph Neural Networks . . . 43
3.6.5 Recent advances in Graph Neural Networks 44
3.6.6 Complex Real World Graphs in Network Science 45

3.6.7 Graph Transfer Learning 45
3.6.8 Explainability in Graph Neural Networks 45
3.6.9 Computational Efficiency 46
3.7 Strengths and Weaknesses of Graph Neural Networks 47
3.7.1 Strengths and Advantages 47
3.7.2 Limitations and their Proposed Solutions 48

3.1 What is a graph?

Graphs are data structure instances: a set of nodes(entities) and edges (relationships between nodes). A features is a property of the entity represented by the node, edge or graph. Graphs can consist of the following key elements: nodes, edges, graphs, sub-graphs, node features, edge features, graph features, walks and paths, and local neighbourhoods.

3.1.1 What are the types of graphs

As per Zhou et al., we can categorize graphs as follows [29]

- Directed and Undirected Graphs.

  The edges in the directed graph are directed towards a particular node. Edges depict the direction of a connection between nodes. Alternatively, undirected graphs are connected via edges with no directions, assuming that a single undirected edge between nodes is two directed edges.

- Homogeneous and Heterogeneous Graphs

  Homogeneous graphs have nodes and edges of the same or identical type. Alternatively, heterogeneous graphs have nodes and edges of different types.

- Static and Dynamic Graphs.

  Static graphs are graphs whose topology and input features do not change with time. Alternatively, dynamic graphs are graphs whose topology and input features can change with time. Further, the graphs commonly used in network science include weighted and attributed graphs, multimodal or

heterogeneous graphs, knowledge graphs, multi-layer graphs, temporal graphs and spatial graphs. We could further join the graph types to create new hybrid graph types such as multi-layer spatial graphs or multimodal temporal graphs.[28]

3.1.2 Mathematical Representations of Graph

- An adjacency matrix A of a graph $G = (V, E)$, with a set of nodes V and a set of edges E is defined as a square matrix of size $(|V|X|V|)$ such that the element A_{ij}, is 1 when there is an edge from node i to node j and A_{ij} is 0 when there is no edge. Adjacency matrices are always symmetric when no direction is defined for the edge: for undirected graphs.

- An edge list is another compact representation of a graph that is represented as a list of edges. An edge list, expressed as EL of a graph $G = (V, E)$, is defined as a list of size $|E|$. The element EL_i is a pair: representing the start and the end node of the edge i.

- A degree matrix is defined on a graph $G = (V, E)$ and is defined as matrix of size $(|V|X|V|)$ such that D_{ij} is $\deg(v_i)$ if $i = j$ else, it is 0. Here, $\deg(v_i)$ is defined as the degree of the node and it counts the number of times an edge terminates at that node.

- Laplacian Matrix: A Laplacian of a graph $G = (V, E)$ with $|V|$ nodes is defined as: $L = D - A$ where D is the degree matrix and A is the adjacency matrix.

3.1.3 Properties: Graph Metrics and Node Metrics

Each complex graph comprises a set of intrinsic properties that could be characterised locally or globally. We describe a set of advanced metrics that could be considered to characterise the complicated dynamics of a network [101]. We define these metrics as we use them understand and infer the best representation of the graph data set for our task.

- Integration metrics measure how the nodes of the graph tend to be interconnected with each other.
- Segregation metrics quantify the presence of communities or a group of interconnected nodes inside a graph.
- Centrality metrics evaluate and assess the importance of individual nodes in a graph.

Integration metrics

- Global efficiency is defined for all pairs of nodes in a graph as the average of the inverse shortest path length. It quantifies the extent to which the information transfer in the network is efficient. Let us assume that the l_{ij} is the shortest path between 2 nodes in the graph, let these nodes be node i and node j, global efficiency can be expressed as:

$$\frac{1}{q(q-1)} \sum_{i \in V} \frac{1}{l_{ij}} \tag{3.1}$$

  Here q is $|V|$, which represents the order or the number of nodes of the graph. A fully connected graph will have higher efficiency than a graph of circular shape. It would be shorter to reach any node from a fully connected graph, whereas it would take multiple steps in a circular graph to reach from one node to another.

- Local efficiency of a node is calculated by evaluating the neighbourhood of the specific node, excluding itself.

Segregation metrics

- The clustering coefficient quantifies to what extent are the nodes are clustered together. The clustering co-efficient is defined as the 'fraction of triangles around a node' [101]. A triangle in this context is defined as a completely linked subgraph of three nodes with three edges. The clustering coefficient can be calculated on the node level and as an average on the graph level.

- Transitivity is defined as the ratio between the observed closed triplets and the maximum number of closed triplets in the graph. It's another variant of clustering and is defined as a global property of the graph.

Centrality Metrics

- Degree centrality measures the number of incident edges on a specific node i.

- Closeness centrality quantifies how much a node is close to other nodes in a graph [101]. It refers to the average distance of a node i to all other nodes in the network; if l_{ij} is the shortest path between node i and node j, the closeness centrality is defined as

$$\frac{1}{\sum_{i \in V, i!=j} l_{ij}} \tag{3.2}$$

- Betweenness centrality describes how much a node acts as a bridge between other nodes [101]. Despite having poor connections, a node can still be strategically connected, making sure the graph is overall well-linked. Consider, that L_{wj} is the total number of shortest paths between 2 nodes, node w and node j respectively and $L_{wj}(i)$ is the total number of shortest paths between node w and j through the node i , in that case betweenness centrality is:

$$\sum_{w!=i!=j} \frac{L_{wj}(i)}{L_{wj}} \tag{3.3}$$

3.1.4 Graph Network Embedding

We can analyse graphs at different levels of granularity at the node, edge and graph levels. [101]

- Node level: In a graph $G = (V, E)$, the objective is to classify each vertex v belonging to a set of nodes V into the correct class. The dataset can include a single graph (or multiple graphs) G and a list of pairs of nodes and their corresponding labels, depicting their category.

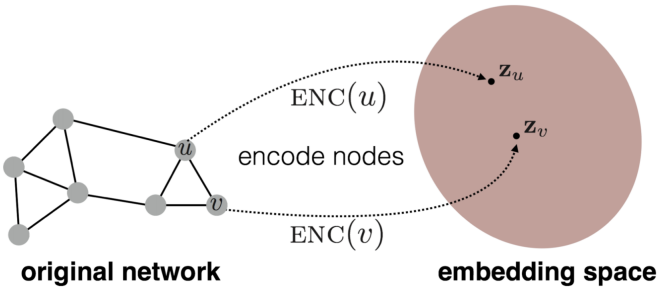

Figure 3.1: Embeddings of a Graph[148]

- Edge Level: In a graph $G = (V, E)$, the objective is to classify each edge e belonging to a set of edges E into the correct class. The dataset also includes a graph G and a list of pairs of edges and their corresponding labels, depicting their category.

- Graph Level: In a graph $G = (V, E)$, the objective is to build an algorithm such that it classifies the whole graph into the correct class. The dataset here would include a set of graphs and and their corresponding labels.

A network embedding is defined as a task that aims to learn a mapping function $f : G \rightarrow \mathbb{R}^n$, from a discrete graph into a continuous domain. Function f will transform the graph G to a low dimensional vector representation to preserve the graph's local and global properties.

3.1.5 Graph Representation Learning

Graph representation learning techniques are used for modelling, analysis, and learning networks in applied sciences such as biology, medicine, physics and chemistry. Network Science is identified as a set of principles to organize complex interactions which connect network structure to different entities. As per [27], it is defined as 'the study of network representations of physical, biological, and social phenomena leading to predictive models of these phenomena.' This discipline draws from graph theory from mathematics, statistical mechanics from physics, data mining and visual analytics from computer science, inferential modelling from statistics, and social structure from sociology.

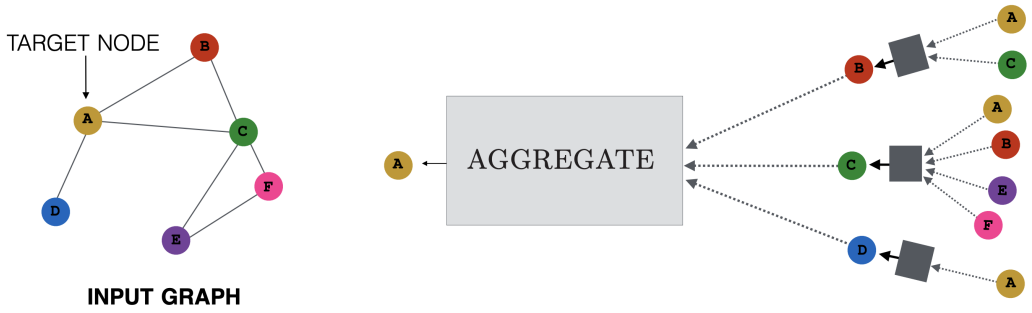

Figure 3.2: Message Passing Networks: Aggregation from a node's local neighbourhood[148]

3.1.6 Machine Learning in Network Science

Machine learning on graphs and network science has been studied for several years. According to [28], it can be broadly categorized into seven methods: graph-theoretic techniques, network diffusion, topological data analysis, manifold learning, shallow network embeddings, graph neural networks and generative models. Theoretic techniques calculate a deterministic value for identifying patterns in a graph. Network diffusion captures the influence and significance of nodes. Topological data analysis summarises the different types of views of the shape of graph data. Manifold learning finds the graphical structure of the data and obtains its low-dimensional embedding. Shallow network embeddings generate node embeddings by directly encoding node similarities in the input graph. Generative models create graphs with properties of interest. In this project, we specifically focus on graph neural networks.

3.2 Message Passing Networks

A GNN is distinguished by the fact that it employs a type of neural message passing [148]. In such a framework, vector messages are passed between nodes and neural networks are used to update them [128].We take an input graph $G = (V, E)$ with node features $\mathbf{X} \in \mathbb{R}^{d \times |\mathcal{V}|}$. Using the node features, we generate an output of node embeddings: $\mathbf{z}_u, \forall u \in \mathcal{V}$.

A hidden embedding $\mathbf{h}_u^{(k)}$ of node $u \in \mathcal{V}$ is updated according to information collected from u's graph neighbourhood $\mathcal{N}(u)$ during each message passing step in

a graph neural network. The message-passing update is mathematically expressed such that the UPDATE function and AGGREGATE function are both arbitrary differentiable functions (such as neural networks). Here $\mathbf{m}_{\mathcal{N}}(u)$ is the aggregated message from u's neighbourhood $\mathcal{N}(u)$ [148]. (To differentiate the embeddings and functions at various rounds of message passing, we apply superscripts.)

$$\mathbf{h}_u^{(k+1)} = \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)},\ \text{AGGREGATE}^{(k)}\left(\left\{\mathbf{h}_v^{(k)}, \forall v \in \mathcal{N}(u)\right\}\right)\right) \tag{3.4}$$

$$= \text{UPDATE}^{(k)}\left(\mathbf{h}_u^{(k)}, \mathbf{m}_{\mathcal{N}(u)}^{(k)}\right) \tag{3.5}$$

.

For each iteration k of the graph neural network, the AGGREGATE function takes the embeddings of the nodes in u's graph neighbourhood $\mathcal{N}(u)$ and transforms them to produce a message $\mathbf{m}_{\mathcal{N}(u)}^{(k)}$ based on the aggregated neighbourhood information. To produce the updated embedding $\mathbf{h}_u^{(k)}$ of node u, the UPDATE function combines the message $\mathbf{m}_{\mathcal{N}(u)}^{(k)}$ with the previous embedding $\mathbf{h}_u^{(k-1)}$ of node u. At $k = 0$, the initial embeddings for all nodes are set to the input features: $\mathbf{h}_u^{(0)} = \mathbf{x}_u, \forall u \in \mathcal{V}$. After performing k iterations of message passing, the output of the final layer of the graph neural network results in node embeddings. GNNs created in this manner are permutation equivariant by design since the AGGREGATE function accepts a set as input[148].

$$\mathbf{z}_u = \mathbf{h}_u^{(K)}, \forall u \in \mathcal{V} \tag{3.6}$$

3.2.1 K-Hop Neighbourhood

At each iteration, each node in the graph collects information from its immediate neighbourhood. With each iteration, every node embedding includes more and more information from the farther reaches of the graph [148]. Every node embedding includes information from its 1-hop neighbourhood after the first iteration when k=1. It means the node embedding contains information about the features of

its immediate graph neighbours, which could be reached by a path of length 1 in the graph. After k iterations, every node embedding contains information from its k-hop neighbourhood.

3.2.2 Node Embeddings

The embedding $\mathbf{h}_u^{(k)}$ of node u can contain graph structural information. It may contain information about the degrees of all the nodes in u's k-hop neighbourhood after k rounds of GNN message passing, which can be helpful for a variety of purposes. The embeddings for each node additionally contain information about all the features in their k-hop neighbourhood after k rounds of GNN message passing.

3.2.3 Message Passing with Self Loops

To simplify the neural message passing method further, adding self-loops to the input graph and skipping the explicit update step is common. We redefine message passing as:

$$\mathbf{h}_u^{(k)} = \text{AGGREGATE}\left(\left\{\mathbf{h}_v^{(k-1)}, \forall v \in \mathcal{N}(u) \cup \{u\}\right\}\right) \tag{3.7}$$

The aggregation step is performed over the node's neighbours, as well as the node itself and the AGGREGATE function, is now applied over the set $\mathcal{N}(u) \cup \{u\}$. The advantage of adding self-loops is that we do not need to explicitly define an update function since the aggregation method defines the update [148]. Overfitting may be reduced by simplifying message passing in this manner, but it significantly restricts the expressivity of the GNN since the information from the node's neighbours cannot be distinguished from information from the node itself.

3.2.4 Basic Graph Neural Networks

The most basic GNN framework is a simplified version of the original GNN models [129, 130] and is mathematically defined as:

$$\mathbf{h}_u^{(k)} = \sigma\left(\mathbf{W}_{\text{self}}^{(k)}\,\mathbf{h}_u^{(k-1)} + \mathbf{W}_{\text{neigh}}^{(k)} \sum_{v \in \mathcal{N}(u)} \mathbf{h}_v^{(k-1)} + \mathbf{b}^{(k)}\right) \tag{3.8}$$

Here, $\mathbf{b}^{(k)} \in \mathbb{R}^{d^{(k)}}$ is the bias term, $\mathbf{W}_{\text{self}}^{(k)}, \mathbf{W}_{\text{neigh}}^{(k)} \in \mathbb{R}^{d^{(k)} \times d^{(k-1)}}$ are trainable parameter matrices and σ is an elementwise non-linearity.

3.3 Graph Convolutional Networks

3.3.1 Symmetric Normalisation

The simplest neighbourhood aggregation procedure merely adds the neighbour embeddings together. This method has the disadvantage of being unstable and extremely sensitive to node degrees. This huge discrepancy in magnitude may cause numerical instabilities and make optimisation challenging [148]. One alternative is to normalise the aggregation process according to the degrees of the nodes involved. Taking an average rather than a total is the simplest method. The authors in [102] apply the symmetric normalisation:

$$\mathbf{m}_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \frac{\mathbf{h}_v}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(v)|}} \tag{3.9}$$

3.3.2 Graph Convolutional Networks

The graph convolutional network is one of the most important fundamental graph neural network models. It uses both symmetric-normalized aggregation and the self-loop update method. As a result, the GCN model defines the message forwarding function as follows:

$$\mathbf{h}_u^{(k)} = \sigma\left(\mathbf{W}^{(k)} \sum_{v \in \mathcal{N}(u) \cup \{u\}} \frac{\mathbf{h}_v}{\sqrt{|\mathcal{N}(u)| \, | \, \mathcal{N}(v) \, |}}\right) \tag{3.10}$$

GCN has proven to be one of the most popular and successful GNN designs. It is commonly used as an effective baseline for most research papers[102].

3.4 Set Aggregations

Neighbourhood aggregation is essentially a set function: a collection of neighbour embeddings $\{\mathbf{h}_v, \forall v \in \mathcal{N}(u)\}$ being mapped to a single vector $\mathbf{m}_{\mathcal{N}(u)}$. Aggregation

functions must be permutation invariant, since there is no natural ordering of a node's neighbours [148].

3.4.1 Neighbourhood Attention Mechanisms

Applying attention [133] is a widely known strategy for improving the aggregation layer in graph neural networks. The fundamental principle behind attention is to assign each neighbour an attention weight or importance. This attention weight is then used to weigh that neighbour's influence during the aggregation phase. Graph Attention Network [52] was the first GNN model to employ the strategy:

$$\mathbf{m}_{\mathcal{N}(u)} = \sum_{v \in \mathcal{N}(u)} \alpha_{u,v} \mathbf{h}_v \tag{3.11}$$

where, $\alpha_{u,v}$ is the attention on neighbor $v \in \mathcal{N}(u)$ as we are aggregate information at node u.The attention weights are calculated as: [52]

$$\alpha_{u,v} = \frac{\exp\left(\mathbf{a}^\top \left[\mathbf{W}\mathbf{h}_u \oplus \mathbf{W}\mathbf{h}_v\right]\right)}{\sum_{v' \in \mathcal{N}(u)} \exp\left(\mathbf{a}^\top \left[\mathbf{W}\mathbf{h}_u \oplus \mathbf{W}\mathbf{h}_{v'}\right]\right)} \tag{3.12}$$

Here, $\mathbf{W}$ is a trainable matrix, $\mathbf{a}$ is a trainable attention vector, and $\oplus$ represents the concatenation operation. We can have several attention heads, similar to the popular transformer architecture [134], but this practice is less frequent in the graph neural networks literature. The representational ability of a GNN model by adding attention can be enhanced, particularly if we have prior knowledge that certain neighbours are more informative than others.

3.5 Generalised Update Functions

The AGGREGATE operator has invited the greatest attention from scholars in academia and has led to various new designs and variants, especially with the GraphSAGE framework that developed the concept of generalized neighbourhood aggregation.[114] However, the UPDATE operator is equally significant in determining the power and inductive bias of a GNN model.

3.5.1 Over-Smoothing

Over-smoothing is a frequent problem with GNNs that generalised update methods may potentially alleviate. The general premise behind over-smoothing is that after multiple iterations of GNN message passing, all nodes in the graph's representations could become remarkably similar. Such a similarity in node representations is especially true in simple GNN models and architectures that use the self-loop update method. Over-smoothing makes it difficult to construct deeper GNN models, which take full advantage of the graph's longer-term dependencies [148]. It can formally be defined by quantifying the influence of each node's input features $\mathbf{h}_u^{(0)} = \mathbf{x}_u$ on the last layer embedding of all other nodes in the graph $\mathbf{h}_v^{(K)}, \forall v \in \mathcal{V}$.

Using the notion of influence, [135] demonstrates that: 'when we are using a K-layer GCN-style model, the influence of node u and node v is proportional the probability of reaching node v on a K-step random walk starting from node u.[148]' However, when K approaches infinity, the influence of each node approaches a stable distribution, implying that local neighbourhood information is lost. Developing and building deeper models could negatively affect performance when employing simple GNN models—especially those that use the self-loop update method. We lose knowledge about local neighbourhood structures as additional layers are added, and our learnt embeddings get over-smoothed, reaching a nearly uniform distribution.

3.5.2 Skip Connections

When the information gathered from node neighbours during message passing starts to dominate the updated node representations, we may anticipate over smoothing. The updated node representations will be too dependent on the incoming message aggregated from neighbours at the cost of the preceding layers' node representations. Using vector concatenations or skip connections, which attempt to explicitly maintain information from earlier rounds of message passing during the update phase, is a logical solution to over smoothing.

$\text{UPDATE}_{\text{base}}$ represents the base update function we build upon. We further define skip-connection updates on top of it. Concatenation is an example of a skip

connection update to retain more node-level information during message passing.

$$\text{UPDATE}_{\text{concat}}\left(\mathbf{h}_u, \mathbf{m}_{\mathcal{N}(u)}\right) = \left[\ \text{UPDATE}_{\text{base}}\left(\mathbf{h}_u, \mathbf{m}_{\mathcal{N}(u)}\right) \oplus \mathbf{h}_u\right] \tag{3.13}$$

where the previous-layer representation of the node is concatenated with the output of the base update function.

3.5.3 Jumping Knowledge Connections

We presume node embeddings utilised in a downstream task are the same as the GNN's final layer node embeddings. However, rather than depending only on the last layer's output, an alternative approach for improving the quality of the final node representations is to utilise the representations at each layer of message passing. The final node representations can be defined as:

$$\mathbf{z}_u = f_{\text{JK}}\left(\mathbf{h}_u^{(0)} \oplus \mathbf{h}_u^{(1)} \oplus \ldots \oplus \mathbf{h}_u^{(K)}\right) \tag{3.14}$$

where f_{JK} is an arbitrary differentiable function and such a strategy is called Jumping Knowledge connections[108]. For many cases, the function is specified as an identity function, implying that we concatenate the node embeddings from each layer.In the paper, authors also consider alternative methods such as max-pooling and LSTM attention layers[108]. Jumping Knowledge often results in consistent gains across a broad range of activities and is a strategy that may be used in a number of situations.

3.6 Overview of Graph Neural Networks

3.6.1 History of Graph Neural Networks

RNNs were the first to be applied on DAG(directed acyclic graphs) in the 1990s [59]. Approximately a decade later, RNNs and Feedforward NNs were used to tackle cycles [60]. The underlying idea behind the methods was to build a state transition system that iterates until it converges. Recently in 2015, CNN architectures led to breakthroughs in all machine learning areas and led to the rediscovery of GNNs. The

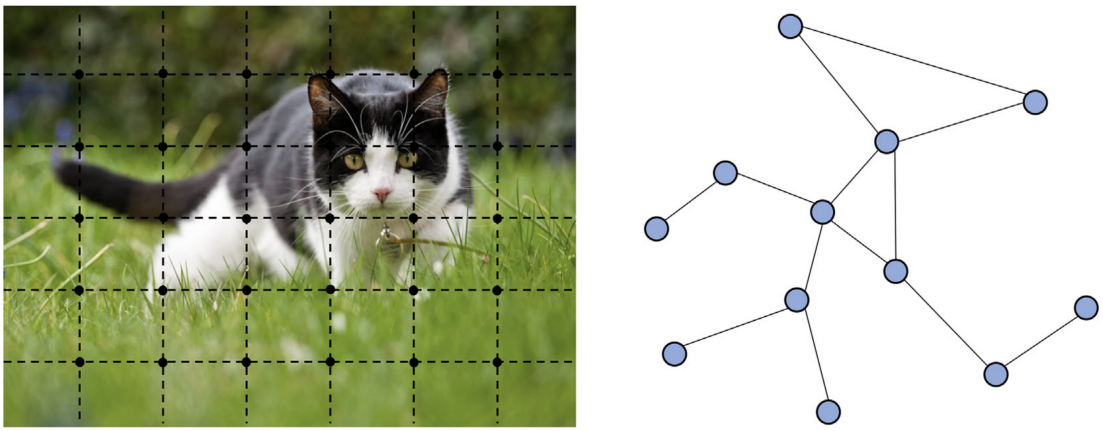

Figure 3.3: Comparison of Euclidean and Non-Euclidean Data

main principles of CNNs are; first, local connections; second, shared weights; third, multiple layers [61], which are equally significant for learning on graphs. However, CNN can only be used for Euclidean data like images and texts. [29] Therefore, geometric deep learning is used to extend deep learning to non-Euclidean spaces[62].

3.6.2 Types of Graph Neural Networks

Graph Convolutional Networks [102] are the most well-known enhanced variants of Graph Neural Networks, which apply the concepts of Convolutional Neural Networks to graphs. The GNNs are further divided into 2 categories: spectral-based approaches and the spatial-based methods. Spectral-based techniques rely on the eigen-decomposition of the graph's adjacency matrix, making them less appropriate for big data processing or generalisation to unknown data. On the other hand, spatial-based techniques rely on aggregating information from each node's neighbourhood, allowing the algorithm to analyse the network in batches and therefore handle huge graphs.

3.6.3 Training Settings for Graph Neural Networks

We have three settings for training: supervised, semi-supervised and unsupervised [29]. In supervised models, labelled data is available for training. In semi-supervised models, we have access to a small set of labelled data and a large set of unlabelled

data for training. Examples of supervised and semi-supervised setting include node classification. In the test phase, the model can be evaluated using inductive or transductive settings. In the inductive settings, the model is trained and tested on unseen dataset points and in transductive settings, the model is trained and tested only on seen data points. Further, there have been recent suggestions of mixed approaches[63]. Lastly, in the unsupervised setting, only unlabelled data is available for the model to generate representations and perform the task; here, node clustering is an example[29].

3.6.4 Computational Modules in Graph Neural Networks

A GNN model is built by combining the computational modules. We have three types of computational modules: propagation modules, sampling modules, and pooling modules.

- Propagation Modules

  In the propagation modules, the information is propagated between nodes such that the aggregated data can collect topological network structure and features. Here, both the convolution and the recurrent operator are utilized to aggregate information from the neighbours. Further, the skip connection operation is utilized to gather data from past representations of the node and alleviate the over-smoothing problem.

- Sampling modules

  The sampling modules are required for large graphs and are often used in combination with the propagation module. Large graphs are defined as those whose graph Laplacian matrix does not fit the device's memory[29].

- Pooling modules

  Pooling modules are often used to derive information from nodes, especially when we require representations of graphs and high-level subgraphs.

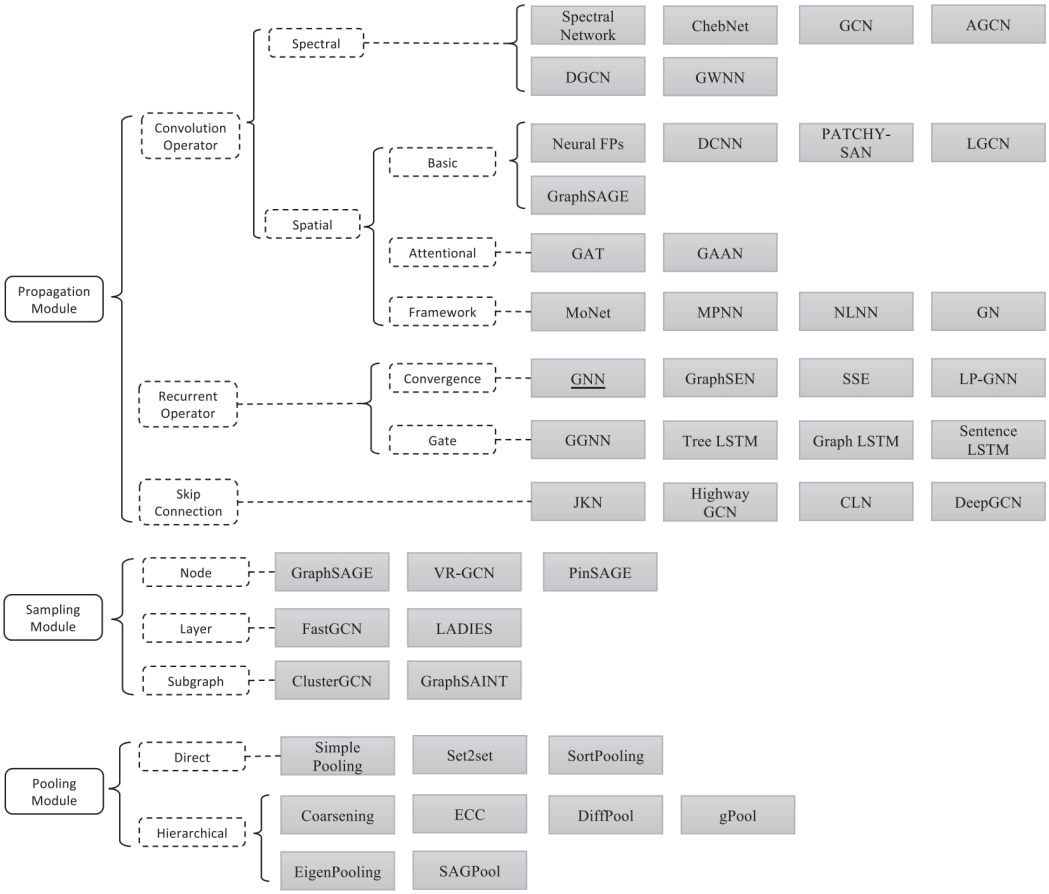

Figure 3.4: Computational Modules of a Graph Neural Network[29]

3.6.5 Recent advances in Graph Neural Networks

GNN variants such as graph convolutional network (GCN)[38], graph attention network (GAT)[52], and graph recurrent network (GRN) have shown leading-edge performance on several deep learning tasks.[29] Graph Attention Networks (GAT) assigns an attention score to each node during the message aggregation step. [52] Further improvements have been led by applying the transformed self-attention mechanism [70] to Graph Neural Networks [71, 72, 73, 74]. GNNs ability to collect graph structural details has been improved by JK-Net [75] that incorporates skip-connections. To capture high order local graph structures, MixHop [76] uses a higher-order adjacency matrix. Graph Pooling methods, such as DiffPool [77], learn a graph's topological structures. To better explain GNNs, practical improvements are complemented by extensive theoretical studies. These theories include linking the

expressive powers of GNNs to the Weisfeiler-Lehman test [78], label propagation[79], and universal invariance [80, 81].

3.6.6 Complex Real World Graphs in Network Science

The progress in standard GNNs is limited and, one could say, overfitted to the popular machine learning tasks. Standard GNNs only recognize homogeneous graphs, but many graphs in network science are heterogeneous. [82, 83, 88] developed new aggregation mechanisms to account for the heterogeneity of different node and relation types in practical and realistic networks. Additionally, standard GNNs can only deal with static graphs and cannot work with dynamic graphs, [82, 84, 85] have suggested different types of dynamic update mechanisms that can take in a series of graphs. [86, 87] recommend sampling methods to increase the scalability of GNNs on real-world data.

3.6.7 Graph Transfer Learning

It is key to use transfer learning to allow rapid adaptation from large pre-trained GNN models, as labels are rarely available due to the extensive expertise required in manual labelling. Recent work, such as [89], have developed unsupervised pre-training techniques for learning better node representations, such as local graph structure context prediction. Further, self-supervised methods[90] are also intended to take advantage of unsupervised network statistics prediction to aid in downstream task prediction.

3.6.8 Explainability in Graph Neural Networks

In computer science, interpretability is defined as the degree to which a person can consistently predict a model's result, but they may not know why. In contrast, explainability is defined as the degree to which a person can understand the underlying cause of the model's result and know why. GNN models are not end-to-end interpretable as a large number of parameters parameterize them. However, they can be explainable to the extent to which they specify why the model made

a particular prediction. For instance, GNNExplainer [91] can be used in graph level prediction tasks to identify relevant sub-graphs and their attributes. Another framework, NIFTY (uNIfyingFairness and stabiliTY), can be compatible with any kind of GNN and be used to learn fair and stable data representations[92].

3.6.9 Computational Efficiency

As per [126], the execution patterns for GCN on the Reddit dataset depict that L1 Cache Miss Rate and Last Level Cache Miss Rate in the aggregation step are very high. Moreover, the Executed IPC is only 0.46. Due to limited feature vectors capable of being loaded into cache, the memory access is random. It causes poor data reuse among neighbour vertices, resulting in poor efficiency of cache utilisation. The peak memory usage almost reaches 32GB leading to a high risk of out of memory issues. Therefore, it is suggested to use state of the art feature decomposition methods to reduce computational costs.

The choice of dense or sparse implementations of the graph's edges is essential when developing and training GNNs on hardware. The number of edges in a graph influences the memory and speed bottleneck because there are usually more edges than nodes.

The use of sparse adjacency matrices allows memory to scale linearly with the number of edges, allowing for the processing of significantly larger graphs. However, the sparse indexing operations necessary to perform sparse matrix multiplication might take longer than their dense equivalents – this is an important area of software and hardware acceleration development. Sparse operations, on the other hand, are a major barrier in present deep learning hardware technology, and if next-generation hardware significantly improves its speed, sparse edge GNN implementations might gain a competitive advantage. Although research is ongoing to improve the infrastructure, physicists do not anticipate having access to dedicated accelerators such as GPU, TPU, or FPGA in a computing environment in HEP. Thus, time complexity of the models in production should be carefully considered[8].

3.7 Strengths and Weaknesses of Graph Neural Networks

3.7.1 Strengths and Advantages

- Graph data representations are more rich and relatable in network science. Many physical systems are modelled as networks from particles, jets and events; the graph's abstract structure can represent them more naturally than other data types such as grids and sequences.

- Graph neural networks are designed for real-world graph data, but not all tasks on the graph could be solved accurately. A common example is whether a GNN can distinguish between two isomorphic graphs and if not, then any task of discriminating them would not yield any positive results. However, since graphs are permutation invariant, it makes it mandatory that two isomorphic graphs will always be indistinguishable. The Weisfeiler-Leman Test is an algorithm for graph isomorphism testing, and GNNs are no more powerful than 1-WL[29]. According to [79], GCNs and GraphSAGE are far less discriminative than the WL(Weisfeiler-Leman) test. GINs (GraphIsomorphism Networks) are proposed as they are more expressive variants of GNN[79]. Recent works investigate GNNs with finite depth and width and discuss the limitations on their behaviour of GNNs with an increase in the model depth.

- Recent literature has seen growing applications of graph neural networks[29] due to its increase in accuracy in classification based tasks. As seen in Chapter 2, they have been shown to outperform both traditional and other deep learning architectures [18, 19, 21].

- It has also been seen that graph neural networks perform faster, as specified here [140]. The stability and generalization of single-layer GNNs have been analyzed with different types of convolutional filters and depend on the filters' biggest eigenvalue[29]. Further, it has been concluded that attention enables GNNs to generalize to larger and noisier graphs [29].

- The outputs of GNN models are permutation-invariant or equivariant to input features [29]. Since many physical entities have a graphical structure but might not be of sequential order, the permutation invariant nature of graphs is suitable for making predictions.

- High-quality benchmark datasets on a large scale help accelerate ML research, and graph learning is in the phase of transitioning from small node classification datasets to large, scalable and reliable datasets.[29]

3.7.2 Limitations and their Proposed Solutions

- Graph neural networks is not the sole answer to all physics-based problems in the ML pipeline but might be a step towards better data representation and therefore leading to better algorithmic results. However, this is an opportunity for multi-disciplinary teams to join their efforts and improve upon existing literature.

- GNNs can be vulnerable to adversarial attacks, which consider their structural information. However, there are recent developments in both how to attack [93] and defend graphs [94], and they are explained in more detail here [95].

- As GNNs are also black-box models, there is a lack of interpretability for most tasks. However, there has been ongoing research and progress to explain the node, edge and graph level predictions [91].

- There can be different types of biases in the dataset and algorithms. As evidenced in other deep learning algorithms[96], GNNs also suffer from the same problems of fairness and equality. In many social and medical contexts, algorithmic bias has been detected against minorities. However, in particle physics experiments, data collection is a more rigorous and ethical process, independent of any social entities. Therefore, bias is less likely to be a big problem. The only bias possible is observer's bias or a sampling bias, at the level of data acquisition when event filters are applied on the detector's output

that could cause inaccurate representations and lead to incorrect predictions. However, such triggers are very well-defined for selecting 'interesting events', thus, are less likely to introduce any major disadvantages.

- There can be difficulties in pre-training on graphs. In NLP and CV, many self-supervised methods have been successfully used to help models learn from unlabelled data[97]. However, recent literature on pre-training on graphs is not sufficient as they focus on different problems and aspects of it[98, 99]. Future work is expected to grow in this direction and significantly aid scientific fields[29].

- There is an abundance of particle data across different research organizations. There are many standards for data acquisition, data simulation, data quality, and data types in HEP. Data sets can be numerical files, image files, event records with various features, and each of these can differ in terms of representation across two experiments alone. Similarly, the software packages and methods to process and distribute the data vary as well. Therefore to build machine learning models across different HEP experiments, they should be built independent of the HEP-related dependencies and rely more on general purpose and open source frameworks.

- GNNs require more computational resources as they are required to perform high dimensional matrix multiplications. Low-scale solutions use free compute from GoogleColab; however, these are not suitable for large-scale deployments. Further, as per empirical observations, edge-related computations have been a major bottleneck, and sampling techniques for GNN are still not efficient enough and need improvement [100].

- A more generic disadvantage of machine learning models in large scale particle physics is it's lack of reproducibility [40]. Due to the scale and complexity of the operation, diversity of the data processing in each organization and confidentiality of research, most experiments are not reproducible.

- Lastly, we are assuming; all organizations have equal access to resources such as computational resources to run GNNs, financial resources to protect data, legal resources to be compliant, and human-based technical resources to fix any bugs. Practically, these changes are only expected to be implemented in the future when deep learning becomes affordable. Despite ongoing research of improvements to the infrastructure, it is less likely that physicists will have access to dedicated accelerators in a computing environment in HEP.

4
Methodology

Contents

4.1 Introduction 52
4.2 Particle Physics Data 52
4.2.1 Event Data 53
4.2.2 Simulated Dataset 53
4.3 Data Generation and Conversion 54
4.3.1 Data handling framework 54
4.3.2 Simulation Frameworks 54
4.3.3 Data Conversion to Compatible Formats 54
4.4 Data Description 55
4.4.1 Dataset Version 1 55
4.4.2 Dataset Version 2 56
4.4.3 Data Analysis and Preprocessing 57
4.5 An overview of the GNN Pipeline 57
4.5.1 Practical Implementation 57
4.5.2 Introduction to Graph Generation in Particle Physics . 57
4.5.3 Construction of a Graph 57
4.5.4 Role of Edges in a Graph 58
4.5.5 Size of a Graph 58
4.6 Definition of the Machine Learning Task 59
4.6.1 What are the entities and relations that could also be represented as nodes and edges, respectively? 59
4.6.2 What is the desired output, such as predictions at the edge, node, or graph level? 59
4.6.3 Is it necessary to have a global output network to create graph-level outputs? 59
4.6.4 How many message-passing steps should be utilised to spread information among the graph's remote nodes? . 59
4.7 Edge-generation strategies 60

4.7.1 edge-KNN 60
4.7.2 edge-radius 60
4.7.3 edge-label 60
4.8 Graph Processing 61
4.8.1 Fixed-size Graph dataset 61
4.8.2 Variable-size Graph dataset 61
4.8.3 Edges in Fixed and Variable size dataset 61
4.9 Model Evaluation Metrics 61
4.9.1 Accuracy 62
4.9.2 Node Accuracy 62
4.9.3 Event Accuracy 62
4.10 Model Methodology 63
4.10.1 Graph Network Blocks 63
4.10.2 Attention mechanism 63
4.10.3 Stacking Graph Network Blocks 64
4.10.4 GNN architectures 64
4.11 Selecting the Graph Network Blocks 64
4.11.1 GCNConv 64
4.11.2 ChebConv 65
4.11.3 TAGConv 66
4.11.4 Jumping Knowledge 66
4.11.5 SAGEConv 67
4.11.6 GINConv 67
4.11.7 GATConv 67
4.11.8 SuperGATConv 68
4.11.9 Conclusion 69

4.1 Introduction

This chapter delves into the project's pipeline, from data collection, processing, analysis to the model implementation. Several iterations and modifications were made to accommodate these intricacies. We build a data building upon the strategies in literature and design graph neural network models suitable for our task.

4.2 Particle Physics Data

Particle physics is an interesting and rich field to apply machine learning techniques. The data collected in its operations is theoretically sound, abundant, and reliable and is straightforward to simulate both the generated and reconstructed data. In

High Energy Physics, 'data' usually refers to data collected from real events in the detector, whereas simulation or 'MC' refers to simulated data.

4.2.1 Event Data

As soon as a collision is identified as interesting after the trigger process, it is recorded. The process of aggregating together all the information from many sub-detectors in a particle collider is called event building. Overall, putting together all the incoming data from the detector is a complex process as the detector elements read out several collisions simultaneously, and only a portion of it is useful and thus, stored. The event size ranges from a few kilobytes to a few megabytes depending on the detector type and its purpose and directly depends on the number of particles generated per collision[116]. As specified, the data of many events together are stored in a single file and temporarily saved on hard disks as buffers. Data is then shipped to permanent storage on tapes and is extracted for either reconstruction or physics analysis. For example, at the current scale and scope of experiments at the LHC, the total amount of data per experiment in a year reaches a few petabytes. [40]

4.2.2 Simulated Dataset

Simulations are required to understand and identify the effects of detectors on observations for accurate and precise measurements. Using the Monte Carlo technique, a detector simulation reproduces the behaviour of the detector as closely as possible. The steps comprising a simulation include: a collision generator simulates which particles are created in a collision, an accurate description of the geometry of all active (sensitive and operating) and inactive (frames, cooling devices, cables) elements in the detector and a transport software to simulate the interaction of the particle in the collision with the detector material[116].

Particles after an event collision can decay, deflect or get stuck. The active detector elements record the energy deposits of the particles. Eventually, the simulated signals are reconstructed with the same software used to reconstruct

the real experimental data. The simulation results allow physicists to correct the distortions caused by the detector on the specific measurement.

4.3 Data Generation and Conversion

4.3.1 Data handling framework

The ROOT framework is C++ based and an object-oriented framework for data processing and analysis and has evolved as the standard tool in the High Energy Physics community. It has been developed since 1995 at CERN and holds extensive 2D and 3D Tools for the visualisation of particle physics data. In this project, we use the Delphes framework, which is directly dependent on the ROOT framework. [112]

4.3.2 Simulation Frameworks

At high-energy colliders, multipurpose detectors are extremely complex systems. Their simulations are typically done with the GEANT [118] package. However, the final observables used in analyses frequently need complicated reconstruction techniques, which consume much time. Delphes is a C++ framework [118, 119, 120] for a fast simulation of a generic particle collider experiment. A particle tracking system embedded in a magnetic field, calorimeters, and a muon system are all a part of the simulation. Delphes supports a range of available data file formats as an input, allowing it to process events from various generators. Delphes outputs data in the form of a ROOT ntuple [120, 121].

4.3.3 Data Conversion to Compatible Formats

The raw data of 10,000 events must be transformed into a compatible format. The simulation dataset was generated with Delphes, and the data had been saved as a .root file. The simulated datasets were produced and shared by Dr Michele Selvaggi and Dr Loukas Gouskos, research scientists, currently working at CERN. The ROOT files allow us to organise physics objects or data into tree structures. Further, we set up a Delphes environment locally and programmed a script to

Dataset Features Original	Physical Significance
N	Node index for each event
PID	Process ID: Identifies a particle
St	Status: Informs if stable or not
Pt	Tranvserse momentum
E	Energy of the particle
Mass	Mass of the particle
Eta	Pseudorapidity
M1	Mother 1 of the current particle
M2	Mother 2 of the current particle
D1	Daughter 1 of the current particle
D2	Daughter 2 of the current particle

Table 4.1: Raw Features of the Dataset

extract the labels and engineered features for our experiment from the .root file and saved them as a .csv file in a tabular format.

4.4 Data Description

4.4.1 Dataset Version 1

Initially, the dataset consisted of various raw features such as process ID, status, physical parameters and node indices of all the decay products in an event collision, as illustrated in the table. An initial dataset contained all the particles in the event decay chain and was a complete picture of the event collision. While processing the first dataset, our goal was to discriminate between particles decaying from the charm and anti-charm quarks in an event decay chain. However, after data analysis and exploration on the first version of the dataset, we found out that the daughter particles of charm and anti-charm quark were conflicting, leading to issues in discriminating stable particles into these two classes. Therefore, we could not generate labels for particles if they were decaying from charm quark or not. The domain experts informed us that we could not separate particles decaying from the charm and anti-charm quarks, as explained in Chapter 2. Thus, we reassessed and revised the problem statement to classify the particles decaying from bosons

Node Features	Physical significance
PID	Process ID
pos_r	position: vertex 4-vector(distance from the centre)
pos_theta	position:vertex 4-vector(polar angle)
pos_phi	position:vertex 4-vector(azimuth angle)
pos_t	position:vertex 4-vector(time)
mom_p	momentum 4-vector(transverse momentum)
mom_theta	momentum 4-vector(polar angle)
mom_phi	momentum 4-vector(azimuth angle)
mom_mass	momentum 4-vector(mass)

Table 4.2

instead of quarks. In the end, we finalised and reframed the problem to discriminate between particles decaying from the Higgs and Z boson during an $e - e+$ collision.

4.4.2 Dataset Version 2

We proceed to generate the labels based on the features available in simulation data sets. There are three categories of labels: particles decaying from Higgs, particles decaying from Z Boson, and particles decaying from others. Labels were generated using the information about the decay of each particle in an event decay chain: using variables 'D1' and 'D2'. We aim to train a model on the simulation dataset using the labels generated. Later, we will test the model on the simulation dataset, where we will have access to the labels required for calculating accuracy and evaluating the model.

As explained in Chapter 2, only the final state(stable) particles of the event decay chain are detected or observed during the experiment. Therefore, we filter and process our dataset based on stable particles only.

Further, we enhanced the dataset, selected features representative of the task and incorporated engineered features to the dataset: such as the process ID, vertex-4-vector calculated with the particle's position(x,y,z,time) and the momentum 4-vector. Overall, these nine features cover the relevant spatial and physical properties related to the particle [123].

4.4.3 Data Analysis and Preprocessing

Most machine learning algorithms perform considerably better when dealing with features on the same scale; thus we standardise our data using min-max scaling as it is an important step in the preprocessing phase. The dataset is of high quality and we don't have any missing values since it's a simulation generated dataset.

4.5 An overview of the GNN Pipeline

The GNN pipeline contains the following steps: First, we find graph structure in the data. Second, we identify the graph type and scale. Third, define the kind of training and design loss function accordingly. Lastly, we build the GNN model using task-specific computational modules as [29].

4.5.1 Practical Implementation

The most common programming frameworks are pytorch and pytorch geometric, deep graph library from Amazon, graph nets from Deepmind, spektral with tensorflow, and Jax with jgraph[15]. The complexity level of technical documentation and ease of use and operation for the software frameworks vary significantly. We can find exhaustive state of the art software frameworks and hardware accelerators exclusively for graph neural networks here[127]. We develop our data and models using: sklearn, numpy, pandas, matplotlib, pytorch, and pytorch-geometric.

4.5.2 Introduction to Graph Generation in Particle Physics

According to domain experts, a critical task in designing algorithms is applying our understanding of the underlying physics to capture the relationship between the nodes in the input graph and the model architecture.

4.5.3 Construction of a Graph

The molecular structure and social networks have inherent connections that can be easily represented as nodes and edges, with both of them signifying an entity and a relation. However, the nature of the relationships between distinct nodes in a

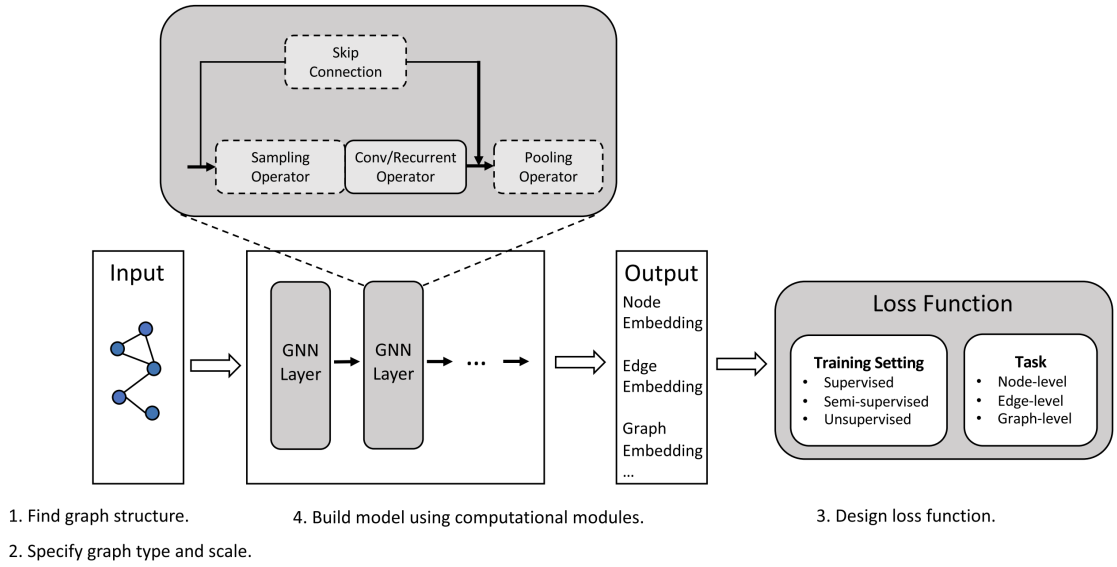

Figure 4.1: Graph Neural Networks Pipeline [29]

graph is unclear in most particle physics applications. As a result, a well-rounded decision must be taken regarding how to create a graph from the inputs.

4.5.4 Role of Edges in a Graph

The graph's edges serve three purposes: first, the edges are node-to-node communication channels. Second, the input edge features can denote a relationship between objects and encode physics-motivated variables about that relationship. Third, latent edges hold relational information computed during message transmission, allowing the network to encode variables it considers important to the task.

4.5.5 Size of a Graph

When the input sets are small, the most common and straightforward solution is to create a fully linked graph, which allows the network to learn which object connections are essential. The computational cost of utilising a neural network to construct an edge representation or compute attention weights becomes prohibitive in bigger sets when the number of edges between all nodes increases.

4.6 Definition of the Machine Learning Task

The first task is to determine the data representation for the specific application. Next, it should be decided if the model learns a function at the node, edge or graph level. A trivial case in HEP involves applications of the jet, event or particle classification. In our project, we perform particle classification by identifying the origin of decay of the final state particle. Many decisions were taken on how the data may be portrayed as a graph and how to compute the embeddings using available GNN architectures:

4.6.1 What are the entities and relations that could also be represented as nodes and edges, respectively?

We represent an event collision as a graph and the stable particles as nodes. The edges are generated based on literature approved strategies as discussed in the following sections as edge-radius, edge-knn, edge-label.

4.6.2 What is the desired output, such as predictions at the edge, node, or graph level?

The desired outcome is a node-level prediction of the particle class: whether it decayed from an H boson, Z boson, or other particles in an $e + e-$ collision.

4.6.3 Is it necessary to have a global output network to create graph-level outputs?

Our node classification task is limited to the Higgs production in the HZ mode. Therefore, all the event graphs are of the same type of decay. Thus, event classification or identification is out of the scope of our research unless other Higgs production modes are also investigated.

4.6.4 How many message-passing steps should be utilised to spread information among the graph's remote nodes?

We perform experiments with network depth to evaluate the optimal number of message passing steps for node classification.

Dataset	Parameters
edge-radius	radius = 0.2
edge-knn	K = 8
edge-label	NA

Table 4.3: Hyperparameters for Dataset Generation

4.7 Edge-generation strategies

We are working with an undirected, static, and homogeneous graph where nodes though belonging to different classes are essentially of the same type or entity, that is, particles. The scale of the graph (more than 50 nodes) makes it difficult to construct a graph with fully connected edges, as suggested in [8]. We implement and construct graphs in three different ways. Edge Generation is implemented as an edge list in the COOCoordinate list format) format, and as the data objects of the pytorch-geometric library.

4.7.1 edge-KNN

As per the domain experts in [8], 'Given a distance measure between nodes, some criterion for connecting them needs to be formulated, such as connecting k-nearest neighbors in the feature space.' Therefore, our first strategy is to connect the k-nearest neighbours of each node for each event separately.

4.7.2 edge-radius

Motivated by the methodology in [25], we project the features in 2 dimensions using principal component analysis and calculate the pair of distances for each node in the graph. Here, euclidean distance between each pair of nodes was calculated. We specify a radius threshold, and we connect all the particles within a specific radius together.

4.7.3 edge-label

We also build a graph dataset by connecting all the nodes of the same class in an event (or having the same labels) to assess the performance difference by comparing

Graph dataset	Number of nodes*	Number of edges
edge-radius	60	3540
edge-knn	60	480
edge-label	60	3368

Table 4.4: A sample event is used to illustrate the characteristics

it with edge-radius and edge-knn.

4.8 Graph Processing

4.8.1 Fixed-size Graph dataset

We padded extra nodes to ensure the event graph had the same number of nodes for each event. We also created a new class label for the padded nodes to accommodate the new set of nodes for model training.

4.8.2 Variable-size Graph dataset

The graph dataset was created with a variable number of nodes per event. The minimum number of nodes per event was 48, whereas the maximum number was 130, in our dataset.

4.8.3 Edges in Fixed and Variable size dataset

Edges were only generated for real particle nodes, such that the padded nodes in the fixed-size graph were not connected (isolated nodes). On the other hand, edges in the variable-sized graphs were generated for each node based on the three strategies described.

4.9 Model Evaluation Metrics

We define two measures of success for supervised node classification: node accuracy and event accuracy. We evaluate the model performance on event accuracy as it helps discriminate and compare the different models more effectively.

4.9.1 Accuracy

Accuracy that the node or particle is correctly classified in an event.

$$\text{Accuracy } = \frac{\text{Number of correctly classified nodes}}{\text{Total number of nodes in an event}} \tag{4.1}$$

To illustrate this, we share an example: let us assume that total number of nodes (stable particles in an event)is 70, and number of correctly classified nodes is also 70, then the accuracy for a single event over all nodes is given by: $70/70 = 100$ percent.

4.9.2 Node Accuracy

Accuracy that the node or particle is correctly classified in an event. Node accuracy is the means of all accuracies reported over the size of the dataset n.

$$\text{Node Accuracy } = \frac{1}{n}\sum_{i=1}^{n} a_i = \frac{a_1 + a_2 + \cdots + a_n}{n} \tag{4.2}$$

where a_i is the accuracy of the ith event collision in the dataset of size n.

To illustrate this, we share an example: let us assume that total number of events is 3, and accuracy for each event is 96, 98, and 99, then the node accuracy for a single event over all nodes is given by: $(96 + 98 + 99)/3 = 97.6666$ percent.

4.9.3 Event Accuracy

Event Accuracy is calculated on a set of events, where each event has a specific node accuracy, then we obtain the frequency of events that were classified with 100 percent node accuracy. We further illustrate this with an example:

$$EventAccuracy = \frac{\text{Number of event classified with 100 percent node accuracy}}{\text{Number of events in the dataset}} \tag{4.3}$$

To illustrate this, we share an example: let the, number of event classified with 100 percent node accuracy be 80, and the number of events in the test dataset (dataset size) be 100, the event accuracy is given by $80/100 = 80$ percent.

4.10 Model Methodology

Graph Neural Networks learn a representation of the graph to generate a low-dimensional embedding of the graph nodes. The end-to-end training allows for the learning of this representation to represent the structural properties of the particle graph that are relevant to the task of node classification. When a task requires classification at the node level, the embedding's representation of each node is directly used. Therefore, we utilise the node embeddings generated by the model to make predictions.

4.10.1 Graph Network Blocks

The update functions and aggregation functions form a GN(Graph Network) block as discussed in Chapter 2. The update functions accept a set of entities, apply the same function to each entity, resulting in an updated representation. The aggregation functions take a set of entities and produce a fixed-size representation for the whole set by implementing an order invariant function to group together the representations of the entities[125]. The model architecture is designed to represent a logical combination of inputs pertinent to the learning task. We specify which GN blocks are used and how we stack them. In many cases, the update functions may be implemented as neural networks, giving the learning processes the most flexibility. As it would incur computational expenses, we started with a small scale efficient design and progressively increased its complexity until the algorithm's performance met our expectations.

4.10.2 Attention mechanism

An essential factor to consider when developing and implementing an attention mechanism is whether certain aspects of the input data are more significant than others. They are all based on the idea of computing weights that indicate the relative importance of different elements in a set using a neural network or a predefined function. These weights are used to construct weighted sums of the representations of the individual elements in the GN block functions [8].

4.10.3 Stacking Graph Network Blocks

GN blocks placed consecutively serve the same purpose for building the node representations as stacked layers in any neural network design such as a CNN or DNN. As a result, introducing more GN blocks to the model increases its depth and expressive power. Further, the node has only exchanged information with its immediate connected neighbours after one iteration of the message passing in a single GN block. Many iterations with a GN block, either the identical block applied multiple times or different blocks applied in a sequence, enhances each node's receptive field since its neighbouring nodes' representations were previously updated with information from their neighbours. [8] Skip connections (residual connections), which combine the input and output, prevent corruption of the updated representations and preserve the gradient signal over multiple message-passing steps [79].

4.10.4 GNN architectures

Different architectures exist for designing a GNN, but they are all equivalent as their output comprises a graph with learned node representations. These embeddings are further used to perform the actual task [8, 125]. We explore the Graph Convolutional Network[102] in depth and we also outline other GN blocks used for our research project: GAT[52], SAGE[114], GIN[79], ChebNet[115], JKNet[108] with GCN, TAGCN[144] and SuperGAT[113].

4.11 Selecting the Graph Network Blocks

To compare the performance of GNNs against NNs, we first implement an MLP to identify model performance solely on node features.

4.11.1 GCNConv

We build the Graph Convolutional Network[102] and establish a GNN baseline. In all model experiments, we stack the convolutional layers in a depth-wise fashion and train each model variant on all the three graph datasets generated.

The graph convolutional operator is given by:[136]

$$\mathbf{x}'_i = \mathbf{\Theta} \sum_{j \in \mathcal{N}(v) \cup \{i\}} \frac{e_{j,i}}{\sqrt{\hat{d}_j \hat{d}_i}} \mathbf{x}_j \tag{4.4}$$

where $\mathbf{x}_i$ is the node embedding of node i, $\mathbf{x}_j$ is a node embedding of node j, $\mathcal{N}(v)$ is the neighbourhood of node v, Θ is the weight matrix, with $\hat{d}_i = 1 + \sum_{j \in \mathcal{N}(i)} e_{j,i}$ where $e_{j,i}$ represents the edge weight from the source node j to the target node i and is by default set to 1.0.

4.11.2 ChebConv

Next, we explore spectral GNNs in the domain such as ChebNet[115], and TAGCN[144]. The ChebNet model allows us to conduct convolution on graph-like datasets. Convolution in the spatial domain is equal to multiplication in the Fourier domain, according to the convolution theorem. As a result, rather than executing convolution directly in the spatial domain, the graph data and filter will be transformed into the Fourier domain. We apply element-wise multiplication, then use the inverse Fourier transform to transfer the result back to the spatial domain[115]. The authors propose the spectral filters to be strictly localised in a ball of radius k, where k refers to the number of hops from the central node. We implement ChebNet to compare its performance with GCNs and assess how spectral graph neural network methods perform on the task.

Chebyshev spectral graph convolutional operator is described as:[136]

$$\mathbf{X}' = \sum_{k=1}^{K} \mathbf{Z}^{(k)} \cdot \mathbf{\Theta}^{(k)} \tag{4.5}$$

where $\mathbf{Z}^{(k)}$ is computed recursively:

$$\begin{aligned} \mathbf{Z}^{(1)} &= \mathbf{X} \\ \mathbf{Z}^{(2)} &= \hat{\mathbf{L}} \cdot \mathbf{X} \\ \mathbf{Z}^{(k)} &= 2 \cdot \hat{\mathbf{L}} \cdot \mathbf{Z}^{(k-1)} - \mathbf{Z}^{(k-2)} \end{aligned} \tag{4.6}$$

where $\hat{\mathbf{L}}$ is the scaled and normalized Laplacian $\frac{2\mathbf{L}}{\lambda_{\max}} - \mathbf{I}$

4.11.3 TAGConv

As we highlighted that physicists do not anticipate having access to dedicated accelerators, the time complexity of the models in production should be carefully considered. We experiment with the spectral TAGCN method. It offers a method for designing a set of fixed-size learnable filters to conduct convolutions on graphs in a systematic manner. When these filters scan the graph to conduct convolutions, their topologies adapt to the graph's topology. TAGCN inherits the characteristics of convolutions in CNN for grid-structured data and follows the definition of convolution in graph signal processing. Spectral graph convolutional neural networks require approximation to the convolution to reduce computational complexity, resulting in performance degradation. However, as per [144], TAGCN is computationally cheaper since no approximation to the convolution is required.

Topology adaptive graph convolutional networks operator can be defined as follows:

$$\mathbf{X}' = \sum_{k=0}^{K} \left(\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\right)^k \mathbf{X}\mathbf{\Theta}_k \tag{4.7}$$

where A denotes the adjacency matrix and $D_{ii} = \sum_{j=0} A_{ij}$ its diagonal degree matrix and k denotes the number of hops and by default it is set to 3.

4.11.4 Jumping Knowledge

Additionally, we also implement Jumping Knowledge (JKNet) networks, designed to solve over-smoothing as discussed in Chapter 3. In the JKNet[108] model, we perform the aggregation using a weighted summation of node embeddings where a bi-directional LSTM provides the attention scores.

Jumping Knowledge layer is an aggregation module based on the weighted summation computed as:[136]

$$\sum_{t=1}^{T} \alpha_v^{(t)} \mathbf{x}_v^{(t)} \tag{4.8}$$

here attention scores $\alpha_v^{(t)}$ are provided by a bi-directional LSTM.

4.11.5 SAGEConv

Further, we move into the spatial domain and build a basic graph network block using the SAGE[114] convolutional operator. SAGE is suitable for the inductive learning setting, as the model is can be trained on a collection of graphs to produce node embeddings for previously unseen nodes or unseen graphs, as long as the graphs have the same attribute format as the training data.

The Graph SAGE operator can be defined as follows, where W_1, W_2 are weight matrices: [136]

$$\mathbf{x}'_i = \mathbf{W}_1\mathbf{x}_i + \mathbf{W}_2 \cdot \text{mean}_{j\in\mathcal{N}(i)}\,\mathbf{x}_j \tag{4.9}$$

4.11.6 GINConv

However, as per [79], both GCN and SAGE neighbourhood aggregations are limited in their discriminative power, so we proceed to build a model with GIN convolutional operators.

The graph isomorphism operator which helps build more expressive graphs is defined as follows: [136]

$$\mathbf{x}'_i = h_\Theta\left((1+\epsilon)\cdot\mathbf{x}_i + \sum_{j\in\mathcal{N}(i)}\mathbf{x}_j\right) \tag{4.10}$$

where ϵ is a scalar with a default value of 0 and h_Θ is a neural network (MLP).

4.11.7 GATConv

Similarly, we extend our experiment to include a model with graph attention layers, and we implement the baseline attention mechanism in GAT[52] as discussed in Chapter 3 and compare it with the current state of the art SuperGAT[113].

Graph attentional operator, as discussed in Chapter 3 is defined as: [136]

$$\mathbf{x}'_i = \alpha_{i,i}\mathbf{\Theta}\mathbf{x}_i + \sum_{j\in\mathcal{N}(i)}\alpha_{i,j}\mathbf{\Theta}\mathbf{x}_j \tag{4.11}$$

where the attention weights $\alpha_{i,j}$ are calculated as:

$$\alpha_{i,j} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^\top \left[\mathbf{\Theta}\mathbf{x}_i \| \mathbf{\Theta}\mathbf{x}_j\right]\right)\right)}{\sum_{k \in \mathcal{N}(i)\cup\{i\}} \exp\left(\text{LeakyReLU}\left(\mathbf{a}^\top \left[\mathbf{\Theta}\mathbf{x}_i \| \mathbf{\Theta}\mathbf{x}_k\right]\right)\right)} \tag{4.12}$$

4.11.8 SuperGATConv

The self-supervised graph attention network (SuperGAT) is an enhanced graph attention model for noisy graphs. Authors in [113] employ a self-supervised task in which the attention value is used as input to estimate the probability of a node-to-node connection. To understand how edges are encoded in graph attention, they understand what graph attention learns and how it relates to the existence of edges. SuperGAT focuses on two widely utilised attention mechanisms, GAT's original single-layer neural network (GO) and dot-product (DP). Two SuperGAT versions are explored, scaled dot product (SD) and combined GO and DP (MX) to examine the strength of GO and DP. We implemented superGAT to assess how the average degree of the various graph datasets and their homophily can determine which of the two graph attention models could best capture the relational significance and offer the most accurate node representations.

The SuperGATConv operator is defined as follows:

$$\mathbf{x}'_i = \alpha_{i,i}\mathbf{\Theta}\mathbf{x}_i + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j}\mathbf{\Theta}\mathbf{x}_j \tag{4.13}$$

where the two types of attention are computed as:

$$\alpha_{i,j}^{\text{MX}} \text{ or SD} = \frac{\exp\left(\text{Leaky ReLU}\left(e_{i,j}^{\text{MX or SD}}\right)\right)}{\sum_{k \in \mathcal{N}(i)\cup\{i\}} \exp\Big(\text{ LeakyReLU }\left(e_{i,k}^{\text{MX}} \text{ or SD}\right)\Big)} \tag{4.14}$$

$$e_{i,j}^{\text{MX}} = \mathbf{a}^\top \left[\mathbf{\Theta}\mathbf{x}_i \| \mathbf{\Theta}\mathbf{x}_j\right] \cdot \sigma\left(\left(\mathbf{\Theta}\mathbf{x}_i\right)^\top \mathbf{\Theta}\mathbf{x}_j\right) \tag{4.15}$$

$$e_{i,j}^{\text{SD}} = \frac{\left(\mathbf{\Theta}\mathbf{x}_i\right)^\top \mathbf{\Theta}\mathbf{x}_j}{\sqrt{d}} \tag{4.16}$$

The self-supervised task is a edge prediction task which uses the attention values as input to estimate the probability of a node-to-node edge.

$$\phi_{i,j}^{\mathrm{MX}} = \sigma\left(\left(\mathbf{\Theta}\mathbf{x}_i\right)^{\top} \mathbf{\Theta}\mathbf{x}_j\right) \tag{4.17}$$

$$\phi_{i,j}^{\mathrm{SD}} = \sigma\left(\frac{\left(\mathbf{\Theta}\mathbf{x}_i\right)^{\top} \mathbf{\Theta}\mathbf{x}_j}{\sqrt{d}}\right) \tag{4.18}$$

4.11.9 Conclusion

We discussed the process and reasons to create three different strategies for edge-generation, selected the most suitable architectures according to the literature and designed eight graph neural networks variants by stacking the same type of GN blocks in a depth-wise fashion.

5 Results

Contents

5.1 Introduction 71
5.1.1 Event wise classification 71
5.1.2 Multiclass-classifier: 71
5.1.3 Default Case: 72
5.2 MLP: Establishing a Baseline with neural networks . . 72
5.2.1 Objective: MLP-2 72
5.2.2 Results: 72
5.3 MLP: Network Depth 73
5.3.1 Objective: Network Depth of 2, 4, 8 73
5.3.2 Results: 73
5.4 Graph Convolutional Network: An in-depth exploration 73
5.4.1 Objective: GCN-2 73
5.4.2 Results: 73
5.5 GCN: Most Representative Dataset 74
5.5.1 Objective: GCN-2 on edge-radius, edge-KNN, edge-label 74
5.5.2 Results: 75
5.6 GCN: Network Depth Experiment 75
5.6.1 Objective: Network Depth of 2, 4, 8, 16 75
5.6.2 Results: 76
5.7 GCN: Accuracy with Different Dataset Sizes 80
5.7.1 Objective: Dataset split into 0.2, 0.4, 0.6, 0.8 80
5.7.2 Results: 80
5.8 GCN: Hyperparameter Search 81
5.8.1 Objective: Find optimal hyperparameters for GCN-2 on edge-KNN 81
5.8.2 Results: 81
5.9 GNN-variants: Model Evaluation and Comparison . . 81

5.9.1 Objective: MLP, GCN, ChebNet, SAGE, GAT, GIN, JKNet, TAGCN and SuperGAT 81
5.9.2 Results: 83
5.10 Conclusion 85
5.10.1 Could machine learning techniques be used to predict the sets of final state particles originating from the Higgs Boson and Z boson in the HZ production mode at the future lepton collider? 85
5.10.2 How can graph neural networks be used to model event collisions as graphs and particles as nodes for the supervised task of node classification? 85
5.10.3 How can we generate relationships or links between final state particles? What is the most optimal type of graph data representation of the stable particles detected at a collider? 86
5.10.4 What is the most promising graph neural architecture to correctly predict all the nodes' labels in a graph? . . . 86

5.1 Introduction

This chapter examines and evaluates the results acquired during the research and the project's objectives and methods for investigating our research questions. Each experiment reported in this part originated from interrogating previous initiatives. The selected layout aims to follow this natural sequence, justifying the actions taken and offering a comprehensive view of the results obtained.

5.1.1 Event wise classification

We performed the particle classification task in an event wise fashion on the GNN models. Event-wise implies that the model processed every event(graph), and then it made predictions on each particle(node) before proceeding to the next event(graph). Thus, supervised node classification was carried out by considering all particles of a single event within a batch.

5.1.2 Multiclass-classifier:

We aimed to process the events(graphs) and classified each particle(node) of the event(graph) into three categories: decaying from the Higgs boson, Z boson or other

particles. We implemented multi-classification on three classes using variable event graphs (with varying numbers of particles). The variable-sized graphs outperformed fixed-sized graphs in our setup, so we have performed all our experiments with variable-sized graphs.

5.1.3 Default Case:

We split the dataset into 60:20:20 across training, validation and test phases. We set 16 hidden channels, and processed the graphs with variable sizes in the default case. The ReLU activation function is used in all the models. To prevent overfitting, a dropout layer is added, with a dropout rate of 0.5. A softmax activation function follows the output layer with one neuron. The Adam Optimizer is used to train the model, using a learning rate of 0.01. The cross-entropy loss function is used as the loss function. All model performance metrics, node accuracy and event accuracy are reported on the test dataset.

5.2 MLP: Establishing a Baseline with neural networks

5.2.1 Objective: MLP-2

Our objective is to analyze and report how the neural network performs on the supervised learning task of classifying final state particles based on their source of decay using only the node(particle) features. The experiment is carried out to establish a baseline that will be successively improved upon in the project. The experiment aims to build a comparison between the neural network baseline against the graph neural network baseline. Further, it also helps us observe if the neighbourhood aggregation techniques add any significant gains in performance.

5.2.2 Results:

The two-layer MLP-2 model was trained on the node features of the dataset and established a baseline result with an event accuracy of 87.2637 per cent.

Model	Network Depth	Dataset	Event Accuracy	Node Accuracy
MLP	2	Node Features	87.2637	99.9538
MLP	4	Node Features	83.3637	99.9494
MLP	8	Node Features	0	95.8934

Table 5.1: MLP: Model Performance with Network Depth

5.3 MLP: Network Depth

5.3.1 Objective: Network Depth of 2, 4, 8

Our objective is to train the neural network model on the node features with varying network depths and assess if deep neural networks increase the event accuracy performance metric.

5.3.2 Results:

We observed that MLP-2 (87.2637 percent) outperforms MLP-4 (83.3637 percent) and MLP-8(0 percent) on the event accuracy metric. Therefore, we observe shallow neural networks are more effective than deep neural networks.

5.4 Graph Convolutional Network: An in-depth exploration

5.4.1 Objective: GCN-2

Our objective is to establish a baseline for GNN models using the GCN model. Further, we analyze the training and loss curves to identify the number of epochs fixed for all GNN experiments. The objective is to ensure a fair model comparison against other architectures.

5.4.2 Results:

The two-layer GCN-2 model built with the default parameters, results in an event accuracy of 90.7206 per cent on the KNN dataset. The number of epochs is set to 51 as the node classification accuracy saturates after 35 epochs. The GNN

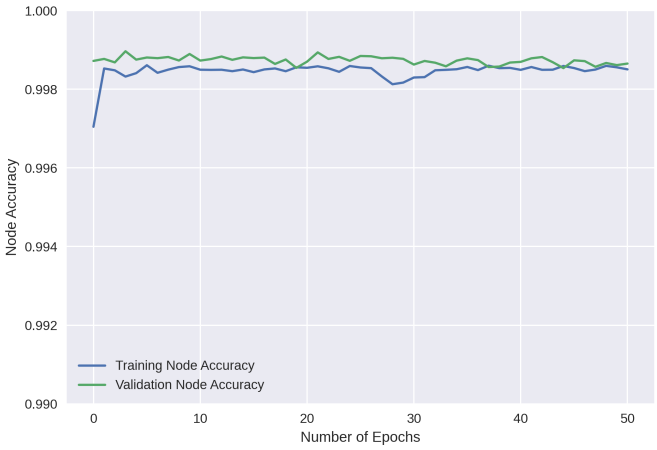

Figure 5.1: Training and Validation Accuracy Curves on GCN-2: edge-KNN

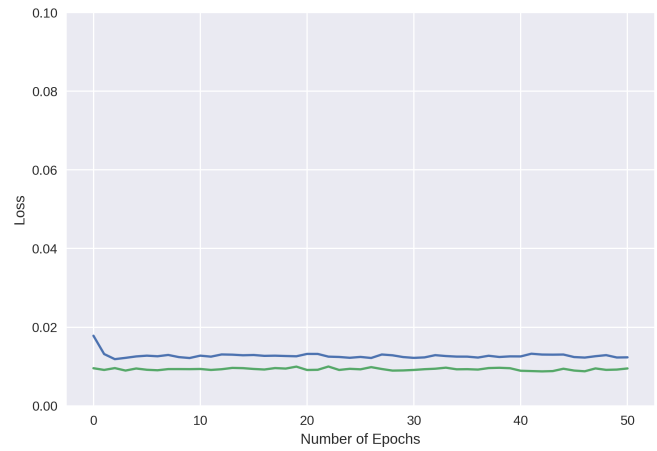

Figure 5.2: Training and Validation Loss Curves on GCN-2: edge-KNN

baseline reported here in the experiment would be further improved on in the course of the project.

5.5 GCN: Most Representative Dataset

5.5.1 Objective: GCN-2 on edge-radius, edge-KNN, edge-label

Our objective is to find the most representative graph dataset by training the model on edge-radius, edge-KNN and edge-label datasets.

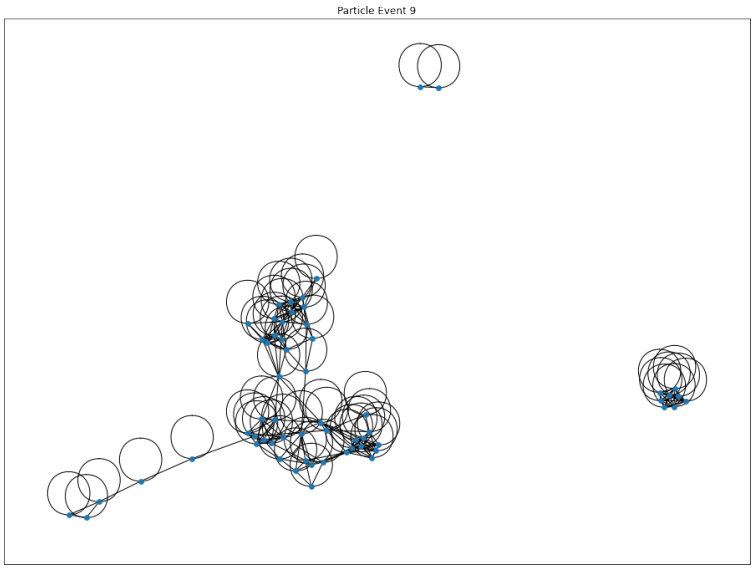

Figure 5.3: Networkx Visualisation of a sample event: edge-KNN

5.5.2 Results:

The event accuracy of GCN-2 on edge-radius was 13.0618 per cent, edge-KNN was 90.7206 per cent, and edge-label was 0 per cent. Therefore, we observe that the edge-KNN is the most representative graph dataset. The results of the GCN model can be directly correlated with the visualization of each type of graph and their corresponding node and graph metrics. The edge-KNN model's visualization is more distinct and helps the classifier discriminate the particles effectively. Further, the local properties of a sample event and the global properties of edge-KNN events across multiple events also well distributed. We keep in mind the most representative dataset: edge-KNN to perform our hyperparameter experiments on the GCN model. Additionally, since the number of edges are the speed bottleneck, as discussed in Chapter 3, edge-KNN is also the most efficient with minimum number of edges as shown in Chapter 4.

5.6 GCN: Network Depth Experiment

5.6.1 Objective: Network Depth of 2, 4, 8, 16

We aim to find the optimal network depth for the GCN model on particle classification. We perform this experiment to investigate the performance across different

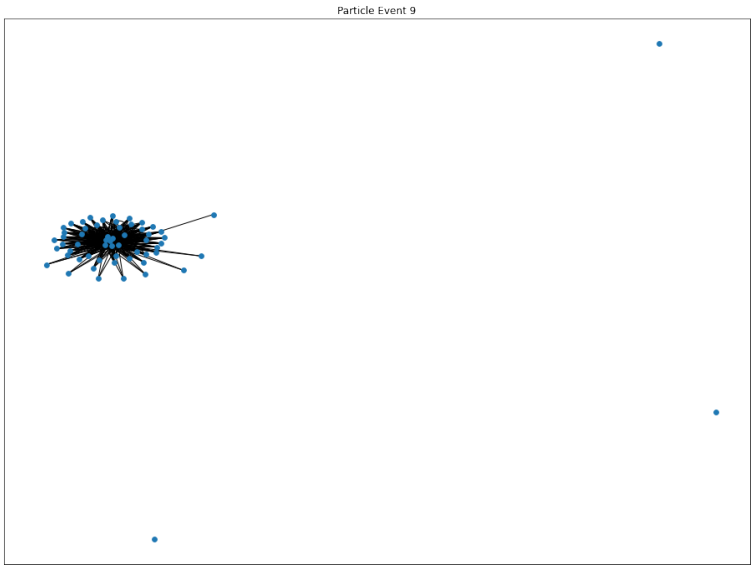

Figure 5.4: Networkx Visualisation of a sample event: edge-radius

KNN - Node Metrics on Event ID = 9
Node 22 has the largest degree value 15.
Node 49 has the largest betweenness centrality value 0.21056706050397578.
Node 50 has the largest closeness centrality value 0.28065458796025716.
Node 22 has the largest degree centrality value 0.2542372881355932.
Node 10 has the largest clustering value 1.0.
Node 34 has the largest average neighbor degree value 9.6.

Table 5.2: Maximum Value of Node Metrics of edge-KNN

network depths and make an informed decision about selecting an appropriate network depth for future GNN architectures explored in the course of the project. The most optimal depth is evaluated based on the event accuracy performance metric.

5.6.2 Results:

As per research [104], shallow networks outperform deep graph networks, which is empirically observed here in our experiment. It highlights the problem of over smoothing: the model has aggregated across all the nodes in the graph, causing the learned node embeddings for each node to be similar to all other nodes in the graph as discussed in Chapter 3. Over-smoothing decreases a GNN's discriminating ability, as observed in the results.

The dilemma of using deep architectures on graphs, where depth refers to the

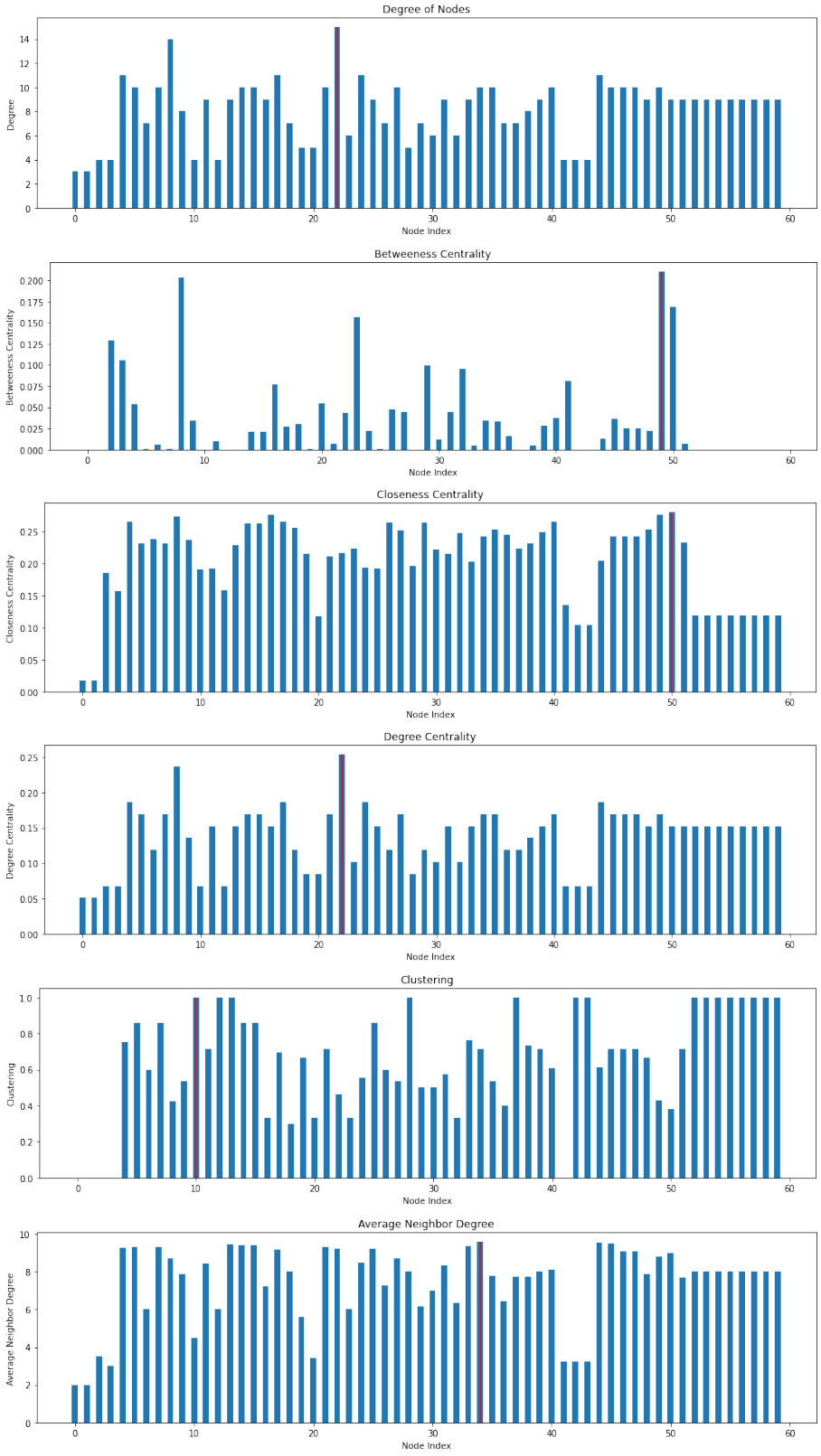

Figure 5.5: Node Metrics for a sample event: edge-KNN

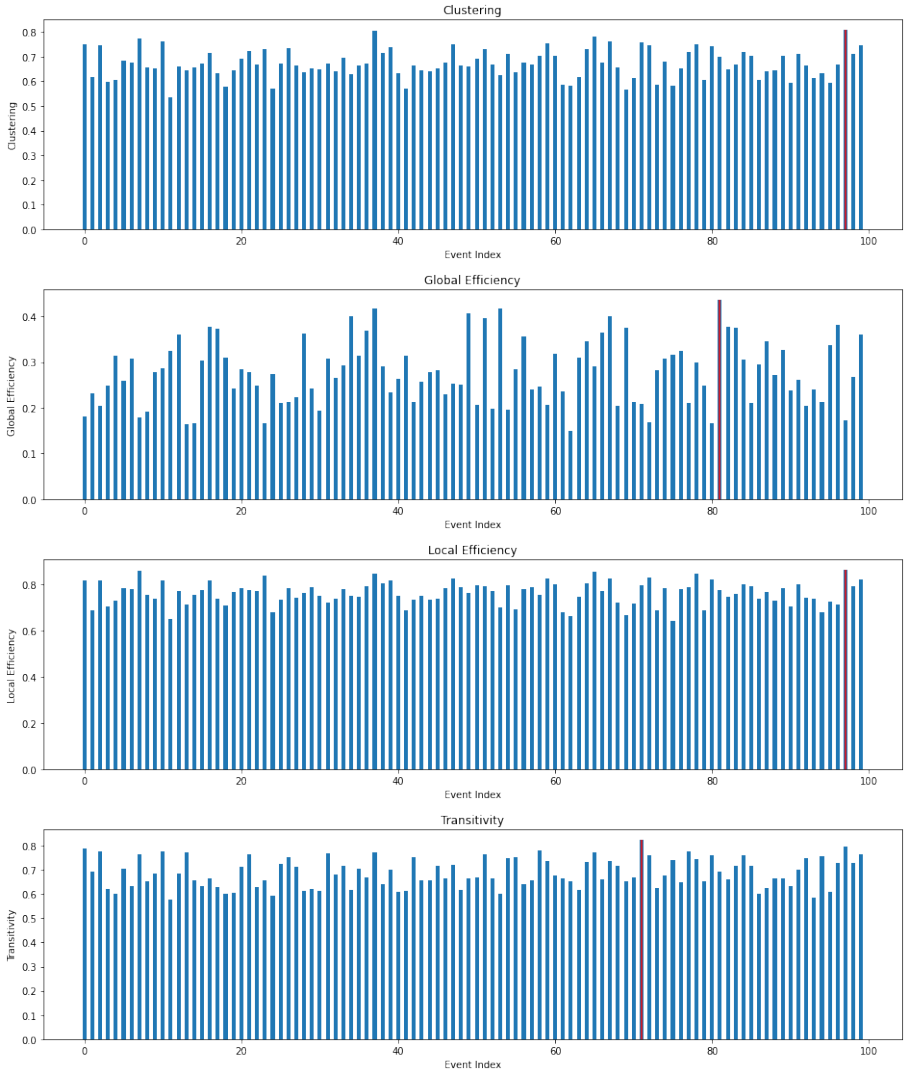

Figure 5.6: Graph Global Metrics for a sample of 100 event:edge-KNN

number of stacked graph convolutional layers, is crucial. Due to vanishing gradients and feature smoothing, deep GNNs are notoriously difficult to train [105, 106, 107]. Although recent work depicts that these issues can be addressed to some extent [108, 109, 110, 111, 112], comprehensive experiments conducted in [112] demonstrate that depth often does not bring any major performance gain over shallow baselines. Proceeding with shallow GNNs is particularly important in all scenarios where scalability is a primary concern, such as large-scale industrial systems at CERN.

We experimented with network depth in the GCN model with 2, 4, 8 and 16 layers to observe changes in event accuracy. We observed that shallow graph networks outperform deep graph networks. The event accuracy on the KNN dataset

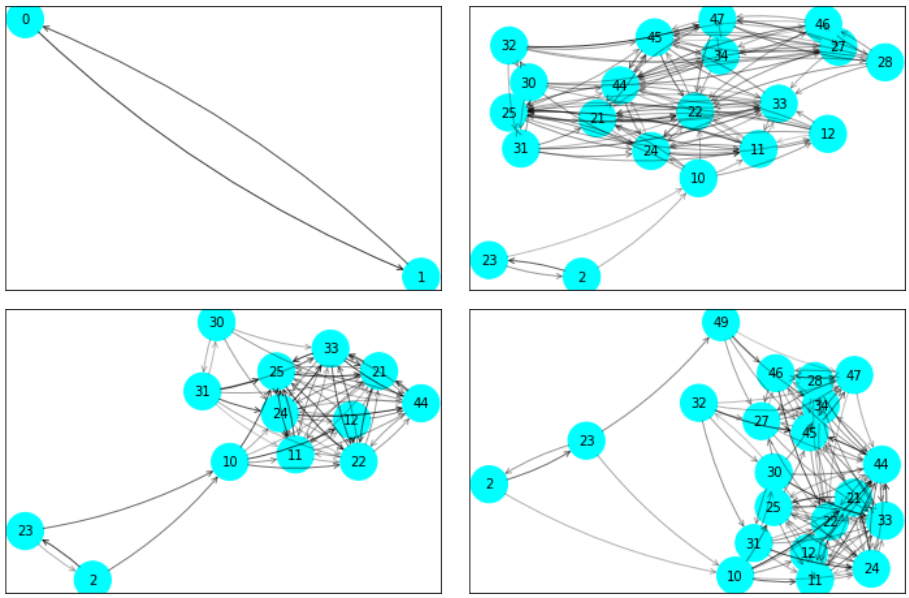

Figure 5.7: Visualisation of Subgraph to Explain GCN-2 predictions for node at indices(L-R): 0, 11, 12, 21 using [91]: edge-KNN

Model	Depth	Dataset	Event Accuracy	Node Accuracy
GCN	2	KNN	90.7206	99.8618
GCN	2	Radius	13.0618	97.0824
GCN	2	Label	0	95.8934
GCN	4	KNN	84.6716	99.7607
GCN	4	Radius	0.0059	95.8934
GCN	4	Label	0	95.8934
GCN	8	KNN	46.9039	99.676
GCN	8	Radius	0	95.8934
GCN	8	Label	0	95.8934
GCN	16	KNN	0.0049	95.8934
GCN	16	Radius	0	95.8934
GCN	16	Label	0	95.8934

Table 5.3: GCN: Network Depth Experiment

for GCN 2 is 90.7206 per cent, GCN-4 is 84.6716 per cent, GCN-8 is 46.9039 per cent, and GCN-16 is 0.00049 per cent. Consequently, we design the next experiments with network depths of two and four as they outperform other depth values.

Model	Dataset Size	Dataset	Event Accuracy	Node Accuracy
GCN	20% of dataset	KNN	89.7353	99.8079
GCN	20% of dataset	Radius	9.4706	96.9705
GCN	20% of dataset	Label	0	95.8092
GCN	40% of the dataset	KNN	91.8701	99.6715
GCN	40% of the dataset	Radius	10.4559	97.0927
GCN	40% of the dataset	Label	0	95.742
GCN	60% of the dataset	KNN	89.6912	99.7916
GCN	60% of the dataset	Radius	13.1781	97.0374
GCN	60% of the dataset	Label	0	95.8441
GCN	80% of the dataset	KNN	90.3039	99.7483
GCN	80% of the dataset	Radius	12.5784	97.1147
GCN	80% of the dataset	Label	0	95.8886

Table 5.4: GCN: Accuracy with Different Dataset Sizes

5.7 GCN: Accuracy with Different Dataset Sizes

5.7.1 Objective: Dataset split into 0.2, 0.4, 0.6, 0.8

Our objective is to observe the changes in performance metrics based on the size of the dataset. The experiment was conducted to assess the next direction of our research and help us identify if we need to train the model on more data or to execute experiments with different GNN architectures.

5.7.2 Results:

We observe minor changes in event accuracy with respect to the dataset size. We trained the GCN-2 model on 20 per cent, 40 per cent, 60 per cent, 80 per cent of the dataset to observe the changes in the GCN-2 performance with increasing dataset sizes. We observe that the event accuracy fluctuates with minor changes across the four datasets. Event accuracy in increasing order of dataset sizes is reported as 89.7353 per cent, 91.870 per cent, 89.6912 per cent and 90.3039 per cent. As the results indicate, exploring the diverse types of models in the current graph neural network literature and reporting their results first would prove to be more promising.

5.8 GCN: Hyperparameter Search

5.8.1 Objective: Find optimal hyperparameters for GCN-2 on edge-KNN

We examine individual contributions of our model design choices across various hyperparameters of GCN. We aim to identify any significant increase or decrease in event accuracy, but we limited our exploration on the edge-KNN as it was previously observed to provide the most representative dataset.

5.8.2 Results:

On the GCN-2 network, we experimented with different dropout values(0.1, 0.2, 0.3, 0.4,0.5) different number of hidden channels(10,16,32,64,128) and different learning rates (0.1, 0.01, 0.001, 0.0001, 0.00001), different weight decay (0.5, 0.05, 0.005, 0.0005, 0.00005) and different optimisers(adam, sgd, rmsprop). The optimal parameters based on the event accuracy performance metric were selu as non-linearity, 16 hidden channels, 0.0001 as learning rate, 0.00005 as weight decay, and dropout value of 0.2. Adam and SGD optimizers were of comparable performance. An interesting observation was made when GCN-2 with a weight decay of 0.00005 led to a sharp increase in event accuracy from 90.7206 percent to 94.9716 percent.

5.9 GNN-variants: Model Evaluation and Comparison

5.9.1 Objective: MLP, GCN, ChebNet, SAGE, GAT, GIN, JKNet, TAGCN and SuperGAT

We train and test all the graph neural network models on three datasets: edge-radius, edge-KNN and edge-label to observe changes in performance against the NN baseline and GNN baseline, to compare each model variant and to assess if they follow the same pattern observed for GCN experiments.

Model	Hyperparamters (Note)	Event Accuracy	Node Accuracy
GCN	relu	90.7206	99.8618
GCN	elu	90.0892	99.8177
GCN	selu	91.0265	99.861
GCN	gelu	88.9549	99.7944
GCN	leaky_relu	90.6088	99.8405
GCN	tanh	90.4912	99.8209
Model	**Hyperparamters (Note)**	**Event Accuracy**	**Node Accuracy**
GCN	hidden channels = 10	88.9971	99.875
GCN	hidden channels = 16	90.7206	99.8618
GCN	hidden channels = 32	90.4853	99.7893
GCN	hidden channels = 64	90.4951	99.7854
GCN	hidden channels = 128	90.3627	99.8686
Model	**Hyperparamters (Note)**	**Event Accuracy**	**Node Accuracy**
GCN	dropout=0.1	91.0461	99.7852
GCN	dropout=0.2	91.052	99.7987
GCN	dropout=0.3	90.9598	99.8032
GCN	dropout=0.4	90.951	99.8467
GCN	dropout=0.5	90.7206	99.8618
Model	**Hyperparamters (Note)**	**Event Accuracy**	**Node Accuracy**
GCN	learning rate = 0.1	54.4539	99.9149
GCN	learning rate = 0.01	90.7206	99.8618
GCN	learning rate = 0.001	91.3824	99.7973
GCN	learning rate = 0.0001	91.5716	99.7992
GCN	learning rate = 0.00001	91.2735	99.8064
Model	**Hyperparamters (Note)**	**Event Accuracy**	**Node Accuracy**
GCN	weight decay = 0.1	0	95.8934
GCN	weight decay= 0.01	0	95.8934
GCN	weight decay = 0.001	89.3147	99.8096
GCN	weight decay = 0.0001	90.7206	99.8618
GCN	weight decay = 0.00001	94.9716	99.9235
Model	**Hyperparamters (Note)**	**Event Accuracy**	**Node Accuracy**
GCN	optimiser = adam	90.7206	99.8618
GCN	optimiser = sgd	90.7245	99.8097
GCN	optimiser = rmsprop	90.248	99.8333

Table 5.5: Model Performance for Different Hyperparameters in GCN-2 on edge-KNN

Model	Depth	Dataset	Event Accuracy	Node Accuracy
MLP	2	KNN	87.2637	99.9538
GCN	2	KNN	90.7206	99.8618
Cheb	2	KNN	95.4245	99.9591
SAGE	2	KNN	95.0667	99.9518
TAGCN	4	Radius	91.3637	99.9611
GAT	2	KNN	95.8284	99.9414
GIN	2	KNN	88.1157	99.7083
JK	2	KNN	86.3539	99.9302
SuperGAT	4	KNN	96.4912	99.9494

Table 5.6: GNN-variants: Model Evaluation and Comparison

5.9.2 Results:

- To accomplish the supervised node classification task on diverse networks, we implemented various architectures: MLP, GCN, ChebNet, SAGE, GAT, SuperGAT, GIN, JK and TAGCN on three datasets: edge-KNN, edge-radius, and edge-label. The models were built by stacking graph convolutional layers, implemented with network depths of 2 and 4 due to the previous results of the network depth experiment favouring shallow networks. Default parameters were applied unless otherwise specified. Since node classification accuracies of most of the models were on the same scale of approximately 99 percent, the event accuracy gives a better insight into the classifier's performance and is a suitable metric for evaluating the jet clustering capabilities of the model. A detailed comparison of the best performing model of each category is illustrated with event accuracies reported on the test dataset.

- All models outperformed their counterparts on the edge-KNN dataset except for TAGCN-4. It performed better on the edge-radius dataset with an event accuracy of 91.3637 per cent compared to 86.2598 per cent on the edge-KNN dataset.

- The majority of the models performed better with a network depth of 2 than 4. However, TAGCN-4 and SuperGAT-4 performed better than their two-layer counterparts.

Model	Depth	Hyperparamters	Event Accuracy
GAT	2	heads=1	95.8284
GAT	2	heads=2	20.7843
GAT	2	heads=4	24.1284
GAT	2	heads=6	24.8716
GAT	2	heads=8	25.4216
GAT	2	heads=10	25.4216

Table 5.7: Model Performance of GAT

Model	Depth	Hyperparameters	Event Accuracy
superGAT	4	heads = 1, attn=MX	96.4912
superGAT	2	heads = 1, attn=MX	95.9304
superGAT	2	heads = 8, attn=MX, edge ratio = 0.8	5.8735
superGAT	2	heads =1, attn=SD	3.9784
superGAT	2	heads= 8, attn=SD, edge ratio = 0.8	5.4098

Table 5.8: Model Performance of superGAT

- We also perform experiments with different hyperparameters for three GNNs: ChebNet, GAT and superGAT to observe any significant changes in the performance metrics. We noted that the attention mechanisms in GAT and superGAT models decreased their event accuracy to a great extent when multiple heads were used.

- The best performing model is superGAT-4, with 96.4912 per cent event accuracy on the edge-KNN dataset. We observe that SuperGAT(MX) outperforms SuperGAT(SD), as it's better able to capture the average degree and homophily of the edge-KNN dataset.

- Though most models perform impressively well on edge-KNN compared to other generated graph datasets, ChebNet's performance on the edge-label resulted in an event accuracy of 99.8265 percent. The strictly localized filters implemented in ChebConv layers allows the ChebNet model to learn the node embeddings more accurately on the edge-label dataset (outperforming superGAT). However, the edge-label is generated by connecting nodes from the same class and utilized only for comparison in this study. It is still a strong indication that a better graph representation could lead to significant

Model	Depth	Cheby-shev filter	Dataset	Event Accuracy
ChebNet	2	k = 2	KNN	0.954245
ChebNet	2	k = 3	KNN	0.923353
ChebNet	2	k = 4	KNN	0.917618
ChebNet	2	k = 2	Label	0.998265

Table 5.9: Model Performance of ChebNet

performance gains. Increasing the Chebyshev filter size k from k=2 to k=3, 4 in ChebNet decreases the performance. Our experiments on varying k and with network depths 2 and 4 are also summarised in the table.

5.10 Conclusion

The results highlight significant improvements against the established baseline. We have answered all the research questions enlisted in Chapter 2.

5.10.1 Could machine learning techniques be used to predict the sets of final state particles originating from the Higgs Boson and Z boson in the HZ production mode at the future lepton collider?

Yes, machine learning techniques can be used to classify particles based on its origin of decay, as evidenced by the results of the MLP experiment, which only used features: process ID, vertex 4-vector and momentum 4-vector.

5.10.2 How can graph neural networks be used to model event collisions as graphs and particles as nodes for the supervised task of node classification?

Each event collision is treated as a graph of variable size, with a different number of particles in an event decay chain. Labels are generated using the information regarding the sequence of particle decay in the 'D1' and 'D2' features from the simulation dataset. Stable particles are filtered based on each particle's 'status' feature and each of the stable particles is treated as a node. After the dataset of final state particles is generated and processed, we build a graph dataset through various

edge generation techniques between pairs of nodes, process each graph as a batch in the training process and make predictions on unseen graphs. The GCN experiments support the statements in [8] that graphs are better representations of event collisions when compared to datasets built solely on the particle(node) features for each event.

5.10.3 How can we generate relationships or links between final state particles? What is the most optimal type of graph data representation of the stable particles detected at a collider?

We define the graph for the electron-positron collision by generating edges on the final state(stable) particles observed using three techniques: edge-KNN, edge-Radius, and edge-label. We further analyze the results and performance of the graph neural networks across the three datasets to find edge-KNN is the most optimal type of graph representation.

5.10.4 What is the most promising graph neural architecture to correctly predict all the nodes' labels in a graph?

We find that event accuracy is a better performance metric than the mean node classification accuracy. The baseline MLP-2 event accuracy is 87.2637 per cent, and GCN-2 is 90.7206 per cent on the edge-KNN dataset. The most promising model is the four-layer GNN architecture, superGAT-4, with 96.4912 per cent event accuracy on the edge-KNN dataset.

In this chapter, we successively explored and reported the design choices and performance metrics for the graph neural networks used to classify the final state particle of an electron-positron collision for the Future Circular Collider (FCC-ee).

6 Conclusions

Contents

6.1 Overview 87
6.2 Limitations of Our Work 88
6.3 Key Contributions and Results 89
6.4 Future work and Directions 89
6.4.1 Higgs Production Modes 89
6.4.2 Data representation 90
6.4.3 Lepton Collider Datasets 91
6.4.4 Extending Model Designs 91
6.4.5 Exhaustive evaluation on Random Seeds 91
6.4.6 Fast Inference 91
6.4.7 Larger Datasets 92
6.5 Conclusion 92

6.1 Overview

The project uncovers important results and conclusions. Our work is the first research project applying graph neural networks for clustering jets at a future FCC-ee experiment. The focus of the present dissertation is to examine a set of methods to generate graph representations of particles of an event and evaluate a set of neural network models on each of these graph representations.

Our approach is designed to compare three diverse representations of the dataset

and nine different neural network models. The project pipeline is implemented in a reproducible fashion and sub-divided into experiments. Each pipeline, driven by physics-related requirements, is programmed to collect, produce, process, analyse, and visualise the dataset, graphs, models, predictions and performance metrics.

6.2 Limitations of Our Work

- We proceeded our experiment with the assumption that once the model predicts the class of the particle to a sufficient degree of accuracy, we will be able to evaluate its performance on the experimental dataset. A Monte Carlo simulation generates the dataset used in this work. The experimental dataset is unavailable to us as the FCC experiment is not in operation as of 2021. Therefore our work is limited to evaluating algorithms only on the MC data. However, as reported in [8], much of the recent work in particle physics, on graph neural nets is carried out on simulation datasets.

- The in-depth analysis presented in this dissertation only contains tools, architectures and predictions on the stable(final-state) particles expected to be observed at the detector. The interest of this restriction is to offer meaningful points of comparison with the selected literature and make sure the model can be used both on the experimental dataset and the simulation dataset.

- The main production mode concerned with this study is the Higgs Strahlung process, or the HZ mode, where the electron and positron collision decays to HZ, where H is Higgs Boson, and Z is Z boson[30]. We have limited our dataset and study to such events for the thesis. Further, our in-depth analysis of our work is limited to ten thousand event collisions only, due to resource constraints. Given more time, we would like to train the model over larger number of iterations and track the evolution of performance metrics over a million events.

6.3 Key Contributions and Results

Our work has been shown to successfully and reliably achieve promising results. Our analyses uncovered the following major observations:

- The majority of the GNN models perform better on the edge-KNN dataset. We generated three types of datasets (edge-KNN, edge-Radius, edge-Label); the edge-knn dataset is the most suitable representation. Almost all models perform well on it in comparison to edge-radius and edge-label. Further, in terms of graph processing, the model performs better on variable-sized graphs in comparison to the fixed-size graphs.

- Baseline: MLP-2 event accuracy is 87.2637 percent and GCN-2 is 90.7206 percent on the edge-KNN dataset. As can be observed, the graph convolutional network outperformed neural networks. After extensive investigations, we uncovered network depths of 2 and 4 are the most suitable for the specified task of supervised node classification.

- The best performing model is built by stacking 4 layers of superGAT's convolutional operator. SuperGAT-4 outperforms other GNN networks and reports an event accuracy of 96.4912 percent on the edge-KNN dataset. We implement and optimise various GNN architectures: GAT, SAGE, GIN, JKNet, ChebNet, TAGCN, superGAT, and see improvement across all models against MLP baseline on the event accuracy metrics.

- Therefore, we have successfully developed a graph neural network that can effectively discriminate among final state(stable) particles originating from the H boson, Z boson or other bosons using spatial and physical features.

6.4 Future work and Directions

6.4.1 Higgs Production Modes

Our research clearly illustrates the importance of graph neural networks for clustering jets in aan e-e+ collision, but it also raises the question of whether these also are

equally suitable for other Higgs Boson production modes. Therefore, based on these conclusions, practitioners should consider future work towards developing a model for classifying the final state particles of the electron-positron collision events, where the Higgs Boson is produced via the vector boson fusion [30].

6.4.2 Data representation

Data representation is a key contributing element towards the model performance.

- Community detection algorithms: We applied the popular community detection algorithm: Louvain algorithm on a set of collision events and found that it could distinguish H and Z events into different clusters. One could try linking the nodes in a community as a fully connected graph within a single event graph and report its performance.

- Physics-informed radius: The edge-radius method to generate edges could further be optimised by using physics-informed distance metrics as discussed by authors in [9] and [22]. The radius used to generate the graph could be further enhanced by calculating the geometric or angular distance in the pseudo-rapidity and azimuth plane $\Delta R \equiv \sqrt{\delta\phi^2 + \delta\eta^2} < 0.3$ [18]. As per [8], 'Edges can be formed based on a relevant metric such as the ΔR between particles in a detector, or the physical distance between detector modules.'

- Random Edges: The edge formation is essentially random in the early phases of training, enabling the network to explore whether node representations should be closer together in the newest space. Injecting random edges in addition to linking nodes to k-nearest neighbours in recent space, by connecting a small number of random connections to nodes further away in latent space could be explored[8].

- Using an RL agent to generate edges: A reinforcement learning agent is introduced in a recent article [26] that explores an input graph to reach nodes that should be linked by new edges. Its policy is optimized to increase the

performance of some downstream tasks, so the nodes it connects to new edges boost the task's performance.

- Reverse engineering: We noticed that ChebNet performed with an event accuracy of 99.8265 per cent on the edge-label dataset. One could use graph similarity metrics or graph matching(similarity) algorithms to build a similarly rich representation.

6.4.3 Lepton Collider Datasets

To better understand the implications of these results, future studies could extend upon our work build and evaluate models on simulation datasets from other future electron-positron colliders as discussed in Chapter 2.

6.4.4 Extending Model Designs

Due to time constraints, we could only optimise GCN, Chebnet, GAT, and superGAT. Further exploration and optimisations of the best performing models should be the next step. Additionally, we stacked identical GN blocks to build our models[125]. Based on the results of our experiments, stacking of different types GN blocks could be considered and explored.

6.4.5 Exhaustive evaluation on Random Seeds

The probabilistic nature of the findings is another significant factor that needs a thorough examination. The model should be tested and performed multiple times with different random seeds, focusing on the efficiency of each approach to get better statistically reliable estimates of its performance.

6.4.6 Fast Inference

The ultimate goal is to help the FCC-ee experiment deploy the trained model and use it for fast inference. In this case, transfer of knowledge, conversion of data into suitable formats, and code from CPU to FPGAs would become necessary.[23, 24]

6.4.7 Larger Datasets

The model should be evaluated with a larger sample of event collisions to eliminate statistical fluctuations in the results.

6.5 Conclusion

The present dissertation focuses on the task of the supervised node classification over multiple graphs. More generally, discriminating between particles (represented as nodes) generated from different ancestors(origin of decay) for each event collision(represented as a graph). Consequently, the results of our work could be useful for jet tagging and jet energy calibration[31]. It paves the path for an interesting set of future experiments and testing our methodology to provide the FCC collaboration with novel and useful tools and capabilities.

In conclusion, our strategy to model spatial and physical information into a graph structure has shown significant results and offers promising avenues of growth in the experimental particle physics context. Advancements in graph representational learning and node classification tasks would also provide particle physicists with a more specialised toolbox of methodologies, helping to make the next significant breakthroughs unveiling the fundamental laws and elements of our Universe.

Bibliography

[1] M. K. Gaillard, P. D. Grannis, and F. J. Sciulli, "The standard model of particle physics," Reviews of Modern Physics, vol. 71, no. 2, pp. S96–S111, Mar. 1999, doi: 10.1103/revmodphys.71.s96.

[2] CERN News, "CERN Data Centre passes the 200-petabyte milestone," CERN, 2017. https://home.cern/news/news/computing/cern-data-centre-passes-200-petabyte-milestone.

[3] CERN, Processing: What to record? CERN Document Server, 2012.

[4] P. W. Higgs, "Broken Symmetries and the Masses of Gauge Bosons," Physical Review Letters, vol. 13, no. 16, pp. 508–509, Oct. 1964, doi: 10.1103/physrevlett.13.508.

[5] F. Englert and R. Brout, "Broken Symmetry and the Mass of Gauge Vector Mesons," Physical Review Letters, vol. 13, no. 9, pp. 321–323, Aug. 1964, doi: 10.1103/physrevlett.13.321.

[6] The ATLAS Collaboration, "Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC," 2012.

[7] S. D. Bass, A. De Roeck, and M. Kado, "The Higgs boson implications and prospects for future discoveries," Nature Reviews Physics, vol. 3, no. 9, pp. 608–624, Jul. 2021, doi: 10.1038/s42254-021-00341-2.

[8] J. Shlomi, P. Battaglia, and J.-R. Vlimant, "Graph neural networks in particle physics," Machine Learning: Science and Technology, vol. 2, no. 2, p. 021001, Jan. 2021, doi: 10.1088/2632-2153/abbf9a.

[9] D. Xu and Y. Tian, "A Comprehensive Survey of Clustering Algorithms," Annals of Data Science, vol. 2, no. 2, pp. 165–193, Jun. 2015, doi: 10.1007/s40745-015-0040-1.

[10] F. Gianotti and G. F. Giudice, "A roadmap for the future," Nature Physics, vol. 16, no. 10, pp. 997–998, Sep. 2020, doi: 10.1038/s41567-020-01054-6.

[11] European Strategy Group, 2020 Update of the European Strategy for Particle Physics. Geneva: CERN Council, 2020.

[12] M. Benedikt and F. Zimmermann, "The physics and technology of the Future Circular Collider," Nature Reviews Physics, vol. 1, no. 4, pp. 238–240, Mar. 2019, doi: 10.1038/s42254-019-0048-0.

[13] M. Benedikt, A. Blondel, P. Janot, M. Mangano, and F. Zimmermann, "Future Circular Colliders succeeding the LHC," Nature Physics, vol. 16, no. 4, pp. 402–407, Apr. 2020, doi: 10.1038/s41567-020-0856-2.

[14] S. Stapnes, "The Compact Linear Collider," Nature Reviews Physics, vol. 1, no. 4, pp. 235–237, Mar. 2019, doi: 10.1038/s42254-019-0051-5.

[15] E. Sicking and R. Ström, "From precision physics to the energy frontier with the Compact Linear Collider," Nature Physics, vol. 16, no. 4, pp. 386–392, Apr. 2020, doi: 10.1038/s41567-020-0834-8.

[16] S. Michizono, "The International Linear Collider," Nature Reviews Physics, vol. 1, no. 4, pp. 244–245, Feb. 2019, doi: 10.1038/s42254-019-0044-4.

[17] X. Lou, "The Circular Electron Positron Collider," Nature Reviews Physics, vol. 1, no. 4, pp. 232–234, Mar. 2019, doi: 10.1038/s42254-019-0047-1.

[18] V. Mikuni and F. Canelli, "ABCNet: an attention-based method for particle tagging," The European Physical Journal Plus, vol. 135, no. 6, Jun. 2020, doi: 10.1140/epjp/s13360-020-00497-3.

[19] E. A. Moreno et al., "JEDI-net: a jet identification algorithm based on interaction networks," The European Physical Journal C, vol. 80, no. 1, Jan. 2020, doi: 10.1140/epjc/s10052-020-7608-4.

[20] CMS Collaboration., "Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC.," 2012.

[21] H. Qu and L. Gouskos, "Jet tagging via particle clouds," Physical Review D, vol. 101, no. 5, Mar. 2020, doi: 10.1103/physrevd.101.056019.

[22] J. Arjona Martínez, O. Cerri, M. Spiropulu, J. R. Vlimant, and M. Pierini, "Pileup mitigation at the Large Hadron Collider with graph neural networks," The European Physical Journal Plus, vol. 134, no. 7, Jul. 2019, doi: 10.1140/epjp/i2019-12710-3.

[23] D. Rousseau, "Resource-efficient inference for particle physics," Nature Machine Intelligence, vol. 3, no. 8, pp. 656–657, Aug. 2021, doi: 10.1038/s42256-021-00381-4.

[24] C. N. Coelho et al., "Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors," Nature Machine Intelligence, vol. 3, no. 8, pp. 675–686, Jun. 2021, doi: 10.1038/s42256-021-00356-5.

[25] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. W. Battaglia, "Learning to Simulate Complex Physics with Graph Networks," arXiv:2002.09405 [physics, stat], Sep. 2020.

[26] D. D. Johnson, H. Larochelle, and D. Tarlow, "Learning Graph Structure With A Finite-State Automaton Layer," arXiv:2007.04929 [cs, stat], Nov. 2020.

[27] Committee on Network Science for Future Army Applications, Network Science. National Research Council, 2006.

[28] M. M. Li, K. Huang, and M. Zitnik, "Representation Learning for Networks in Biology and Medicine: Advancements, Challenges, and Opportunities," arXiv:2104.04883 [cs, q-bio], Apr. 2021.

[29] J. Zhou et al., "Graph neural networks: A review of methods and applications," AI Open, vol. 1, pp. 57–81, 2020, doi: 10.1016/j.aiopen.2021.01.001.

[30] H. Gray and P. Janot, "Higgs Physics," Comptes Rendus. Physique, vol. 21, no. 1, pp. 23–43, 2020, doi: 10.5802/crphys.8.

[31] X. Ju and B. Nachman, "Supervised Jet Clustering with Graph Neural Networks for Lorentz Boosted Bosons," arxiv.org, Aug. 2020, doi: 10.1103/PhysRevD.102.075014.

[32] CERN, "The Standard Model at CERN," Home.cern, Apr. 08, 2019. https://home.cern/science/physics/standard-model.

[33] M. Cooke, "DOE Explains...the Higgs Boson," Energy.gov. https://www.energy.gov/science/doe-explainsthe-higgs-boson.

[34] O. M. Gomez, "Five mysteries the Standard Model can't explain," symmetry magazine, 2018. https://www.symmetrymagazine.org/article/five-mysteries-the-standard-model-cant-explain.

[35] J. Crawford, J. Standeven, and W. Kissel, "Accelerator Glossary of Terms," 2011.

[36] CERN, "Accelerators at CERN," Home.cern, Jul. 16, 2019. https://home.cern/science/accelerators.

[37] M. Benedikt et al., FCC-ee: The Lepton Collider: Future Circular Collider Conceptual Design Report Volume 2. CERN Document Server, 2018.

[38] R. Atkin, "Review of jet reconstruction algorithms," Journal of Physics: Conference Series, vol. 645, p. 012008, Oct. 2015, doi: 10.1088/1742-6596/645/1/012008.

[39] CERN, "Computing at CERN," home.cern. https://home.cern/science/computing.

[40] HEP Software Foundation, "A Roadmap for HEP Software and Computing RD for the 2020s HEP Software Foundation," 2017. [Online]. Available: https://arxiv.org/pdf/1712.06982.pdf.

[41] S. Sekmen, "Beyond the Standard Model Physics at the High Luminosity LHC," arXiv:1902.03942 [hep-ex, physics:hep-ph], Feb. 2019.

[42] P. C. Bhat, "Multivariate Analysis Methods in Particle Physics," Annual Review of Nuclear and Particle Science, vol. 61, no. 1, pp. 281–309, Nov. 2011, doi: 10.1146/annurev.nucl.012809.104427.

[43] L. Mou, G. Li, L. Zhang, T. Wang, and Z. Jin, "Convolutional Neural Networks over Tree Structures for Programming Language Processing," arXiv:1409.5718 [cs], Dec. 2015.

[44] Y. Shen, S. Tan, A. Sordoni, and A. Courville, "Ordered Neurons: Integrating Tree Structures into Recurrent Neural Networks," arXiv:1810.09536 [cs], May 2019.

[45] A. Radovic et al., "Machine learning at the energy and intensity frontiers of particle physics," Nature, vol. 560, no. 7716, pp. 41–48, Aug. 2018, doi: 10.1038/s41586-018-0361-2.

[46] S. R. Qasim, J. Kieseler, Y. Iiyama, and M. Pierini, "Learning representations of irregular particle-detector geometry with distance-weighted graph networks," The European Physical Journal C, vol. 79, no. 7, Jul. 2019, doi: 10.1140/epjc/s10052-019-7113-9.

[47] CMS Collaboration, "Particle-flow reconstruction and global event description with the CMS detector," Journal of Instrumentation, vol. 12, no. 10, pp. P10003–P10003, Oct. 2017, doi: 10.1088/1748-0221/12/10/P10003.

[48] D. Bertolini, P. Harris, M. Low, and N. Tran, "Pileup per particle identification," Journal of High Energy Physics, vol. 2014, no. 10, Oct. 2014, doi: 10.1007/jhep10(2014)059.

[49] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated Graph Sequence Neural Networks," arXiv:1511.05493 [cs, stat], Sep. 2017.

[50] J. Arjona Martínez, O. Cerri, M. Spiropulu, J. R. Vlimant, and M. Pierini, "Pileup mitigation at the Large Hadron Collider with graph neural networks," The European Physical Journal Plus, vol. 134, no. 7, Jul. 2019, doi: 10.1140/epjp/i2019-12710-3.

[51] M. Cacciari, G. P. Salam, and G. Soyez, "The anti-ktjet clustering algorithm," Journal of High Energy Physics, vol. 2008, no. 04, pp. 063–063, Apr. 2008, doi: 10.1088/1126-6708/2008/04/063.

[52] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, "Graph Attention Networks," arXiv:1710.10903 [cs, stat], Feb. 2018, [Online]. Available: https://arxiv.org/abs/1710.10903.

[53] D. Guest, K. Cranmer, and D. Whiteson, "Deep Learning and Its Application to LHC Physics," Annual Review of Nuclear and Particle Science, vol. 68, no. 1, pp. 161–181, Oct. 2018, doi: 10.1146/annurev-nucl-101917-021019.

[54] The ATLAS Collaboration, "Quark versus Gluon Jet Tagging Using Jet Images with the ATLAS Detector," CERN Document Server, 2017. https://cds.cern.ch/record/2275641/files/ATL-PHYS-PUB-2017-017.pdf.

[55] X. Ju and B. Nachman, "Supervised Jet Clustering with Graph Neural Networks for Lorentz Boosted Bosons," arxiv.org, Aug. 2020, doi: 10.1103/PhysRevD.102.075014.

[56] The ATLAS collaboration, Impact of Alternative Inputs and Jet Grooming on Large-R Jet Performance. CERN Document Server, 2019.

[57] ATLAS Collaboration, "Optimisation of large-radius jet reconstruction for the ATLAS detector in 13 TeV proton-proton collisions," arXiv:2009.04986 [hep-ex], May 2021, doi: 10.1140/epjc/s10052-021-09054-3.

[58] A. Chakraborty et al., "Revisiting Jet Clustering Algorithms for New Higgs Boson Searches in Hadronic Final States," arXiv:2008.02499 [hep-ph], Dec. 2020, [Online]. Available: https://arxiv.org/abs/2008.02499.

[59] A. Sperduti and A. Starita, "Supervised neural networks for the classification of structures," IEEE Transactions on Neural Networks, vol. 8, no. 3, pp. 714–735, May 1997, doi: 10.1109/72.572108.

[60] F. Scarselli, M. Gori, Ah Chung Tsoi, M. Hagenbuchner, and G. Monfardini, "The Graph Neural Network Model," IEEE Transactions on Neural Networks, vol. 20, no. 1, pp. 61–80, Jan. 2009, doi: 10.1109/tnn.2008.2005605.

[61] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, May 2015, doi: 10.1038/nature14539.

[62] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric Deep Learning: Going beyond Euclidean data," IEEE Signal Processing Magazine, vol. 34, no. 4, pp. 18–42, Jul. 2017, doi: 10.1109/msp.2017.2693418.

[63] H. Wang and J. Leskovec, "Unifying Graph Convolutional Neural Networks and Label Propagation," arXiv:2002.06755 [cs, stat], Feb. 2020.

[64] P. Han et al., "GCN-MF," Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining, Jul. 2019, doi: 10.1145/3292500.3330912.

[65] J. Ingraham, V. Garg, R. Barzilay, and T. Jaakkola, "Generative Models for Graph-Based Protein Design," Advances in Neural Information Processing Systems, vol. 32, 2019.

[66] Y. Xie, J. Peng, and Y. Zhou, "Integrating Protein-Protein Interaction Information into Drug Response Prediction by Graph Neural Encoding," 2019.

[67] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering," arXiv:1606.09375 [cs, stat], Feb. 2017, [Online]. Available: https://arxiv.org/abs/1606.09375.

[68] D. K. Duvenaud et al., "Convolutional Networks on Graphs for Learning Molecular Fingerprints," Advances in Neural Information Processing Systems, vol. 28, 2015.

[69] P. Veli ˇckovi c, "Theoretical Approaches to Graph Neural Networks," University of Cambridge, Zoom Seminar, 2021.

[70] A. Vaswani et al., "Attention is All you Need," Advances in Neural Information Processing Systems, vol. 30, pp. 5998–6008, 2017.

[71] E. Choi et al., "Learning the Graphical Structure of Electronic Health Records with Graph Convolutional Transformer," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 01, pp. 606–613, Apr. 2020, doi: 10.1609/aaai.v34i01.5400.

[72] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous Graph Transformer," Proceedings of The Web Conference 2020, Apr. 2020, doi: 10.1145/3366423.3380027.

[73] S. Yan, Y. Xiong, and D. Lin, "Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition," arXiv:1801.07455 [cs], Jan. 2018.

[74] S. Yun, M. Jeong, R. Kim, J. Kang, and H. J. Kim, "Graph Transformer Networks," Advances in Neural Information Processing Systems, vol. 32, 2019.

[75] K. Xu, C. Li, Y. Tian, T. Sonobe, K. Kawarabayashi, and S. Jegelka, "Representation Learning on Graphs with Jumping Knowledge Networks," arXiv:1806.03536 [cs, stat], Jun. 2018.

[76] S. Abu-El-Haija et al., "MixHop: Higher-Order Graph Convolutional Architectures via Sparsified Neighborhood Mixing," proceedings.mlr.press, May 24, 2019. http://proceedings.mlr.press/v97/abu-el-haija19a.html

[77] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković, "Geometric Deep Learning: Grids, Groups, Graphs, Geodesics, and Gauges," arXiv:2104.13478 [cs, stat], May 2021, [Online]. Available: https://arxiv.org/abs/2104.13478.

[78] R. Ying, J. You, C. Morris, X. Ren, W. L. Hamilton, and J. Leskovec, "Hierarchical Graph Representation Learning with Differentiable Pooling," arXiv:1806.08804 [cs, stat], Feb. 2019, [Online]. Available: https://arxiv.org/abs/1806.08804.

[79] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How Powerful are Graph Neural Networks?," arXiv:1810.00826 [cs, stat], Feb. 2019, [Online]. Available: https://arxiv.org/abs/1810.00826.

[80] H. Wang and J. Leskovec, "Unifying Graph Convolutional Neural Networks and Label Propagation," arXiv:2002.06755 [cs, stat], Feb. 2020, [Online]. Available: https://arxiv.org/abs/2002.06755.

[81] F. B. Fuchs, D. E. Worrall, V. Fischer, and M. Welling, "SE(3)-Transformers: 3D Roto-Translation Equivariant Attention Networks," arXiv:2006.10503 [cs, stat], Nov. 2020.

[82] Z. Hu, Y. Dong, K. Wang, and Y. Sun, "Heterogeneous Graph Transformer," Proceedings of The Web Conference 2020, Apr. 2020, doi: 10.1145/3366423.3380027.

[83] X. Wang et al., "Heterogeneous Graph Attention Network," The World Wide Web Conference, May 2019, doi: 10.1145/3308558.3313562.

[84] A. Pareja et al., "EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, no. 04, pp. 5363–5370, Apr. 2020, doi: 10.1609/aaai.v34i04.5984.

[85] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein, "Temporal Graph Networks for Deep Learning on Dynamic Graphs," arXiv:2006.10637 [cs, stat], Oct. 2020.

[86] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, "Cluster-GCN: An Efficient Algorithm for Training Deep and Large Graph Convolutional Networks," Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining, pp. 257–266, Jul. 2019, doi: 10.1145/3292500.3330925.

[87] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "GraphSAINT: Graph Sampling Based Inductive Learning Method," arXiv:1907.04931 [cs, stat], Feb. 2020.

[88] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling Relational Data with Graph Convolutional Networks," arXiv:1703.06103 [cs, stat], Oct. 2017.

[89] W. Hu et al., "Strategies for Pre-training Graph Neural Networks," arXiv:1905.12265 [cs, stat], Feb. 2020.

[90] Y. You, T. Chen, Z. Wang, and Y. Shen, "When Does Self-Supervision Help Graph Convolutional Networks?," arXiv:2006.09136 [cs, stat], Jul. 2020.

[91] R. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, "GNNExplainer: Generating Explanations for Graph Neural Networks," arXiv:1903.03894 [cs, stat], Nov. 2019.

[92] C. Agarwal, H. Lakkaraju, and M. Zitnik, "Towards a Unified Framework for Fair and Stable Graph Representation Learning," arXiv:2102.13186 [cs], Jun. 2021.

[93] D. Zügner, A. Akbarnejad, and S. Günnemann, "Adversarial Attacks on Neural Networks for Graph Data," Proceedings of the 24th ACM SIGKDD

International Conference on Knowledge Discovery Data Mining, Jul. 2018, doi: 10.1145/3219819.3220078.

[94] D. Zhu, Z. Zhang, P. Cui, and W. Zhu, "Robust Graph Convolutional Networks Against Adversarial Attacks," Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining, Jul. 2019, doi: 10.1145/3292500.3330851.

[95] A. Singh, S. Sengupta, and V. Lakshminarayanan, "Explainable Deep Learning Models in Medical Image Analysis," Journal of Imaging, vol. 6, no. 6, p. 52, Jun. 2020, doi: 10.3390/jimaging6060052.

[96] C. O'Neil, "Weapons of Math Destruction," www.penguin.co.uk, 2017.

[97] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," arXiv.org, 2018. https://arxiv.org/abs/1810.04805.

[98] Z. Hu, Y. Dong, K. Wang, K.-W. Chang, and Y. Sun, "GPT-GNN," Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining, Jul. 2020, doi: 10.1145/3394486.3403237.

[99] J. Zhang, H. Zhang, C. Xia, and L. Sun, "Graph-Bert: Only Attention is Needed for Learning Graph Representations," arXiv:2001.05140 [cs, stat], Jan. 2020.

[100] "Empirical analysis of performance bottlenecks in graph neural network training and inference with GPUs," Neurocomputing, vol. 446, pp. 165–191, Jul. 2021, doi: 10.1016/j.neucom.2021.03.015.

[101] C. Stamile , A. Marzullo, and E. Enrico Deusebio, "Graph Machine Learning," Packt, 2021. https://www.packtpub.com/product/graph-machine-learning/9781800204492.

[102] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," arXiv:1609.02907 [cs, stat], Feb. 2017, [Online]. Available: https://arxiv.org/abs/1609.02907.

[103] Y. Rong, W. Huang, T. Xu, and J. Huang, "DropEdge: Towards Deep Graph Convolutional Networks on Node Classification," arXiv:1907.10903 [cs, stat], Mar. 2020.

[104] F. Fabrizio , R. Emanuele , E. Davide , C. Ben , B. Michael , and M. Federico , "SIGN: Scalable Inception Graph Neural Networks," arXiv.org, Apr. 2020, [Online]. Available: https://arxiv.org/abs/2004.11198.

[105] R. Li, S. Sheng Wang, F. Zhu, and J. Huang, "Adaptive Graph Convolutional Neural Networks," https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16642, 2018.

[106] J. Klicpera, A. Bojchevski, and S. Günnemann, "Predict then Propagate: Graph Neural Networks meet Personalized PageRank," arXiv:1810.05997 [cs, stat], Feb. 2019.

[107] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu, "A Comprehensive Survey on Graph Neural Networks," IEEE Transactions on Neural Networks and Learning Systems, pp. 1–21, 2020, doi: 10.1109/TNNLS.2020.2978386.

[108] K. Xu, C. Li, Y. Tian, T. Sonobe, K. Kawarabayashi, and S. Jegelka, "Representation Learning on Graphs with Jumping Knowledge Networks," arXiv:1806.03536 [cs, stat], Jun. 2018, [Online]. Available: https://arxiv.org/abs/1806.03536.

[109] S. Gong, M. Bahri, M. M. Bronstein, and S. Zafeiriou, "Geometrically Principled Connections in Graph Neural Networks," arXiv:2004.02658 [cs], Apr. 2020.

[110] G. Li, M. Müller, A. Thabet, and B. Ghanem, "DeepGCNs: Can GCNs Go as Deep as CNNs?," arXiv:1904.03751 [cs], Aug. 2019, Accessed: Sep. 14, 2021. [Online]. Available: https://arxiv.org/abs/1904.03751.

[111] L. Zhao and L. Akoglu, "PairNorm: Tackling Oversmoothing in GNNs," openreview.net, Sep. 25, 2019.

[112] Y. Rong, W. Huang, T. Xu, and J. Huang, "DropEdge: Towards Deep Graph Convolutional Networks on Node Classification," arXiv:1907.10903 [cs, stat], Mar. 2020, [Online]. Available: https://arxiv.org/abs/1907.10903.

[113] D. Kim and A. Oh, "How to Find Your Friendly Neighborhood: Graph Attention Design with Self-Supervision," openreview.net, Sep. 28, 2020.

[114] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive Representation Learning on Large Graphs," arXiv:1706.02216 [cs, stat], Sep. 2018, [Online]. Available: https://arxiv.org/abs/1706.02216.

[115] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering," arXiv:1606.09375 [cs, stat], Feb. 2017, [Online]. Available: https://arxiv.org/abs/1606.09375.

[116] A. Kalweit, "Data analysis in particle physics," 2015.

[117] S. Otten et al., "Event Generation and Statistical Sampling for Physics with Deep Generative Models and a Density Information Buffer," arXiv:1901.00875 [hep-ex, physics:hep-ph, physics:physics], Feb. 2021.

[118] J. Allison et al., "Recent developments in Geant4," Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, vol. 835, pp. 186–225, Nov. 2016, doi: 10.1016/j.nima.2016.06.125.

[119] Delphes, "WorkBook – Delphes," cp3.irmp.ucl.ac.be, 2018.

[120] CERN, "ROOT - An Object Oriented Data Analysis Framework," 1997.

[121] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering," arXiv:1606.09375 [cs, stat], Feb. 2017, [Online]. Available: https://arxiv.org/abs/1606.09375.

[122] R. team, "ROOT: analyzing petabytes of data, scientifically.," ROOT, 2021. https://root.cern/.

[123] CERN, "Physics Vectors," Cern.ch, 2021. https://root.cern.ch/root/html534/guides/users-guide/PhysicsVectors.html.

[124] B. Sanchez-Lengeling, E. Reif, A. Pearce, and A. Wiltschko, "A Gentle Introduction to Graph Neural Networks," Distill, vol. 6, no. 8, Aug. 2021, doi: 10.23915/distill.00033.

[125] P. W. Battaglia et al., "Relational inductive biases, deep learning, and graph networks," arXiv:1806.01261 [cs, stat], Oct. 2018, [Online]. Available: https://arxiv.org/abs/1806.01261.

[126] A. Zhou et al., "Optimizing Memory Efficiency of Graph Neural Networks on Edge Computing Platforms," arXiv:2104.03058 [cs], Apr. 2021, [Online]. Available: https://arxiv.org/abs/2104.03058.

[127] S. Abadal, A. Jain, R. Guirado, J. López-Alonso, and E. Alarcón, "Computing Graph Neural Networks: A Survey from Algorithms to Accelerators," arXiv:2010.00130 [cs, stat], Jul. 2021, [Online]. Available: https://arxiv.org/abs/2010.00130.

[128] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural Message Passing for Quantum Chemistry," arXiv:1704.01212 [cs], Jun. 2017, [Online]. Available: https://arxiv.org/abs/1704.01212.

[129] C. Merkwirth and T. Lengauer, "Automatic Generation of Complementary Descriptors with Molecular Graph Networks," Journal of Chemical Information and Modeling, vol. 45, no. 5, pp. 1159–1168, Aug. 2005, doi: 10.1021/ci049613b.

[130] F. Scarselli, M. Gori, Ah Chung Tsoi, M. Hagenbuchner, and G. Monfardini, "The Graph Neural Network Model," IEEE Transactions on Neural Networks, vol. 20, no. 1, pp. 61–80, Jan. 2009, doi: 10.1109/tnn.2008.2005605.

[131] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. Salakhutdinov, and A. Smola, "Deep Sets," arXiv:1703.06114 [cs, stat], Apr. 2018, [Online]. Available: https://arxiv.org/abs/1703.06114.

[132] C. R. Qi, H. Su, K. Mo, and Guibas, Leonidas J, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," arXiv.org, 2016. https://arxiv.org/abs/1612.00593.

[133] D. Bahdanau, K. Cho, and Y. Bengio, "Neural Machine Translation by Jointly Learning to Align and Translate," arXiv.org, 2014. https://arxiv.org/abs/1409.0473.

[134] A. Vaswani et al., "Attention Is All You Need," arXiv.org, 2017. https://arxiv.org/abs/1706.03762.

[135] K. Xu, C. Li, Y. Tian, T. Sonobe, K. Kawarabayashi, and S. Jegelka, "Representation Learning on Graphs with Jumping Knowledge Networks," arXiv:1806.03536 [cs, stat], Jun. 2018, [Online]. Available: https://arxiv.org/abs/1806.03536.

[136] M. Fey, "PyG Documentation — pytorch geometric 2.0.0 documentation," pytorch-geometric.readthedocs.io, 2021.

[137] Wikipedia, "2019 redefinition of the SI base units," Wikipedia, Oct. 19, 2020.

[138] B. Carithers and P. Grannis, "TOP QUARK," SLAC, 1995. [Online]. Available: https://www.slac.stanford.edu/pubs/beamline/25/3/25-3-carithers.pdf.

[139] P. W. Higgs, "Broken Symmetries and the Masses of Gauge Bosons," Physical Review Letters, vol. 13, no. 16, pp. 508–509, Oct. 1964, doi: 10.1103/physrevlett.13.508.

[140] The ATLAS Collaboration, Quark versus Gluon Jet Tagging Using Jet Images with the ATLAS Detector. CERN Document Server, 2017.

[141] P. T. Komiske, E. M. Metodiev, and J. Thaler, "An operational definition of quark and gluon jets," Journal of High Energy Physics, vol. 2018, no. 11, p. 59, Nov. 2018, doi: 10.1007/JHEP11(2018)059.

[142] G. Kasieczka, N. Kiefer, T. Plehn, and J. M. Thompson, "Quark-Gluon Tagging: Machine Learning vs Detector," SciPost Physics, vol. 6, no. 6, p. 069, Jun. 2019, doi: 10.21468/SciPostPhys.6.6.069.

[143] J. Leskovec, "CS224W - Machine Learning with Graphs," web.stanford.edu, 2021. http://web.stanford.edu/class/cs224w/.

[144] J. Du, S. Zhang, G. Wu, J. M. F. Moura, and S. Kar, "Topology Adaptive Graph Convolutional Networks," arXiv:1710.10370 [cs, stat], Feb. 2018, [Online]. Available: https://arxiv.org/abs/1710.10370.

[145] CERN, "Future Circular Collider - Image selection," cds.cern.ch, 2019. https://cds.cern.ch/images/OPEN-PHO-ACCEL-2019-001-2.

[146] CMS Experiment (un-official), "Reference Frame," Quantumdiaries.org, 2016.

[147] The ATLAS Collaboration, "ATLAS observes elusive Higgs boson decay to a pair of bottom quarks," ATLAS, 2018. https://atlas.cern/updates/press-statement/observation-higgs-boson-decay-pair-bottom-quarks.

[148] W. L. Hamilton, Graph representation learning. San Rafael Morgan Et Claypool, 2020.