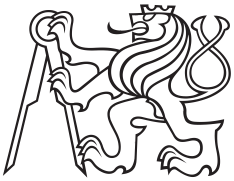
Master's Thesis



Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

A New Trackster Linking Algorithm Based on Graph Neural Networks for the CMS Experiment at the Large Hadron Collider at CERN

Bc. Jekatěrina Jaroslavceva

Supervisor: Prof. Mgr. Ondřej Chum, Ph.D.,
Department of Cybernetics, CTU
Supervisor–specialist: MSc. Felice Pantaleo, Ph.D.,
Experimental Physics Department, CMS, CERN
Study program: Cybernetics and Robotics
May 2023

MASTER‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Jaroslavceva Jekatěrina** Personal ID number: **474416**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

II. Master’s thesis details

Master’s thesis title in English:

A New Trackster Linking Algorithm Based on Graph Neural Networks for the CMS Experiment at the Large Hadron Collider at CERN

Master’s thesis title in Czech:

Nový algoritmus pro spojování 3D energetických depozitů založený na grafových neuronových sítích pro experiment CMS na velkém hadronovém urychlovači v CERNu

Guidelines:

The CERN High-Luminosity Large Hadron Collider (HL-LHC) project aspires to significantly increase the number of collisions in the LHC, thereby presenting a formidable challenge in terms of data volume and complexity for the current reconstruction algorithms.
The CMS Collaboration is devising a novel endcap calorimeter system, HGCAL, which will predominantly use silicon sensors to ensure adequate radiation tolerance and to preserve granular information in the readout in order to mitigate the effects of pile-up. Within the CMS Software (CMSSW), a reconstruction framework, TICL (The Iterative CLustering), is being developed to fully harness the granularity and other key detector features, such as particle identification and precision timing, to counteract pile-up in the very dense environment of HL-LHC. In production, the TICL reconstruction will be required to reconstruct particle properties by clustering over 500,000 individual energy deposits produced at a rate of 1 MHz.
The project aims to investigate Graph Neural Networks (GNN) as a means of linking together Tracksters, i.e. clusters of energy deposited around energy density peaks. Such a correct linking would reduce the dimensionality of the input problem size by an order of magnitude and dramatically enhance the physics outreach of the experiment under the harsher conditions of the HL-LHC.
The student will be required to study GNN network architectures that are appropriate for the task, familiarize herself with the simulation data used for the HGCAL and generate a dataset, implement a functional GNN-based trackster linking solution, which will be compared with the current linking baseline used in CMSSW. The proposed solution must be developed in Python and be exportable to ONNX format for further integration into the CMSSW.

Bibliography / sources:

[1] Pantaleo, Felice, and Marco Rovere. The Iterative Clustering framework for the CMS HGCAL Reconstruction. No. CMS-CR-2022-037. 2022.
[2] Qasim, Shah Rukh, Jan Kieseler, Yutaro Iiyama, and Maurizio Pierini. "Learning representations of irregular particle-detector geometry with distance-weighted graph networks." The European Physical Journal C 79, no. 7 (2019): 1-11.
[3] Di Pilato, Antonio, Ziheng Chen, Felice Pantaleo, and Marco Rovere. "Reconstruction in an imaging calorimeter for HL-LHC." Journal of Instrumentation 15, no. 06 (2020): C06023.
[4] Qu, Huilin, and Loukas Gouskos. "Jet tagging via particle clouds." Physical Review D 101, no. 5 (2020):056019.
[5] Wang, Yue, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. "Dynamic graph cnn for learning on point clouds." Acm Transactions On Graphics (tog) 38, no. 5 (2019): 1-12.
[6] Velickovic, Petar, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. "Graph attention networks." stat 1050 (2017): 20.

Name and workplace of master's thesis supervisor:

prof. Mgr. Ondřej Chum, Ph.D. Visual Recognition Group FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **11.01.2023** Deadline for master's thesis submission: **26.05.2023**

Assignment valid until: **22.09.2024**

prof. Mgr. Ondřej Chum, Ph.D. Supervisor's signature	prof. Ing. Tomáš Svoboda, Ph.D. Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce her thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student's signature

Acknowledgements

I would like to extend my heartfelt appreciation to all those who have taken part in shaping my academic journey. I am deeply grateful to my supervisors, Ondřej Chum and Felice Pantaleo. Their expert guidance and mentorship have been instrumental in enabling me to undertake this truly captivating project for my master's thesis.

I thank my CMS colleagues for the many suggestions received during the development of this work. I would like to extend a special thanks to Wahid Redjeb, who has been an exceptional mentor and provided constant support and assistance throughout the project, ranging from dataset generation to integrating the final results into the production framework. I am also indebted to Eduard Cuba for his insightful contributions during our regular meetings. My acknowledgment extends to Marco Rovere, Loukas Gouskos, Huilin Qu, and Benedict Maier, who played a pivotal role in developing the TICL Framework, for their valuable comments and engaging discussions.

Most importantly, I want to express deep gratitude to my family for their love and support that enabled me to pursue my goals. Equally important, I would like to thank my loved one, Tomáš Twardzik, for his constant encouragement and belief in me. His unwavering love and understanding have been a constant source of strength for me, and I am truly fortunate to have him by my side.

Declaration

I declare that I wrote the presented thesis on my own and that I cited all used information sources in compliance with the methodical instructions about the ethical principles for writing an academic thesis.

In Prague, 25. May 2023

.....
signature

Abstract

The upcoming High-Luminosity Large Hadron Collider (HL-LHC) upgrade is set to increase the number of particle collisions, which presents a significant challenge to existing reconstruction algorithms. To address the associated rise in data complexity, the Compact Muon Solenoid (CMS) at LHC is developing a new endcap High-Granularity Calorimeter (HGCAL) that can withstand the HL-LHC's harsher conditions and investigate high-energy collisions. During the particle shower reconstruction phase in HGCAL, 3D graph structures called tracksters are produced, believed to originate from the same physics object. However, due to the detector's irregular geometry, physics processes, and particle overlaps (pile-up), tracksters are often fragmented, degrading the reconstruction quality.

In this thesis, machine learning approaches are investigated, with a particular emphasis on Graph Neural Network (GNN) models, to enhance event reconstruction through improved calorimetric clustering. An end-to-end trainable GNN-based algorithm for accumulating incomplete energy fragments into well-formed tracksters is proposed with this goal. The algorithm is integrated into the CMS Software package as a linking plug-in, and its clustering and physics reconstruction performance is evaluated on simulation data. The model presented in the thesis outperforms the currently used rule-based state-of-the-art benchmark in all metrics and improves ParticleFlow reconstruction even in the challenging environment of HL-LHC.

Keywords: Calorimetric clustering, Graph neural networks, HGCAL, HL-LHC, TICL, CLUE, CMS, Trackster linking, Event Reconstruction

Abstrakt

Nadcházející upgrade High-Luminosity Large Hadron Collider (HL-LHC) významně zvýší počet částicových srážek, což představuje výzvu pro stávající algoritmy rekonstrukce srážek kvůli souvisejícímu nárůstu objemu a složitosti dat. V důsledku této změny bude instalován nový High-Granularity kalorimetr (HGCAL) pro experiment Compact Muon Solenoid (CMS), který je mimo zvýšené přesnosti také schopen odolat silnější radiaci v HL-LHC. Během rekonstrukce částicových srážek v HGCAL, pomocí topologického spojování energetických depozitů v jednotlivých vrstvách detektorů vznikají 3D energetické shluky (trackstery). Avšak kvůli nepravidelnosti detektoru, fyzikálním procesům a překryvům částic (pile-up) jsou trackstery často fragmentovány, což snižuje kvalitu rekonstrukce.

V této práci jsem prozkoumala aplikaci strojového učení, specificky pak Grafových Neuronových Sítí (GNN), pro zlepšení kalorimetrického shlukování, které je stěžejním prvkem rekonstrukce částicových srážek. Výsledkem mé práce je end-to-end trénovatelný algoritmus na bázi GNN, který spojuje neúplné energetické fragmenty do celistvých tracksterů. Součástí práce byla i následná integrace algoritmu do CMS Software frameworku. Validace výkonnosti shlukování a fyzikální rekonstrukce modelu proběhla na simulovaných datech částicových srážek, přičemž výsledky všech uvedených metrik indikují výrazné zlepšení oproti současně využívané state-of-the-art metodě.

Klíčová slova: Kalorimetrické shlukování, Grafové neuronové sítě, HGCAL, HL-LHC, TICL, CLUE, CMS, Rekonstrukce částicových srážek

Contents

Figures

Tables

Acronyms

AI	Artificial Intelligence
CERN	European Laboratory for Particle Physics
CLUE	CLUstering of Energy algorithm
CMS	Compact Muon Solenoid experiment
CMSSW	CMS simulation and reconstruction Software
CNN	Convolutional Neural Network
DAG	Directed Acyclic Graph
DFS	Depth-First Search
ECAL	Electromagnetic Calorimeter
FC	Fully Connected Layer
GAT	Graph Attention network
GNN	Graph Neural Network
HCAL	Hadronic Calorimeter
HEP	High Energy Physics
HGCAL	High-Granularity Calorimeter
HL-LHC	High-Luminosity Large Hadron Collider upgrade
HLT	High-Level Trigger, a collection of software trigger algorithms
IP	Interaction Point
L1	Level-1 Trigger
LC	Layer-Cluster produced by the CLUE algorithm
LHC	Large Hadron Collider
LR	Learning Rate
LS2	Long Shutdown 2, second LHC long shutdown
LS3	Long Shutdown 3, third LHC long shutdown scheduled for around the end of 2023
ML	Machine Learning
MIP	Minimum Ionizing Particle
NN	Neural Network
PF	Particle-Flow reconstruction
PU	Pile-Up, an average quantity of proton-proton collisions per bunch crossing
QCD	Quantum Chromodynamics – a theory describing strong interactions
SM	Standard Model of particle physics, is a theory concerning the electromagnetic, weak, and strong nuclear interactions
TICL	The Iterative Clustering framework used for HGCAL reconstruction
TriDAS	Trigger and Data Acquisition System

Chapter 1

Introduction

The Large Hadron Collider (LHC) [EB08], the largest and most powerful particle ring-shaped accelerator in the world, is located at the European Organization for Nuclear Research (CERN) on the Franco-Swiss border in proximity to Geneva. The LHC operates by accelerating protons in both clockwise and anticlockwise directions, resulting in particle collisions at four distinct detector points (ATLAS [ABT+08], CMS [Col08], ALICE [AQA+08], and LHCb [AJ+08]) along its 27-kilometer long accelerator ring. Particle collisions transform energy into mass, producing secondary particles spreading in all possible directions. During its ongoing operational phase, the LHC generates proton beam collisions at energies of 13.6 TeV[1] with a luminosity[2] of 10^{34} $\mathrm{cm}^{-2} \cdot \mathrm{s}^{-1}$. The conclusion of the current LHC run is projected for 2025, after which the LHC and its associated experiments will be subject to extensive modernizations for the High Luminosity LHC (HL-LHC) [Col21] project, aiming to significantly increase the number of collisions in the accelerator. By 2029, the HL-LHC project is expected to achieve an instant luminosity five times greater than the LHC's nominal value, increasing simultaneous collisions by a factor of ten beyond the original LHC design. The LHC update is essential to unlocking its further discovery potential and ensuring its operation continues beyond 2029. On the other hand, it poses a significant challenge for the detectors in handling high radiation dose rates and an increased number of simultaneous interactions referred to as pile-up (PU).

One of the detectors facing the challenges posed by the HL-LHC update is the CMS detector – one of the two multi-purpose experiments at the LHC. To address these challenges, the CMS sub-detectors will undergo renovation, with the current endcap calorimeters being fully replaced with the High Granularity Calorimeter (HGCAL) [Col17b] endcaps further described in Section 2.6.1. The endcaps are situated at the end of the cylindrical detector, boasting an impressive ∼6.5 million data channels, divided into 47 layers, and strategically positioned to cover the forward region in the direction of the incident beams where most interactions happen. The calorimeter detects the energy deposits produced by the interactions of particles with its sensors. Although the construction of the HGCAL is still ongoing, precise simulations described in Section 5.1 permit the generation of datasets of particle interactions with the calorimeter (Figure 1.1), facilitating the exploration of reconstruction methods for future detector deployment.

[1] The kinetic energy of colliding protons is half that figure, 6.8 TeV, allowing them to travel at 99.999999% of the speed of light.

[2] Rate of collisions.

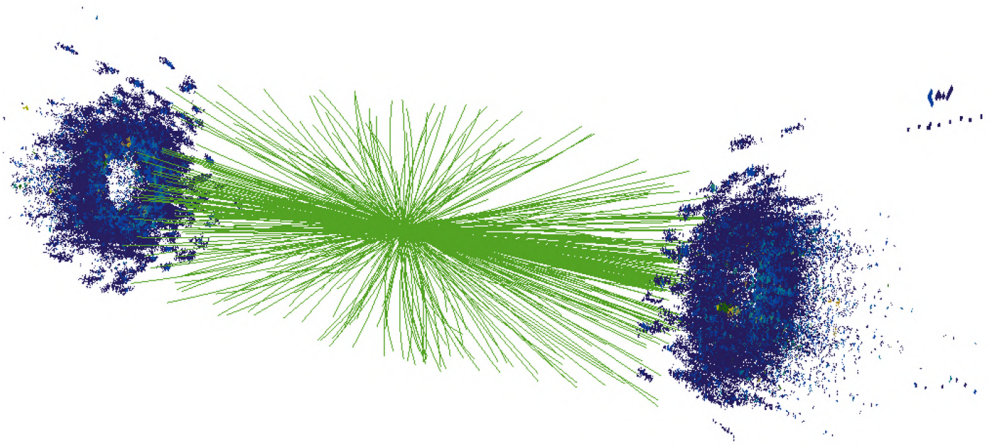


Figure 1.1: Visualization produced by the CMS Fireworks software, showing the simulation of a 140 PU particle collision event expected after the HL-LHC upgrade, in which energy deposits are captured by both HGCAL endcaps. The simulated particles follow tracks highlighted in green and, upon interaction with the calorimeter, deposit energies into the sensors, generating energetic cascades shown in dark blue.

The upgrade's impact extends beyond the need for new infrastructure since the increase in data volume and complexity also poses a challenge for existing reconstruction algorithms. In particular, the HGCAL, with its finer granularity, represents a significant departure from previous calorimeters, requiring the development of innovative algorithms to effectively reconstruct particle showers. To meet these evolving requirements, the Iterative CLustering (TICL) framework [PR22] is being developed as part of the CMS Software (CMSSW) reconstruction framework. In operation, TICL will be required to reconstruct particle showers from over $\mathcal{O}(10^5)$[3] individual energy deposits at a rate of ∼750 kHz during the online high-level triggering (HLT). Particle shower reconstruction aims to assign a label to each energy deposit in the calorimeter, also referred to as **hit**, with all hits produced by a single particle sharing the same label. The use of traditional clustering algorithms is hindered by more than a cubic increase in computational complexity as the sample size increases, which poses a problem with the HL-LHC in sight. Therefore, to reduce the complexity of the reconstruction problem, a series of two clustering steps are proposed, decreasing the problem size by one order of magnitude in every step. First, a density-based clustering algorithm CLUE [RCDP+20] is utilized to cluster hits on each detector layer as further discussed in Section 3.3.1, resulting in topological structures called **layer-clusters** (LCs). LCs are further clustered in three dimensions through the application of the CLUE3D [PR22] algorithm, producing high-homogeneity **tracksters**. Tracksters are the 3D topologically connected energy deposits thought to be part of a single reconstructed physics object, creating direct acyclic graphs (DAGs). The current HGCAL reconstruction produces thousands of tracksters per single proton-proton collision. However, the fragmentation of tracksters stemming from a single particle shower is a common occurrence, caused by factors outlined in Section 4.2, including detector irregular geometry, varying gaps between detector layers, noise, physical processes such as secondary components initiated by bremsstrahlung [KM59] for hadronic showers, and particle overlaps. Additionally, rejecting pile-up is important for correct event reconstruction. To achieve this, the CLUE3D algorithm has been carefully tuned to avoid excessive clustering, which also

[3] Please be aware that in this context, $\mathcal{O}(\cdot)$ does not refer to complexity in terms of the Big-O notation but rather indicates the order of magnitude. This notation for the order of magnitude will be utilized consistently in this work.

leads to the creation of multiple separate trackster fragments. As a result of the above, a supplementary trackster **linking** step is necessary post the three-dimensional clustering phase to improve the reconstruction process.

This thesis is an endeavor to develop strategies for improving particle shower reconstruction using machine learning (ML). ML-based techniques are recently demonstrating promising advances in terms of time complexities and physics performance. Such parameterized models learned from sufficient amounts of data often yield superior results compared to traditional approaches. Additionally, in anticipation of potential changes in the development of the HGCAL detector, data-driven methods would enable swift algorithm adaptation through learning on updated data. However, leveraging ML approaches typically necessitates careful task formulation, extensive model training, hyperparameter tuning, and data processing. The objective of this thesis is to boost the performance of collision event reconstruction in HGCAL by investigating ML methods, particularly Graph Neural Network (GNN) models for the task of trackster linking. The task comes down to identifying associated fragmented energy deposits using trackster features, such as their shapes, spatial characteristics, and neighborhood information, and connecting them together, producing more precise representations of the underlying physics objects.

1.1 Key Contributions

- In this thesis, an end-to-end trainable GNN is developed for linking tracksters emanating from a common particle shower. The problem is framed as a task of edge classification in the collision event graph, with the aim of determining whether the connections between tracksters belong to the same shower. The network is based on custom-implemented Edge Convolution layers with attention and designed to provide a balance between performance and computing resources needed for inference. It is trained using `PyTorch` library and equipped with the capability to be exported to the `ONNX` format for inference in the production environment.
- As a part of this work, three Monte Carlo-based simulated datasets of varying complexities have been created for the task of trackster linking. These datasets are used for training and performance evaluation of the implemented networks.
- In this document, I report the physics and computational performance of the proposed algorithms. It is shown that the GNN-based linking methods significantly improve the physics response with respect to a rule-based geometric linking benchmark and demonstrate reconstruction improvement even in a high PU environment.
- The proposed GNN is integrated within the CMS production environment and is available in the form of a linking plug-in during the HGCAL reconstruction stage.
- Additionally, a set of adapted clustering metrics has been devised to address the non-uniformity of data point contributions when evaluating algorithms' clustering performance.
- Finally, I investigated several methods for trackster skeletonization, which involves generating a backbone representation of the calorimetric energy structures in the form of graphs. The purpose of skeletons ranges from enabling time propagation along the skeletons to allowing extraction of additional trackster features.

1.2 Thesis Structure

This thesis is structured as follows: first, Chapter 2 discusses the theoretical background of particle physics relevant to this work, and provides a brief overview of the CMS experiment and its sub-detectors with a focus on the HGCAL, in the context of which this work was carried out. The following Chapter 3 delves into the reconstruction process for the HGCAL implemented in the TICL framework. In particular, the currently-used geometric linking is described in this Chapter, which is used as a baseline for the performance evaluation of the proposed solution. Then, the trackster linking task is introduced in Chapter 4, reviewing the relevant work to the calorimeter clustering problematics and providing motivation for this work. The created datasets are presented in Chapter 5. After that, the GNN-based linking algorithm developed as a result of this project is discussed in Chapter 6, along with the details related to the proposed graph neural network architecture. Training and performance evaluation of the GNN linking method with respect to a rule-based benchmark and other experiments are presented in Chapter 7. Finally, Chapter 8 summarizes the outcomes of the work and outlines potential avenues for future research.

For readers solely interested in the statistical modeling aspect of this work, it is possible to proceed directly to the task formulation and dataset description outlined in Chapters 4 and 5, respectively, bypassing Chapters 2 and 3. While the information provided in these preceding sections is of relevance to dataset creation and decision choices for the proposed GNN architecture, direct referencing will be employed to ensure the reader's time is utilized optimally.

Chapter 2

CMS Detector at the Large Hadron Collider

At high-energy colliders such as the CERN LHC, high-velocity particle beams collide and result in the creation of massive, potentially undiscovered particles from the conversion of energy to mass according to the axiomatic mass-energy equivalence equation $E = mc^2$. These newly formed particles, being unstable in their vast majority, rapidly, even before reaching the first sensitive layer of CMS at 1.3 cm from the interaction point, decay into other particles. The resulting particles can also be unstable and undergo further decay in a cascade of reactions, forming a *decay chain* that ultimately terminates with the production of stable particles, which can often be detected. The decay process is governed by the laws of conservation of energy and momentum, meaning that the mass of the mother particle is always greater than the combined mass of the daughter particles, and the difference in mass is released as the kinetic energy of the decay products. Hence, to create heavier and typically more interesting particles, higher-energetic collisions are generally targeted.

Stable particles are very few; they include protons, electrons, photons, and neutrinos, with other particles eventually decaying into their admixture. Theoretical physics provides information on the specific mixtures of stable particles arising from the decay of a given parent particle, along with the probability of that particular mixture occurring. Therefore, accurate detection and characterization of the resulting decay products hold the key to reconstructing particle collisions, also referred to as **events**. Detecting these mixtures of particles is a primary objective of the detectors at the LHC, enabling the assignment of the detected particles to their original constituents through the subsequent software reconstruction steps.

This chapter presents a brief overview of the key concepts in the Standard Model (SM) of particle physics, summarizing the fundamental principles underlying modern particle physics, including electromagnetic, weak, and strong interactions. Furthermore, the particle types utilized for training the linking networks are discussed within this context. Then, I provide an overview of the LHC and the Compact Muon Solenoid (CMS) experiment, including its various sub-detectors. Considering the goal of this thesis lies within improving the reconstruction of the High Granularity Calorimeter, the main focus of this chapter will be on it.

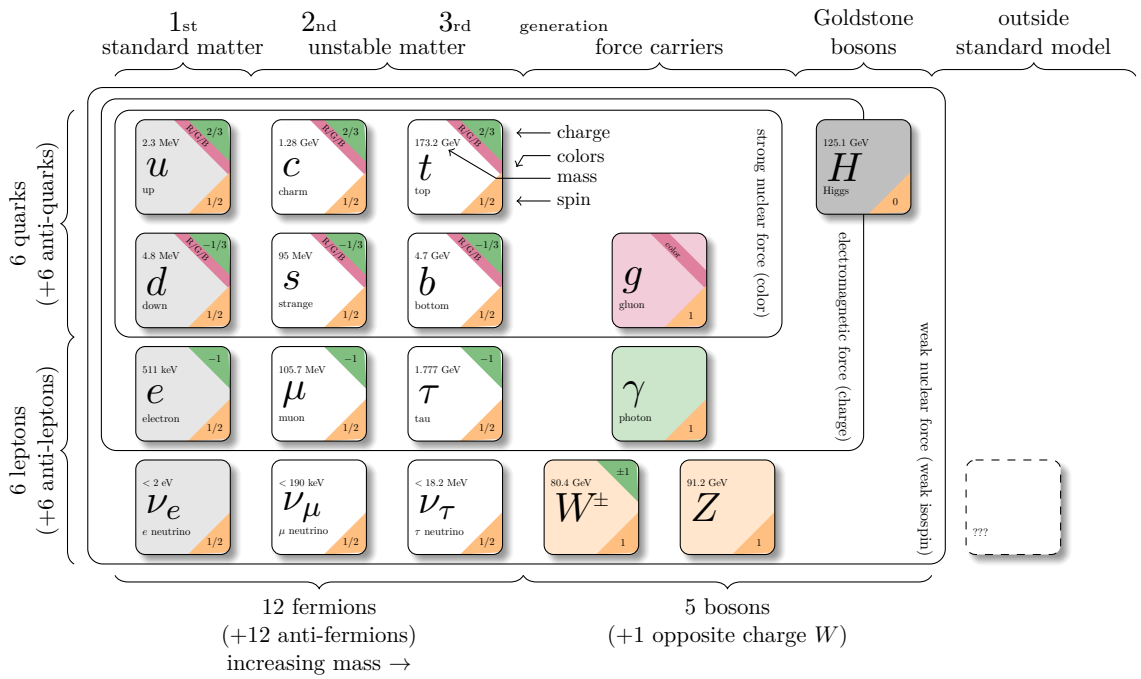


Figure 2.1: The Standard Model of particle physics. Programmed in `TikZ` by Carsten Burgard [Bur21].

2.1 The Standard Model

The SM constitutes a theoretical framework for the understanding of modern particle physics, encompassing the description of both the elementary particles and the fundamental interactions that govern them. The elementary particles, considered the fundamental building blocks of matter, can be differentiated into *bosons* and *fermions* based on their intrinsic angular momentum, or spin, with bosons possessing an integer positive spin and fermions exhibiting a positive half spin. The Standard Model encompasses twelve fermions and five bosons defined as force carriers mediating the strong, weak, and electromagnetic fundamental interactions such as gauge bosons or explaining the mass of other particles in the case of the Higgs boson. Fermions are categorized into three distinct generations – groups of particles that exhibit similar behaviors, each comprising two *leptons* and two *quarks* (Figure 2.1). A lepton generation encompasses three particles with a charge of $-e$ and their corresponding *neutrinos* (electron, electron neutrino, muon, muon neutrino, tau, tau neutrino), while a quark generation includes a positive quark with a charge of $2/3e$ and a negative quark with a charge of $-1/3e$. The six types of quarks, known as flavors, are up u, down d, charm c, strange s, top t, and bottom b. Each of the twelve fermions is paired with an antiparticle possessing the same mass as its corresponding ordinary matter counterpart but exhibits opposite electric charge and magnetic moment.

The Standard Model adopts Quantum Field Theory to provide a consistent theoretical description of the electromagnetic, weak, and strong interactions and incorporate the Higgs mechanism, while not being able to include gravity as of now. The interactions are attributed to the exchange of force-carrier particles – bosons, which mediate the interaction between particles of matter by transferring discrete amounts of energy. Each of the three fundamental forces has its own corresponding boson. The strong force is mediated by the gluon, the

electromagnetic force by the photon γ, and the weak force by the $\mathcal{W}$ and $\mathcal{Z}$ bosons.

The strong force is a short-range force (approximately 10^{-15} m) responsible for a robust binding between quarks, thereby resulting in the formation of composite particles referred to as *hadrons*. The hadrons are comprised of either a quark-antiquark pair (mesons) or three quarks (baryons), such as a proton (uud) or neutron (ddu). *Pions*, which are the particles mainly used in this work, are the lightest of the mesons, and they come in three different types: neutral π^0, positive π^+, and negative π^-. Pions are typically produced in high-energy collisions and decay rapidly. *Kaons* are slightly heavier mesons that also come in three versions, K^0 and $K^\pm$.

The electromagnetic force acts on any particle with a non-zero electric charge and is responsible for binding electrons to nuclei, among others. The infinite range of the electromagnetic force is owed to the zero rest mass of the photons, which mediates this interaction. The weak force is a very short-range force being able to change the flavor of quarks and responsible for radioactive phenomena. It is mediated by $\mathcal{Z}$, $\mathcal{W}^\pm$ bosons discovered at CERN in 1983. In order to introduce mass terms, the Higgs mechanism is included in the SM. The Higgs boson is generated by the Higgs field, which pervades all of space, and the interaction with the Higgs boson gives particles their masses.

2.2 The Large Hadron Collider

The Large Hadron Collider, located approximately 100 meters underneath the surface in close proximity to Geneva, comprises a 27-kilometer-long ring and serves as the most powerful particle accelerator ever built. Important parameters for the characterization of the particle accelerator performance are the instantaneous luminosity, $\mathcal{L}$, and cross-section σ, which define a potential number of particle interactions via the equation $\frac{dN}{dt} = \mathcal{L} \cdot \sigma$, where the quantity $\frac{dN}{dt}$ represents the number of collisions per unit of time. The luminosity is determined completely from the colliding beam properties, and for Gaussian-shape n bunched beams colliding head-on, it can be expressed as

$$\mathcal{L} = f \frac{n N_a N_b}{4\pi \sigma_x \sigma_y} \quad , \tag{2.1}$$

where the bunch cross-section profiles at the interaction point σ_x, σ_y are perpendicular to their flight direction. N_a and N_b denote the number of particles in each of the two particle bunches, which repeatedly collide at frequency f during the experiment.

The performance of a high-energy accelerator can also be evaluated through its capacity to produce beam collisions useful for the high-energy physics community, quantified through the integrated luminosity $L = \int \mathcal{L} dt$. L defines the amount of data delivered by the LHC over a given period of time, usually a year of data taking. The typical unit of integrated luminosity is the inverse femtobarn. To provide a sense of scale, 1 fb^{-1} corresponds to approximately 100 trillion (10^{12}) proton-proton (pp) collisions.

Designed to collide oppositely circulating bunches of protons at rates up to 40 million collisions per second, the LHC was originally engineered to operate with a center-of-mass energy of $\sqrt{s} = 7$ TeV and a luminosity of $\mathcal{L} = 10^{34}$ cm$^{-2}\cdot$ s^{-1}. It is worth noting that the LHC is not limited to proton collisions, as it can also collide heavy ions such as Pb-Pb (and also p-Pb) with an energy of 2.8 TeV per ion and luminosity of 10^{27} cm$^{-2}\cdot$ s^{-1}, but this is

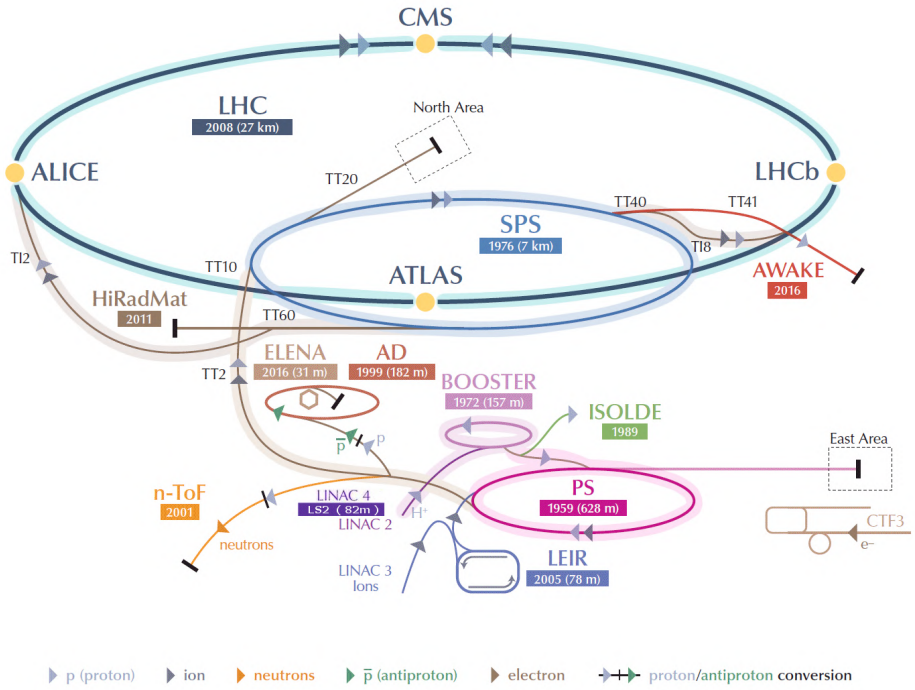


Figure 2.2: Schematic representation of the LHC and the accelerator complex with the relevant parts highlighted in their respective colors. The accelerator chain includes the proton source, LINAC2, proton synchrotron booster (PSB), proton synchrotron (PS), super proton synchrotron (SPS), and ultimately the LHC. Positioned along the circular pathway of the LHC are four main detectors, namely CMS, ATLAS, LHCb, and ALICE. The image is courtesy of CERN [Lop22].

beyond the scope of this work. The luminosity of the LHC decays during an experiment run with a lifetime of approximately 14.9 hours, owing to losses from collisions, photon emissions along the circular path, and scattering at residual air. As a result, fresh bunches of protons are injected into the LHC every one or two days.

At the beginning of the experiment, hydrogen atoms are stripped of their electrons, and the extracted protons are subsequently accelerated through several pre-accelerators, gradually increasing their kinetic energy. Protons are first injected into the LINAC2, rising their energy to 50 MeV ($\sim 0.31c$[1]), followed by the Proton Synchrotron Booster that takes them to 1.4 GeV ($\sim 0.916c$). Subsequently, the Proton Synchrotron raises the energy to 26 GeV ($\sim 0.9994c$) and, finally, the Super Proton Synchrotron propels the protons to 450 GeV ($\sim 0.999998c$, or just ~ 649 m/s slower than the speed of light). Along the LHC ring, protons are accelerated within two adjacent vacuum tubes to the energy of 6.8 TeV[2] before being subjected to collisions at the centers of the four experiments (Figure 2.2): ATLAS (A Toroidal LHC ApparatuS), CMS (Compact Muon Solenoid), ALICE (A Large Ion Collider Experiment) and LHCb (LHC Beauty), which aim to capture information regarding the direction, energy, and properties of the resulting decay products from inelastic particle collision events. To achieve this, the detectors are partitioned into sub-detectors, each equipped to gather complementary information.

[1]Calculated with $\frac{v}{c} = \sqrt{1 - \left(1 - \frac{m_0c^2}{E_k + m_0c^2}\right)^2}$, where $m_0c^2 = 938$ MeV for protons.

[2]The ultimate velocity of protons can then reach $\sim 0.999999991c$ in LHC, or ~ 2.7 m/s slower than the speed of light.

2.3 Compact Muon Solenoid

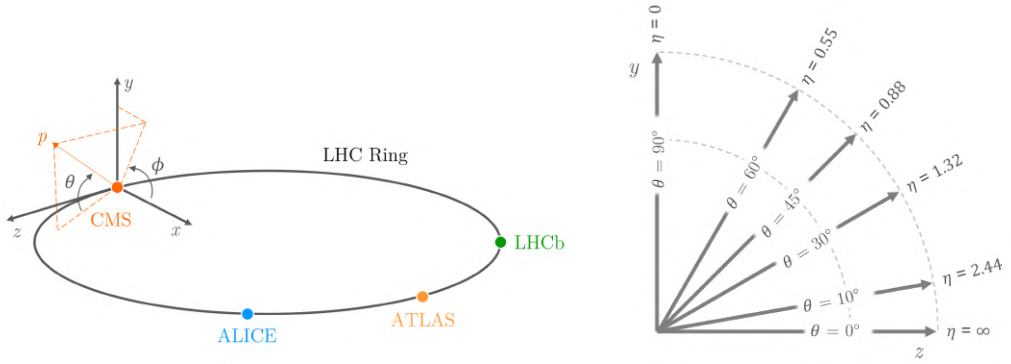
The Compact Muon Solenoid experiment is one of the two general-purpose detectors at LHC that have played a crucial role in numerous groundbreaking discoveries in particle physics, including the co-discovery of the Higgs boson [Col12] in 2012 along with the ATLAS experiment. The CMS experiment was envisioned with the purpose of uncovering the Higgs Boson and exploring new physics phenomena that may exist beyond the boundaries of the current SM paradigm. The experiment aims to shed light on some of the most pressing questions in particle physics, including the nature of dark matter and energy, the origin of matter-antimatter asymmetry (the Charge-Parity Violation), the unification of forces, and allows to explore the viability of supersymmetry, string theory, or extra dimensions. In addition, CMS was designed to perform precise measurements of various SM physics processes and test their validity, such as the production of $\mathcal{W}$ and $\mathcal{Z}$ bosons, top quarks, and heavy-flavor physics, among others. Although CMS has similar scientific goals as the ATLAS experiment, it uses different technical solutions as well as distinct event reconstruction software techniques.

The Compact Muon Solenoid gets its name from its three defining properties. First of all, at 14.6 meters high and 21.6 meters long, it is quite *compact* for all the detector material it accommodates. Interestingly, relative to ATLAS, which has approximately 1.5 times the diameter of CMS and two times its length, CMS at 12 500 tons is almost double the weight of ATLAS detector. Secondly, CMS is designed with a strong emphasis on detecting muons, which serve as excellent signatures of interesting physics. They can be easily identified and, contrary to electrons and mesons, can only come from the decay of heavy and, hence, possibly interesting particles. For example, muons can be produced in the decay of $\mathcal{W}$ and $\mathcal{Z}$ bosons, which are important in the study of the weak nuclear force. As a result, muon chambers contribute strongly to the CMS Trigger system – the system trying to identify worth-to-study pp interactions from the multitude of non-interesting events. Lastly, the solenoid in the CMS name refers to the fact that a *solenoid* (the most powerful one to have been created at the time of the construction) is in the core of the detector's construction since a magnetic field of substantial strength is necessary to attain precision in the measurement of the momentum of charged particles.

2.3.1 Coordinate System and Conventions

To provide an accurate depiction of its sub-detectors' function, the CMS experiment employs a specific coordinate system and conventions. The experiment adopts a right-handed Cartesian coordinate system, with the origin located at the proton beams' interaction point (IP). The z-axis of the system is oriented along the anti-clockwise beam direction, the y-axis points upward perpendicular to the LHC plane, and the x-axis extends towards the center of the LHC ring (Figure 2.3a and Figure 2.4).

In addition to the Cartesian system and in view of the cylindrical geometry of the detector, the experiment also utilizes spherical coordinates. The azimuth half-angle ϕ indicates the angle in the $x-y$ plane measured from the positive x-axis, and the polar angle θ denotes the angle relative to the positive z-axis with $\theta = 0°$ corresponding to the beam axis. Consequently, the transverse particle momentum, p_{T}, is defined using both coordinate systems as follows:



(a) : Schematic representation of the CMS coordinate system.

(b) : Relation between the pseudorapidity η and the polar angle θ.

Figure 2.3: CMS coordinate system.

$$p_{\mathrm{T}} = \sqrt{p_x^2 + p_y^2} = p \cdot \sin\theta \quad . \tag{2.2}$$

In the context of collider physics, the use of pseudo-rapidity η, defining the angle between the particle momentum and the beam axis, is favored over the polar angle θ since the particle production is generally constant as a function of rapidity. The conservation of transverse momentum under Lorentz transformation along the z-axis leads to:

$$\eta = -\ln\left(\tan\frac{\theta}{2}\right) \quad . \tag{2.3}$$

Pseudo-rapidity varies from 0 at $\theta = \pi/2$ to $\pm\,\mathrm{inf}$ at $\theta = 0$ as shown in Figure 2.3b. In this reference frame, it is straightforward to describe the locations of the CMS sub-detectors, installed radially from the inside out, as outlined in detail in the following sections. The concept of the "forward" direction is used to refer to regions of the detector that are close to the beam axis at high $|\eta|$ values. Additionally, the spatial separation between two particles, denoted as ΔR, can be formulated using their azimuthal angle and pseudo-rapidity, and is also Lorentz-invariant along the z-axis:

$$\Delta R = \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2} \quad . \tag{2.4}$$

Another important event's property is the negative sum of the transverse momenta of all the reconstructed particles, known as missing transverse momentum p_{T}^{miss}. This quantity is derived from the detector's symmetry around the interaction point and its hermetic nature. It is commonly interpreted as the total transverse momentum of neutrinos or other hypothetical non-interacting particles that have escaped the detector without leaving any detectable signature.

2.3.2 CMS Sub-Detectors

The CMS detector is nested cylindrically along the beam axis and consists of the following main sub-detectors (from the inside out): an all-silicon inner tracker, a crystal Electromagnetic

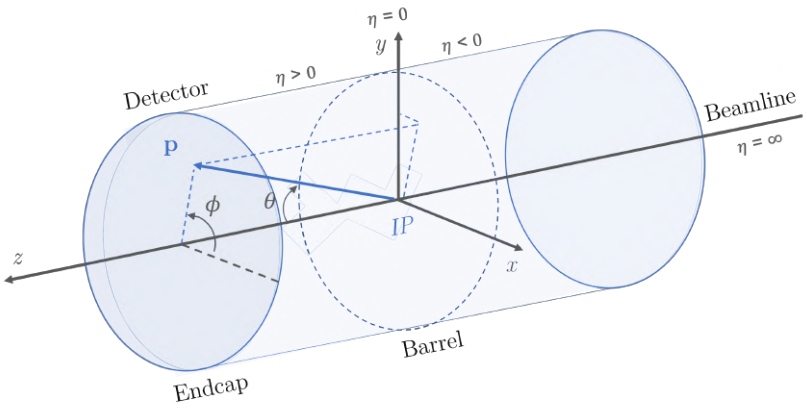


Figure 2.4: A more detailed visualization of the coordinate system adopted by the CMS detector. The detector is split into two regions – barrel cylindrically surrounding the beam axis and the two endcaps perpendicular to the beam direction.

calorimeter (ECAL), a brass and scintillator Hadronic calorimeter (HCAL), all of which operate within a superconducting solenoid magnet, and gas-ionization muon chambers embedded in the solenoid flux-return yoke outside of the solenoid's magnetic field. A simplified layout of the CMS cross-section is captured in Figure 2.5.

Tracker ($r < 1.2$ m, $|\eta| < 2.5$)

At the core of the detector setup, positioned closest to the IP, lies the tracker, comprised of multiple concentric layers of silicon sensors. Only charged particles trigger a signal in the tracker by ionizing the semiconductor and producing signals called hits. Tracker consists of pixel and micro-strip layers, capturing the spatial position of the particle passage. The hit information obtained from the multiple layers of the detector allows for the reconstruction of particle trajectories, referred to as *tracks*, and the identification of their origins, or *vertices*. This information is used to study both the primary pp interactions and secondary vertices resulting from particle decays. In conjunction with a powerful uniform magnetic field of $B = 3.8$ T, the trajectory reveals the particle's charge q and transverse momentum p_{T} orthogonal to the magnetic field. The underlying concept of particle tracking is predicated upon the principle of the deflection of charged particles in a magnetic field, with the orientation of the bend in the track revealing the polarity of the charge. Assuming elementary charge, the momentum p_{T} is estimated according to the relativistic relation $p_{\mathrm{T}} = qBr/\sqrt{1-(v^2/c^2)}$, where r is the bend radius of the particle trajectory.

Calorimeters

While the tracker focuses on measuring particle trajectories with an emphasis on minimizing loss of particles' energies[3], the subsequent sub-detector layers, referred to as *calorimeters*, function to entirely halt the particles, thereby absorbing their full energy (destructive measurement). Unlike in tracker, this process allows detecting even neutral particles. However, as high-energy primary particles traverse through dense calorimeter material, they do not lose their energy all at once. Instead, they create cascades of lower-energy secondary particles, referred to as particle showers. These low-energy particles are eventually absorbed and deposit

[3]Coming from nuclear interactions, multiple scattering, bremsstrahlung, and photon conversion.

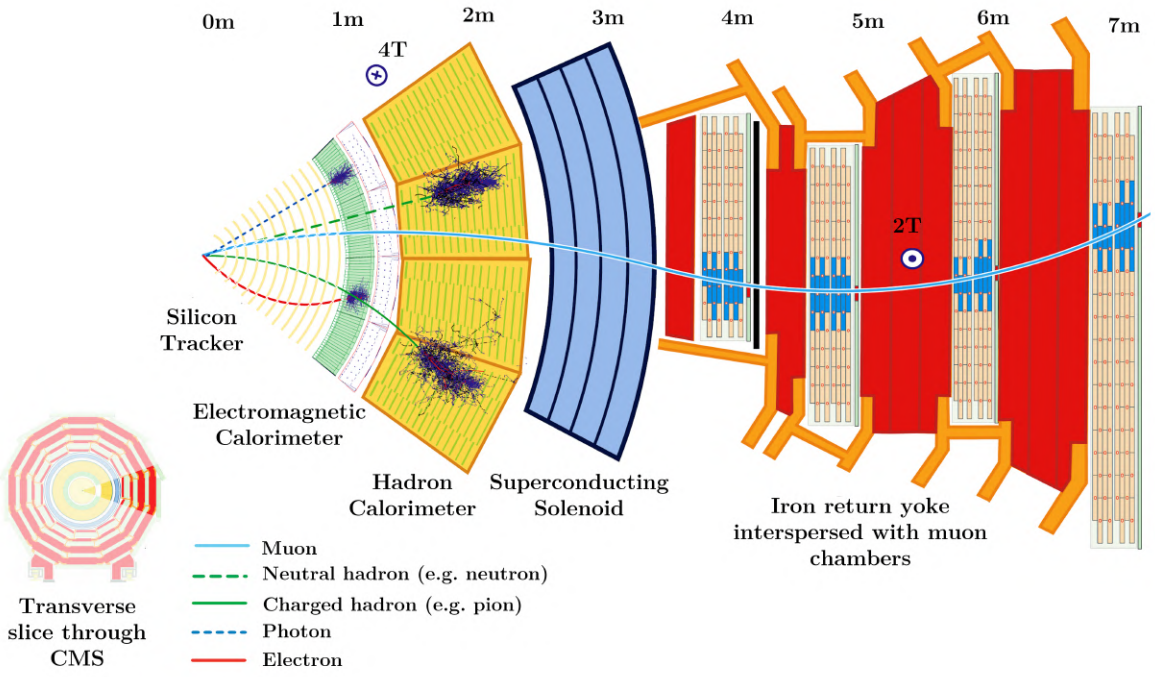


Figure 2.5: CMS transverse slice illustration displaying various sub-detectors and individual particle interactions. Image is based on materials sourced from CERN [Col23].

their energy into the calorimeters. CMS employs two calorimeters: the ECAL and the HCAL.

The Electromagnetic Calorimeter ($1.2\ \text{m} < r < 1.8\ \text{m}$, $|\eta| < 3$). The ECAL is positioned immediately exterior to the tracker and consists of approximately 76 000 dense scintillating lead tungstate $PbWO_4$ crystals, with short radiation lengths resulting in compact electromagnetic showers. ECAL is a homogeneous detector, meaning it is built of a single medium, which plays the role of both absorber and detector. The principle aim of the ECAL is to measure the energy of incident particles interacting electromagnetically, such as $e^{\pm}$ and γ, which get absorbed and release their full energy in the ECAL. The electromagnetic showers produced by these particles are detected as clusters of energy recorded in neighboring detector cells, which are read out by silicon avalanche photodiodes, allowing determining both the energy and the direction of the particles. Charged and neutral hadrons may also initiate hadronic showers in the ECAL, which are subsequently fully absorbed in the hadron calorimeter. ECAL consists of the barrel part, the endcap part[4], and a pre-shower system in front of the endcap.

The Hadronic Calorimeter ($1.8\ \text{m} < r < 2.9\ \text{m}$, $|\eta| < 5$). The ECAL is surrounded by brass and scintillator sampling HCAL, made up of layers of passive absorber alternating with active detector layers. HCAL's main parts, the hadron barrel and the hadron endcap are located inside the magnetic field. In contrast, two other parts, the hadron outer and a forward hadronic calorimeter, are extended beyond the magnet and serve as a means of reducing the loss of energy from high-energy jets. The forward region[5] of the HCAL consists of steel

[4]Barrel is part of the detector cylindrically surrounding the beam axis, and the two endcaps are the detector regions perpendicular to the beam direction.

[5]The "forward region" of a detector refers to the part of the detector closest to the direction of the particle beam, where particles produced at small angles relative to the beam direction can be detected.

absorbers with embedded quartz fibers. Traversing charged particles produce Cherenkov light in the fibers proportional to the original hadron energy and detected with photomultipliers. In contrast to ECAL, HCAL focuses on detecting particles interacting hadronically. Such particles are, for example, protons, neutrons, pions, and kaons. Since these are composite subatomic particles, HCAL plays an essential role in identifying quarks by measuring the energy and direction of jets – tightly-focused sprays of particles produced by the hadronization of quarks or gluons.

The Muon chambers ($4.0\ \text{m} < r < 7.4\ \text{m}$, $|\eta| < 2.4$)

Muons and other low-interacting particles, such as neutrinos, are able to traverse the calorimeters with little to no interactions. While neutrinos escape CMS undetected, muons, which have a lifetime long enough to leave CMS (on average $2 \cdot 10^{-6}$ s), are detected by additional muon chambers constituting the outermost shell of the CMS detector. The muon system is comprised of four distinct gaseous ionization detectors: Drift Tubes (DT) in the barrel region, Cathode Strip Chambers (CSC) in the forward region, Resistive Plate Chambers (RPC), and gas electron multipliers. These parts are integrated within the consecutive layers of the iron return yoke of the superconducting magnet, which provides a region of the "uniform" magnetic field outside the solenoid and acts as a filter for other particles produced in a collision apart from muons and neutrinos. The DT and CSC detectors are responsible for accurately determining the position of muons, which in turn allows for precise measurement of their momentum. Meanwhile, the RPC chambers have lower resolution ($\sim$ cm) but are very fast (around a few ns to give a signal) and are therefore used for bunch-crossing identification and provide information to the Level-1 trigger system described in the following section. The operation of gaseous detectors relies on the ionization of gas atoms by a passing charged particle. This ionization process frees electrons and positive ions, which are then detected by electronics as they are driven towards the anode (or the cathode for ions) by the electric field.

2.4 Trigger and Data Acquisition System

Due to the limited storage and processing capabilities, retaining all the events generated at CMS for analysis is not feasible since more than a billion pp interactions will take place every second inside the CMS detector after the HL-LHC upgrade. Additionally, only a minuscule portion of the beam crossings result in *hard* head-on interactions with sufficient momentum transfer, which are the ones producing potentially interesting interactions most likely to reveal new phenomena. Thus, a preliminary step of event selection (i.e., high-energetic particle interactions or their unusual combinations) is performed before the data is prepared for offline analysis. The CMS Trigger and Data Acquisition System (TriDAS) [KST+17] plays a vital role in sequentially reducing the massive data flow produced by 40 MHz (rate of up to 80 TBps) proton-proton beam crossings to a manageable level of at most 7.5 kHz of interesting events for storage and further analysis[6].

Level-1 Triggering. TriDAS operates in several stages, beginning with a fast Level-1 (L1) trigger based on a hardware system of programmable electronics, able to select or reject events

[6] values as per Phase-2, see the next section. Note that as of the current CMS run in 2023, the triggering frequencies are approximately 5 to 7.5 times lower than the ones mentioned in this section.

in real-time (in about 12.5 µs after the bunch crossing) based on simplified data from the calorimeters and muon RPCs. During this period, high-resolution data from the detectors is temporarily stored in buffers, waiting to be processed further. A decision on whether to tentatively accept the data to be further passed to the High-Level Trigger (HLT) or reject it is made based on the detection of so-called *trigger primitives*, – candidate objects such as ionization deposits consistent with muons, energy clusters consistent with particles exceeding certain transverse momentum thresholds, missing transverse energy E_{T}^{miss}, or jets. As a result, the L1 triggering stage reduces the event rate from 40 MHz to no more than about 750 kHz [Col21].

High-Level Trigger. The HLT is a software-based trigger performing a more sophisticated event analysis, including applying fast, simplified pattern recognition algorithms and event reconstruction, similar to that done during the offline reconstruction, but with an average processing time of the order of 1 s per event. The HLT follows a sequence of algorithms searching for specific signatures, which involve increasingly complex reconstruction and filtering. Once one of the filters fails, the event is discarded, and the remaining parts (such as the computationally expensive particle-flow algorithm) of the HLT triggering are skipped, saving processing time. Additionally, to reduce any dead time, the reconstruction process is limited to specific areas around L1 or higher-level objects, further reducing the time required. The selected events are temporarily stored on a local disk before being transferred to the CMS Tier-0 center at CERN for long-term storage and offline analysis. The rate of events selected by the HLT ranges from 2.5 kHz to 7.5 kHz, with the vast majority of events being processed immediately.

Despite the utilization of triggers, the sheer volume of data still generated and stored on disk remains substantial, reaching about 12 Petabytes of data annually when CMS is performing at its peak. Hence, the requirement for highly efficient algorithms at all stages is imperative.

2.5 High Luminosity LHC Upgrade

At the current moment, LHC has collected around 450 fb^{-1} of pp collision data and is currently preparing for its High-Luminosity upgrade [ABNR17], which will allow it to run at five times its original design instantaneous luminosity and consecutively increase the number of particle collisions almost ten times[7]. This increase in luminosity will allow collecting more hard collision data, leading to more precise measurements, and will enable the investigation of rare processes difficult to observe with the current LHC configuration. The HL-LHC, or the Phase-2 operation, will start after the three-year-long third Long Shutdown (LS3) in the years 2026–2029. The timeline of the upgrade is depicted in Figure 2.6.

The upgrade also brings the challenge of an unprecedented number of simultaneous PU collisions per crossing and significant concerns regarding radiation damage to detectors. At the nominal HL-LHC luminosity, the expected average pile-up is 140, while for the ultimate scenario from 2035 and beyond, this number increases to 200. In comparison, as of 2022, the

[7]The HL-LHC is scheduled to operate at a nominal luminosity of $\mathcal{L} = 5 \cdot 10^{34}\ \mathrm{cm}^{-2} \cdot\ \mathrm{s}^{-1}$, with a built-in margin to achieve an *ultimate* performance of $7.5 \cdot 10^{34}\ \mathrm{cm}^{-2} \cdot\ \mathrm{s}^{-1}$. Furthermore, HL-LHC aims at delivering an integrated luminosity of up to 3000 fb^{-1} over about ten years of operation, starting from 2029.

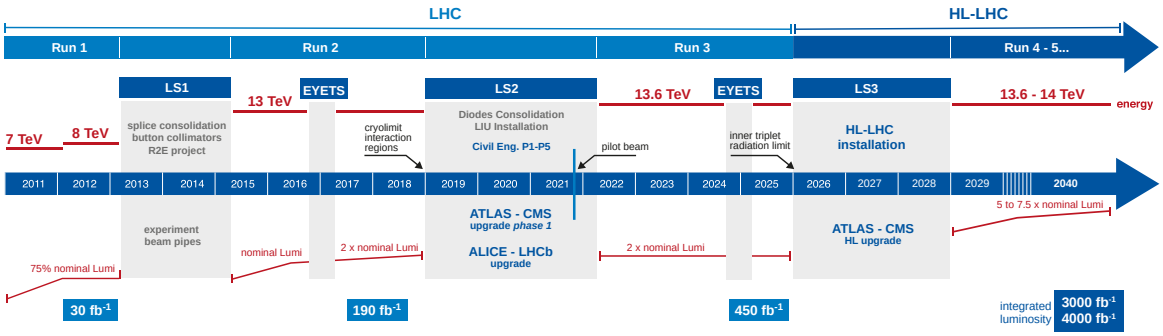


Figure 2.6: The timeline displays the previous and expected collision energy (upper line) and instantaneous luminosity (lower line) of the LHC from 2011 to 2040. The LHC runs on a cycle of alternating phases, wherein data collections take place during operational runs, and the machine undergoes upgrades and maintenance during shutdowns to prepare for the next run. The operation of the LHC from 2011 consists of four runs with three shutdown periods. In the Run-1 from 2011 to 2013, LHC delivered about 30 fb^{-1} pp collision at $\sqrt{s} = 7 - 8$ TeV. In the Run-2 from 2015 to 2018, LHC produced 190 fb^{-1} collisions at $\sqrt{s} = 13$ TeV. In 2023, the LHC is after its second long shutdown period, in Run-3, operating at the maximum collision energy of $\sqrt{s} = 13.6$ TeV. After Run-3, LHC will be upgraded to a higher luminosity, reaching five times as much as the current value (as of 2023) [lhc].

level of pile-up was, on average, just 35, and it has recently been increased[8] to reach about 60 simultaneous collisions per bunch crossing. The average pile-up is defined as follows:

$$\langle PU \rangle = \frac{L\sigma_{pp}}{n_b f} \quad , \tag{2.5}$$

where σ_{pp} is the inelastic pp cross-section, n_b is the number of bunches present in the beam, and f is its revolution frequency.

2.6 The Main CMS HL-LHC Upgrades

To maintain a good physics performance in the challenging conditions of the HL-LHC, the CMS Collaboration, along with the other LHC experiments, is planning to carry out a series of extensive upgrades. These upgrades are expected to be commissioned during the three-year LS3, with the main focus on the following aspects:

- The high radiation doses experienced in the HL-LHC environment necessitate a complete overhaul of the tracker and the endcap calorimeter systems. Updates are also required for the electronics systems in both the barrel calorimeters and the muon detectors.
- To cope with the increase in PU rate, highly granular readouts are necessary, along with the introduction of precision timing. Upgrades to the electronic readout of the ECAL and HCAL barrels are planned to increase the granularity and provide additional timing measurements [Arc18] by introducing a new Minimum Ionizing Particle (MIP) Timing Detector (MTD) [BTdF00] placed in front of the calorimeters. Novel methods for mitigating PU during the event reconstruction must also be developed.

[8] As of the beginning of 2023.

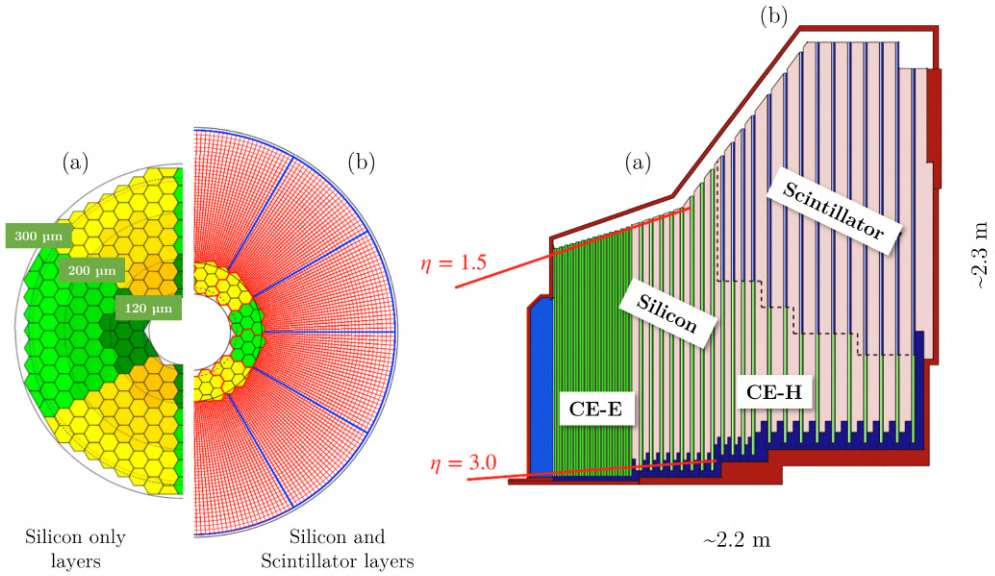


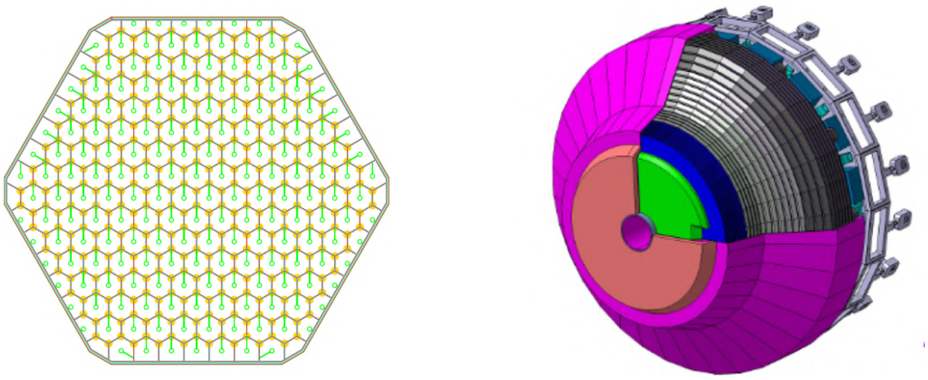
Figure 2.7: Cross-section view of the HGCAL endcap on the longitudinal-radial $z - r$ plane is shown on the right, indicating CE-E and CE-H sections of the calorimeter. CE-E has 26 full Si layers, while CE-H has seven full Si layers plus 14 Si-scintillators hybrid layers. Representative layouts of an all-silicon CE-E layer (a) with variable sensor thicknesses and CE-H layer (b) with a mixture of silicon at high η and scintillator at lower η are shown on the left. Silicon wafers are shown as yellow and green hexagonal cells, while scintillators are shown as red mesh. Darker, medium, and lighter shades of hexagons represent silicon wafers with thicknesses of 120, 200, and 300 µm, respectively. Re-drawn based on [Col17b].

- The expected high luminosity greatly increases the data stream, necessitating improvements to the L1 Trigger primitives and the overall TriDAS system.

While most of the ECAL and HCAL barrels will be retained, the endcap calorimeters in the forward region of the detector, which are subject to much greater radiation damage, will be entirely replaced with the High Granularity Calorimeter [Col21]. The HGCAL is radiation-hard and is designed to offer enhanced shower separation, particle identification, and precise timing information. This thesis focuses on the reconstruction of the HGCAL, so a more detailed discussion of its design is provided in the following section.

2.6.1 The High Granularity Calorimeter

Two HGCAL endcaps will be mounted on both sides of the CMS detector, each covering the $1.5 < |\eta| < 3.0$, corresponding to the forward region of the CMS detector. Each endcap will be divided into two compartments: electromagnetic CE-E and hadronic CE-H. Different regions of the detector will experience distinct levels of radiation exposure. Hence, the thickness of the active sensors needs to be adjusted to achieve the best balance between resolution and radiation hardness. Regions closer to the IP will experience a higher particle flux and require thinner sensors. In contrast, regions further away, where the showers are more spread out, will experience lower particle flux and may use thicker sensors, leading to detector irregular geometry. The CE-E comprises 26 active hexagonal silicon layers interleaved with CuW, Cu, and Pb absorbers. Each layer is divided into 8-inch hexagonal sensors varying in the active



(a) : Zoom-in view of the Si wafer. **(b) :** The High Granularity Calorimeter endcap.

Figure 2.8: Subfigure (a) shows the hexagonal silicon wafer, while (b) presents a 3D rendering of the HGCAL endcap [Col17b].

thickness of 120, 200, and 300 µm with several single readout diodes with an active area of 0.5 or 1.0 cm^2. The hexagonal silicon sensor layout is shown in Figure 2.8, along with the 3D rendering of the HGCAL detector. The CE-H consists of 21 layers, including seven all-silicon layers with finer sampling at smaller radii and 14 silicon/scintillator mixed squared-sensor ($\sim$4–30 cm^2 in size) layers in the outer regions exposed to less radiation dose. The scintillator tiles are arranged in a grid pattern in the $r - \phi$ plane, with their size increasing with the radius of the grid. The arrangement of the silicon modules in the CE-E and in the front part of the CE-H, as well as the mixed technology of silicon sensors and scintillator tiles employed in the back part of the CE-H, are shown in Figure 2.7.

The CE-E and CE-H compartments result in a total detector depth of around 2.2 meters fitting 47 total layers. The HGCAL has an area of approximately 620 m^2 of silicon sensors and 370 m^2 of plastic scintillator tiles with the silicon photomultiplier readout. The silicon sensors provide a total of 6 million channels that are read out individually, while the plastic scintillator tiles feature 240 thousand channels. HGCAL's unprecedentedly high granularity will allow for capturing fine details of particle showers as they propagate through the detector material at the finest level ever achieved, which is important for preserving the resolution of electromagnetic energy measurements and compensating for the effects of hadronic showers.

The five-dimensional information provided by the HGCAL (energy, x, y, z, and time t) is well-suited for particle-flow reconstruction, which is a technique discussed in the following Chapter, used to identify and measure the properties of individual particles within an event by combining information coming from different sub-detectors. The precise timing information provided by the HGCAL will help distinguish clusters of energy deposits coming from different particles in a single bunch crossing. This is practical for mitigating the effects of pile-up by rejecting hits recorded outside a certain time interval Δt. Timing will also be useful for locating the vertex of triggered hard interactions within the dense environment of the HL-LHC. Finally, the high granularity and precise timing information of the HGCAL can benefit ML and pattern recognition algorithms by making use of the high data dimensionality to improve the accuracy of particle reconstruction. However, the irregular geometry of the HGCAL means that uniform algorithms are not appropriate, as further discussed in Chapter 4, and adapted methods must be developed to take full advantage of the detector's capabilities.

Chapter 3

Event Reconstruction

The coalescence of data from the diverse sub-detectors employed in the reconstruction of events at CMS is a challenging task demanding careful consideration of both the required physics performance and the available computational resources. These considerations show to be of particular significance for the HL-LHC, where in light of the collision rate of 40 MHz, corresponding to a proton bunch separation of just 25 ns, and an average PU level of 140 (or 200 in Run-5), the resulting proliferation of signals from the sub-detectors, numbering up to several million, requires the reconstruction algorithms to demonstrate good performance in face of formidable computational demands. HGCAL reconstruction is handled by the TICL framework, whose purpose is to associate multiple individual energy deposits that are likely to originate from the same particle to form a single reconstructed object, known as a TICL candidate. The TICL reconstruction not only associates the energy deposits but also provides information about the particle's identification and estimates its 4-momentum[1].

This chapter provides a brief overview of the CMS techniques for reconstructing physics objects and particle candidates from detector data, allowing for the interpretation of events and their underlying physics processes. This chapter is primarily centered on the reconstruction in HGCAL since the main objective of this thesis is to improve it.

3.1 Particle Interactions

The CMS detector is capable of observing several types of particles and their corresponding antiparticles, each of which interacts with the detector in a specific manner. These direct observations include photons, electrons and positrons, charged hadrons (e.g., protons, pions $\pi^{\pm}$ and kaons $K^{\pm}$, etc.), neutral hadrons (e.g., neutrons, π^0, K^0, etc.), and muons.

To summarize what was already partially mentioned in the previous Chapter, charged particles are identified by associated tracks in the tracker detector, whereas calorimeters exhibit two distinct types of interaction processes depending on the interacting particle type – electromagnetic and nuclear processes. Electromagnetic interaction results in a cascade

[1] A particle's motion in the collider can be described using Special Relativity, which among others uses a vector with four components called the Lorentz Vector or a 4-momentum. The components of the vector are the particle's energy and the three components of its momentum, providing nearly complete information about the particle, except for its spin. From this vector, the rest mass uniquely defining a certain particle type, can be inferred.

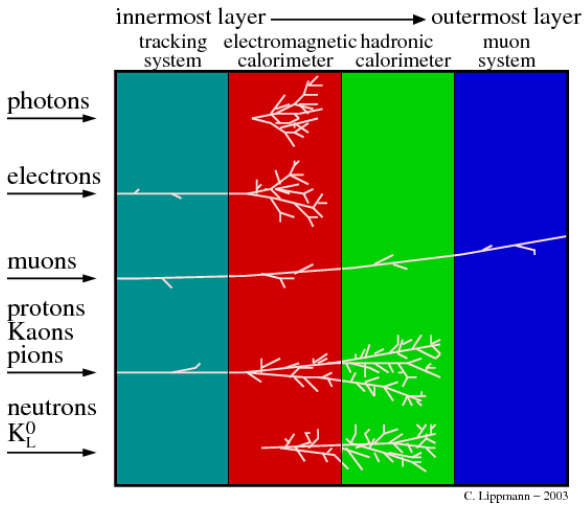


Figure 3.1: Schematic diagram of the interaction of various particles within the CMS sub-detectors. The diagram illustrates that photons interact primarily with the electromagnetic calorimeter, while electrons lose their energy in the ECAL. Muons pass through both calorimeters and are detected in the muon system, while charged and neutral hadrons are stopped in the hadronic calorimeter [Lip12].

of particles that produce ellipsoid-shaped showers, with the cascade continuing until the shower components reach a critical energy threshold at which no new particles are created. Electrons with enough energy (usually above ∼10 MeV) predominantly lose energy and produce secondary photons via bremsstrahlung. High-energy photons commonly convert into secondary electrons and positrons through the e^+e^- pair production. In contrast, heavier hadronic particles are responsible for creating hadronic shower cascades. These particles interact primarily through strong nuclear interaction, leading to the formation of multiple disconnected ellipsoid-like showers. In hadronic showers, particles such as neutrons, pions, or protons, as well as charged particles such as electrons and positrons, are often created as secondary particles. Therefore, both the electromagnetic and hadronic parts of the detector can detect hadronic showers, but the interactions are largely observed in the hadronic part due to the presence of heavier absorber plates.

As a result of the above, particles in the detector are identified through their distinct interaction patterns. Photons, which are neutral particles, do not leave tracks, instead, they interact electromagnetically and leave deposits in the ECAL. Electrons and positrons create track signatures in the tracker and deposit their energy in the ECAL. Hadrons are much heavier and can cross the full ECAL before depositing their energy in the thicker HCAL. Charged hadrons interact in both the tracker and HCAL, as well as provide a part of the signal in the ECAL. Neutral hadrons, such as neutrons, do not leave hits in the tracker and produce signals mostly in the HCAL, with some initial interaction in the ECAL. Muons, on the other hand, traverse all detector components and are identified by signals in the inner detector and in the muon chambers. Neutrinos do not produce detectable signals in any of the sub-detectors and escape undetected, although their presence can be indirectly inferred from the transverse missing energy in the event. An illustration of the interactions in the corresponding sub-detectors can be seen in the accompanying Figure 3.1.

3.2 ParticleFlow Reconstruction

Before any higher-level reconstruction can be performed in the CMS detector, traversing particles generate merely three fundamental types of information: *tracking hits*, *energy deposits*, and *time* information. The primary aim of the ParticleFlow (PF) [Sc+17] event reconstruction algorithm is to use data from the individual sub-detectors to identify and reconstruct all particles emerging from a collision. The algorithm takes inputs in the form of particle-flow elements within individual sub-detectors, including tracks reconstructed from tracker hits and muon systems, as well as calorimeter clusters produced by clustering energy deposits. The tracks and clusters are then connected through a linking[2] process to create PF blocks, which effectively summarize the behavior of a possible particle candidate across all sub-detectors. For instance, a charged particle track can be extrapolated to the ECAL sub-detector and linked with a compatible ECAL cluster to form a block. Linking also extends to the HCAL and muon sub-detectors, and additional information, such as calorimeter shower shapes, is taken into consideration at this stage. The PF algorithm can reconstruct electrons, positrons, muons, photons, charged and neutral hadrons, hadronic jets, missing transverse momentum p_{T}^{miss}, and other physics objects. Jets, which are collimated streams of particles that originate from the fragmentation of quarks or gluons, are reconstructed using jet tagging algorithms [Col13], which allow for the identification and measurement of their energies.

In the first step of the PF algorithm, the muons and electrons are reconstructed, and their signals are removed from the measured data in order to isolate charged hadron candidates. This allows the remaining energy depositions in calorimeters to be associated with the traces. If the energy is consistent with the momentum of the reconstructed tracks, their 4-Vector is determined. However, if the energy deposited in calorimeters is significantly higher than expected from the track, it suggests the presence of an additional neutral particle. In this case, the algorithm attempts to reconstruct a supplementary overlapping photon in the ECAL or a neutral hadron in the HCAL. The details of the reconstruction and linking of the PF elements can be found in [Sc+17].

3.3 HGCAL Reconstruction

The HGCAL's reconstruction algorithm's purpose is to process and cluster HGCAL hits, creating physics objects then used by the PF algorithm. Due to overlapping particle showers, reconstruction in HGCAL is a difficult task, especially in the dense environment of the HL-LHC run. The HGCAL's high granularity has the advantage of reducing occupancy and improving energy deposit resolution, but, on the other hand, it inherently brings higher computational demands for reconstruction while keeping very stringent second-level time demands as shown in Figure 3.2.

Particles traversing the calorimeter shatter sub-particles when interacting with the detector layers, as displayed in Figure 3.4. The initial step in the HGCAL data reconstruction process involves uncalibrated energy deposits referred to as recHits (uncalibrated), which provide raw information in the form of 32-bit numbers for each detector channel proportional to the released energy. These values are subsequently calibrated by mapping them to their

[2]Note, that this linking is different from trackster linking, which is an objective of this thesis.

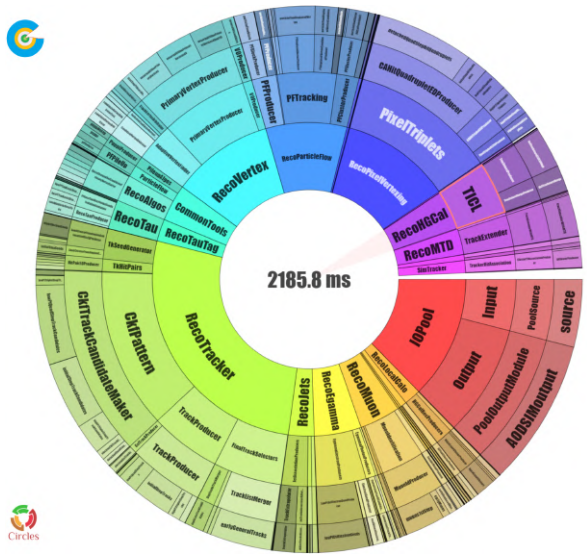
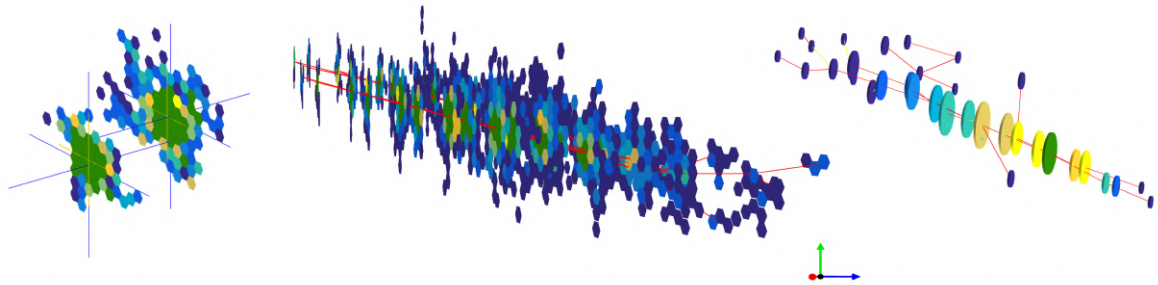


Figure 3.2: Full CMS offline Phase-2 reconstruction running on $t\bar{t}$ (top – anti-top quark) events in 200 PU. The average time of the HGCAL reconstruction in the TICL framework (taking ∼4% of the total reconstruction time of about 55 s) is shown for a single event on a single-core CPU. The currently used trackster linking procedure takes up 1/3 of the TICL pipeline time (883 ms of 2.19 s per a single event).

corresponding positions on the detector's geometry, which is determined by the unique identification id for specific wafers. Once the calibration process is complete, calibrated **recHits** are produced. These are 5D data points, including the spatial coordinates of the cell (x, y, z relative to the center of the detector), energy E, and timing information t. In the context of shower reconstruction, the primary objective of the clustering algorithm is to group individual energy deposits that stem from a particle shower. Due to the detector's high lateral granularity, the number of hits per layer is substantial, making it computationally favorable to assemble hits into 2D clusters on a layer-by-layer basis and subsequently link these 2D clusters across different layers. For that, recHits are inputted into the Iterative CLustering (TICL) framework [PR22], handling HGCAL reconstruction, and are first clustered within their respective detector layers using a density-based algorithm known as CLUE [RCDP+20]. The resulting **layer-clusters** are then connected across different layers to create 3D energy deposits, called **tracksters**, with elongated oval shapes one would expect from particle traversal, using the CLUE3D [PR22] algorithm. In the final stage, TICL outputs a list of particles, along with their identification probabilities and kinematic properties calculated from trackster properties. Similar to the PF algorithm, TICL also incorporates information from other parts of the CMS detector, such as surrounding tracking and timing detectors, to improve the accuracy of the reconstruction.

Specific properties of layer-clusters, tracksters and TICL candidates are outlined in detail in Chapter 5. Apart from that, during event simulations, simulated tracksters are avaliable, forming the ground truth for the reconstruction. In the upcoming sections, I will outline the 2D and 3D analogs of the CLUE algorithm and subsequently provide a concise overview of the modular TICL framework exploiting the above algorithms.



(a) Hits from two consecutive layer-clusters.

(b) Hits from two closeby particles across HGCAL layers.

(c) Trackster graph.

Figure 3.3: TICL calorimetric clustering stages. First, energy deposits are clustered into topological structures called layer-clusters (a); (b) shows reconstructed hits deposited by two close particles indicated with red lines interacting with the calorimeter. LCs created from reconstructed hits are then connected into graph structures referred to as tracksters (c).

3.3.1 Layer-Cluster Formation: the CLUE Algorithm

CLUE, or CLUstering of Energy, is the layer-clustering algorithm, taking in a set of n recHits. To give a sense of scale, events with about 200 pile-up interactions are estimated to produce an order of $\mathcal{O}(10^5)$ recHits per bunch crossing. The algorithm then generates a set of k LCs, typically one order of magnitude smaller than n (i.e., $\mathcal{O}(10^4)$), reducing the problem size for the following reconstruction tasks. CLUE's performance in TICL scales linearly with the number of hits n to be clustered, which prevents a timing or memory explosion in high-occupancy environments. The algorithm's efficiency is improved by optimizing data structures to support fast queries. In contrast to other density-based algorithms, which inherently involve serial processes, CLUE offers a parallelizable computation process.

CLUE is a density-based algorithm that operates under the assumption that high-density areas within a cluster are relatively distant from other high-density areas. The algorithm, illustrated in Figure 3.5, starts with the construction of the fixed-grid spatial index for neighborhood querying, followed by computing a local density ρ, which is defined as a weighted sum of deposited energy in its neighboring cells:

$$\rho_i = \sum_{j \in N_{d_c}(i)} w_j \chi(d_{ij}) \quad , \tag{3.1}$$

with $N_{d_c}(i)$ being the neighbors of the hit i at the cut-off distance d_c, $\chi(d_{ij})$ is the convolutional kernel (either a flat, Gaussian, or exponential function), d_{ij} is the distance between the hits i and j, and w_j the weight of the hit j given by its energy.

In the next step, the distance δ to the nearest hit with a higher local density (*nearest-higher*) is computed for every point. If a hit's local density is greater than a pre-defined cut-off density $\rho \geq \rho_c$ and its δ is higher than the minimum separation requirement δ_c, it is promoted as a *seed* – the core of a cluster. Conversely, cells with a local density less than ρ_c and δ greater than the minimum separation requirement for outliers δ_o, are classified as *outliers*. Outliers are the hits that are far from other clusters and do not have enough density to form clusters on their own. Cells that do not meet the criteria of being a seed or an outlier are considered *followers* and are assigned to their nearest cell with a higher density. Each seed and its followers are then iteratively grouped into separate clusters. Outliers and their descendant followers are excluded from forming clusters, which helps to eliminate low-density deposits,

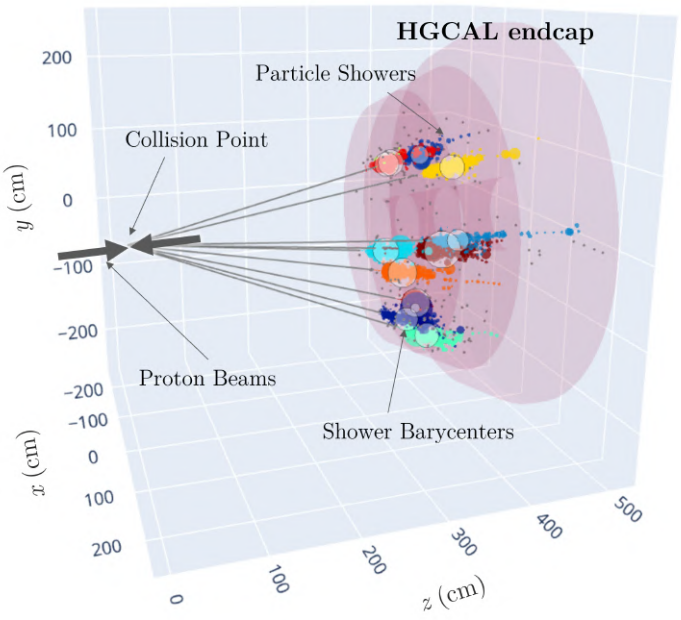


Figure 3.4: Example of a simulated collision event, producing ten particle showers captured in a single HGCAL endcap. The collision of protons at the collision point generates secondary particles that travel towards the endcaps. These particles interact with the calorimeter material, resulting in the production of particle showers.

such as electronic noise.

The query of the d_c-neighborhood is a common operation in density-based clustering algorithms, which tends to be computationally expensive. To optimize this process for CLUE, a fixed-grid spatial index is created for each layer of the calorimeter using a histogram-like data structure. This index registers the indices of 2D points into square bins within the grid based on the 2D coordinates of the points. When CLUE needs to query the d-neighborhood of the hit i, denoted as $N_d(i)$, it only needs to loop over the points within the bins covered by the square window $(x_i \pm d, y_i \pm d)$. Since the value of d is typically small and the granularity of the points is constant, the time complexity of querying $N_d(i)$ using a fixed-grid spatial index is $\mathcal{O}(1)$. In the region of the detector with scintillator tiles, CLUE's implementation alters slightly as it defines and calculates the local density and the nearest neighbor in the $\eta - \phi$ space.

3.3.2 Trackster Formation: the CLUE3D Algorithm

CLUE3D [PR22] is a density-based pattern recognition algorithm, newly introduced into the TICL framework, that shares many similarities with CLUE. CLUE3D operates on layer-clusters generated by CLUE, also considering the longitudinaldimension[3] and clusters them together across layers. The algorithm starts by calculating the local density for each LC on layer j. For this, it searches for all LCs in adjacent layers $j \pm k$, whose projected distance on layer j lies within Δ from LC_i, and adds their energies to compute the local density of LC_i with the use of a kernel function. Next, for each LC, the algorithm finds the nearest LC with a

[3]Using LC's barycenter (z, η, ϕ) spatial position.

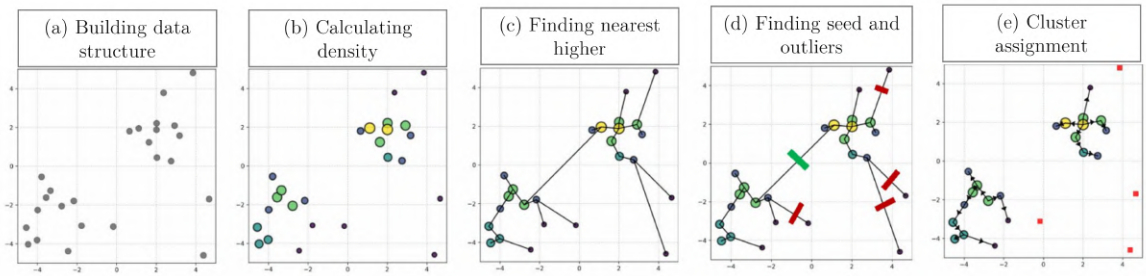


Figure 3.5: Demonstration of the CLUE algorithm. (a) In the first step, the data structure for neighborhood searching is built. (b) The algorithm then calculates the local density ρ, represented by the color and size of each point. (c) The nearest-higher and separation values δ are then calculated for each hit, as indicated by the black lines. (d) Based on the local density and separation values, the algorithm promotes a point as a seed if both ρ and δ are large or demotes it to an outlier if ρ is small and δ is large. Outliers are represented by red squares. (e) Finally, the algorithm propagates cluster indices from seeds through their chains of followers. Picture from [RCDP+20].

higher local density. Based on the local density and the distance to the nearest higher-density LC, the algorithm identifies seeds, followers, and outliers in a similar fashion as CLUE. Finally, it builds 3D clusters, or **tracksters**, by aggregating seeds and their followers. The number of resulting tracksters is typically reduced by another order of magnitude compared with the input LC number and is now in order of $\mathcal{O}(10^3)$.

The tracksters produced by CLUE3D are direct acyclic graphs representing energy flow (Figure 3.3(c)), with directed edges linking LCs to their nearest higher. Because CLUE3D is tuned for a high PU rejection rate, tracksters would often represent only a fragment of a larger shower. However, the primary objective of the reconstruction process is to encapsulate a complete particle shower into a single trackster analogous to the simulated trackster found in the ground truth data explained in Chapter 5. To address the fragmentation issue, a process called **linking**, which is the goal of this work, is employed in the following step to account for disconnections along the various branches of the same particle shower. More rarely, in a high PU environment where overlapping showers occur, tracksters can also result from the merging of multiple overlapping particles. The trackster graph nature allows diverse analysis opportunities, including particle identification and energy regression, shower shape analysis, identification of missing information, and analysis of energy flow. The topology of the graph can be analyzed, enabling the identification of sub-components. By doing so, it becomes possible to react appropriately by either linking or splitting[4] tracksters as needed.

3.3.3 TICL Framework

The Iterative CLustering is a modular HGCAL reconstruction framework being developed within the official CMS software (CMSSW) package (Figure 3.6). TICL is designed with modularity in mind, with each stage of the reconstruction being decoupled to enable independent development by multiple collaborators. This design also allows for easy swapping of the algorithms for comparison and testing. The framework is currently being ported to heterogeneous architectures to enhance computing performance. Aiming to establish a consistent PF framework across the entire detector, the upcoming objective involves expanding

[4]Trackster splitting is not a part of the TICL framework as of this moment.

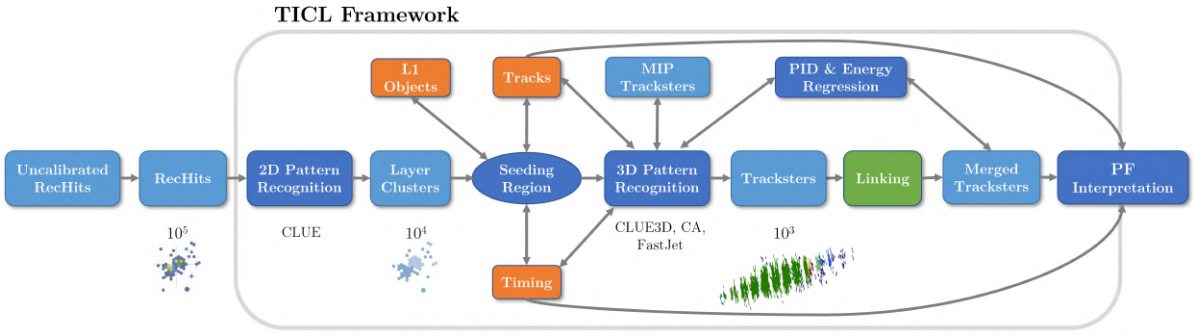


Figure 3.6: Schematic overview of the TICL framework. First, layer-cluster selection is applied to the reconstructed hits from the detector to form 2D clusters in the individual calorimeter layers. Then the pattern recognition algorithm connects the 2D layer-clusters between the layers into candidate tracksters. The TICL data clustering flow reduces the problem size by one order of magnitude at each step. Tracksters are then refined into merged tracksters through the linking step, from which individual particle probabilities and properties are identified. Arrows show connections between different parts of the reconstruction. Tracks and Timing (orange blocks) are information coming from other CMS sub-detectors (tracker and MTD, respectively). The linking step improved in this thesis is represented with a green block.

TICL to encompass the barrel region of the CMS detector.

Pattern Recognition

The Pattern Recognition algorithm in the TICL reconstruction pipeline connects 2D layer-clusters between the layers to form candidate tracksters. As a result of the framework's modularity, there are several algorithms available for trackster construction, such as Cellular Automaton [PR22], FastJet [CSS10], and CLUE3D, which is currently used as a default. In this thesis, the term *tracksters* will refer specifically to the tracksters constructed by CLUE3D. In addition to constructing tracksters, the algorithm consolidates information for each trackster, such as its barycenter[5] position, energy-weighted Principal Components (PCA) determining the shower propagation direction and shape through the three principal axes, transverse momentum, timing information, particle identification probabilities (PID), and regressed energy.

By default, the pattern recognition algorithm operates globally, covering the entire HGCAL region. However, it can be restricted to a smaller volume by confining it to the area defined either using HGCAL internal information (such as layer id, previous iteration or min/max number of hits in a layer-cluster, the momentum of the object) or external information (direction of tracks reconstructed in the tracker, Level-1 trigger objects, timing). This restriction allows targeting individual reconstruction of specific objects and is especially helpful when performing faster regional reconstruction during online event selection by the HLT (as discussed in Section 2.4). To impose these restrictions, the algorithm uses *seeding regions*. A seeding region is defined as a window in the $\eta - \phi$ space on a specific layer, narrowing down the number of LCs contributing to the pattern recognition algorithm based on seed information. The algorithm then applies a pattern recognition algorithm to all available LCs within the targeted seeding region.

The pattern recognition algorithm can therefore be applied iteratively, with each layer-

[5]Spatial coordinates of the energetic center of the object, computed as the energy-weighted average of the detector hits (on the layer-cluster level), or layer-clusters (on the trackster level).

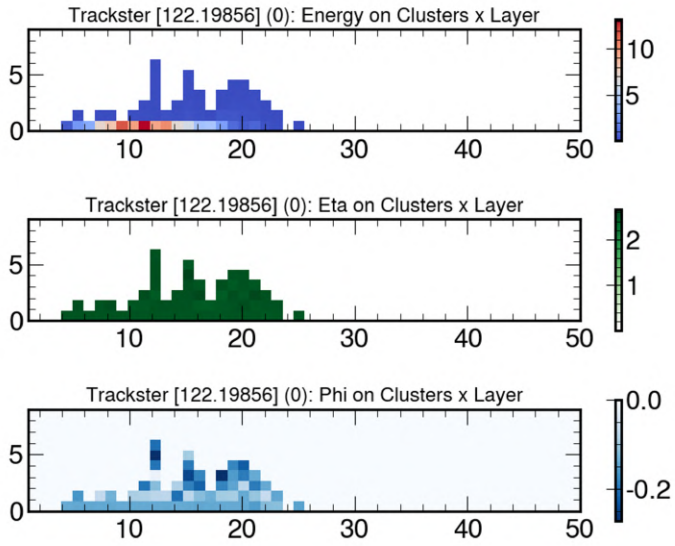


Figure 3.7: The "image" of a 122 GeV photon trackster as an input to the CNN for PID and energy regression. The three "colors" are energy, η and ϕ, shown in separate plots. The horizontal axes represent the HGCAL layers, and the vertical one corresponds to the LCs in the layer, with a maximum of 10 layer-cluster "pixels" per layer [DPCPR20].

cluster assigned a fraction of the total energy of each LC that is yet available for reconstruction. During each iteration, the algorithm reduces the fraction value for LCs associated with the reconstructed trackster. The use of floating-point fraction not only facilitates the sharing of layer-clusters between different TICL iterations but also allows them to be shared between multiple tracksters.

The next stage of the TICL reconstruction involves refining the candidate tracksters and associating them with particle showers, which further enables the identification of individual particle probabilities and properties.

Particle Identification and Energy Regression

The tracksters (both before and after linking) undergo energy regression and particle identification (PID) through a Convolutional Neural Network (CNN) model [DPCPR20]. The CNN model takes an "image" of the tracksters as input, where the first dimension corresponds to the HGCAL layers and the second to the maximum of 10 layer-cluster "pixels" per layer, as shown in Figure 3.7. Each pixel has three properties representing positions η and ϕ, and the energy of the LCs. The model predicts both the particle type and regressed energy using two separate network branches, each with two fully connected layers on top of a shared stack of three convolutions and two fully connected layers. The PID is capable of predicting eight particle ID classes: photon, electron, muon, neutral pion, charged hadron, neutral hadron, and additionally, "ambiguous" (e.g., shower overlaps) and "unknown" types. The last two probabilities are constantly set to zero at this point since the PID for these types is still in development. PID results are used to set the Particle Data Group ID (PDG ID) of the reconstructed tracksters. The PDG ID system, which is widely used in particle physics, including in Monte Carlo event generators (Section 5.1), assigns a unique identifier to each type of particle, including codes for all known elementary particles, composite particles, and atomic nuclei, as well as hypothetical particles beyond the SM.

These data, along with collections of tracksters and tracks, are used as input for the linking module responsible for clustering miss-connected tracksters generated by the pattern recognition algorithm.

Geometric Iterative Linking

A geometric iterative linking algorithm [Nan22] has been recently implemented in CMSSW and is currently the only linking algorithm in use. It performs linking by projecting tracks and tracksters onto a common surface and establishing geometric links between them by verifying their compatibility in terms of energy and time. This approach is based on the assumption that physics objects belonging to the same particle will be geometrically close after projecting them onto a common surface. Although not perfect, this provides a reasonable approximation for a large number of tracksters produced in collision events and allows for simplifying the problem by reducing its size to two dimensions.

Linking is applied on CLUE3D tracksters, producing a new collection of merged tracksters. At the first HGCAL layer, tracks of sufficient energy (above 2 GeV) are propagated to the first and the last CE-E layers while masking muon tracks. The projection of trackster barycenters is then done via linear extrapolation back to the CMS vertex. The trackster-to-track linking procedure is then performed at the first or the last CE-E layer for charged candidates, and trackster-to-trackster linking for neutral candidates at the last CE-E layer. After the projections, the link-finding procedure is executed, performing a $\eta - \phi$ search for tracksters around the propagated seeding tracks or tracksters. Each entity discovered within the defined ΔR range is then referred to as a "link", representing a potential association between two objects. Tracks that do not have linked tracksters are directly promoted to charged candidates. For neutral candidates, the process is slightly different. Track linking is not performed, but tracksters are still propagated to the first and last CE-E HGCAL layers, and geometric compatibility between tracksters at the last CE-E layer is checked. Tracksters without links are directly promoted to neutral candidates. During the iterative construction of the final objects (`TICLCandidates`), any unfeasible links are removed, aided by considerations of energy and time compatibility.

Trackster Property Aggregation

Once the `TICLCandidates` are constructed through the linking algorithm, trackster property aggregation closes the HGCAL reconstruction pipeline. The same properties as for the original tracksters are re-calculated given the information of individual constituent LCs. The merged trackster is passed through an energy regression and PID step and, based on PID probabilities, is categorized as hadronic if the total probabilities of being an electron or a photon are less than fifty percent. If marked as hadronic, charged candidates are denoted as pions $\pi^{\pm}$ and neutrals as kaons K^0. If not hadronic, candidates are classified as electrons, positrons $e^{\pm}$, or photons γ, depending on their charge. The charge is determined from the bend of the associated track. For charged candidates, their Lorentz vector is established using the track momentum and the regressed energy of the corresponding merged trackster. In the case of neutrals, the combined barycenter of the accumulated tracksters is utilized as the direction, along with the regressed energy of the merged tracksters.

As a result, TICL produces a full reconstruction of the physics objects in the HGCAL. This work focuses on enhancing the linking process of the TICL pipeline, aiming to provide a more efficient linking solution compared to the existing conservative geometric linking method, as described in the following Chapter 4.

Chapter 4

Trackster Linking

The 3D clustering step of TICL is often challenging, especially when dealing with non-aligned showers, tainted by noise, or affected by other overlapping physics objects. To address this issue, this thesis introduces a new linking algorithm based on Graph Neural Networks (GNN) that takes a set of potentially fragmented tracksters as input and produces better-formed tracksters that represent particle showers more extensively. This process improves the energy resolution of particle shower reconstruction by minimizing the absolute difference between the simulated and reconstructed shower energies. In this chapter, the thesis first delves into the problem of trackster linking and explore the various challenges that it presents. After that, it provides an overview of the related work.

4.1 Motivation

Causes of Trackster Fragmentation. Multiple challenging physics processes result in the creation of particle shower secondary components, making it difficult for the pattern recognition algorithm to correctly merge LCs into complete 3D structures since the CLUE3D algorithm specializes in single-blob aligned shower clustering, overlooking misaligned trackster formations. For instance, in the hadronic section of the detector, intermediate Minimum Ionizing Particles (MIP) may be produced, leading to trackster splitting. Similarly, for electrons in the EM section of the detector, showers initiated by the bremsstrahlung[1] can lead to the formation of unwanted separate tracksters. Additionally, particles may interact before even entering the calorimeter, also leading to clusters that need to be linked together. Another difficult form of fragmentation may occur when a secondary particle formed by an interaction follows a deviating track. The different material compositions and sensor types used in the electromagnetic and hadronic compartments of the HGCAL result in varying energy densities, causing tracksters coming from hadronic showers to be split at the boundary between the compartments. As such, simulated trackster B in Figure 4.1 demonstrates a strongly-interacting particle initiating a shower in the electromagnetic section of the calorimeter, located in the front part of the HGCAL (first 26 layers), then stopping interacting and subsequently triggering the creation of a second shower in the hadronic region of the calorimeter.

[1] Radiation emitted by charged particles as they are decelerated when passing through the electric field of an atom or nucleus.

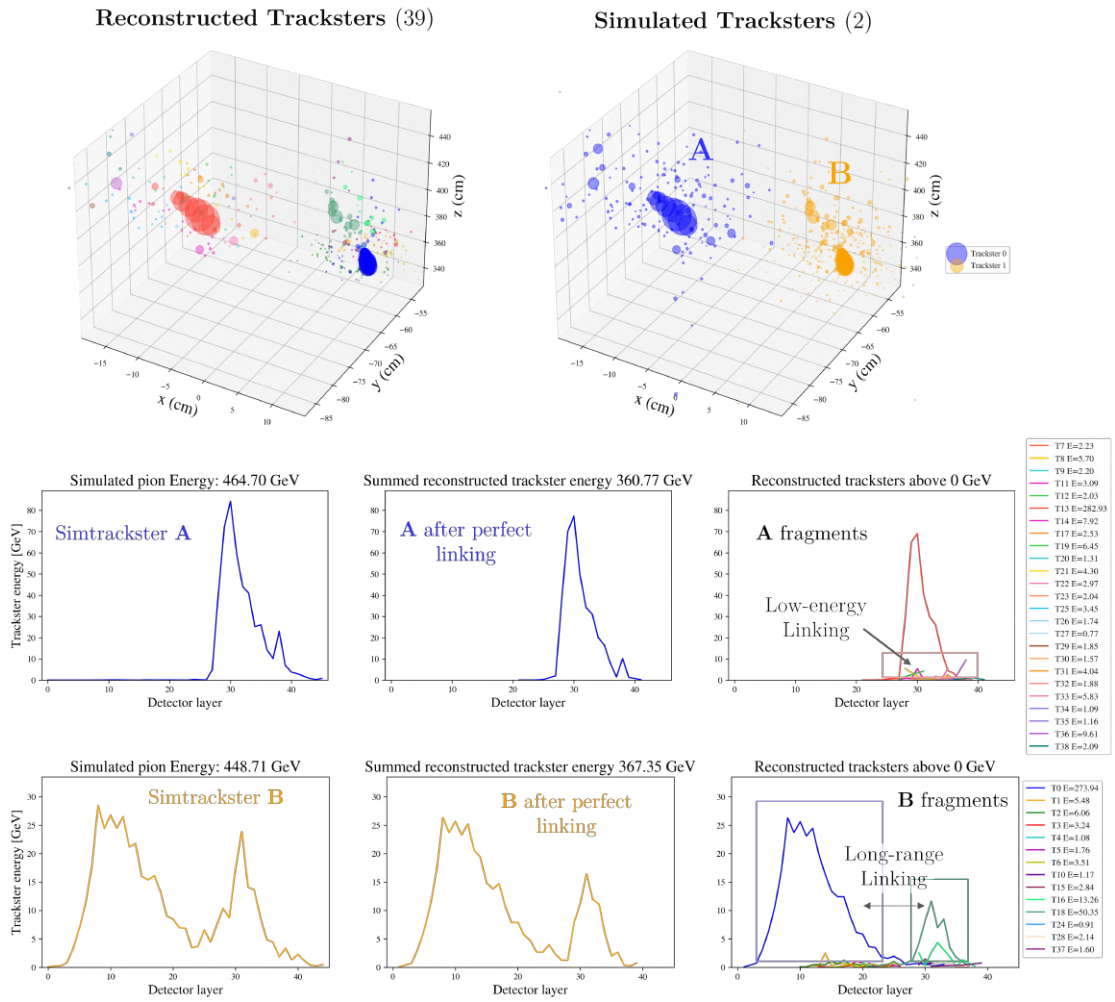


Figure 4.1: The figure showcases a double closeby pion event reconstruction in 0 PU. The top row of the figure displays a 3D visualization of the reconstructed and simulated particle showers. The two consecutive rows correspond to longitudinal shower energy profiles of the individual simulated tracksters (A and B), marked with the same colors as in the 3D plots. In the energy profiles plots, the left-most picture shows the simulated energy profile, the middle picture depicts the profile reconstructed through perfect linking, and the right-most picture displays the individual trackster before linking. The total simulated tracksters' energy is 913.4 GeV, whereas the total reconstructed energy is 728.1 GeV (79.71%). The number of matched tracksters is 25 for simulated trackster A and 14 for B.

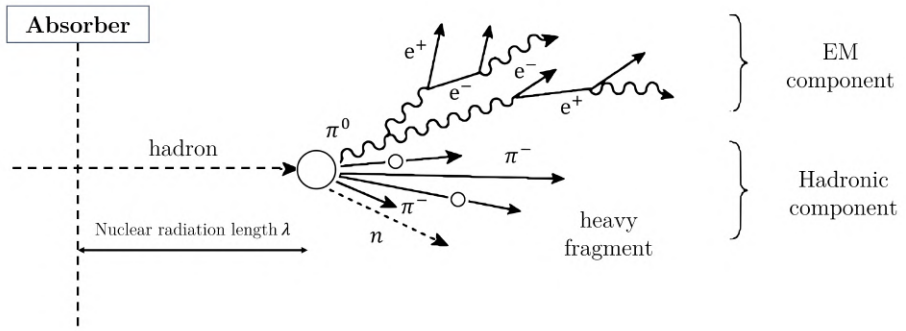


Figure 4.2: Schematic diagram of a hadronic shower development initiated by a π_0 meson, resulting in both hadronic and EM components. Re-drawn based on [LLW18].

The development of hadronic showers is also rather complex in itself. Figure 4.2 shows a schematic representation of such a shower, which begins with the hard collision of an incident hadron with a nucleus. The shower consists of a narrow core of electromagnetic components caused by photons from π_0/η_0 mesons, with a surrounding halo that is dominated by charged hadrons. Additionally, the ratio of electromagnetic and hadronic portions can vary significantly from event to event. The response of the HGCAL detector to electromagnetic particles is different from that to hadronic particles, leading to a difference in their energy resolution. This means HGCAL is not able to provide an equal response to EM and hadronic particles, and is thus considered a non-compensating calorimeter. The hadronic shower also includes invisible energy from binding energy, nuclear recoil, neutrinos, and late components, in addition to the visible energy carried by particles in the shower. All these factors contribute to a limited hadronic energy resolution and the creation of multiple trackster components.

Apart from challenging physics processes, numerous simultaneous collisions within the detector, forming a high PU environment, impose a strict limit on the clustering sensitivity to prevent the formation of a single cluster encompassing multiple particle showers. CLUE3D is therefore tuned for PU, unequivocally leading to some tracksters being split into multiple pure (homogeneous) yet low-efficiency (incomplete) trackster candidates.

Geometric Linking Baseline. While a rather conservative geometric iterative linking described in Section 3.3.3 does a reasonable job re-connecting single particle fragments in 0 PU, as shown in Figure 4.3, with just a small efficiency drop at low energies, the pile-up and multiple particles have a big impact on the reconstruction efficiency of the hadronic showers. This drop in performance is especially evident in the high η regions, where the detector occupancy is the highest [WR].

The above considerations necessitate exploring limitations of the geometric linking and breaching these limitations with investigating alternative approaches to trackster linking. Apart from improving event interpretation, such linking also aims to reduce the computational complexity driven by the large number of tracksters even in simple events with no pile-up. As such, an average number of tracksters produced by two close-by pion showers in 0 PU, with energies ranging from 10 to 600 GeV, reaches 28 on average, as will be further shown in Table 5.1. Furthermore, correct trackster linking is important to ensure the matching of the reconstructed energy deposits in the calorimeter to the objects reconstructed elsewhere in the detector, such as tracks, preventing double counting of energy.

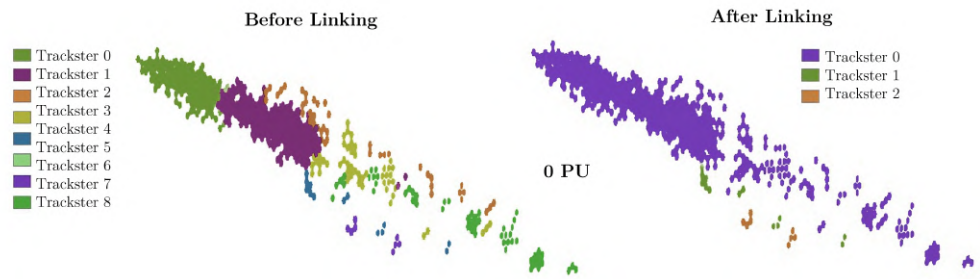


Figure 4.3: The image displays the tracksters reconstructed from a solitary simulated charged pion generated at the CMS vertex. On the left-hand side, the tracksters created by CLUE3D are shown in different colors on individual hit-level. The merged tracksters collection obtained by applying the geometric linking procedure to the CLUE3D tracksters is shown on the right-hand side. The linking has combined several tracksters into a larger single trackster (shown in violet), while some small tracksters remain unmerged due to their misalignment. In an ideal scenario with a single particle, we anticipate seeing only a single trackster after linking. The visualization was created using the Fireworks tool [Nan22].

4.2 Challenges

The trackster fragment re-connection algorithm faces several challenges, including linking tracksters over long distances (as further shown in Figure 5.8), particularly in high PU with numerous overlapping particle showers, even such that some hits are only fractionally assigned to a certain reconstructed object. Showers to be re-connected might have irregular shapes. Apart from that, an incorrect connection of non-matching tracksters implies a loss of information. Finally, ensuring compatibility of the merged tracksters with tracks is also a non-trivial aspect. Failure to match energy deposits from a charged particle with the corresponding track leads to the creation of fake neutrals and double counting of energy.

4.3 Problem Definition

The task of accumulating hits belonging to the same particle is traditionally framed as a 3D point clustering problem. The TICL framework already clusters hits in 2D, creating layer-clusters, and in 3D, generating trackster fragments. By using this iterative clustering, instead of handling large numbers of detector hits, it is now possible to focus on high-level objects characterized by sets of features, such as trackster spatial properties, energies, graph-based features, and more. Then, the linking problem can also be defined as a clustering task, but on the trackster level. Given individual trackster fragment features, the goal of linking is to find an assignment in which tracksters coming from the same particle (simulated trackster in the simulation environment) are connected, while the ones coming from different showers are kept separate.

4.3.1 Data

Due to the sparsity of detector data, each collision event can be represented with a point cloud, with each point being a trackster. This approach has the advantage of avoiding the

need to impose an order on the data or store it in a grid-like structure, which could result in information loss or prove challenging given the heterogeneous geometry of the detector. In addition to being represented as point clouds, the data can be naturally represented with graphs in a sense of a particle shower interaction tree given the knowledge of the detector geometry, as described in Section 5.4. A graph is made up on top of the point cloud with geometrically built edges that represent potential relationships between the nodes. Graphs are a particularly appealing way of expressing data from particle detectors as they allow us to easily employ graph-based techniques for linking, such as graph neural networks.

In this study, I use point cloud/graph datasets with varying levels of complexity and rate of particle shower overlap, explained in Chapter 5. In general, two problem settings are taken into account. The simpler cases are single and multi-particle datasets with no additional interactions (0 PU), for which all layer-cluster ground truth assignments are known. Second, I consider datasets that have additional PU interactions, for which the ground truth is only available for the hard-scattering particle showers in the event.

4.4 Related Work

This section presents an overview of the related literature for particle shower reconstruction and calorimetric clustering tasks, general clustering approaches for non-physical point clouds, and similar attempts at applying ML to calorimetric data as the ones presented in this thesis.

4.4.1 Unsupervised Clustering Methods

First, I cover traditional unsupervised clustering methods that are directly applicable to point clouds without the requirement for a graph structure. It should be emphasized that the number of clusters in the particle shower reconstruction is unknown in advance, which restricts the direct use of techniques like K-means [Mac67] and Gaussian Mixture Models (GMMs) [Rey09] that call for pre-specifying the number of clusters. Multiple runs of these algorithms can still be used to estimate the appropriate number of clusters, but this increases their time complexity. As a result, we opt to deal with distribution, density, and hierarchy-based algorithms rather than centroid-based methods. Several of the described methods were utilized as baselines for comparing the GNN linking performance (Section 7.3).

Centroid-based clustering. Centroid-based clustering uses iterative algorithms initializing K cluster centroids as points in the data's feature space. Each data point is assigned to the centroid based on a certain criterion. The centroids are then updated with respect to all data points in the cluster until convergence. Popular methods include K-means for convex clusters, Gaussian Mixture Models (GMMs), Mean Shift (MS) [CP15], and Affinity Propagation (AP) [MQY23]. GMMs assume that the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters. MS is a non-parametric method finding centroids by iteratively shifting them towards the local maxima of the density function without the need to pre-specify the number of clusters beforehand. AP iteratively sends messages between data points to identify representative points (exemplars) and assigns data points to clusters based on proximity to them. Each data point sends messages to update its beliefs about which point should serve as its exemplar. Exemplars are selected if they

are sufficiently similar to many other samples and if they have been chosen by a significant number of samples to be representative of themselves. AP also automatically determines the number of clusters based on the input data.

Hierarchical clustering. Hierarchical clustering iterative algorithms can be broadly classified into two types: agglomerative and divisive. Agglomerative clustering starts with each data point as a separate cluster and iteratively merges the closest clusters until a stopping criterion is met. Divisive clustering, on the other hand, starts with all data points as a single cluster and iteratively divides the clusters into smaller clusters. The proximity between clusters can be evaluated with various metrics such as single linkage (distance between the closest cluster data points), complete linkage (distance between their farthest points in the clusters), average linkage (average distance between all pairs of data points from both clusters) and Ward's method [ML11] (minimizes the variance of clusters), etc.

Density-based clustering. Density-based clustering determines clusters based on the local density of data points, rather than relying on the centroids. This approach allows identifying clusters with irregular shapes and varying densities, making them useful in applications like HGCAL reconstruction. The most commonly used density-based clustering algorithm is DBSCAN [SSE+17], which works much like CLUE (Section 3.3.1). OPTICS [ABKS99] is a generalization of DBSCAN, constructing a hierarchical ordering of points in the dataset based on reachability distance, while HDBSCAN [MHA17] uses a hierarchical approach to automatically determine the optimal number of clusters and appropriate neighborhood radius values for each cluster by performing DBSCAN over varying distance parameter values.

4.4.2 Supervised Machine Learning Techniques

The classic rule-based algorithms utilized in high energy physics (HEP) are heavily reliant on the factorization of individual steps and extensive domain-specific knowledge. Their performance, however, is restricted by the ideal-case assumptions used in their development. As a result, ML-based approaches, refining the classical reconstruction, are progressively incorporated into the reconstruction pipelines. The use of ML approaches has been fundamental to event processing in HEP studies. For example, NN models have been essential in tasks like particle shower reconstruction by clustering [JFC+20], energy regression [CKP+17, QG20, Val22] in calorimeters and tracking devices, particle identification via classification [CKP+17, Val22, AIK20], and jet tagging [Col17a, BGS22, CKSS15, QG20, BKV+20]. The effectiveness of ML approaches is largely owed to the availability of precise simulation of the detector components and physics processes, allowing for the creation of vast amounts of labeled data. However, deep learning techniques face a limitation in accurately estimating their uncertainties and have a tendency to rely on non-physical characteristics present in the training data to achieve high performance.

Motivation for ML techniques in HEP. The utilization of machine-learning methods presents a distinct advantage over traditional algorithms since ML models are designed to be automatically optimizable and require the definition of a loss function to train, providing a quantifiable performance metric. Conversely, classic HEP algorithms often lack a comprehensive quantitative measure and rely on manual parameter tuning, requiring significant expertise and manpower. Furthermore, ML algorithms are highly parallelizable, making them

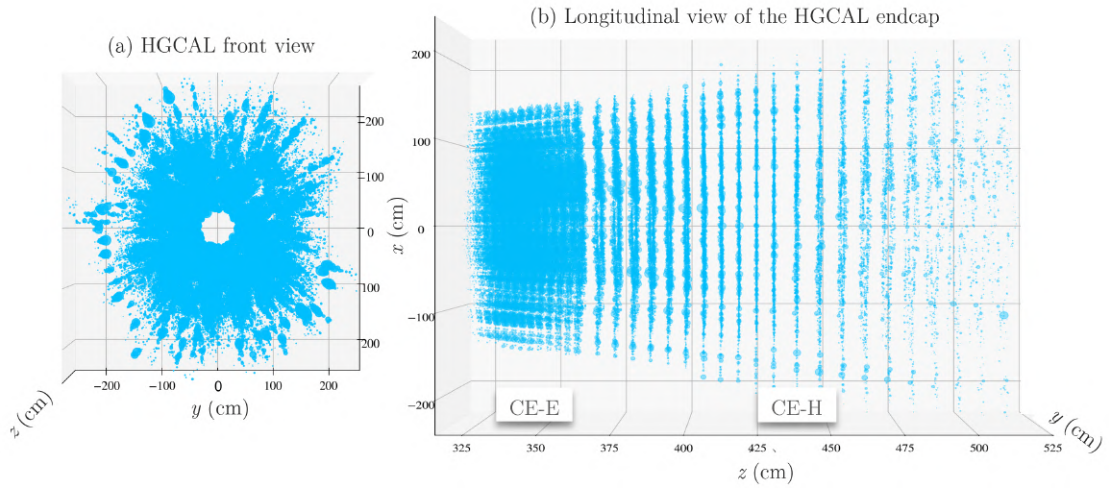


Figure 4.4: Demonstration of the HGCAL detector irregularity through a combination of 100 events from the `MultiparticleOPU` dataset (see Chapter 5) to outline the detector's geometry. Each point corresponds to a layer-cluster with a size proportional to its respective energy. (a) depicts the front view of the data captured by the detector, with no data captured in the center, corresponding to the LHC beampipe. The longitudinal view (b) reveals the irregular geometry of the HGCAL endcap, with two well-separated parts clearly visible: the dense and compact electromagnetic part CE-E, and the hadronic part CE-H, with larger gaps between the layers and thicker absorber plates. As the distance from the first layer increases, the density of the captured energy decreases.

well-suited for execution on specialized hardware such as graphics processing units (GPUs) or field programmable gate arrays (FPGAs) [DHH+18]. Apart from that, NNs provide the added benefit of abstracting away from the specific geometry of the detector, which can be irregular, such as in the case of the HGCAL. A notable source of inspiration for this work's comparison and model designs is the study of CNN and GNN applications on detectors with irregular shapes in [QKIP19] and [DV22].

ML challenges for HEP data. The integration of ML algorithms into HEP pipelines, on the other hand, faces two major challenges: the heterogeneous nature of the detector data and the requirement to formulate the problem as a minimization task. The detector data is highly irregular, stemming from the presence of multiple sub-detectors, each with its unique geometry. Within a sub-detector, such as the HGCAL, the geometry is engineered in accordance with physics considerations, exhibiting a high-resolution close to the interaction point and a coarser resolution further away. Furthermore, the detector layers are not densely packed, featuring ample interstitial space (Figure 4.4). Consequently, neural networks that necessitate regular grid structures, such as Convolutional Neural Network (CNN) architectures [KSH+12], despite their exceptional performance and highly optimized implementations, are impractical for direct use, as they require the representation of particle detectors as arrays of sensors with a regular structure.

Another hurdle in deploying NNs for particle reconstruction entails training the network to anticipate a variable number of particles from an unknown number of inputs. Although there are numerous algorithms and training techniques for object detection in dense data, such as images, the majority of them still require the presence of well-defined boundaries or a certain level of density which facilitate the exploitation of anchor boxes or strategic points in the objects being detected. However, particle interactions in a detector typically overlap

substantially, and their sparsity makes it difficult to determine central points or distinct boundaries. In order to solve this problem, the Object Condensation approach [Kie20] was put forth building on the developments in GravNet [QKIP19] architecture. It condenses object properties into condensation points, which can be selected by the network based on a high confidence score. This study could contribute to this approach by aggregating the information via condensation from the underlying layer-cluster structures to be used as additional features during the higher-level trackster linking procedure. However, this technique is not explored here.

Adaptation of HEP data for CNNs. One way to adapt data for use in convolutional neural networks (CNNs) is to transform it into regular grid structures either by 3D voxelization [CMW+17] or 2D bird's eye view (BEV) projections [LVC+19]. Regular structures make it possible to address the problems caused by sparse and variable-size representations and use well-researched 2D or 3D image techniques, such as CNNs, to extract the point-cloud local features. However, much like sampling, this transformation results in a loss of detail and resolution due to the fixed grid sizes. As a result, the precise regional feature representations required for successful clustering are typically missing from the feature maps at high-level CNN layers. Using grid-based methods, therefore, seems inappropriate given the HGCAL's high granularity and the fact that particle showers cannot be accurately represented at lower resolutions.

Point cloud methods. Apart from grid-based methods, a complementary approach for point cloud processing is applying neural networks directly to point clouds, such as in the case of PointNet [QSMG17]. The PointNet learns spatial embeddings of the data points and afterward aggregates individual point features to a global point cloud signature without capturing local structure. It is used for object classification and segmentation using semantic labels. The success of convolutional architectures, however, has shown that local structures can be effectively utilized for these tasks. Therefore, in their subsequent work, the authors of PointNet proposed to use PointNet recursively on a nested partitioning of the input point set in a hierarchical fashion to allow capturing local features, resulting in the PointNet++ architecture [QYSG17]. Yet, unlike computer vision tasks, where semantic labels are frequently used to characterize object classes or attributes, HEP data frequently lacks such labels for the shower components, making these approaches less straightforward for our task.

GNNs. Networks capable of learning geometry are particularly compelling for irregular detectors. Such networks are Graph Neural Networks, built to function on graph data made up of features and pairwise connections of elements, which is an alternative approach to incorporating weight sharing, local connectivity, and specialized domain knowledge. GNNs were first introduced in a paper by [SGT+08] in 2008, and since then, these models have been utilized in various fields such as social networks, knowledge graphs, recommender systems, and 3D shape analysis, as discussed in an extensive review by Jie Zhou et al. [ZCH+20]. Unlike CNNs, Graph Networks can learn the suitable representation of physics objects without imposing constraints on the geometry of the detector. Additionally, no detector geometry-imposed data pre-processing is necessary. Although this method is promising, its drawback, if used as initially proposed, is the requirement to connect every vertex to every other vertex, making memory and computation demand prohibitively high for large graphs. This issue is resolved by defining only a portion of connections between neighbors in a representation of learnable space, where each vertex's features are updated based on this limited number of

neighboring nodes. These neighborhood graphs could be either static or inferred from the input data rather than imposed during pre-processing, which also makes the adjacency matrix defining the relationships between the input data trainable.

One relevant paper for this study is the Dynamic Graph CNN (DGCNN) [WSL+19], which introduced the EdgeConv layer for computing data point features by aggregating edge features of k-Nearest Neighbors (k-NN) of each point. The key idea behind DGCNN is to learn a dynamic graph structure from the input data, used to perform graph convolutions. This contrasts traditional graph convolutional networks requiring a fixed graph structure as input. Unfortunately, this network needs a lot of computational power to dynamically calculate neighbor associations in high-dimensional space, making it impossible to utilize in the linking task as it is. The EdgeConv-like approach is adopted in this thesis for message-passing, with further details on EdgeConv given in Section 6.1.1. However, in contrast to the DGCNN, no dynamic graph update is used; instead, a static graph enabling a single pre-calculation of the related nodes is utilized for the reasons discussed in Chapter 6. Also, unlike DGCNN, which selects a pre-defined number of neighbors, in this work, I employ every connection existing in the pre-computed graph. This design choice is the result of two major factors. To start with, in high-dimensional embedding space, k-NN is a slow procedure. Secondly, the number of neighbors to be chosen is heavily influenced by the event's amount of pile-up.

GNNs in HEP. Several GNN models have been explored for refining particle reconstruction in HEP, such as GravNet and GarNet [QKIP19], ParticleNet [QG20], and GNN by Ju et al. [JFC+20], all of which demonstrate the applicability of message-passing GNN architectures to the similar task as ours. GravNet is applied to the particle reconstruction task starting from hits. It bases on the DGCNN approach with the main goal of improving resource-demanding high-dimensional k-NN search problem by separately learning the feature embeddings and the low-dimensional coordinate space for neighborhood aggregation. By doing so, this architecture aims to keep a trainable space representation at minimal computational costs and achieves almost a factor of 10 faster inference times compared to the DGCNN model. ParticleNet also makes extensive use of EdgeConv operations and adopts the dynamic graph update methodology. It is applied to the problem of jet tagging, which is the identification of the quarks initiating a collimated spray of particles (jets). However, in contrast with the DGCNN, ParticleNet makes a number of distinct design choices to better suit the jet tagging objective, including the appropriate number of neighbors tuning, configuring the MLP in EdgeConv, incorporating skip connections across the layers and modifying the pooling techniques. However, due to the substantial dimensionality of the data in high pile-up scenarios, the aforementioned architectures are still inadequate for direct particle reconstruction from recHits in terms of required time complexity. To address this problem in this thesis, I use higher-level energy structures (tracksters) as the inputs to the graph network, reducing the number of network inputs by two orders of magnitude and making the inference time more appropriate.

Graph Attention. The success of attention-based models in natural language processing (NLP) has inspired researchers to investigate their graph-based implementations. In particular, Veličković et al. introduced the graph attention network (GAT) [VCC+17], using attention mechanisms to aggregate information from nodes' neighborhoods, allowing the model to selectively focus on relevant information from the graph, rather than processing all the information equally. The presence of incorrect edges in event graphs built for the linking task motivates the incorporation of an attention mechanism in the linking problem, as discussed

in Chapter 6. Additionally, variable spatio-temporal separation of the tracksters allows easy calculation of the additional hand-crafted similarity features between the tracksters.

Other NN Architectures. Apart from CNNs and GNNs, different HEP-related studies investigate recurrent networks [FCM+18] and transformers [QLQ22]. Qu et al., for example, reported cutting-edge performance of the Particle Transformer (ParT) [QLQ22] on the same problem explored in their prior ParticleNet work on jet tagging. The results reveal that ParT outperforms ParticleNet by a slight margin on all datasets tested. The authors believe that the efficacy of ParT stems mostly from augmented self-attention, which incorporates physics-inspired paired interactions as well as machine-learned dot-product attention.

Summary. Overall, the utilization of GNNs holds great potential for overcoming the challenges posed by the irregular data structure of the detector, enabling abstraction from its intricate geometry. On a negative note, it may require additional processing to create graphs from the initial point cloud data, and requires more computational resources compared to architectures that do not employ neighborhood aggregation.

Chapter 5

Event Simulation and Datasets Generation

A significant amount of data is typically required for training neural networks due to their inherent ability to learn complex patterns and generalize from examples. Since the construction of the HGCAL detector is yet to be completed, it is necessary to use highly detailed collision simulations based on the actual detector geometry for improving HGCAL reconstruction algorithms. Such simulations are beneficial for detector behavior comprehension, subsequently helping in their design, allowing to define their operational requirements, and comparing the obtained experimental data with the theoretical predictions. In experimental particle physics, Monte Carlo methods [CC75] are commonly employed for such simulations. The CMSSW facilitates sophisticated simulation tools and provides a wide range of configuration options for running both simulation and reconstruction within individual CMS sub-detectors. This chapter outlines the process of event simulation in CMSSW and subsequent dataset generation for the task of trackster linking. I present the three generated datasets for network training and physics performance evaluation used in this work, as well as the graph-building process involved in dataset creation.

The production of the samples discussed in this section was done using CMSSW pre-release `CMSSW_12_6_0_pre3` with the `TICL_v4` version of CLUE3D. The same release was used for processing the simulated samples. The geometry used to simulate the detector is `V16` (`D99`).

5.1 Event Simulation

The simulation process of the particle passage through matter involves three main stages described below: physics process generation, detector simulation, and digitization.

Physics Process Generation. The first step in processing Monte Carlo events is generating sets of outgoing particles resulting from the interactions between incoming particles according to the quantum field theory (QFT). The CMSSW software provides interfaces to various physics event generators, with the `PYTHIA` [SAC+15] generator being the most widely used in production thus far. The event generation starts with initiating the generator with a configuration of the key descriptions of events to be generated, such as the decay, beam parameters, and the parton distribution functions (PDFs)[1]. Once the event generation stage

[1] PDF gives the probability of finding partons (quarks and gluons) in a hadron as a function of the fraction of the proton's momentum carried by the parton.

is complete, the generator produces a set of final-state particles for each event according to the chosen QFT model. The output particles are then ready for hadronization, particle shower production, and decay of unstable particles, which are achieved through the `PYTHIA` software.

Detector Simulation. In the detector simulation stage, `GEANT4` (for GEometry ANd Tracking) [AA+03] is used to simulate the energy depositions of the produced particles in the CMS detector based on the physics of the interaction between the particles and the materials. For that, a detailed model of the detector is created, including information about the full detector geometry, its materials, the effect of the magnetic field, etc. After this step, the pre-computed pile-up interactions are added to the events according to the instantaneous luminosity and expected average pile-up.

Digitization. Next, in the digitization stage, the detector readout electronics response to passing particles is modeled. As a result, digitized signals that closely resemble the real CMS detector signals are produced. These signals are passed into the offline reconstruction with the same triggering procedures and reconstruction algorithms used for the experimental data captured by the detector. This ensures the consistency of the simulation.

Particle Guns. CMSSW supports the production of particle showers in HGCAL using a set of the `Particle Gun` generator modules. Particle guns have the ability to produce one or multiple particles, identified by their PDG IDs, by default originating from a common vertex located at $(0, 0, 0)$ with properties uniformly distributed within a user-specified range. Additionally, each particle can be accompanied by its corresponding anti-particle or another particle of the same PID, with the opposite momentum vector. This study utilized the `CloseByParticleGun` and `FlatRandomPtGun` for sample production. `CloseByParticleGun` allows for the generation of particles at any location in the CMS, as well as configuring the $\eta - \phi$ particle ranges, particle separation ΔR, energy, number of particles, overlapping, particle IDs (i.e., electrons and positrons with PDG ID ± 11, photons ID 22, pions $\pi^{\pm}$ ID ± 211 or kaons K^0 ID 130), their pointing, distances from a beamline, and z range. The particles are then generated randomly within the provided feature ranges. Closeby gun mode is capable of creating several vertices pointing back to the CMS vertex $(0, 0, 0)$. Then, if more than one particle is requested, each will be created at a different vertex, uniformly spaced in the ΔR range, which is the arc-distance between two consecutive vertices over the circle of radius R.

Pile-Up Generation. In the scenario involving pile-up generation, the particles from the hard scattering vertices are initially created without additional pile-up. Afterward, the pre-computed PU interactions are introduced in a random fashion, using a Poisson distribution representing the expected number of interactions per event. This simulates the effect of additional lower-energy interactions in the detector, which can interfere with the primary particles of interest.

HGCAL simulation. At CERN experiments, the interesting particles are often rare and highly unstable, quickly decaying into other elementary particles. Therefore, the focus of experiments is on reconstructing well-known fundamental particles, which can be involved in other, more complex interactions. As a part of the HGCAL reconstruction, TICL applied to the simulated detector data produces the structures called `RecoTracksters`, reconstructed by the framework. Along with recotracksters, structures referred to as `SimTracksters` are

produced. Those are the simulated tracksters built by clustering all the simulated hits representing a single complete particle shower. This study uses simtracksters as the ground truth for training and to measure the performance of the reconstruction.

5.2 Generated Linking Datasets

Simulated data offers the advantage of generating large volumes of data with relative ease. In this study, data was generated through simulation using the most recent geometric layout of the HGCAL, as described previously. To investigate the HGCAL linking problem, we choose to generate three datasets ranging from simpler (`CloseByPions0PU`) to more complex scenarios (`Multiparticle0PU`), as well as the scenario similar to one expected to occur during the actual run of the HL-LHC (`SingleParticle140PU`). The TICL pipeline, excluding the geometric linking step, processes the generated events in all three datasets, resulting in the production of fragmented tracksters after the CLUE3D step. Additionally, candidate tracksters are generated by running the complete TICL pipeline using the default geometric linking algorithm (Section 3.3.3) for comparative analysis. Detailed information about all the datasets can be found in Table 5.1, while examples of events for each dataset are illustrated in Figure 5.1.

All three datasets primarily focus on hadronic interactions, which present the most difficult case for linking, as discussed in Chapter 4. For this purpose, majority[2] of the particles in the datasets are pions (PDG ID 211), preferred for this task as they undergo hadronic interactions, resulting in more trackster fragments compared to particles interacting electromagnetically.

Closeby Double Pions Dataset. Objectively the easiest of the selected scenarios is the `CloseBy Pions0PU` dataset containing two closely located pions with a possibility of overlap and no additional pile-up. This dataset aims to enhance our understanding of particle shower properties and allows us to investigate the linking challenges regarding underlying graph construction. Moreover, it enables to scrutinize the shower fragmentation patterns and is used as a simple scenario for tuning the proposed linking algorithms.

For this scenario, two pions are fired towards the same region of the detector, with energies uniformly distributed between 10 GeV and 600 GeV, maximum separation of $\Delta R = 15$ cm, and η uniformly distributed between 1.7 and 2.7, covering almost the whole HGCAL detector. The particles in this and `Multiparticle0PU` dataset are produced with the `CloseByParticleGun` generator right before the HGCAL's first layer, allowing no prior interactions before HGCAL. While the particles do not originate at the origin, they still point to the CMS vertex.

Multiparticle Dataset. Following that, we progress to a more intricate `Multiparticle0PU` dataset involving multiple particles of varying types (see Table 5.1) with higher overlaps. As the number of particles increases, the level of overlap intensifies (mostly the low-energy tracksters around the main trackster), and the surrounding neighborhood of high-energy tracksters encompasses a greater number of tracksters generated by other particle showers, making the dataset more complex for the linking task than the one considered in the previous scenario. Additionally, this dataset serves as a means of testing NN's ability to generalize to different particle types.

[2]Or all of the particles, such as in the case of double pion dataset.

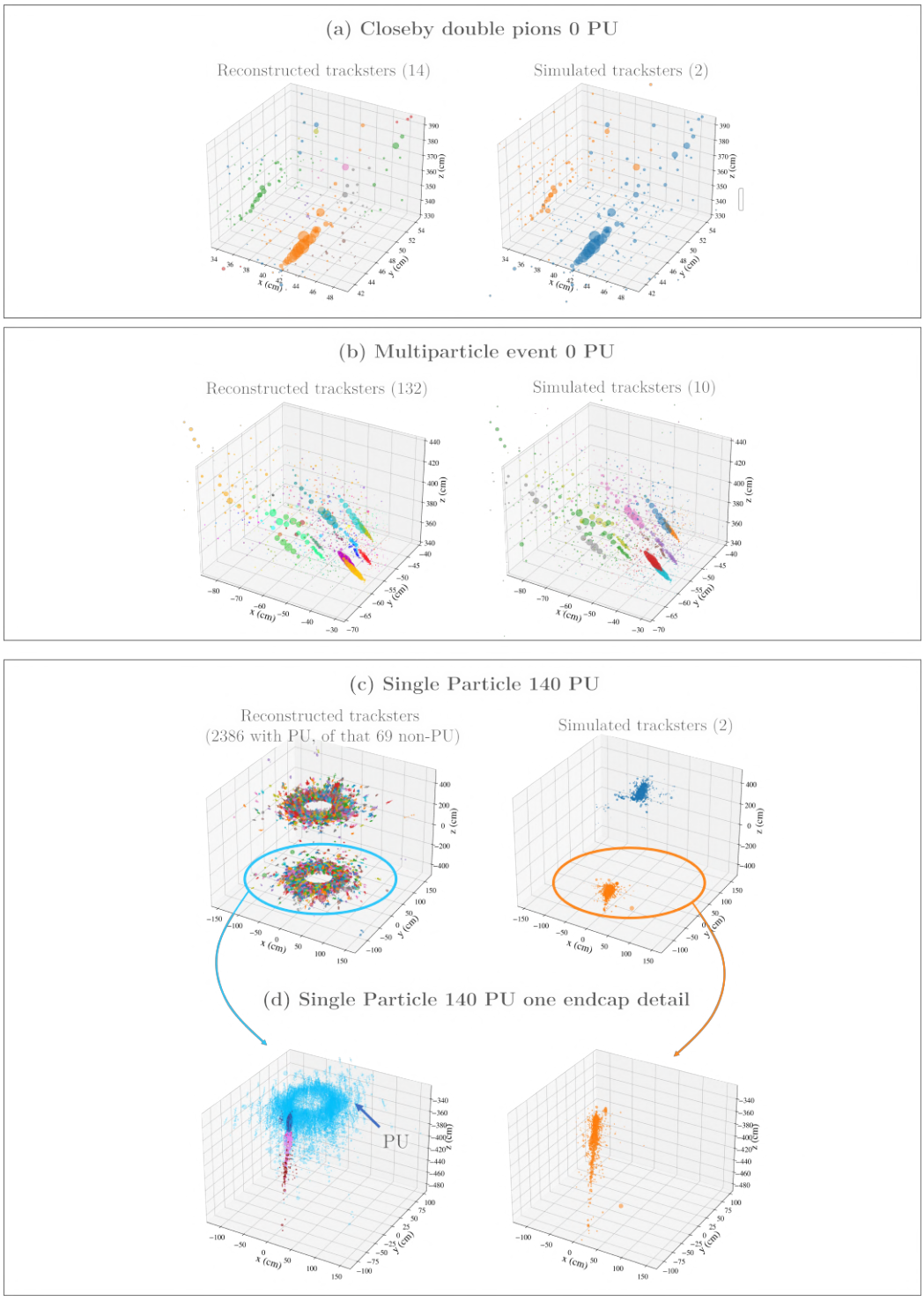


Figure 5.1: Illustration of events in the three linking datasets. Panel (a) shows the reconstructed (left) and the simulated (right) versions of two-pions scenario from the `CloseByPions0PU` dataset. The (b) panel displays the same for multiple random particles, numbering 10 in this event, from `Multiparticle0PU`. Panel (c) shows two pions, one per each endcap, in 140 PU from `SingleParticle140PU`. The detail on one of the hard-scattering particles is provided in (d), while all PU tracksters are visualized in blue. Each point represents a layer-cluster, and LCs belonging to the same trackster fragments, as clustered by CLUE3D for reconstruction or belonging to the same simtrackster in simulation, are displayed in the same colors. The z-axis is rotated upwards for better visualization. In the PU scenario, two detector endcaps can be seen. In reconstruction cases, many more trackster fragments are present than in the simulation scenario (14 vs. 2 for closeby pions, 132 vs. 10 for multiparticles, and 69 vs. 2 for the PU scenario).

In each event, 10 to 50 particles with energies ranging from 10 to 600 GeV are generated and shot in front of a single HGCAL endcap with the same η ranges as in `CloseByPions0PU`. Given the necessity of accommodating a larger number of particles within the region, a much higher separation distance ΔR of 62 cm has been chosen for this scenario. Unlike the case of two particles where ΔR can be viewed as the maximum pairwise distance between them, for a dataset containing multiple particles, ΔR defines a window around the first generated particle, within which other particles can be positioned.

Pile-Up Dataset. To simulate the expected conditions of the experiments in the HL-LHC run, we incorporate pile-up in the final and, presumably, the most challenging `SingleParticle140PU` dataset used in this work. Pile-up simulates the effect of multiple proton-proton collisions occurring simultaneously, resulting in multiple overlapping particle showers in the detector. To remind the reader, this dataset is generated in such a way that the ground truth clustering is available only for the hard-scattering particle showers in the event, as exemplified by the pion shower depicted in Figure 5.1(d). As can be seen, this dataset has significant overlap among particle showers, with many high-energy tracksters in the neighborhood of other non-matching high-energy tracksters.

For each event, a single hard-scattering particle shower is generated per HGCAL endcap, resulting in two simulated CaloParticle (discussed later) tracksters per event. The dataset includes a variety of particles produced at the CMS vertex. However, the mere presence of a particle at the vertex does not imply it will manifest as single tracksters in HGCAL. Two plausible scenarios emerge: firstly, the particle may not interact before entering HGCAL, and we observe a particle shower emanating from the single primary particle (CaloParticle in simulation). Alternatively, the particle may interact prior to entering HGCAL, resulting in the creation of numerous lower-energy particles. In the latter scenario, each new particle produces a separate trackster (SimClusters in simulation). The final detected particle arrangement hinges upon the initial particle's lifetime, energy, and decay branching ratio. It is worth noting that the produced dataset is dominated by pions, accounting for 80% of the dataset, since they present the most complex cases for the linking problem. This, unlike in previous datasets, happens for two reasons. Firstly, same as before, pions produce multiple non-aligned trackster fragments in both the electromagnetic and hadronic compartments of the HGCAL. Secondly, a large portion of the pile-up interactions also involves pions, with their behavior mimicking that of the particles of interest, making it arduous for the network to differentiate between PU and non-PU trackster fragments during the linking process.

For the generation of the events in this dataset, the `FlatRandomPtGun` is used with the transverse momentum p_{T} range from 10 GeV/c to 100 GeV/c. The decision to use a different particle gun was driven by the need to generate pile-up from the CMS collision area, the beam spot, in the later stages of the simulation using the same gun. Unlike in the previous two datasets, where the goal of linking is to merge matching trackster fragments in the whole event, here, the goal is to accurately reconstruct the shower associated with each hard-scattering particle despite the presence of thousands of other tracksters generated by an average of 140 simultaneous particle collisions. Apart from that, it is crucial to avoid merging together tracksters coming from PU.

Table 5.1: Properties of the three linking datasets (`CloseByPions0PU`, `Multiparticle0PU` and `SingleParticle140PU`) used in this study. Edges refer to the edges created through the 0.2 $\eta - \phi$ bounded graph (Section 5.4.3). Due to significant differences in properties among the particles in `SingleParticle140PU`, their values are presented separately based on the particle type.

Properties	`CloseByPions0PU`	`Multiparticle0PU`	`SingleParticle140PU`
Particle gun	`CloseByParticle`	`CloseByParticle`	`FlatRandomPt`
Separation ΔR	15 cm	62 cm	-
Average PU	0	0	140
PDG IDs	211 (pions)	22 (photon), 11 (electron), 211 (pion), -11 (positron), 130 (neutral kaon), 321 (positive kaon), -321 (negative kaon) Picked uniformly	80%: 211 (pion); 5%: 22 (photon), 5%: 11 (electron), -11 (positron); 5%: 15 (tau); 5%: 130 (neutral kaon), 321 (positive kaon), -321 (negative kaon)
Num. of simtracksters CP per event	2	10-50	Pion, Tau, Electron: 2 (1 per each endcap); Kaon, gamma: 1
ϕ range	$[-\pi, \pi]$	$[-\pi, \pi]$	$[-\pi, \pi]$
η range	[1.7, 2.7]	[1.7, 2.7]	[1.5, 3]
p_{T} range	-	-	[10, 100] GeV/c
Energy range	[10, 600] GeV	[10, 600] GeV	-
Interactions before HGCAL	No	No	Yes
Num. of events in the dataset (train/val/test)	46.5k / 5.8k / 5.8k	19.7k / 4.9k / 4.9k	14.5k / 3.6k / 3.6k
Ratio of train/val /test events	80%/10%/10%	80%/10%/10%	80%/10%/10%
Average number of edges per event $\bar{N}_e$	230	2805	Electron: 20.25 Tau: 81.7 Pion: 341.25 Kaon: 160.5 Photon: 29.35
Average number of true edges per event $\bar{N}_e^t$	181	1156	Electron: 4.75 Tau: 29.85 Pion: 145.25 Kaon: 66.6 Photon: 5.65

Continued on next page

Continued from previous page

Properties	`CloseByPions0PU`	`Multiparticle0PU`	`SingleParticle140PU`
Average number of false edges per event $\bar{N}_e^f$	49	1649	Electron: 15.5 Tau: 51.85 Pion: 196.0 Kaon: 93.9 Photon: 23.7
Total num. of edges in the training dataset N_e	∼10.7 mil.	∼41.1 mil.	∼4.1 mil.
Total num. of true edges in the training dataset N_e^t	∼8.4 mil (78.4%)	∼16.9 mil. (41.2%)	∼1.6 mil. (39.8%)
Total number of false edges in the training dataset N_e^f	∼2.3 mil (21.6%)	∼24.2 mil. (58.8%)	∼2.5 mil. (60.2%)
Dataset Imbalance (true/false edges)	78.4% / 21.6%	41.2% / 58.8%	39.8% / 60.2%
Avg. num. of non-PU tracksters per event	-	-	Electron: 5.8 Tau: 10.6 Pion: 28.7 Kaon: 14.7 Photon: 5.5
Avg. num. of PU tracksters per event	-	-	Electron: 2008.9 Tau: 2020.0 Pion: 2066.3 Kaon: 2022.9 Photon: 2023.6
Avg. num. of all recotracksters per event $\bar{N}$	28.2	215.4	Electron: 2014.7 Tau: 2030.6 Pion: 2095.0 Kaon: 2037.6 Photon: 2029.1

5.3 Raw Generated Data

To facilitate data analysis, CERN has developed a specialized package for particle physics data analysis called ROOT [BR97]. ROOT provides tools for storing, manipulating, and analyzing large datasets generated from particle physics experiments. ROOT files are organized in a tree-like data structure, with particular branches belonging to specific reconstruction steps or simulated objects. In other words, each ROOT file can be thought of as a dictionary, where the branches are the dictionary's keys, and particular events are the key values. The ROOT files generated for the datasets in this study consist of the following nine sub-trees:

- `clusters` – layer-clusters produced by CLUE (Section 3.3.1) from hits.
- `tracksters`, or recotracksters – tracksters produced by CLUE3D (Section 3.3.2) from LCs.
- The `simtrackstersSC` sub-tree contains details regarding the simulated tracksters generated by a single `SimCluster` (SC). SimClusters are secondary particles produced when a simulated particle interacts prior to entering the HGCAL. These byproducts can then enter HGCAL and are registered as distinct clusters of energy deposits.
- `simtrackstersCP` – simulated tracksters produced by a single `CaloParticle` (CP), a simulated particle that can also interact prior reaching the HGCAL, but the separate clusters detected in HGCAL are connected to produce only a single simtrackster corresponding to the CP.
- `trackstersMerged` – tracksters originated by the geometric linking algorithm applied to `tracksters`.
- `candidates`, also referred to as the `TICLCandidates`, is the output of the geometric linking algorithm discussed in Section 3.3.3. This collection features trackster fragments merged into larger tracksters, and is utilized as a baseline for evaluating the performance of the developed algorithms. `TICLCandidates` collection is the ultimate result of the TICL pipeline.
- `graph`, or a `TICLGraph`, represents a graph of the particle interactions in the whole event.
- `associations` sub-tree provides associations between simulated and reconstructed objects, including both `RecoToSim` and `SimToReco` score, allowing to estimate the ground truth for our experiments (Section 5.4.6).
- `tracks` contain information regarding the tracks of charged particles detected by the Tracker (Section 2.3.2).

During the simulation process, multiple ROOT files, referred to as ntuples, are generated, each containing a specific number of events. Due to the higher memory requirements and slow ntuple access, these raw files are loaded sequentially; necessary data is pre-processed and stored in much faster `pickle` files used during the training process. The subsequent sections provide a detailed description of each ROOT sub-tree, including its properties and function in our final processed datasets.

Clusters

The CLUE algorithm produces a layer-cluster collection, which is referenced to by both simtracksters and recotracksters. Neither the `tracksters` nor the `simtrackster` data objects directly contain the constituent LCs information, but instead, refer to them using a property called `vertex_index`. It is worth noting, that simulated and reconstructed tracksters include different LCs, making ground truth assignment between the two more complicated. The properties of layer-cluster collection are described in Table A.1 and include features accumulated from the individual hit properties (such as the accumulated time, barycenter, and energy information) or coming from the CLUE step (i.e., local density).

Tracksters, SimtrackstersSC, SimtrackstersCP, and TrackstersMerged

In essence, a trackster is a collection of layer-clusters, where every LC is identified by an index to the cluster collection. The attributes in Table A.2 provide further details about each

of the four trackster sub-trees. These collections include aggregated LC properties, such as barycenter positions, shape properties, accumulated energies, etc., as well as PIDs and LC seeds.

Simulated particle showers are represented with simtracksters used as the reference for the reconstruction process. The simulated data is available at the granularity of individual hits during the simulation, but this low-level information is not saved in the ntuples. Since multiple particles may contribute to the energy of a single simulated LC (`vertices_multiplicity` property), the fraction of energy contributed by each simtrackster is computed and assigned to the corresponding LC (not stored in the ntuples). In the reconstruction evaluation, this energy fraction plays a pivotal role in accurately computing the shared energy between simulated and reconstructed tracksters. Conversely, this fraction is always unity during the reconstruction process since each LC is exclusively assigned to a single trackster.

Candidates

`Candidates` sub-tree stores the `TICLCandidates` constructed through the geometric linking algorithm. This sub-tree provides properties relevant to geometric linking outlined in Table A.3. Among these properties, the most pertinent for comparing with the developed GNN model is the `trackster_in_candidate`, giving individual trackster fragment assignments.

Graph

To construct the `TICLGraph`, the algorithm searches for other tracksters within a 0.2 $\eta - \phi$ window opened on the barycenter of each trackster. The window's size is sufficient to encompass all of the significant trackster fragments without requiring a fully-connected graph of the event. A trackster inside the window is referred to as an *inner* if it is nearer to the CMS vertex than the trackster of interest and as an *outer* if it is farther away. Just three properties are available for `TICLGraph` outlined in Table A.4. Every node in a graph is a recotracksters with the corresponding index in the CLUE3D `tracksters` collection.

Tracks

As was mentioned in Section 3.3, TICL incorporates external data from various CMS sub-detectors. One such sub-detector is the tracker, which provides `tracks` information with properties shown in Table A.5. After excluding muon tracks, only those with a transverse momentum p_{T} greater than 2 GeV/c are propagated to the front face of the HGCAL. This information is then utilized by geometric linking, unlike the GNN approach, which does not rely on track data. Nevertheless, incorporating track information in GNN is advantageous since the tracksters formed must be consistent with the tracks, giving additional constraints on the clustering in the linking step. Additionally, the number of tracks leading to the detector precisely determines the number of charged particles expected to be reconstructed. Therefore, the number of final recotracksters is lower-bounded by the number of tracks.

Associations

The evaluation of reconstructed tracksters can be carried out by comparing them with the ground truth obtained from the simulation. To achieve this, an `associator` is provided, which rates the quality of the reconstructed tracksters against the ground truth and supplies

labels for training the ML algorithms. For this reason, `associations` are used to evaluate the performance of the linking procedure.

As a reminder, the simulation of particle interactions with the detector is performed at the hit level (not saved in the generated ntuples); each hit is associated with the corresponding simulated particle. The TICL pipeline is applied to the simulated hits as if they were recorded during a standard detector run. However, the pattern recognition stage does not directly operate on hits, but rather on higher-level energetic structures, LCs, for which initial simtrackster assignments are unavailable. The associator algorithm furnishes a metric by comparing reconstructed tracksters to their simulated counterparts. Association scores are derived by considering the degree of overlap between hits belonging to a specific simulated trackster (can be both SC and CP) and the corresponding reconstructed trackster, and vice versa. While a perfect correspondence between reconstructed and simulated tracksters would entail a one-to-one mapping, the complex nature of the interactions between particles and the detector can result in situations where a simulated trackster is fragmented into several reconstructed tracksters, or a single reconstructed trackster encompasses multiple LCs from different simulated tracksters.

In the initial stage of the TICL pipeline, hits are clustered into LCs. Thus, before the association scores for tracksters can be computed, they must first be accumulated for each layer-cluster. Therefore, following the completion of the CLUE algorithm, the shared energy between each LC i and each simtrackster s in the event is computed by taking the weighted sum of the hit energies E_h of the LC, with the weights given by the fraction fr_h^s of the hit energy deposited by the simtrackster s, and normalized by the total energy E_i of the LC:

$$fr_i^s = \frac{\sum_{h \in i} fr_h^s \cdot E_h}{E_i} \quad . \tag{5.1}$$

The fraction ranges from 0 to 1, being 0 if no overlap of LC i hits with the simtrackster s is observed. These LC energy fractions are then utilized to calculate the pairwise `Sim-to-Reco` scores between tracksters and simtracksters based on their LC shared energy, normalized by the square of the total energy of simtracksters:

$$\text{score}_{s,t} = \frac{\sum_{i \in s} \min\left(\left(fr_i^t - fr_i^s\right)^2, \left(fr_i^s\right)^2 \right) \cdot E_i^2}{\sum_{i \in s} \left(fr_i^s\right)^2 \cdot E_i^2} \quad , \tag{5.2}$$

where i refer to individual constituent LCs of the simtrackster, fr_i^t and fr_i^s are the energy fractions of LC i that has been assigned to the reconstructed trackster t and to the simtrackster s, respectively. The `Sim-to-Reco` score measures the degree to which a simulated object s is accurately represented with the reconstructed object t with a score of 0 indicating a perfect match between the objects. The LCs not shared between the objects move the score towards high values, eventually reaching 1 for two completely unrelated objects. The `Reco-to-Sim` score is similarly defined and can be obtained simply by interchanging the roles of the reconstructed and simulated tracksters. Meaning-wise, it establishes opposite links between the reco- and simtracksters than the `Sim-to-Reco` scores. All associations between objects are meant to be one-to-many, i.e., a single object can be linked to multiple objects, with each link assigned a specific score. Pairs of objects with a `Reco-to-Sim` or `Sim-to-Reco` score less than 0.2 are deemed to be *associated*.

The `associators` sub-tree of the generated ROOT datasets contains the properties listed in Table A.6. While these properties are also available for the `trackstersMerged`, they are not utilized in this analysis and are not shown in the table.

5.4 Processed Linking Datasets

The intent of the dataset processing, in view of using graph neural networks for linking, is to represent collision events as graphs, with each node representing a trackster fragment generated by the CLUE3D procedure and edges indicating interactions between them. These graph structures will then be used as inputs to the GNN. Due to the high number of tracksters in pile-up datasets, simply constructing fully-connected graphs is not computationally viable from the network's point of view. Consequently, there is a need to explore various graph construction techniques to reduce the number of graph edges, as detailed in the following sections. To start with, the selection of node and edge features is described, followed by the graph construction considerations.

5.4.1 Node Features

For each recotrackster, which will serve as a separate node in the event graph, we select a set of features grouped into four categories: spatial and kinematic properties, particle identification, and substructure features. These properties are extracted from the generated ntuple data and provide a total of 33 features per trackster.

Spatial Properties. This category comprises 21 trackster properties arising from the aggregation of its constituent LCs and hits. The properties include:

- `trackster_barycenter_x/y/z/eta/phi`: Positions of tracksters' barycenters, computed by energy-weighted average of the constituent LC coordinates, represented in both cartesian (x, y, z) coordinate space and (η, ϕ) space. Unlike layer-cluster barycenters, the z trackster coordinate may not align with any particular detector layer but rather fall between the two adjacent layers.
- `EV1/EV2/EV3`: The first three eigenvalues of energy-aware PCA applied to the collection of constituent LCs.
- `eVector0_x/y/z`: The first principal component coordinates x, y, z of energy-aware PCA.
- `sigmaPCA1/2/3`: The first three component-wise reconstruction errors from energy-aware PCA.
- `num_LCs/hits`: Number of LCs and hits per trackster, excluding 1-hits LCs, which are disregarded during reconstruction.
- `z_min/z_max`: Minimum and maximum coordinates of the trackster PCA skeleton[3] (Section 5.4.4) components along the z-axis.
- `length`: Shower length in terms of the number of occupied HGCAL layers normalized with the full number of HGCAL layers (47 per endcap).

[3] A skeleton can be thought of as a simplified trackster backbone structure.

- `LC/trackster_density`: LCs and trackster densities in event computed either in the entire detector or in a particular seeding region, depending on the setup of the TICL pipeline. These densities are obtained by dividing the full number of LCs/tracksters by an estimated volume of reconstruction, which is approximately equal to $2 \times (3 - 1.5) \times (2 \times 47)$ in the case of the full HGCAL. This calculation is based on the HGCAL volume with two η ranges from 1.5 to 3, containing 47 layers in each of its two endcaps. Normalization is needed since the networks can potentially be run on the volumes limited by the seeding regions. Densities are the same for every trackster in the event.

The last two features should contribute to the adaptation of the GNN to events with varying levels of pile-up[4], since the GNN is expected to behave differently in these scenarios due to the higher object density in pile-up events. In the context of multiple trackster fragments, spatial features are very informative, since tracksters located close to each other and pointing in the same direction or originating from the same spatial point are likely to arise from the same CP.

Kinematic Properties. The kinematic property category comprises just three features:

- `time`: Trackster barycenter time information, accumulated from the LC times.
- `log E` / `log E_EM`: Natural logarithms of the trackster's total raw and total electromagnetic energy (i.e., energy deposited in the CE-E) in GeV units. The logarithms are taken due to the long-tailed nature of the energy distribution and are protected from negative values[5].

The time information is valuable for identifying tracksters that could have originated from the same particle since they would exhibit similar time information. With regards to energy, tracksters with lower energy levels can serve as indicators of small fragments to be linked to the higher-energy tracksters.

Particle Identification Properties. The particle identification category comprises six features representing probabilities of the following particle types from recotrackster PIDs:

- `Photon`: Probability of a trackster being produced by a photon (PDG ID 22).
- `Electron`: Probability of a trackster being produced by an electron (PDG ID 11).
- `Muon`: Probability of a muon (PDG ID 13).
- `Neutral Pion`: Probability of a neutral pion (PDG ID 111).
- `Charged Hadron`: Probability of a charged hadron (PDG ID 211, 321 or 2212).
- `Neutral Hadron`: Probability of a neutral hadron (PDG ID 130, 2112 or 0).

Considering that different particles undergo distinct interactions, the aforementioned properties hold value in defining separate linking strategies suitable for diverse particle types.

[4] Especially, for networks trained on the data with varying levels of pile-up.
[5] Specifically, $\log(E + 1)$ and $\log(E_{EM} + 1)$ are used to ensure non-negative values

Structural Properties. After constructing the graph of an event (as described further in Section 5.4.3), additional trackster information can be extracted. One such piece of information is the normalized node degree $d_{i,norm}$ for each trackster i, calculated by dividing the number of edges incident upon the node by the maximum node degree d_{max} in the graph:

$$d_{i,norm} = \frac{1}{d_{max}} \sum_{j \in \mathcal{V}} \mathbf{A}(i,j) \quad , \tag{5.3}$$

where $\mathbf{A}$ is the adjacency matrix, and $\mathcal{V}$ is the set of vertices in the graph. Similarly, degree centrality c_d is calculated by normalizing with the maximum *possible* degree (i.e., $n-1$ for a graph with n nodes) rather than the maximum degree in the graph. Additionally, the clustering coefficient c_i [Kai08] is derived for each trackster i, measuring the proportion of closed triangles in the neighborhood of the node:

$$c_i = \frac{2 \cdot |e_{jk} : v_j, v_k \in \mathcal{N}_i|}{d_i(d_i - 1)} \quad , \tag{5.4}$$

where the numerator gives the number of triangles through node i with $\mathcal{N}_i$ referring to the node's neighbors in the graph, and d_i is the non-normalized node degree. Despite the potential for incorrect edges in the event graph, these features provide a useful overview of node connectivity. It should be noted that the normalized node degree is calculated only for the outgoing node edges, while the degree centrality is calculated for the undirected graph. This is done to minimize the correlation between the two properties.

5.4.2 Edge Features

Particle interaction features, or edge features, are derived from the geometrical and kinematic properties of the tracksters connected with an edge in the event graph. Specifically, for a pair of tracksters i, j, a suite of the following features is calculated:

- Edge energy difference $\Delta E_{ij} = |E_i - E_j|$ for each edge e_{ij} in the event connecting trackster with energies E_i and E_j.
- Δz barycenter-barycenter separation in z axis, computed in terms of the Euclidean distance between the pair of trackster barycenters.
- Euclidean distance between the two closest skeleton points belonging to the pair of tracksters.
- Euclidean distance between the two furthest skeleton points belonging to the pair of tracksters.
- Barycenter separation in the transverse plane $\Delta R = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.
- The spatial compatibility, expressed through the angle α between the primary components of the trackster skeletons, given by $\alpha = \arccos(\mathbf{v}_i \cdot \mathbf{v}_j)$, where $\mathbf{v}_i$ and $\mathbf{v}_j$ are the principal component vectors. The choice of an angle, instead of just a dot product is motivated by the skewed distribution of the PCA product, with values below 0.6 being a rarity, typically occurring in cases of incomplete fragments characterized by a low number of LCs and, consequently, an inconsistent PCA.
- Barycenter time compatibility $\Delta t = |t_i - t_j|$, if available. If not available, it is set to -99, as in the ntuples for the missing time values.

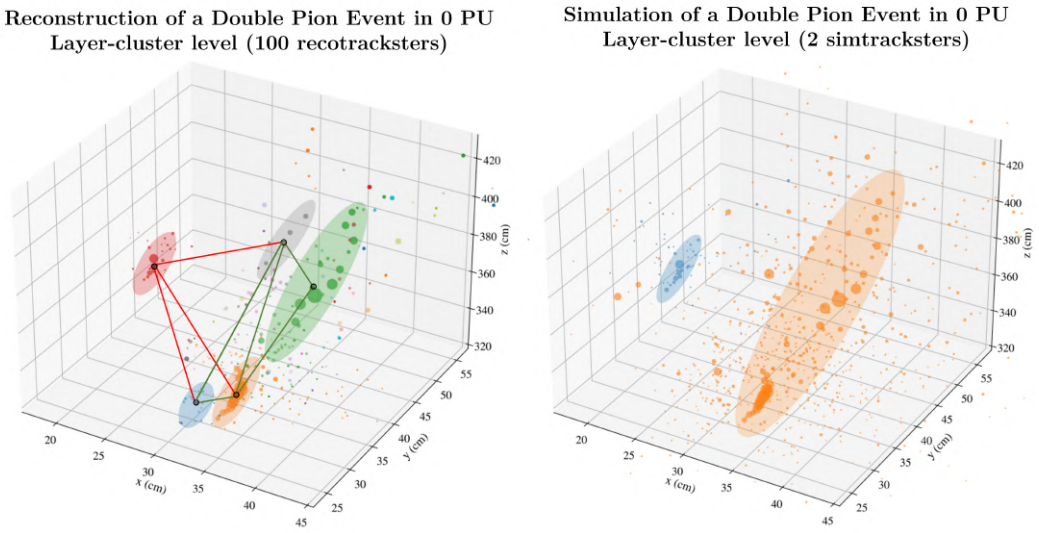


Figure 5.2: Schematic example for an event of double closeby pions shot in front of the HGCAL in 0 PU. On the left, reconstructed tracksters numbering 100 are shown in different colors, with each circle representing a LC. Only five major tracksters are visualized in an ellipsoid form to improve the readability of the plot, along with the edges connecting their barycenters under the condition they fall within a 0.2 $\eta - \phi$ window. True edges are shown in green, and the false ones are in red. On the right, a similar visualization corresponding to the same simulated event and comprising two simtracksters is shown.

Because ΔE and Δz typically have long-tail distributions, their logarithms with protection against negative values are taken as the interaction features. It is also important to note that the irregular geometry of the detector poses a challenge to Euclidean distance metric, since it exhibits non-uniform performance across different layers, as depicted in Figure 4.4. The electromagnetic section, being more densely packed, contrasts with the hadronic section, which features wider gaps between its layers. As a solution, we adopt both Euclidean distance and the number of layers between the tracksters, despite their intrinsic correlation.

5.4.3 Event Graph Building: Eta-Phi Bounded Graph

The first explored approach for event graph building involves the `TICLGraph` present in the raw ROOT dataset files. This method establishes connections between trackster barycenters and other nearby *inner* tracksters (lying closer to the CMS primary vertex) whose barycenters fall within a 0.2 $\eta - \phi$ cylindrical region around the axis of the trackster of interest (Figure 5.2). The axis of a trackster is determined by the line connecting its barycenter with the collision point at the center of the detector. The baseline approach for this method employs a maximum cylinder length of 200 cm, corresponding to almost the full HGCAL length, enabling the connection of very distant longitudinal objects. This threshold is set according to the trackster separation experiments discussed later. A trackster is linked to its nearest neighbor if it has no neighbors within the window, to avoid disconnected nodes in the graph. This situation might happen when we are dealing with a complete trackster that does not require further links or a remote fragment of a shower. The size of the window is chosen to be sufficient to guarantee that the majority of trackster pairs coming from the same particle have an edge between them or are connected via intermediate nodes. This approach is motivated by the angular compatibility of tracksters reconstructed from the same particle, which tend to lie within

the aforementioned angular window. As a result, the GNN is directed to link longitudinally aligned tracksters. The pseudo-code for this algorithm is outlined in Algorithm 1.

The construction of the $\eta - \phi$ bounded graph is a simple and efficient approach for building event graphs, allowing to avoid fully-connected graphs for event representation. It makes it possible to keep the number of edges to a physically feasible set that is still large enough to account for the majority of potential connections. However, it solely relies on trackster barycenter information, neglecting trackster internal structures. Consequently, this event graph might struggle to connect tracksters whose barycenters are distant yet have closely spaced individual layer-clusters. Apart from this issue, $\eta - \phi$ graph building fails to account for the detector's irregular geometry, where the η region varies significantly with increasing distance from the CMS vertex. Therefore, graph building can benefit by introducing a variable-size neighborhood window in the future.

Graphs for Pile-Up Events. The construction of the pile-up training graph requires additional considerations. Although the inference graph is constructed analogously to the aforementioned process, constructing the training graph poses a challenge since there is no available ground truth data for the pile-up tracksters. Consequently, only the edges from the tracksters that are associated with the simtracksters are considered for the graph building process.

5.4.4 Event Graph Building: Skeleton-Based Graph

As an alternative to the `TICLGraph` strategy, which focuses solely on trackster barycenters, I developed a skeleton-based graph building approach making use of internal trackster structures, referred to as *skeletons*, to drive the graph construction process. To achieve this, several algorithms for trackster skeletonization have been developed with varying degrees of complexity, as described in the following sections. The first method involves iterative energy-weighted addition to the skeleton, providing extensive structures covering the majority of LCs in the tracksters. Another approach, providing simpler skeletons, is based on energy-aware principal component analysis with secondary components. Aside from that, I have explored iterative random sample consensus (RANSAC) [FB81] skeletonization, which may be particularly suitable for events with minimum ionizing particles (MIPs), whose LCs are typically well-aligned single hits. The application of PCA and RANSAC methods in trackster skeletonization results in the emergence of primary components, which are energy-driven, and secondary components, which are geometry-driven. In constructing the event graph, only the primary components are considered for linking tracksters to tracksters. However, the secondary components are retained for future investigations concerning linking tracksters to minimum ionizing particles. Additionally, as a part of future work, one can extract additional properties for network training from these skeleton structures.

The skeletonization of tracksters serves a dual purpose – apart from facilitating event graph building, it also enables propagating of time information along the skeleton structures. This feature is especially beneficial for low-energy LCs, which may not have accurate timing information available. While this study does not delve into this aspect, the PCA-based algorithm developed herein is presently being employed in the time-improvement study for HGCAL. Additionally, this technique also enables investigation into the time compatibility of the tracksters, which, however, falls beyond the scope of this work.

Skeletonization provides a valuable solution for scenarios where particle showers have

Algorithm 1: $\eta - \phi$ Graph Building and Edge Labeling (Section 5.4.6) for the training dataset.

Data: `tracksters`, `associations`, and `graph` for a single event; maximum trackster separation threshold `max_dist` (default: 200 cm).
Result: list of `edges`, `edge_labels`, `simtrackster_match` and `edge_scores` for the constructed graph.

Initialize output variables with empty lists.
Algorithm *BuildEtaPhiGraph*
  for *tr_i* $\in$ *tracksters* **do**
    `score_tr_i` $\leftarrow$ best (minimum) `tr_i recoToSim_score` from `associations`.
    `best_simtr_i` $\leftarrow$ simtrackster ID corresponding to `tr_i` calculated through a `recoToSim` map **if** `score_tr_i` < 0.2 **else** `None`.
    `simtrackster_match`.append(`best_simtr_i`)
    if *best_simtr_i != None* **then**
      for *tr_j* $\in$ *graph.linked_inners(tr_i) for which* Dist*(tr_j,tr_i)* $\leq$ *max_dist* **do**
        `edges`.append([`tr_j`,`tr_i`]) `// Create an edge`
        `score_tr_j` $\leftarrow$ best (minimum) `tr_j recoToSim_score` from `associations`.
        `best_simtr_j` $\leftarrow$ simtrackster ID corresponding to `tr_j` calculated through a `recoToSim` map **if** `score_tr_j` < 0.2 **else** `None`.
        if *best_simtr_i == best_simtr_j* **then**
          `edges_labels`.append(1) `// Edge is true`
          `edges_scores`.append(GetEnergyScore(`tr_i, tr_j`))
        else
          `edges_labels`.append(0) `// Edge is false`
          `edge_scores`.append(0.)
    if *tr_i has no neighbours in* $\eta - \phi$ *window* **then**
      `nearest_tr` $\leftarrow$ FindNearestNeighbour(`tr_i`)
      edges.append([`tr_i`, `nearest_tr`])
      `edges_labels`.append(0) `// Edge is false`
      `edge_scores`.append(0.)

multiple primary components. Upon linking two tracksters, their data is merged, and PCA directions are recalculated, which can result in the loss of the individual backbones of the original tracksters. To overcome this issue, we propose utilizing primary components of the skeletons to enhance the representation of merged tracksters for hadronic showers with multiple energy blobs, as well as for electrons merged with their bremsstrahlung photons. Additionally, skeletonization facilitates identifying and quantifying substructures within the trackster, such as individual energy blobs or branches, as well as graph features, which can be used as additional features for NN training.

Once the trackster backbones have been acquired, we create an event graph by analyzing the nearest skeleton points of each tracksters falling within the same 0.2 $\eta - \phi$ window. This approach enables more accurate distance calculations between tracksters as opposed to relying solely on trackster barycenters. The selection of a specific skeletonization method is contingent on the objective of the task at hand. Each method of skeletonization yields a distinct type of graph. The iterative graph is composed of layer-clusters as nodes, forming

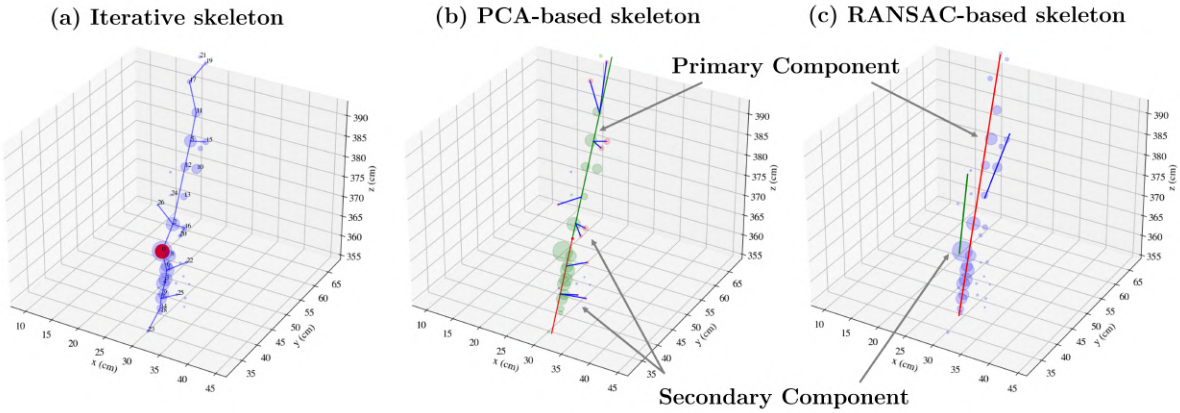


Figure 5.3: Trackster skeleton examples generated by three implemented algorithms: (a) iterative, (b) PCA-based, and (c) RANSAC-based, for the same pion trackster fragment. In the PCA skeleton example, the primary component is divided into two parts, depicted in green and red, while the secondary components are shown in blue.

a connected graph. The PCA-based graph contains a primary component that does not generally pass through the individual LCs, which is complemented by secondary components that are connected to LCs at one of their endpoints. This skeleton is also a complete graph. In contrast, the RANSAC-based skeletonization generates a set of disconnected segments that are not necessarily associated with layer-clusters, resulting in a disconnected graph. In the present work, PCA-based skeletons, offering simple and descriptive internal structures, showed to be the most effective.

Iterative Energy-Aware Skeleton Building

The proposed iterative skeleton creation algorithm is outlined in Algorithm 2. It takes trackster's LC data as an input and defines a distance threshold of $d = 2$ cm for LCs to be considered covered by the skeleton. Starting with the CLUE3D seed LC, the LC data is sorted based on descending energy, and for each LC, the algorithm calculates the direct distance to the closest skeleton edge. We are only interested in LC that are not covered by the skeleton (i.e., the distance is greater than d); otherwise, we consider them explored. If the new non-covered LC is closer to the seed than the closest endpoint of the closest edge, the algorithm checks if breaking an old connection into two with the explored LC as a middle-point affect already explored LCs (i.e., they still stay within d interval from the set of edges, possibly different ones). If no explored nodes are affected, we perform the edge splitting (line 7 of Algorithm 2).

However, if the new non-covered LC is further from the seed than the closest endpoint of the closest edge (line 11 in Algorithm 2), the algorithm checks if removing the closest edge and connecting the new LC directly to the seed affects explored LCs. If that is not the case, we perform the edge deletion. Finally, if neither of the above two cases is satisfied (line 14 of Algorithm 2), the algorithm adds a new edge connecting LC to the closest endpoint of the closest edge to the skeleton. The process continues until the skeleton has covered all LCs above an adaptive energy threshold.

The skeleton creation method employed in this study involves pruning the edges so that they are not incident on every LC in the trackster, simplifying the backbone. While this approach yields a highly descriptive graph that captures the internal structure of the trackster,

Algorithm 2: Iterative skeleton creation pseudo-code.

Data: trackster's LC data, distance threshold `d` (default: 2 cm).
Result: trackster skeleton in terms of it's `edges`.

Initialization: `edges` and `explored_LCs` are empty sets.
Algorithm *BuildSkeletonIteratively*
  Add CLUE3D seed to the set of `explored_LCs`.
  `ordered_LCs` ← LCs (except for the `seed`) sorted in descending order of their energies.
  for *lc* ∈ *ordered_LCs* **do**
    Find `lc`'s direct distance `min_dist` to the closest skeleton edge `min_edge = (close_ep, far_ep)`
    if $|lc - seed| < |close\_ep - seed|$ *and breaking* `min_edge` *up with the* `lc` *as a midpoint does not affect* `explored` **then**
      `edges.del(min_edge)`
      `edges.add((close_ep, lc))`
      `edges.add((LC, far_ep))`
    else if $|lc - seed| > |close\_ep - seed|$ *and removing* `min_edge` *while connecting* `lc` *directly to* `seed` *does not affect* `explored` **then**
      `edges.del(min_edge)`
      `edges.add((far_endpoint, lc))`
    else
      `edges.add((closest_ep, lc))`

it also generates an excessive number of edges for hadronic events making it impractical for use in event graph building.

PCA-Based Skeleton Building

The layer-clusters of the highest energy are typically aligned with the shower's directional axis, while the lower-energy LCs tend to lie farther away. This characteristic high-energy core of the shower is leveraged to construct a trackster skeleton using a weighted PCA. The skeleton's primary component is constructed by limiting the first eigenvector direction to the trackster's energetic core. After that, the secondary components are iteratively built for high-energetic LCs in the descending order of their energy if not already covered by the skeleton (i.e., within a certain distance threshold). Secondary components are created by connecting those LCs to the projections of the nearest higher energy LC already covered by the skeleton on the principle component, thus following the energy chain.

However, applying PCA on LCs within a trackster and using the first principal component as the trackster direction may yield erroneous results, particularly for incomplete tracksters fragments. This is attributed to the unreliable direction estimation obtained from PCA for such tracksters. Consequently, in situations where insufficient layer-clusters are covered by the PCA, the trackster skeleton is represented as a single point – trackster's barycenter.

RANSAC-Based Skeleton Building

RANSAC, also known as Random Sample Consensus [FB81], is an iterative technique for estimating the parameters of a mathematical model from a set of observed data that may

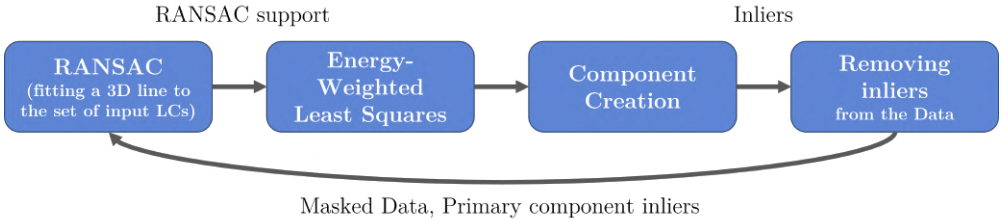


Figure 5.4: RANSAC-based trackster skeletonization pipeline. In the first step, LC data are fed to RANSAC, fitting a line to the data and generating its support. In the next step, the line is re-fitted with the energy-weighted least squares on RANSAC support. After this stage, the skeleton component is created and its inliers are removed from the data. The process is repeated until the stopping condition is met.

contain outliers. The RANSAC-based approach to skeletonization (Figure 5.4) entails a three-step iterative process. First, the algorithm fits a 3D line to the trackster's LCs by randomly selecting pairs of LCs and treating them as hypothetical inliers. Using these LCs, we determine the position and direction vectors of the estimated line model. Next, we test all the remaining LCs against this estimated model by calculating the distance of each LC to the line. Points whose distance is less than a specified threshold are classified as inliers, while those outside the threshold are outliers. We calculate the support for this model based on the number of inliers. The line with the maximum support above the threshold (we require at least 4 energetic LCs to be covered by the line) is chosen after a fixed set of iterations and re-fitted with energy-weighted least squares on RANSAC support set. The algorithm then limits the range of the line to the energetic segment (by projecting inliers LCs), forming the primary component. Finally, the inliers covered by the component (i.e., fall inside a $d = 2$ cm region from the segment) are removed from the available LC data. The process is then repeated, with the only difference – one of the points for RANSAC iteration is now chosen from the inliers of the primary component. The created segments in the following iterations are called secondary components. This iterative process continues until enough LCs above a certain energy threshold are not covered by the skeleton components, and we are able to find a line model for them.

If we are unable to obtain enough RANSAC inliers by the end of all iterations, we conclude that no line model exists, and the trackster skeleton is represented either by the previously identified components or by a single point – its barycenter. The latter scenario is typically observed when dealing with small trackster fragments without aligned LCs. Unlike other skeleton methods, RANSAC's secondary components generally do not intersect with the primary component, creating a set of line segments rather than a connected graph. While RANSAC is effective for well-aligned tracksters, it may struggle with complex hadronic events, resulting in multiple parallel components.

5.4.5 Reduced Graphs

In the event of high PU scenarios, the complete $\eta - \phi$ graph of the endcap might not be feasibly processed in a single step for rapid online reconstruction. As indicated in Table 5.1, the trackster count in 140 PU reaches above 2000, resulting in more than 10k edges as shown in Figure 5.7. In lieu of this, when computational complexity presents a bottleneck, the TICL framework can be leveraged to concentrate specifically on individual seeding regions. Graphs can then be constructed exclusively in these regions, in proximity to L1 objects or tracks,

resulting in a substantial reduction of computational complexity. Moreover, the utilization of seeding regions facilitates parallel region processing. Each region may be processed as an instance in a batch at the network input, requiring padding (or a different mini-batching approach shown in Section 6.2.3) due to variations in size.

5.4.6 Ground Truth Edge Labeling

In order to differentiate between the true and false edges in the constructed event graphs, we first need to determine the similarity scores between the simulated and reconstructed tracksters. This similarity is determined based on the associator scores discussed in Section 5.3. To this end, each edge in the graph is assigned a binary label {0,1}, where 1 (or a true edge) indicates that the connected tracksters originate from the same simtrackster (either CaloParticle or SimCluster). In contrast, 0 (false edge) indicates that they originate from different objects. This study concentrates on connecting tracksters that originate from the same CaloParticle. To remind the reader, CaloParticle refers to the parent of a particle in the simulated decay graph situated closest to the vertex (potentially, prior to entering HGCAL).

In addition to the binary labels, each true edge is assigned a score ranging between 0 and 1, calculated based on the shared energy between the reconstructed trackster and the corresponding simulated trackster. This score helps to account for the imbalance in the dataset and prioritizes the connection of tracksters that contribute the most to the total simtrackster energy:

$$score(i,j) = \begin{cases} (1-\delta_i)\cdot\frac{E_i^s}{E_i} + (1-\delta_j)\cdot\frac{E_j^s}{E_j} & \text{if edge is } \texttt{True}, \\ 0 & \text{if edge is } \texttt{False}, \end{cases} \tag{5.5}$$

where δ_i^s is the `Reco-to-Sim` associator score for trackster i coming from simtrackster s, E_i^s denotes the shared energy between i and s, and E_i is the total raw energy of i. Edge scores for false edges are set to 0.

5.5 Dataset Analysis

This section is devoted to gaining insights into the properties of the problem and the nature of the data, with a particular emphasis on hadronically interacting particles.

Feature Correlations. The presented correlation matrix in Figure 5.5 showcases the interdependence among the selected network features in the single pion events in 140 PU, where only hard-scattering trackster components are considered (i.e., events are stripped of pile-up tracksters). The following observations are made:

- Notably, a positive correlation exists between the trackster barycenter coordinate and the first eigenvector components, indicating that the trackster direction points back to the CMS vertex, where they were produced. In contrast, this correlation does not hold exactly for the double pion dataset in 0 PU (refer to Figure A.2 in the appendix), where the x and y components exhibit a strong correlation with their respective barycenter

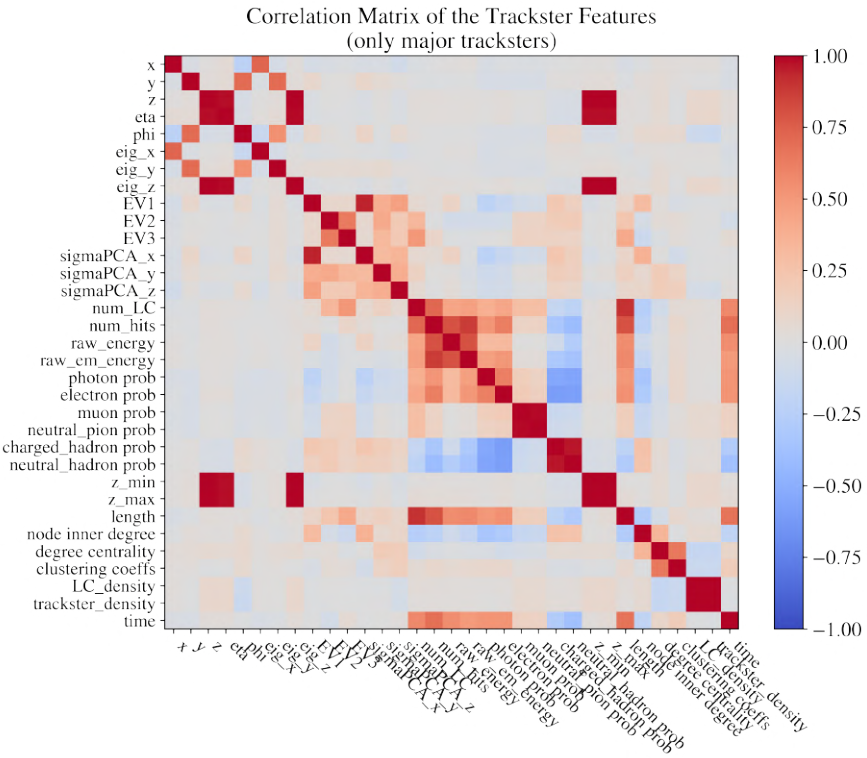


Figure 5.5: The correlation matrix pertaining to the chosen trackster features of 5000 events from a solitary pion in the 140 PU dataset, excluding tracksters caused by pile-up interactions.

coordinates, while the z component displays a much weaker correlation. The reason for such a difference might be that particles in the two datasets are generated at different vertices. In the case of no PU datasets, the particles are produced right before the first HGCAL layer, while still pointing to the CMSW origin. On the other hand, particles in PU dataset are generated directly from the CMS vertex.

- Additionally, the second and third eigenvalues, which quantify trackster's shapes in the plane perpendicular to its main axis, exhibit a strong correlation, implying the symmetry of tracksters in the transverse plane.

- A considerable correlation between σ_{PCA1} and the first eigenvalue is evident. σ_{PCA1} is the first component-wise reconstruction error from energy-aware PCA applied to constituent LCs, while the first eigenvalue measures the trackster's extent along its principal axis. Thus, when the trackster has a well-defined axis, the reconstruction error in that direction will be minimized, leading to a smaller value for σ_{PCA1} and a larger value for the first eigenvalue.

- As expected, raw energy manifests a positive correlation with raw electromagnetic energy. Additionally, a positive correlation between the number of LCs and energy and some particle types is evident.

Very similar interdependencies were observed for the pile-up data; the correlation matrix difference between pile-up and reconstructed tracksters can be seen in the appendix (Figure A.4).

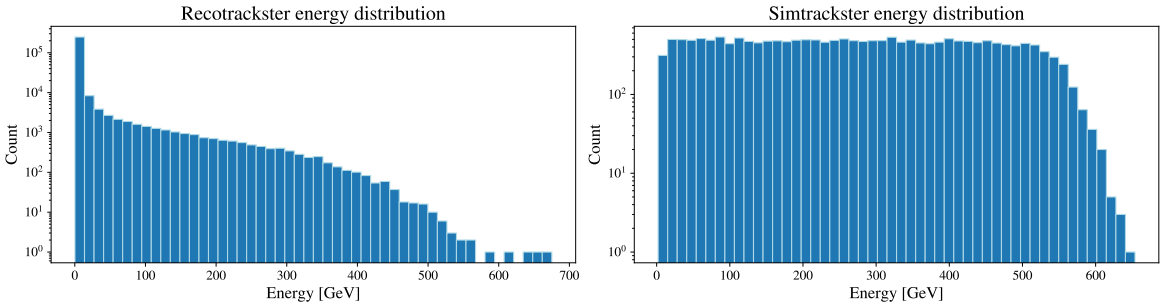


Figure 5.6: Energy distribution for reco- and simtracksters for 10 000 events from the double pions dataset in 0 PU. Original simulated particles are in the range 10–600 GeV, with their energies uniformly distributed.

Energy Distribution. The CLUE3D step tends to split high-energy showers into smaller tracksters fragments due to the conservative clustering settings trying to prevent merging with the pile-up tracksters, as shown in Figure 5.6. In contrast to the simulated particles, which display a uniform distribution of energies ranging from 0 to 600 GeV, the reconstructed data is dominated by tracksters with low energies. It is expected that the reconstructed trackster energy distribution can be improved by trackster linking.

Graph Parameters. Figure 5.7 displays the edges formed by the $\eta - \phi$ event graph building algorithm in the high pile-up scenario. We observe some very long true edges connecting tracksters belonging to the same CaloParticle, which prompts to investigate the edge length distributions (Figure 5.8). For double pion events, the peak of the true length distribution can be seen at 9 cm, while for single pion events in PU it is around 7 cm. However, the length of the true edges can range above 96 cm and 98 cm (95% percentiles) for double pions and single pions in pile-up, respectively. These long edges usually belong to connections between lower-energy tracksters with relatively low associator scores. Although the distribution of true edges remains consistent in both the pile-up and no pile-up scenarios, the distribution of false edges differs significantly. More negative edges are generally observed in the pile-up scenario due to the presence of additional pile-up tracksters in the event. This results in an additional peak in the false edge distribution for pions in PU at around 45 cm length.

Similar analysis is performed for other particles and presented in Table 5.2, where considerable variations in edge distributions with respect to particle types are observed. Particularly, photons and electrons are associated with relatively shorter true edges, 95% of same-shower tracksters lie within a distance of merely 16–18 cm. Conversely, pions, kaons, and taus generate true edges that can extend up to 98, 92, and 91 cm, respectively, at their 95% percentile. These observations are anticipated since photons and electrons typically interact in the electromagnetic region of the calorimeter, while pions can interact in both regions, notably the hadronic one, leading to longer shower successions and hence longer edges.

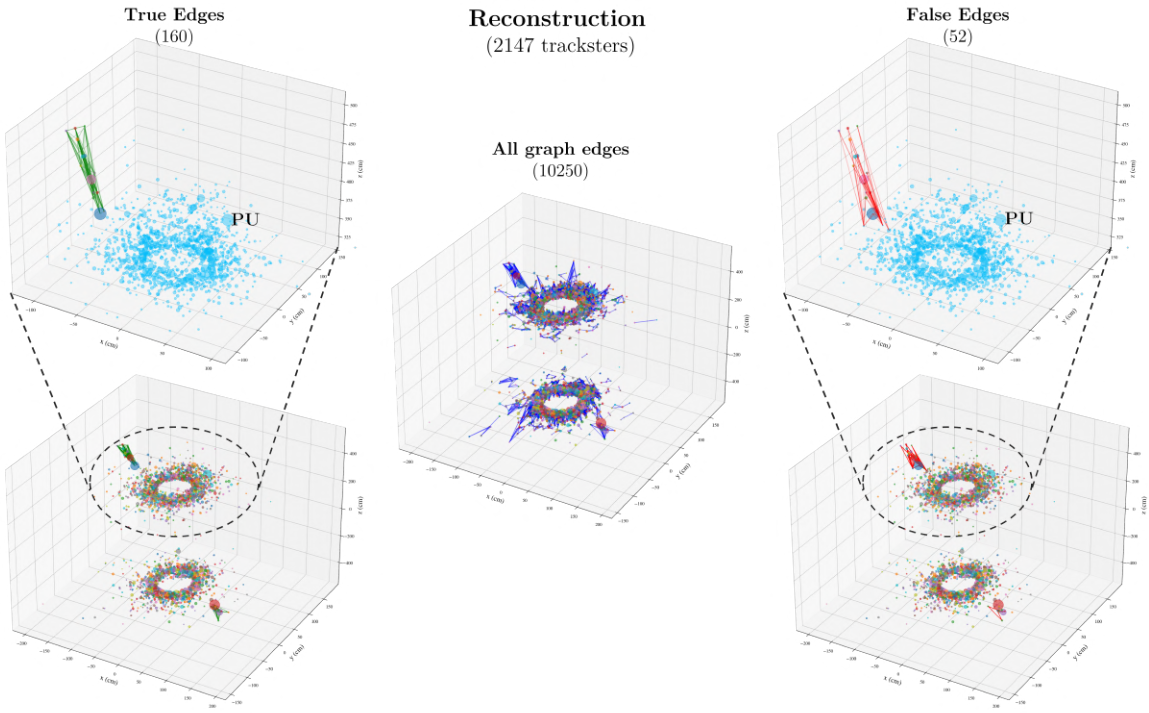
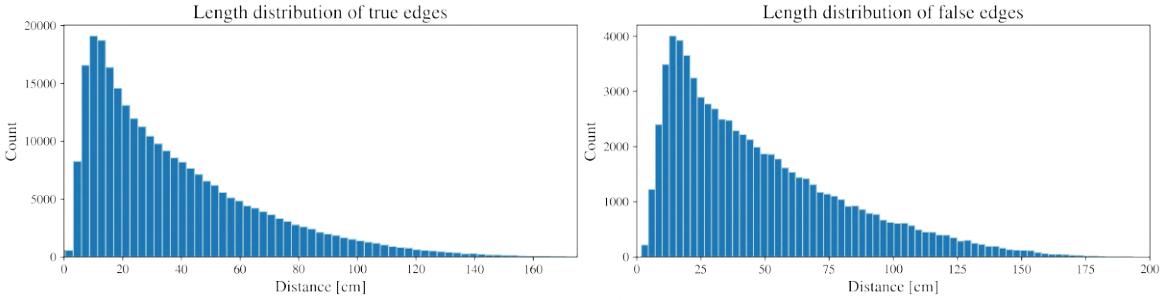


Figure 5.7: Event of a single pion in 140 PU with 160 true edges in green (on the left), all 10250 edges in blue (middle), and 52 false edges in red (right) coming from the $\eta - \phi$ graph visualized. Unlike in previous pictures, each point is a trackster with a size proportional to its energy. Note that the z axis points up for a better visualization perspective. Zoom-in versions of the true and false graphs are presented, with PU tracksters shown in blue.

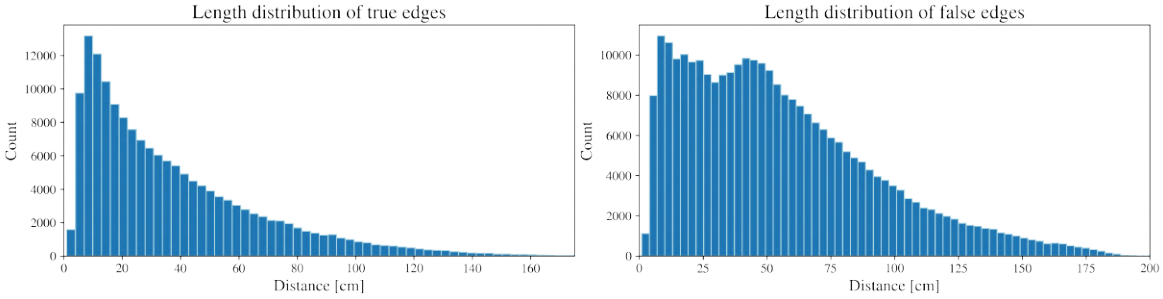
Table 5.2: $\eta - \phi$ graph-related properties of the three linking datasets. Edges refer to the edges in the constructed 0.2 $\eta - \phi$ bounded graph. Due to significant differences in properties among the particles in `SingleParticle140PU`, their values are presented separately.

Properties	`CloseByPions0PU`	`Multiparticle0PU`	`SingleParticle140PU`
Peak of length distribution of true / false edges	9 cm / 14 cm	10 cm / 17 cm	Electron: 10 cm / 12 cm Tau: 4 cm / 3 cm Pion: 7 cm / 10 cm Kaon: 15 cm / 25 cm Photon: 6 cm / 12 cm
95% percentile of length distribution of true / false edges	96 cm / 116 cm	98 cm / 119 cm	Electron: 16 cm / 21 cm Tau: 91 cm / 113 cm Pion: 98 cm / 131 cm Kaon: 92 cm / 117 cm Photon: 18 cm / 25 cm

The distribution of associator scores in relation to edge lengths is also examined in Figure 5.9. The majority of well-associated tracksters for true edges are found below 50 cm distance. Similarly, for false edges, most of the worst-associated tracksters are also located in regions below 50 cm.

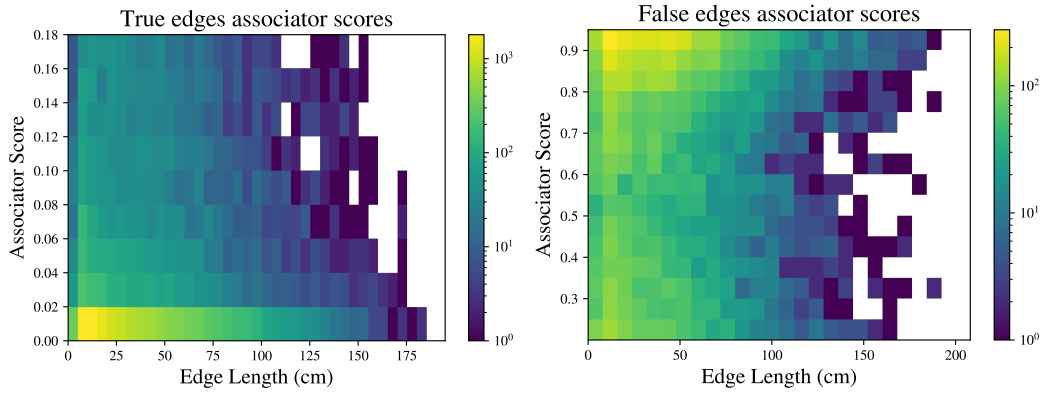


(a) : Distribution of true and false edge lengths for 1500 events from the double pions in 0 PU dataset. True edges: mean length of 38 cm, std 29 cm, minimum length 0.5 cm, maximum 193 cm. False edges: mean 48 cm, std 34 cm, minimum 1.7 cm, and maximum 193 cm.



(b) : Distribution of true and false edge lengths for 1500 events from the single pion in 140 PU dataset. True edges: mean length 37 cm, std 30 cm, minimum length 1.0 cm, maximum 210 cm. False edges: mean 56 cm, std 38 cm, minimum 1.0 cm, maximum 213 cm.

Figure 5.8: True and false edge length distributions in $\eta - \phi$ event graphs.



(a) : True edges.

(b) : False edges.

Figure 5.9: Histogram of edge length and Reco-to-Sim associator scores for 5500 events of a single pion in 140 PU. A lower associator score indicates a better association between tracksters. If the score is less than 0.2, we consider the tracksters to belong to the same CaloParticle.

Trackster Fragmentation. Figure 4.1 from the preceding section demonstrates the phenomenon of trackster fragmentation showcased with their energy profiles along the detector layers. The longitudinal energy profiles obtained by aggregating deposited energies in each detector layer are analyzed with respect to the true energy profile of the simulated particles. To determine the simulated particle corresponding to each reconstructed trackster, association scores are utilized. It can be seen that even the perfect linking of tracksters cannot restore the original energy distribution perfectly, as some of the hits are eliminated during the reconstruction process, and some LCs may already have been wrongly merged to different tracksters before the linking step.

The event displayed in Figure 4.1 demonstrates two typical cases of recotrackster fragmentation. Simtrackster A (blue) is composed of a primary recotrackster containing most of the energy and additional low-energy tracksters (generally, with energies below 10 GeV). As for simtrackser B, it exemplifies a major trackster splitting due to varying energy densities at the border of the CE-E and CE-H sections of the detector, which is a common occurrence for hadronically interacting particles.

The impact of fragmentation is particularly significant in pion reconstruction data, as the longer shower lengths and the differences in shower density in the electromagnetic and hadronic parts of the HGCAL exacerbate the fragmentation effect. Lower-energy photon and electron reconstruction, on the other hand, exhibits superior performance, as their shorter and well-aligned showers result in the reconstruction of their entire showers with only a small proportion of low-energy miss-aligned fragmented tracksters.

5.5.1 Final Dataset Parameters

The scenario of two nearby particles is a fundamental case, allowing for the examination of fragmentation properties and algorithm tuning. The case of multiple particles poses a challenge to the algorithm's scalability, requiring it to handle multiple particles of varying types simultaneously. Lastly, the pile-up scenario simulates a high-overlap production environment to test the proposed approach's practical applicability.

As evidenced by Table 5.1, the datasets exhibit an assortment of sizes. Of note, the dataset with the highest particle density, featuring multiple particles, possesses merely 29.5k events. Conversely, the lower density double pion dataset is constituted of 58k collision events. Due to the elevated computational demands, the pile-up dataset has a relatively reduced quantity of events – just 21.7k. The construction of datasets of such proportions without the presence of pile-up entails several hours of computing time using the globally distributed CMS data analysis infrastructure, whereas the pile-up dataset requires several days for generation and post-processing.

For the double pion dataset, each event has an average of 28.2 tracksters, resulting in a relatively low number of edges per event (230). Therefore, a larger dataset size is preferred in this scenario. In contrast, for multi-particle events, the average number of tracksters per event is much higher at 253.1, resulting in an average number of edges of 2805. Thus, fewer events are needed to achieve the same total number of tracksters (even fewer so for edges) as in the previous case. Overall, for these two datasets, we obtain a total order of $\mathcal{O}(10^6)$ tracksters.

However, our ultimate objective is framed as an edge classification task, thus our primary concern lies in the quantity of edges, as opposed to the count of events or tracksters. A high

number of tracksters results in an even greater number of edges in event graphs. For $\eta - \phi$ graphs, which are mostly used in this study, we obtain approximately 10.7 million and 41.1 million edges for double pions and multiparticle datasets, respectively.

In the case of the pile-up dataset, the amount of relevant data for training in each event is relatively low, as we only construct a local graph around the hard-scattering trackster. On average, pile-up tracksters comprise over 98% of all tracksters in events, and their average number is around 2035 per event. Only a single simtrackster per endcap is generated for this scenario, and the corresponding reconstructed tracksters' average number varies significantly depending on the particle type. Pions are the most complex case, producing an average of 28.7 reconstructed tracksters (similar to the no pile-up scenario), while particles such as photons and electrons produce only around 5.5–5.8 reconstructed tracksters on average. However, tracksters produced by pile-up cannot be excluded from the dataset, as they must also be evaluated in the reconstruction, and their number is relatively high. Therefore, our final dataset contains 24.8 reconstructed tracksters related to the simulated particles and 2081 tracksters per event overall.

The pre-processing stage of the datasets presented a substantial bottleneck throughout the experimentation process. Even with the utilization of functions compiled to machine code, running at native machine code speed, the preparation of datasets of the sizes employed in this study requires two to three days. Considering the need for multiple iterations of the datasets to select relevant features and trying different strategies to graph building, the processing stage significantly hindered the progress of the experiments.

Dataset Imbalance

Another noteworthy aspect of Table 5.1 is the imbalance of datasets, with varying degrees of edge class distributions for each scenario. Specifically, in the double pion case, a positive imbalance is observed, with 78.4% of edges being true and only 21.6% negative. Conversely, in the multiparticle and PU scenarios, a negative imbalance is present, with only 41.2% and 39.8% true edges on average, respectively. Such substantial imbalance presents a challenge for models trained on a particular dataset, as their performance is likely to deteriorate when dealing with data from other datasets. Additionally, it presents a challenge for neural network training, as it can lead to prioritization of classifying all edges into one of the classes. One approach to addressing this problem is to randomly sample incorrect edges to balance the classes. However, this is not a viable option since these edges do not represent hard negative examples and may not contribute significantly to the network's learning. Moreover, for NNs, cumulative easy negatives loss oftentimes overwhelms the total loss, which degenerates the model. Therefore, we employ two strategies: first, we introduce edge scores (as in Section 5.4.6) instead of binary labels to improve label distribution in case of positive imbalance in the double pion dataset. Second, we utilize a specialized balanced loss function that is discussed in further detail in subsequent chapters (Section 6.4).

Chapter 6

Methodology

While the geometric linking is based on an imperative, rule-based approach, using supervised ML to parametrically define linking based on simulation samples may increase the physics reach of the experiments by permitting a more thorough reconstruction. Additionally, as highlighted in the previous chapter, ML-based algorithms seem better suited to the uneven, high-granularity HGCAL geometry. As a result of careful considerations based on the previously published studies, this work primarily focuses on supervised link-level prediction using GNNs in artificially constructed graphs of collision events. This chapter overviews graph neural network concepts relevant to this work, introduces the proposed architecture, and explains the design choices behind it. Finally, strategies for evaluating the reconstruction performance of the candidate solutions are described.

6.1 Overview of Graph Neural Networks

As discussed in the previous chapter, GNNs are a type of deep learning architecture designed to learn functions on graphs. Their utilization is widespread across various machine-learning problems related to graphs, including node-wise classification or regression, graph classification, link prediction, anomaly, and community detection. Given that the trackster collections can be easily represented as graphs, it is plausible to employ GNN methods for linking. GNNs aggregate neighborhood information and adapt to uneven detector geometry, which is highly suitable for this task, albeit with greater temporal complexity for larger adjacency matrices. For this reason, the present work concentrates on GNN linking at the trackster level, in contrast to other works on GNN calorimetric clustering, which commence with hits. The process of creating tracksters from hits is fast and reduces the problem input sizes' complexity from $\mathcal{O}(10^5)$ to just $\mathcal{O}(10^3)$, leading to a four-order-of-magnitude reduction in adjacency matrix sizes.

GNN types. GNN architectures are classified into four primary branches: recurrent (RecGNNs), convolutional (ConvGNNs), spatial-temporal (STGNNs), and graph autoencoders (GAEs). In RecGNNs, nodes repeatedly exchange messages with their neighbors until a stable equilibrium is reached. GAEs, on the other hand, encode nodes or entire graphs into a latent space and recover graph data from the encoded information in an unsupervised fashion. Meanwhile, STGNNs aim to learn hidden patterns from spatial-temporal graphs. ConvGNNs

are the most commonly used variation among GNNs, extending the concept of convolution from grid data to graphs, generating representations of nodes by aggregating their own features with the features of their neighbors. Unlike RecGNNs, ConvGNNs employ multiple graph convolutional layers to extract high-level node representations. Notably, ConvGNN models such as ParticleNet have shown excellent performance in relevant problems within the HEP field. As such, this section provides a detailed review of ConvGNNs' operation.

Graph Representation. The raw detector data is essentially an unordered set of N objects. However, the set can be supplemented into a graph by considering N_e geometric or physical links between objects. In terms of the linking problem, nodes in the graph represent tracksters, and the edges indicate their pair-wise relationships. Such graph representation is given by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of N nodes and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of n_e edges. Each node in the point cloud $\mathcal{V} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\} \subset \mathbb{R}^F$ contains F node properties, and each edge can have F' edge attributes.

The relationship between the nodes is classified as either directed or undirected. It can be represented by an adjacency matrix, which is a binary $N \times N$ matrix (usually sparse) whose elements indicate whether one vertex is linked to another. An edge index in the coordinate list format, defined as a $2 \times N_e$ matrix with each column containing the node indices of each edge, is an alternative to adjacency matrix encoding of the graph. Although this compact representation is advantageous for incremental matrix generation and smaller memory requirements, a conversion to a compressed sparse or dense format is frequently required for arithmetic operations.

GNN Graph Building Approaches. At the data pre-processing stage, the vertices in a graph-based network are often linked based on predetermined criteria, such as the k-NN graph or a fully-connected graph for small input sets, allowing the network to learn all object relations. Graph edges then describe the pathways for information flow. Edges may also be created using learned representations, such as those employed by the DGCNN or GravNet architectures. In the mentioned architectures, the graph is not kept static, but rather the k-NN search is performed in each convolution block using the latent features. This way, rather than treating the graph $\mathcal{G}$ as a fixed constant, the architects learn to produce a different neighborhood graph $\mathcal{G}^t = (\mathcal{V}^t, \mathcal{E}^t)$ at each graph convolutional layer. Nevertheless, the repetitive k-NN search suffers from the curse of dimensionality due to the high-dimensional feature spaces. Because of the growing intricacy of calculating distances between data points and the decline in the reliability of Euclidean distance with higher dimensions, utilizing large feature vectors becomes impractical. ML packages often employ naive k-NN graph implementations with $\mathcal{O}(n^2)$ time complexity. Although more efficient implementations, like k-d trees, exist for k-NN graph construction, I am unaware of a differentiable implementation that could be used during network training. To address this issue, an alternative approach adopted in this study is to create a static graph beforehand and maintain its consistency during the training process. This technique is much less computationally demanding as it only performs graph construction once. However, it does not incorporate an advantage of DGCNN design – creating connections between the nodes that are similar in the obtained latent representation, even if they are far in the original feature space. In such case, proximity in feature space differs from proximity in the input feature space, resulting in non-local diffusion of information throughout the point cloud.

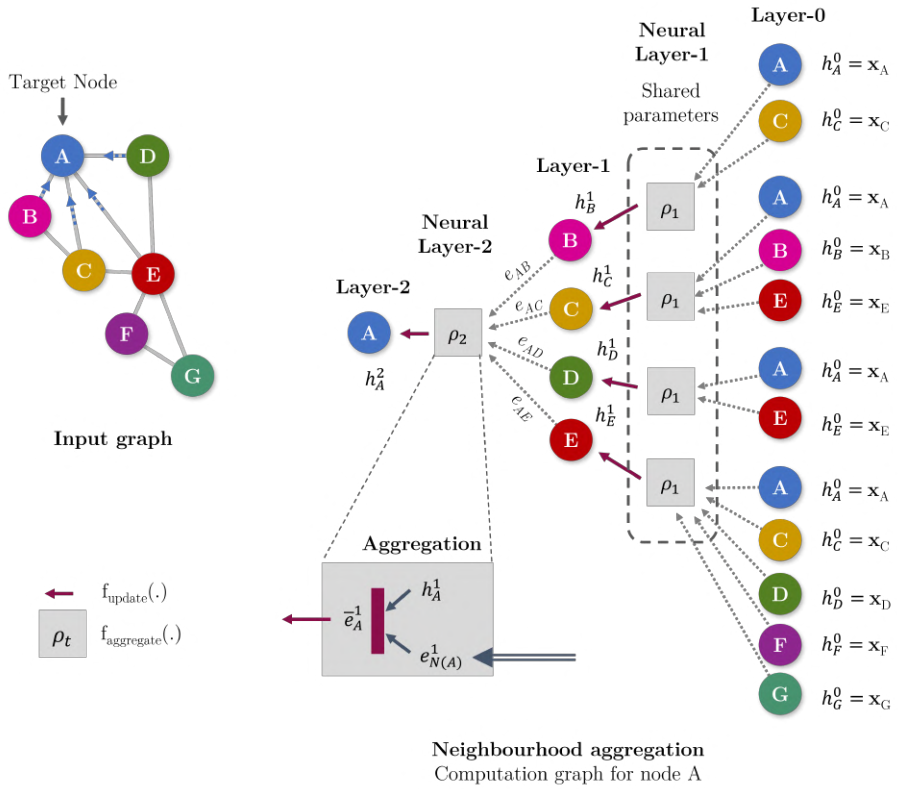


Figure 6.1: Given an input graph on the left, graph convolution outputs the embedding of the target node (e.g., the blue node A) by aggregating the information from neighboring nodes, as shown in the computation graph on the right. The nodes' level-0 hidden states (initial embeddings) are denoted as $\mathbf{h}_i^0$, and the embeddings after the first neural layer as $\mathbf{h}_i^1$. The aggregated embedding of the target node's neighbors is indicated by $\bar{\mathbf{e}}_{N(A)}^1$. Combining these embeddings and passing them through the node's activation unit creates a new embedding for node A, containing information about its neighbors. With each layer of message passing, the node learns more about its neighborhood and distant neighbors. It should be noted that the previous state of the element being updated is also used in its update (self-loop).

Message Passing. GNNs operate by passing messages across a graph, which, as illustrated in Figure 6.1, consists of two main steps: *aggregations* and an *update*. First, the edge update function calculates updated edge attributes, also known as *messages*, based on the edge $\mathbf{e}_{ij}^{t-1}$ and node attributes $\mathbf{h}_i^{t-1}$ at the $(t-1)$ message-passing level, with a set of learnable features Θ:

$$\mathbf{e}_{ij}^t = \phi_{t,\Theta}^e \left(\mathbf{e}_{ij}^{t-1}, \mathbf{h}_i^{t-1}, \mathbf{h}_j^{t-1}\right) \quad . \tag{6.1}$$

Next, the calculated messages are aggregated per node by considering either all edges or a subset of edges incident on this node i given by $N(i) = \{j_1, \ldots, j_{n_i}\}$:

$$\bar{\mathbf{e}}_i^t = \rho_{t,j:N(i)} \left(\mathbf{e}_{ij_1}^t, \ldots, \mathbf{e}_{ij_{n_i}}^t, \mathbf{h}_i^{t-1}\right) \quad . \tag{6.2}$$

Finally, the node update function

$$\mathbf{h}_i^t = \phi_{t,\Theta'}^v \left(\bar{\mathbf{e}}_i^t, \mathbf{h}_i^{t-1}\right) \tag{6.3}$$

uses the aggregated edge attributes and the current node attribute to compute each node's updated hidden states $\mathbf{h}_i^t \in \mathbb{R}^{F_t}$. The specific functions for update and aggregation depend on the GNN architecture. The update functions $\phi_{t,\Theta}^e$ and $\phi_{t,\Theta'}^v$ are typically implemented as trainable NNs, while the aggregation functions $\rho_{t,j:N(i)}$ are permutation invariant and take a variable number of arguments (i.e., functions such as mean, maximum, minimum, sum, etc.).

6.1.1 Edge Convolution

A particular message-passing mechanism, whose modified version is used in this work, is the Edge Convolution [WSL+19]. The EdgeConv update function for the hidden states is expressed as follows:

$$\mathbf{h}_i^t = \rho_{j:N(i)} \left(\phi_{t,\Theta} \left(\mathbf{h}_i^{t-1}, \mathbf{h}_j^{t-1} \right) \right) \quad , \tag{6.4}$$

where ρ is the channel-wise invariant aggregation function (usually sum or maximum), $\phi_{t,\Theta} : \mathbb{R}^{F_{t-1}} \times \mathbb{R}^{F_{t-1}} \to \mathbb{R}^{F_t}$ is a nonlinear edge update function with a set of learnable parameters Θ. According to the terminology used in the preceding section, $\phi_{t,\Theta}$ can be considered as a variant of $\phi^e_{t,\Theta}$ without employing edge features for the update. The node update function ϕ^v_t is an identity, directly returning the aggregated edge attribute as a new h^t_i.

Graph convolutional networks discussed so far suffer from the limitation of assigning equal importance to every neighbor, which is not desirable as some nodes are more essential than others. To address this, an attention mechanism assigning a weighting factor to each connection to account for the varying importance of different nodes is leveraged.

6.2 Explored ML Approaches

A progressive approach was taken in the experiments involving the NN linking models, starting with simpler architectures and gradually increasing their complexity. Specifically, the first set of experiments involves a fully connected pairwise network, followed by the use of GNNs, culminating with the addition of attention mechanisms. The details of these experiments are described in the following sections. Both networks can be applied either on the whole graph of the event or a selected region.

6.2.1 Linking Problem Framing

This work frames the trackster linking problem as a clustering task tackled as a supervised link-level classification using NNs in artificially constructed graphs of collision events. Specifically, the proposed approach involves training a classifier to predict the presence or absence of a link between two tracksters based on their respective embeddings.

At first glance, using link prediction on a graph with the ultimate goal of clustering might seem unreasonable, since it does not directly address clustering. However, the neighborhood aggregation approach employed by GNNs allows for deciding whether two tracksters should be merged based on the conditional probabilities of other nodes being connected. This is in contrast to a simple Multi-Layer Perceptron (MLP), making predictions independently for all trackster pairs defined by the presence of an edge in the input event graph. Edge classifiers for differentiating a variable number of objects in the data are not new in the HEP field and are shown to be a rather effective approach [FCM+18, JFC+20]. However, it also entails a set of rigorous constraints. Firstly, all potentially true edges need to be included in the graph during the pre-processing stage to enable classification by the network since the model cannot create edges not present in the input event graph. Finally, to construct the relevant object, the same connections must be reassessed again after the network inference by imposing a

threshold on the connection score. Possibly, a post-processing step, such as a graph search, might also be required. Finally, the binary nature of edge classification limits its utility in scenarios with significant overlaps and fractional assignments.

6.2.2 MLP Pair-Wise Linking

At the outset of the experiments, a simple multi-layer feed-forward neural network, also known as MLP, is employed. This strategy facilitates an exploration of the trackster linking probability in a pair-wise manner without recourse to their neighborhood information. The underlying problem is framed as a binary classification task, whereby the model must make a binary decision on whether two given tracksters should be connected.

This pair-wise approach serves a triple purpose: it is used as a baseline for subsequent experiments with GNNs, helping to determine the significance of neighborhood information; additionally, it enables an initial pre-selection of edges in scenarios with large graphs, where some edges can be pre-pruned before being inputted into the GNN; finally, part of this model may serve as a pre-trained component of the future GNN model.

Model Architecture. The MLP-based trackster linking network architecture is depicted in Figure 6.2 created in Netron[1] web application. The network has two inputs (node features and edge index) and two outputs (node embeddings and edge predictions). The node feature dimensionality is $N_B \times \mathrm{F}$, where N_B is the total node number in all the graphs in a batch of size B; $F = 33$ is the node feature dimension. The edge index is provided in the coordinate list format with dimensionality $2 \times N_{e,B}$, where $N_{e,B}$ is the number of edges in the whole batch.

The network consists of two modules – node encoder and edge classifier. The former utilizes two or three hidden layers (depending on the training dataset size) for creating latent node representations of dimension $F' = 256$, which are then concatenated according to the edge index to create edge embeddings of dimension 512. Those are then passed to the edge classifier module comprising another three hidden layers (with output sizes 256, 128, 1) and the final sigmoid activation. The model's output is, therefore, a single value in the range from 0 to 1 for each trackster pair defined by the edge index ($E_B \times 1$), which can be thought of as their linking probability. To prevent overfitting, dropout layers with probability 0.2 are used between the fully connected (FC) layers in both node embedding blocks and the edge classifier. Except for the final sigmoid activation, the FC layers are interleaved with LeakyReLU [XWCL15] non-linearities. The hyperparameters, such as layer sizes, non-linearity types, dropout rates, and network depth, were determined through careful hyperparameter tuning. Apart from edge predictions, the network also outputs edge embeddings ($N_B \times F'$) for their visualization and analysis.

6.2.3 General Considerations and Design Choices

Both MLP and GNN models adhere to a set of common guidelines for data pre-processing, model design choices, and inference, which are described in this subsection.

[1] Netron URL: https://netron.app

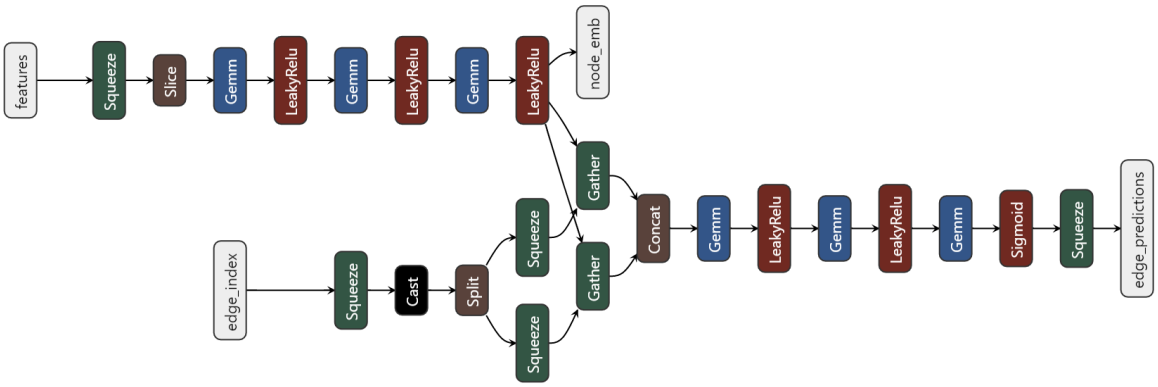


Figure 6.2: MLP linking model architecture. The network has two inputs: node features ($N_B \times F$) and edge index ($2 \times E_B$); and two outputs: edge predictions ($E_B \times 1$), and node embeddings ($N_B \times F'$). The input node features are first standardized across the events and passed to the node encoder – a sequence of three FC layers (represented as `Gemm` in the scheme, which stands for the general matrix multiplication) interleaved with LeakyReLU activations and Dropout (not shown in the figure). At the end of this pipeline, we receive latent representations of the nodes, making one of the network outputs. These embeddings are concatenated according to the provided edge index to create edge embeddings, passed to the link predictor module. Link predictor comprises a similar block as the one creating node embeddings, except it finishes with the Sigmoid activation at the output. The output after the sigmoid activation is the tensor of edge predictions between every two pairs of tracksters defined by the edge index. The figure was generated using a model in evaluation mode exported to ONNX format and visualized in Netron app. Some features are standardized differently from others, as described in the text (represented by Slice blocks at the input).

Mini-Batching. Instead of processing event graphs one at a time, mini-batching groups a set of examples into a single representation, which can be processed in parallel. Typically, for example, in the image domain, this involves resizing or padding each input to conform to a fixed shape, prior to being stacked along an additional dimension in a mini-batch. This way, the batch size, referring to the first dimension of the NN input, can differ in size while the inner dimensions of the input matrix are preserved. However, the events have highly diverse sizes depending on the scenario. For example, the number of input nodes ranges from just two tracksters in the double pions case to as many as 2500 tracksters in the 140 PU dataset, making the above-mentioned batching approaches inefficient or resulting in excessive memory consumption. Hence, preserving the inner dimensionality in the number of nodes and edges becomes problematic once multiple graphs are processed within the same batch.

A solution to the mini-batching problem is to perform an input transformation $(B, N, F) \rightarrow (N_B, F)$, where $N_B = \sum_{i=1}^{B} N_i$ is the total number of graph nodes in the mini-batch. This approach involves stacking adjacency matrices in a diagonal manner, creating a single graph containing multiple isolated subgraphs; similarly, node features are concatenated in the node dimension. To ensure that the individual graphs do not interact with each other, an offset to the edge indices is introduced to ensure that they correctly identify the nodes in the concatenated graph. The information about the initial sizes of individual graphs and the batch indices of nodes and edges is also retained. This extra batch index is taken into account in all aggregation operations, including EdgeConv used in the GNN model, as described in the subsequent section. The same principle applies when the graph includes additional fields or an edge index. These fields are remapped to the new indices in the merged graph set to prevent point operations that aggregate neighborhood information from overlapping with

other event samples. As a result, the final mini-batch is fed into a model in the same way as a single larger graph.

Compared to other batching techniques, this approach has significant advantages. Firstly, GNN operators that rely on message passing do not need modification since nodes in different graphs cannot exchange messages. Secondly, this method has no computational or memory overhead, as it does not require any padding of node or edge features. Moreover, adjacency matrices are used in a sparse format, with only non-zero entries stored, and stacking them diagonally does not involve additional memory consumption.

Standardization. To facilitate easy integration into the CMSSW environment, the model needs to function as a black box that can be loaded into the environment with minimal data manipulation in C++. Therefore, we opt to keep the feature pre-processing and standardization as part of the model. Unfortunately, some of the input features require distinct standardization, making it impossible to use layer normalization [BKH16]. As such, pre-processing of time-based data necessitates additional feature cleansing due to the abundance of missing values in low-energy tracksters, as mentioned in Section 5.4. These missing values are assigned a value of -99, and hence, standardization is performed solely on the available values, while the absent ones are set to a negative constant ten standard deviations away from the mean. To handle event density in terms of LCs and tracksters per layer, standardization is not suitable since these values are uniform for every node in the event. Instead, we aim to rescale these values to a meaningful range which is achieved by dividing the number of LCs and tracksters by their expected order in 140 PU[2].

Normalization across batches is inappropriate since events' independence must be preserved. Moreover, the events are processed individually in the production environment, making the batch size always one when not using the seeding regions. Hence, data is normalized only within events rather than across the entire dataset or a batch.

ONNX Runtime Inference. Apart from the above, the model has to be adapted for the Open Neural Network Exchange[3] (ONNX) [BLZ+19] format exportation. ONNX is an open-source format enabling interoperability between different deep-learning frameworks. As such, it allows models to be trained in one deep learning framework and then exported to another for inference without the need for re-implementation. ONNX defines a common set of operators and data types, allowing models to be moved between frameworks such as `PyTorch`, `TensorFlow`, and `Caffe2`, among others. Since the models in this work are trained in Python with the use of `PyTorch` [PG+19] library, while the production TICL pipeline in CMSSW is implemented mostly in C++, we choose to convert the models to ONNX format for direct inference in C++.

CMSSW supports ONNX Runtime inference capable of running multiple threads, which are assigned to process events in the event loop using the first-come-first-serve principle. When the model is loaded into ONNX Runtime, it is converted into an in-memory representation. To minimize memory usage, a single global ONNX Runtime session is shared among all threads, ensuring the model has only one copy in memory. Then, when multiple threads request inference with their input data, the session accepts and serializes those requests and produces output data.

[2]The LC number is divided by 10^4, while the trackster number by 10^3.
[3]ONNX url: `https://onnx.ai/`

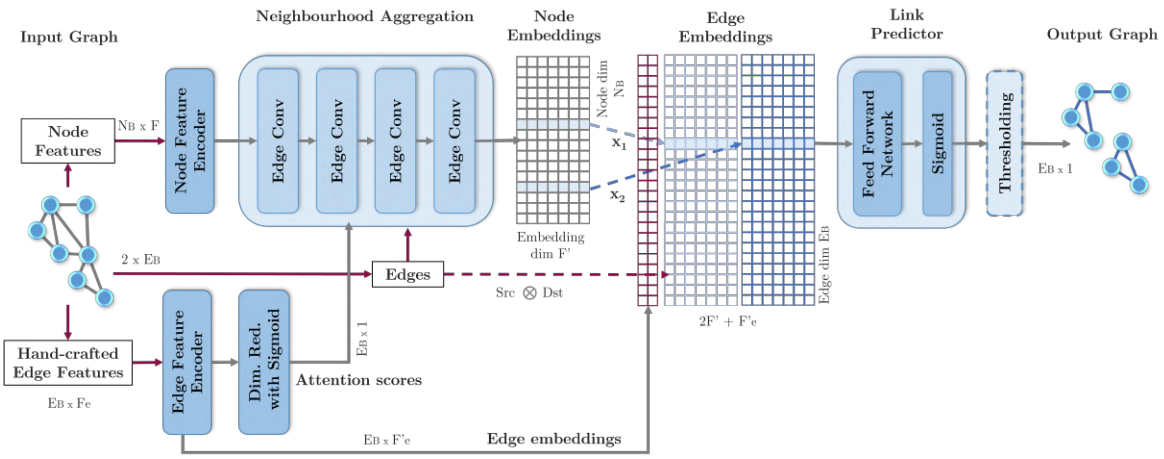


Figure 6.3: A schematic representation of a GNN linking network is shown. The network receives three inputs: node features, edge features, and edge index, and outputs individual node embeddings and prediction scores for each edge in the graph. Edge scores are then thresholded to obtain the final output graph. To begin with, both node and edge features are standardized and inputted to the respective latent space encoders – fully connected networks with LeakyReLU activations and dropout. The edge features are then utilized to create attention scores per every edge, which are further leveraged during the neighborhood aggregation process in four consecutive EdgeConv blocks with skip-connections. The final edge embeddings are constructed by concatenating the latent representations of the hand-crafted features with the respective node embeddings. Such embedding is created for every edge in the graph and passed to the link predictor module, a two-layer FC network with a sigmoid output activation, producing edge predictions.

However, exporting models to ONNX format requires careful consideration due to several potential pitfalls. For instance, certain `PyTorch Geometric` library modules, such as pre-implemented Edge Convolution, cannot be exported to ONNX and thus require custom implementations. It is necessary to avoid if conditions, for loops, and in-place operations in model implementation. This necessitates the careful design of the custom modules with the use of matrix or indexing operations only. Certain tensor indexing patterns are also not exportable. Once the model is successfully exported, the resulting `.onnx` file contains a binary protocol buffer with both the network structure and parameters of the exported model, which can be directly utilized in the CMSSW environment.

6.2.4 GNN Linking

Proceeding beyond the simple MLP model, the motivation for incorporating trackster neighborhoods stems from the premise that bringing in additional information about the shower might improve the efficacy of edge classification. The GNN model builds on top of the pair-wise MLP approach, where the relationship between a pair of tracksters is predicted. For neighborhood aggregation, custom-implemented EdgeConv-based message-passing blocks, exportable to ONNX and extended with an attention mechanism, are utilized.

6.2.5 Model Architecture

The schematic GNN Linking network architecture is depicted in Figure 6.3. The model operates through a sequence of modified EdgeConv layers with attention, each layer progres-

sively incorporating information from the latent neighborhood in the graph. Following the neighborhood aggregation, the features corresponding to each node and edge are subsequently processed via dense layers to produce the link-prediction output. In the forthcoming section, I describe the specifics of this architecture.

Network Inputs. For a batch of events with a total of $N_B = \sum_{i=1}^{B} N_i$ tracksters interconnected with E_B edges, GNN Linking model makes use of three sets of inputs:

- The node feature input $\mathbf{X}$ includes a tensor of $F = 33$ features for every trackster in a mini-batch and forms a tensor of shape $(N_B \times F)$, much like in the MLP architecture.
- The interaction input $\mathbf{U}$ is a tensor of F_e (here, $F_e = 8$) features for every pair of interacting tracksters, in shape $(E_B \times F_e)$. The term interaction (edge features) here refers to any feature involving a pair of tracksters, which may or may not be related to the physical forces between them.
- Graph configurations $\mathbf{E}$ defined in terms of an edge index in the coordinate list format, which is a $(2 \times E_B)$ tensor. It should be noted that the original edge indexes of individual events in the batch are re-indexed to create a batch index (in accordance with Section 6.2.3) suitable for the parallel processing of multiple graphs.

Standardization and Pre-Processing. As specified in section 6.2.3, the node and edge input features undergo heterogeneous pre-processing and standardization across the events at the network input based on the nature of each feature. Moreover, logarithmic transformations (with protection against the negative values) are applied to features with long-tailed distributions, such as raw energy, raw EM energy, ΔE, and Δz separation.

Latent Space Projections. The node and interaction input features pre-processing are each followed by MLP networks projecting them to a F'- and F'_e-dimensional embeddings, $\mathbf{x}^0 \in \mathbb{R}^{N_B \times F'}$ and $\mathbf{u}^0 \in \mathbb{R}^{E_B \times F'_e}$, respectively. Both projections are performed using two dense layers interlaid with LeakyReLU activations and Dropout layers. Notably, the *node* encoder shares the same architecture as the respective MLP sub-module. Thus, a pre-trained MLP node encoder is utilized to pre-initialize the weights of the one used in the GNN architecture. In this work, the node hidden dimension F' is set to 64, providing a 64-dimensional latent representation for every graph node. Meanwhile, the *edge* feature encoder, which handles the hand-crafted edge features, shares the same architecture but uses a smaller edge hidden dimension F'_e of 32.

Self-Attention. The presence of imbalanced datasets containing numerous negative edges, coupled with the fact that each neighboring node may exert varying degrees of influence on the target node, serves as a strong motivation for incorporating attention scores into the GNN architecture. The present work calculates edge attention scores $\boldsymbol{\alpha}$ exclusively from the latent projections of hand-crafted edge features $\mathbf{u}^0$. These features serve a dual purpose: first of all, they are appended to the edge embeddings created by concatenating individual node embeddings, thus providing supplementary features. This skip-connection was empirically found to improve model performance. Second, they undergo dimensionality reduction via a sigmoid-activated dense layer (i.e., linear transformation, parametrized by an attention weight vector $\mathbf{a}_\Theta : \mathbb{R}^{F'_e} \times \mathbb{R}$) to produce a single score α_{ij} per each edge direction ranging from 0 to 1:

$$\alpha_{ij} = \frac{1}{1 + \exp(-\mathbf{a}_{\Theta}^{T}(\mathbf{u}_{ij}^{0})^{T})} \in [0, 1]. \tag{6.5}$$

These scores are referred to as attention scores and are learned from the input features. They serve as weights to modulate the neighbors' contribution during message passing. In our formulation, the scores are kept constant during neighborhood aggregation iterations and are not updated based on the node embeddings learned in each iteration. This serves as a form of regularization to prevent overfitting and improve the generalization performance of the model. By keeping the score static, the model also saves computation time in each iteration by re-using the same score, which leads to a faster training and inference time, particularly when dealing with large graphs, as in our problem.

Nonetheless, maintaining a static attention score for all neighborhood aggregations may not be the most effective strategy as it fails to adapt the nodes' contributions according to the graph's local structure at each step. To examine the impact of this approach, experiments were carried out to compare the static attention score strategy with an alternative method inspired by GAT. In GAT, attention scores are computed by concatenating the most-recently transformed node embeddings for the adjacent nodes and passing them through a shared attention mechanism, providing a single real number. This mechanism applies a softmax function to the dot product of a learnable weight matrix $\mathbf{a}$ and the concatenated projected node features $[\mathbf{W}\mathbf{h}_i^t \,||\, \mathbf{W}\mathbf{h}_k^t]$, producing a set of attention coefficients for each edge:

$$\alpha_{ij}^{t} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^{T}\left[\mathbf{W}\mathbf{h}_i^t \,||\, \mathbf{W}\mathbf{h}_j^t\right]\right)\right)}{\sum_{k\in\mathcal{N}(i)} \exp\left(\text{LeakyReLU}\left(\mathbf{a}^{T}\left[\mathbf{W}\mathbf{h}_i^t \,||\, \mathbf{W}\mathbf{h}_k^t\right]\right)\right)} \quad . \tag{6.6}$$

This allows the model to dynamically adapt the importance of neighboring nodes at each layer t of the network based on the current representation of the nodes, providing a more adaptive approach compared to static attention scores. However, since the dataset in this study comprises large graphs, this method requires more computation and does not yield significant improvements. As a result, it was concluded that retaining the same weight scores for every EdgeConv iteration is more reasonable.

As observed in Equation 6.6, the softmax function ensures that the attention scores sum up to one, endowing them with the interpretation of a probability distribution over the neighboring nodes. Nevertheless, in the linking problem, we strive to have attention scores in the interval from zero to one, without necessarily achieving probability distribution interpretation, as several tracksters may have a comparable strong influence on the target trackster. As a result, the sigmoid activation function is opted for.

It is also important to note that the effect of trackster j on trackster i does not necessarily imply that the effect of i on j is equal, requiring the attention weight coefficients α_{ij} to be asymmetric. To address this, two sets of attention weight vectors are employed in this work, $\mathbf{a}_{\Theta_{dir}}$ and $\mathbf{a}_{\Theta_{rev}}$, for each direction in message-passing, utilizing the same edge feature vector as their input according to Equation 6.5.

Edge Directionality. As the edges in the input graph do not have any inherent directionality, they are converted into an undirected form before message passing is applied. Additionally, self-loops are introduced to allow for a node's own information to be incorporated into its update during message passing.

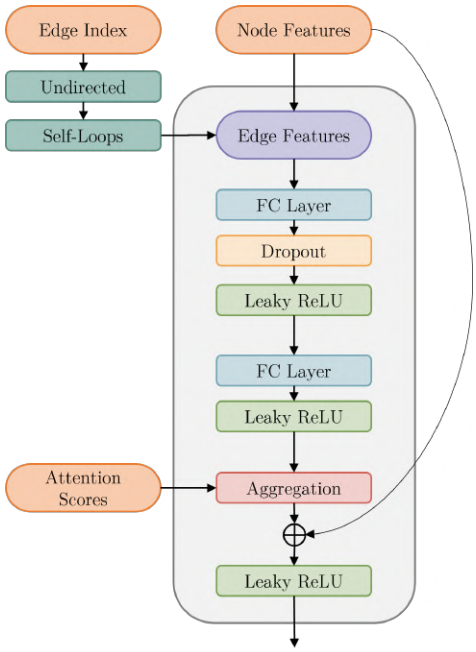


Figure 6.4: Edge Convolution with attention schematic. Three inputs are provided to the block: undirected edge index with self-loops, node features, and attention scores. Edge features are created by concatenating the node features according to the provided edge index. The edge features are processed through a two-layer FC network and aggregated for each node with the corresponding attention scores. The aggregation output is then passed through an additional activation function.

Neighborhood Aggregation. In the neighborhood aggregation step, the initial trackster embeddings $\mathbf{x}^0$ are fed into a stack of $L = 4$ modified Edge Convolution blocks to generate new embeddings, $\{\mathbf{x}^1, \ldots, \mathbf{x}^L\}$. These custom Edge Convolution blocks use the addition aggregation function with learned attention scores $\boldsymbol{\alpha}$ and self-loops to update the node representations. Moreover, skip connections from the input node representations $\mathbf{h}_i^t$ are employed to ensure better stability of the node embeddings. The feature update is defined as:

$$\mathbf{h}_i^{t+1} = \mathbf{h}_i^t + \sum_{j \in \mathcal{N}(i) \cup \{i\}} \alpha_{ij} \cdot \mathrm{NN}(\mathbf{h}_i^t \,||\, \mathbf{h}_j^t - \mathbf{h}_i^t) \quad , \tag{6.7}$$

where $\mathcal{N}(i)$ represents the set of neighboring nodes of i and NN denotes a neural network operating on the concatenated node features. The attention scores for the node itself are set to $\alpha_{ii} = 1$, allowing the node to update its own representation. The NN is given by an MLP with two hidden layers with LeakyReLU activations and dropout after the first dense layer-activation block. It takes input of dimensionality $N_B \times 2 \cdot F'$ ($N_B \times 128$) and produces the output in a form $N_B \times F'$ ($N_B \times 64$). This way, applying the transformations on node features while aggregating their neighborhood information retains the input dimensionality on the output. A schematic of the edge convolution block used in this work is presented in Figure 6.4.

The number of neighborhood aggregation iterations serves to define the receptive field of a trackster. The objective is to enable a trackster to gather information not only from its immediate surroundings within the same shower but also from showers in its neighborhood, capturing global information and potentially resulting in a lower merging rate between

the showers with respect to the simpler pair-wise MLP. In this work, four iterations were experimentally deemed appropriate to ensure each node within the graph has access to relevant information up to four edges away from itself. This allows for sufficient coverage of the surrounding area and neighboring showers without becoming computationally prohibitive.

Aggregation Function. The aggregation function used in this study is the weighted summation. As shown by [XHLJ18], the summation aggregation is useful when the target node is expected to be influenced by multiple features in its neighborhood, and their combined effect is important, which aligns with the trackster linking problem. In contrast, mean aggregation captures the distribution of the elements; max aggregation proves to be advantageous in identifying representative elements, and min aggregation is useful when the target node is expected to be influenced by the weakest features among its neighbors.

Neighbourhood Graph. In this study, the neighborhood graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is kept static, with the same set of edges $\mathcal{E}$ we aim to predict. Empirical findings of this work suggest that maintaining the graph structure static, as opposed to recomputing it using nearest neighbors in the feature space generated by each layer as in the DGCNN [WSL+19], is more desirable. This is due to multiple considerations: the first is the challenge of determining the appropriate number of nearest neighbors. The network is expected to function in scenarios with varying levels of PU, with more neighbors required in denser scenarios. Secondly, not every node in the graph should have the same number of neighbors (i.e., the number changes depending on the local node density). Finally, apart from better computational complexity, a static graph allows limiting a set of edges to a physically meaningful set.

Edge Embeddings Construction. Following the enhancement of node embeddings with neighbor information, the next step involves the construction of edge embeddings $\mathbf{e}_{ij}$ for the edges of interest $(i, j) \in \mathcal{E}$ by aggregating the adjacent node representations $(\mathbf{h}_i^L, \mathbf{h}_j^L)$ and additional edge features $\mathbf{u}_{ij}^0$ as:

$$\mathbf{e}_{ij} = \rho\left(\mathbf{h}_i^L, \mathbf{h}_j^L, \mathbf{u}_{ij}^0\right) \quad , \tag{6.8}$$

where ρ is the aggregation function. In this work, aggregation is accomplished by concatenating the feature vectors: $\rho\left(\mathbf{h}_i^L, \mathbf{h}_j^L, \mathbf{u}_{ij}^0\right) = \left(\mathbf{h}_i^L \,||\, \mathbf{h}_j^L \,||\, \mathbf{u}_{ij}^0\right) \in \mathbb{R}^{1 \times (2 \cdot F' + F'_e)}$. Several alternative methods include computing node representations' mean, sum, or element-wise product.

Link Classification. The final edge embeddings $\mathbf{e} \in \mathbb{R}^{E_B \times (2 \cdot F' + F'_e)}$ undergo further processing by passing them through a link prediction block, which also follows a two-layer FC network architecture. The output of this block is then fed into a final sigmoid activation, producing an edge classification score per every edge in $\mathcal{E}$, representing its probability of being true.

Neighbourhood Aggregation Implementation. The expression for feature update presented in Equation 6.7 is performed through matrix operations rather than individual per-node computations. This permits the transformation and aggregation steps to be performed simultaneously for all nodes, leading to a more efficient implementation. However, since the number of neighboring nodes can vary across nodes, the implementation is rather tricky and requires additional indexing. The update of the feature vector $\mathbf{X}^t$ in the matrix form is calculated as follows:

$$\underset{N_B\times F}{\mathbf{X}^{t+1}} = \underset{N_B\times F}{\mathbf{X}^{t}} + \underset{N_B\times(N_B+2E_B)}{\mathbf{SRC}^T \odot \mathbf{\Lambda}^T} \times \mathrm{NN}\left(\underset{(N_B+2E_B)\times F}{\mathbf{SRC}\cdot\mathbf{X}^t \,||\, (\mathbf{DST}-\mathbf{SRC})\cdot\mathbf{X}^t}\right) \quad , \tag{6.9}$$

where $\odot$ is the matrix-vector element-wise product. In the above, matrices **SRC** and **DST** represent the stacked edge indices, guiding aggregation and the concatenation of the node neighbors:

$$\underset{(N_B+2E_B)\times N_B}{\mathbf{SRC}} = \begin{bmatrix}\mathbf{I}\\ \mathbf{S}\\ \mathbf{D}\end{bmatrix} \quad , \qquad \underset{(N_B+2E_B)\times N_B}{\mathbf{DST}} = \begin{bmatrix}\mathbf{I}\\ \mathbf{D}\\ \mathbf{S}\end{bmatrix} \quad , \tag{6.10}$$

where $\mathbf{I} \in \mathbb{R}^{N_B\times N_B}$ is the identity matrix corresponding to the nodes' self-loops, $\mathbf{S} \in \mathbb{R}^{E_B\times N_B}$ and $\mathbf{D} \in \mathbb{R}^{E_B\times N_B}$ are the edge source and destination matrices. We consider edges to be undirected, hence the duplicity. Finally, the neighbor weight vector is created through the stacking of the vector of ones (weights for self-loops) with the bidirectional edge attention:

$$\underset{(N_B+2E_B)\times 1}{\mathbf{\Lambda}} = \begin{bmatrix}\mathbf{1}\\ \boldsymbol{\alpha}_{dir}\\ \boldsymbol{\alpha}_{rev}\end{bmatrix} \quad . \tag{6.11}$$

The above matrices are very large and sparse, with N_B typically being around 2500 and E_B exceeding 10 000 for a single PU event. As a result, more efficient matrix multiplication methods must be employed. One possible solution is to use sparse matrix operations, but due to limited support for sparse operations in ONNX, an alternative approach had to be implemented. Instead of using matrix multiplication, the calculations are carried out much faster through indexing, which also avoids duplicating information in **SRC** and **DST** matrices.

6.3 Supertrackster Building

The proposed NN models infer edge scores, representing the probability for a pair of tracksters of coming from the same particle shower, for each edge in the input graph. A common approach to extracting the final clusters from such a graph is identifying its connected components after thresholding the edge scores and eliminating the edges whose scores are below the threshold. Connected components are defined as sets of vertices in a graph that are linked to each other through paths. Each connected component forms a cluster, referred to as a *supertrackster*, containing the recotracksters that the model predicts should be linked.

To identify connected components in a graph, a graph traversal algorithm, such as depth-first search (DFS), is utilized to explore the graph from each unexplored node. This algorithm starts by selecting a node and traversing as far as possible along each branch before backtracking. During traversal, each node is labeled as visited to prevent re-visiting. After exploring all the nodes in a component, the algorithm proceeds to the next unvisited node, repeating the process until all nodes in the graph have been visited.

The proposed networks, as demonstrated in the experimental section, may suffer from occasional edge misclassifications. As can be seen later, these are typically incorrect edges connecting low-energetic tracksters to higher-energetic ones. This can pose a potential issue for the DFS-based cluster-creating approach since a single false positive edge connecting two

otherwise well-separated particle showers (for example, through a low-energy trackster) could result in their erroneous merging. In the case of the MLP model, we mitigate this issue by post-processing of the model predictions. Specifically, for each low-energy trackster, a check is performed to determine whether removing any connection would result in creating new supertracksters above a given energy threshold. If this is the case, the edge is then removed accordingly. In contrast, in the GNN approach, trackster pairs are evaluated in a more holistic manner, facilitating the model to consider the possibility of a low-energy trackster connecting to multiple high-energy tracksters and distinguishing whether they belong to the same particle. As a result, the requirement of a low-energy trackster connecting only to a single high-energy trackster is relaxed.

Although the models' edge-wise predictions may be imprecise, it can be further seen that the model can still effectively capture the community structure of particle showers in both cases, describing the pattern of stronger connections within a community than between communities. By predicting continuous values of edge scores instead of binary true/false values, the model supplies valuable information about the structure of the collision event graph, which can be viewed not only as the network certainty but also as a strength of connections between the tracksters. Such scores can be harnessed to detect communities in the graph, a task commonly referred to as *community detection* in network science.

Community Detection. Several community detection algorithms have been proposed in the literature, including the Louvain algorithm [BGLL08], which is particularly suited for large graphs. The Louvain algorithm optimizes a modularity function by initially placing each node in its own community and then iteratively merging communities to maximize the modularity score. Its time complexity is $O(n \log n)$, where n is the number of nodes in the graph. Other than that, popular community detection algorithms include infomap [EBR17], using random walks on the graph to detect communities with complexity $O(m)$, where m is the number of edges; spectral clustering [VL07] computing the eigenvectors of the graph Laplacian matrix and then using these eigenvectors to cluster the nodes. However, due to its $O(n^3)$ time complexity, spectral clustering is only applicable to small graphs. Another algorithm suitable for smaller graphs is the Girvan-Newman [NG04], $O(m^2 n)$, a divisive algorithm working by iteratively removing edges with the highest betweenness centrality.

6.4 Loss Function

As discerned from the analysis in Chapter 5, the datasets used in this study exhibit a significant imbalance, namely an abundance of true edges for the double pion dataset and an excess of negative edges for the multiparticle and PU datasets. Addressing this issue is crucial to avoid models' bias towards the majority class during training. For this reason, the Focal Loss [LGG+18] is adopted, focusing on learning hard miss-classified examples, and thus reducing the impact of the over-represented class.

The Focal Loss builds on top of the binary cross-entropy loss by adding a modulating factor down-weighting the contribution of well-classified examples. Formally, let $y \in \{0, 1\}$ be the binary ground truth label, and $p \in [0, 1]$ be the predicted probability of the positive class. The binary cross-entropy loss is defined as:

$$\mathrm{BCE}(p_t) = -\log(p_t) = -y\log(p) - (1-y)\log(1-p) \quad , \tag{6.12}$$

for

$$p_t = \begin{cases} p & \text{if } y = 1, \\ 1-p & \text{otherwise.} \end{cases} \tag{6.13}$$

To mitigate class imbalance, a common approach is introducing a weighting factor $\alpha_t = \alpha \in [0,1]$ for class $y = 1$, and $\alpha_t = 1 - \alpha$ for $y = 0$. In practical settings, α may be determined using inverse class frequency. The α-balanced binary cross-entropy loss is then defined as follows:

$$\mathrm{BCE}_\alpha(p_t) = -\alpha_t \log(p_t) \quad . \tag{6.14}$$

The BCE_α loss exclusively regulates the weights of positive and negative samples, overlooking the distinction between easy and hard samples. In contrast, the Focal Loss was conceived to address both aspects simultaneously, by further modifying the α-balanced binary cross-entropy with a modulation factor:

$$\mathrm{FL}(p_t) = -\alpha_t (1-p_t)^\gamma y \log(p_t) \quad , \tag{6.15}$$

where γ is a focusing parameter that controls the rate at which easy examples are down-weighted. γ is usually set to a value between 0 and 5, with higher values indicating a stronger focus on hard examples and 0 corresponding to BCE_α.

Apart from dealing with class imbalance, FL allows handling cost-sensitive scenarios by assigning higher penalties to certain types of errors based on their relative cost. This is done by modifying the class weighting parameter, which is particularly important in applications such as our problem, where the cost of a false positive error (incorrectly merging two tracksters) is much higher than that of a false negative error (incorrectly stating that two tracksters are disconnected). The former leads to an immediate loss of information, while the latter can still allow for alternative means of connection through different edges. The choice of α in our experiments is determined by the proportion of positive and negative edges in the dataset, as well as the degree to which penalizing false positives versus false negatives is prioritized. The value of γ is set to 2, which results in focused training on difficult examples. Specifically, an example with $y = 1$ and $p_t = 0.8$ has a 25 times smaller loss than would be obtained with the original BCE loss formulation. However, the focal loss is defined for the binary labels. While binary edge labels are used in most of the experiments, the alternative edge scores defined in Section 5.4.6 are continuous and range from 0 to 1.

Quality Focal Loss (QFL) [LWW+20] softens the one-hot category label and leads to a possible float target $y \in [0,1]$. Specifically, $y = 0$ denotes the negative samples with 0 quality score, and $0 < y \leq 1$ stands for the positive samples with target edge score y.

$$\mathrm{QFL}(p, y) = -|y-p|^\beta \left((1-y)\log(1-p) + y\log(p)\right) \quad . \tag{6.16}$$

Similarly to the Focal Loss, the term $|y-p|^\beta$ with $\beta \geq 0$ acts as a modulating factor. Specifically, the modulating factor is relatively large if an example's quality estimation is inaccurate and deviates from its label y, enabling the model to focus on learning hard examples. Unlike the Focal Loss, the cross-entropy part $-\log(p_t)$ is expressed in its complete form, and the scaling factor $(1-p_t)^\gamma$ is replaced with the absolute difference between the estimated probability p and its true label y, denoted as $|y-p|^\beta$. The absolute value ensures that this term is non-negative. With the QFL loss, the model can be directly trained on the edge scores defined in Section 5.4.6. The choice of the specific loss for different datasets is described further in Chapter 7.

6.5 Performance Evaluation

Given the complexity and diversity of the linking problem, it is not feasible to rely on a single performance metric to assess the effectiveness of a linking algorithm. To thoroughly evaluate linking performance, we advocate for a series of metrics. The overall reconstruction result can be evaluated in terms of clustering performance, regardless of the problem formulation or applied linking method. Because the simulated data for all objects in the calorimeter is available, we can use clustering metrics introduced in Section 6.5.1 with a focus on the reconstruction quality of independent particle showers. Since linking is formulated as edge prediction, and this problem is naturally framed as a two-class classification task, the standard confusion matrix metrics are also adopted: the accuracy and the area under the ROC curve (AUC), as described in Section 6.5.2. Apart from that, it is necessary to look at the physics performance with the energy-aware purity and efficiency (introduced in Section 6.5.3) metrics aligned with the convention adopted by the CMS experiment. Time constraints are another critical consideration. A highly accurate model may take too long to process events, rendering it impractical for real-world scenarios. Thus, a balance must be struck between the model's complexity and its performance. The following sections go into more detail regarding the aforementioned evaluation procedures.

6.5.1 Standard Clustering Metrics

Although access to the ground truth cluster labels in the task is available, evaluating the performance of clustering algorithms is not as straightforward as with supervised classification methods, which can directly use metrics like precision and recall given the predicted labels. Instead, the clustering evaluation metric should assess whether the predicted clustering *partitions* the data in a way similar to the ground truth set rather than focusing on the numerical values of the cluster labels.

Some of the commonly used clustering metrics are based on set matching (Homogeneity, Completeness, and V-measure), information theory (normalized mutual information (NMI) and adjusted mutual information (AMI)), and pair object counting (random (RI) and adjusted random indices (ARI)) as described in [dSCF+12]. Additionally, the B-Cubed (B3) measure is often used, considering recall and precision in terms of pair counting. Nonetheless, none of these methods account for the non-uniformity of individual data point contributions, which is crucial for evaluating the trackster linking problem. Incorrect merging of two high-energy tracksters should be penalized more than merging low-energy, noise-like tracksters. Therefore, to account for this non-uniformity, I propose using energy-weighted variants of some standard clustering metrics, which are discussed in more detail below.

In addition, a variety of alternative intrinsic measures, such as the Silhouette score, Dunn index, or the Calinski-Harabasz index [AAAB15], are used to evaluate clustering effectiveness in the absence of ground truth labels. These metrics quantify the anticipated clusters' compactness and separation. However, because they do not rely on ground truth labelings, they are unable to give efficient metrics for the linking problem and are thus excluded from this study.

Nomenclature. Most standard metrics, including homogeneity, completeness, AMI, and ARI, rely on a common basis – they can all be calculated using a contingency table. To construct

such a table, we start with a set of n objects denoted as $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, representing individual recotracksters in the event. Then, we consider a partition $\mathcal{U} = \{U_1, \ldots, U_R\}$ of $\mathbf{X}$ generated by a clustering algorithm, and a ground truth partition $\mathcal{V} = \{V_1, \ldots, V_K\}$ based on a priori information independent of $\mathcal{U}$. The set $\mathcal{U}$ are the final tracksters generated by the linking algorithm, referred to as *supertracksters* in what follows. $\mathcal{V}$ is a set of ground truth simtracksters. A contingency matrix describes the relationship between the true and predicted clustering labels. In its standard form, it is a matrix $\mathbf{C}$ such that its element $C_{v,u}$ gives the number of samples from the true class v observed in the predicted class u. In case when the weights of the individual points are not uniform, we assume that every sample has an assigned energy $\mathbf{E} = \{\mathbf{e}_1, \ldots, \mathbf{e}_n\}$, scaled to the range from zero to one. This study proposes an energy-weighted version of the contingency matrix, denoted as $\mathbf{C}_{en}$, whose elements $E_{v,u}$, instead of the number of samples, contain accumulated energies of samples from the true class v that are observed in the predicted class u. It should be noted that when the energy of each point is uniform, we get the standard definitions of the contingency matrix, as the accumulated unity energies technically turn into the number of samples.

Energy-Aware Homogeneity, Completeness, and V-measure. In 2007, Rosenberg and Hirschberg [RH07] introduced two desirable objectives for cluster assignments, namely *homogeneity* h and *completeness* c. The former requires that each cluster should solely contain members belonging to the same ground truth cluster, while the latter necessitates that all members of a particular ground truth cluster should be assigned to the same predicted cluster. Both objectives range from 0 to 1, with a higher value indicating better performance. The harmonic mean of these two measures, known as the V-measure, is calculated as follows:

$$v = 2 \cdot \frac{h \cdot c}{h + c} \quad , \tag{6.17}$$

where the formal expressions for homogeneity and completeness scores are given below:

$$h = 1 - \frac{H(\mathcal{V} \,|\, \mathcal{U})}{H(\mathcal{V})} \quad , \quad c = 1 - \frac{H(\mathcal{U} \,|\, \mathcal{V})}{H(\mathcal{U})} \quad . \tag{6.18}$$

H in the above denotes the entropy function. Unlike in the original implementation, instead of the standard entropy, which yet again works with the numbers of samples, I exploit its energy-aware analog. Let E be the total energy of samples for clustering, E_v and E_u the accumulated energies of the samples belonging to ground truth cluster v and predicted cluster u, respectively; $E_{v,u}$ is the accumulated energy of samples from ground truth cluster v assigned to cluster u obtained from the energy-aware contingency matrix. The conditional energy-aware entropy of the ground truth partition, given the predicted cluster assignments, is then given by:

$$H(\mathcal{V} \,|\, \mathcal{U}) = -\sum_{v=1}^{|\mathcal{V}|} \sum_{u=1}^{|\mathcal{U}|} \frac{E_{v,u}}{E} \cdot \log\left(\frac{E_{v,u}}{E_u}\right) \quad . \tag{6.19}$$

And the energy-aware entropy of the ground truth partition is:

$$H(\mathcal{V}) = -\sum_{v=1}^{|\mathcal{V}|} \frac{E_v}{E} \cdot \log\left(\frac{E_v}{E}\right) \quad . \tag{6.20}$$

The conditional entropy $H(\mathcal{U}|\mathcal{V})$ and the entropy of the predicted clusters $H(\mathcal{U})$ are defined symmetrically. The energy-aware analogs of homogeneity and completeness have the same

properties as the standard definitions, and when uniform energies are assigned to all points, their standard definitions are obtained. The utilization of energy-aware variants yields higher scores than standard metrics in cases of incorrectly clustering lower-energy tracksters. Conversely, it demonstrates a substantial drop when a high-energy trackster is clustered inaccurately.

Energy-Aware Adjusted Random Index. The adjusted random index is a measure of the similarity between two cluster assignments, independent of label permutations, with scores ranging from -0.5 to 1 and low core implying a poor agreement between the cluster labels. The ARI calculation involves constructing a pair-wise confusion matrix capturing the true positive (TP), false negative (FN), false positive (FP), and true negative (TN) quantities of trackster pair assignments in both ground truth and predicted clusters.

To compute the pair-wise confusion matrix, we first generate sets of trackster pairs, P_V in ground truth assignment and P_U in predicted partition, belonging to the same cluster. We also define P_X as all the pairs realizable from $\mathbf{X}$. Using P_V, P_U, and P_X, the values of the pair-wise confusion matrix are expressed as:

- $\mathrm{TP} = |P_U \cap P_V|$ represents the number of times a pair of elements are clustered together in both the ground truth and predicted partitions.
- $\mathrm{FN} = |P_V \setminus P_U|$ is the number of times a pair of elements is in the ground truth partition but not in the predicted one.
- $\mathrm{FP} = |P_U \setminus P_V|$ is the number of times a pair of elements is in the predicted partition but not in the ground truth one.
- $\mathrm{TN} = |P_X \setminus (P_U \cup P_V)|$ is the number of pairs not members of any clusters in either partition.

We define $N = |P_X| = \mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN} = \binom{n}{2}$, as the total number of all possible trackster pairs in the event. The unadjusted random index is proportional to the number of sample pairs clustered together in predicted and true assignments or different in both, and is given by:

$$\mathrm{RI} = \frac{\mathrm{TP} + \mathrm{TN}}{N} \quad . \tag{6.21}$$

However, the unadjusted random index does not guarantee that random label assignments will get a value close to zero, particularly if the number of clusters is in the same order of magnitude as the number of samples. To counter this effect, the expected $\mathbb{E}[\mathrm{RI}]$ random index is discounted of random labeling, and the adjusted random index is then defined the follows:

$$\mathrm{ARI} = \frac{2 \cdot (\mathrm{TP} \cdot \mathrm{TN} - \mathrm{FN} \cdot \mathrm{FP})}{(\mathrm{TP} + \mathrm{FN}) \cdot (\mathrm{FN} + \mathrm{TN}) + (\mathrm{TP} + \mathrm{FP}) \cdot (\mathrm{FP} + \mathrm{TN})} \quad . \tag{6.22}$$

In the event of non-uniform data point weights, we construct the sets of pairs that produce TP, TN, FP, and FN values. However, instead of extracting their counts, we compute their pairwise energies, which are subsequently aggregated to yield the weighted TP_e, TN_e, FP_e and FN_e. The energy-aware ARI is then calculated using the original formulation with these values, and it retains the same range and properties as the unweighted ARI.

Pair-Wise Precision and Recall. At this stage, given the above definitions, pair-wise energy-weighted precision P_p and recall R_p can be computed:

$$P_p = \frac{\mathrm{TP}_e}{\mathrm{TP}_e + \mathrm{FP}_e} \quad , \qquad R_p = \frac{\mathrm{TP}_e}{\mathrm{TP}_e + \mathrm{FN}_e} \quad . \tag{6.23}$$

Those, however, are not used in this work, since a more comprehensive recall and precision defined through the B-Cubed metric are utilized.

Energy-Aware B-Cubed Precision and Recall. The B-Cubed metric is based on the computation of the fraction of trackster pairs $(\mathbf{x}_i, \mathbf{x}_j)$ assigned to the same supertrackster U in the predicted clustering (i.e., $U(i) = U(j)$), which were also classified into the same simtrackster V in the ground truth clustering (i.e., $V(i) = V(j)$), and vice versa. The precision measure P_b assesses the extent to which the pairs of reconstructed tracksters originate from the same simulated trackster. On the other hand, the recall R_b evaluates the degree to which pairs of reconstructed tracksters originating from the same particle are classified into the same supertrackster. Assuming that recotrackster i is associated with a supertrackster $U(i)$ in the predicted clustering and a ground truth simtrackster $V(i)$, and n as the total number of recotracksters in the event, the B-Cubed metrics are expressed as follows:

$$P_b = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{|U(i)|} \sum_{j \in U(i)} \delta(i,j) \quad , \qquad R_b = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{|V(i)|} \sum_{j \in V(i)} \delta(i,j) \quad . \tag{6.24}$$

Indicator function $\delta(i,j)$ in the above is defined as:

$$\delta(i,j) = \begin{cases} 1 & \text{if } U(i) = U(j), \text{ i.e., } i \text{ and } j \text{ belong to the same simtrackster,} \\ 0 & \text{otherwise.} \end{cases} \tag{6.25}$$

Technically, Equations 6.24 compute the precision and recall for each recotrackster individually, subsequently averaging the values to obtain the final metric scores. The main difference between the pair-wise and B-cubed precision is that the former measures the agreement between the predicted and ground truth clusters in a pair-wise fashion, whereas B-Cubed precision assesses the accuracy of the predicted clustering for each data point with respect to its own ground truth cluster. B-Cubed precision is, therefore, more focused on evaluating how well the clustering algorithm is able to correctly assign each data point to its own cluster.

To accommodate non-uniform tracksters' energies, instead of working with their counts, we again exploit their energies:

$$P_{be} = \frac{1}{E}\sum_{i=1}^{n} \frac{1}{E_{U(i)}} \sum_{j \in U(i)} \delta(i,j) \cdot \mathbf{e}_j \quad , \qquad R_{be} = \frac{1}{E}\sum_{i=1}^{n} \frac{1}{E_{V(i)}} \sum_{j \in V(i)} \delta(i,j) \cdot \mathbf{e}_j \quad , \tag{6.26}$$

where E is the full energy in the event, $E_{U(i)}$ and $E_{V(i)}$ are the energies of the simtrackster and supertrackster that i belongs to, respectively; while $\mathbf{e_j}$ is the energy of the reconstructed trackster j.

One limitation of this metric in relation to the physics analysis described in Section 6.5.3 stems from the fact that during the simulation at the hit level, multiple particles can contribute energy to a single recotrackster, resulting in some of the hits forming the recotrackster being

included in multiple simtracksters. This scenario is characterized by the shared trackster energy fraction available during simulation, which should, ideally, be included in this evaluation. Nevertheless, the formation of individual recotracksters is handled by CLUE3D and falls outside the scope of this work; therefore, comparative results are prioritized, and the bias introduced by preceeding reconstruction steps is disregarded. The final evaluation in the CMSSW environment is relied on to address the aforementioned effect.

F-score. After precision and recall are calculated, F-score can be computed. A general F_β score uses a positive real factor β, where β is chosen such that recall is considered β times as important as precision:

$$F_\beta = (1 + \beta^2) \cdot \frac{P \cdot R}{\beta^2 \cdot P + R} \quad . \tag{6.27}$$

Two commonly used values for β are 2, weighing recall higher than precision, and 0.5, weighing recall lower than precision. For $\beta = 1$ it turns into the traditional F-score, which is the harmonic mean of precision and recall:

$$F_1 = 2 \cdot \frac{P \cdot R}{P + R} \quad . \tag{6.28}$$

6.5.2 Edge Prediction Metrics

Since the network is trained to assign probabilities to graph edges, we evaluate the NN's performance per-edge during training. While per-edge evaluation does not directly estimate the network's clustering effectiveness, they are closely linked.

It is important to consider the imbalance present in the dataset during this evaluation. In the context of ML, an imbalanced dataset occurs when the distribution of classes is heavily skewed towards one class compared to the others. In such scenarios, the number of samples in the minority class is significantly smaller than in the majority class. In many cases, the classifier may perform well in terms of overall accuracy by simply predicting the majority class most of the time, while its performance on the minority class remains poor. Consequently, out-of-the-box metrics such as precision, recall, and F1-score can become biased toward the majority class, leading to misleading evaluation results that may not fully capture the classifier's performance on the minority class. One approach is to use metrics unaffected by imbalance, such as the Receiver Operating Characteristic (ROC). Alternatively, the contributions of individual edges can be balanced by assigning weights inversely proportional to their frequency, as in the case of balanced accuracy (BA) [BOSB10]. A multitude of metrics to evaluate the performance of the NN models on a per-edge basis is employed, all of which are pre-implemented in the `scikit-learn` Python library:

- Weighted macro-averaged precision, recall, and F1-score capture the average scores across all classes, while giving more weight to the performance of the minority class.
- Balanced accuracy computes each class's arithmetic mean of sensitivity (recall) and specificity with class-balanced sample weights $w_j = \frac{n_e}{2n_{e,j}}$, where n_e is the total number of edges in the event, $n_{e,j}$ is the number of all edges in the respective class j (i.e., either true or false edge).

- The confusion matrix provides a visual representation of the number of correctly classified and misclassified predictions. True Positive Rate (TPR) and False Positive Rate (FPR) are analyzed in particular.
- ROC curve offers a graphical depiction of the binary classifier system's performance by plotting the TPR versus FPR for distinct classification thresholds. The Area Under the Curve (AUC) serves as a metric for evaluating the classifier's performance. One of the advantages of ROC curves is that they are agnostic to class skew, as it concentrates on the TPR and FPR, irrespective of the dataset's balance.

As previously mentioned, the relative impact of false positives (type I error) and false negatives (type II error) on the performance of the network is another factor to consider. This problem, where misclassification errors have varying penalties for different classes, is known as the "cost-sensitive" or an "imbalanced cost" problem. In this work, false positives present a bigger issue than false negative predictions. Therefore, a higher classification threshold can be set at the network's output, treating edges as positive only for the higher predicted probabilities.

6.5.3 Physics Performance Evaluation

The edge labeling during dataset preparation involved the utilization of associator scores (Section 5.3), providing a mapping score between the reco- and simtracksters. To remind the reader, the associator score is a numerical score in the interval $[0, 1]$ reflecting the degree of energy containment of the reconstructed objects in the simulated objects and the other way around. In an ideal reconstruction scenario, a single reconstructed trackster corresponds to a single simulated object and vice versa. In practice, however, the simulated trackster often gets fragmented into multiple reconstructed tracksters, or some recotracksters comprise energy from multiple simtracksters. The evaluation of these scenarios can be conducted through associator scores, where a higher number of tracksters with a score closer to zero indicates an improved reconstruction. In the context of calorimetric reconstruction, the TICL framework employs four associator-based measures and one shared-energy-based measure to assess physics reconstruction performance across the events:

- The **efficiency** is a shared-energy-based measure, defined as the number of reconstructed objects associated with simulated objects that share at least 40% of their energy, scaled by the total number of simulated objects. It should be noted that this is a many-to-one mapping, implying that multiple simtracksters may be associated with a single recotrackster. For instance, if two simulated objects are represented with a single reconstructed object that shares over 40% of its energy with both, the resulting efficiency is 1, indicating optimal performance. Therefore, efficiency is a rather weak criterion and serves the only purpose of being sure that "something has been reconstructed", with some level of goodness imposed by the requirement on the shared energy between the two objects.
- More stringent requirements are put in place by the **purity**, defined as the number of simtracksters that have been associated with at least one reconstructed object (i.e., `Sim-To-Reco` score < 0.2), scaled by the total number of simulated objects. A high purity indicates a high level of confidence in the reconstruction algorithm. While efficiency quantifies the number of reconstructed simulated objects, purity measures the proportion

of correctly associated simulated objects. Occasionally, an algorithm may exhibit high efficiency but low purity, indicating that numerous reconstructed objects are incorrectly associated with the simulated objects.

- The **merge rate** is defined as the number of tracksters associated with more than one simulated object divided by the total number of reconstructed tracksters. A high merge rate indicates ambiguity in the reconstruction, potentially resulting in information loss.
- **Fake rate** is one minus the ratio between the reconstructed objects associated with a simulated one, divided by all the reconstructed ones.
- The **duplicate rate** is calculated by dividing the number of simulated objects associated with more than one reconstructed object by the total number of simulated objects.

Purity and efficiency are the ultimate performance indicators we aim to enhance in this study. Nevertheless, these metrics rely on lower-level data (i.e., hits) that are not included in the dataset since a decision to abstract from lower-level information has been made to reduce the algorithm's computational demands. Consequently, the performance of the models cannot be directly optimized with respect to these metrics, and their evaluation necessitates direct inference of the linking algorithms within the TICL reconstruction pipeline.

Ideally, the goal of the reconstruction algorithm is to attain a purity and efficiency of 1. However, it is not currently achievable with the existing metric definitions, as discussed in Chapter 7. Furthermore, the merge, fake, and duplicate rates would all be zero in the ultimate scenario.

Chapter 7

Experiments and Discussions

This chapter provides a comprehensive review of the conducted experiments and their outcomes, discusses the identified limitations of the experimental approach, and highlights potential avenues for future research. I report on the performance of the explored approaches to trackster linking and discuss the implementation of a plug-in system in CMSSW that exploits the GNN-based algorithm.

7.1 Approach Summary

Datasets of varying overlap levels are generated for model training, ranging from well-separated two-particle scenarios to partially overlapping multi-particle, and high-overlap PU datasets as discussed in Chapter 5. In general, the experimentation process encompasses several phases starting with the simplest dataset consisting of only two closeby pions, and simple pair-wise binary classification with the MLP linking model. Subsequently, based on the accumulated knowledge, the experimentation proceeds toward more complex scenarios with multiple particles and PU. The pair-wise approach is also expanded to consider trackster neighborhoods using message-passing mechanisms outlined in Chapter 6. To identify the best-performing models for each dataset, AUC and BA are employed, along with a set of adapted energy-aware clustering metrics. Linking baselines are established with the standard clustering methods and geometric linking, the default linking algorithm as of `TICL_v4`.

7.2 Experimental Setup

The particle shower reconstruction processing pipeline for the entire CMS detector is implemented in the CMSSW Framework[1], written mostly in C++. The system supports the deployment of deep learning models through the Open Neural Network Exchange format. However, to integrate NN models into the pipeline, it is also necessary to implement feature extraction, input pre-processing, output model result interpretation, and post-processing. To do so, I introduce an NN-based linking plug-in for the TICL pipeline, incorporating the above steps and the network inference in C++. The integration of the linking plug-in into CMSSW is the main objective of this work, and it is essential for the model performance evaluation. However, the complexity of the environment makes it unsuitable for exploratory analysis

[1]CMSSW url: https://github.com/cms-sw/cmssw

and network training. For this reason, to streamline the workflow, all the experiments are conducted in Python, and only the final trained models are exported to be integrated into the CMSSW framework.

For performing interactive data analysis and network training, as presented in this thesis, I employ the SWAN[2] (Service for Web-based ANalysis) CERN cloud service. The used SWAN container is equipped with an Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz with 16 GB of RAM, and an NVIDIA Tesla T4 GPU with 16 GB of memory. The experiments are conducted using a set of Jupyter Notebooks in SWAN and an accompanying Python code-base. The datasets used in the experiments are stored in CERNBox[3], directly accessible from the SWAN environment.

This work employs multiple Python libraries to manipulate physics data, develop models, and implement evaluation metrics. Collision event data is serialized using the CMSSW `NTuplizer` module to an array-based ROOT format, which is read in Python with the help of the `Uproot` library [PDB+20]. To handle variable-sized nested arrays in `NumPy` [HMvdW+20] fashion, the `Awkward Array` library [PES+20] is utilized. In addition, I utilize a suite of tools for graph analysis, visualization, and evaluation metrics. These include the `networkx` [HSSC08] library for graph analysis, `matplotlib` [Hun07] for visualization, and `scikit-learn` [PVG+11] for out-of-the-box evaluation metrics. On the machine learning side, I implement the MLP and GNN-Linking models using `PyTorch 1.11` [PG+19] and export the trained networks to the `ONNX` format for inference. Notably, this work does not leverage the `PyTorch Geometric` [FL19] library, which includes pre-implemented geometric architectures, since their export to ONNX is not currently supported. Thus, the models used in this study are implemented solely with the `PyTorch` library.

7.3 Baseline: Standard Clustering Methods

Prior to exploring ML techniques for reconnecting trackster fragments, I perform an evaluation of classical clustering methods applied to the trackster linking problem. This evaluation aims to establish a baseline allowing for meaningful comparison and facilitating analysis of the strengths and limitations of ML approaches. It also serves as a benchmark for comparison with the current default linking algorithm in TICL. The standard clustering techniques are applied exclusively to the datasets without PU. This is because of the anticipated poor performance due to the undesirable merging of pile-up with the tracksters of interest. The clustering described in this section was performed in Python using the `sklearn.cluster` module.

The idea is to apply standard clustering algorithms to individual tracksters, where an event is considered as a point cloud of tracksters. This contrasts CLUE3D, which performs clustering on LC-level with finer granularity. K-means algorithm is known to suffer from high sensitivity to outliers and does not fit non-spherical data distributions by default; therefore, it is not used as a baseline. Instead, DBSCAN, GMMs, Mean shift (MS), agglomerative clustering (AGG), and affinity propagation (AFF) algorithms are used. The evaluation experiments are run with different normalized sets of features (all features, only spatial features, and spatial features accompanied by energy). Upon evaluation presented in Table 7.1, it was found that the most effective sets of features were those solely comprising the spatial coordinates, such as x, y, z,

[2]SWAN url: https://swan-k8s.cern.ch/
[3]CERNBox url: https://cernbox.cern.ch/

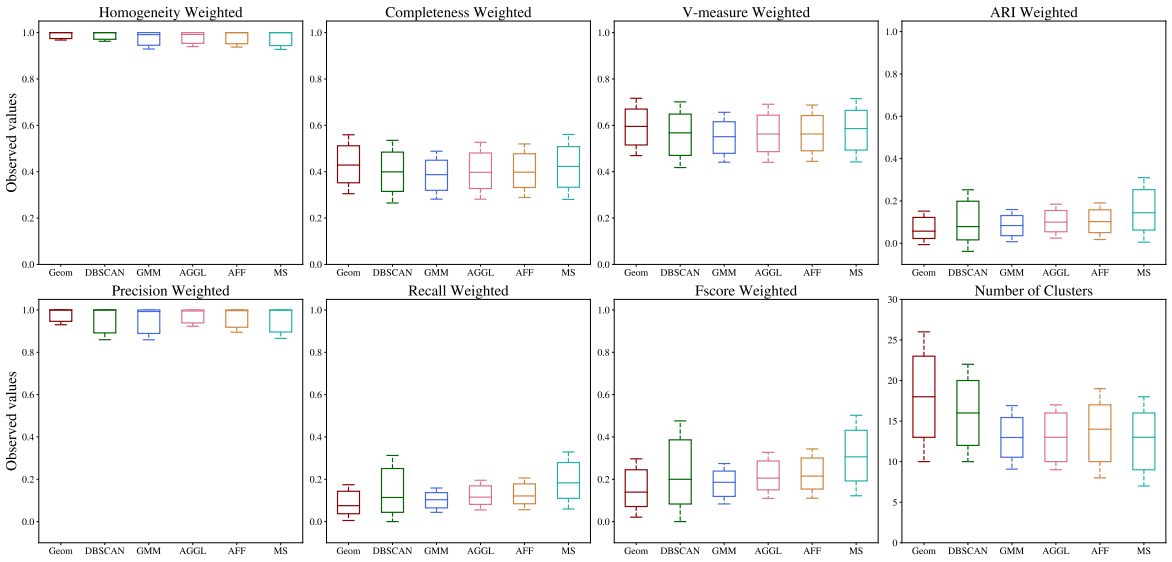


Figure 7.1: Evaluation of standard clustering methods versus the geometric linking for the double pion dataset.

or alternatively, $\eta, \phi, r = \sqrt{x^2 + y^2 + z^2}$, additionally weighted by the energy of individual tracksters in DBSCAN method. However, limiting the feature sets to just spatial coordinates and energy is not expected to yield good performance. This assumption comes from the fact that the preceding highly-tuned CLUE3D algorithm has already effectively leveraged spatial features on the lower granularity. Moreover, CLUE3D is specifically optimized for high pile-up rejection. Thus, relying solely on spatial features for determining whether to merge tracksters would be redundant and has a high potential of degrading the homogeneity of the tracksters.

The performance of standard clustering algorithms on the double pion dataset is shown in Table 7.1 and Figure 7.1, where it is also contrasted with the geometric linking results. In a later section, the performance of geometric linking is presented in Table 7.2 for the same scenario. The suitable parameters for the algorithms are determined based on the validation dataset's best B-Cubed F1 score, with comparable homogeneity to the geometric linking ($h = 0.959$ for double pions and $h = 0.955$ for multiple particles). Then, the evaluation is repeated on the test dataset using the chosen parameters and the best set of trackster features.

Among the explored baselines, MS showed the highest improvement in recall compared to geometric linking, increasing the recall by 0.104 while only decreasing precision by 0.005 in the two-particle case. For multiparticles, recall is improved by 0.05, while precision in decreased by 0.03(Figure A.5). Additionally, MS reduced the average number of tracksters in two close-by pion events from 28.2 to 12.7 (an average reduction of 6 tracksters compared to the geometric linking), and from 215.4 to 94.5 in the multiparticle case. Overall, the conventional clustering algorithms demonstrate a tendency toward a similar level of performance as the geometric linking approach, given the imposition of high homogeneity. Nonetheless, leveraging the embedding space derived from the trained models for clustering purposes is anticipated to lead to superior results. Further investigation of this hypothesis is presented in Section 7.7.

Table 7.1: Double pion dataset clustering with standard clustering methods. All evaluation metrics consider the energy of individual tracksters as described in Section 6.5. In centroid-initialized algorithms, k stands for the number of clusters; t is the number of iterations, and n is the number of data samples. Best values are highlighted in bold. The initial average number of tracksters is $\bar{N} = 28.2$. DBSCAN is additionally weighting point contributions by their energy.

Algorithm	DBSCAN	GMM	AGG	AFF	MS
Complexity	$\mathcal{O}(n \log n)$	$\mathcal{O}(tn^2c)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(tn^2)$
Initialization	density	centroids	density	distance	centroids
Homogeneity	0.941	0.953	**0.956**	0.954	0.938
Completeness	0.401	0.377	0.401	0.340	**0.423**
V-measure	0.547	0.530	0.551	0.550	**0.568**
ARI	0.136	0.084	0.119	0.120	**0.177**
B-Cubed Precision	0.876	0.887	**0.920**	0.903	0.899
B-Cubed Recall	0.169	0.101	0.138	0.139	**0.214**
B-Cubed Fscore	0.253	0.177	0.231	0.232	**0.322**
Avg. num. trackst.	16.2	13.1	13.2	13.6	**12.7**
Best feature set	x, y, z, (E)	x, y, z	x, y, z	η, ϕ, r	x, y, z

Double Pions Settings. The optimal parameters for the algorithms were determined according to the sufficient level of homogeneity (above 93%) and best B-Cubed Fscore. For DBSCAN, the value of $\epsilon = 0.55$ is identified by a knee in the nearest neighbor distances plot [RS16], and a minimum number of samples is set to $\mu = 1$. For GMMs, the number of clusters k is set per event by evaluating performance for $k \in$ range from 2 to 20. In agglomerative clustering, the optimal distance threshold is found to be $d = 1$. The preference parameter is set per event for affinity propagation according to the minimal Euclidean distance between the trackster. Finally, the optimal bandwidth for MS is set to be $b = 0.7$.

Multiparticle Settings. Parameters for multiparticle dataset are chosen in the same fashion as for the double pions. Specifically, I use $eps = 0.27$ with $\mu = 1$ for DBSCAN, bandwidth $b = 0.34$ for MS, and $d = 0.5$ for AGGL. The results of the evaluation are provided in the appendix (Table A.7 and Figure A.5).

7.4 Machine Learning Techniques

In this section, I discuss considerations for training the proposed trackster linking networks. Details of the parameters used for individual datasets are reported in Table A.8.

7.4.1 Training Setup

To optimize model performance, various loss functions are evaluated for each dataset, including BCE, α-weighted BCE, FL, and QFL. Ultimately, the loss function delivering the best performance for each scenario is adopted. The Adam optimizer [KB17] is then employed to minimize the selected loss function with the initial learning rate found through the range test as described in Section 7.4.2. The FC layer weights are initialized using Xavier initialization [GB10] with zero bias, except for the last layer. LeakyReLUs [XWCL15] are employed as activation functions to avoid the problem of dead units, and a dropout rate of 0.2 is applied to mitigate overfitting.

Pair-Wise MLP. In the case of the double pions dataset, where the positive imbalance is observed, the continuous edge scores are utilized instead of the binary score labels during training, helping to counteract the imbalance problem. For this dataset, the QFL Loss with $\beta = 2$ is employed. Experiments with the focal loss ($\gamma = 2$, $\alpha = 0.25$) using binary labels for the double pion dataset were also conducted, but the approach with continuous edge labels was found to be superior. Conversely, the other two datasets exhibit negative imbalance, rendering the continuous edge scores down-weighting positive contributions impractical. As a result, the α-balanced focal loss with the focusing parameter $\gamma = 2$ is utilized, with α determined based on the label imbalance and subsequent tuning. Specifically, α is set to 0.45 for the multiparticle dataset and 0.40 for the pile-up dataset, as:

$$\alpha \approx \frac{neg}{pos + neg} \quad , \tag{7.1}$$

where *neg* refers to the number of negative examples in the dataset, whereas *pos* is the number of positive examples, with specific values for each dataset provided in Table 5.1.

The MLP models are trained for 25 epochs, commencing with an initial learning rate of 10^{-3}, lowered during the learning process. A reduced learning rate on plateau [AKBD22] (for double pions and multiparticle datasets) and cosine annealing [LH17] (for pile-up dataset) LR schedulers are utilized for this purpose, with a lower bound of 10^{-6} on the learning rate. The reduced learning rate on plateau scheduler is designed to monitor validation loss, and if there is no discernible improvement of more than 10^{-3} in it for three consecutive epochs, the LR is automatically reduced by a factor of 10.

GNN. Similar considerations were taken into account for the GNN training. Interestingly, suitable training parameters for each scenario were found to be very similar as in the case of the MLP model and are listed in Table A.8. However, achieving training convergence with GNNs typically requires a greater number of epochs compared to MLP models. As such, GNN models were typically trained for 50 epochs, taking up to 7 hours depending on the particular dataset size.

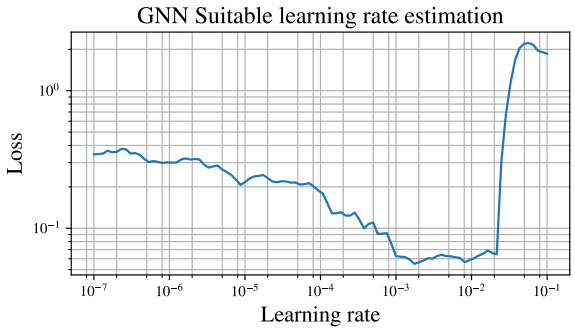


Figure 7.2: Example of the learning rate range test for the GNN network with Adam optimizer and the QFL loss for the double pion dataset. The initial learning rate for this case is chosen to be 10^{-3}.

Last Layer Bias Setting. The datasets used for training are imbalanced, and this knowledge can be utilized to initialize the output layer's bias to reflect the imbalance rates. For instance, in the case of the double pion dataset with a 75% positive and 25% negative ratio, the bias on the logits can be set to predict a probability of 0.75 during initialization. This technique accelerates convergence and eliminates the "hockey stick" loss curve, where the network primarily learns the bias in the first few epochs. Since the sigmoid function is used at the network output, the expected output of the model is:

$$p_0 = \frac{pos}{pos + neg} = \frac{1}{1 + e^{-b_0}} \quad , \tag{7.2}$$

from where the desired initial bias of the last layer is found:

$$b_0 = \ln \frac{pos}{neg} \quad . \tag{7.3}$$

7.4.2 Hyperparameter Tuning

In light of the vast search spaces involved, rather than performing an exhaustive fine-tuning of a particular approach, this work focuses on identifying practical techniques and problem formulations for the trackster linking. Therefore, better performance of the individual models is likely to be achieved by means of better hyper-parameter optimization and augmenting dataset sizes, permitting an increased number of learnable parameters.

Learning Rate Finding

A preliminary step is performed to optimize the learning rate before actual training is initiated. This procedure involves scanning the LR from 10^{-7} to 10^{-1} using 300 mini-batches for training and generating a plot displaying the training loss for various learning rates (Figure 7.2). The resulting training loss plot typically manifests as a basin-shaped curve, suggesting that the optimal learning rate value is positioned within a basin range. The basin's extent spans a broad range of values, indicating that the LR finder only supplies an approximate estimation.

The used approach to finding the optimal learning rate is referred to as the LR range test [SL18]. This involves training the model with a range of learning rates and observing the loss function during training. The learning rate range test is often implemented by gradually

increasing the learning rate from the smallest initial value to a large value over a fixed number of iterations while monitoring the loss function. When starting with a small learning rate, the network begins to converge, and as the learning rate increases, it eventually becomes too large and causes the loss to increase. The suitable learning rate is then chosen to be ten times smaller than the one just before the loss increase.

Model Size

In practice, optimizing the model size is also an important task. Typically, smaller models may not perform satisfactorily owing to the limited capability of feature learning. As the model size increases, the performance may increase to some extent; however, it may subsequently decrease due to the problem of network degradation, i.e., deeper models may face difficulty in learning features, especially when not enough training data is provided. Moreover, a larger model may also lead to significantly increased inference time – a big concern for real-world applications.

In the case of the GNN model, experiments to compare the performance of the reduced and larger variations of the model were conducted. Recall that the final model (for double pions and multiparticles) consists of four EdgeConv blocks (with a hidden dimension of 64), a node encoder with two FC layers, an edge and node feature encoders with two FC layers (output dimensionality 32 and 64, respectively), an attention score network of two layers with a hidden dimension of 32, and a link predictor of two layers with a hidden dimension of 64. It is worth noting that compared to the MLP model, the depth of the encoders is reduced to two layers instead of three, and the hidden dimension size is also lowered for each of the layers (i.e., 64 instead of 128 for feature embeddings).

In contrast, the heavier model is based on the MLP architecture (three-layer networks for each of the sub-modules and higher hidden dimension values). During the experimentation with the heavier model, it was observed that the model achieved a lower training loss compared to the reduced one, indicating that the degradation issue was not encountered yet. However, the evaluation loss failed to match the training loss, suggesting the presence of overfitting. The evaluation results did not demonstrate any improvements by increasing the model's size. Consequently, the decision was made to use the reduced model.

It should also be noted that due to the smaller scale of the pile-up dataset, comprising only 4.1 million edges compared to the 10.7 and 41.1 million edges in the double pions and multiparticle datasets, respectively, the range of considered models was restricted in terms of the number of learnable parameters. The graph-based models have shown to be particularly sensitive to the model architecture, and deeper architectures with many training parameters typically failed to converge. Specifically, extending the dimensions of the feature encoder blocks and increasing the number of EdgeConv blocks led to unsuccessful training. Thus, the final GNN model for the pile-up dataset uses only three EdgeConv blocks instead of four in full GNN models for the other two datasets. Similarly, the sizes of the MLP sub-modules are also reduced to just 2 FC layers for the PU dataset.

Prediction Dynamics Evaluation

To gain insight into the training progress, visualizations of model predictions are generated on a fixed test batch during training. Observing the prediction movement's dynamics makes it possible to intuitively understand how the training is progressing and allows us to better

tune the model parameters. The instability of the network during the fitting process can be detected by the excessive wiggling of the model's predictions, often indicating a struggle to fit the data.

7.4.3 Evaluation During Training and Final Network Selection

Datasets. Each dataset is partitioned into three sets: training, validation, and testing. The training set accounts for 80% of the total data, while the validation and testing sets each account for 10% of the data. The validation set is employed not only for validating the model during training but also for determining the confidence threshold of the model.

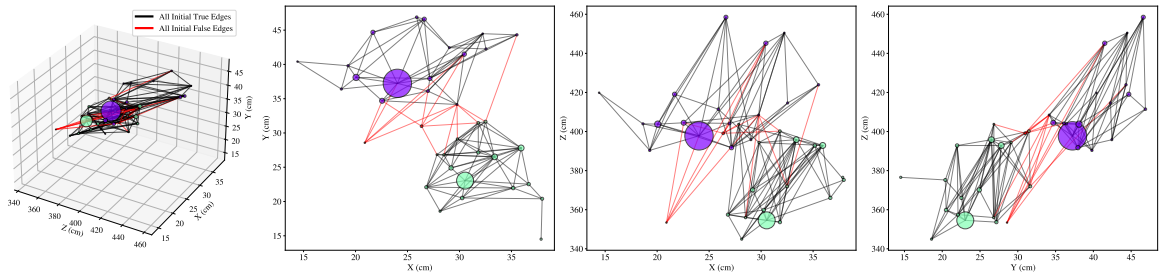
Network Selection. Throughout the training process, the NN's performance is assessed using edge-oriented and clustering metrics. At the end of each epoch, a model snapshot is saved. Once training is finished, the model achieving the best combination of AUC and B-Cubed Recall, along with high homogeneity on the validation dataset, is chosen as the best-performing model. B-Cubed recall, in terms of the physics performance, provides a rough estimate of both the purity and efficiency, while non-compromised homogeneity translates to a low merge rate.

7.5 Performance Evaluation

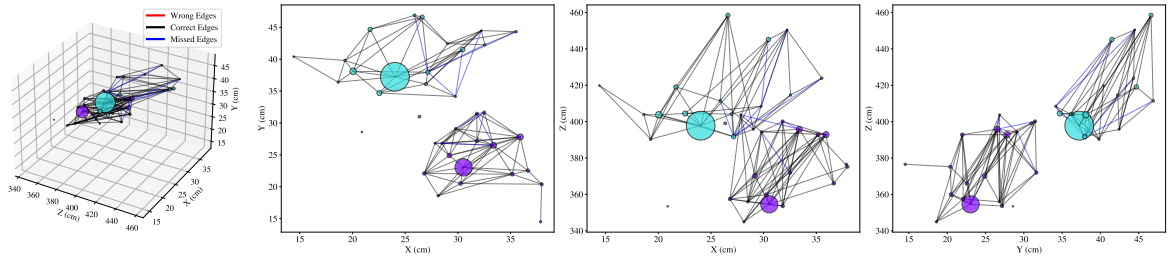
This study employs four distinct evaluation methodologies to quantify the overall performance of the linking algorithms discussed in Section 6.5. Firstly, the generated partitions are evaluated using standard clustering performance metrics, modified to be energy-aware. Secondly, the accuracy of the neural network's per-edge prediction class-balanced performance is evaluated. Yet, since the linking algorithm's primary objective is to improve particle shower reconstruction, it is essential to assess how well the tracksters generated from the same particle are linked together in terms of physics performance. Finally, t-distributed stochastic neighbor embedding (t-SNE) [VdMH08] is employed to visually inspect trackster embeddings produced by NNs. Additionally, the time complexity of algorithms is another factor considered.

7.6 Visual Inspection

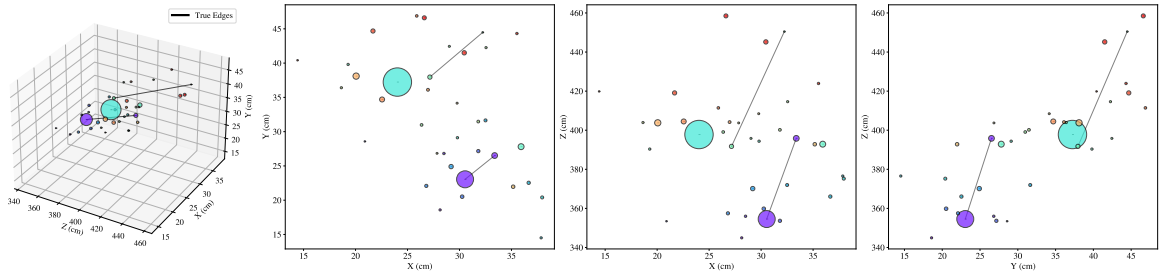
In simpler scenarios, such as the case of double pions, it is visually evident that the GNN-obtained supertracksters are similar to the ground truth, and most of the smaller and distant tracksters are also correctly assigned to the respective supertracksters. However, such visualizations of more complex scenarios can be difficult to interpret due to the crowded nature of the resulting plots. An illustration of the GNN model predictions for a typical double pion event can be observed in Figure 7.3. The figure displays the tracksters' connections without showing individual layer-clusters to maintain visualization readability. Panel (a) presents the original prediction graph input to the network, with the two main simtracksters colored green and violet. The simtracksters are relatively well-separated. In addition, two small unassigned tracksters are located between the simtracksters. The GNN network applied to the event graph (b) successfully separates the simtracksters while connecting most of their internal



(a) : Constructed graph provided as an input to the neural network. The initial number of edges is 128 (shown in black), with a total number of 16 false edges (shown in red).



(b) : Graph of GNN model predictions after the thresholding. The network correctly predicts 97 edges, while missing 15 (shown in blue) due to the high confidence threshold. No false positive edges are predicted.



(c) : Graph of geometric linking predictions. Only two pairs of tracksters are merged.

Figure 7.3: Predicted graph visual inspection for an event with two relatively well-separated closeby pions in 0 PU. Each trackster is visualized as a point at the position of its barycenter with the size proportional to its energy. The figure presents multiple views of the same event (from left to right: 3D, X-Y, X-Z, and Y-Z projections).

components. On the other hand, geometric linking (c) seems to have insufficient merging power.

Numeric Comparison. In contrast to geometric linking, which predicts 34 superclusters for the visualized event by merging only two pairs of trackster fragments, GNN results in five final superstracksters (one more than in the ground truth data). Despite perfect homogeneity for both methods, the completeness of the GNN predicted graph rises to 0.95, while for geometric linking, it makes only 0.375. Additionally, the B-Cubed F-score for geometric linking is very low, just 0.028, while for GNN, it reaches 0.976. Therefore, the metrics comparison supports the visual observations for the event.

7.6.1 Clustering Metrics Evaluation

For each dataset, evaluation of clustering metrics is done separately on respective test sets (the number of events in individual datasets is listed in Table 5.1) after choosing the best-performing models according to the validation data. The results for each dataset are presented in the subsequent subsections. The GNN-based linking approach is found to be highly effective for all three scenarios, surpassing the performance of other explored linking methods, namely pairwise MLP and geometric linking, by a wide margin. As can be seen in what follows, it provides a high level of homogeneity, while keeping a reasonable level of merging even in the most challenging scenario involving multiple particles.

Additionally, the results suggest that the learned approaches significantly outperform the classical clustering methods based on the spatial disposition of the trackster point clouds. The performance of the best clustering results obtained by the MS algorithm, as shown in Table 7.1, was surpassed by the GNN model with a mean F-score performance improvement of 0.567 for the double pions case, and 0.367 in the multi-particles case, as illustrated in Tables 7.2 and 7.3. Therefore, the results strongly suggest that learned approaches are potential options for improving particle shower reconstruction via calorimetric clustering beyond the classical methods.

Double Pion Dataset

The adapted clustering metrics performance for the double pions in 0 PU is listed in Table 7.2 and visually presented in Figure 7.4.

Table 7.2: Double pion dataset clustering performance with geometric linking, MLP and GNN. All evaluation metrics consider the energy of individual tracksters as described in Section 6.5. Best values are highlighted in bold. The initial average number of tracksters per event is $\bar{N} = 28.2$.

Algorithm	Geometric	Pair-wise MLP	GNN model
Homogeneity	**0.959**	0.929	0.949
Completeness	0.427	0.708	**0.854**
V-measure	0.577	0.804	**0.881**
ARI	0.090	0.628	**0.803**
B-Cubed Precision	0.894	0.922	**0.957**
B-Cubed Recall	0.110	0.726	**0.852**
B-Cubed Fscore	0.180	0.812	**0.889**
Avg. num. tracksters	18.75	8.13	**6.10**
Confidence threshold	-	0.90	0.85

The homogeneity metric is essential for ensuring each supercluster comprises only members belonging to the same simtrackster. Notably, lower homogeneity values indicate a loss of information as a result of over-merging. While NN-based methods have been calibrated to

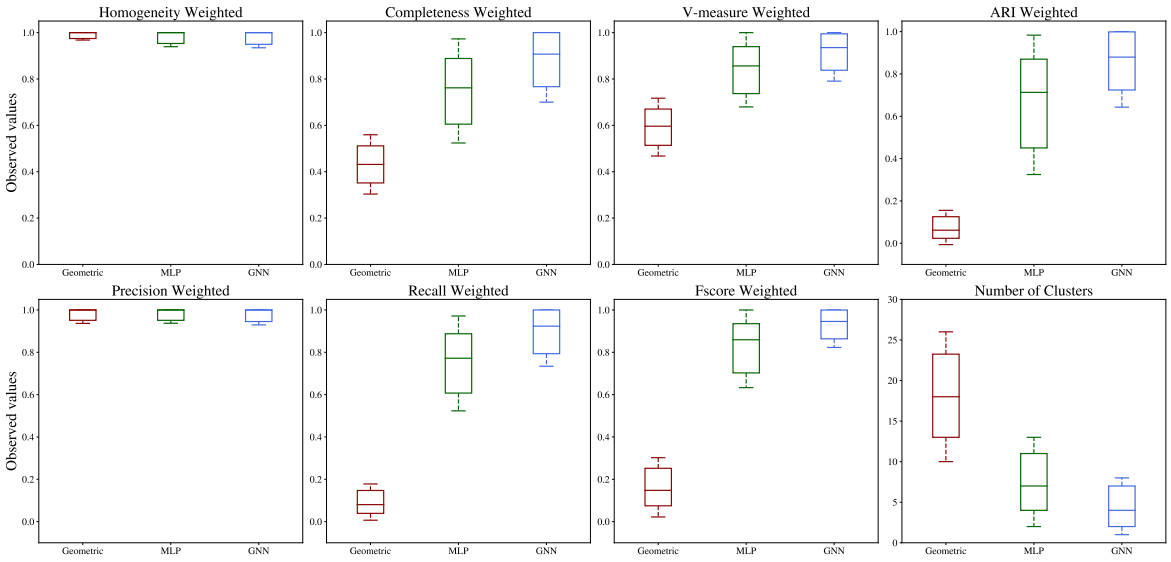


Figure 7.4: Double pion dataset clustering performance of geometric, pair-wise MLP and GNN linking methods. Notably, the GNN method outperforms other linking approaches in all of the metrics, except homogeneity.

achieve comparable levels of average homogeneity as the geometric linking, they exhibit higher standard deviation. However, to enhance the homogeneity of these methods, their confidence threshold can be further increased.

The completeness metric ensures that all elements belonging to a given simtrackster are correctly assigned to the same predicted supercluster. Lower completeness values imply that the algorithm is not capturing macroscopic information of the event and thus fails to merge enough, which correlates to a higher number of final superclusters. Among the three investigated approaches, geometric linking displays the poorest completeness score (42.7%), whereas the MLP reaches a noteworthy increase in completeness of 28% compared to geometric linking. However, this comes at the cost of a higher standard deviation, which is still superior to the average of the geometric linking approach. Finally, GNN linking achieves an impressive 40% improvement in completeness over the geometric linking algorithm.

The V-measure shows a balance of homogeneity and completeness, with equal weight given to both (i.e., $\beta = 1$). In this regard, the GNN approach again exhibits a remarkable performance advantage over the other two methods, with improvements of 30% and 7% compared to the geometric and MLP approaches, respectively.

The utilization of GNN linking results in a notable reduction in the average number of tracksters per event to 6.53, compared to 8.13 for the pair-wise approach and nearly three times as many tracksters for geometric linking. Thus, GNN linking decreases the average number of clusters generated by CLUE3D ($\bar{N} = 28.2$) by over 4.3 times, while maintaining high homogeneity.

In the double-pion dataset, the optimal number of supertracksters, as per the simulation, is typically two. However, there are instances when some CLUE3D tracksters remain unassigned to neither of the simtracksters due to a high Reco-To-Sim score (i.e., higher than 0.2) for all of them. This means that the final ground truth assignment may contain more than just the two final tracksters, leading to a higher number of tracksters predicted by the linking algorithms. Such an event is shown in Figure 7.3, where the total number of the ground

truth tracksters totals four. These unassigned tracksters are typically located at the borders between the simtracksters, resulting in merged hits from both.

Notably, the confidence threshold for the pair-wise approach is set higher than that of the GNN, implying that MLP exhibits a higher merging initiative. This, however, results in a decrease in supertrackster homogeneity. In contrast, geometric linking is optimized for high precision, with insufficient merging capabilities, which aligns with the visual observations.

Multiparticle

Comparable trends to those observed in the double-pion scenario are apparent in the multi-particle case, as evidenced by the clustering results presented in Table 7.3. Additionally, a bar plot illustrating these results is included in the appendix (Figure A.6).

Table 7.3: Ten multiparticles in 0 PU dataset clustering performance with geometric linking, MLP and GNN. All evaluation metrics consider the energy of individual tracksters as described in Section 6.5. Best values are highlighted in bold. The initial average number of tracksters per event is $\bar{N} = 215.4$.

Algorithm	Geometric	Pair-wise MLP	GNN model
Homogeneity	0.960	0.980	**0.984**
Completeness	0.842	0.855	**0.873**
V-measure	0.896	0.912	**0.924**
ARI	0.118	0.338	**0.484**
B-Cubed Precision	0.511	0.792	**0.839**
B-Cubed Recall	0.079	0.245	**0.387**
B-Cubed Fscore	0.131	0.351	**0.498**
Avg. num. tracksters	143.9	159.5	**136.8**
Confidence threshold	-	0.80	0.80

The multiparticle dataset linking presents a more formidable task due to substantial shower overlaps, posing challenges for the network to accurately separate tracksters from different showers and avoid their erroneous merging. Consequently, opting for learned methods with high homogeneity entails a trade-off with recall, which is not as high as that achieved in the double pion dataset (specifically, a decrease of 0.465 in GNN linking recall is observed). Nevertheless, the GNN model continues to outperform both the MLP and geometric linking methods by a wide margin.

Single Particle in 140 PU

The performance on the dataset with pile-up, presumed to be the most challenging, yields somewhat surprising results (Table 7.4 and Figure A.7). All metrics for the NN-based methods demonstrate remarkably high values, with the GNN model consistently outperforming the other methods, except for homogeneity and precision, for which the geometric linking achieves a slightly higher score due to its low merging rate. This unexpected performance can be

attributed to the presence of a single particle in the PU scenario, which eliminates the challenge of predicting incorrect edges between similar-energy particles in close proximity. On the other hand, discrimination between PU tracksters and non-PU tracksters seems to be a much easier task.

Table 7.4: Single particle in 140 PU dataset clustering performance with geometric linking, MLP and GNN. All evaluation metrics consider the energy of individual tracksters as described in Section 6.5. Best values are highlighted in bold. The initial average number of tracksters in the graph per event is $\bar{N} = 28.7$.

Algorithm	Geometric	Pair-wise MLP	GNN model
Homogeneity	**1.000**	0.978	0.991
Completeness	0.577	0.878	**0.900**
V-measure	0.723	0.920	**0.941**
ARI	0.384	0.857	**0.892**
B-Cubed Precision	**1.000**	0.989	0.997
B-Cubed Recall	0.373	0.863	**0.889**
B-Cubed Fscore	0.517	0.916	**0.936**
Avg. num. tracksters	11.54	2.86	**2.74**
Confidence threshold	-	0.70	0.70

Unlike for the other datasets, the term "number of tracksters" in Table 7.4 specifically refers to the tracksters belonging to the training graph, rather than the total number of tracksters in the event. This is because only a subset of event around the primary trackster is utilized in graph creation. On average, the geometric linking tends to merge slightly over half of the tracksters. In contrast, both the MLP and GNN approaches result in a significant reduction of tracksters, roughly ten times fewer. This reduction in tracksters achieved by the learned models indicates a more precise reconstruction of the event. Noticeably, the performance gap between the MLP and GNN models has narrowed, with the GNN model demonstrating only a marginal improvement of a few percentage points on average.

7.6.2 Energy Containment

Additionally, an analysis of energy containment of the supertracksters generated by the linking methods in relation to the ground truth simtracksters is conducted. This assessment was originally introduced in [Nan22] when proposing the current geometric linking approach. The energy intersection over union (EIoU) score is computed between the predicted supertracksters and the simtracksters. The score between a supertrackster t_i and a simtrackster s_j is obtained by dividing the sum of the energy of reconstructed tracksters r, that are common to both, by the sum of energy of tracksters that are contained in their union:

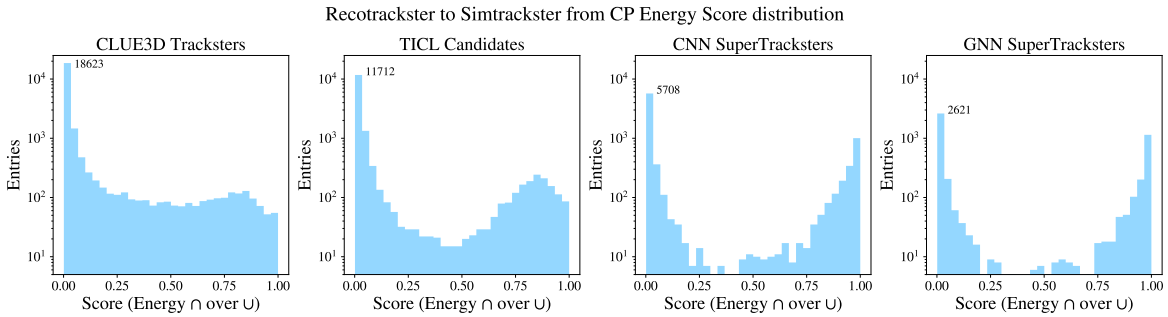


Figure 7.5: Double pion EIoU score distributions for 1000 events for a) initial tracksters produced by CLUE3D, b) candidates created by geometric linking, c) MLP and d) GNN supertracksters (from left to right).

$$\mathrm{EIoU}(\mathrm{t}_i, \mathrm{s}_j) = \frac{\sum_{r_n \in \mathrm{t}_i \cap \mathrm{s}_j} E_{r_n}}{\sum_{t_n \in \mathrm{t}_i \cup \mathrm{s}_j} E_{r_n}} \quad , \tag{7.4}$$

where the association of a CLUE3D trackster to a simtrackster is determined by the associator module. The EIoU score ranges from 0 to 1, with a score of 1 indicating a perfect match between the supertrackster and the simtrackster in terms of their energies, while a score of 0 indicates no energy intersection. EIoU score is initially computed for each supertrackster in the event with respect to all the available simtracksters. The highest EIoU score among the calculated ones is then considered the final score for the supertrackster and is used to guide its final association to a specific simtrackster. The scores for the CLUE3D tracksters, and the candidates from the geometric linking are computed using the same approach.

In Figure 7.5, the EIoU score distribution for the double pion dataset is presented for tracksters reconstructed by CLUE3D and various linking approaches. The original set used to produce the visualization, consisting of 1000 double pion events, features more than 23.3k CLUE3D-produced tracksters, of which 18.6k or 79.8%, have an EIoU score ranging from 0 to 0.01, indicating incomplete clustering. Geometric linking produces 1.5 times fewer clusters than CLUE3D, with 15.5k total trackster candidates, merging 30%, and yielding a slightly better score distribution with fewer low-score tracksters and more scores approaching higher values. On the other hand, MLP and GNN produce almost an order of magnitude fewer supertracksters, 8.3k and 4.6k, respectively, compared to geometric linking. MLP merges over 64.6% of the initial tracksters, while GNN results in an average event size reduction of 80%. A higher fraction of clusters generated using the learned methods has scores closer to perfect matching.

Similar plots for multiparticle and pile-up datasets are presented in Figures A.8 and A.9. Please note that the plots are displayed with a logarithmic y scale, making the absolute difference between the linking methods less obvious.

7.6.3 Per-Edge Evaluation

Performance evaluation of the learned methods also relies on per-edge prediction metrics, facilitating the confidence thresholds determination.

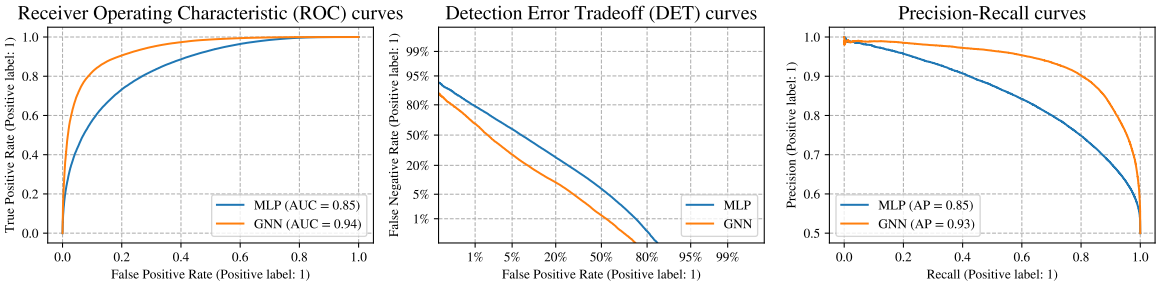


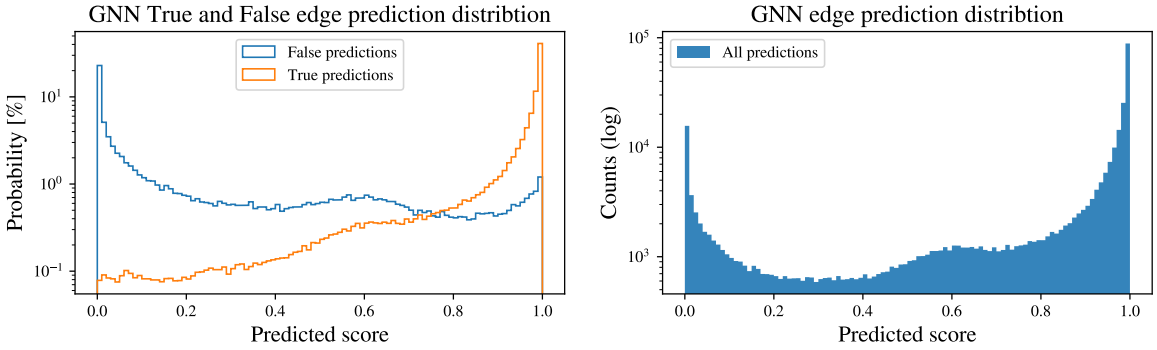
Figure 7.6: ROC, detection error trade-off (DET) and weighted precision-recall curves (from left to right) of the MLP and GNN models trained on double pions.

ROC curves. The ROC and precision-recall curves of the models are analyzed and depicted in Figure 7.6. The MLP model achieves an AUC of 0.85 for the double pion dataset, while the GNN performs even better with an AUC of 0.94. Although the performance is slightly lower for the multiparticle dataset compared to the double pion one (MLP model at 0.85 and the GNN at 0.90), both models still demonstrate the ability to make reliable per-edge predictions in the presence of overlapping showers. Remarkably, both models achieve very high AUC values for the pile-up scenario, with the MLP model at 0.99 and the GNN model at almost 1.0 (Table A.8), indicating very accurate prediction separation.

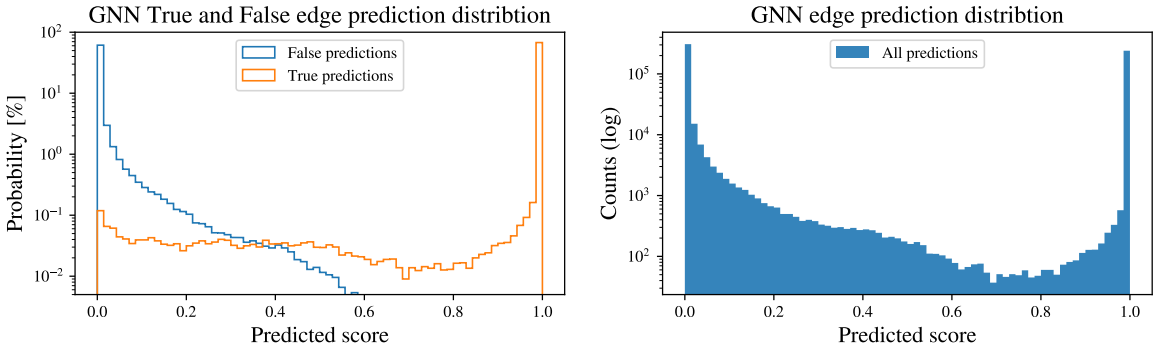
Prediction Distributions. To see if that is the case, the distributions of positive and negative edge predictions by the GNN are examined, as demonstrated in Figure 7.7. By analyzing these distributions, insights into the model's ability to differentiate between positive and negative edges based on their predicted scores are gained. Note that the y scale is logarithmic – despite not being immediately apparent, there is a significant difference between the double pions and PU prediction distributions. In the case of the double pions, there is still a non-negligible presence of false edges with high predicted scores, as depicted in (a) of the corresponding Figure. However, when examining the PU dataset (b), the predictions exhibit a higher level of certainty – over 60% of negative edges are accurately predicted as incorrect with the lowest possible score, and almost 70% of true edges are predicted as correct with the highest achievable score. This contrasts with the double pion scenario, where a much smaller fraction of predictions has such certainties (25% and 40%, respectively). Moreover, none of the negative edges in PU receive a score above 0.6, indicating that setting the confidence threshold to this value would result in almost no false positive predictions.

FP predictions origin. Upon visual inspection, it is evident that the majority of false edges with high scores for the double pions and multiparticle datasets originate from connections between low-energy tracksters located further away from the shower core or at the borders between the showers; while false positive edges in the PU dataset arise mostly from low-energy pile-up tracksters well-aligned with the hard-scattering particle.

Confidence Threshold Setting. According to the precision-recall curves for MLP and GNN models in Figure 7.6, the MLP models tend to require a higher threshold to deliver the same level of precision as the GNN models. In the linking task, the selection of the threshold is guided by the objective of minimizing the probability of FP predictions. While the usual approach is to set the threshold to maximize the F-score, in this task, higher precision over recall is prioritized. To achieve more control over the network performance, a manual threshold



(a) : GNN model predictions for double pion dataset. Over 25% of negative edges are accurately predicted as incorrect with the lowest possible score. Conversely, more than 40% of true edges are predicted as correct with the highest achievable score.



(b) : GNN model predictions for a single particle in PU dataset. Over 60% of negative edges are accurately predicted as incorrect with the lowest possible score. Conversely, almost 70% of true edges are predicted as correct with the highest achievable score.

Figure 7.7: The distribution of the edge scores predicted by the GNN model for true and false edges (on the left), and the full prediction distribution shown in logarithmic scale (on the right).

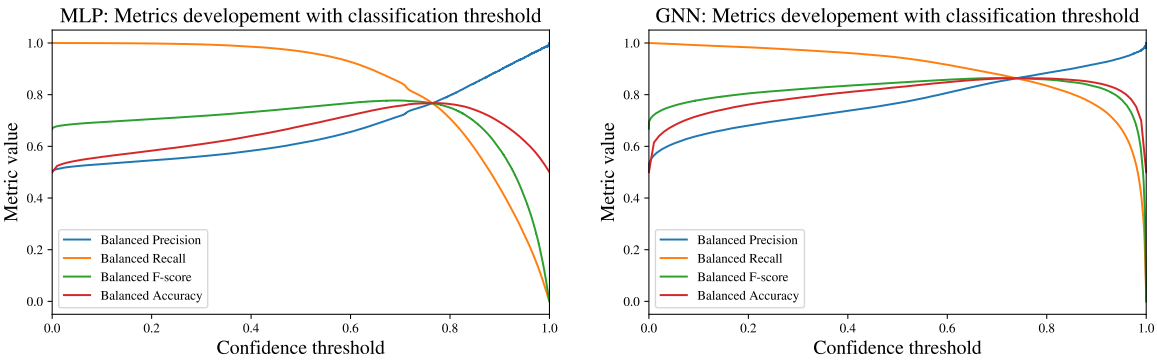


Figure 7.8: Evaluation of per-edge metrics for the models trained on the double pion dataset, illustrating the variation in performance with different confidence thresholds. The left side shows the evaluation results for the MLP model, while the right side displays the results for the GNN model.

selection approach is employed instead of relying on an additional F_β-score with a smaller beta value, giving more weight to precision. By manually setting the threshold, the trade-off between precision and recall is fine-tuned according to the specific requirements of the task. In the case of double pions, a confidence threshold of 0.90 is chosen for the MLP model, while a threshold of 0.85 is selected for the GNN model. The chosen thresholds differ from the F_1-score optimal thresholds, which would have been 0.78 for the MLP and 0.76 for the GNN model. Similar considerations are taken into account for the other datasets and the selected values are presented in Table A.8. After that, to make sure the reasonable threshold is chosen, the sample-weighted F-score and BA development in relation to the threshold value are investigated, as presented in Figure 7.8.

7.6.4 Physics Performance Evaluation

The applicability of the learned models is assessed in a production setting through the evaluation of their physics performance. The models are first exported into the ONNX format and integrated into the CMSSW framework. Guided by the GNN outputs, a DFS algorithm is used to reconstruct candidates as distinct connected components within a graph. This ensures that each trackster is exclusively utilized by a single supertrackster, thereby eliminating the need for ambiguity resolution.

The results in the absence of PU are illustrated in Figure 7.9. By applying a threshold of 0.85 to the GNN predictions, the reconstruction efficiency exceeds 91%, while the purity nearly reaches 50%. This represents an improvement of more than 14% in purity and 22% in efficiency compared to the geometric linking method. The improvement in efficiency and purity is consistent across the varying momentum values, as shown in Figure 7.10. However, this improvement comes at the expense of a slight increase of just over 1% in the merge rate compared to geometric linking. It is worth noting that the merge rate is not solely attributed to the linking itself, as evident from the non-zero CLUE3D merge rate (approximately 1.7%) visualized in Figure 7.9. In contrast, the merge rate for geometric linking tracksters is raised to 2.8%, while for the GNN linking, it increases to 3.9%. Duplicate and fake rates are zero for all methods, revealing a correct one-to-one mapping between simulated and reconstructed objects. Notably, the results observed in the energy-weighted clustering metrics used for the model selection are consistent with the enhancements observed in the production pipeline.

The noticeable decrease in the average number of tracksters, as shown in Figure 7.11, allows for the reconstruction of larger portions of the simtracksters and correlates with improved efficiency. The average number of tracksters per event for the original CLUE3D tracksters is 27.2, with a standard deviation of about 10.5. With the GNN linking approach, the distribution is significantly improved, resulting in an average of only 8.17 tracksters per event and a standard deviation of 7.34. Similarly, the geometric linking approach yields an average of 19.2 tracksters per event with a standard deviation of 7.7. These results are also consistent with the observations made during the previous experiments, implying a correct integration into the production workflow.

Figure 7.12 illustrates the distributions of shared energy and Reco-to-Sim scores for the explored linking algorithms. As expected, the GNN exhibits a shift towards higher energy fractions compared to both geometric linking and the CLUE3D tracksters. This shift is accompanied by a taller peak at a higher shared energy value. Furthermore, the Reco-to-Sim score[4] generally decreases for the GNN, although it follows a similar trend as the geometric linking approach.

[4] Smaller values are superior.

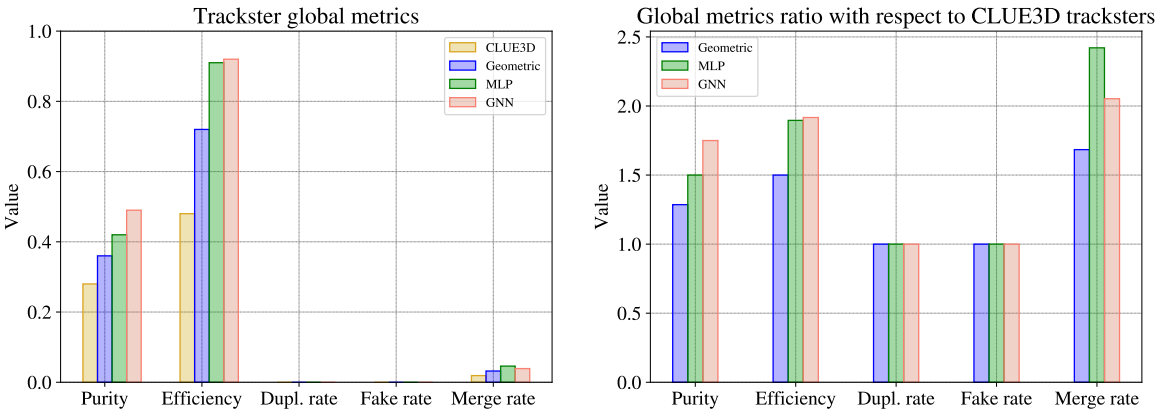


Figure 7.9: Model performance evaluation in the production environment. NN and geometric linking were applied in CMSSW for 1000 double close-by pion events with shower energy ranging from 10 to 600 GeV. The left plot displays the purities, efficiencies, duplicate, fake, and merge rates of the four types of tracksters: CLUE3D tracksters, geometric, MLP, and GNN linking algorithm outputs. The ratio between these trackster metrics with respect to the CLUE3D tracksters is shown on the right.

TTbar. In order to evaluate the generalization capabilities of the network beyond its original training conditions, the application of the GNN network trained on double pions was extended to $t\bar{t}$ events in 0 PU. These events involve the decay of top and antitop quarks, which are interesting due to their large mass, offering insights into indirectly determining the Higgs boson's mass. It can be seen that in a higher-energy/momentum region, GNN reached higher purities and efficiencies than the geometric linking (Figure 7.13). However, most of the tracksters in $t\bar{t}$ events do not exceed energies of 10 GeV (Figure A.10); hence, only a fractional global efficiency improvement of 0.82% is observed for GNN over the geometric linking, while the global purity is decreased by 0.51%. On a positive note, GNN linking reduces the merge rate of these events by 1.5%. Global purities for both methods are very low, reaching just over 5%, while global efficiencies are slightly above 15%. Nevertheless, higher purity values may not be attainable due to substantial information loss during the preceding reconstruction steps. This is exemplified in Figure 7.14, where only a small fraction of tracksters remains available for linking with the CLUE3D trackster count at 45, considerably smaller than the total number of simtracksters at 202.

Despite the lack of improvement in global metrics, these results demonstrate the network's ability to generalize to events not included in the training dataset. Interestingly, the double pion network demonstrates commendable performance even when applied to PU data. Similar experiments were conducted using the GNN trained on multiple particles, applied to double pions. These experiments yielded comparable performance improvements to the network trained exclusively on double pions. However, the reverse process was not successful. When the network trained on double pions was, in its turn, applied to the multiple particle dataset, it frequently led to the erroneous merging of distinct showers, resulting in an increased merge rate.

7.6.5 Previous Reconstruction Steps Bias in Physics Evaluation

The TTbar analysis highlights that relying solely on global metrics is insufficient for evaluating the performance of the methods. Due to the formulation of the linking problem with an

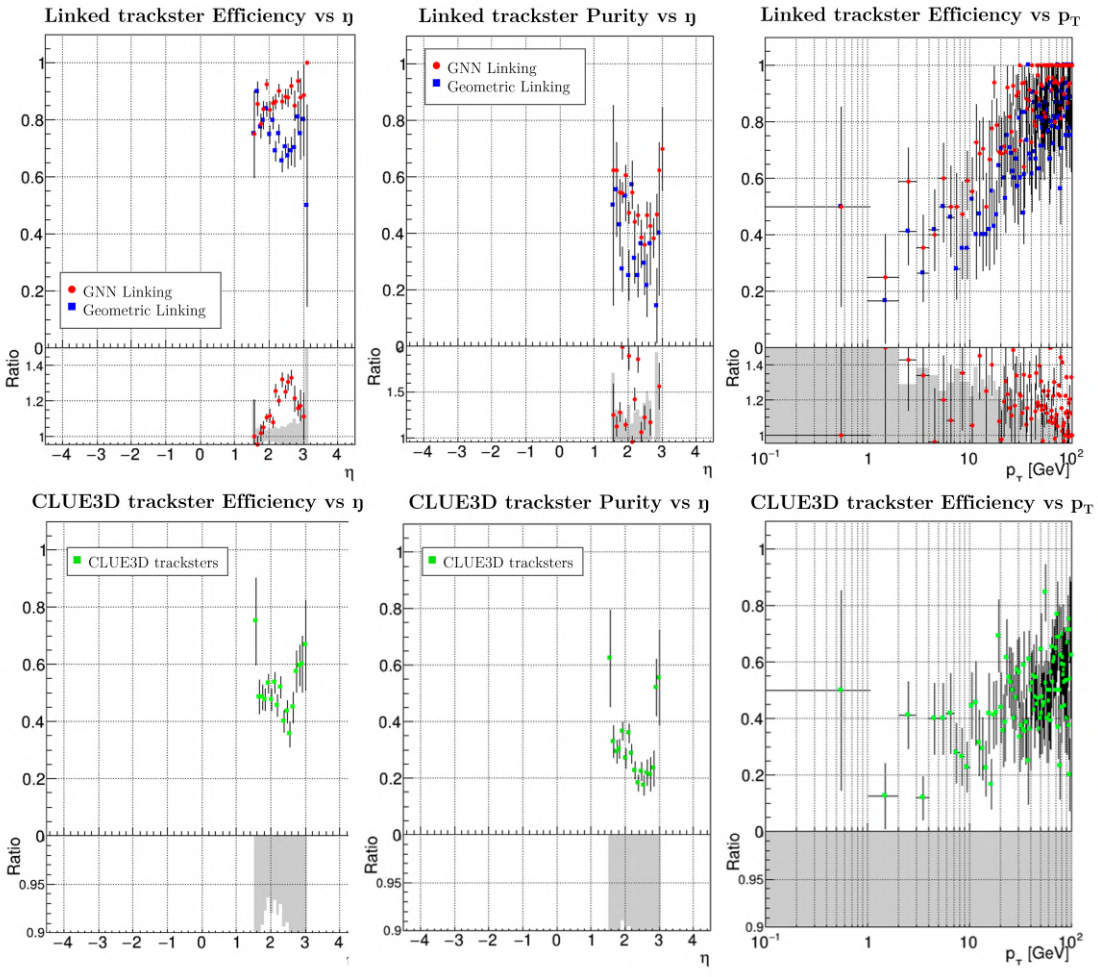


Figure 7.10: The left column shows the efficiency distribution over η for GNN and geometric linking in red and blue, respectively, and CLUE3D tracksters in green. The second column illustrates the purities of the input tracksters and the linking output, categorized by the η of the trackster, using the same color scheme. Finally, the right-most column shows the efficiencies with respect to the value of transversal momentum p_{T}.

objective of combining high-level structures, it is infeasible to re-cluster the previously generated recotracksters, let alone LCs. As a result, potential errors introduced by the previous reconstruction steps must be anticipated (such as a non-zero merge rate prior to linking shown in Figure 7.9). One plausible reason for these errors is when preceding algorithms generate flawed lower-level cluster structures (consisting of hits coming from different simtracksters). In this case, linking cannot eliminate the bias in the performance evaluation.

Figure 7.15 illustrates another bias that cannot be eliminated. It showcases the reconstruction of a double pion event in 0 PU, where the reconstructed data exhibits a significantly lower number of LCs (117) compared to simulation (487). This discrepancy arises because simtracksters encompass all hits related to the simulated particle, while some hits are rejected as noise during the CLUE and CLUE3D reconstruction steps. Due to this noise rejection, LCs are missing in the reconstructed tracksters, making it impossible to attain a perfect performance when comparing against the ground truth data. As per the CLUE definition, LCs containing less than two hits are likely to be rejected as noise. One can notice that

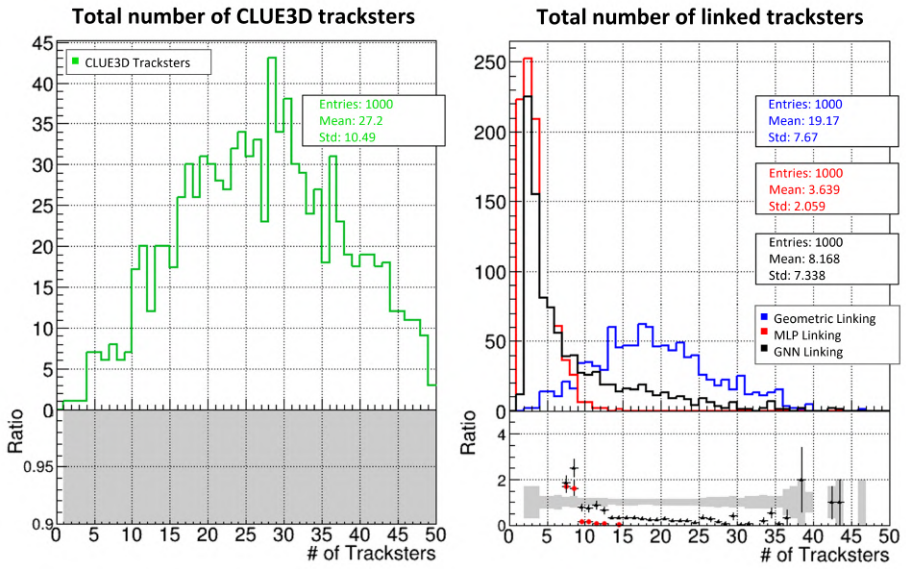


Figure 7.11: Trackster number distributions of the CLUE3D tracksters on the left and GNN (red) and geometric linking (blue) algorithms on the right. The shown data is for 1000 double pions events in 0 PU.

only 31% (153 out of 487) of the simulated LCs in the presented event contain more than one hit. The effect of ignoring single-hit LCs on the reconstructed energy is not negligible. Disregarding all single-hit layer-clusters in simulation in this particular event results in a loss of nearly 20% of the total simtrackster energy. However, a lower energy drop (on average, about 8% for pions) is observed in general.

Therefore, the aforementioned global evaluation suffers from an inherent constraint on the attainable performance ceiling, the upper bound of which remains incalculable. To address this, a plausible remedy involves constraining the research to recotracksters featuring a specific purity threshold or dividing the purity into multiple levels: high, medium, and low, with each category evaluated separately. Notwithstanding, the current study concentrates on the total performance evaluation, and instead of using an absolute frame, it considers the relative improvements with respect to the baselines.

7.6.6 Inspection of PU Merging

In the presence of PU, evaluating the purity and efficiency only informs about the reconstruction of the dominant tracksters in the event, not considering the effect of linking on PU. Hence, an alternative approach is required to examine whether the merging of PU has occurred. To address this, visual techniques are leveraged, both color-labeling of the 3D trackster predictions and T-SNE.

The plot in Figure 7.16 demonstrates the re-connection of the two primary trackster fragments without affecting pile-up tracksters, as indicated by their distinct red color. This is a desirable behavior, however, there is an erroneous merging of one of the well-aligned pile-up tracksters with the green supertrackster, illustrated by the presence of red edges. The merged PU trackster exhibits relatively low energy, and although the merging is undesired, its impact is not considered crucial.

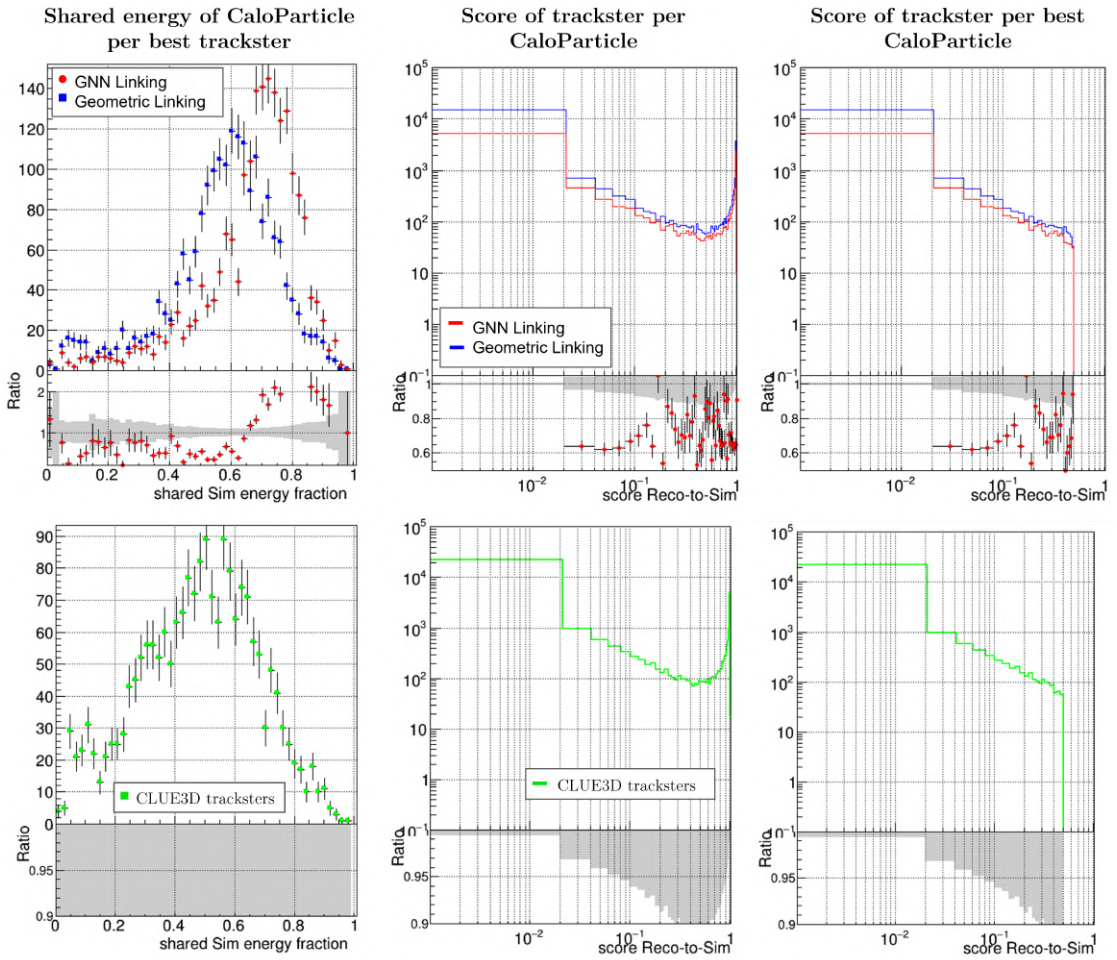


Figure 7.12: CMSSW testing of the shared energy (left) and Reco-to-Sim score after different linking procedures for 1000 double pion events in 0 PU. The GNN linking results are shown in red, while geometric linking is in blue. The middle picture shows the Reco-to-Sim score distribution, while the right-most plot displays this score per the best CaloParticle only. The bottom row shows the same properties for CLUE3D tracksters in green.

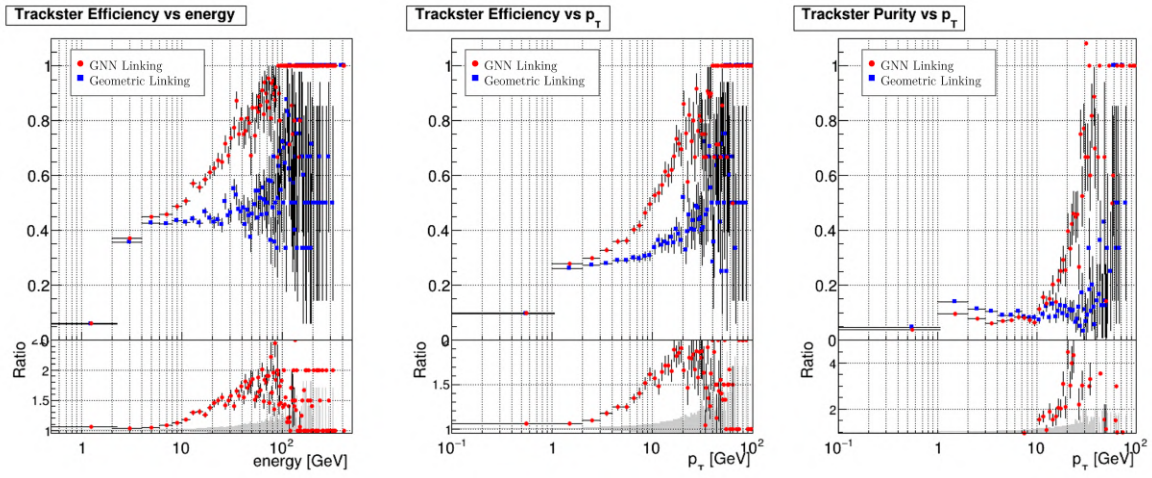


Figure 7.13: $t\bar{t}$ event validation in CMSSW. The first plot on the left illustrates the efficiency distribution for both GNN and geometric linking methods as a function of energy. The subsequent two plots display the distributions of efficiency and purity with respect to transverse momentum.

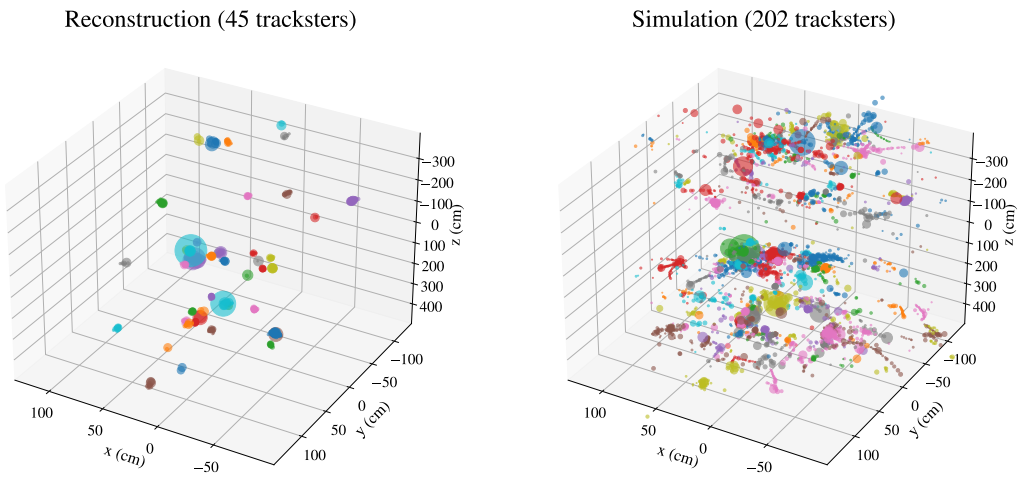


Figure 7.14: Example of a $t\bar{t}$ event. Reconstruction with just 45 tracksters is shown on the left, while the same simulated event with 202 tracksters is on the right. A substantial amount of energy is not captured in the reconstructed event.

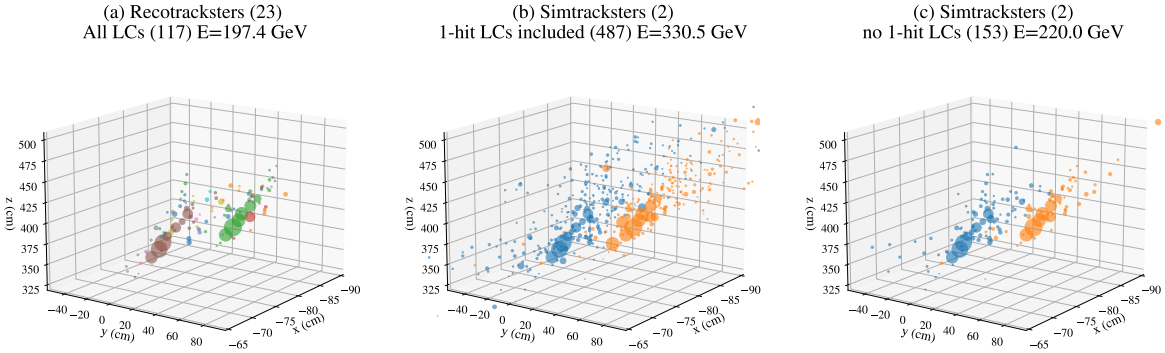


Figure 7.15: Double pion visualization of reconstruction bias. Panel (a) exhibits the reconstructed LCs corresponding to 23 individual recotracksters (before linking). The total number of LCs in the reconstruction amounts to 117, producing an energy of approximately 197 GeV. Panel (b) illustrates the actual simulation data with 487 LCs and a total energy of 330.5 GeV belonging to two simulated tracksters. Panel (c) presents the simtracksters after removing the 1-hit LCs, numbering 334 in total and contributing about 110.5 GeV of energy.

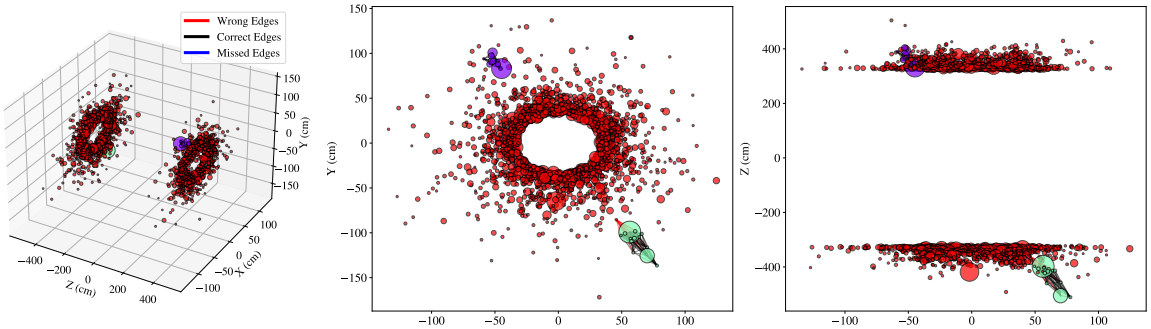


Figure 7.16: Predicted graph visual inspection for an event with pile-up. The image is produced after GNN linking on the whole event graph. Incorrect edges are visualized in red, and correct ones in black. Each trackster is visualized as a point at the position of its barycenter with the size proportional to its energy. Tracksters visualized in red belong to pile-up. The figure presents multiple views of the same event (from left to right: 3D, X-Y, and X-Z projections). If some of the PU tracksters were merged together, it would have resulted in another non-red fragment.

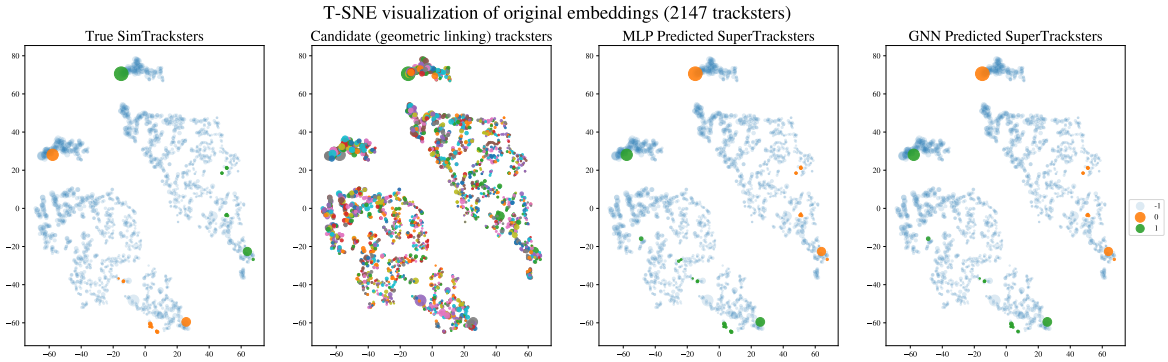
However, if the confidence thresholds on the network outputs are set too low, we observe that all PU tracksters tend to merge together, particularly in the case of the GNN model. Since the network is not trained on pile-up and does not learn to differentiate between individual pile-up tracksters, this behavior is expected but highly undesirable. Interestingly, the MLP model is less prone to pile-up merging, even at lower threshold values. One possible solution to address the issue of pile-up merging is to train the networks using ground truth information for PU, which is currently unavailable but may be incorporated in future developments. Another approach is to use only local graphs in proximity to the main objects of interest, such as Level 1 objects or tracks.

No pile-up trackster merging is observed when a sufficiently high threshold is chosen. This highlights the importance of carefully selecting the threshold value to ensure accurate reconstruction without compromising the separation of PU tracksters.

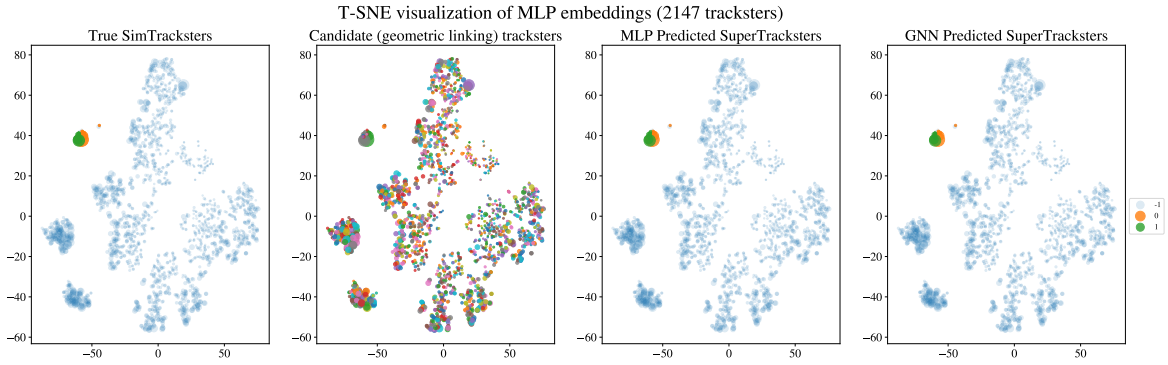
T-SNE Visual Analysis

T-distributed stochastic neighbor embedding is a non-linear dimensionality reduction technique for visualizing high-dimensional data in a low-dimensional space. It constructs a probability distribution over pairs of high-dimensional data points so that similar objects have a high probability of being picked while dissimilar points have a low probability. Similarly, a distribution over the low-dimensional objects matches the high-dimensional distribution as closely as possible. The two distributions are then compared with the Kullback-Leibler divergence [KL51], and the low-dimensional points are iteratively adjusted until the divergence is minimized. The resulting visualization reveals underlying structures in the data that are difficult to discern otherwise in view of high data dimensionalities. T-SNE in this study is used as an out-of-the-box algorithm implemented in `scikit-learn` library with Euclidean distances used as a metric.

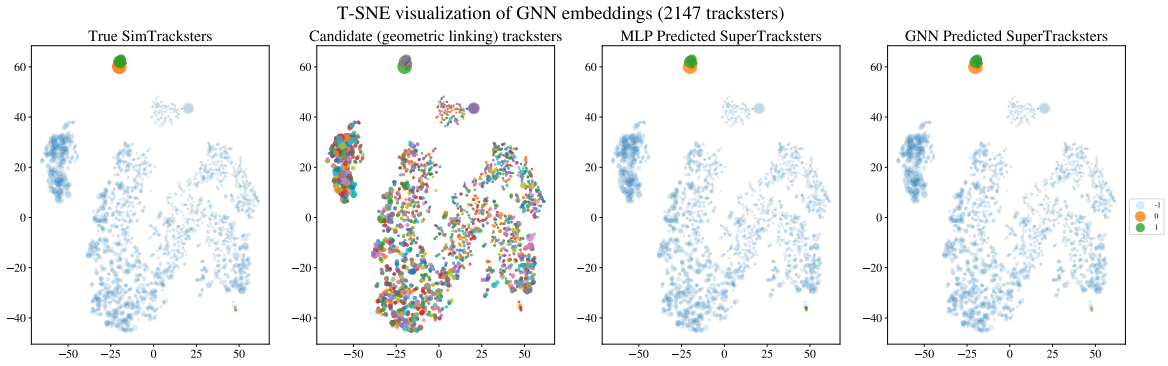
Although the networks in this study are not explicitly trained for metric learning, it is still expected that the learned methods would exhibit close proximity in trackster embeddings compared to the original embeddings. Specifically, the GNN model is anticipated to produce superior trackster embeddings, with related tracksters positioned in close proximity and a clear separation between pile-up tracksters. This expectation arises from the message-passing mechanism employed in GNNs, which facilitates the aggregation of neighborhood information. To validate whether the improved performance of the GNN model can be attributed to an enhanced embedding space, T-SNE projections of the embeddings generated by the MLP and GNN models are examined per event, and shown in Figure 7.17. As a negative control, the T-SNE projection of the raw data space without any embeddings is also included (a). The original feature space lacks any discernible underlying structure (except for clearly separated endcaps). The MLP embeddings in (b) do a better job by moving the corresponding trackster fragments together; although one instance of the simtrackster belonging trackster is positioned further away from the main clusters, and some pile-up tracksters are found in close proximity to the major trackster fragments (as shown in the zoomed figure of the MLP projections Figure 7.18). However, the major trackster fragments are still relatively well-separated from most PU tracksters. The zoomed Figure also indicates that the major tracksters are linearly separable. The MLP model exhibits two minor merging errors where pile-up tracksters are incorrectly assigned to the green supertrackster. In contrast, the GNN model achieves perfect clustering, accurately separating the fragments. The geometric linking yields homogeneous clustering but fails to merge all trackster fragments, resulting in eight fragments instead of the expected two, as seen in Figure 7.18). Another notable observation is that the geometric



(a) : T-SNE visualization of original trackster embeddings.



(b) : T-SNE visualization of MLP trackster embeddings.



(c) : T-SNE visualization of GNN trackster embeddings.

Figure 7.17: The T-SNE projection of the event containing PU showcases the ground truth partitioning, as well as the supertracksters predicted by the geometric linking, MLP and GNN models (from left to right). Each point corresponds to a separate CLUE3D trackster, with its size proportional to the trackster's energy. The blue points indicate PU. For this event, the geometric linking approach demonstrates perfect homogeneity, achieving a completeness score of 0.74 and generating eight tracksters for the two simulated major particles. The MLP model yields a slightly lower homogeneity score, differing by only 0.001, while the GNN model achieves a perfect score. Both the MLP and GNN models exhibit perfect completeness, accurately predicting exactly two supertracksters.

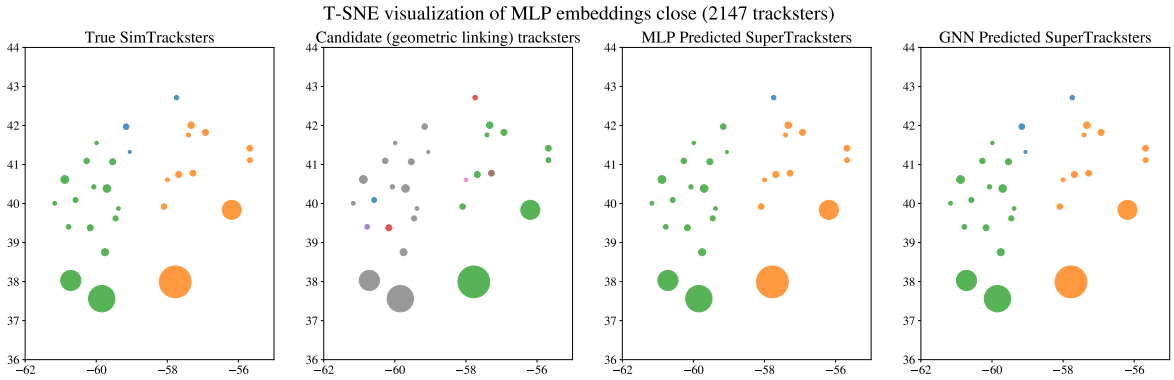


Figure 7.18: Zoomed-in T-SNE projection of MLP embeddings. Geometric linking results in the creation of eight candidates, while MLP and GNN produce exactly two supertracksters, with MLP making two errors and GNN creating a perfect clustering.

linking tends to merge pile-up tracksters. While this behavior may not necessarily be incorrect, it is impossible to validate since no ground truth data is available for the PU tracksters.

Summary. Based on the scatter plots, it is evident that both the MLP and GNN provide superior embeddings compared to the original feature space. However, in contrast to MLP, the GNN embedding demonstrates significantly tighter and more distinct clusters, with all PU tracksters effectively separated. However, since the model is not trained on the PU tracksters, it does not learn to distinguish them from each other. As a result, there is a potential risk of merging PU tracksters together.

7.6.7 Model Complexities

An important quality of deep learning-based models for HEP is the simplicity of the model, the speed of its training and inference on new samples. To quantify these considerations, five metrics for the NN models are measured:

- the number of trainable parameters of the models,
- the training time of each model per data sample,
- average inference times of the models in Python on both CPU and GPU,
- the average inference time per event in CMSSW,
- and, finally, the memory allocation required for the networks.

Although processing multiple batches simultaneously greatly reduces both training and inference times, the data in CMSSW is processed per event, and for simulation purposes, a batch size of one is used during inference to replicate the production environment. Regarding the computational aspects, GNN approaches offer more compact node embeddings with a dimension of 64, compared to 256 for MLP models. However, the edge convolution layers in GNNs are computationally more expensive than the dense layers in feed-forward neural networks. On the other hand, convolutional layers have fewer parameters compared to dense layers, which can be advantageous in terms of memory usage.

Table 7.5: MLP and GNN model parameters. Timing experiments are run on Tesla T4 GPU (16 GB) and Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz. The network is used with a batch size of one in the environment used during model training.

Model	MLP full	MLP red.	GNN full	GNN red.
Trainable parameters	304 641	238 849	119 811	95 043
Params size (MB)	1.22	0.96	0.48	0.38
Trackster-level input in 0 PU: $\mathcal{O}(10^2)$ nodes and $\mathcal{O}(10^3)$ edges				
Input size (MB)	0.06	0.06	0.09	0.09
Forward/backward pass size (MB)	5.54	4.72	16.72	13.03
Avg. CPU inference [s/ev]	0.400	0.332	1.05	0.795
Avg. CPU training [s/ev]	0.424	0.394	1.15	0.965
Avg. GPU inference [s/ev]	0.00189	0.00175	0.00506	0.00424
Avg. GPU training [s/ev]	0.00195	0.00187	0.00540	0.00450
Trackster-level input in 140 PU: $\mathcal{O}(10^3)$ nodes and $\mathcal{O}(10^4)$ edges				
Input size (MB)	0.41	0.41	0.73	0.73
Forward/backward pass size (MB)	46.16	41.04	156.40	121.84
Avg. CPU inference [s/ ev]	0.793	0.680	1.376	1.163
Avg. CPU training [s/ev]	0.840	0.753	1.703	1.412
Avg. GPU inference [s/ev]	0.00222	0.00205	0.00566	0.00480
Avg. GPU training [s/ev]	0.00228	0.00211	0.00602	0.00509
LC-level input in 140 PU: $\mathcal{O}(10^4)$ nodes and $\mathcal{O}(10^5)$ edges				
Input size (MB)	4.10	4.10	7.30	7.30
Forward/backward pass size (MB)	461.60	410.40	1564.00	1218.40
Avg. CPU inference [s/ev]	2.97	3.27	10.3	8.4
Avg. CPU training [s/ev]	3.45	3.31	13.3	11.0
Avg. GPU inference [s/ev]	0.0182	0.0165	0.0477	0.0365
Avg. GPU training [s/ev]	0.0195	0.0175	0.0521	0.0403

Table 7.5 provides the parameter sizes for the MLP and GNN models, including the full and reduced (used for pile-up) versions. The table provides timing information for three different scenarios. The first scenario involves smaller events with approximately 10^2 nodes and 10^3 edges, such as the double pions in the 0 PU dataset. The second scenario deals with events of a larger scale, consisting of $\mathcal{O}(10^3)$ nodes and $\mathcal{O}(10^4)$ edges, corresponding to the full event graph in 140 PU. Lastly, a hypothetical scenario is presented where the network is applied directly to the LCs, ranging one order of magnitude more than the trackster-level, resulting in event sizes of approximately 10^4 nodes and 10^5 edges. Given very slow LC timing results, going down to the level of individual hits (order of 10^5 hits and 10^6 edges) becomes intractable (0.338 s/ev on GPU for a reduced GNN). This experimentally justifies the network application on the trackster-level, since the time complexity is reduced by two magnitudes when compared with the hit-level.

It can be seen that the average inference time of the proposed GNN models scales roughly linearly with the input size, which is necessary for scalable reconstruction in high PU. We also point out that the GNN-based method runs natively on a GPU, with the current runtime for a full 140 PU event being roughly 5 ms/event on a consumer-grade GPU. The algorithm can be easily adapted to computing architectures compatible with popular ML frameworks such as `PyTorch` or `TensorFlow`. This includes GPUs, and there is potential for integration even with the field-programmable gate arrays (FPGAs) using dedicated ML compilers [HRD+20].

It should be noted that Table 7.5 provides timing results under the same conditions as during model training. In this case, the server is shared among multiple users, and several CPU processes could run concurrently. Consequently, these results offer a general overview of the time trends rather than precise timing. For this reason (and for a reason of different hardware), the difference in network timing can be seen when evaluated in the production environment without interference from other processes.

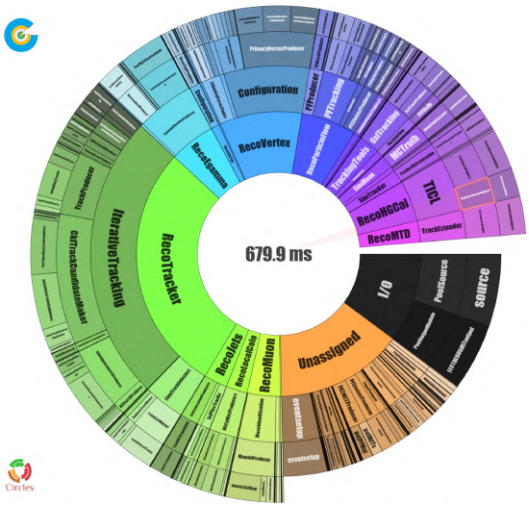
Networks' time and memory allocation, including data pre- and post-processing stages, is directly measured within the CMSSW. In the case of the simple double pions with no pile-up, the GNN model accounts for 5.1 ms of CPU time (0.4% of the entire HGCAL reconstruction pipeline). In this case, the complete CMS reconstruction takes on average 1281 ms. The MLP network exhibits even more efficient timing, requiring only 3.5 ms. In comparison, the geometric linking approach takes 7.2 ms. As a result, for small event sizes, the neural network approaches seem to be faster than the rule-based algorithm.

For the 200 PU events, there are notable changes in the performance of the different approaches, as shown in Figures 7.19 and 7.20. The MLP still exhibits the lowest time requirement, with 551.2 ms, accounting for 1.5% of the total reconstruction time. It also has the lowest allocated memory of 225 MB. However, the GNN is considerably slower in this scenario, taking 1091 ms, which makes up 3.2% of the total reconstruction time. Geometric linking demonstrates reasonable scalability, with a time of 680 ms per event (1.9% of reconstruction time), although it is still around 120 ms slower than MLP-based linking. Regarding allocated memory, geometric linking requires the least amount at 185 MB, equivalent to 1.2% of the total allocated memory. Overall, the allocated memory for the entire reconstruction of the 200 PU events reaches nearly 15.9 GB (TICL reconstruction taking just 2%), with a total time of 36.28 ms per event.

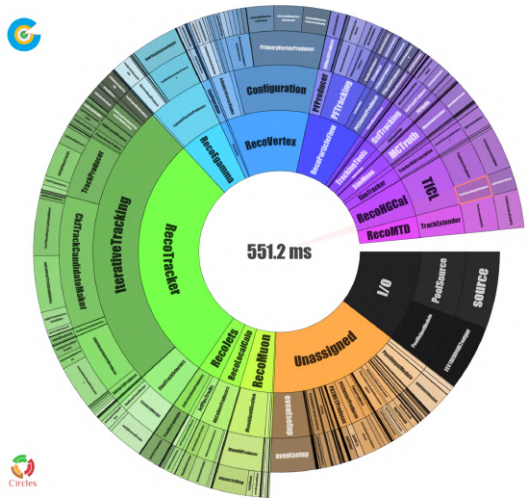
7.6.8 Model Interpretability

In light of the model's growing complexity and diminished interpretability, an investigation was conducted to ascertain the individual feature contributions to the model's output. The Integrated Gradients (IG) algorithm [STY17], which is a widely adopted interpretability technique, was employed for this purpose. IG assigns an importance score to each input feature by approximating the integral of the gradients of the model's output with respect to the inputs.

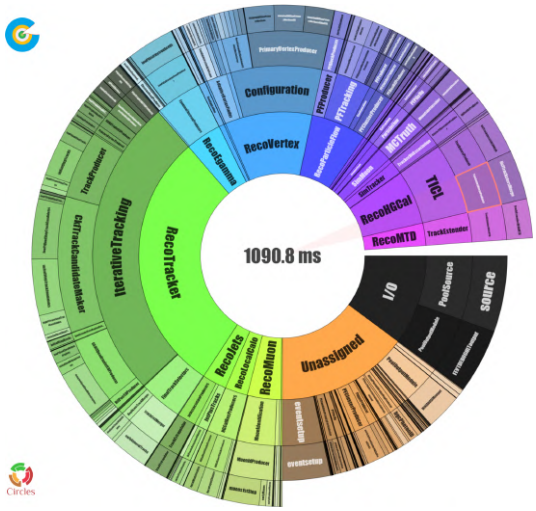
In order to optimize a loss function within a NN, gradients are constructed. To determine the contribution of individual input features to the loss function, the reverse process of integration is employed. The calculation of the integral is inherently challenging and lacks a definitive solution, necessitating the use of a Riemann Sum approximation. This involves dividing the process into a fixed number of steps and establishing a baseline for comparison. The baseline serves as a reference point against which the input vector is evaluated. The



(a) : Geometric linking timing in CMSSW.



(b) : MLP timing in CMSSW.

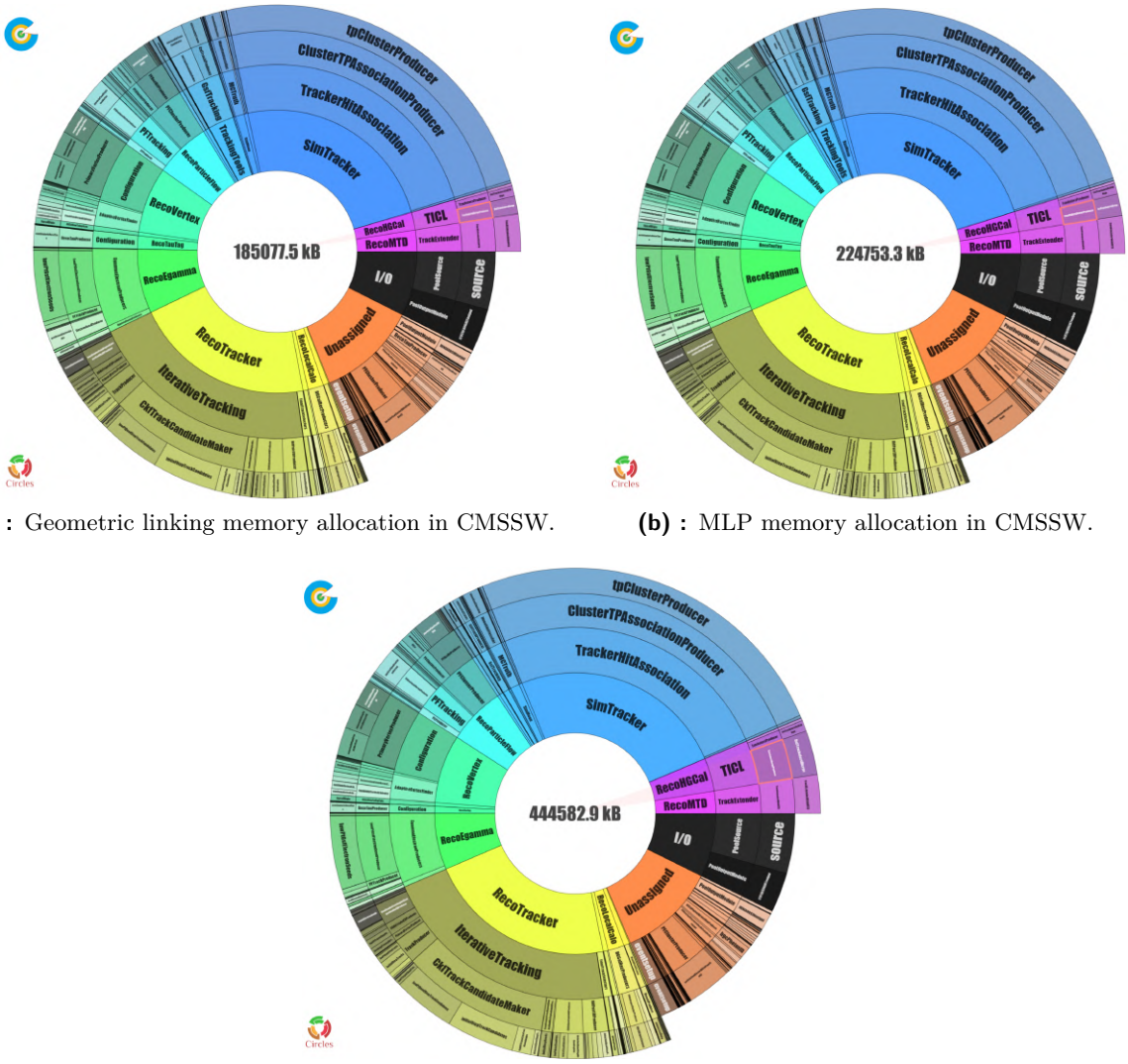


(c) : GNN timing in CMSSW.

Figure 7.19: CMSSW CPU Intel Xeon (Skylake IBRS) timing of the pion events in 200 PU. Time is per one event processing using the corresponding linking method. The timing was performed on an empty server with no additional processes running in parallel. Plots created using [Fwy].

process commences by initializing all node features randomly, with the baseline as the starting point. Subsequently, linear interpolation is performed between the baseline and the original node features. Gradients are then calculated to assess the relationship between feature changes and corresponding variations in the model's predictions. Finally, a numerical approximation of the integral is computed by averaging the gradients, serving as an indicator of the relative importance of each node feature in influencing the model's output.

Figure 7.21 illustrates the feature importance of both the MLP and GNN models for the double pion dataset determined through IG. Both models concur on the significance of spatial features such as z, η, ϕ, and the z component of the principal eigenvector in generating the output. Additionally, the GNN model places considerable emphasis on the z_{min} and z_{max} positions of the trackster skeletons, whereas these feature contributions are less apparent in the MLP model. Certain features, including raw electromagnetic (EM) energy, photon and electron probabilities, as well as LC and trackster density per event, are deemed



(a) : Geometric linking memory allocation in CMSSW.

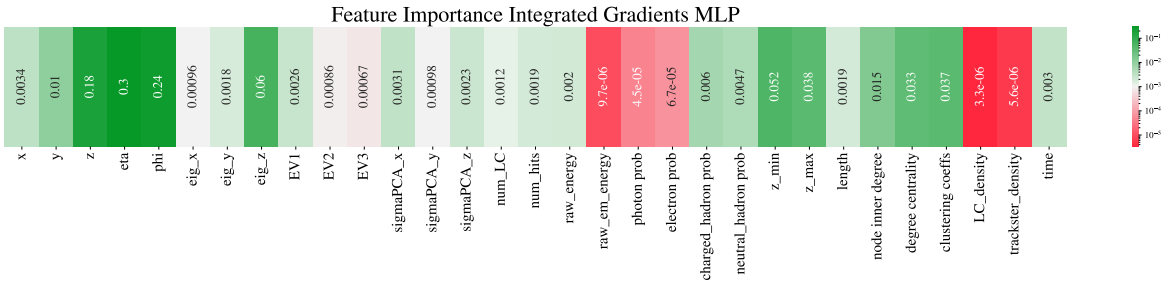
(b) : MLP memory allocation in CMSSW.

(c) : GNN memory allocation in CMSSW.

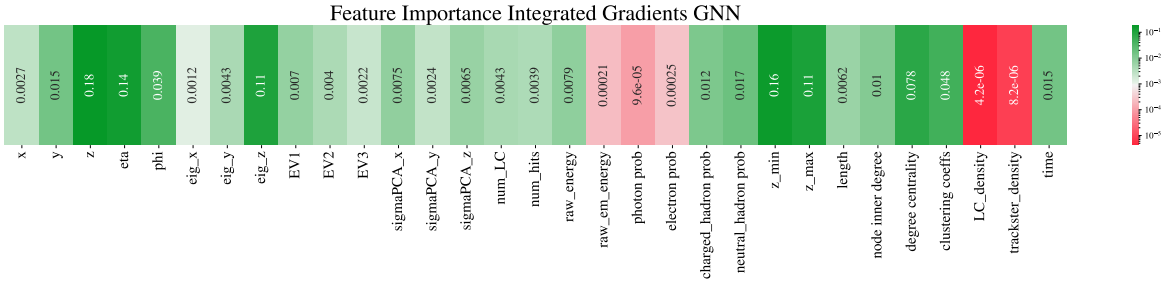
Figure 7.20: CMSSW allocated memory of pions in 200 PU. Memory is per one event processing using the corresponding linking method. Plots created using [Fwy].

inconsequential for both networks' decision-making processes. The disregard for raw EM energy may be attributed to its strong correlation with raw energy. Photon and electron probabilities hold little relevance in this dataset composed solely of pions. Similarly, the densities of LC and tracksters only hold significance when the dataset encompasses varying event sizes (e.g., including both 140 PU and 0 PU), hence their minimal influence in this context is expected. Overall, the GNN model assigns greater importance to features that are relatively less exploited by the MLP model, highlighting the divergent feature preferences between the two architectures.

The feature importance analysis is also done for multiparticle (Figure A.11) and single particle in 140 PU (Figure A.12) datasets. In the case of the multiparticle dataset, a similar overall trend in feature importance is observed as for the double pions, except the GNN assigned higher contributions to the photon and electron PIDs, which is expected since these particle types are now included in the dataset. Additionally, the raw energy feature gained more importance in the GNN model, most likely, due to the presence of multiple particles



(a) : Feature importance of the MLP model trained on double pions in 0 PU.



(b) : Feature importance of the GNN model trained on double pions in 0 PU.

Figure 7.21: Integrated gradients feature importance for the double pion dataset in 0 PU. The values are calculated with respect to the random baselines.

interacting electromagnetically. For the PU dataset, the GNN model showed increased importance for the x barycenter position and the sigma of the x and y PCA components. Furthermore, time-related and graph structural features gained significance in the GNN model.

Summary. Overall, the feature importance analysis revealed consistent patterns across different datasets, with the GNN model often assigning greater importance to features that are not as heavily utilized by the MLP model. A similar analysis can be done for the edge features but is not included in this work.

7.6.9 Model Output Post-Processing

As a part of the experiments aiming to further improve NN predictions, the utilization of the Louvain community detection method as the network post-processing step was explored. Since the output of the network is technically a similarity graph, with each edge having a weight, one can try detecting communities within that graph, whose components would translate to separate supertracksters. Such post-processing was anticipated to be particularly useful for the multiparticle case, which holds more potential for incorrect interconnection between the non-matching showers through a low number of edges, which would likely be removed by community detection. The Louvain method has been chosen for its relatively low complexity $\mathcal{O}(n \log n)$, compared to a more standard spectral clustering with $\mathcal{O}(n^3)$, where n is the number of nodes. It is based on the concept of modularity, measuring the strength of the network division into communities. The Louvain algorithm follows a two-step process: the “greedy” phase and the “aggregation” phase. In the greedy phase, the algorithm assigns each node in the network to its own separate community. Then, it iteratively examines each node and evaluates the potential gain in modularity resulting from moving

the node to a neighboring community. The algorithm evaluates a range of possible moves and selects the one that maximizes the modularity gain. This process is repeated until no further improvement in modularity can be achieved. In the aggregation phase, the network is transformed into a new network where communities detected in the previous phase are represented as "meta-nodes". This step reduces the complexity of the network and allows for the detection of higher-level community structures. The two phases of the Louvain algorithm are iterated multiple times. The edges' strength (i.e., NN predictions) is also considered during the modularity optimization.

Two approaches to Louvain post-processing have been explored. The first one involved applying community detection to the entire graph, where the edge weights were derived from the predictions of the NN. In the second approach, edges below the confidence threshold of the NN were removed before applying community detection to the reduced graph. However, neither of these two approaches demonstrated superior performance compared to the clustering based on original network predictions. When applying the Louvain algorithm to the full graphs, it had a tendency to merge distinct showers together due to the network assigning relatively high scores to even incorrect edges (i.e., the confidence threshold is not accounted for in the used version of the algorithm). Conversely, when applied to the reduced graph, the algorithm tended to excessively separate tracksters, leading to the poorer recall. As a result, such post-processing was deemed inefficient.

7.7 Clustering Model Embeddings

The analysis of trackster embeddings using T-SNE revealed that the network exhibits an ability to group related trackster fragments together in the embedding space, even without explicit training for that purpose. Leveraging this insight, the NN embeddings are utilized as inputs for standard clustering methods. Results of the clustering are compared with what was obtained from spatial clustering described at the beginning of this Chapter. In the case of double pions, we observe improvements across all performance metrics compared to clustering solely based on spatial features (Figure 7.22). The AFF method stood out with the best performance, achieving a homogeneity score of 0.927 and an F-score of 0.415. Moreover, AFF produced an average number of clusters of 6.29, three times less than geometric linking. These results represent a noteworthy improvement of more than 20% in F-score compared to geometric linking while experiencing a decrease of 3% in homogeneity. Furthermore, AFF outperformed its spatial feature-based counterpart by nearly two-fold. However, it is essential to note that AFF still fell significantly short compared to MLP and GNN, lagging behind by more than 20% in terms of F-score. These findings highlight the potential of utilizing trackster embeddings for clustering tasks, since they capture more discriminative information than spatial features alone.

7.8 Contrastive Learning

Clustering in model embedding space has shown potential for learning representative trackster embeddings. This observation has led to the consideration of training the network with the specific metric learning objective. Experiments on contrastive GNN linking have been initiated, although not fully completed as of this moment; therefore, only the outline of the experiments

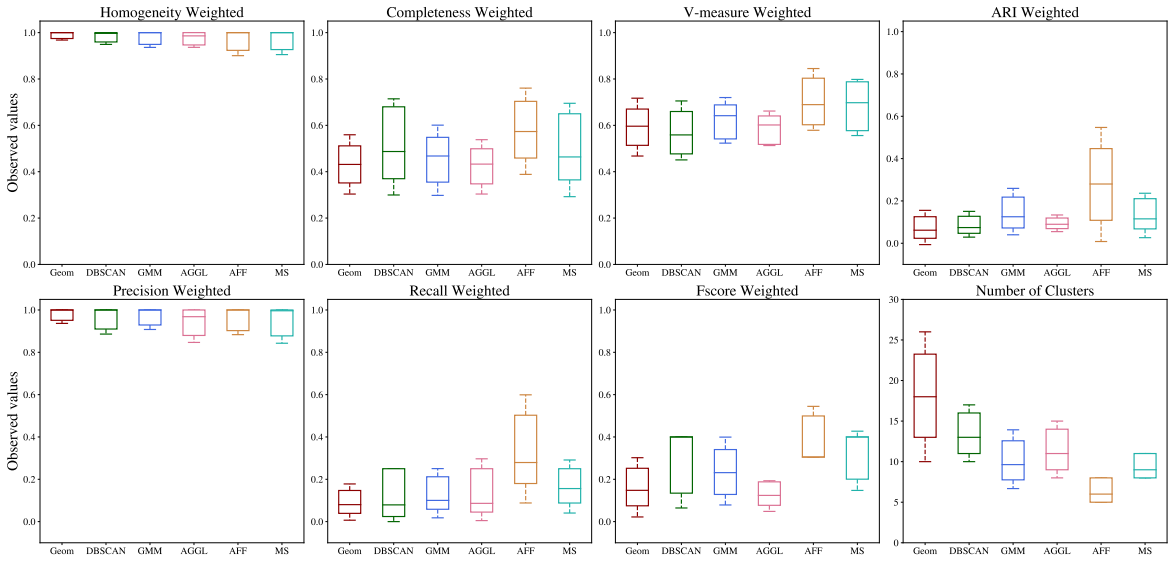


Figure 7.22: Evaluation of the standard clustering methods applied to the GNN trackster embeddings.

is provided without the experimental results. Further exploration and experimentation are needed to fully investigate the potential of contrastive GNN linking. This includes fine-tuning the network architecture, optimizing the contrastive loss function, and conducting evaluations on various datasets.

The idea behind contrastive GNN linking is to train the network to learn a metric maximizing the similarity between related tracksters and minimizing it between unrelated tracksters. The network leverages the same GNN backbone (initial features pre-processing and static-graph EdgeConv layers), stripped of the final edge classification sub-module. The model outputs embeddings for each input trackster in the event. After the network is trained, computing the similarity between two tracksters boils down to computing the dot product between their $\mathcal{L}2$-normalized embeddings (cosine similarity), which is a highly efficient operation. A modern processor can perform millions of such operations per second.

The training process involves training the network in a Siamese fashion – several-stream network sharing its parameters. The network receives a labeled tuple of trackster features as an input (query q, positive p, and negatives n). Positive trackster p comes from the same shower as the query, while the negative sample n comes from a different shower. After the network generates embeddings for the input tracksters, distances between sample representations are computed. The model is then optimized to minimize the distance for similar samples and maximize it for dissimilar samples using contrastive loss [KTW+21]:

$$\mathcal{L} = \begin{cases} \frac{1}{2}\mathrm{d}^2\left(\mathbf{h}_q, \mathbf{h}_j\right) & \text{if } \delta(q,j) = 1, \\ \frac{1}{2}\left(\max(0,\, \alpha - \mathrm{d}(\mathbf{h}_q, \mathbf{h}_j))\right)^2 & \text{if } \delta(q,j) = 0, \end{cases} \tag{7.5}$$

where $\delta(q,j) = 1$ indicates a pair of tracksters coming from the same shower, while $\delta(q,j) = 0$ means they are not. Losses are initially calculated in a pair-wise manner query-positive and query-negative and subsequently aggregated for the whole event. α is a margin hyperparameter that determines when non-matching pairs are separated by a wide enough distance for the loss to disregard them. $d(\mathbf{h}_q, \mathbf{h}_j)$ denotes the Euclidean distance between the $\mathcal{L}2$-normalized trackster embeddings $\mathbf{h}$. The negative vector forces learning in the network, while the positive

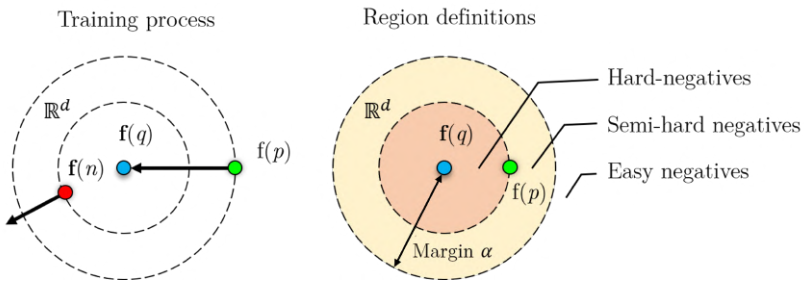


Figure 7.23: The learning process of the metric learning model is depicted on the left of the picture, in which the model learns to minimize distance for similar samples and maximize distance for dissimilar samples. The hard-, semi-, and easy-negative samples are separated on the right side based on their distance from the query feature vector [Jek21].

vector serves as a regularizer. The margin ensures that the model does not linger, enlarging the distinction between a triplet's positive and negative samples when it already does so properly, allowing it to concentrate on more challenging triplets.

During training, for every trackster in the event, positive and negative tracksters are chosen, forming a training tuple. However, randomly sampling training tuples is an inefficient strategy, since many of them may already fulfill the contrastive loss margin criteria. In such a case, the GNN model's weights would remain unchanged, and no learning would occur since an error is not created. A typical technique for choosing the optimal training triplets involves hard negative mining. Adverse pair selection entails iterating over non-matching tracksters that are "hard" negatives (Figure 7.23), meaning they are similar in the descriptor space and incur a high loss. Negative examples are chosen from fragments of the showers other than that the query trackster belongs to. Out of all non-matching tracksters, the k nearest neighbors are selected as negatives. Hard negative mining is repeated for every event at each iteration, hence the hard negatives are chosen based on the current GNN parameters. The positives are selected randomly from the matching shower.

7.9 Summary

The experimental results consistently suggest that incorporating the trackster neighborhood within the point cloud allows the learned models to extract additional information about the reconstructed shower, leading to enhanced reconstruction performance across all datasets. Nevertheless, the timing results obtained in the production environment indicate that the inference time of the GNN network is two times slower compared to the existing linking plug-in. It is worth noting that this situation might change in the future when executed on a GPU within the CMSSW framework. However, at this point, GPU support for network inference in the production environment is currently unavailable.

Chapter 8

Conclusions

This thesis explored a promising approach to trackster linking in the HGCAL reconstruction pipeline using Graph Neural Networks, with the main focus on accumulating separate components of hadronic showers. The task for the network was framed as the between-trackster edge classification in a sparse graph representation of pre-clustered energy deposits. This approach offers significant advantages, including a reduction in problem size by approximately two orders of magnitude compared to starting from lower-level energetic objects such as recHits, and the ability to learn particle shower structures from their graph representation. The approach employed the static collision event graphs with the custom Edge Convolution-based model with attention. The generated weighted graph was then subjected to edge probability thresholding and identification of the connected components, corresponding to individual particle showers. A relatively compact GNN model, comprising approximately 120k parameters, facilitates moderately rapid inference with a full time including pre- and post-processing of approximately 1 s per collision event in 200 PU. Additional optimizations hold the potential of using this network for offline event reconstruction applications.

The model was trained and tested on three separate datasets with varying levels of complexity, generated specifically for this task. Namely, datasets contained events with two close-by pions, multiple randomly chosen particles, and a single particle in pile-up. The proposed algorithm was also tested and deployed in the CMSSW production framework. The results demonstrated that the model performs remarkably well for all explored scenarios. For double pions in 0 PU, it successfully assigned even small distant tracksters to the correct supertrackster, resulting in high-quality and efficient reconstruction. While the performance of the GNN approach on the multiparticle dataset, where avoiding incorrect merging of individual showers together is of difficulty, was not as outstanding, it still generated tracksters with high homogeneity. As for the dataset including pile-up, the network exhibited exceptional performance by effectively distinguishing between PU and non-PU tracksters. As a result, the GNN-based approach showed a significant improvement over the currently used geometric linking by correctly clustering a larger portion of the event's energy across all three datasets. Additionally, it was found that the model trained on double pions also performs well for other particle types, such as $t\bar{t}$ events, and even the pile-up scenario. Likewise, the model trained on the multiparticle samples demonstrated effectiveness for the double pions dataset, indicating its ability to generalize for the cases outside its training domain.

8.1 Future Work

In future work, there are several avenues to optimize the proposed GNN-based network for faster inference. One of them is to explore model compression techniques, such as pruning, aiming to remove unnecessary connections or parameters from the network. Additionally, quantization can be applied to convert the model's parameters from a higher precision format (e.g., float32) to a lower precision format (float8), reducing memory usage and computational requirements. Another direction to explore is the parallel processing of multiple events since the network operates faster in batches rather than on a per-event basis. It could be also beneficial to consider splitting the graph into multiple subgraphs, such as neighborhoods of tracks or L1 objects and using the network on these local graphs, reducing the network input size. Size could also be reduced by applying the GNN as a next step after the geometric linking, with a potential of 40–50% input reduction. Exploring alternative graph-building techniques, such as using a cone instead of a cylindrical window opened at the tracksters' barycenters or skeleton nodes, also holds a promise for reducing the number of links between tracksters. In conclusion, the proposed GNN algorithm offers inherent parallelizability and can leverage hardware acceleration through graphics processing units, field-programmable gate arrays, or specialized ML-specific processors in the production environment.

To enhance the performance and capabilities of the model, extending the model to include track information and internal features learned from the constituent LCs could be an interesting direction. Incorporating track information would potentially allow us to estimate the correct number of charged particles in the event.

While I have shown that a per-edge loss function already converges to an adequate physics performance overall, the model can be further improved with a more physics-motivated optimization criterion. Such a loss function could take into account event-level predictions in addition to or instead of the per-edge ones.

In order to develop an ML-based linking algorithm usable in production, a realistic PU simulated dataset, including detailed ground truth interactions with the detector material, needs to be used for the model training. The optimization and validation process of the network should involve a diverse range of realistic PU events to capture global linking properties, as well as diverse particle gun samples to ensure the model is generalizable. To evaluate the reconstruction performance, a more diverse range of particle types needs to be studied in detail. Additionally, assessing high-level derived quantities such as missing transverse momentum and jet reconstruction should provide a more comprehensive understanding of the reconstruction performance.

When employed in production, the algorithm must be adjusted to frequently changing experimental conditions. This can be addressed by performing periodic retraining with up-to-date running condition data. In realistic GNN training, care must be taken to ensure that the reconstruction quality of uncommon particles and particles in the low-probability tails of distributions are not harmed and that the reconstruction performance remains consistent, which may be addressed with detailed simulations and weighting schemes.

Further investigating clustering techniques in the embedding space produced by NNs also presents an opportunity for exploration. Specifically, training the neural network for metric learning, such as contrastive linking discussed in this thesis, can allow for the efficient separation of individual showers. Although these experiments have been initiated, they require completion to obtain conclusive results.

Appendix A

Additional Figures and Tables

A.1 Raw Dataset Properties

Tables A.1–A.6 give the properties of individual sub-trees in raw ROOT dataset files.

Table A.1: Properties of the `clusters` sub-tree in the raw dataset ROOT files.

Property	Definition
`energy`	Layer-cluster energy, calculated as a sum of the constituent hits' energies.
`position x/y/z/eta/phi`	The LC's barycenter coordinates, computed by energy-weighted average of the constituent hit coordinates, represented in both cartesian (x, y, z) coordinate space and (η, ϕ) space.
`cluster_type`	List of cluster types used to identify successive layers based on their z coordinate: `CE_E_120` = 0, `CE_E_200` = 1, `CE_E_300` = 2, `CE_H_120_F` = 3 (Fine), `CE_H_200_F` = 4 (Fine), `CE_H_300_F` = 5 (Fine), `CE_H_120_C` = 6 (Coarse), `CE_H_200_C` = 7 (Coarse), `CE_H_SCINT_C` = 8 (Scintillators). The first part of the cluster type indicates the detector compartment (CE-E or CE-H), while the second part indicates the sensor sizes (120, 200, or 300 µm) as described in Section 2.6.1.
`cluster_time`	LC's time accumulated from the rechit times. Set to -99 for low-energy layer-clusters.
`cluster_timeErr`	LC's time error.
`cluster_localDensity`	Local energy density of LCs computed by CLUE through the application of a Gaussian convolution kernel.
`cluster_number_of _hits`	Number of hits in LCs.

Table A.2: Properties of the `tracksters`, `simtrackstersSC`, `simtrackstersCP`, and `trackstersMerged` sub-trees in the raw dataset ROOT files.

Property	Definition
`nClusters`	Total number of layer-cluster in the event. It should be noted that the 1-hit LCs (filtered out during reconstruction) are also included in this number.
`nTracksters`	Total number of tracksters in the event.
`time`	Trackster "barycenter" time. Only available for the tracksters reaching a certain energy threshold, otherwise set to -99. It is a distance-weighted average of the LCs times.
`timeError`	Trackster time error.
`raw_energy`	Trackster raw energy accumulated as a sum of the constituent LC energies.
`raw_em_energy`	Trackster electromagnetic raw energy accumulated as a sum of the constituent LC energies.
`trackser_barycenter x/y/z/eta/phi`	Positions of tracksters' barycenters, computed by energy-weighted average of the constituent LC coordinates, represented in both Cartesian (x, y, z) coordinate space and (η, ϕ) space. Unlike layer-cluster barycenters, the z trackster coordinate may not align with any particular detector layer but rather fall between the two adjacent layers.
`EV1/EV2/EV3`	The three eigenvalues of energy-aware PCA applied to the collection of constituent LCs.
`eVector0_x/y/z`	The first principal component coordinates x, y, z of energy-aware PCA.
`sigmaPCA1/2/3`	The three component-wise reconstruction errors from energy-aware PCA.
`id_probabilities`	Vector of 8 probabilities: [photon, electron, muon, neutral pion, charged hadron, neutral hadron, ambiguous, unknown] produced by a PID CNN applied to individual trackster (Section 3.3.3).
`vertices_indices`	Vector of LC indices in the global collection represented as a vector of vectors.
`vertices_x/y/z`	Vector of LC positions represented as a vector of vectors.
`vertices_energy`	Vector of LC energies, represented as a vector of vectors.
`vertices_multiplicity`	Vector of LC multiplicities for each trackster represented as a vector of vectors. It indicates the number of times a particular LC has been used. For reconstructed tracksters, each LC is only assigned to one trackster, so the vector of multiplicities is always a vector of ones.
`layer_cluster_seed`	Index of the CLUE3D LC seed for each trackster.

Table A.3: Table of the `candidates` sub-tree features in the raw ROOT dataset.

Property	Definition
`candidate_charge`	Electric charge of the `TICLCandidates`.
`candiate_pdgID`	PDG ID of the `TICLCandidates`.
`candidate_id_prob`	Vector of 8 probabilities: [photon, electron, muon, neutral pion, charged hadron, neutral hadron, ambiguous, unknown] produced by a PID CNN applied to individual trackster (section 3.3.3).
`candidate_time/ timeErr`	`TICLCandidate` time and time error.
`candidate_px/py/pz`	`TICLCandidate` momentum.
`track_in_candidate`	Tracks assigned to individual `TICLCandidates`.
`trackster_in_ candidate`	Vector of indices of CLUE3D tracksters in the `TICLCandidates`.

Table A.4: Table of the `graph` sub-tree features in the raw ROOT dataset.

Property	Definition
`node_linked_inners`	Indices of the linked *inner* trackster (closer to the CMS vertex) for each trackster.
`node_linked_outers`	Indices of the linked *outer* trackster (further from the CMS vertex) for each trackster.
`isRootTrackster`	Is set to `True` if there are no linked inners, `False` otherwise.

Table A.5: Table of the `tracks` sub-tree features in the raw ROOT dataset.

Property	Definition
`track_hgcal x/y/z/eta/phi`	Tracks propagated to HGCAL front face positions.
`track_hgcal px/py/pz`	Track propagated to HGCAL front face momentum.
`track_pt`	Track transversal momentum p_T.
`track_charge`	Track charge.
`track_time/time_err`	Track time and time error at the HGCAL front face.
`track_time_quality`	Track time quality, > 0.5 is a good quality.
`track_nhits`	Number of hits in the track.

Table A.6: `associations` sub-tree features in the raw ROOT dataset. The same properties are provided for the `trackstersMerged`, but they are not used in this study.

Property	Definition
`recoToSim_SC/` `recoToSim_CP`	For each CLUE3D trackster, a vector of simtracksters SC/CP indices in the Reco-To-Sim map.
`recoToSim_SC_score/` `recoToSim_CP_score`	A list of Reco-to-Sim scores quantifying the degree of similarity between a recotrackster and all simtracksters from SimCluster/CaloParticle in the event. A score of 0 indicating a perfect match and a score of 1 indicating a complete mismatch.
`simToReco_SC/` `simToReco_CP`	For each simtrackster SC/CP, a vector of recotrackster indices in the Sim-To-Reco map.
`simToReco_SC_score/` `simToReco_CP_score`	A list of Sim-to-Reco scores quantifying the degree of similarity between a simtrackster from SimCluster and all tracksters in the event. A score of 0 indicating a perfect match and a score of 1 indicating a complete mismatch.
`recoToSim_SC_sharedE/` `recoToSim_CP_sharedE`	Shared energy between individual recotracksters and all the simtracksters SC or CP.

A.2 Dataset Analysis

Figure A.1 provides an additional energy distribution illustration of the multiparticle dataset, as a part of the dataset exploratory analysis. Figures A.2, A.3, and A.4 give correlation matrices for double pion and multiparticle datasets, and the difference of major and PU tracksters, respectively.

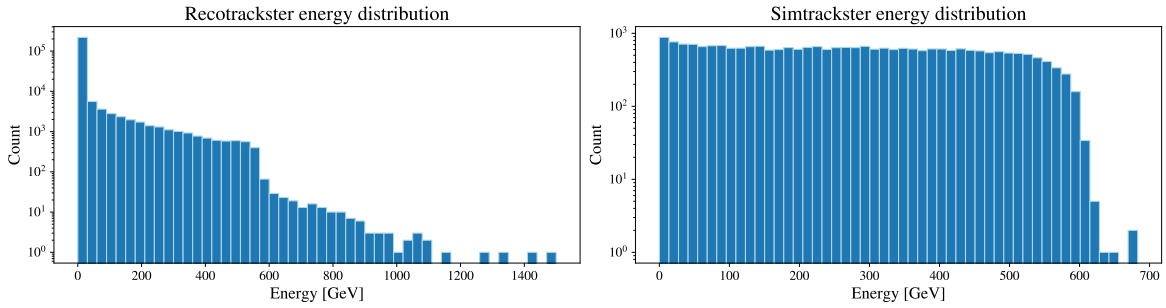


Figure A.1: Energy distribution for reco- and simtracksters for 10 000 events from the multiparticle dataset in 0 PU. Original simulated particle energies are in the range from 10 to 600 GeV, uniformly distributed.

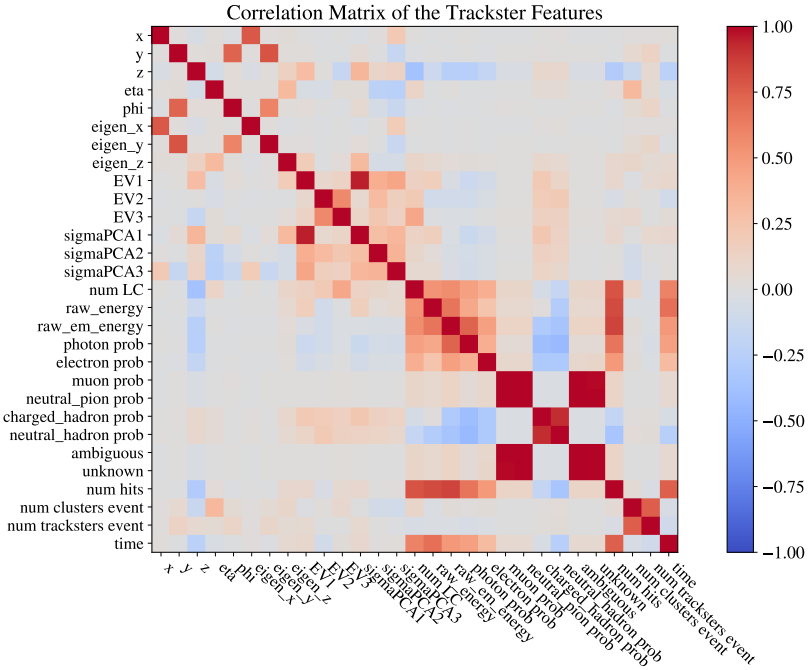


Figure A.2: The correlation matrix for the chosen trackster features for 5000 double pion events in 0 PU dataset.

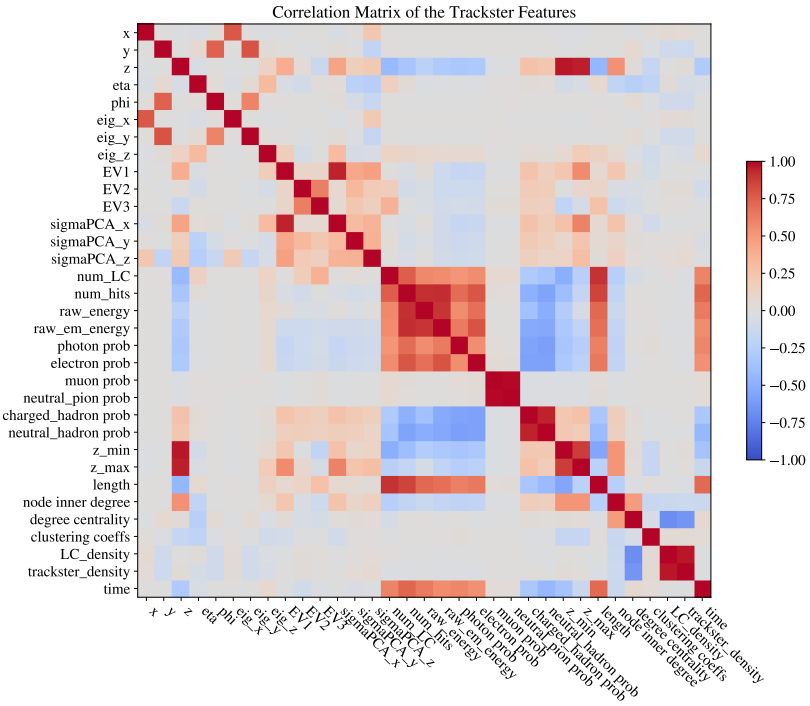


Figure A.3: The correlation matrix for the chosen trackster features for 5000 multiparticle events dataset in 0 PU.

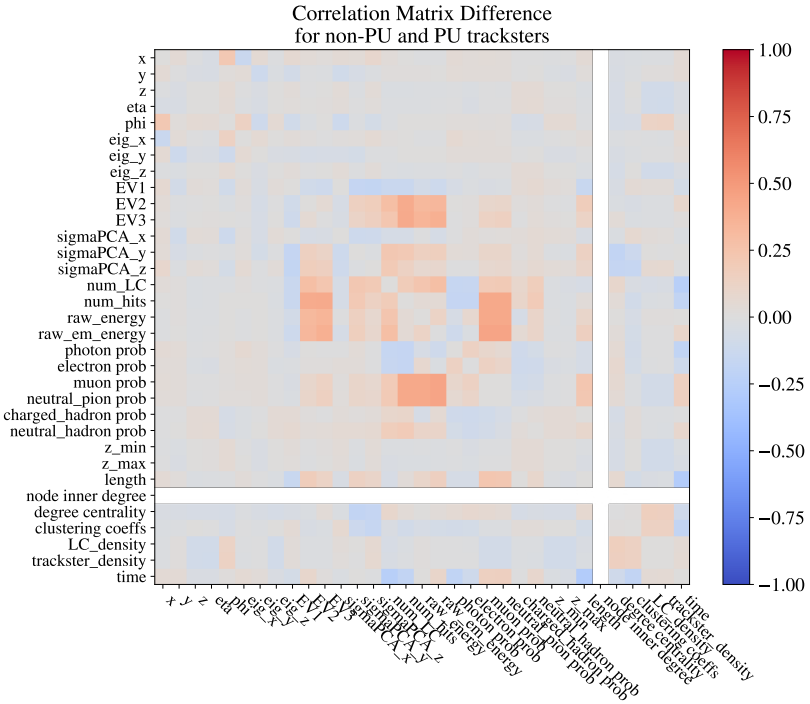


Figure A.4: The correlation matrix difference of the PU and non-PU tracksters for the chosen trackster features in a single pion in 140 PU events. The difference between the two is very subtle. The node's inner degree for PU tracksters not included in the graph, is always zero, indicated by their corresponding white row and column.

A.3 Standard Clustering Methods with Multiparticle Dataset

Table A.7 and Figure A.5 provide additional performance evaluation of the standard clustering methods on the multiparticle dataset. Parameters of these methods are listed in Section 6.5.1.

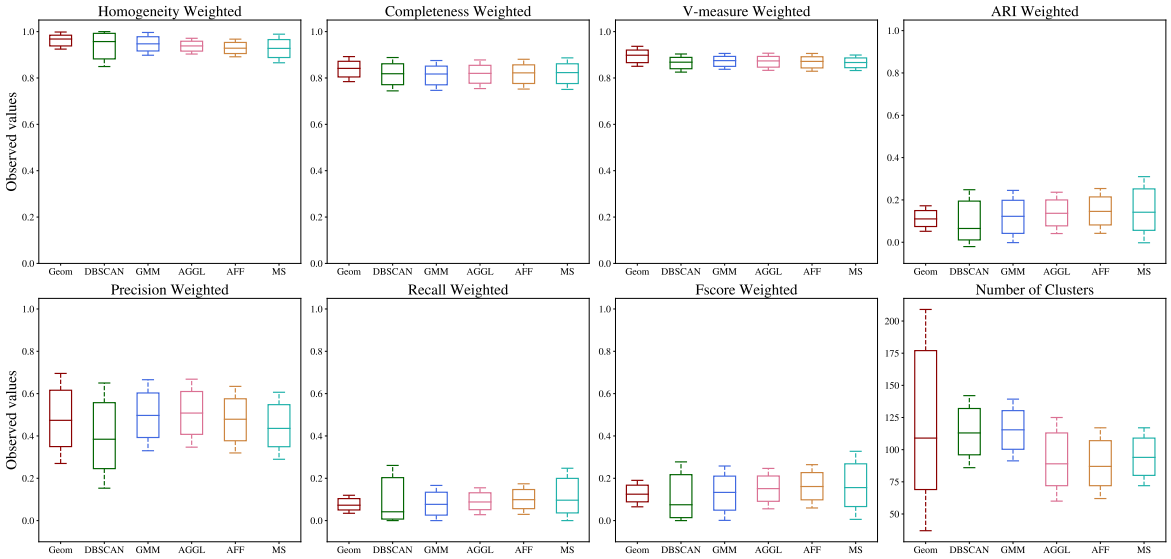


Figure A.5: Evaluation of standard clustering methods versus the geometric linking for the multiparticle dataset.

Table A.7: Multiparticle dataset clustering with standard clustering methods. All evaluation metrics consider the energy of individual tracksters as described in Section 6.5. GMM stands for Gaussian Mixture models, MS for Mean shift, AGG for agglomerative clustering, and AFF for affinity propagation. Best values are highlighted in bold. DBSCAN is additionally weighting point contributions by their energy. The initial average number of tracksters is $\bar{N} = 215.4$.

Algorithm	DBSCAN	GMM	AGG	AFF	MS
Homogeneity	0.930	**0.945**	0.934	0.926	0.934
Completeness	0.813	0.808	0.813	0.813	**0.815**
V-measure	0.863	**0.869**	0.868	0.864	0.864
ARI	0.112	0.126	0.145	0.153	**0.161**
B-Cubed Precision	0.416	**0.509**	0.502	0.476	0.453
B-Cubed Recall	0.115	0.084	0.097	0.106	**0.126**
B-Cubed Fscore	0.124	0.134	0.158	0.167	**0.174**
Avg. num. of trackst.	114.4	113.2	93.3	**90.5**	94.5
Best feature set	$x, y, z, (E)$	x, y, z	x, y, z	η, ϕ, r	x, y, z

A.4 Model Details

Table A.8 provides supplemental information to model training setups discussed in Section 7.4.

Table A.8: Best model parameters for individual datasets

Dataset	Double Pions	Multiparticle	Single Particle in 140 PU
Num. events (Train/Val/Test)	46.5k / 5.8k / 5.8k	19.7k / 4.9k / 4.9k	14.5k / 3.6k / 3.6k
Number of edges	∼10.7 mil.	∼41.1 mil.	∼4.1 mil.
Pos. / Neg. imbalance	78.4% / 21.6%	41.2% / 58.8%	39.8% / 60.2%
Pair-Wise MLP			
Architecture	hidden dim = 256, 3 FC sub-modules	hidden dim = 256, 3 FC sub-modules	hidden dim = 256, 2 FC sub-modules
Loss Function	Quality Focal Loss $\beta = 2$	Focal Loss $\alpha = 0.45$, $\gamma = 2$	Focal Loss $\alpha = 0.40$, $\gamma = 2$
Loss Param.	LR = 10^{-3} Reduced on plateau	LR = 10^{-3} Reduced on plateau	LR = 10^{-3} Cosine annealing
Training epochs	30	25	25
Validation AUC	0.85	0.85	0.99
Confidence threshold	0.90	0.80	0.70
Graph Neural Network			
Architecture	hidden dim = 64 edge dim = 32 4 EdgeConvs	hidden dim = 64 edge dim = 32 4 EdgeConvs	hidden dim = 64 edge dim = 32 3 EdgeConvs
Loss Function	Quality Focal Loss $\beta = 2$	Focal Loss $\alpha = 0.45$, $\gamma = 2$	Focal Loss $\alpha = 0.42$, $\gamma = 2$
Loss Param.	LR = 10^{-3} Cosine annealing	LR = 10^{-3} Cosine annealing	LR = 10^{-4} Cosine annealing
Training epochs	50	45	50
Validation AUC	0.94	0.90	1.00
Confidence threshold	0.85	0.80	0.70

A.5 Clustering Metrics Evaluation

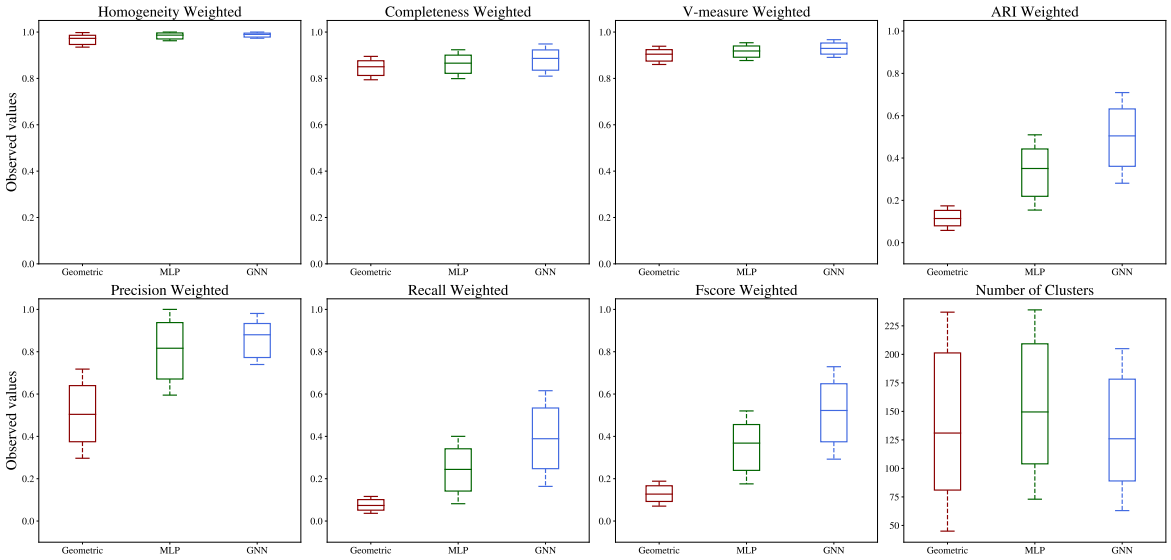


Figure A.6: Multiparticle dataset clustering performance for geometric, pair-wise MLP and GNN linking methods.

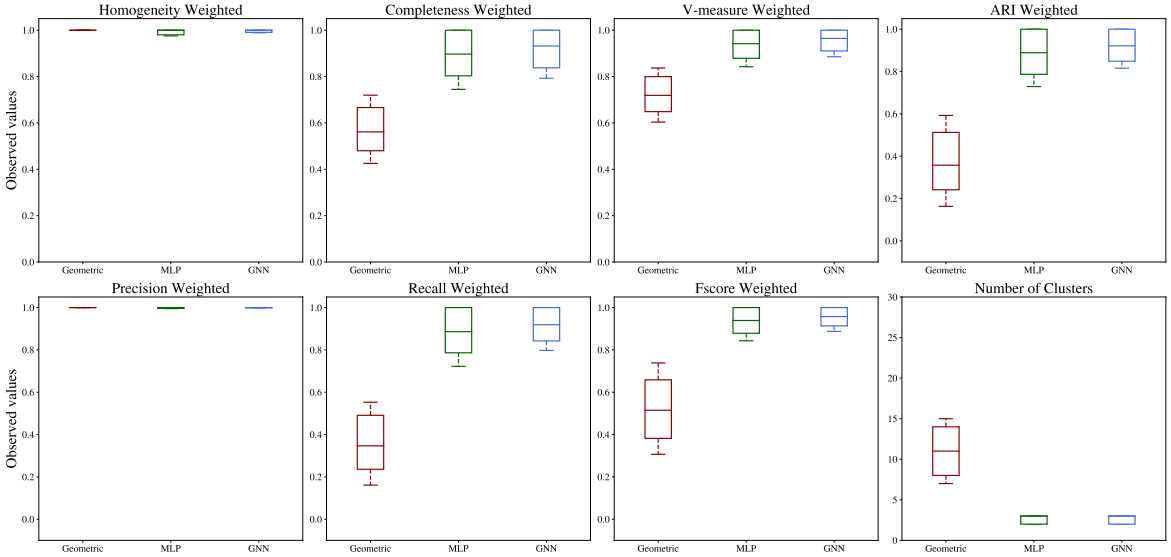


Figure A.7: Pile-up dataset clustering performance for geometric, pair-wise MLP and GNN linking methods.

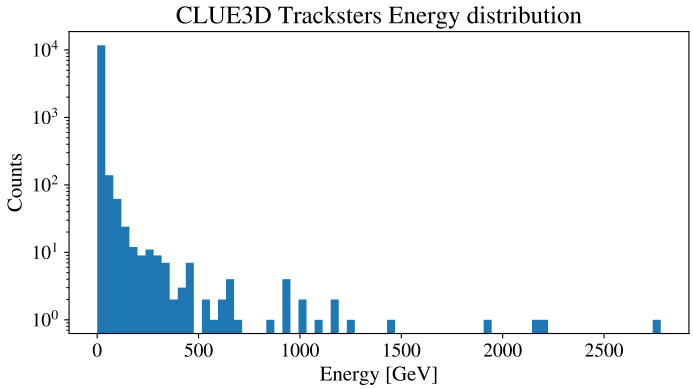


Figure A.10: Distribution of the CLUE3D trackster energies in $t\bar{t}$ events.

A.6 Energy Intersection Over Union

Figures A.8 and A.9 present the EIoU score described in Section 7.6.2 distributions for multiparticle and PU datasets, respectively.

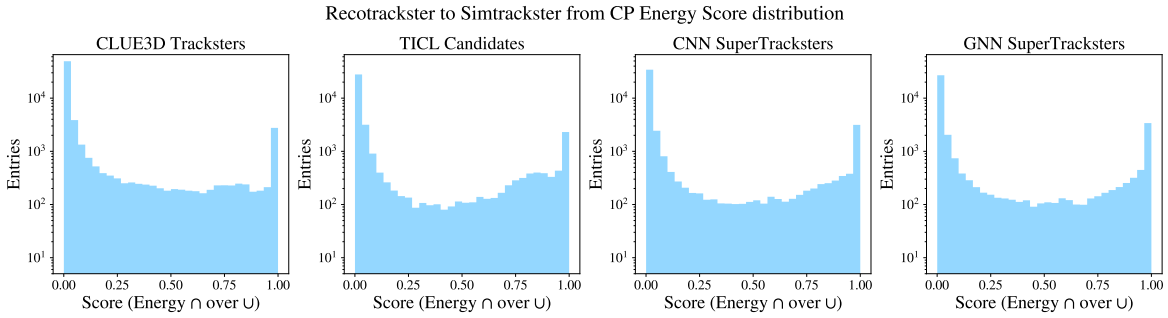


Figure A.8: Multiparticle EIoU score distributions for 1000 events for a) initial tracksters produced by CLUE3D, b) candidates created by geometric linking, c) MLP, and d) GNN supertracksters (from left to right).

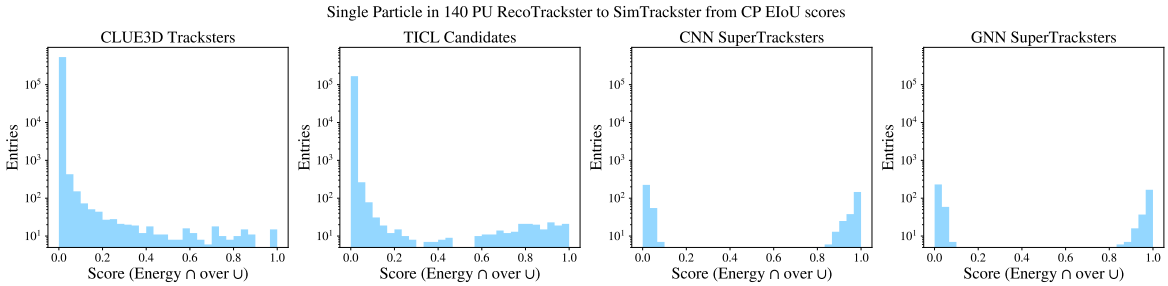
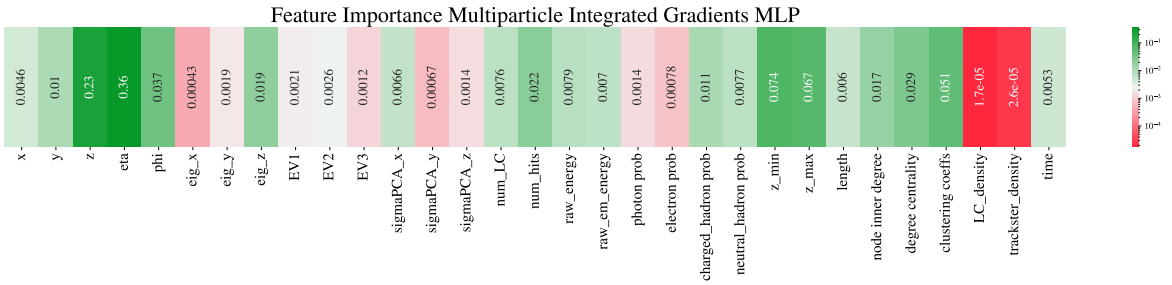


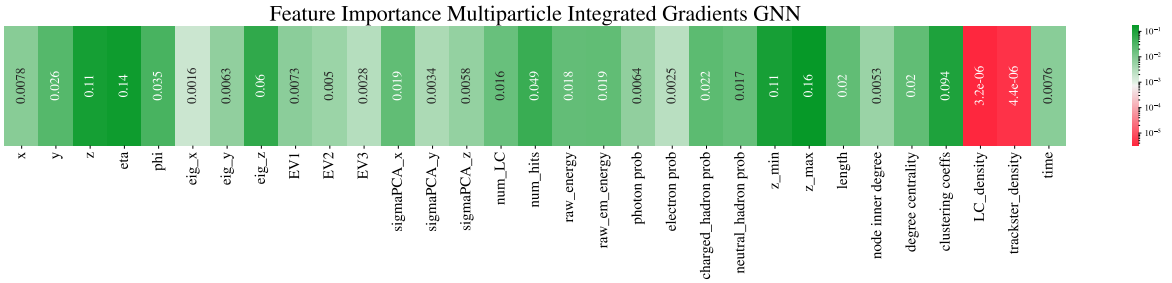
Figure A.9: Single particle in 140 PU EIoU score distributions for 1000 events for a) initial tracksters produced by CLUE3D, b) candidates created by geometric linking, c) MLP, and d) GNN supertracksters (from left to right).

A.7 Model Interpretability

Figures A.11 and A.12 show the importance of individual node features for MLP and GNN networks trained on multiple particles in 0 PU and a single particle in 140 PU datasets, respectively. The importance of the features is estimated through the use of IG, as described in Section 7.6.8.

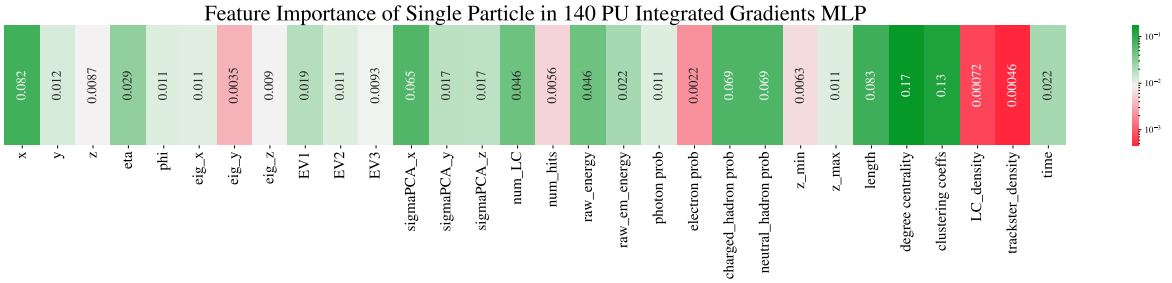


(a) : Feature importance of the MLP model.

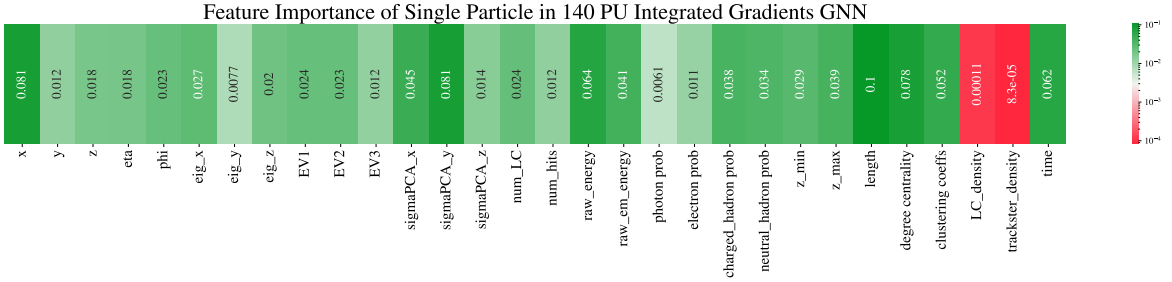


(b) : Feature importance of the GNN model.

Figure A.11: Integrated gradients feature importance for the multiple particles in 0 PU. The values are calculated with respect to the random baselines.



(a) : Feature importance of the MLP model.



(b) : Feature importance of the GNN model.

Figure A.12: Integrated gradients feature importance for the 140 PU dataset. The values are calculated with respect to the random baselines.

Appendix B

Bibliography

[AA+03] Sea Agostinelli, John Allison, et al., *GEANT4—a simulation toolkit*, Nuclear instruments and methods in physics research section A: Accelerators, Spectrometers, Detectors and Associated Equipment **506** (2003), no. 3, 250–303.

[AAAB15] Zahid Ansari, M. F. Azeem, Waseem Ahmed, and A. Vinaya Babu, *Quantitative Evaluation of Performance and Validity Indices for Clustering the Web Navigational Sessions*, 2015.

[ABKS99] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, and Jörg Sander, *OPTICS: Ordering points to identify the clustering structure*, ACM Sigmod record **28** (1999), no. 2, 49–60.

[ABNR17] Giorgio Apollinari, O. Brüning, Tatsushi Nakamoto, and Lucio Rossi, *High luminosity large hadron collider HL-LHC*, arXiv preprint arXiv:1705.08830 (2017).

[ABT+08] Georges Aad, J. Butterworth, J. Thion, U. Bratzler, P. Ratoff, R. Nickerson, J. Seixas, Grabowska-Bold, et al., *The ATLAS experiment at the CERN large hadron collider*, Jinst **3** (2008), S08003.

[AIK20] Juliette Alimena, Yutaro Iiyama, and Jan Kieseler, *Fast convolutional neural networks for identifying long-lived particles in a high-granularity calorimeter*, Journal of Instrumentation **15** (2020), no. 12, P12006.

[AJ+08] A. Alves Jr. et al., *The LHCb detector at the LHC*, Journal of instrumentation **3** (2008), no. 08, S08005.

[AKBD22] Ayman Al-Kababji, Faycal Bensaali, and Sarada Prasad Dakua, *Scheduling Techniques for Liver Segmentation: ReduceLROnPlateau Vs OneCycleLR*, 2022.

[AQA+08] Kenneth Aamodt, A. Abrahantes Quintana, R. Achenbach, S. Acounis, D. Adamová, C. Adler, M. Aggarwal, F. Agnese, G. Aglieri Rinella, Z. Ahammed, et al., *The ALICE experiment at the CERN LHC*, Journal of Instrumentation **3** (2008), no. 08, S08002.

[Arc18] Roberta Arcidiacono, *The CMS ECAL Phase-2 Upgrade for High Precision Timing and Energy Measurements*, 2018 IEEE Nuclear Science Symposium and Medical Imaging Conference Proceedings (NSS/MIC), IEEE, 2018, pp. 1–2.

[BGLL08] Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre, *Fast unfolding of communities in large networks*, Journal of statistical mechanics: theory and experiment **2008** (2008), no. 10, P10008.

[BGS22] Franco Bedeschi, Loukas Gouskos, and Michele Selvaggi, *Jet flavour tagging for future colliders with fast simulation*, The European Physical Journal C **82** (2022), no. 7, 646.

[BKH16] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton, *Layer Normalization*, 2016.

[BKV+20] Emil Bols, Jan Kieseler, Mauro Verzetti, Markus Stoye, and Anna Stakia, *Jet flavour classification using DeepJet*, Journal of Instrumentation **15** (2020), no. 12, P12012.

[BLZ+19] Junjie Bai, Fang Lu, Ke Zhang, et al., *ONNX: Open Neural Network Exchange*, https://github.com/onnx/onnx, 2019.

[BOSB10] Kay Henning Brodersen, Cheng Soon Ong, Klaas Enno Stephan, and Joachim M Buhmann, *The balanced accuracy and its posterior distribution*, 2010 20th international conference on pattern recognition, IEEE, 2010, pp. 3121–3124.

[BR97] Rene Brun and Fons Rademakers, *ROOT—An object oriented data analysis framework*, Nuclear instruments and methods in physics research section A: accelerators, spectrometers, detectors and associated equipment **389** (1997), no. 1-2, 81–86.

[BTdF00] Joel N. Butler and Tommaso Tabarelli de Fatis, *A MIP timing detector for the CMS phase-2 upgrade*, Tech. report, Fermi National Accelerator Lab.(FNAL), Batavia, IL (United States), 1900.

[Bur21] Carsten Burgard, *Example: Standard model of physics*, online, 2021, [Accessed: 2023-02-19] https://texample.net/tikz/examples/model-physics/.

[CC75] Leland Lavele Carter and Edmond Darrell Cashwell, *Particle-transport simulation with the Monte Carlo method*, Tech. report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 1975.

[CKP+17] Federico Carminati, Gulrukh Khattak, Maurizio Pierini, Sofia Vallecorsa, Amir Farbin, B. Hooberman, W. Wei, M. Zhang, B. Pacela, M. Spiropulu Vitorial, et al., *Calorimetry with deep learning: particle classification, energy regression, and simulation for high-energy physics*, Workshop on deep learning for physical sciences (DLPS 2017), NIPS, 2017.

[CKSS15] Josh Cogan, Michael Kagan, Emanuel Strauss, and Ariel Schwartzman, *Jet-images: computer vision inspired techniques for jet tagging*, Journal of High Energy Physics **2015** (2015), no. 2, 1–16.

[CMW+17] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia, *Multi-view 3D object detection network for autonomous driving*, Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2017, pp. 1907–1915.

[Col08] CMS Collaboration, *The CMS experiment at the CERN LHC*, Jinst **3** (2008), S08004.

[Col12] CMS Collaboration, *Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC*, Physics Letters B **716** (2012), no. 1, 30–61.

[Col13] ———, *Identification of b-quark jets with the CMS experiment*, Journal of Instrumentation **8** (2013), no. 04, P04013.

[Col17a] ———, *New developments for jet substructure reconstruction in CMS*, Detector Performance Summary: CMS-DPS-17-027, https://cds. cern. ch/record/2275226 (2017).

[Col17b] ———, *The phase-2 upgrade of the cms endcap calorimeter*, CMS Technical Design Report CERN-LHCC-2017-023. CMS-TDR-019, CERN (2017).

[Col21] ———, *The Phase-2 Upgrade of the CMS Data Acquisition and High Level Trigger*, Tech. report, CERN, Geneva, 2021.

[Col23] ———, *Interactive Slice of the CMS detector*, online, 2023, [Accessed: 2023-02-13] https://cms-docdb.cern.ch/cgi-bin/PublicDocDB/ShowDocument?docid=4172.

[CP15] Miguel Á. Carreira-Perpiñán, *A review of mean-shift algorithms for clustering*, 2015.

[CSS10] Matteo Cacciari, Gavin P Salam, and Gregory Soyez, *FastJet*, 2010.

[DHH+18] Javier Duarte, Song Han, Philip Harris, Sergo Jindariani, Edward Kreinar, Benjamin Kreis, Jennifer Ngadiuba, Maurizio Pierini, Ryan Rivera, Nhan Tran, et al., *Fast inference of deep neural networks in FPGAs for particle physics*, Journal of Instrumentation **13** (2018), no. 07, P07027.

[DPCPR20] Antonio Di Pilato, Ziheng Chen, Felice Pantaleo, and Marco Rovere, *Reconstruction in an imaging calorimeter for HL-LHC*, Journal of Instrumentation **15** (2020), no. 06, C06023.

[dSCF+12] Marcilio C. de Souto, André L. Coelho, Katti Faceli, Tiemi C. Sakata, Viviane Bonadia, and Ivan G. Costa, *A comparison of external clustering evaluation indices in the context of imbalanced data sets*, 2012 Brazilian Symposium on Neural Networks, IEEE, 2012, pp. 49–54.

[DV22] Javier Duarte and Jean-Roch Vlimant, *Graph Neural Networks for particle tracking and reconstruction*, Artificial intelligence for high energy physics, World Scientific, 2022, pp. 387–436.

[EB08] Lyndon Evans and Philip Bryant, *LHC machine*, Journal of instrumentation **3** (2008), no. 08, S08001.

[EBR17] Daniel Edler, Ludvig Bohlin, and Martin Rosvall, *Mapping Higher-Order Network Flows in Memory and Multilayer Networks with Infomap*, Algorithms **10** (2017), no. 4, 112.

[FB81] Martin A. Fischler and Robert C. Bolles, *Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography*, Communications of the ACM **24** (1981), no. 6, 381–395.

[FCM+18] Steven Farrell, Paolo Calafiura, Mayur Mudigonda, Dustin Anderson, Jean-Roch Vlimant, Stephan Zheng, Josh Bendavid, Maria Spiropulu, Giuseppe Cerati, Lindsey Gray, et al., *Novel deep learning methods for track reconstruction*, arXiv preprint arXiv:1810.06111 (2018).

[FL19] Matthias Fey and Jan Eric Lenssen, *Fast graph representation learning with PyTorch Geometric*, arXiv preprint arXiv:1903.02428 (2019).

[Fwy] Fwyzard, *FWYZARD/Circles: Scripts and web pages to visualise CMSSW resource usage with CarrotSearch circles.*

[GB10] Xavier Glorot and Yoshua Bengio, *Understanding the difficulty of training deep feedforward neural networks*, Proceedings of the thirteenth international conference on artificial intelligence and statistics, JMLR Workshop and Conference Proceedings, 2010, pp. 249–256.

[HMvdW+20] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, et al., *Array programming with NumPy*, Nature **585** (2020), no. 7825, 357–362.

[HRD+20] Aneesh Heintz, Vesal Razavimaleki, Javier Duarte, Gage DeZoort, Isobel Ojalvo, Savannah Thais, Markus Atkinson, Mark Neubauer, Lindsey Gray, Sergo Jindariani, et al., *Accelerated charged particle tracking with graph neural networks on FPGAs*, arXiv preprint arXiv:2012.01563 (2020).

[HSSC08] Aric Hagberg, Pieter Swart, and Daniel S Chult, *Exploring network structure, dynamics, and function using NetworkX*, Tech. report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[Hun07] J. D. Hunter, *Matplotlib: A 2D graphics environment*, Computing in Science & Engineering **9** (2007), no. 3, 90–95.

[Jek21] Jaroslavceva Jekaterina, *Image Retrieval via CNNs in TensorFlow2*, B.S. thesis, CTU, FEE. Center of machine perception, 2021.

[JFC+20] Xiangyang Ju, Steven Farrell, Paolo Calafiura, Daniel Murnane, Lindsey Gray, Thomas Klijnsma, Kevin Pedro, Giuseppe Cerati, Jim Kowalkowski, Gabriel Perdue, et al., *Graph neural networks for particle reconstruction in high energy physics detectors*, arXiv preprint arXiv:2003.11603 (2020).

[Kai08] Marcus Kaiser, *Mean clustering coefficients: the role of isolated nodes and leafs on clustering measures for small-world networks*, New Journal of Physics **10** (2008), no. 8, 083042.

[KB17] Diederik P. Kingma and Jimmy Ba, *Adam: A Method for Stochastic Optimization*, 2017.

[Kie20] Jan Kieseler, *Object condensation: one-stage grid-free multi-object reconstruction in physics detectors, graph, and image data*, The European Physical Journal C **80** (2020), 1–12.

[KL51] Solomon Kullback and Richard A Leibler, *On information and sufficiency*, The annals of mathematical statistics **22** (1951), no. 1, 79–86.

[KM59] H. Koch and J. Motz, *Bremsstrahlung cross-section formulas and related data*, Reviews of modern physics **31** (1959), no. 4, 920.

[KSH+12] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, F. Pereira, C. Burges, L. Bottou, and KQ Weinberger, *Advances in neural information processing systems*, 2012.

[KST+17] Vardan Khachatryan, Albert M. Sirunyan, Armen Tumasyan, Wolfgang Adam, E. Asilar, Thomas Bergauer, Johannes Brandstetter, Erica Brondolin, Marko Dragicevic, Janos Erö, et al., *The CMS trigger system*, Journal of Instrumentation **12** (2017), no. 01, P01020–P01020.

[KTW+21] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan, *Supervised Contrastive Learning*, 2021.

[LGG+18] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár, *Focal Loss for Dense Object Detection*, 2018.

[LH17] Ilya Loshchilov and Frank Hutter, *SGDR: Stochastic Gradient Descent with Warm Restarts*, 2017.

[lhc] *CERN accelerating science*, online, [Accessed: 2023-02-13] https://hilumilhc.web.cern.ch/.

[Lip12] Christian Lippmann, *Particle identification*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **666** (2012), 148–172.

[LLW18] Sehwook Lee, Michele Livan, and Richard Wigmans, *On the limits of the hadronic energy resolution of calorimeters*, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment **882** (2018), 148–157.

[Lop22] E. Lopienska, *The CERN accelerator complex, layout in 2022.*

[LVC+19] Alex Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom, *Pointpillars: Fast encoders for object detection from point clouds*, Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2019, pp. 12697–12705.

[LWW+20] Xiang Li, Wenhai Wang, Lijun Wu, Shuo Chen, Xiaolin Hu, Jun Li, Jinhui Tang, and Jian Yang, *Generalized Focal Loss: Learning Qualified and Distributed Bounding Boxes for Dense Object Detection*, 2020.

[Mac67] J. MacQueen, *Classification and analysis of multivariate observations*, 5th Berkeley Symp. Math. Statist. Probability, University of California Los Angeles LA USA, 1967, pp. 281–297.

[MHA17] Leland McInnes, John Healy, and Steve Astels, *HDBSCAN: Hierarchical density based clustering*, J. Open Source Softw. **2** (2017), no. 11, 205.

[ML11] Fionn Murtagh and Pierre Legendre, *Ward's hierarchical clustering method: clustering criterion and agglomerative algorithm*, arXiv preprint arXiv:1111.6285 (2011).

[MQY23] Omar Maddouri, Xiaoning Qian, and Byung-Jun Yoon, *Geometric Affinity Propagation for Clustering With Network Knowledge*, IEEE Transactions on Knowledge and Data Engineering (2023), 1–18.

[Nan22] Abhirikshma Nandi, *New Techniques for Reconstruction in the CMS High Granularity Calorimeter*, Master's thesis, RWTH Aachen University, December 2022, Submitted to the Faculty of Mathematics, Computer Science and Natural Sciences.

[NG04] M. E. J. Newman and M. Girvan, *Finding and evaluating community structure in networks*, Physical Review E **69** (2004), no. 2.

[PDB+20] Jim Pivarski, Pratyush Das, Chris Burr, Dmitri Smirnov, Matthew Feickert, Tamas Gal, et al., *scikit-hep/uproot: 3.12.0*, July 2020.

[PES+20] Jim Pivarski, Charles Escott, Nicholas Smith, Michael Hedges, et al., *scikit-hep/awkward-array: 0.13.0*, July 2020.

[PG+19] Adam Paszke, Sam Gross, et al., *PyTorch: An Imperative Style, High-Performance Deep Learning Library*, Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 8024–8035.

[PR22] Felice Pantaleo and Marco Rovere, *The Iterative Clustering framework for the CMS HGCAL Reconstruction*, Tech. report, 2022.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, *Scikit-learn: Machine Learning in Python*, Journal of Machine Learning Research **12** (2011), 2825–2830.

[QG20] Huilin Qu and Loukas Gouskos, *Jet tagging via particle clouds*, Physical Review D **101** (2020), no. 5, 056019.

[QKIP19] Shah Rukh Qasim, Jan Kieseler, Yutaro Iiyama, and Maurizio Pierini, *Learning representations of irregular particle-detector geometry with distance-weighted graph networks*, The European Physical Journal C **79** (2019), no. 7, 1–11.

[QLQ22] Huilin Qu, Congqiao Li, and Sitian Qian, *Particle transformer for jet tagging*, International Conference on Machine Learning, PMLR, 2022, pp. 18281–18292.

[QSMG17] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas, *PointNet: Deep learning on point sets for 3D classification and segmentation*, Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 652–660.

[QYSG17] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas, *Pointnet++: Deep hierarchical feature learning on point sets in a metric space*, Advances in neural information processing systems **30** (2017).

[RCDP+20] Marco Rovere, Ziheng Chen, Antonio Di Pilato, Felice Pantaleo, and Chris Seez, *CLUE: a fast parallel clustering algorithm for high granularity calorimeters in high-energy physics*, Frontiers in big Data **3** (2020), 591315.

[Rey09] Douglas A. Reynolds, *Gaussian Mixture Models*, Encyclopedia of Biometrics, 2009.

[RH07] Andrew Rosenberg and Julia Hirschberg, *V-measure: A conditional entropy-based external cluster evaluation measure*, Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL) (Prague, Czech Republic), Association for Computational Linguistics, June 2007, pp. 410–420.

[RS16] Nadia Rahmah and Imas Sukaesih Sitanggang, *Determination of optimal epsilon (eps) value on dbscan algorithm to clustering data on peatland hotspots in sumatra*, IOP conference series: earth and environmental science, vol. 31, IoP Publishing, 2016, p. 012012.

[SAC+15] Torbjörn Sjöstrand, Stefan Ask, Jesper R. Christiansen, Richard Corke, Nishita Desai, Philip Ilten, Stephen Mrenna, et al., *An introduction to PYTHIA 8.2*, Computer physics communications **191** (2015), 159–177.

[Sc+17] Albert M. Sirunyan, CMS collaboration, et al., *Particle-flow reconstruction and global event description with the CMS detector*, JINST **12** (2017), no. 10, P10003.

[SGT+08] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini, *The graph neural network model*, IEEE transactions on neural networks **20** (2008), no. 1, 61–80.

[SL18] Samuel L. Smith and Quoc V. Le, *A Bayesian Perspective on Generalization and Stochastic Gradient Descent*, 2018.

[SSE+17] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu, *DBSCAN revisited, revisited: why and how you should (still) use DBSCAN*, ACM Transactions on Database Systems (TODS) **42** (2017), no. 3, 1–21.

[STY17] Mukund Sundararajan, Ankur Taly, and Qiqi Yan, *Axiomatic Attribution for Deep Networks*, 2017.

[Val22] Davide Valsecchi, *Deep learning techniques for energy clustering in the CMS ECAL*, arXiv preprint arXiv:2204.10277 (2022).

[VCC+17] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al., *Graph attention networks*, stat **1050** (2017), no. 20, 10–48550.

[VdMH08] Laurens Van der Maaten and Geoffrey Hinton, *Visualizing data using t-SNE*, Journal of machine learning research **9** (2008), no. 11.

[VL07] Ulrike Von Luxburg, *A tutorial on spectral clustering*, Statistics and computing **17** (2007), 395–416.

[WR] CMS Collaboration Wahid Redjeb, *Poster: The TICL reconstruction at the CMS Phase-2 High Granularity Calorimeter Endcap*.

[WSL+19] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon, *Dynamic graph CNN for learning on point clouds*, Acm Transactions On Graphics (tog) **38** (2019), no. 5, 1–12.

[XHLJ18] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka, *How powerful are graph neural networks?*, arXiv preprint arXiv:1810.00826 (2018).

[XWCL15] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li, *Empirical evaluation of rectified activations in convolutional network*, arXiv preprint arXiv:1505.00853 (2015).

[ZCH+20] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun, *Graph neural networks: A review of methods and applications*, AI open **1** (2020), 57–81.