Pattern recognition for particle shower reconstruction

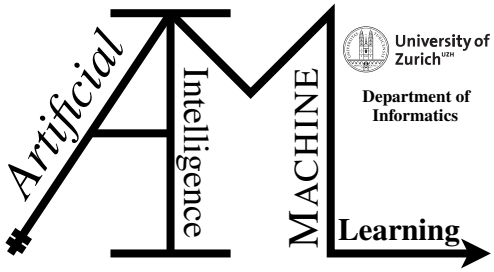
Exploring AI-based methods for calorimetric clustering at the CMS HGCAL

Master's Thesis

Eduard Cuba
19-771-393

Submitted on
March 1, 2023

Thesis Supervisor
Prof. Dr. Manuel Günther
Felice Pantaleo, PhD



Master's Thesis

Author: Eduard Cuba, eduard.cuba@uzh.ch

Project period: September 1, 2022 - March 1, 2023

Artificial Intelligence and Machine Learning Group
Department of Informatics, University of Zurich

Acknowledgements

I would like to sincerely thank Prof. Dr. Manuel Günther and Felice Pantaleo, Ph.D., for their supervision. This endeavor would not have been possible without Dr. Maria Girone, my CERN supervisor and the leader of data-driven use-cases towards exascale within CoE RAISE[1]. I would also like to acknowledge Wahid Redjeb for his support with dataset preparation and evaluation in the production environment, Ian Fisk, Ph.D. from the Flatiron Institute for support with the benchmarking and providing a testing environment for pre-processing and training, and Jekaterina Jaroslavceva for valuable discussions about the model architecture and training. This work was supported by the CoE RAISE within the scope of methods for event reconstruction and classification at the CERN HL-LHC.

[1] CoE RAISE: https://www.coe-raise.eu

Abstract

The number of collisions in the upcoming runs of the Large Hadron Collider at CERN will increase significantly. The increasing amount of data and a higher granularity of the newly developed calorimetric detectors pose a substantial data volume and complexity challenge to the current particle shower reconstruction algorithms. This thesis aims to explore the feasibility of machine-learned models scalable to large data volumes for improving the reconstruction quality of calorimetric particle showers via calorimetric clustering. The goal of calorimetric clustering is to recognize and reconnect fragmented energetic components of particle showers described by three-dimensional spatial structures called tracksters. We show that machine-learned models are viable methods for calorimetric clustering and provide a significant reconstruction performance benefit over classical clustering approaches. Furthermore, we investigate the feasibility of node classification and link prediction problem formulations for training graph neural networks. Experimentally, we show that graph-based models provide a better reconstruction performance, more compact data representation, and better scalability on the tested datasets than feed-forward neural networks.

Contents

Chapter 1

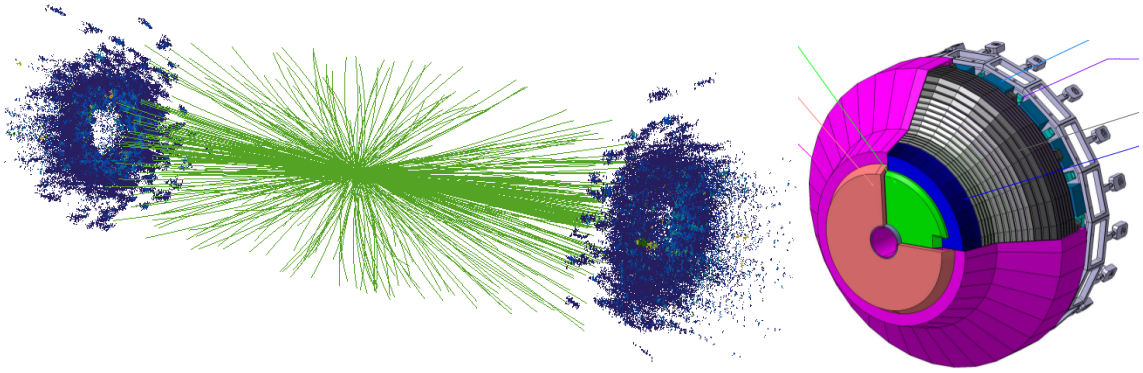
Introduction

The particle shower reconstructions in experimental physics traditionally rely on complex algorithmic workflows. However, with the rising amounts of collected data and increasing data granularity, the traditional workflows fail to keep up with the processing demands. The developments in artificial intelligence-based techniques show promising advances in complexity, scalability, and physics performance. Furthermore, data-driven AI methods enable quick adaptation of the algorithms to problems by learning from data. Given enough data, learned parameterized models often provide better results than traditional approaches. However, several aspects of their deployment need to be considered. Switching from engineered algorithmic workflows to data-driven models presents a complete shift in the computing paradigm, includes a different design process, requires new task formulations, and depends on extensive model training, hyperparameter tuning, and manipulating extensive quantities of data. This work explores modern, scalable machine-learned methods for shower reconstructions in experimental particle physics. Primarily, it is focused on the task of calorimetric clustering; toward meeting the increasing processing demands with higher data granularity, better reconstruction quality, and supporting new scientific discoveries.

1.1 Artificial intelligence at the Exascale

The European effort CoE RAISE[1] aims to explore and exploit AI technologies for complex applications running on future Exascale systems (systems capable of processing exabytes of data, with ExaFLOPS computing performance). Its goal is to develop innovative AI methods for heterogeneous high-performance computing (HPC) workflows capable of scaling towards exascale in simulation and data-driven processes. Data-driven use cases generally consist of large datasets obtained from measurement devices. In experimental particle physics, measurement devices (detectors) capture up to 40 million particle collisions per second, with hundreds of particles interacting and producing new particles in each collision. With the energy of the collisions being gradually increased and the measurement capabilities of the detectors reaching much greater granularity, the processing workflows need an improvement in the order of magnitude to keep up. Given the expected trend in the computing capacity increase at CERN, the European Organization for Nuclear Research, particle reconstruction algorithms need to gain a factor of 10 in performance to keep up with the processing requirements (Pilato et al., 2020). Within the CoE RAISE mission to explore scalable data-driven AI workflows, we investigate novel approaches employing AI methods, mainly learned parameterized models, targeting higher data granularity and better particle shower reconstruction performance.

[1] CoE RAISE: https://www.coe-raise.eu/about



(a) A particle collision with energy deposits captured by calorimeters on both sides

(b) The CMS High-Granularity Calorimeter

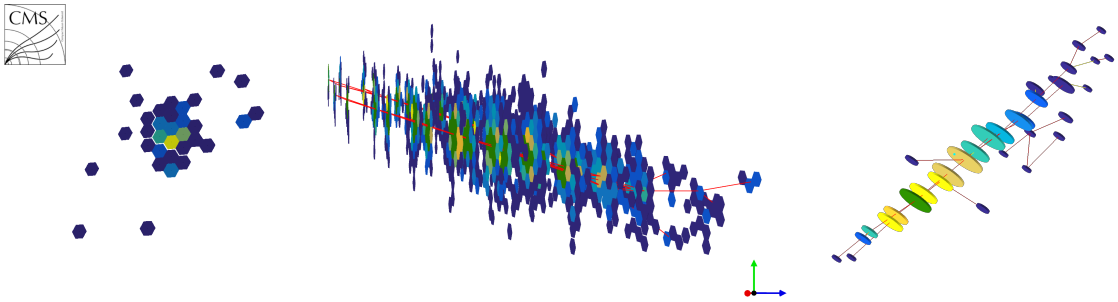
Figure 1.1: CALORIMETRIC PARTICLE SHOWERS. *Figure (a) shows a particle collision event, with simulated particle tracks in green, captured by calorimeters on both sides. Upon interaction with the calorimeter, the particles deposit energies into the sensors, producing an energetic cascade. Figure (b) shows a rendering of the High-Granularity Calorimeter designed by the CMS experiment at CERN (Rapacz, 2017).*

1.2 Particle shower reconstruction

Particle shower reconstruction processes energetic collision measurements captured by the detector and infers particle properties and identification probabilities. This work is primarily focused on calorimetric reconstructions. Calorimeters are a particular type of detector that measures the energy deposited by particle showers on a dense pixel grid organized in several layers. Given the spatial structure of the calorimeter, they work as large 3D cameras, capturing the cascade of particle interactions on each layer and measuring the deposited energy, see Figure 1.1. One snapshot of the deposited energies in the detector is called an **event**. Each event represents a collision of two bunches (swarms of particles) traveling in the opposite direction of the particle accelerator. The massive energy the collisions create turns into the mass of new particles that trigger interactions with the detector. The background of particle collisions is further discussed in Section 3.1.

The goal of the calorimeter is to capture the energetic properties of the newly produced particles. The captured energy deposits are then reconstructed into particle showers by trying to reconnect all energy deposits originating from the same initial particle. In other words, the reconstruction aims to assign a label to each energy deposit (hit) in the calorimeter. In the ideal case, all hits from a single particle (and only those hits) share the same label. Related work refers to this problem as **calorimetric clustering** (Ju et al., 2020). The produced clusters of energy deposits are then connected into particle showers via a process called **linking** and used to determine the properties of the initial high-energy particle that produced the shower.

Standard particle shower reconstruction algorithms such as ParticleFlow (CMS Collaboration, 2009) combine classical calorimetric clustering algorithms calibrated to a specific shower type and a linking algorithm connecting the energy clusters into particle showers. This work investigates AI-based approaches to improve calorimetric clustering algorithms, primarily focusing on machine-learned models.



(a) Hits in a single layer-cluster (b) Hits from 2 particles (red line) across layers (c) Layer-clusters connected into a trackster

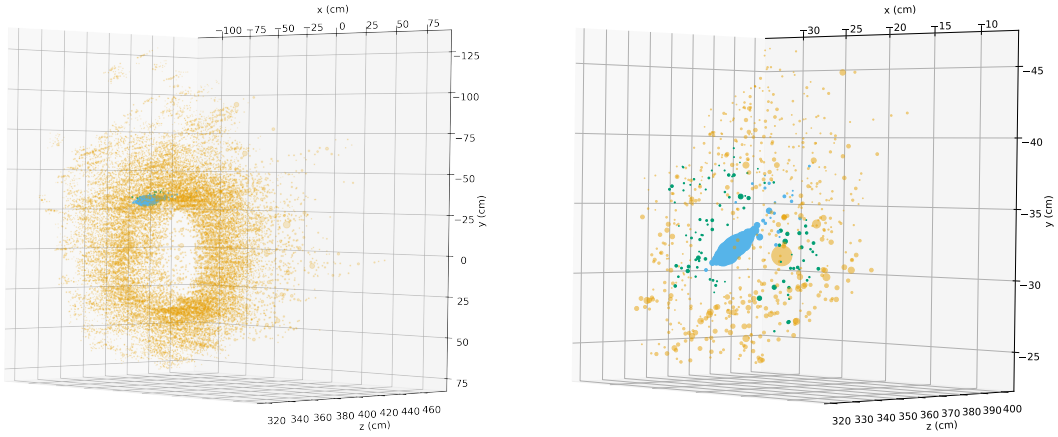
Figure 1.2: STAGES OF CALORIMETRIC CLUSTERING IN HGCAL. *First, energy deposits (hits) are clustered into topological structures called layer-clusters (a). Particles traversing the calorimeter layers form particle shower; Figure (b) shows reconstructed hits deposited by two particles interacting with the calorimeter. Then, layer-clusters created from reconstructed hits connected into spatial structures called tracksters (c).*

1.2.1 Calorimetric clustering at CMS HGCAL

Specifically, this work considers the calorimetric clustering problem at the CMS High-Granularity Calorimeter (HGCAL, further introduced in Section 3.1.3). The HGCAL reconstruction process is split into several stages. Each stage produces higher-level clustering objects as shown in Figure 1.2. The energy deposits (hits) are first topologically clustered within the calorimeter layers into structures called **layer-clusters**. Each layer-cluster seeks to collect all deposits from a single particle shower on the given layer. A particle interacting with successive detector layers creates an elongated ellipsoid-like cascade of layer-clusters shown in Figure 1.2(b). The following steps aim to reconnect these layer-clusters of a single particle shower across the detector layers, forming structures called **trackster** shown in Figure 1.2(c). In the scope of this work, we also use simulated data. The term **simtrackster** refers to a simulated trackster containing all layer-clusters of the particle shower. Thus, each simtrackster represents a single complete particle shower.

Pattern recognition in calorimetric clustering

The formation of tracksters is an iterative process, representing a pattern recognition problem. The term comes from recognizing individual particle showers within many layer-clusters created by overlapping showers. First, a conservative three-dimensional spatial clustering is applied, creating many smaller tracksters (also referred to as trackster fragments). Then, tracksters are reconnected with other nearby tracksters formed by the same particle by a process called **trackster smoothing**. The goal of trackster smoothing is to recover locally connected components of the particle shower (without gaps along the shower axis). Finally, a process called **linking** is applied to reconnect components of the particle shower over longer distances, where the reconnected tracksters may be separated by gaps in layers. The difference between these two steps is the locality. While smoothing addresses reconstructing locally connected components of the shower, linking associates these locally connected components of the same particle shower over larger distances, handling branching showers of the same shower, and particles producing multiple showers in different parts of the calorimeter (with gaps along the shower axis). The complete pattern recognition process is described in Section 3.3.3.



(a) Photon trackster within the endcap

(b) Photon trackster within its neighborhood

Figure 1.3: RECONSTRUCTION OF A PHOTON TRACKSTER. *Visualization of reconstructed layer-clusters of a photon shower rendered by their barycenters and energy within the background of other particle showers. Figure (a) shows the photon trackster within the calorimeter. Figure (b) shows a zoomed-in view into the area around the reconstructed trackster. The largest photon trackster is shown in blue, and neighboring tracksters from the same simulated particle (simtrackster) are shown in green. The metrics by which the tracksters from the same simtrackster are identified are discussed in Section 4.2.1.*

1.2.2 Problem definition

The part of calorimetric clustering addressed in this work is called **trackster smoothing**. Trackster smoothing is the intermediate step of the pattern recognition process, aiming to improve the particle shower reconstruction on the local level, primarily by reconnecting disconnected fragments (low-energy tracksters) to larger, connected components (high-energy tracksters) of the reconstructed shower as shown in Figure 1.3. Tracksters and layer-clusters are described by a series of features (further described in Section 4.2), most importantly their:

1. **barycenter**: 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 trackster layer-clusters (on the trackster level),

2. **energy**: the total energy of the object, computed as the sum of layer-cluster hit energies or trackster layer-clusters energies.

The task can be formulated as a **clustering problem**: given a description of the tracksters, find an assignment in which tracksters containing layer-clusters of the same simtrackster are connected, and tracksters containing layer-clusters of different simtracksters are kept separate. However, the true simtracksters are naturally not known in reconstruction. Therefore, the algorithm has to rely on spatial, energetic, and other trackster attributes to recognize which tracksters were produced by the same particle and thus belong to the same simtrackster in the ground truth data.

Alternatively, the task can be defined on a local, trackster, level as given the trackster and its description, identify all tracksters in the area around the trackster (neighborhood) originating

from the same particle shower (sharing layer-clusters with the same simtrackster). Compared to the clustering formulation above, this formulation allows us to specify the task as a **classification problem** and use the related techniques. Furthermore, this formulation is preferred when the ground truth (simtracksters) is only available for selected particle showers within the calorimeter, and the overall clustering cannot be evaluated.

Available data

Due to the spatial properties of the tracksters and layer-clusters, each collision event is described by a **point cloud** – an unordered set of energetic points in space. Datasets with various levels of complexity and particle shower overlap are available. Generally, two problem settings are considered. First are single or multi-particle datasets, where all hits and layer-clusters in tracksters are contained in simtracksters (ground truth data is available for every hit). Second, we consider datasets with additional background interactions, referred to as **pile-up**, for which the ground truth is unavailable. In this case, the simtracksters are available only for selected particle showers, such as the photon shower in Figure 1.3. The goal is to recover all tracksters representing the simtrackster from the background. The detailed data characteristics are described in Section 4.1.2.

Problem evaluation

Independent of the selected problem formulation and trackster smoothing method, the overall reconstruction result can still be regarded and evaluated as clustering. For datasets where the simulated data is available for all objects in the calorimeter, the overall reconstruction of the event can be evaluated by using clustering metrics introduced in Section 5.1.2.

In pile-up datasets where simulated data is available only for a part of the reconstructed event, we focus on the reconstruction quality of a single particle shower (represented by a single simtrackster). In physics, this process is also referred to as improving the **energy resolution** of a particle shower reconstruction by minimizing the absolute difference between the simulated and reconstructed shower energy (Ochando, 2017). Standard confusion matrix metrics such as precision and recall can be applied to the layer-cluster collection of the simtrackster and reconstructed tracksters to describe the reconstruction quality. Related work on calorimetric clustering also commonly uses **purity** and **efficiency**, defined in Section 5.1, as an energy-aware variation of precision and recall.

1.2.3 AI-methods for trackster smoothing

Related work in calorimetric clustering includes specialized clustering algorithms such as the one employed in ParticleFlow (CMS Collaboration, 2009), density-based clustering algorithms, such as CLUE by Pantaleo and Rovere (2022), as well as learned parametric models such as convolutional neural networks (Niedermayer, 2017), and recently, adaptations of graph neural networks (Ju et al., 2020; Qasim et al., 2019).

In this work, we focus on two classes of methods. First, we explore classical clustering and partitioning techniques, such as K-Means, Gaussian Mixture Models, or DBSCAN, applicable to the clustering formulation of the problem. Second, we explore learned parametric models, in which the model parameters are inferred from the data and applied in reconstruction. Given the iterative reconstruction approach, in which we focus on reconnecting tracksters rather than individual energy hits, and recent advancements in the related work, we will emphasize adaptations of graph neural networks and related point cloud-based approaches.

1.3 Agenda

This work explores the feasibility of adapting AI-based methods for improving particle shower reconstruction performance at the High-Granularity Calorimeter (HGCAL) by recognizing and reconnecting tracksters from the same particle. Due to the challenges in calorimetric reconstruction, such as highly complex data, large data volumes, and real-time processing requirements, developed approaches should be evaluated in terms of physics performance, computational complexity, and scalability on heterogeneous HPC architectures. Therefore, the following aspects shall be addressed in the scope of the work:

Evaluation: Performance metrics for evaluating the candidate approaches' improvement in the reconstructed quality need to be established. These metrics should evaluate the overall event reconstruction performance independent of the selected method.

Methods: Appropriate AI methods for the clustering and classification formulation of the problem shall be investigated, leading to the identification of the best suitable models for improving the reconstruction quality. The classical clustering approaches should be used as a baseline for evaluating the performance of the learned models.

Comparison: The identified methods should be compared with the baselines in terms of physics performance using the proposed evaluation metrics. Furthermore, training and inference requirements and scalability towards Exascale for online and offline reconstruction in heterogeneous HPC environments should be examined.

Research questions

Within the scope of work defined above, the following research questions shall be investigated:

1. Are machine-learned methods viable candidates to improve particle shower reconstruction in calorimetric clustering, and do they provide an advantage over classical clustering methods?
2. Which approaches and settings of machine-learned models, in terms of task formulation, model architecture, and loss functions, are suitable for the calorimetric clustering task? Do graph neural networks provide a benefit in reconstruction performance and computational complexity over feed-forward neural networks?

These goals align with CoE RAISE's efforts to develop scalable AI methods for data-driven use cases for future Exascale HPC systems with heterogeneous architectures. Improving the particle reconstructions at the CMS HGCAL will enable the researchers to retain more data about the particle collisions, perform more accurate analysis and support the development of new physics theories beyond the standard model and discovery of new particles.

Chapter 2

Related work

The task of calorimetric clustering can be regarded as a problem of connecting sets of objects in space. Unordered sets of objects with spatial coordinates can be represented as point clouds. This chapter reviews relevant literature concerning particle shower reconstruction at HGCAL, machine-learned methods for calorimetric clustering, and the general classification and segmentation methods for point cloud data. The tasks on point cloud data addressed in related work vastly differ across the application domains. In graph terminology, nodes represent data objects (*e. g.*, tracksters), and edges describe relationships between them. Wu et al. (2021) broadly categorizes the graph problems into:

1. node-level: where per-node outputs are produced for classification or regression task (*e. g.*, given a set of tracksters, classify which tracksters belong to a photon shower),
2. edge-level: relating to link prediction – predict whether two nodes are likely to be linked together (have an edge between them), and
3. graph-level: relating to graph classification, where class labels are predicted for the entire graph (*e. g.* given a graph of tracksters describing a particle shower, predict the type of the initial particle).

Furthermore, the training setup may differ based on the learning task and available labeled data. The common setups include:

1. supervised graph classification: predicting the class label for the entire graph,
2. semi-supervised classification: where labels are predicted given a partially labeled graph (*e. g.*, predict node labels for a graph of tracksters, given a trackster of interest), and
3. unsupervised representation learning – learning to represent the graph in a latent space.

2.1 Clustering methods

Various clustering methods can support the problem of connecting nearby tracksters given a point cloud. We review classical clustering algorithms directly applicable to the problem (*i. e.*, without constructing a graph), particularly algorithms based on a partition, density, distribution, and hierarchy (Xu and Tian, 2015). An essential consideration in clustering is that the number of clusters in particle shower reconstruction is not known a priori. For this reason, we will categorize the clustering algorithms as centroid-based, for which defining the number of centroids is required; agglomerative (hierarchy-based), where a hierarchical tree containing all the points based on node-link distance is formed; and density-based, where the number of clusters is selected based on a configurable density threshold.

2.1.1 Centroid-based clustering

Centroid-based approaches like K-Means (MacQueen, 1967) are likely the most straightforward and common forms of clustering. A preset number of cluster centroids can be initialized randomly and iteratively updated based on the nodes assigned to the centers until convergence is reached. The main advantages of k-means are the simplicity and scalability to larger datasets due to favorable runtime complexity.

Gaussian-Mixture Models (GMM) (Rasmussen, 1999) extend the simple centroid-based approach by Gaussian distribution models. The goal is to identify the parameters of the Gaussian mixture by maximizing the likelihood of the distributions generating the presented data samples. Meanshift (Cheng, 1995; Comaniciu and Meer, 2002) algorithm, like k-means, is also based on centroids. However, instead of defining their number, it assumes that the points are drawn from some probability density function and selects the centroids as maxima of this function.

2.1.2 Agglomerative clustering

Agglomerative clustering is a family of algorithms based on hierarchy. The most common flavor uses Ward's criterion (Ward, 1963) as the linkage method between data points. Points are hierarchically linked into a dendrogram structure, which can be used to explore merging and splitting the clusters. However, to obtain a result, this structure has to be cut at some point, usually selected by the number of clusters, such as in centroid-based clustering.

2.1.3 Density-based clustering

DBSCAN (Ester et al., 1996), like Meanshift, also considers the local density. However, instead of placing centroids onto the maxima of that function, it transforms the space using the density function, moving the outliers further away and constructing a dendrogram of points in the transformed space. Afterward, it cuts the branches of the dendrogram based on a distance threshold and discards branches with too few samples (given a threshold) as noise. An extended version, HDBSCAN (McInnes et al., 2017), performs DBSCAN over varying distance parameter values and integrates the results to find clustering with the best stability, which removes the need to tune the distance parameter for variable datasets.

A significant difference between DBSCAN and k-means and GMM is that density-based algorithms can return only clusters instead of partitioning the point cloud. In other words, not all data points must be included in the resulting clusters, and noise can be omitted. Particular for HGCAL reconstruction, it is essential to note the work on The Iterative Clustering framework (Pantaleo and Rovere, 2022), along with the CLUE and CLUE3D clustering algorithms (Rovere et al., 2020) serving as the basis of this work, further introduced in Section 3.3.1.

2.2 Learned parametric models

Machine learning on irregular data structures is infamously problematic. Efforts in developing machine learning methods on point representations are among the few relatively successful ones dealing with such data. However, in many cases, a series of constraints apply. This work will primarily focus on the node- and link-level prediction setup in a semi-supervised setting. Operations on point clouds usually follow one of four paths: voxelization into a regular grid, direct operations on the point cloud, neighborhood-based methods building graphs or graphlets, or methods using projection into a latent representation.

The pattern recognition problem in particle physics differs from the typical machine learning tasks in computer vision. An overarching difference is the absence of semantic labels for the shower components, compared to the traditional supervised learning setup of point cloud classification and segmentation tasks. Previous studies on using graph neural networks (GNNs) for calorimetric clustering by Ju et al. (2020) and Qasim et al. (2019) demonstrate the applicability of message-passing GNN architectures to the problem. The use of GNNs in high-energy physics for particle tracking and reconstruction is well summarized in a survey by Duarte and Vlimant (2022a).

2.2.1 Direct operations on point clouds

While dealing with point clouds is essential and well-researched for computer vision, with use cases such as scene understanding for self-driving cars, the problem semantics differs from dealing with energy deposits. Learned methods on point clouds typically work with a fixed number of data points, focused on classification and segmentation. These approaches come from computer vision problems, where the data points can be sampled from the original object without losing too much information.

One of the first successful applications of neural networks on point cloud data was presented by Qi et al. (2017a) in PointNet, applied to 3D object classification and segmentation tasks. PointNet is trained on a point cloud classification task by aggregating point-wise features using pooling operators or a point cloud segmentation task by concatenating the local point features with the global feature vector. Both tasks require semantic labels describing the point cloud class or the segment assignment. In PointNet++, Qi et al. (2017b) apply PointNet recursively on a nested partitioning of the input point cloud, which allows the networks to learn local features on increasing contextual scales. First, the point cloud is partitioned into overlapping regions from which local features are extracted. Then, the local features are grouped into larger units. The local regions are selected by spherical Euclidean neighborhoods. However, applying PointNet++ to a particular problem requires carefully selecting suitable scales.

An alternative approach is presented by Thomas et al. (2019) in Kernel Point Convolution (KPConv). In KPConv, convolutional operations are applied directly on the point cloud using a convolutional kernel function defined by point values instead of a grid kernel typical for the convolutional filters applied on regular data.

Contrary to scene segmentation or object classification problems, data point sampling in particle reconstruction is not feasible as the data points carry semantic information, including timing and deposited energy. Hence, sampling would cause losing crucial information. For this reason, methods developed for scene understanding in computer vision, including representation learning methods for volumetric shapes, are unsuitable for calorimetric reconstructions.

2.2.2 Voxel grid methods

Voxel grid methods refer to a series of approaches that map point clouds into regular structures to alleviate the problems stemming from sparse, variable-size representations. This transformation enables the application of methods for manipulating 3D images or scans, which are relatively well-researched due to many use cases in the medical and engineering domains. However, similarly to sampling, voxelization leads to a loss of precision. Furthermore, the high granularity of the calorimeter causes increased sparsity, especially since the particle showers are not volumetric shapes, which can be well represented in a lower resolution.

Following the success of convolutional neural networks (CNNs) for image recognition started by Krizhevsky et al. (2012), extending the convolutions to 3D is a natural choice. However, it is not

evident that the same CNN architectures would perform well on largely sparse 3D grids. Moreover, operations on such grids can quickly become computationally intractable with increasing resolution. Nevertheless, thanks to highly optimized CNN implementations, voxelized CNNs still produce competitive results, as shown by Wang et al. (2019). The sparsity issue could be further addressed by introducing spatial embeddings, as suggested by Jaderberg et al. (2015) in spatial transformers networks.

Maturana and Scherer (2015) in Voxnet approached the problem by introducing volumetric occupancy grids, a form of voxelization into a lower-dimensional space in which the cost of 3D convolutions is manageable. Another approach is presented by Choy et al. (2019) addressing the sparsity issue in 4D spatiotemporal convolutional neural networks for processing 3D video in time by introducing differentiable sparse convolutional layers. In particle shower reconstruction, the detector hits do also contain timing information. For now, the timing is usually omitted on the trackster level. However, it could be relevant for improving the recognition of individual particle showers.

2.2.3 Neighborhood-based methods

Deep learning methods for graph data, mainly based on convolution, were subjected to rigorous research in the last decade, yielding a series of complementary approaches. Wu et al. (2021) in their survey on graph neural networks (GNNs), categorize the proposed methods into recurrent GNNs, convolutional GNNs (Kipf and Welling, 2017), graph auto-encoders (Kipf and Welling, 2016), and spatial-temporal GNNs. The spatial applications of convolutional GNNs construct a graph of the point cloud, capturing the spatial dependencies between the points.

In PointCNN, Li et al. (2018b) introduce an X-transformation-based convolution that gradually transforms inputs into fewer representation points with richer features. The transformations are based on k nearest neighboring points. When trained for classification, the points are aggregated in each step, reducing the number of points. On the other hand, the number of points is preserved for segmentation tasks. Later research in spatial convolutional graph neural networks draw inspiration from the message-passing mechanism formalized by Gilmer et al. (2017), previously applied in recurrent GNNs.

Wang et al. (2019) in Dynamic Graph CNN (DGCNN) for learning on point clouds further develop the idea of using k-nearest neighbors in convolutions. In DGCNN, the authors present an operator called EdgeConv based on the message-passing mechanism representing a convolutional layer for graph structures. For each point, EdgeConv computes edge features for the point's neighbors and aggregates them. The details of EdgeConv are further introduced in Section 3.4.1.

Fueled by the success of attention-based models in natural language processing triggered by Vaswani et al. (2017), researchers also investigated graph-based implementations. For example, Veličković et al. (2018) in graph attention networks introduce a mechanism that attends to the first-order neighborhood of each node using k-independent attention heads. While the neighborhood for graphs is defined by hops, it could also be formed by k-nearest neighbors in spatial applications. In point transformers, Zhao et al. (2021) apply the self-attention mechanism to point clouds using k-neighborhood around each data point. Point transformers include a trainable position encoding step to better encode the spatial representation, where a multi-layer perceptron is applied to encode the respective position of neighboring point pairs.

2.2.4 Graph neural networks in particle physics

Tasks commonly solved on graphs are semantically different from those on point clouds. Graphs, such as social networks used for link prediction (Liben-Nowell and Kleinberg, 2003), contain semantic links representing relationships between nodes. While k-nearest neighbors could be

used to establish links between the shower fragments, their order and cardinality are arbitrary. Furthermore, methods applied to graph-related tasks often involve analyzing large static graphs rather than many small graphs. Therefore, methods successfully used to learn on static graphs might not transfer to point clouds.

Researchers in related literature have applied GNNs to the particle identification task using hits, tracks, or cluster patterns. Relevant examples include Machine Learned Particle Flow by Pata et al. (2021) and Point Transformers by Mikuni and Canelli (2021). In JEDI-net, Moreno et al. (2020) use interaction networks (Battaglia et al., 2016) to describe the relations between constituents of the graph. Then, a sparse representation of the event is handled directly, without particular preprocessing or geometric assumptions. As a result, JEDI-net obtains competitive results on related tasks while having relatively few parameters.

Qu and Gouskos (2020) refined the DGCNN concept for manipulating point cloud data in particle physics in their work on ParticleNet by adjusting the pooling methods and introducing skip connections between layers. In parallel, Qasim et al. (2019) introduced GravNet to mitigate the DGCNN problem of running k-nearest neighbors in high-dimensional spaces by separating the coordinate and feature space. Building on the developments in GravNet, Kieseler (2020) introduced the concept of object condensation – a loss function-based approach to collapse specific clusters to representative points.

Recent advances in applying transformer architecture to jet tagging were published by Qu et al. (2022) in Particle Transformers, reporting state-of-the-art performance. While particle transformers show a state-of-the performance in jet tagging and outperform ParticleNet by a small margin on all evaluated dataset sizes, the power of transformer architecture only becomes apparent trained on millions of training samples. Particle Transformers models also use five times more parameters than ParticleNet (2.14M and 370k, respectively). On the other hand, they require 40% fewer floating point operations in inference than the ParticleNet architecture.

2.2.5 Representation learning

Learning structural representations of the point cloud is the foundation of link prediction. Zhang et al. (2021) identify two main directions of the research in GNN-based link prediction methods: graph auto-encoders (Kipf and Welling, 2016), and SEAL (Zhang and Chen, 2018). Graph auto-encoders first apply a GNN to the entire graph and compute the representation for each node. Then, they compute link representation by aggregating features of the nodes connected by the link. On the other hand, the SEAL approach applies GNN to enclosing sub-graphs around each link, where the enclosing sub-graph of a node pair is induced by the nodes' neighbors up to a certain number of hops.

Another relevant research direction in learning 3D pattern recognition is unsupervised representation learning. Representing the tracksters in a latent space is helpful for feature extraction. Encoding tracksters in a regular low-dimensional space allows using less complex models. In 3D ShapeNets, Wu et al. (2015) suggest representing 3D shapes by a probability distribution of binary variables on a 3D voxel grid and a convolutional deep belief network to learn the distribution. The use of auto-encoder networks for learning representations on point clouds was explored by Achlioptas et al. (2018), where 1D convolutional layers with kernel size one are used, after which column-wise maximum pooling is employed to create the basis for the latent space. In the related work on graphs and point clouds, latent representations are usually a by-product of learning the full-scale model. When it comes to feature extraction in general, generic point cloud features are partially an under-investigated area of research. As Qi et al. (2017a) argue, most of the point cloud features in related literature are handcrafted for specific tasks, commonly being statistical properties of points, intrinsic or extrinsic, on the local (point) or the global (point cloud) level. Generally, it is not trivial to find an optimal feature combination for a specific task.

2.3 Task evaluation

Calorimetric clustering can be phrased as a segmentation problem connecting all fragments to the source particle or a classification problem by focusing on a single particle and identifying all fragments that belong to it. Standard evaluation metrics such as precision, recall, and F-score can be applied in the classification setting. In the segmentation setting, evaluation becomes more complex, and the problem needs to be treated as a clustering task. Various similarity metrics for comparing clusterings were proposed in the related literature, including the variation of information (Meilă, 2007), normalized mutual information (Xu et al., 2003), and B^3 precision and recall (Amigó et al., 2009). Metrics for evaluating unordered point sets commonly include Chamfer distance and Earth Mover's distance (Rubner et al., 2000). In representation learning, Achlioptas et al. (2018) also introduce tailored metrics for point cloud fidelity and diversity. In calorimetric reconstructions, The Iterative Clustering (TICL) framework (CMS Collaboration, 2022) uses a score to evaluate the reconstruction performance introduced in Section 4.2.1. On a similar task of directly assigning energy hits to particle showers, Qasim et al. (2019) introduce a loss function to evaluate the fractional assignment of energy deposits to the detector sensors for each particle shower and a clustering energy response metric for two overlapping showers.

Chapter 3

Background

Improving the particle shower reconstruction quality at the High-Granularity Calorimeter poses a unique problem in a rather under-investigated area of machine learning on the boundaries of point cloud segmentation without semantic labels, pattern recognition, and clustering. This chapter describes the detector's characteristics, the state of the current reconstruction pipeline, the nature of the problem, requirements, and feasibility of the methods from related work.

3.1 Large Hadron Collider

The Large Hadron Collider (LHC) at CERN is arguably one of the most complex scientific instruments ever built. CERN is home to some of the largest scientific collaborations in the world, running multiple experiments around the collider. The 27-kilometer-long LHC machine accelerates particles to nearly the speed of light. Protons circulate in clockwise and anticlockwise directions and collide at four points along the ring. At these points, see Figure 3.1, the energy of the proton collisions is transformed into mass and emits new particles in all directions. These experiments aim to capture and identify the produced particles, detect anomalies, and support scientists in understanding the underlying principles of particle physics. Captured observations can then be analyzed and evaluated against the projections of the Standard Model, the current description of interactions between the basic building blocks of matter. Understanding these fundamental interactions is essential to develop and refine theories going beyond the standard model, such as supersymmetry, string theory, or extra dimensions, that aim to correct for the phenomena not explained by the current formulation of the Standard Model, including gravity, dark matter, neutrino masses, or matter-antimatter asymmetry (Pomarol, 2012). Conditions created in the LHC are close to the ones in the very early stages of the universe. Therefore, capturing and understanding their nature in detail lets scientists better understand the nature of the universe.

3.1.1 Event reconstruction and classification at HL-LHC

The LHC operates in alternating cycles of runs when collisions and data collection occur and shutdowns when the machine undergoes maintenance and upgrades for the next run. The upgrades during a shutdown include detector improvements and increasing the energy of the collisions. Before the next run, the LHC will undergo a significant upgrade to extend its discovery potential. In this upgrade, known as the High-Luminosity Large Hadron Collider (HL-LHC), the luminosity (number of collisions per square centimeter, per second) will be increased by a factor of 5, and the total integrated luminosity (over one year) by a factor of 10 (Brüning and Rossi, 2015).

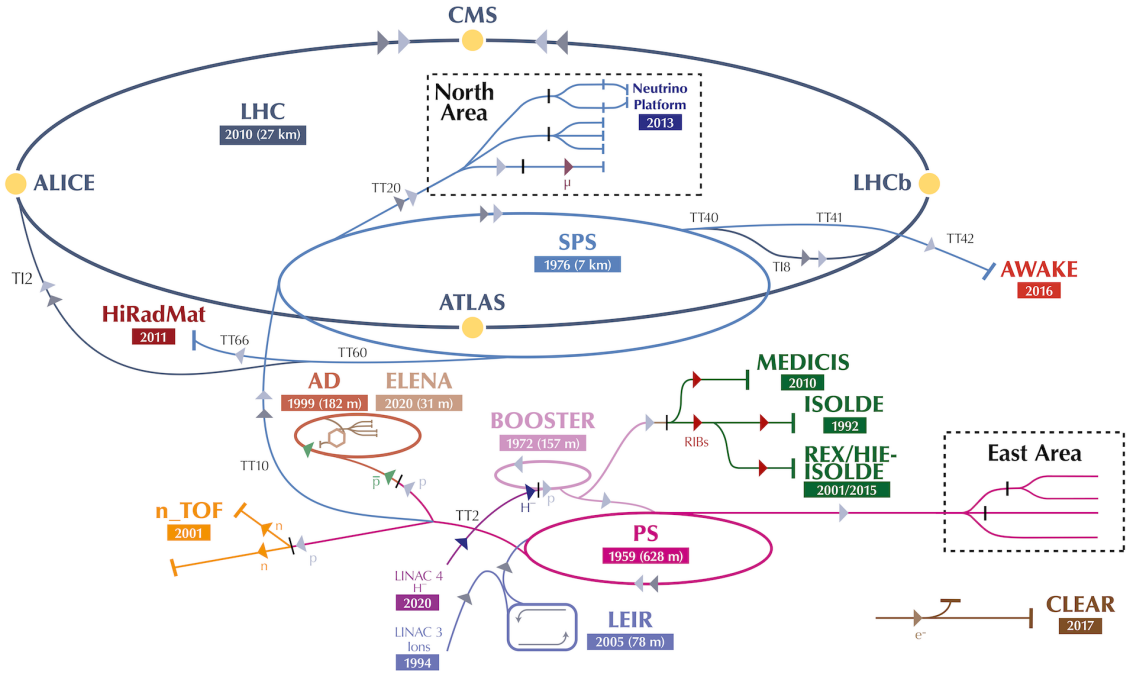


Figure 3.1: THE CERN ACCELERATOR COMPLEX. *Hydrogen atoms stripped of electrons are accelerated to the near speed of light by a series of linear (LINAC) and circular (BOOSTER, PS, SPS) accelerators before reaching the Large Hadron Collider (LHC). On top of the schematic is the 27-kilometer-long ring of LHC with the four major experiments: ATLAS, ALICE, CMS, and LHCb (Lopienska, 2022).*

This increase in collisions poses significant data volume and complexity challenges to the current reconstruction pipelines, mainly due to high pile-up (simultaneous interactions) in detectors, especially for calorimetric reconstructions (Martelli, 2017; CMS Collaboration, 2017).

Event reconstruction and classification at the CERN's LHC is one of the data-driven use cases in the CoE RAISE efforts towards exascale scalability[1]. The use-case explores the applicability of AI methods in high-energy physics, specifically particle reconstructions in the phase of high-luminosity LHC. The main focus of the use case is developing AI-based methods trained on the simulation data, which can scale to the data rates required by the HL-LHC phase. In detail, the efforts include manipulating large data sets, exploring methods feasible for processing point cloud data, distributed hyper-parameter optimization, and portability of the developed solutions to run on heterogeneous architectures.

3.1.2 Compact Muon Solenoid

Compact Muon Solenoid (CMS) detector is one of the four major experiments at the LHC. It is a large general-purpose detector with a broad physics program to observe any new physics phenomena present in the collisions and search for extra dimensions or particles that could make up dark matter. CMS acts like a large high-speed 3D camera taking snapshots of collisions up to 40 million times a second. Most of the particles created in the collisions are unstable and quickly transform into other stable particles. The detector features a powerful solenoid magnet that bends electrically charged particles to help to identify and measure particle momenta. The captured

[1]Event reconstruction and classification at HL-LHC: https://www.coe-raise.eu/event-reconstruction

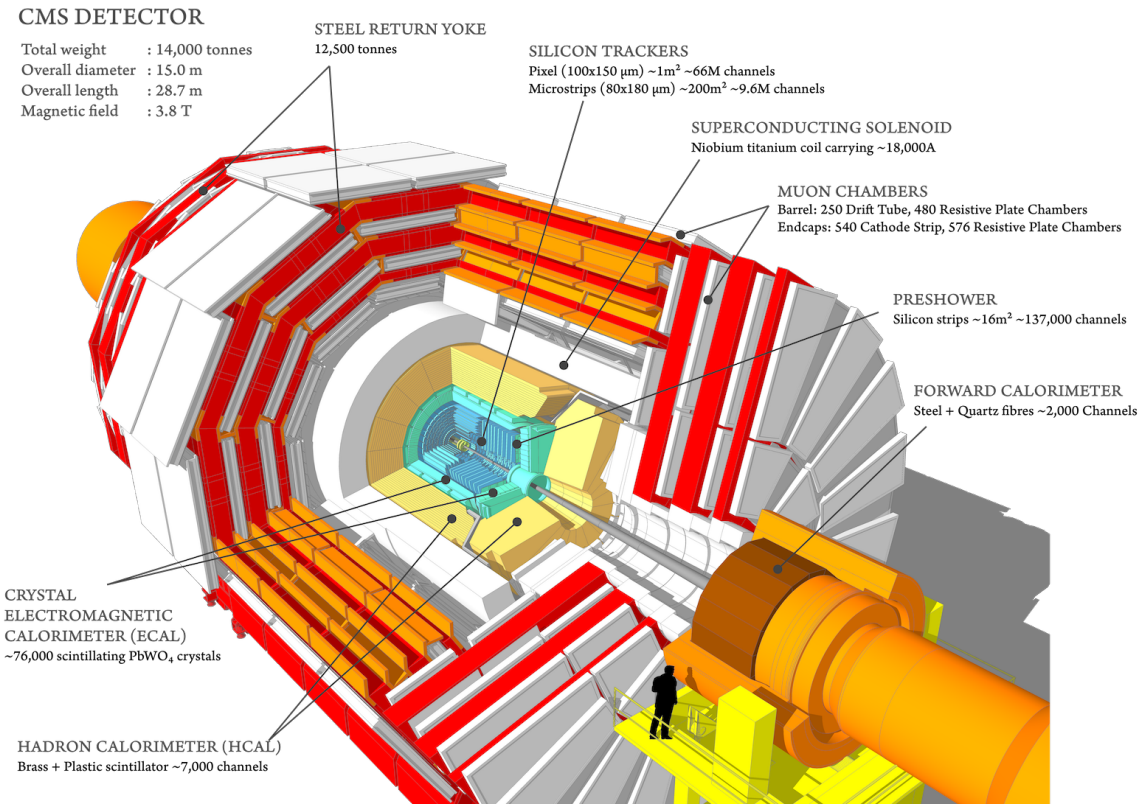


Figure 3.2: THE CMS DETECTOR. *Cutaway view of the CMS detector and its subsystems by Sakuma and McCauley (2014). The Tracker part of the detector forms a cylinder around the collision point, while the crystal electromagnetic calorimeter (ECAL) and hadron calorimeter (HCAL) are visible on the sides (endcaps). The newly developed HGCAL detector will replace ECAL and HCAL modules in both endcaps.*

information from all parts of the detector is then pieced together in analysis to reconstruct the entire interaction, including the transformations of the unstable particles.

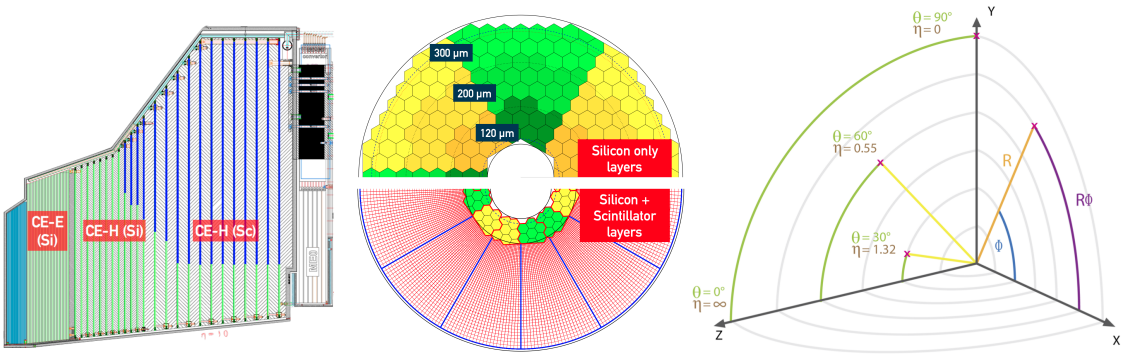
The CMS detector has two major elements to capture collisions: Tracker and Calorimeters. Tracker is used to precisely capture the trajectory of bent particles, while the purpose of calorimeters is to measure the electric charge of the particles. Currently, two kinds of calorimeters are used: Electromagnetic Calorimeter (ECAL) and Hadron Calorimeter (HCAL), see Figure 3.2.

Coordinate system

The detector coordinate system (Figure 3.3(c)) has the horizontal x-axis pointing inside of the accelerator ring, vertical y-axis pointing upwards, and the z-axis pointing west along the direction of the beam (see the position of the CMS detector on the beamline in Figure 3.1). Additionally, a cylindrical coordinate system defined in (η, ϕ) space is used. The azimuthal angle ϕ is measured from the x-axis in the (x, y) plane. A polar angle θ is defined in the (R, z) plane, where R is the radial coordinate in the (x, y) plane. Ten, the pseudo-rapidity η is derived from the polar angle θ as:

$$\eta = -\ln\left(\tan\frac{\theta}{2}\right) \tag{3.1}$$

with values in $(-\infty, \infty)$, where $\theta = 0\,\mathrm{deg}$ coincides with the beamline axis (Pantaleo, 2017).



(a) Front view (b) Side view (c) Coordinate system

Figure 3.3: GEOMETRY OF A HIGH-GRANULARITY CALORIMETER ENDCAP. *Transverse view (a) shows the disposition of the 50 detector layers along the z-axis in a part of the detector above the beam line. Colors distinguish the silicon sensors in the electromagnetic (CE-E) section and the bottom part of the hadronic (CE-H, Si) section and scintillator tiles in the upper part of the hadronic (CE-H, Sci) section. Layout view (b) shows the silicon-only (top) and mixed silicon-scintillator (bottom) layers. The representations in the figure's bottom and top halves are not up to scale due to the radius difference in CE-E (Si) and CE-H (Sci) sections (Lobanov, 2019). Figure (c) shows the detector coordinate system, with the x-axis pointing inside the accelerator ring, the y-axis pointing upwards, and the z-axis pointing west along the direction of the beam (Pantaleo, 2017).*

3.1.3 High-Granularity Calorimeter

As part of the HL-LHC upgrade program, the CMS collaboration is developing a new High-Granularity Calorimeter to replace the existing endcap calorimeters (ECAL and HCAL on the ends of the detector, see Figure 3.2). The High-Granularity Calorimeter will be a part of the CMS and provide precise 3D-positions and timing of the energy deposits, which enable discrimination of clusters from pile-up within a single bunch crossing (Ochando, 2017). Particles emitted from the collisions will interact with the HGCAL sensors and produce deposits of energy (hits). Construction of the HGCAL is still ongoing in preparation for the future High-Luminosity run; however, precise simulations allow the creation of large datasets of particle interactions with the calorimeter on the hit level and investigate reconstruction methods for future detector deployment.

Calorimeter geometry

The High-Granularity Calorimeter consists of two parts: an electromagnetic section (CE-E) of 28 layers with silicon sensors with higher radiation tolerance and a hadronic section (CE-H) with eight layers of silicon sensors and 14 mixed silicon-scintillator layers in the less radiation-intensive area. Reconstruction of particle flows in HGCAL in high pile-up environments is challenging due to overlapping particle showers. Electromagnetic particles will primarily interact with the layers in the electromagnetic section (CE-E). In contrast, hadronic particles will interact with the layers in the hadronic section (CE-H). The $600\,\mathrm{m}^2$ of silicon sensors contain 6 million channels (individual data sources), while the 400 plastic scintillator tiles contain another 240 thousand channels. The sensors are organized in hexagonal structures in the silicon layers to maximize the detection granularity. The detector geometry setup is displayed in Figure 3.3.

A crucial aspect to consider in reconstruction is the irregularity of the detector. Disposition of

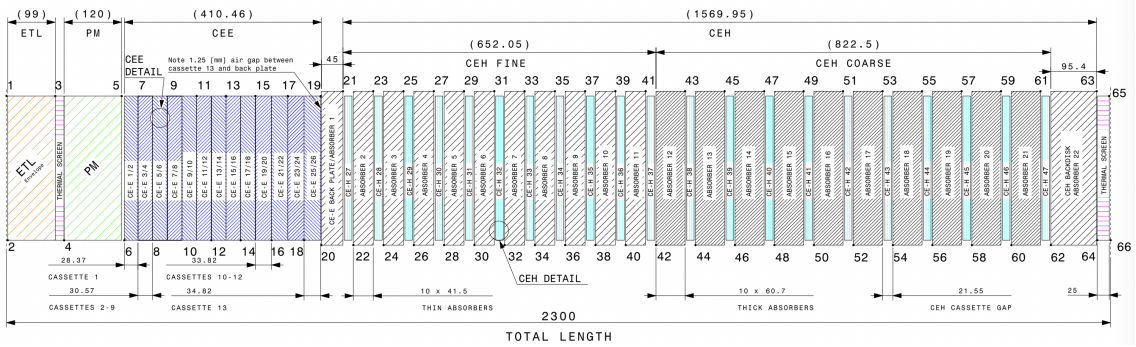


Figure 3.4: DETECTOR LAYERS. *Layer widths in the first part of the detector – the electromagnetic section of the calorimeter are* 28.37 mm *for the first cassette,* 30.57 mm *for cassettes 2 to 9,* 33.82 mm *for cassettes 10 to 12, and* 34.82 mm *for the cassette 13. Each cassette holds two* 4 mm *silicon modules separated by a* 6.2 mm *thick cooling plate. Following a* 45 mm *lead absorber plate and* 1.25 mm *air gap, ten fine hadronic layers follow with a width of* 21.55 mm *separated by* 41.5 mm *thick absorber plates. The layers 36-47 in the coarse CE-H section of the detector are separated by thicker absorber plates of* 60.7 mm *(Rapacz, 2017).*

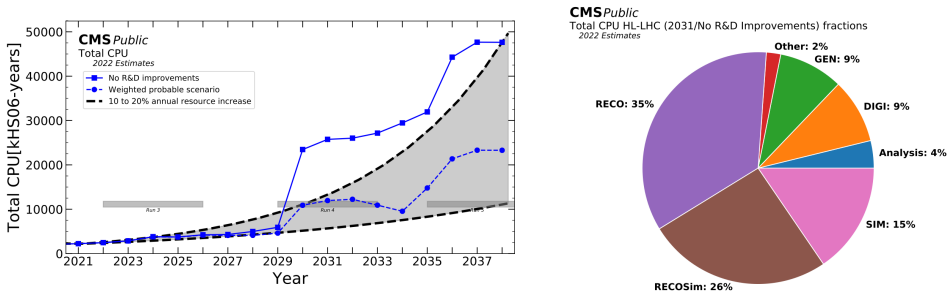
the sensors within a layer varies, as well as the distances between individual layers (as shown in Figure 3.4). This irregular disposition means traditional distance metrics such as Euclidean distance and derived clustering methods will not produce consistent results throughout the detector. For this reason, custom distance metrics considering the detector layers should be examined. Alternatively, parameterized machine learning models should be able to natively learn to work with irregular distances as long as the training labels were assigned correctly using the adapted metrics.

3.2 Particle showers

When particles formed after the collision interact with the material of the detector, they produce a cascade of secondary particle showers with progressively decreasing energy. The purpose of the calorimeter detector is to capture this progressive decay in energy and the cascade of particle interactions. As explained by Fabjan and Gianotti (2003), depending on the particle type, two interaction mechanisms are observed in calorimeters: electromagnetic processes and strong (nuclear) processes. Particles such as electrons, positrons, and photons interact primarily (or exclusively) via electromagnetic force. After interacting with the detector material, particles of sufficiently high energy (typically above 1 GeV) produce secondary photons via a process called bremsstrahlung, or secondary electrons and positrons via pair production process. This cascade continues in the subsequent layers of the calorimeter until the components of the shower reach a critical energy at which no new particles are produced. This process creates the typical ellipsoid-like shower shape. However, some particles, especially hadrons, can produce multiple disconnected ellipsoid-like showers while interacting with the detector.

Particle interactions

Heavier, hadronic particles such as protons and pions (and other particles made up of quarks) similarly produce a hadronic shower cascade. Hadronic particles primarily interact with the detector material through many strong (nuclear) processes. Secondary particles produced in



(a) CMS future CPU needs projection

(b) Breakdown of projected CPU usage

Figure 3.5: PROJECTION OF CMS CPU NEEDS FOR HL-LHC. *With a 10-20% annual increase in computing capacity, significant improvements in the computational complexity of the algorithms are needed to meet the HL-LHC data rate. (a) shows the projection of the future CPU needs for the next LHC runs. (b) shows the breakdown of the future projected CPU needs for HL-LHC. The reconstruction (RECO, in purple) is expected to make up 35% of the CPU time, and simulation of the reconstruction (RECOSim, in brown) up to 26% (CMS Offline Software and Computing, 2022).*

hadronic showers include new hadrons such as neutrons, pions, or protons and charged particles such as photons, electrons, and positrons. Therefore, hadronic showers can be captured in both electromagnetic and hadronic parts of the detector, although the interactions are primarily observed in the hadronic part due to heavier absorber plates.

3.2.1 Data acquisition

The experiment data is processed in several steps consisting of several systems. While the readout systems of the detector run at $40\,\mathrm{MHz}$, producing data at the rate of up to $80\,\mathrm{TBps}$, this throughput will be reduced to $750\,\mathrm{kHz}$ by a highly optimized system called Level 1 Trigger in the detector cavern. A series of parallelized algorithms running on FPGA boards process the data in real-time, selecting interesting events for physics analysis and forwarding them out of the detector cavern. On the surface, the following computer system, called High-Level Trigger (HLT), further reduces the output to $7.5\,\mathrm{kHz}$ and forwards a small fraction of the initially collected events (up to $61\,\mathrm{GBps}$) to permanent storage and offline processing facilities (CMS Collaboration, 2021). The HLT is a high-performance computing facility with standard processors and NVIDIA T4 GPUs for running the asynchronous algorithms reconstructing and selecting interesting events for storage and offline processing. This work aims to support the particle shower reconstruction process using AI methods, which will be able to process the events on HLT rates in real time using commercially available GPUs.

Computational challenges

The High-Granularity Calorimeter in a high-luminosity setting poses significant readout data processing challenges. An order of 10^5 hits is captured every 25 nanoseconds, 40 million times per second, in snapshots called events. Each event is produced by a crossing of two bunches (groups) of particles. Due to the high luminosity, up to 200 simultaneous interactions per bunch crossing are expected. The high number of simultaneous interactions, called pile-up (PU), become a

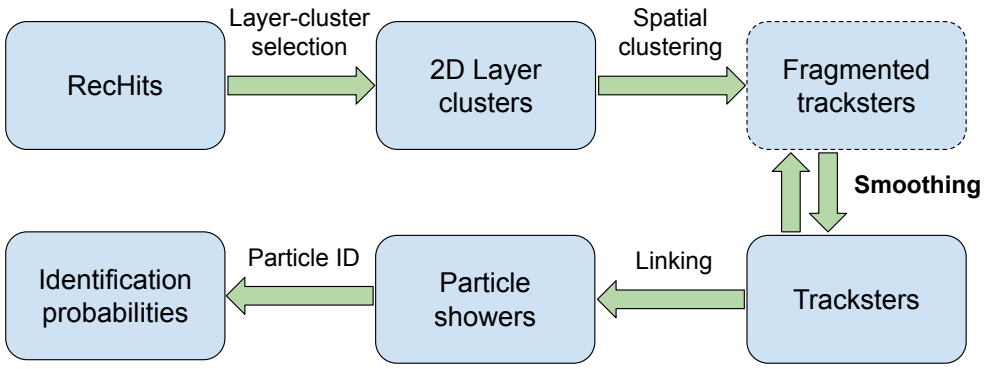


Figure 3.6: TICL PIPELINE. *First, layer-cluster selection is applied to the reconstructed hits from the detector to form 2D clusters in the individual calorimeter layers. Secondly, the pattern recognition algorithm connects the 2D layer clusters between the layers into candidate tracksters. These are then refined into tracksters and linked to particle showers, from which individual particle probabilities and properties are identified.*

tremendous challenge for the reconstruction of individual particle showers (Ochando, 2017). Current particle tracking algorithms scale worse than quadratically with the number of hits (Duarte and Vlimant, 2022b) and LHC experiments rely on intensive research and development to comb the growing computational requirements, as shown in Figure 3.5.

3.3 Reconstruction pipeline

Particle shower reconstruction aims to determine the properties of the initial high-energy particle that produced the shower. The task of calorimetric reconstructions of electromagnetic and hadronic showers is to connect all energy deposits originating from the same initial particle. The energy deposits are recorded as hits on the detector layers as the particle traverses through the detector. These hits are then iteratively clustered into particle showers, which are then used to determine the properties of the initial high-energy particles. Finally, events containing numerous particle showers are analyzed to understand the entirety of the particle collision.

3.3.1 The Iterative Clustering Algorithm

Each event is expected to produce $O(10^5)$ hits. Reconstructed hits are described by the spatial coordinates of the cell (x, y, z) relative to the center of the detector, energy, and timing information. Reconstructed hits are iteratively clustered in the multiple stages of The Iterative Clustering (TICL) introduced by Pantaleo and Rovere (2022) to reduce the reconstruction complexity. The entire TICL pipeline is shown in Figure 3.6. The clustering process can be summarized in three steps (Pilato et al., 2020):

1. Particles deposit energy in the layers of the calorimeter, creating hits.
2. Hits are clustered on each layer, forming layer-clusters. This process is rather conservative to avoid combining hits from other particle showers.
3. Layer-clusters are connected, forming tracksters (collections of layer-clusters), in a process referred to as pattern recognition.

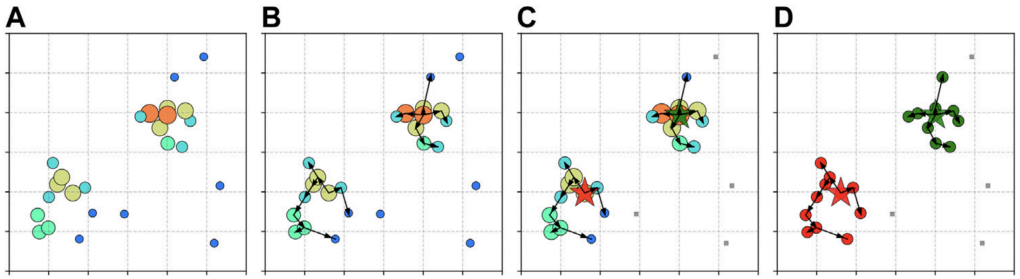


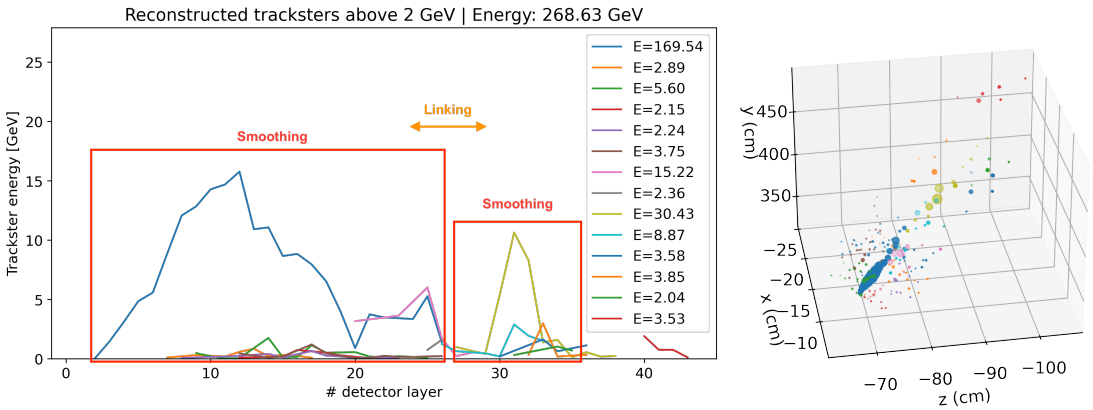
Figure 3.7: CLUE ALGORITHM. *In this example, the points are distributed in a 6x6 fixed grid. First (A), each point's spatial index and local density are computed. Then (B), CLUE finds the nearest neighbor with higher energy within the point's neighborhood. Next (C), CLUE promotes a point as a seed if the local density and the separation are large or as an outlier if the local density is low and the separation is large. Finally (D), CLUE expands the clusters from seeds using the follower lists. The remaining gray squares are outliers without followers and are discarded as noise (Rovere et al., 2020).*

3.3.2 Clustering in 2D

In the first step of TICL, 2D-clusters on each detector layer, called layer-clusters, are formed using the energy-density-based clustering algorithm CLUE (Rovere et al., 2020) optimized for high-occupancy scenarios where the number of clusters is larger than the average number of hits in a cluster. TICL utilizes a spatial index with a fixed grid to avoid expensive sequential nearest neighbor scans over all points. The algorithm computes the local density for all the points in the grid and nominates the highest-density points as cluster seeds. All the other points are then assigned to follower lists of their nearest neighbor with higher energy. At the same time, outliers are rejected based on the separation and density parameters. Finally, the follower lists are used to build clusters; see Figure 3.7. As all this has to be done in real-time during the detector run-time, it is paramount for the clustering to be very fast, efficient, and parallelizable. The selected algorithm design allows for efficient n-way parallelization for computing the local density, distances, seeds, and outliers, while the cluster expansion can be implemented in a k-way parallelization. The decision to run CLUE layer-wise allows running the algorithm on all detector layers in parallel. In each event, the layer-clustering algorithm reduces the hit multiplicity by one order of magnitude to $O(10^4)$ layer-clusters.

3.3.3 Pattern recognition

The process of connecting layer-clusters belonging to the same particle is referred to as pattern recognition – recognizing individual particle showers in a large set of layer-clusters produced by many, often overlapping showers. In previous versions of TICL, this process was implemented using cellular automatons (Pantaleo and Rovere, 2022). This work divides pattern recognition into several parts: clustering the layer-clusters in 3D based on their spatial properties, connecting nearby tracksters into locally complete connected showers, and then connecting the shower fragments over larger distances. The pattern recognition step pipeline can be run in multiple iterations and parallelized using multiple seeding regions. A seeding region is defined by an (η, ϕ) plane around a selected layer-cluster. The pattern recognition algorithm can then be applied to available layer-clusters in the region, producing tracksters. The already processed layer-clusters can then be removed in the following iterations of the algorithm (Pilato et al., 2020).



(a) Longitudinal energy profile (b) 3D view

Figure 3.8: RECONSTRUCTION OF A PION. *Display of a pion shower reconstruction in the TICL pipeline. (a) shows the longitudinal energetic shower profile of the pion constructed by summing up the layer-clusters energy in each detector layer. Tracksters are first smoothed within the connected regions of the energy. Then, linking is applied to connect the smoothed tracksters along the z-axis. (b) shows a corresponding 3D visualization of the pion shower.*

Clustering in 3D Reconstructed layer-clusters are connected into spatial structures across the detector layers, called tracksters, using a spatial implementation of the CLUE algorithm called CLUE3D (Pantaleo and Rovere, 2022). Tracksters represent fragments of a particle shower. The overall reconstruction goal is to represent an entire particle shower by a single trackster (as represented by a simtrackster in the ground truth data). The output of this step is in $O(10^3)$ tracksters. However, many simultaneous collisions in the detector impose a tight bound on the spatial clustering sensitivity to avoid creating one large bulk of many particle showers. This constraint leads to particle showers being split into multiple tracksters. At this point, the energy deposits are merged into spatial structures (tracksters), yet the particle showers are fragmented. Clustering solely on the spatial position and energy can no longer be applied without reducing the purity of the shower – merging with tracksters representing different simtracksters.

Trackster smoothing The process of locally connecting fragments of the particle shower based on higher-level properties, such as geometric shape or direction, is called trackster smoothing and is the goal of this thesis. Smoothing aims to enhance the reconstruction quality by reconnecting nearby tracksters formed by the same initial particle. Smoothing is expected to work locally, producing connected shower components (with no gaps on the z-axis). It is not expected to handle gaps along the z-axis or different branches of the same particle shower. The output should be well-reconstructed, elongated ellipsoid-like objects with a smooth longitudinal energy profile, representing a part of the shower connected along the z-axis, see Figure 3.8. Disconnects along the z-axis and different branches of the same particle shower are handled in a process called **linking**. Well-performing trackster smoothing should improve the linking performance and alleviate linking complexity, as the local shower components are better represented.

Linking The purpose of linking is connecting tracksters produced by the same initial particle, which might include connecting parts of the shower over larger distances. For example, as shown in Figure 3.8, a pion may produce a shower in the front part of the calorimeter (electromagnetic

section), cease to interact around the 26th layer, and then produce a second shower in the hadronic section. Other scenarios include branching when a secondary particle produced in an interaction continues with a diverging trajectory. This fragmentation is not addressed in the trackster smoothing part, which is only concerned with reconstructing single, continuous segments of the particle showers. However, trackster smoothing and linking are complementary and work towards the same goal. This work considers handling disconnects on the z-axis as the boundary between the two. Nevertheless, other approaches might be viable and explored in future work, including single-step smoothing and linking.

While the linking algorithms are still under development, and AI-based approaches are also being explored in this part. The current TICL version includes a rather conservative geometric linking implementation based on the trackster direction given (η, ϕ) and timing (Nandi, 2022).

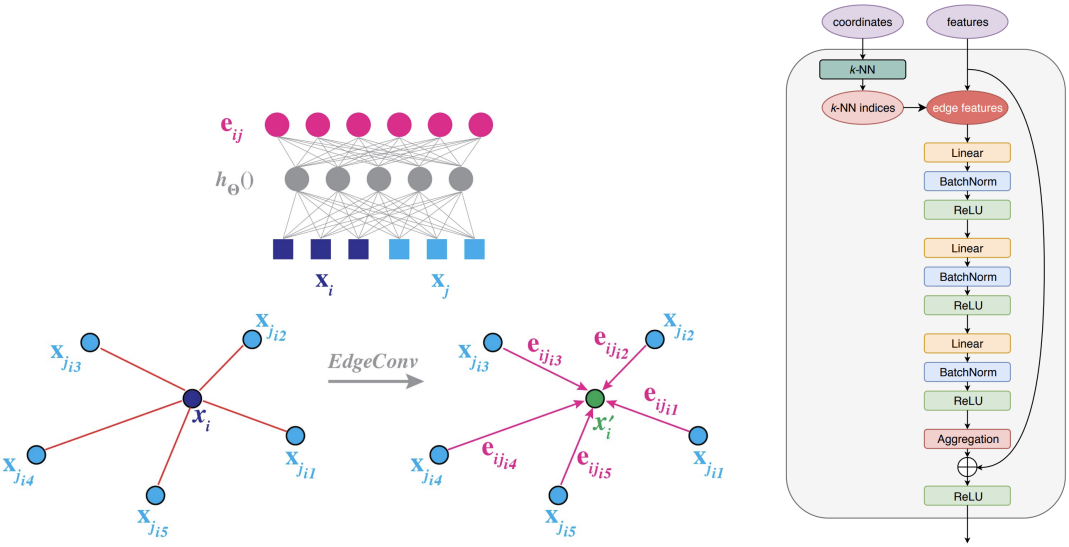
3.3.4 Particle identification

Particle identification closes the loop of the reconstruction pipeline. The final step is determining the properties of the initial particle that caused the shower. Particle identification returns identification probabilities across particle types for each trackster. This process uses convolutional neural networks for particle identification using a fixed-size trackster representation. In the study by Pilato et al. (2020), tracksters were represented as images of dimensions (50, 10, 3), where the features along the last dimensions are (energy, η, ϕ) layer-clusters in each layer of the detector. The number of layer-clusters was limited to 10 (second dimension), and the remaining pixels were zero-padded. After training the model on a dataset of 40 thousand events and four particle types (10 thousand events per particle type), the model learned to distinguish between muons, charged hadrons, electrons, and photons, with some remaining confusion between the last two.

The effect of pile-up In high-occupancy settings, with up to 200 simultaneous particle interactions per event, implementing single-shot approaches to analyze all collisions simultaneously is infeasible due to many overlapping particle showers. For this reason, the iterative approach of TICL focusing on particular seeding regions can recover the showers trackster by trackster. Once the tracksters are separated from the background noise by pattern recognition, linking algorithms, and particle identification can be applied on pure tracksters within the seeding region and compatible timing. Trackster smoothing and linking could be merged in a no-pileup setting, as their task is semantically similar, the only difference being the distance between the candidate tracksters and the amount of background noise. However, joint approaches of full reconstruction directly from layer-clusters using larger, more complex models are still being explored.

3.4 Learned methods for trackster smoothing

In this section, AI methods applicable to the task of trackster smoothing are analyzed, and the involved techniques are introduced in greater detail. In particular, methods that utilize the fragments' spatial properties to reconnect the initial particle's energy deposits are considered. The outcome of applying these methods should be reconnecting the incomplete tracksters generated by the initial particle into well-formed tracksters, which better describe the underlying physics phenomena and reduce the computational complexity driven by the high fragmentation.



(a) EdgeConv operation (b) EdgeConv block

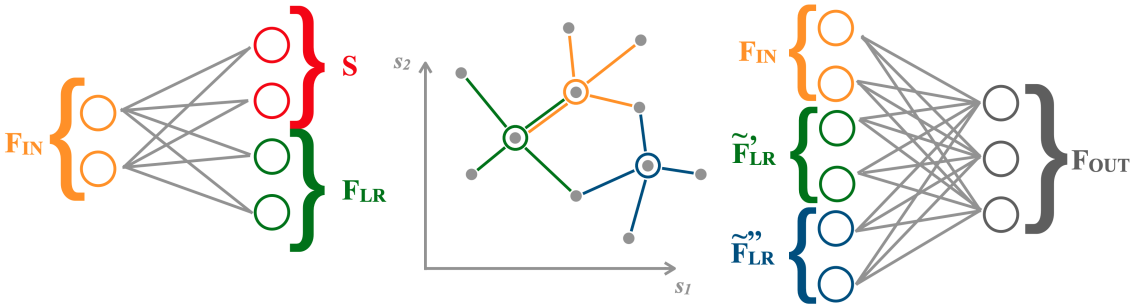
Figure 3.9: EDGE CONVOLUTION. *(a) Performing the EdgeConv operation on the neighborhood of x_i. First, the edge values e_{ij} for x_i and its neighbors x_j are computed. Then, values e_{ij} are aggregated into x'_i. (b) Graphical representation of the EdgeConv block used in ParticleNet, picturing the neighborhood computation on the top, learnable non-linear transformation function in the middle, and the aggregation step on the bottom (Wang et al., 2019; Qu and Gouskos, 2020).*

3.4.1 Graph convolutional networks

Models based on Dynamic Graph Convolutional Networks (DGCNN), such as ParticleNet (Qu and Gouskos, 2020), provide a competitive performance on related problems in particle physics. Moreover, their relative simplicity and great flexibility make them a good starting point for designing new methods for pattern recognition on point clouds, as demonstrated by Ju et al. (2020) and Qasim et al. (2019). This section describes the inner working of DGCNN and the EdgeConv (Wang et al., 2019) operator in greater detail. Similar to the problem of dealing with text sequences of variable length in natural language processing, models working on 3D point sets need to support inputs of varying sizes. In contrast to sequential text processing, the point sets are inherently unordered, and presenting permutations of the same input set should lead to reasonably close results. Such as, in transformer models for sequence processing, aggregation is the solution for having variable context size. DGCNN and ParticleNet combine the information from the points' neighborhoods using sort- or average-pooling mechanisms.

Edge convolution

The idea of interaction between graph nodes in a convolutional fashion was introduced in various forms in convolutional, gated, or interaction network models. Based on their similarity, Gilmer et al. (2017) generalized the idea into the Message Passing Neural Networks (MPNN) framework. The message passing on time steps t, given message functions M_t, vertex update functions U_t, and edge features e_{ij} updates the hidden states h_i^t of each node i in graph based on messages



(a) Features are projected into coordinate space S and feature space F_{LR}

(b) EdgeConv is applied to the k-nearest neighbors in the latent space S

(c) Learned features are concatenated into the output representation F_{OUT}

Figure 3.10: THE PROCEDURE OF THE GRAVNET LAYER. *First, the input features are projected into coordinate space S and feature space F_{IN} by a dense layer (a). Then, k-nearest neighbors are identified, and the EdgeConv operation is performed (b), creating output for the next GravNet block. Lastly, features from all GravNet blocks are concatenated into the final output vector F_{OUT} (c). (Qasim et al., 2019)*

m_i^{t+1} given by:

$$m_i^{t+1} = \sum_{j \in N(i)} M_t(h_i^t, h_j^t, e_{ij}) \tag{3.2}$$

$$h_i^{t+1} = U_t(h^t, m_i^{t+1}) \tag{3.3}$$

where $N(i)$ denotes the neighbors of node i in graph. Inspired by PointNet and the message-passing-based convolution operators, Wang et al. (2019) introduce an Edge Convolution operation on point clouds (EdgeConv, see Figure 3.9(a)), where the neighborhood is defined by a k-nearest neighbors local sub-graph with edges in $\mathcal{E}$ around i, including a self-loop. The $\sum$ operation in (3.2) is generalized to an aggregation operation $\square$. The EdgeConv update function for each hidden state h_i^t is then defined by:

$$h_i^{t+1} = \square_{j:ij \in \mathcal{E}} h_\Theta(h_i^t, h_j^t) \tag{3.4}$$

where $\square$ is the aggregation function (*e.g.* $\sum$ or max), and h_Θ: $\mathbb{R}^{F_t} \times \mathbb{R}^{F_t} \to \mathbb{R}^{F_{t+1}}$ is the edge-function – a learnable non-linear projection from the feature space F_t into the feature space F_{t+1}. In terms of the MPNN framework, h_Θ is equivalent to a message function M_t without considering the edge features, and the vertex update function U_t directly returns the new state as the previous state is already considered via the self-loop included in the nodes' edge list.

The EdgeConv operators are contained in blocks. Each block contains the edge feature concatenation based on the edge list, followed by three blocks of dense layers with batch normalization (Ioffe and Szegedy, 2015) and rectified linear unit function, see Figure 3.9(b). A pooling layer then aggregates the output of the linear layers for each edge. Qu and Gouskos (2020) in ParticleNet further suggest adding a skip-connection – a concatenation of the EdgeConv block output with the original nodes' features to propagate the original node features deeper into the network.

Point neighborhoods

Multiple approaches for defining the point neighborhood exist. The standard approach used in DGCNN and ParticleNet is defining the neighborhood of each point using a k-nearest neighbors search. The k-nearest neighbors search is performed in each EdgeConv block using the latent

representation of the data points. In the first layer, ParticleNet constructs the neighborhoods based on the polar coordinates, while DGCNN employs a spatial transformation block. However, in the following layers, the k-nearest neighbors algorithm in high-dimensional spaces suffers from the curse of dimensionality. Therefore, using large feature vectors becomes infeasible due to the growing complexity of computing the distance and the deteriorating quality of the Euclidean distance with growing dimensionality.

An alternative approach would be to pre-build the k-nearest neighbors graph based on the input features, such as the trackster barycenters, and keep it fixed throughout the reconstruction. This approach is much less computationally demanding as it constrains the neighborhood search to fewer (ideally spatial) dimensions and only runs the search once. However, it does not inherit certain DGCNN design benefits, such as connecting to nodes similar in the learned latent representation, even though they might be far away in the original spatial space.

Qasim et al. (2019) in their work on GravNet suggested a combination of these approaches, where the learned spatial and feature representations are kept separate, as described in Figure 3.10. This combined approach delivers good computational complexity and clusters in meaningfully low-dimensional spaces. However, it still utilizes a learned spatial representation, which enables the network to connect nodes, which might have initially been too far to be connected by a traditional k-nearest neighbors search. Two separate dense layers transform the input features of each node, the output of which are the spatial and feature vectors. The spatial vectors are then used to establish the k-nearest neighbors, followed by an EdgeConv operation producing the input to the next layer.

Based on the DGCNN architecture, GravNet is also organized in blocks. The GravNet model uses four blocks, each containing three dense layers with 64 nodes, a hyperbolic tangent activation function, and a GravNet layer with $S = 4$ coordinate dimensions and $F_{LR} = 22$ features to propagate. Following the blocks, the output of all four is concatenated into the final output vector; see Figure 3.10(c).

Input and output dimensionality

In the EdgeConv setup, loading the point cloud and applying the transformations on points individually while aggregating their neighborhood information retains the input dimensionality on the output. In a graph classification setting, as introduced in Chapter 2, the EdgeConv part can be considered as a learnable deep feature embedding, on top of which a feature-wise pooling layer could be applied to produce a fixed-size representation for classification tasks. The dimensionality transformation then follows:

$$F : X^{(N,F)} \rightarrow Y^{(F')} \tag{3.5}$$

where N is the number of nodes, F the number of input features per node, and F' the number of output features (classes). In node classification, the feature vector per node is used directly, preserving the dimensionality of the input, *i. e.*:

$$F : X^{(N,F)} \rightarrow Y^{(N,F')} \tag{3.6}$$

The link prediction task predicting relationships between nodes produces the output in E, the number of edges in the graph, *i. e.*:

$$F : X^{(N,F)} \rightarrow Y^{(E,F')} \tag{3.7}$$

where an output of shape F' (number of classes) is produced for each edge $e \in E$. The set of output edges can be defined by the input graph (as an edge index) or produced dynamically using k-nearest neighbors.

Advanced mini-batching While the first dimension of the input of the neural network can differ in size – defining the dimensionality of the output, the inner dimensions of the input matrix must be preserved. Preserving the inner dimensionality in the number of nodes becomes problematic once multiple graphs are processed within the same batch. The solution is to apply an input transformation in which the points in the point clouds of different samples are concatenated together:

$$F : X^{(B,N,F)} = X^{(BN,F)} \rightarrow Y^{(BN,F')} = Y^{(B,N,F')} \tag{3.8}$$

where B is the batch size. An additional batch index is created to prevent the individual point clouds from interacting with each other. The batch index has to be respected in all aggregation operations, such as EdgeConv, or pooling layers within the model. The same logic applies if the graph has additional fields or an edge index: the fields are re-mapped to the new indices in the concatenated point set, such that point operations aggregating neighborhood information do not overlap with the other data samples.

3.5 Problem summary

To summarize, this work aims to develop a scalable AI-based algorithm for locally reconnecting energy deposits coming from the same particle in the High-Granularity Calorimeter. In the previous steps of the reconstruction pipelines, the individual calorimeter hits are clustered into small spatial objects (tracksters), with the goal of high inter-tracksters purity. The goal of this work is trackster smoothing, utilizing the spatial, shape, and neighborhood information to recognize the particle showers patterns and reconnect the smaller tracksters to larger ones representing connected segments of the particle shower. The results of trackster smoothing are then further reconnected along the particle shower axis and used to determine the properties of the initial particle.

Chapter 4

Data

Research in particle physics and designing new detectors requires access to fully detailed simulations of particle interactions. Access to accurate simulations is paramount for modeling new detectors, developing new reconstruction algorithms, and validating the physics models against the measurements. The CMS experiment software provides advanced simulation tools for interactions within the detector based on the Geant4 (Agostinelli et al., 2003) toolkit using Monte-Carlo methods. CMS software[1] is a vast software framework enabling a modular implementation of the particular detector sub-systems and extensive configuration options for running the simulation and reconstruction. CMSSW workflows enable the generation of precise particle showers interactions within the detector. The design and construction of HGCAL for HL-LHC are ongoing; therefore, reconstruction algorithms must be prepared on particle showers simulated using the actual HGCAL geometry. This chapter describes available data and methods for evaluating the reconstructed showers against the ground truth.

4.1 Simulating particle showers

Geant4 supports generating calorimetric particle showers using a generator called `Particle Gun`. Its HGCAL extension `CloseByParticleGun` can be configured to shoot particles into the detector with HGCAL geometry from the desired location along the detector beamline. Parameters of the generator, such as particle type (*e. g.*, a photon, an electron, or a pion), number of particles, energy range, distance from the beamline, and angle with the beamline (η), can be configured. While it is impossible to simulate unknown or not well-described rare particles, particles that are the subject of research at CERN experiments are typically very unstable and quickly decay into other elementary particles that can be simulated. Therefore, we focus on reconstructing known elementary particles, such as photons and pions, which can be a part of other, more complex interactions. Dependencies and overall interactions in the detector are then examined in separate in-depth physics analyses. For example, Figure 4.1 shows a simulation of an electron shower passing through HGCAL layers.

4.1.1 Pileup

Under the HL-LHC conditions, 140 to 200 simultaneous interactions are expected to occur in the detector in each bunch-crossing. This phenomenon is referred to as pile-up (PU). In the expected production setting, this will lead to thousands of tracksters of all kinds of particles being produced with a high degree of overlap.

[1] CMSSW documentation: `http://cms-sw.github.io`

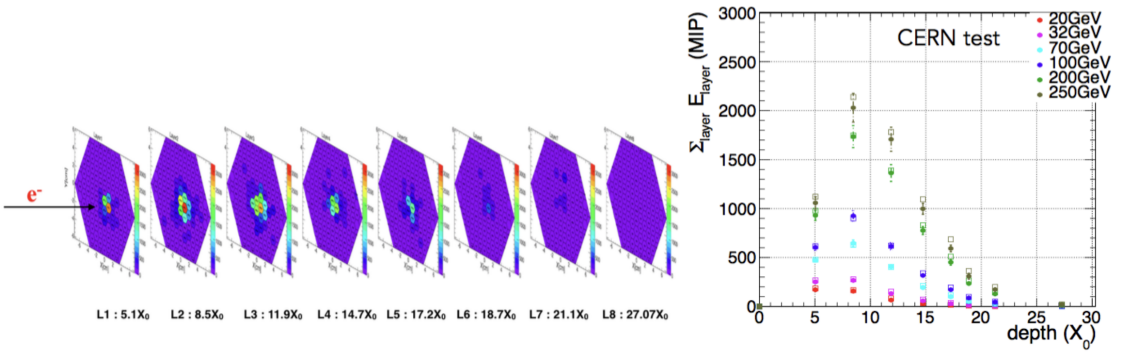


Figure 4.1: HGCAL ELECTRON SHOWER PROFILE. *Event display of an electron passing through 8 HGCAL layers (left) and longitudinal shower profile measured in data and compared to simulation as a function of the shower depth (right) (Martelli, 2017).*

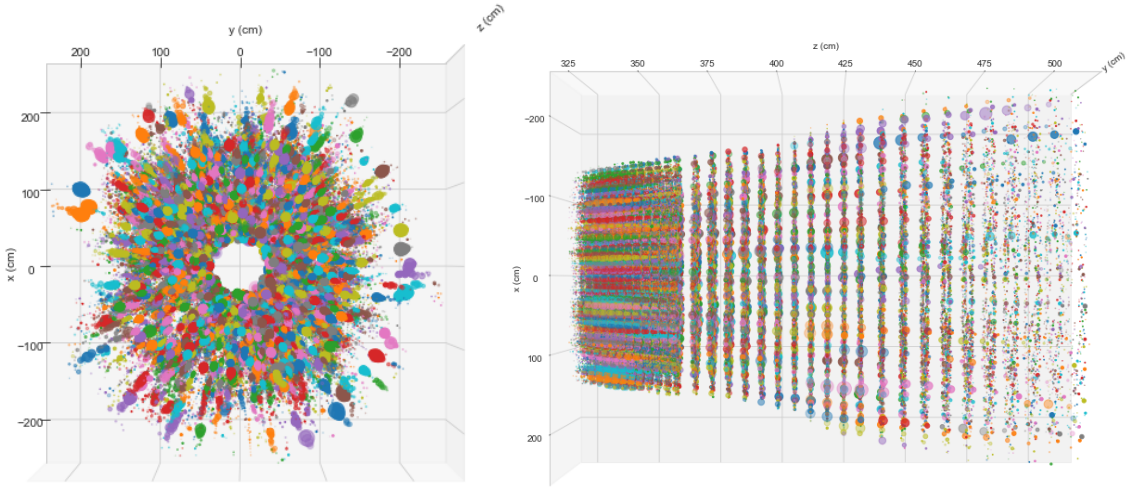
4.1.2 Dataset types

To first understand the properties of the particle showers, we generate datasets of two close-by particles, such as photons and pions, with limited overlap. This dataset enables us to study shower fragmentation properties and establish particle profiles. Later, we proceed to more involved datasets with overlapping particles and apply the same reconstruction process tuned to the higher-overlap scenario.

Two close-by particles In the case of two-close by particles, simulated photons or pions are shot into the same region of the detector, 15 cm apart. We use this dataset to study the energetic profiles of the particles and the splitting behavior of the clustering algorithms. Furthermore, it provides a baseline case for ensuring the energy fragments are assigned to the correct particle shower in reconstruction. Particle showers in this dataset are well separated as most of the tracksters within a high-energy trackster neighborhood (defined in Section 5.3.1) belong to the same simtrackster.

Multi-particle In a more complex case (see Figure 4.2, we generate a dataset of multiple overlapping particles of different kinds. In each event, we focus on a set of 10-50 particles shot into the same detector region. Here, more particles introduce increasing overlap, and the neighborhood of high-energy tracksters contains more tracksters produced by other particle showers.

Datasets with pile-up Finally, we include pile-up to simulate the properties expected in the production. Pile-up represents the energy deposits generated by unknown simultaneous interactions for which the ground truth definition is unavailable. Here, a single simulated particle is shot into the calorimeter occupied by thousands of tracksters produced by 200 simultaneous particle collisions (pile-up of 200) and then focus on recovering the energy of this particle from among the background. The particle showers in pile-up datasets overlap significantly, leading to most tracksters in the high-energy neighborhood of high-energy tracksters belonging to other particle showers.



(a) Front view

(b) Side view

Figure 4.2: MULTI-PARTICLE DATASET. *Front view (a) and side view (b) of layer-clusters in 50 multi-particle events provide an outline of the detector geometry. The size of each layer-cluster is proportional to its energy. The side view (b) reveals the irregular geometry of the detector. The dense electromagnetic part of the detector makes up only one-quarter of the z-axis variance. The remaining layers in the hadronic part contain thicker absorber plates placed further away from each other.*

4.1.3 Dataset size

An advantage of using simulated data is that more data can be generated. Typically, we work with dataset sizes from 5 to 20 thousand events. In the two-pion case, each event contains 28.1 tracksters on average. For a multi-particle event with ten simtracksters, the average number of tracksters is 96.8. Generally, $O(10^6)$ tracksters are available in datasets without pile-up. Depending on the approach (how many trackster connections are considered), this might produce a much greater number of labels. Using the available computer systems, datasets of this size can be generated in several hours.

In the pile-up case, the share of data relevant for reconstruction in each produced sample is relatively low. Only a single simtrackster is generated within a considerable pile-up (typically 3 to 4 thousand tracksters). Here, the datasets that can be generated within one or two days are on the lower end of the dataset-size spectrum with 5 to 10 thousand events. On average, a single photon shower in a pile-up environment has 2.1 trackster fragments in reconstruction produced by this shower. For pions, the average number of relevant reconstructed tracksters for a single shower is 15.2. However, the negative connections (tracksters produced by other particles) also need to be evaluated in the reconstruction. Due to the high pile-up, the relative number of negative connections is relatively high.

4.2 Dataset structure

CMSSW is using ROOT Framework[2] for manipulating the collision data. Generated data, as well as outputs of each of the succeeding reconstruction steps, are stored in root files. Each root file contains a tree-like data structure, with branches for particular reconstruction steps, including

[2]ROOT framework: https://root.cern

the CLUE output – layer-cluster collection, CLUE3D output – trackster collection, linking output – merged tracksters collection and evaluation score from the associators.

Layer-clusters The layer-clusters produced by CLUE are stored in a collection shared and referred to by both simtracksters and reconstruction tracksters. In this sense, tracksters and simtrackster data objects do not directly contain the layer-clusters but refer to them by a collection index. This commonality is crucial for evaluating the trackster reconstruction against the simtracksters. Each layer-cluster is described by:

1. number of hits,
2. energy (sum of the hits' energy),
3. barycenter coordinates computed as an energy-weighted average of the hit coordinates, expressed in (x, y, z) coordinate space, and cylindrical (η, ϕ) space,
4. radius, as the energy-weighted mean distance from the barycenter,
5. layer ID (given by the z coordinate, used to identify successive layers),
6. local energy density (computed by CLUE using a Gaussian convolution kernel).

Tracksters are collections of layer-clusters. The layer-clusters belonging to each trackster are identified by an array of indices to the CLUE layer-cluster collection. The same CLUE layer-cluster collection is used for reconstruction tracksters (coming from CLUE3D), merged tracksters (from geometric linking or smoothing), and simtracksters. Tracksters are described by:

1. list of layer-clusters (by index to the global collection),
2. barycenter coordinates – computed as the energy-weighted average of layer-cluster coordinates, expressed in (x, y, z), and in cylindrical (η, ϕ), space, here, the z coordinate might fall between the detector layers,
3. total energy (sum of layer-cluster energies),
4. total electromagnetic energy (sum of layer-cluster energies in the electromagnetic section),
5. the first principal component eigenvector, first three eigenvalues, and the first three component-wise reconstruction errors from principal component analysis (PCA) applied to the barycenters of layer-clusters in the trackster, and
6. one-hot vector of particle identification probabilities (photon, electron, muon, neutral pion, charged hadron, neutral hadron, ambiguous, unknown), produced by a convolutional neural network applied to the trackster (see Section 3.3.4).

Simulation tracksters (simtracksters) represent complete particle showers of single particles and are used as the ground truth definition. The simulated data is available on the hit level. Reconstructed layer-clusters containing the simulated hits are associated with the share of energy from the individual simulated particles. As some layer-clusters may contain energy from different particles, each layer-cluster is in a simtrackster assigned a fraction expressing the share of layer-cluster energy coming from the given simtrackster. In evaluation, this energy fraction is necessary to correctly compute the shared energy with tracksters (based on the shared layer-clusters). In reconstruction, this fraction is always one as each layer-cluster is only assigned up to one trackster.

4.2.1 Associators

Interactions with the detector are simulated on the hit level. Thus, ground truth for the particle shower reconstruction is created by associating all the deposited hits with the simulated particle. The regular TICL pipeline can be applied, as-if the hits were coming from the data-taking; however, the pattern recognition part (trackster smoothing included) does not directly operate on hits. Therefore, scores are assigned to the intermediate products of the pipeline (layer-cluster, tracksters), as described by CMS Collaboration (2022). These scores are essential for finding the simtracksters contained in the reconstructed tracksters and, conversely, finding tracksters that represent particular simtracksters.

As the simulated data are only available on the hit level, a fraction of energy shared with each simtrackster is first computed for every layer-cluster in the event (4.1). Then, given the layer-cluster-simtrackster fractions, pairwise scores for tracksters and simtracksters are computed by looking at the energy of layer-clusters shared by the simtracksters and the trackster, divided by the total energy of the simtrackster (4.2).

After the first step of the pipeline – creating layer-clusters using the CLUE algorithm, each layer-cluster i is assigned a fraction fr_i^t with respect to simtrackster t:

$$fr_i^t = \frac{\sum_{hit \in i}(fr_{hit}^t \cdot E(hit)}{E(i)} \tag{4.1}$$

where the sum of the layer-cluster hit energies $E(hit)$ multiplied by the fraction of the hit fr_{hit}^t deposited by simtrackster t is divided by the total energy of the layer-cluster E_i. Then, for a simtrackster s, and Trackster t the simToReco associator score is defined as:

$$score_{s,t} = \frac{\sum_{i \in s} min[(fr_i^s)^2, (fr_i^s - fr_i^t)^2] \cdot E(i)^2}{\sum_{i \in s}((fr_i^s)^2 \cdot E(i)^2)} \tag{4.2}$$

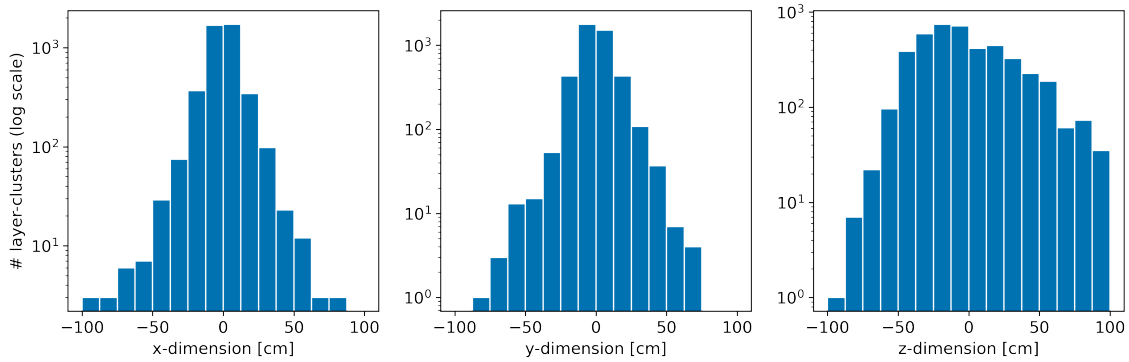
The simToReco score between s and t coveys how well is the simulated object s described by the reconstructed object t. The score for the reconstructed object t and the simulated object s recoToSim is identical by simply swapping the meaning of reconstructed and simulated objects s and t. The computed scores assign 0 to a perfect match and 1 to objects with no energy in common. In this work, we aim to assess the performance of the 3D pattern recognition algorithm unaffected by the 2D clustering performance. Therefore, the scores are considered on the layer-cluster level rather than the hit level.

4.3 Data characteristics

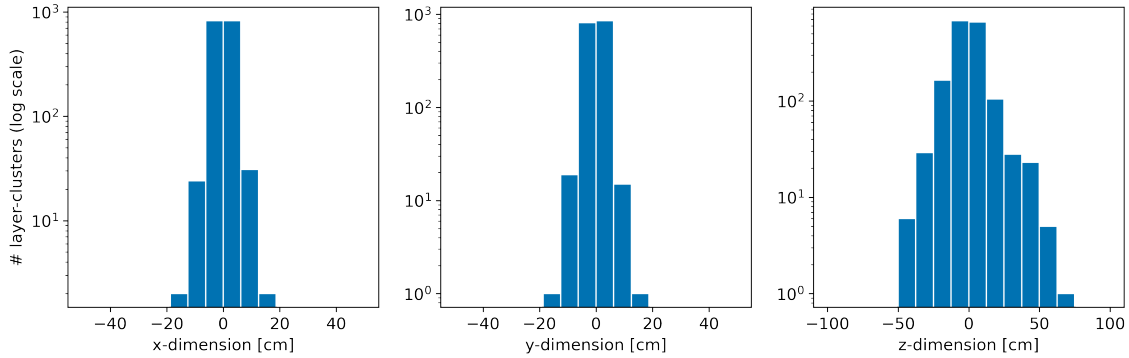
This section documents the exploratory data analysis performed on the datasets, drawing insights about the nature of the data and properties of the problem. First, we examine the spatial, energetic, and fragmentation properties of photons and pions. Capturing the fragmentation behavior in isolation helps us to understand where to search for the missing parts of the particle shower in a high-occupancy scenario. Figures in this section describe the pion case as pions are more complicated to reconstruct than photons and, therefore, more interesting to analyze; the analysis of the photon case can be found in Appendix A.1.

4.3.1 Spatial profile

Particle showers typically produce ellipsoid-like shapes elongated in the direction of the shower. Photons are characterized by interactions in the electromagnetic section of the calorimeter, while



(a) Spatial distribution in simulated pions



(b) Spatial distribution in reconstructed pion tracksters

Figure 4.3: LAYER-CLUSTER DISTRIBUTION IN PION SHOWERS. *Layer-cluster deviations from the barycenter of 100 simulated pions along x, y, and z axis (a), and the deviations of the corresponding reconstructed tracksters (b). Due to fragmentation, the reconstructed tracksters are much smaller in x and y dimensions and shorter in z.*

pions can interact in both sections, mainly the hadronic one, which leads to longer shower successions. Figure 4.3 shows the typical spatial distribution of layer-clusters in a pion shower and the distribution of the corresponding CLUE3D reconstruction output (smoothing input). The distribution along the x and y axes is symmetric, with most layer-clusters along the shower axis. The distribution is more dispersed along the z-axis, with the peak in the forward region. Pion layer-clusters are more dispersed from the barycenter than photons and show a heavier tail due to interactions in the hadronic part of the calorimeter. As a result, photon showers are typically shorter, as the interactions occur only in the detector's forward part.

4.3.2 Energetic profile

While the spatial profiles describe the general shape of the shower, they do not consider the layer-cluster energy. The reconstructed energy is the most crucial aspect for evaluating the reconstruction quality, as the core – high-energy layer-clusters of the shower are much more important than low-energetic noise. Therefore, contributions of the core layer-clusters are essential. In general, the outputs of the CLUE3D step tend to split higher-energetic shower into smaller tracksters due to the conservative clustering settings, see Figure 4.4.

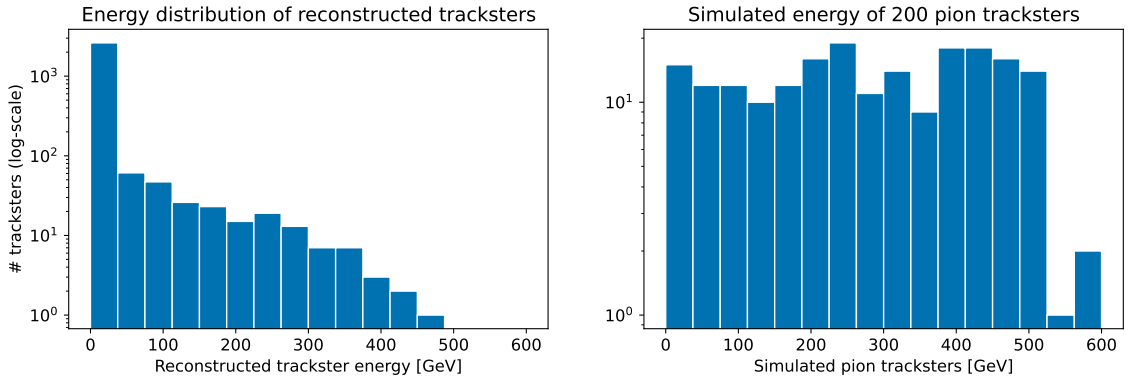
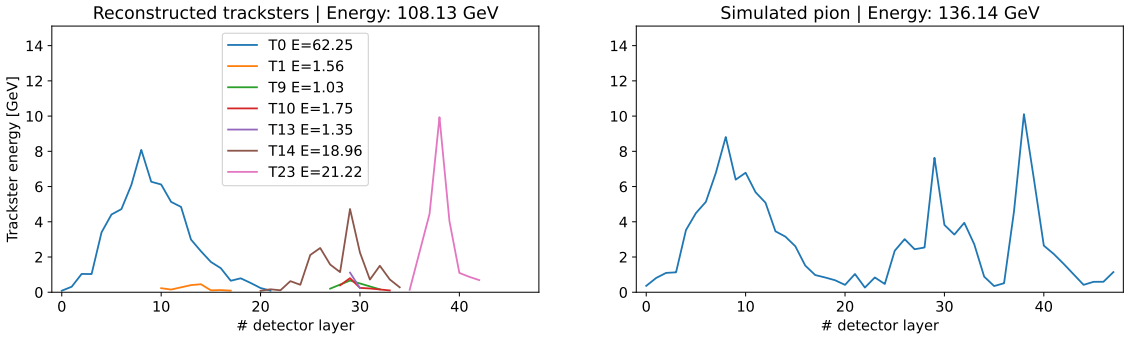


Figure 4.4: ENERGY DISTRIBUTION IN RECONSTRUCTED PIONS. *Distribution of energy in reconstructed tracksters given simulated pions sampled in the energy range from* 0 *to* $600\,\mathrm{GeV}$. *High-energy showers are not represented in the reconstructed tracksters, while most fragments are in the sub-*$100\,\mathrm{GeV}$ *range.*

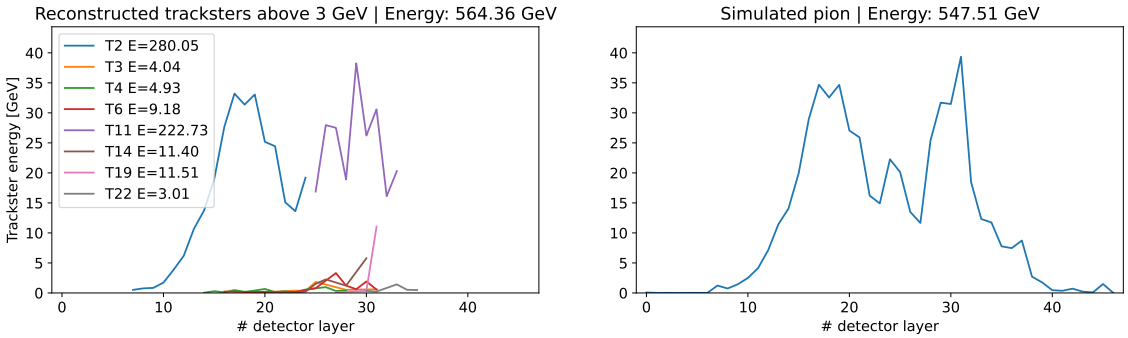
4.3.3 Fragmentation characteristics

The fragmentation is better visible on the pion reconstruction, especially along the z-axis, due to the longer inter-layer distances in the hadronic section. Reconstructions of lower-energy photons generally yield better performance, often reconstructing the entire shower. Higher-energy showers of both photons and pions tend to exhibit fragments further away from the barycenter along the x and y axes. We explore this by looking at the longitudinal energy profiles computed by summing up the energies of the shower deposited in each detector layer. Each reconstructed trackster is assigned to the simulated particle by looking at the largest fraction of shared energy – selecting the simulated particle that makes up most of the reconstructed trackster energy. As shown in Figure 4.5, a particle shower at an individual detector layer is usually represented by a single higher-energy trackster, sometimes accompanied by a series of low-energy fragments, typically below $10\,\mathrm{GeV}$.

Note that the trackster smoothing aims to reconnect the fragments with an overlap on the z-axis, primarily low-energy to high-energy fragments. The longer-distance dependencies are addressed further in the process – in the linking step, as described in Section 3.3.3.



(a) Lower-energy pion shower



(b) Higher-energy pion shower

Figure 4.5: LONGITUDINAL PION SHOWER ENERGY PROFILES. *Profile of a lower-energy pion shower (a), with several small fragments with an overlap on the z-axis – typically further away from the shower axis and three large fragments along the z-axis with gaps on the borders of the electromagnetic and hadronic sections. The higher energy shower (b) produces a larger amount (in total 17, only ones about 3 GeV displayed) of smaller fragments with an overlap on the z-axis (right).*

Chapter 5

Approach

We explore classical clustering and machine-learned techniques for improving the reconstruction performance at the HGCAL by connecting clusters of hits originating from the same initial particle following the three-dimensional clustering step. The goal is to improve the reconstruction quality by locally reconnecting fragments of the showers with an overlap on the z-axis. As calorimetric shower reconstructions under the HL-LHC conditions are computationally demanding due to highly complex data and large data volumes, we evaluate the solutions in terms of physics performance, computational complexity, and scalability. Furthermore, due to the requirements for deployment on the high-level trigger system, we emphasize support for deployment on GPUs.

As described in the Section 1.2.2, the task at hand can be posed as pattern-recognition-guided clustering or a classification problem. The preferred formulation of the problem depends on the method and the selected dataset. For example, in datasets without pile-up, clustering formulation might be appropriate (multi-particle case), while classification formulation is preferred in pile-up datasets. This chapter is organized as follows:

1. **Evaluation** – we define an evaluation strategy to assess the reconstruction performance of the candidate solutions. The evaluation metrics provide a score for the overall reconstruction process, independent of the selected approach and the problem formulation. The generality of the metrics enables comparison between the approaches on particular datasets. However, due to the different problem formulations, a direct comparison across datasets with and without pile-up might not be possible.

2. **Baseline approaches** – we explore the feasibility of addressing the problem using classical methods to establish a baseline for comparison. These methods include clustering techniques and naive engineered solutions to the problem.

3. **Machine-learned methods** – we investigate neural network architectures based on state-of-the-art approaches in related work adapted to our problem formulation.

The methods will be evaluated in the three problem scenarios as introduced in Section 4.1.2: two close-by particles case, the multi-particle case, and the pile-up case. The two close-by particles scenario is a base case that has to be well covered by any of the approaches and enables us to study the fragmentation properties and tune the algorithms. The multi-particle case challenges the algorithm's scalability to simultaneously process multiple particles of different types. Finally, the pile-up case simulates the application in a production-like high-overlap environment.

5.1 Evaluation strategy

This section defines a general evaluation metric measuring the overall performance improvement to enable comparison across different problem formulations. The associator method utilized in the dataset preparation provides a score for mapping reconstructed and simulated tracksters. The associator evaluates how the energy of simulated tracksters is contained in the reconstructed tracksters. At the same time, it conversely evaluates the energy of which reconstructed tracksters make up the simtracksters. Ideally, a single reconstructed trackster would map precisely to a single simulated trackster and vice versa. However, in the actual setting, the simulated trackster is often split into several reconstructed tracksters. Contrarily, the reconstructed trackster might also combine energy from multiple simulated tracksters. The associator scores provide detailed information about the mapping between individual tracksters and enable us to explore the reconstruction improvements in individual events. The CMS Software provides metrics to evaluate reconstruction performance across events by defining the following metrics (CMS Collaboration, 2022):

- **Efficiency** – Number of simtracksters that are reconstructed by one trackster containing at least half of their energy, divided by the total number of simtracksters. Intuitively, this metric is related to recall. It measures how many simtracksters have a trackster representing at least half of their energy, given the total number of simtracksters. However, this is a many-to-one mapping, so if a single trackster represents more than half of the energy of two different simtracksters, both simtracksters have a trackster satisfying the condition, the efficiency is 1.

- **Purity** – Number of simulated tracksters associated to at least one reconstructed trackster with a simToReco score (see Section 4.2.1) lower than 0.2 divided by the overall number of simtracksters. Intuitively, this metric is related to precision, measuring the share of simtracksters that have tracksters carrying energy primarily from the given simtrackster.

- **Merge rate** – Ratio between the tracksters associated with more than one simtrackster divided by the overall number of reconstructed tracksters.

However, recomputing the associator scores after re-connecting the reconstructed tracksters requires access to the hit-level energy information, which is not contained in the generated dataset. Therefore, re-generating the scores would require a new run of the data generation pipeline with the pattern recognition algorithm integrated into CMSSW. This setup is not very flexible and too tedious for experimentation purposes. Therefore, to enable comparison between different approaches without implementing them in CMSSW, we propose an evaluation metric for overall event reconstruction performance, including all the tracksters in an event, by adapting the B^3 precision and recall metrics by Bagga and Baldwin (1998). We base our implementation on the work by Li et al. (2018a).

5.1.1 B-CUBED precision and recall

B-CUBED (B^3) precision and recall are based on computing the fraction of layer-cluster pairs assigned to the same trackster in one clustering, which are also assigned to a single trackster in the other clustering. The B^3 precision (P) evaluates whether pairs of layer-clusters reconstructed into the same trackster originate from the single particle. Conversely, B^3 recall (R) considers whether pairs of layer-clusters originating from the same particle in the simulation are assigned

to the same trackster. Mathematically, B^3 precision and recall can be expressed as follows:

$$P = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{|T(i)|} \sum_{j \in T(i)} B(i,j) \tag{5.1}$$

$$R = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{|L(i)|} \sum_{j \in L(i)} B(i,j) \tag{5.2}$$

where $T(i)$ is the trackster that contains layer-cluster i, and $L(i)$ are all layer-clusters of the same simtrackster as layer-cluster i. N is the total number of layer-clusters. The function $B(i,j)$ returns 1 if a pair of layer-clusters (i,j) of the same simtrackster are in the same trackster and 0 otherwise.

5.1.2 Adapted B-CUBED metrics

For application in particle reconstruction, two additional requirements need to be considered. Firstly, due to the simulation being performed on the hit level, multiple particles can contribute energy to a single layer-cluster, for example, in the case of overlapping particle trajectories. Some layer-clusters are therefore included in multiple simtracksters, each simtrackster contributing to the layer-cluster by a given energy fraction. As forming layer-clusters is not in the scope of this work, we will focus on comparative results, ignoring the bias introduced by the previous steps of the reconstruction. Secondly, each layer-cluster has an assigned energy, which implies that some layer-clusters have more weight for the reconstructed trackster than others. A pair of two high-energy layer-clusters on the particle trajectory provides a more significant energy contribution to the total trackster energy and therefore is of higher importance than a low-energy layer-cluster pair on the outside of the trackster. Low-energy layer-cluster could be discarded as electromagnetic noise without a significant effect on the total trackster energy, while the high-energy layer-clusters define the core of the trackster.

Energy-aware score

To accommodate that a layer-cluster can be a part of multiple simtracksters, we introduce an additional sum and a normalization factor given by the layer-cluster energy for all simtracksters t that contain the layer-cluster i. Each layer-cluster i has a fraction fr_i^t, as defined in (4.1), of energy $E(i)$ that it shares with the simtrackster t. The concept is generally introduced for any trackster or simtrackster t. However, layer-clusters in reconstruction are only assigned to single tracksters. Therefore, the fraction for reconstructed tracksters is always one. Instead of normalizing by the total number of layer-clusters N, we weight layer-cluster scores by their energy and normalize by the total event energy E. Furthermore, we introduce a weighting factor $E_t(i,j)$, defined as the product of layer-clusters i and j energies in trackster t:

$$E_t(i,j) = (E(i) \cdot fr_i^t) \cdot (E(j) \cdot fr_j^t) \tag{5.3}$$

Then, for a pair layer-clusters (i,j) of trackster t, we introduce a score $B(i,j)$ as a fraction of simtracksters shared by (i,j) over the union of fractions of simtracksters they occur in (which sum up to 2, as fractions of a single layer-cluster must sum up to 1).

$$B(i,j) = \frac{1}{2} \sum_{t \in T : i,j \in t} (fr_i^t + fr_j^t) \tag{5.4}$$

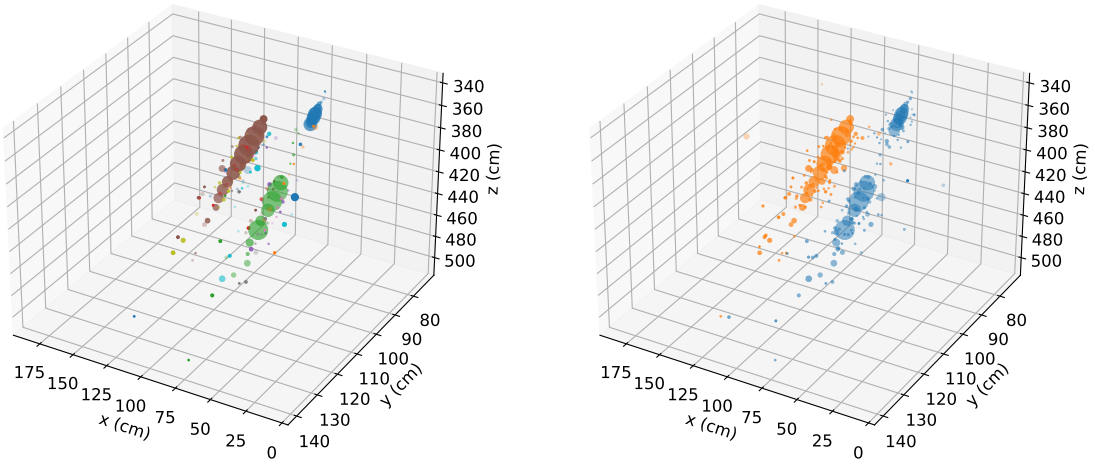


Figure 5.1: TWO CLOSE-BY PIONS. *Two close-by pions data sample plotted by layer-cluster barycenters and energy, with the two simulated particles on the right and 43 reconstructed fragments on the left. The simulated showers consist of 841 layer-clusters, 265 of which contain more than a single hit. The reconstructed shower only contains 220 layer-clusters, while 181 contain more than a single hit.*

Finally, we multiply the score $B(i,j)$ with the weighting factor $E_t(i,j)$:

$$P = \frac{1}{E}\sum_{i=1}^{N}\frac{E(i)}{|T(i)|}\sum_{t\in T(i)}\frac{1}{E(i,t)}\sum_{j\in t}B(i,j)E_t(i,j) \tag{5.5}$$

$$R = \frac{1}{E}\sum_{i=1}^{N}\frac{E(i)}{|L(i)|}\sum_{l\in L(i)}\frac{1}{E(i,l)}\sum_{j\in l}B(i,j)E_l(i,j) \tag{5.6}$$

where $t \in T(i)$ are the tracksters that contains layer-cluster i, and $l \in L(i)$ are all the layer-clusters of the same simtracksters as layer-cluster i. $E(i,t)$ is the total energy of layer-cluster pairs (i,i') for all $i' \in t$, energy $E_t(i,j)$ is the layer-cluster pair energy of i and j in trackster t, and E is the total energy of the event.

Noise considerations

Figure 5.1 shows a reconstruction of a sample two-pion event. Only 265 out of 841 layer-clusters in the simtracksters contain more than a single hit. In reconstruction, only 220 layer-clusters are present, 181 of which have multiple hits. The explanation for this discrepancy is that simtracksters contain all hits related to the simulated particle. In contrast, a portion of these hits will be rejected as noise in the CLUE and CLUE3D steps of the reconstruction. The noise rejection leads to missing layer-clusters in the reconstructed tracksters, which means that a perfect recall in the pattern recognition when comparing against the ground truth data can never be achieved. From the CLUE definition, it is clear that layer-clusters that do not contain at least two hits will likely be rejected as noise. For example, in a multi-particle event, only 35% of layer-cluster in simulation is present in the reconstructed tracksters on average, while only 46% of all layer-cluster contain at least two hits. This work does not aim to improve the recall beyond the scope of trackster

smoothing. Therefore, all single-hit layer-clusters in simtracksters will be ignored in evaluating the reconstruction.

The effect of ignoring single-hit layer-clusters on the reconstructed energy is relatively small. In the example of Figure 5.1, ignoring all the single-hit layer-clusters leads to losing 7% of the total simtrackster energy. By applying the proposed energy-aware B^3 metrics on the sample without single-hit layer-clusters, the event reconstruction has a precision of 1.0, as all layer-cluster pairs in reconstructed tracksters are found in the same simtrackster, and a recall of 0.11 due to the high fragmentation (mitigation of which is the goal of trackster smoothing).

Linking gap

The simulation output considers all hits from the same particle to belong to a single simtrackster even if there is a considerable disconnect between the energy deposits, such as the blue simtrackster on the right in Figure 5.1. Connecting tracksters with a disconnect on the z-axis is the subject of linking, yet the evaluation metrics do not make this distinction. While the improvement in recall remains valid, the absolute value is limited by the maximal considered distance between the fragments. Evaluation in CMSSW does not provide an overall score for the pattern recognition step but instead observes the improvement in purity and efficiency, which loosely translate to the introduced B^3 precision and recall. For this reason, we will focus on recall improvements rather than the absolute value itself. This gap is not a concern for production deployment, as the CMSSW evaluation addresses the longer-distance disconnects in the linking step. A "target" ground truth dataset where tracksters are only reconnected locally based on the simulation data could be generated to evaluate whether the theoretical limit for improving the recall has been reached.

5.2 Baselines

Along the lines of the defined evaluation metrics, the challenge is to increase the reconstruction recall without sacrificing much of the precision. The preceding clustering steps in the reconstruction tuned conservatively to avoid merging layer-clusters from different particle showers. Before diving into machine-learned methods to reconnect the energy fragments driving up the recall, we evaluate the performance of classical approaches such as centroid and density-based clustering, principal-component analysis, and heuristic approaches to set up baselines to support comparison. Within the heuristic approaches, we consider naive, threshold-based implementation described in Section 5.2.2. Setting up rigorous baselines will enable us to reason about the benefits and drawbacks of applying machine-learned approaches to the problem.

Connecting all the hits created by the same particle can be expressed as the problem of point clustering in 3D. Following the TICL framework, hits are already clustered in 2D – creating layer-clusters, and 3D – creating trackster fragments. Iterative clustering abstracts away the problem of dealing with numerous hits from the detector layers and instead focus on the higher-level objects, described by trackster barycenter, energy, number of vertices, and extracted features such as principal axes or graph-based metrics. In the experiments, we will work our way up, starting from the simplest dataset case, and establish baselines for each scenario using the applicable clustering methods.

5.2.1 Clustering methods

Depending on the setup, we consider centroid-based, density-based, and agglomerative algorithms presented in Table 5.1. Algorithms used in the previous steps of the reconstruction pipeline

Table 5.1: OVERVIEW OF THE CONSIDERED CLUSTERING ALGORITHMS. *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 (Xu and Tian, 2015; Cheng, 1995). (*) CLUE scales linearly in the range of multiplicity m relevant for HGCAL and $n > k \gg m \equiv n/k$.*

Algorithm	k-means	GMM	DBSCAN	MeanShift	Agglomerative	CLUE
Initialization	centroids	centroids	density	density	distances	density
Complexity	$O(nkt)$	$O(n^2kt)$	$O(n \log n)$	$O(n^2t)$	$O(n^2)$	$O(n)$*

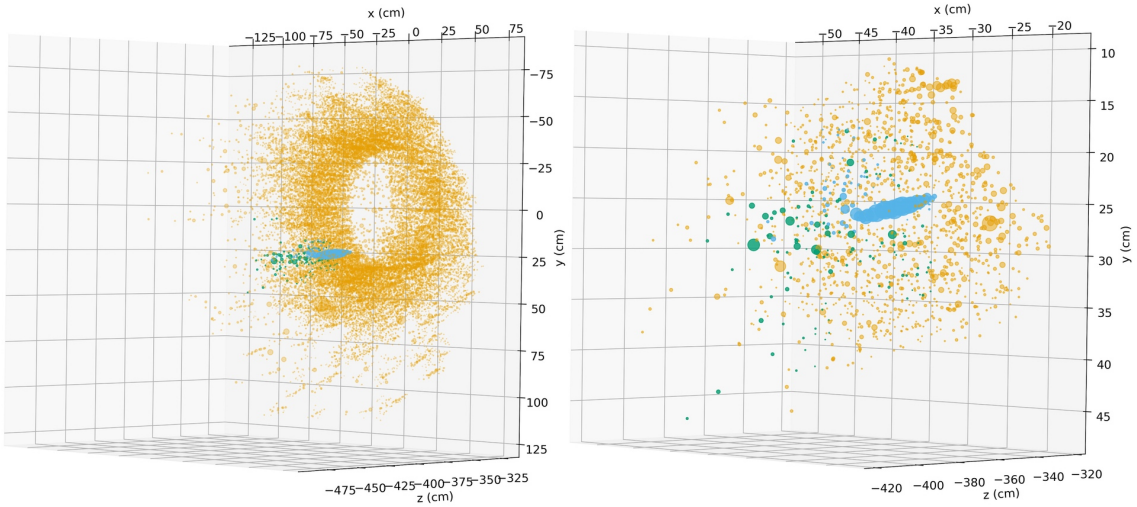
use density and distance parameters to initialize the clusters, as the number of particles is not known in advance in the production environment. In the particle recognition step of the reconstruction, we also consider centroid-initialized approaches, as once the layer-clusters are connected into tracksters, the number of centroids can be initialized by the number of tracksters minus the number of incomplete fragments that we aim to reconnect, for example, based on an energy threshold or a binary classification model. K-means algorithm suffers from high sensitivity to outliers and does not fit non-spherical data distributions by default. In this regard, using Gaussian mixture models (GMM) or non-Euclidean distance metrics should be considered instead. GMM models, on the other hand, suffer from higher computational complexity as more parameters need to be fitted. The suggested model should also perform better than CLUE and CLUE3D algorithms with more relaxed settings, which imposes a natural baseline for the problem.

5.2.2 Naive approach

Finally, we will consider simple algorithmic implementations based on domain knowledge. We consider a cylindrical neighborhood (discussed in Section 5.3.1) around a high-energy trackster, in which all low-energy tracksters with their barycenter within the cylinder are connected. This method resembles centroid-based clustering algorithms using a custom distance metric in a limited radius, with a centroid defined by the trackster barycenter. It also provides a direct baseline for neighborhood-based learned methods described in the Section 5.4.

5.3 Machine-learned methods

Due to the high number of tracksters in the calorimeter and the number of connection candidates growing quadratically in this number, it is not feasible to attempt to solve the entire endcap reconstruction in one giant step. Instead, we expect to focus on smaller cylindrical regions around high-energy tracksters, see Figure 5.2. As the simulation data and associator scores will not be available in the production setting, we select the tracksters of interest based on an energy threshold. The task at hand can thus be formulated as: given a high-energy trackster, reconnect to all low-energy fragments deposited by the same particle in the trackster neighborhood. Contrary to a single-step approach, the neighborhood-based approach is highly parallelizable – in the number of non-overlapping neighborhoods or high-energy tracksters if overlaps are permissible. For each high-energy trackster, we can either consider a pairwise binary decision, independent of all the other tracksters in the neighborhood or a point cloud approach considering all tracksters at once. Including the neighborhood information could enable the classification by gaining additional information about the particle shower.



(a) Pion trackster within the endcap

(b) Pion trackster within its neighborhood

Figure 5.2: NEIGHBORHOOD OF A PION TRACKSTER. *A cylinder neighborhood of a pion trackster in the high pile-up dataset. (a) Position of the pion trackster within the calorimeter endcap. (b) A zoomed-in view into the neighborhood of the reconstructed trackster. The high-energy pion trackster of interest is shown in blue, while neighboring tracksters coming from the same simulated pion with an associator score below 0.5 are shown in green.*

In this section, we narrow the problem definition to operating on trackster neighborhoods and discuss the features used to describe tracksters. Subsequently, Section 5.4 and Section 5.5 discuss the pairwise and graph-based approaches machine-learned approaches for the particle shower reconstruction.

5.3.1 Neighborhood definition

The high number of low-energy fragments (as discussed in Section 4.3) motivates approaching the trackster smoothing task along the lines of reconnecting low-energy fragments to higher-energy tracksters. Therefore, for each high-energy trackster, we consider all low-energy tracksters within a certain neighborhood; see Figure 5.2.

Due to the irregular geometry of the detector, traditional distance metrics such as 3D Euclidean distance do not perform uniformly across layers. While a certain distance threshold might perform well in the electromagnetic section with short inter-layer gaps, it might barely cover two consecutive layers in the hadronic section. Instead, we consider numerical layer identifiers on the z-axis for same-particle fragment identification.

We define the trackster neighborhood by a **cylindrical region** around the trackster axis. The **trackster axis** is defined by the line connecting its barycenter with the center of the detector (collision point). At the same time, the cylinder is bounded by the first and last layer occupied by the trackster. Then, we consider a trackster to be in another trackster's neighborhood if its barycenter is within the other trackster's cylindrical region. The radius of the cylindrical region is a free parameter and serves as a threshold defining the size of the considered neighborhood.

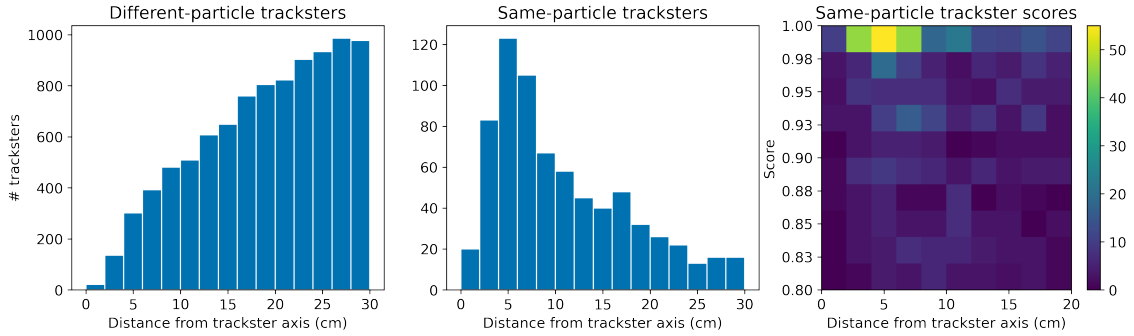


Figure 5.3: NEIGHBORHOOD OF PION TRACKSTERS. *Distance of the neighborhood tracksters to a high-energy pion trackster axis in an environment with a pile-up of 200, over 10 thousand trackster pairs. Pions show a heavier tail of the same-particle (associator score below 0.2) distance. However, as shown in the figure on the right (where the "Score" is one minus the associator score), most of the tracksters with high scores are located within 10 centimeters radius.*

Neighborhood radius

The optimal radius threshold depends on the dataset given by the distances between generated particles, particle types, and whether the pile-up is present. In production, we can either find a threshold fitting all particle types, as long as the chosen reconstruction method will support this, or utilize the provided identification probabilities of the trackster of interest to select one.

In Figure 5.3, we show the distance distribution of the same and different particle tracksters from a selected high-energy trackster axis. The high-energy trackster is selected as the one best representing the simtrackster given the associator scores. Due to high fragmentation, pions have a heavier tail of the same-particle trackster distance distribution beyond 20 centimeters. However, fragments with good associator scores are primarily within the 10 centimeters radius. For photons, nearly all candidate tracksters are located within 15 centimeters. Most of the same-photon fragments are within the 10 cm radius; see Figure A.4 in the Appendix A.1. Reconnecting fragments from the same particle over longer distances becomes infeasible in a high pile-up environment. Given the associator scores, most of the tracksters of interest lie within the 10 centimeters radius region; thus, we will consider it our default setting.

5.3.2 Feature extraction

Metrics expressing the spatial and energetic properties of the tracksters are needed to abstract the layer-cluster granularity. Describing the tracksters by features beyond their position and energy enables the recognition of related energy deposits based on the shower shape and direction. Here, we describe the subset of extracted features, including point cloud (layer-cluster) based aggregates, geometrical features, and structural graph features.

Aggregate features

The underlying data is inherently a point cloud. One way to describe it is to aggregate the attributes of its constituents, including the number of layer-clusters and the total layer-cluster energy. From the definition of the problem, we know that the candidates typically have low energy and few layer-clusters.

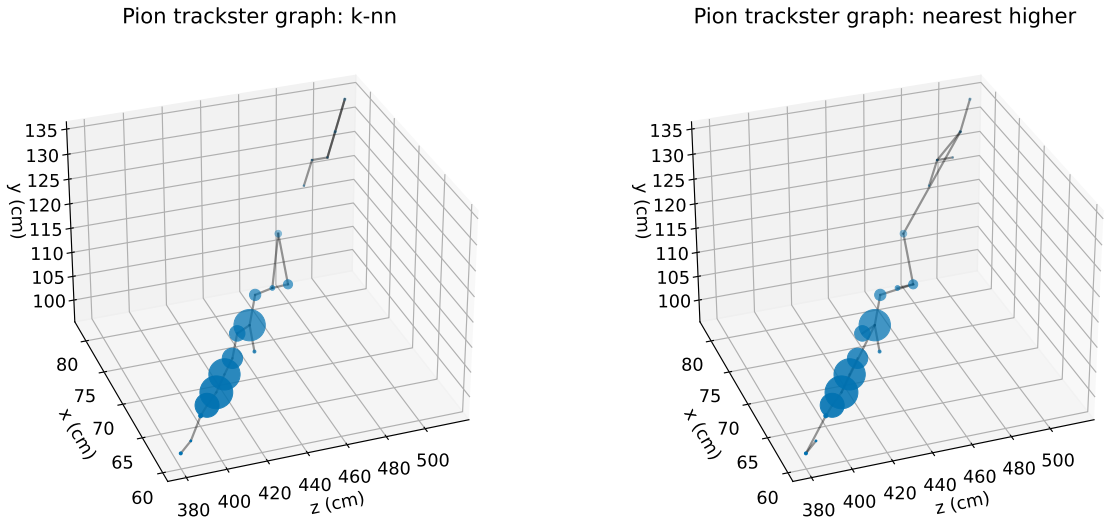


Figure 5.4: GRAPH OF A PION TRACKSTER. *A graph constructed from the pion trackster point cloud using k-nearest neighbors (left) and nearest-higher energy neighbor algorithms (right), where k (the number of neighbors) is 1. The nearest-higher algorithm with the $k = 1$ is a tree generation algorithm and produces a connected graph.*

Geometrical features

Geometrical features are meaningful in the context of multiple tracksters. Generally, nearby tracksters pointing in the same direction or stemming from the same point in space could be generated by the same particle. Our subset includes:

1. barycenter, as the mean of layer-cluster coordinates weighted by their energy, expressed in Cartesian and polar coordinates,
2. principal axes of the trackster computed from the layer-cluster barycenters, and their deviations,
3. coordinates of the first and the last layer-cluster along the z-axis.

In combination with the aggregate features, geometrical features provide a high-level view of the tracksters' disposition in space.

Structural graph features

In addition to the high-level aggregate and geometric features, we introduce a set of structural features to capture the inner disposition of the trackster. The highest-energy layer-clusters are typically along the directional axis of the shower. In contrast, the lower-energy layer-clusters tend to be further away, as shown in Figure 5.2. We exploit the characteristic high-energy core of the shower to construct edges between the tracksters by a simple tree-building algorithm: for each layer-cluster, find the nearest neighbor with higher energy and connect to it if it exists. Compared to the k-nearest neighbors method used to construct graphs from point clouds in DGCNN by Wang et al. (2019), connecting to higher-energetic neighbors ensures that the graph is connected. The connectivity is ensured because each layer-cluster, but the highest-energy one, must have at least one layer-cluster with higher energy. Then, each node is connected to at least one

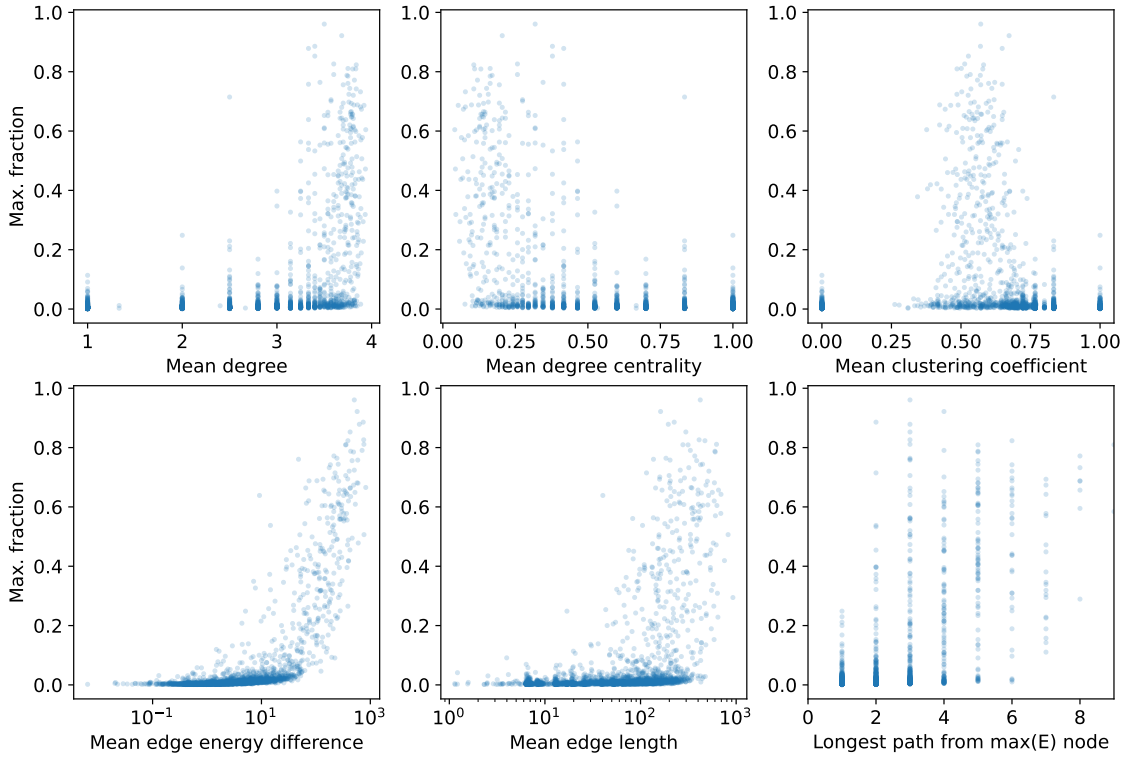


Figure 5.5: GRAPH FEATURES OF PION TRACKSTERS. *Extracted structural features from a set of pion tracksters. On the y-axis, we show the highest energy fraction of any simtrackster shared by the trackster. In other words, the highest simtrackster fraction reconstructed by the trackster. Each feature shows the dependency between the feature value and the highest fraction. For example, tracksters with high mean degree centrality are unlikely to represent a significant part of any simtrackster.*

other node, highlighting the energy chain in the high-energetic core, see Figure 5.4. In a practical setting, connecting to multiple higher-energy neighbors could be beneficial to support the flow of information through the graph.

Building a connected graph enables the use of traditional network science methods describing the graph structure, commonly referred to as graph kernels, to infer information about the trackster. The descriptions can be based on the graph (trackster) level or node (layer-cluster) level. On the graph level, aggregation methods referred to as Bag of Nodes (Hamilton, 2020) and path-based methods are applicable (Kriege et al., 2020). We consider the following attributes:

1. mean node degree, where the degree captures the number of node's neighbors,

2. mean degree centrality, capturing the average node degree normalized by the maximum possible degree in the network ($n-1$ for n nodes),

3. mean clustering coefficient, where the clustering coefficient c for a node u, with $T(u)$ being the number of triangles through node u, is computed as (Kaiser, 2008):

$$c_u = \frac{2 \cdot T(u)}{deg(u) \cdot (deg(u) - 1)} \tag{5.7}$$

4. mean edge energy difference, capturing the mean difference in energy $\Delta E_{ij} = |E(i) - E(j)|$ for each edge ij in the graph,

5. the mean edge length, computed as the mean Euclidean distance between barycenters of nodes i and j for each edge ij, and

6. the longest path from the highest-energy layer-cluster, which, starting in the layer-cluster of the highest energy centrality, finds the shortest path to all the other nodes using Dijkstra's algorithm and returns the length of the longest path. This metric is based on the intuition that well-reconstructed tracksters have a long, linearly connected high-energetic core of the shower and, therefore, a longer path from the center to the leaf nodes.

Compared to the geometric features, structural features can be meaningful independently, not only in combination with the other candidate trackster. For example, degree, clustering, and centrality metrics give an insight into whether the graph is formed around a single traversing particle – thus has a high-energy core or a rather unorganized bulk of energy deposits, see Figure 5.5. The latter is more likely to be a disconnected fragment outside the shower rather than a part of the shower's core.

Furthermore, numerous methods for comparing graphs, sub-graphs, and evaluation of graph isomorphism are available; however, while using the features extracted from the graph can be meaningful to describe the tracksters, given the nature of the problem, directly comparing the graphs' isomorphism might not convey whether the two trackster were produced by the same particle. Therefore, we limit ourselves to using the extracted features and not directly considering the constructed trackster graph structures in reconstruction.

5.4 Pairwise approach

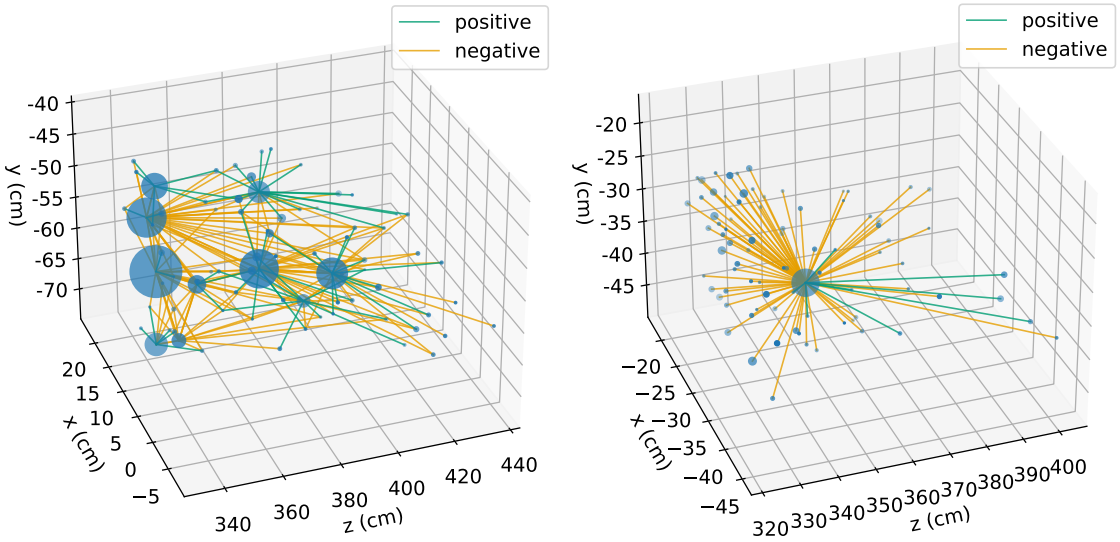
This section introduces a pairwise machine-learned approach for calorimetric clustering based on a standard multi-layer feed-forward neural network. A multi-layer feed-forward neural network, also called multi-layer perceptron (MLP), is a universal function approximator (Hornik et al., 1989) and has a long history of applications across domains, including physics applications such as the classification of proton and electron whistlers by Miniere et al. (1996).

We explore the task of classifying trackster pairs in isolation, that is, given a pair of a high-energy trackster and a low-energy trackster, decide whether the same particle produced the two (see Figure 5.6). This setup simplifies formulating the problem to a binary classification task – given two tracksters, give a binary decision on whether they should be connected. Furthermore, the pairwise approach serves as a baseline for graph methods and an introduction to learned parameterized methods without introducing any graph or point-cloud-specific operations.

5.4.1 Pairwise scores

The high-energy tracksters around which the classification is performed are selected by an energy threshold e_m. However, in the pile-up case, the simtracksters are only available for selected tracksters and not for the rest. Therefore, apart from the energy, the tracksters are only selected for smoothing if at least half of their energy belongs to the simulated particle – as the rest cannot be evaluated.

After selecting a high-energy trackster $t : E(t) \geq e_m$, pairwise samples with all low-energy tracksters t' in the neighborhoods are produced by concatenating the trackster pair features. Each sample is scored between zero and one using the recoToSim associator scores. The pairwise score



(a) Multi-particle dataset (10 cm neighborhood)

(b) Pion dataset with pileup (15 cm neighborhood)

Figure 5.6: EDGE-BASED PERSPECTIVE. *An edge-based visualization of a multi-particle event (a) and a high-pileup pion event (b). Tracksters are plotted by their barycenters with the point size defined by their energy. Edges between tracksters denote candidate connections (pairs) within neighborhoods of tracksters above 50 GeV.*

$score(t, t')$ for a (high-energy, low-energy) trackster pair (t, t'), given a simtrackster collection S is computed as:

$$s_t = \arg\min_{s \in S}(recoToSim(t, s)) \tag{5.8}$$

$$score(t, t') = (1 - recoToSim(t, s_t))(1 - recoToSim(t', s_t)) \tag{5.9}$$

where s_t is the simtrackster with the lowest recoToSim score for high-energy trackster t.

Dataset imbalance

Depending on the dataset setting, edge labels are very imbalanced. For example, while in the pile-up case, only 8% of the connections within a 10 cm radius are positive (given a recoToSim associator score below 0.2), this figure is 51% for the multi-particle case and 88% for the two pion case, see Figure 5.6. The significant imbalance means that methods tuned on the two-pion or multi-particle case will likely perform poorly on the pile-up case and vice-versa. Additionally, the associator scores are not binary but continuous. For labels, we use one minus the associator score; hence 1 means a perfect match, and 0 means no connection. Generally, tracksters with a recoToSim score below 0.2 (thus label score above 0.8) are considered a good match.

5.4.2 Loss function

Addressing the large imbalance in the data is essential for training learned models to avoid learning the majority class only. We consider the Focal Loss (Lin et al., 2017) method to focus on the

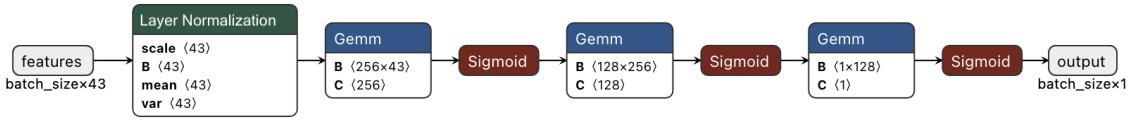


Figure 5.7: PAIRWISE MODEL SCHEMA. *The pairwise model uses a simple multi-layer perceptron architecture with three general matrix multiplication (Gemm) layers prepended by a layer-normalization step. On the input, a pair of tracksters and their mutual distance is encoded into 43 features. The output of the model is a single value for each pair.*

training on a sparse set of hard examples and prevent the model from defaulting to the easy negatives. As the majority class is generally easy to predict (positive for two pions and negative for pile-up datasets), focusing on the misclassified samples naturally solves the class imbalance. The Focal Loss is based on cross-entropy for binary classification. The cross-entropy is defined as:

$$CE(p, y) = \begin{cases} -\log(p) & \text{if } y = 1 \\ -\log(1-p) & \text{otherwise,} \end{cases} \tag{5.10}$$

where $y \in \pm 1$ is the ground-truth class and $p \in [0, 1]$ is the probability for the class $y = 1$ estimated by the model. By defining p_t as:

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1-p & \text{otherwise,} \end{cases} \tag{5.11}$$

the cross entropy (CE) formulation can be rewritten as $CE(p, y) = CE(p_t) = -\log(p_t)$. Focal Loss introduces a modulating factor $-(1-p_t)^\gamma$ to weight-down the easy samples from the majority class and focus on the sparse hard samples. The parameter γ is commonly set to 2, which leads to an example classified with $p_t = 0.9$ to have $100\times$ lower loss compared to the original CE formulation. Focal Loss (FL) is then defined as:

$$FL(p_t) = (1-p_t)^\gamma \log(p_t). \tag{5.12}$$

However, the Focal Loss formulation works with binary labels $y = \pm 1$, while the sample scores defined in (5.9) are continuous. Li et al. (2023) extend the cross-entropy part (5.10) into the complete formulation CE_c:

$$CE_c(x, y) = -((1-y)\log(1-x) + y\log(x)) \tag{5.13}$$

and generalize the scaling factor $-(1-p_t)^\gamma$ into the absolute distance between the estimate x and its continuous label y as $|y-x|^\gamma$, with $|.|$ as the absolute value operator. Subsequently, they define a loss objective called Quality Focal Loss (QFL):

$$QFL(x, y) = -|y-x|^\gamma((1-y)\log(1-x) + y\log(x)). \tag{5.14}$$

In the QFL formulation, the model can be directly trained on the scores defined in (5.9) while focusing on the hard samples.

5.4.3 Model

We choose a simple multi-layer perceptron (MLP) based model, using two or three hidden layers. Sizes of the layers (*e. g.*, 256, 128, 128), non-linearity types, and dropout rate are determined

in hyper-parameter tuning. The deployment goal for the model is integration into CMSSW by exporting it to the ONNX format. To ease the integration, we choose to keep the normalization as a part of the model. Therefore, we use layer normalization by Ba et al. (2016). Unlike commonly used batch normalization (Ioffe and Szegedy, 2015), layer normalization is independent of the batch size and performs the normalization feature-wise over all elements. While a batch normalization layer applied to the input data, as proposed by Qasim et al. (2019) is meaningful if the samples within the batch are related and sampled within the same geometrical space, the pair-wise approach considers the tracksters pairs independently. Therefore, normalizing given other samples within the batch could introduce a bias in the production deployment. An example model schema[1] is shown in Figure 5.7.

Reconstruction and evaluation

The performance of the model on the learning task in training is evaluated against the dataset labels from the validation set using standard metrics such as precision, recall, and the area under the receiver operating characteristic (ROC) curve (AUC) (Bradley, 1997). Once the best-performing model is identified by AUC, the qualitative improvement in event reconstruction performance is evaluated using B^3 precision and recall on a sample of unseen events. This two-step approach first identifies the best-performing model on the given dataset, and then the B^3 metrics compare multiple approaches and the overall improvement in event reconstruction.

To enable the application of the B^3 evaluation, predicted trackster pairs must first be used to reconstruct the event. However, due to the possible overlaps between simulated particles, this is a non-trivial task. In the pile-up case, only one simtrackster is addressed at a time, and overlapping showers could be addressed sequentially to avoid conflicts. However, conflicts must be resolved during run-time in the reconstruction in the case of two or more simtracksters. For example, if a low-energy trackster is predicted to belong to two different high-energy tracksters, applying both connections would potentially connect two tracksters coming from two different particle showers, merging the showers. We address this by post-processing the pairwise predictions within an event. For each low-energy trackster t', find a high-energy trackster $t = \arg\max_{t:E(t)\geq e_m} p(t,t')$ with the highest predicted score, and if $p(t,t') \geq c_t$, where c_t is the minimum connection score threshold, connect the tracksters.

5.5 Graph approach

This section considers approaching the problem as a point cloud of tracksters. This approach aims to simultaneously evaluate all the tracksters within the entire event or a selected neighborhood (such as in Figure 5.6). This idea is motivated by the assumption that including nearby tracksters brings in additional information about the shower, which supports the classification performance. This problem can be phrased in two ways: a binary classification task, where each trackster is predicted to be a part of the shower or not, or a link prediction task, where edges denoting connections between tracksters are predicted.

5.5.1 Classification task

In the classification formulation, the event, or the selected trackster neighborhood, is interpreted as a point cloud of tracksters. The downside of this approach is that it can only be applied to

[1] Schema generated using Netron: https://netron.app

a fixed number of particles at a time, like two pions, or a single high-energy trackster neighborhood within pile-up (in the multi-particle case, we can also consider the background particles as pile-up and process the event sequentially). In GravNet, Qasim et al. (2019) addressed a similar calorimetric clustering problem directly on hits deposited by two close-by pion showers $s_1, s_2 \in S$ by defining a loss function:

$$L = \sum_{s \in S} \frac{\sum_i \sqrt{E_i y_{is}}(p_{is} - y_{is})^2}{\sum_i \sqrt{E_i y_{is}}} \tag{5.15}$$

where p_{is} and y_{is} are the predicted and true energy fractions in calorimeter sensor i and shower s, weighted by the square root of $E_i y_{is}$, the total energy deposit in sensor i of shower s.

We extend the loss function L to application on tracksters rather than direct energy deposits in the calorimeter sensors. The GravNet training setup considered two classes of the shower s, the foreground class f_c and the background class b_c, with labels assigned at random. This formulation fits both the high-energy trackster neighborhood approach or two particles case. However, in the pile-up case, the simtrackster for the background class is not available. Therefore, we formulate the loss function only given the foreground-class simtracksters s, and a collection of reconstructed tracksters T by separating the foreground and background terms as:

$$L = \frac{\sum_{t \in T} E(t,s)(p_{ts} - y_{ts})^2}{1 + \sum_{t \in T} E(t,s)} + \frac{\sum_{t \in T}(E(t) - E(t,s))((1 - p_{ts}) - (1 - y_{ts}))^2}{1 + \sum_{t \in T} E(t) - E(t,s)} \tag{5.16}$$

where $y_{ts} = 1 - recoToSim(t,s)$ is the associator score of trackster t with respect to simtrackster s, p_{ts} is the predicted probability of t being of the same class as s, $E(t,s)$ is the shared energy of t and s, and $E(t)$ is the total energy of t. We omit the square roots and add 1 to the denominator for better numerical stability. If advanced mini-batching is applied, each event's loss must naturally be computed separately.

Reconstruction and evaluation

As in the pairwise case, first, the best-performing model is identified by the AUC or loss on the validation set. Due to the binary setting, reconstructing the event for B^3 evaluation given the model predictions is simpler.

In the two-pion case, reconstructed tracksters are be classified into two classes. Due to the simplicity of the events and only two simtracksters being present, the neighborhood radius constraint can be omitted. First, we choose the highest-energy reconstructed trackster in the event and define the foreground class f_c as all the tracksters having the same predicted label as the selected trackster (predicted score above some threshold c_t). Conversely, we define the background class b_c as all the other tracksters in the event. Finally, we connect all the tracksters in f_c and b_c, forming two **super-tracksters**, and evaluate them against the simtracksters.

In the pile-up case, two adjustments are necessary. First, as the pile-up events only contain one simtrackster, only the foreground class f_c will be considered in the evaluation. Secondly, due to the radius constraint imposed by the pile-up, we consider all high-energy tracksters that share at least half of their energy with the simtrackster. However, multiple high-energy tracksters with overlapping neighborhoods may be selected. We choose to merge overlapping predicted foreground classes (super-tracksters) that share at least 50% of their energy to enable comparison, as connecting such high-energy tracksters is permitted in the pairwise approach. If less than 50% of the super-trackster energy is shared, the conflicting tracksters are assigned to the super-trackster with higher energy. This heuristic produces an equivalent target improvement in B^3 as the pairwise approach, enabling model comparison. However, improving conflict resolution via learned

methods could significantly improve reconstruction quality. This issue is further discussed in Section 7.1.2.

5.5.2 Link-prediction task

Secondly, the problem can be formulated as a link prediction task. This approach is a direct extension of the pairwise case, where the relationship between a pair of tracksters denoting whether they belong to the same particle shower is predicted. Directly including the edge labels also permits processing the whole event with multiple high-energy tracksters at once. GNN-based link-prediction methods such as Graph Auto-Encoders (GAE) (Kipf and Welling, 2016) first obtain individual node representations through a GNN and then represent links by aggregating the nodes of interest. While Zhang et al. (2021) showed that the GAE approach cannot learn the structural link representations in the graph and might fail to distinguish isomorphic links, this is not a concern for operations on the point clouds, as they inherently are unordered sets with no fixed semantic links. However, as the graph structure is not defined, we follow the sub-graph-based as used in SEAL by Zhang and Chen (2018) to learn node representations given their neighborhoods using a DGCNN-based network architecture. Then, we construct a latent representation $h_{tt'}$ of the edge of interest (t, t') by aggregating the adjacent node representations (h_t, h'_t) as:

$$h_{tt'} = \sigma(h_t, h'_t) \tag{5.17}$$

where σ is the aggregation function. We chose to aggregate the node representations by concatenating their feature vectors. Other commonly used approaches include node representations' mean, sum, and bilinear or Hadamard product (Liben-Nowell and Kleinberg, 2003). The edges of interest are selected as in the pairwise approach by examining neighborhoods of high-energy tracksters and providing them as a labeled edge list along with the point cloud. The score for trackster pair (t, t') is defined analogously to the pairwise approach by (5.9). GAE and SEAL models use cross entropy-based loss functions to train models for a link prediction task. Due to the edge label imbalance corresponding to the pairwise case, we will use Quality Focal Loss in training.

Reconstruction and evaluation

The link prediction output can be evaluated analogously to the pairwise case, except the trackster pairs are not evaluated in isolation. Therefore, the model is aware of low-energy tracksters connecting to multiple high-energy tracksters and might be able to distinguish whether they belong to the same particle. Therefore, the requirement of low-energy tracksters connecting only to a single high-energy trackster can be lifted, which provides a potential for improvement in the reconstruction performance.

Models

In both task formulations, we follow the DGCNN paradigm (Wang et al., 2019) with the EdgeConv neural message passing, see Figure 5.8. Based on the related applications in physics and high flexibility, we used an adapted version of the ParticleNet architecture (Qu and Gouskos, 2020), which replaces the max aggregation of nearest neighbors' hidden states with a mean aggregation operation. EdgeConv layers within the network can either build the edges dynamically, such as in the original DGCNN or work on static pre-built graphs, as proposed by Ju et al. (2020). Using a fixed structure based on the spatial properties of the tracksters comes with a lower computational cost and would be preferred for performance reasons. However, it loses the possibility of letting the learned model reshape the point cloud. The be The last layers of the network are

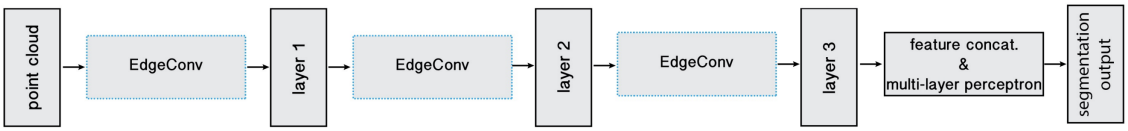


Figure 5.8: DGCNN ARCHITECTURE. *The architecture of the Dynamic Graph CNN by Wang et al. (2019). First, three layers of EdgeConv are gradually applied to the point cloud, aggregating information from the latent neighborhood in each step. Then, features for each node or edge are passed through a dense layer producing the segmentation or link-prediction output.*

adapted to suit the task formulation, either keeping the node-wise dimensionality followed by a SoftMax function for the foreground and the background class or encoding nodes into edges by concatenating their feature vectors based on an edge list and producing edge-wise scores.

5.6 Approach summary

This work considers the problem of selecting high-energy tracksters and reconnecting them with the energy fragments produced by the same particle in the trackster neighborhood. The neighborhood is defined by a cylinder around the high-energy trackster, limited by the trackster length on the z-axis and a certain radius. Datasets with different levels of overlap are available, from well-separated two-particle datasets and partially overlapping multi-particle datasets to high-overlap pile-up datasets. The overlap in the dataset affects the level of imbalance between the classes. We use standard evaluation metrics such as AUC to find the best-performing models on each dataset and a set of adapted clustering metrics based on B^3 precision and recall to evaluate the impact of the given approach on the event reconstruction performance. To enable comparison, we establish baselines using clustering methods. The learned models utilize features that describe the spatial and structural properties of tracksters to abstract away from the layer-cluster granularity. First, we consider a simple case of pairwise binary classification, where we concatenate features of two tracksters and train a model to decide whether they belong to the same simtrackster. Later, we expand this approach to the entire trackster neighborhood using graph and point cloud methods based on neural message passing mechanisms.

Chapter 6

Experiments

This chapter focuses on selecting AI methods applicable to calorimetric clustering and comparing the learned approaches with classical clustering methods. First, the pairwise case is described, explaining the experimental setup, pre-processing, feature extraction, and model training considerations. Next, the findings are evaluated against the baselines to answer the first research question, whether using machine-learned methods provides a benefit in the first place. Then, the approach is extended to graph-based models in search of superior reconstruction performance and better computational complexity, answering the second research question – what architectures are viable, and do graph and point cloud methods bring a benefit beyond the pairwise MLP approach. Finally, in the next chapter, we compare the two in terms of physics performance and complexity and argue about the scalability toward large datasets.

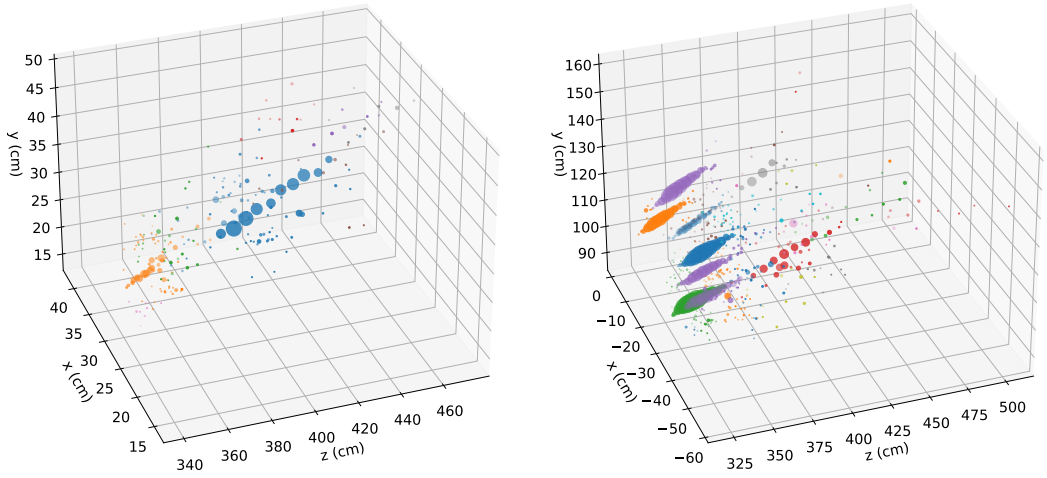
6.1 Setup

The CMS software implements a processing pipeline for the particle shower reconstruction, connecting all the relevant systems. Deploying deep learning models on the system is supported through the Open Neural Network Exchange format (ONNX[1]). However, integration into the pipeline requires implementing the feature extraction, input preparation, and model result interpretation in CMSSW using C++. Additionally, due to the system complexity and dependencies, the pipeline has to be run on a compatible, CernVM-FS-enabled (Blomer et al., 2017) high-performance computing system.

While integration into CMSSW is essential for precise physics performance evaluation, the complexity of the environment does not make it well-suited for exploratory analysis and experimentation. For this reason, we will conduct the experiments in Python. We use the available libraries to manipulate physics data, develop AI-based models, and implement the adapted evaluation metrics based on B^3 in Python. The experiments are conducted using a set of Jupyter Notebooks with an accompanying Python code-base, both published at GitHub[2]. The raw collision data format is native to CMSSW yet not well suited for manual processing in Python. For this reason, a CMSSW module called *ntuplizer* was developed to serialize the events to an array-based format, where events are hierarchically structured from tracksters to layer-clusters. Reading the content of ROOT files in Python is handled by using the Uproot library[3].

We use PyTorch[4] for training learned models and PyTorch Geometric[5] for GNN experiments.

[1] ONNX format: https://onnx.ai
[2] Code repository: https://github.com/edcuba/TICLPatternReco
[3] Uproot library: https://uproot.readthedocs.io
[4] PyTorch documentation: https://pytorch.org
[5] PyTorch Geometric documentation: https://pytorch-geometric.readthedocs.io



(a) Meanshift on a two-pion event

(b) Meanshift on a ten-particle event

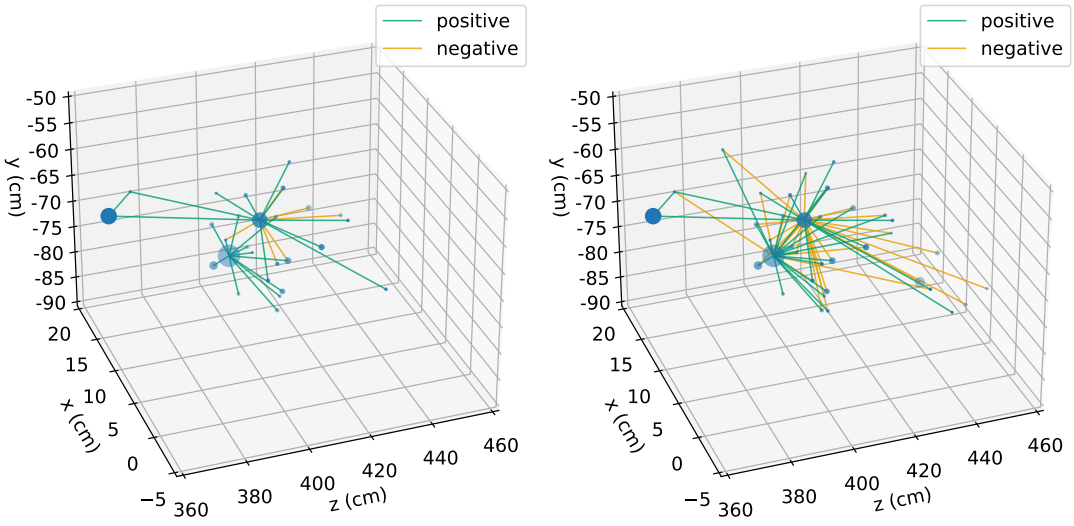
Figure 6.1: APPLICATION OF MEANSHIFT ALGORITHM. *In the two-pions case (a), 36 tracksters on the input were clustered into six tracksters, improving the B^3 recall from 0.12 to 0.43, while decreasing the precision from 0.98 to 0.9. In the ten-particle case (b), the input was fragmented into 71 tracksters, which were clustered into 28, improving the recall from 0.07 to 0.13, along with a drop in precision from 0.85 to 0.84.*

Graph datasets are implemented using a `torch_geometric.data.InMemoryDataset` class that handles variable graph sizes using the batch trick and array re-indexing. The data preprocessing was performed in an HPC environment. Models were trained on an HPC node with an NVIDIA V100 GPU.

6.2 Clustering baselines

Classical clustering baselines are considered in the most direct problem formulation. The clustering algorithms are based on centroids defined by the high-energy tracksters (K-means), density (DBSCAN), bandwidth (Meanshift), or maximum linking distance parameters (agglomerative). The algorithm inputs are the spatial positions of the trackster barycenters weighted by the trackster energy where applicable. The results are presented at the end of this chapter in Table 6.1.

The best-performing parameters for each dataset are identified by the improvement in F-score with $\beta = 0.5$. Using $\beta < 1$ prioritizes precision to recall. Using a low β drives the algorithms to focus on improving recall without considerably sacrificing precision. The Meanshift algorithm provides the best improvement in $F_{0.5}$-score, gaining 0.168 in recall while losing only 0.039 in precision for the two-particles case and 0.126 for 0.046 in the ten-particle case, respectively. In addition, Meanshift reduced the average number of tracksters in the two close-by pions events from 28.1 to 9.2 and from 96.8 to 37.1 in the multi-particle case. Figure 6.1 shows example outputs of the Meanshift algorithm.



(a) Trackster pairs within a 10 cm neighborhood

(b) Trackster pairs within a 20 cm neighborhood

Figure 6.2: PAIRWISE APPROACH: TWO CLOSE-BY PIONS. *Candidate trackster pairs extracted from the neighborhood of reconstructed tracksters above* 40 GeV *from two pion showers within the radius of* 10 cm *(a) and* 20 cm *(b). Tracksters representing the same simtrackster are connected by green (positive) edges, and tracksters from different simtracksters are connected by orange (negative) edges.*

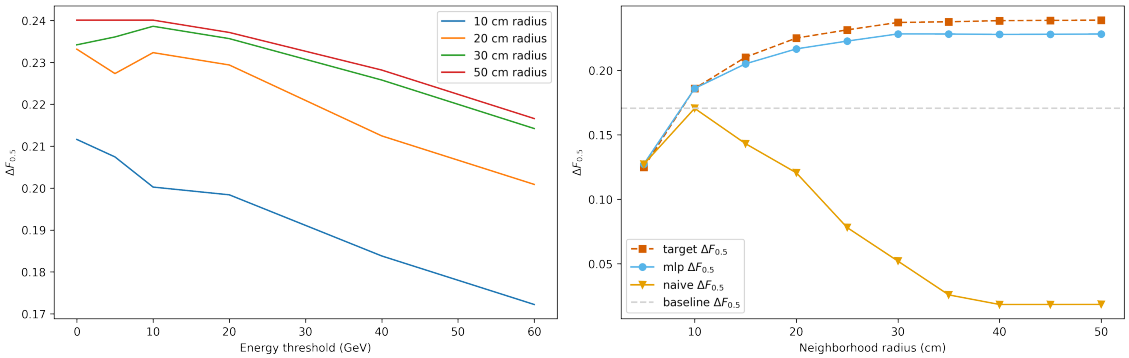
6.3 Pairwise approach

First, we conduct a series of experiments based on the pairwise approach. These experiments aim to test the approach's feasibility, find viable parameters, and compare the model performance to the baselines. Due to the particle shower overlaps leading to different dataset imbalances, particular datasets exhibit various levels of reconstruction difficulty. We start with the two-pion case, find the viable neighborhood, and evaluate the performance of the pairwise approach. Then, we proceed to the multi-particle and pile-up cases and investigate which methods scale to the higher-occupancy scenarios. For the definition of the reconstruction scope, we define the following two parameters of the candidate pairs selection:

1. **Energy threshold** e_m: the minimum energy of a trackster to be selected for trackster smoothing. Setting $e_m = 0$ leads to exploring the neighborhood of each trackster in the event while setting a higher threshold reduces the smoothing scope to fewer, higher-energy tracksters.

2. **Neighborhood radius** r_n: radius of the cylindrical region around the selected trackster. This parameter controls the set of tracksters considered in the neighborhood of the smoothed trackster, see Figure 6.2. For practical reasons, we will also apply this parameter to control the cylinder's length beyond the endpoints of the tracksters, as it enables us to experiment with selecting the entire endcap by setting the r_n to a large value (100cm).

A combination of low e_m and high r_n leads towards quadratic complexity in the number of tracksters. Therefore, for practical purposes of the reconstruction, we will identify a suitable threshold with a good reconstruction performance and low complexity. The foreground-background ratio depends on the selected radius r_n. Given a small enough neighborhood, most of the pairs within



(a) The effect of e_m on target $\Delta F_{0.5}$

(b) The effect on r_n on reconstruction $\Delta F_{0.5}$

Figure 6.3: TWO CLOSE-BY PIONS EVENTS: PAIRWISE APPROACH. *Figure (a) shows the effect of minimum selection energy e_m and neighborhood radius r_n on the targeted reconstruction improvement in two close-by pions events. The targeted improvement is calculated by correctly assigning all tracksters within the r_n neighborhood of high-energy tracksters (with energy above e_m) to the high-energy trackster of the same simtrackster. The improvement measure is the difference in $F_{0.5}$ of B^3 precision and recall between the reconstruction input and the ground truth (target) established using the associator scores. Figure (b) shows the performance improvement obtained by the naive approach (connecting to all tracksters within the neighborhood) and the MLP model. Each trackster can be only connected to one high-energy trackster selected by the highest model prediction score.*

the neighborhood will be from the same particle shower – see Figure 6.2, while extending the radius towards the whole endcap will lead to a roughly balanced dataset as they are two particles. Using a small neighborhood gives rise to a simple naive approach – reconnecting to everything within the neighborhood.

6.3.1 Model training

MLP models given the architecture described in Section 5.4, are trained using the Quality Focal Loss function with $\gamma = 2$, Adam optimizer (Kingma and Ba, 2015) and a cosine-annealing learning rate scheduler as proposed by Loshchilov and Hutter (2017) with the initial learning rate of 10^{-2} decaying to 10^{-5}. Details of the used datasets are reported in Table A.1 in Appendix A.2.

6.3.2 Two pions dataset

Figure 6.3(a), reports the effect of energy threshold e_m on the target improvement in $F_{0.5}$ score given the generated pair labels of the reconstructed event. The plots support the intuition that reconstruction of higher-energy tracksters neighborhoods leads to better results than smoothing all tracksters if a large enough radius is included. Therefore, we choose a threshold $e_m = 10\,\text{GeV}$ as it provides consistent results when extending the neighborhood radius. Furthermore, we consider a larger radius $r_n = 50\,\text{cm}$ motivated by the low level of overlap in two-particle datasets. We observed that training on a larger radius leads to better generalization performance.

Interpretation

Figure 6.3(b) describes the effect of r_n on improving a two-pion event reconstruction, comparing the target improvement given the ground truth with the MLP predictions and the naive approach performance. The overall reconstruction performance is evaluated on a set of 20 unseen events. The precision in the naive approach starts to deteriorate at the radius of $10\,\mathrm{cm}$; below this point, there is no significant overlap between the particles. On the other hand, the gain in $\Delta F_{0.5}$ only starts to plateau the radius of $30\,\mathrm{cm}$. The MLP model reaches the target reconstruction given the ground truth, even with a large radius (up to $50\,\mathrm{cm}$). The best result was obtained on the radius of $30\,\mathrm{cm}$ with $\Delta F_{0.5}(MLP) = 0.228$. This result considerably improves upon the clustering methods presented in Table 6.1. Furthermore, by improving on the naive algorithm's performance, this experiment shows that the MLP model can learn to distinguish tracksters from the same particle beyond the distance from the smoothed trackster axis – as considered in the naive approach with $\Delta F_{0.5}(naive) = 0.170$, still considerably improving upon the clustering baselines.

6.3.3 Multi-particle dataset

In this experiment, we evaluate the performance on a dataset of ten particles of various types. The dataset has a greater level of overlap due to the layer-clusters containing fractions of energy from multiple simtracksters. However, the minimum energy e_m is expected to depend on the simtracksters with the highest-fragmentation characteristics (pions). Therefore, as in the two-pion case, $e_m = 10\,\mathrm{GeV}$ seems to provide the best potential in reconstruction improvement; see Figure 6.4(a). Furthermore, as most of the tracksters are in the sub-10 GeV region (82% in this dataset), using a $10\,\mathrm{GeV}$ threshold significantly reduces the computational complexity compared to evaluating all trackster pairs. We consider a radius of $30\,\mathrm{cm}$ of the smoothed tracksters.

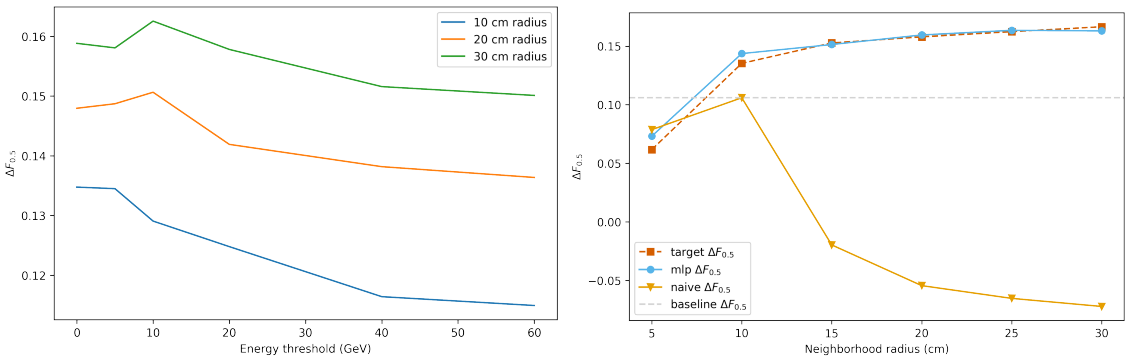
Interpretation

Figure 6.4(b) reports the model's performance against the naive approach and the smoothing target. The overall reconstruction improvement is evaluated on 20 unseen 10-particle events. The model was able to significantly outperform the naive approach and reach the smoothing target throughout the considered neighborhood radius $r_n = 30\,\mathrm{cm}$.

6.3.4 Pion in 200PU dataset

The pile-up dataset is semantically different from the two-particle or ten-particle datasets. While in the previous cases, all the reconstructed energy comes from the simulated particles; the pile-up case is made up of a single particle within hundreds of unknown particles, for which the simulated data are unavailable. The two- and multi-particle cases represent a partitioning problem, while the pile-up case is a pattern recognition problem that aims to distinguish signal from the background. This setup is infeasible for the previously discussed baselines based on partitioning, such as K-Means. Furthermore, it requires an amendment in the evaluation strategy, as due to the lack of labels for the majority of trackster, the comparison among two clusterings is no longer possible.

Instead of the whole event, we only consider the tracksters that share at least half of their energy with the simtrackster for smoothing candidates. In this setup, the B^3 precision describes the energy-weighted ratio of layer-clusters in the tracksters shared with the simtrackster, and B^3 recall describes the ratio of simtrackster layer-clusters that were contained in by a single trackster in the reconstruction. In this experiment, we consider a single pion shot into the calorimeter in a pile-up of 200. We consider $r_n = 15\,\mathrm{cm}$ and minimum selection energy $e_m = 5\,\mathrm{GeV}$.



(a) The effect of e_m on target $\Delta F_{0.5}$

(b) The effect on r_n on reconstruction $\Delta F_{0.5}$

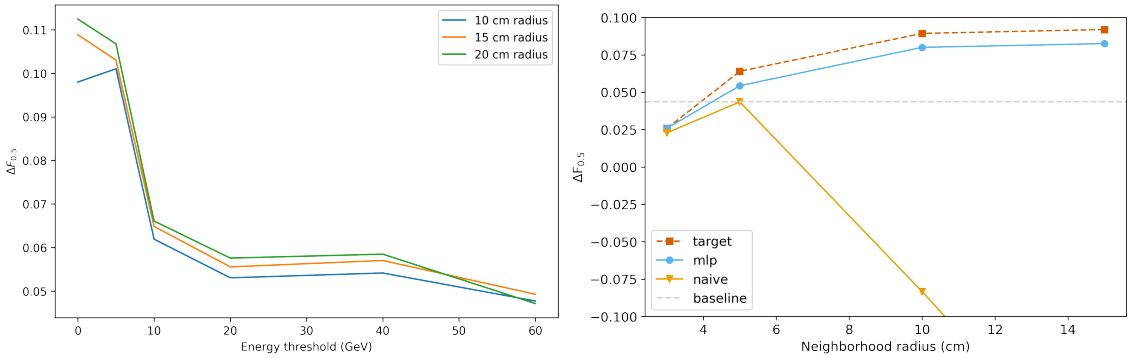
Figure 6.4: MULTI-PARTICLE EVENTS: PAIRWISE APPROACH. *Figure (a) shows the effect of minimum selection energy e_m and neighborhood radius r_n on the target $\Delta F_{0.5}$ in a multi-particle event with 10 simtracksters. The targeted improvement is calculated by correctly assigning all tracksters within the r_n neighborhood of high-energy tracksters (with energy above e_m) to the high-energy trackster of the same simtrackster. The $\Delta F_{0.5}$ is measured as the difference between the input event and reconstruction output B^3 scores. Figure (b) shows the performance of the naive approach (connecting to all tracksters within the neighborhood) and the MLP model. Each trackster can be only connected to one high-energy trackster selected by the highest model prediction score.*

Interpretation

Figure 6.5(a) reports the impact of the minimum selection energy e_m on the target reconstruction improvement. The overall reconstruction improvement in B^3 metrics was evaluated on 20 unseen events. The pion showers in a pile-up environment are significantly fragmented, and most of the tracksters are in the sub-10 GeV energy range. Therefore, setting $e_m > 5\,\mathrm{GeV}$ leads to a significant drop in the target reconstruction improvement. Figure 6.5(b) shows that the MLP model significantly outperforms the baseline and can recognize fragments of the pion shower from within the pile-up even with increasing radius. However, it fails to reach the reconstruction target.

6.4 Graph-based approach

In this section, we address the second research question of this work: do graph neural networks provide a benefit over the pairwise MLP approach in calorimetric clustering, and what are the feasible methods and task formulations? Namely, we compare the node classification and link prediction problem formulations implemented on top of graph neural networks for learning on point clouds using the ParticleNet architecture. The search space for best-performing models in a combination of three datasets with two problem formulations is vast. Given the available datasets and constrained by time and the available processing resources, we report the identified models and their results on a best-effort basis. Our goal is to identify feasible methods. Each approach could be improved individually by expanding the datasets and extensive hyper-parameter tuning, as discussed in Section 7.2.2.



(a) The effect of e_m on target $\Delta F_{0.5}$

(b) The effect on r_n on reconstruction $\Delta F_{0.5}$

Figure 6.5: PION IN 200PU: PAIRWISE APPROACH. *Figure (a) shows the effect of minimum selection energy e_m on the target $\Delta F_{0.5}$ in a pile-up events with a single pion simtrackster. The target $\Delta F_{0.5}$ is computed by connecting all tracksters within the r_n neighborhood of high-energy tracksters to the high-energy trackster of the same simtrackster. The high-energy trackster are tracksters with energy above e_m that share at least half of their energy with the pion simtrackster. Figure (b) shows the performance of the MLP model and the naive approach, given the neighborhood radius r_n. The $\Delta F_{0.5}$ is measured as the difference in $F_{0.5}$ B^3 precision and recall scores of input event and reconstruction output.*

6.4.1 Models

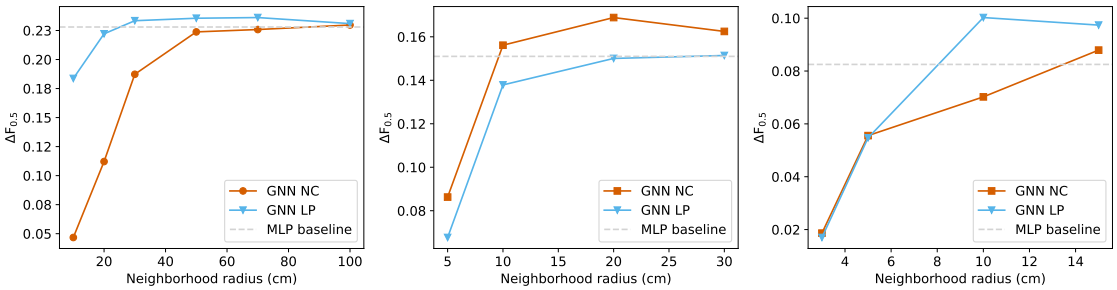
We use adaptations of the ParticleNet architecture by Qu and Gouskos (2020), adapted to the problem formulations. ParticleNet (64^2) and ParticleNet (64^3) are adaptations with two and three EdgeConv layers with 64 nodes each, using eight nearest neighbors, without the skip connections, followed by a dense layer of 256 nodes. ParticleNet (32×64) is an adaptation of ParticleNet-Light with two EdgeConv layers of 32 and 64 nodes each, using seven nearest neighbors with no skip connections, followed by a dense layer of 128 nodes. We use the dynamic graph approach to identify the nearest neighbors in the latent space in each EdgeConv block. Details of the model setups applied to the individual datasets are provided in Table A.1 in Appendix A.2.

Node classification

The node classification models produce two probabilistic class outputs via a SoftMax function. In the two-pion case, entire events are processed at once, classifying tracksters as foreground and background (the highest-energy trackster is assigned the foreground class). This setup significantly simplifies the reconstruction. In the other two dataset scenarios, the foreground-background predictions are performed individually on neighborhoods of tracksters selected for smoothing. Then, the outputs are merged as described in Section 5.5.1.

Link prediction

The link prediction case uses a single logit output per edge. Entire event graphs are considered at once, while edges between the tracksters selected for smoothing and their r_n neighborhoods are selected as labels. This setup produces one graph per event, and the reconstruction is performed by merging all tracksters connected by an edge with a predicted probability above 0.5. Contrary to the pairwise approach, all low-energy tracksters can be merged with multiple high-energy tracksters.



(a) Two pions dataset (b) Ten particles dataset (c) Pion in 200PU dataset

Figure 6.6: GNN APPROACH: TASK FORMULATION COMPARISON. *The comparison between the node classification (GNN NC) and link prediction (GNN LP) approaches on two pions (a), ten particles (b), and pion in a pile-up of 200 (c) datasets. The models are compared in the reconstruction improvement $\Delta F_{0.5}$, between the input event and smoothing output, given the neighborhood radius r_n around the high-energy tracksters selected for smoothing. The high-energy trackster selection setting in each dataset is equivalent to the pairwise approach; see Section 6.3. The best results obtained by the MLP models are provided as baselines for comparison.*

6.4.2 Interpretation

Figure 6.6 reports the performance of the node classification and link prediction models on the two pion, ten particles, and pile-up dataset, compared to the baseline imposed by the best pairwise MLP results. In the two pions case, shown in Figure 6.6(a), the performance of the GNN models is comparable to the MLP approach, outperforming it by a small margin. The link prediction performed the best when constrained to a $50\,\mathrm{cm}$ radius. In the ten particles case, shown in Figure 6.6(b), the link prediction shows equivalent results to the MLP approach. The node classification performs better, however, at the cost of considering each high-energy trackster and its neighborhood in the event separately. In the pile-up case, compared in Figure 6.6(c), both node classification and link prediction show improvement in the reconstruction performance, with the link prediction achieving the best result.

6.5 Results

Table 6.1 compares the baseline performance of the classical clustering algorithms. Table 6.2 provides a summary of the improvement in $\Delta F_{0.5}$ of the tested approaches. The dataset details are provided in Table A.1 in Appendix A.2. Generally, the graph neural network-based approaches achieved higher scores by being able to consider a larger radius. The link prediction approach produced the best results on the two-pion and single-pion in the pile-up case, while the node classification approach scored best in the ten particles case. However, the simpler MLP approach provided a strong baseline in all scenarios.

Finally, Table 6.3 compares the prediction inference time per $100\,000$ samples and 1000 events (throughput) for smoothing tracksters selected from 8000 events with pion in a pile-up of 200.

Table 6.1: CLUSTERING BASELINES. *This table compares the baseline clustering performance on two pions and ten particle datasets, using the trackster barycenters, weighted by the raw energy where applicable. The best-found algorithm configuration given the improvement in F-score with $\beta = 0.5$ is presented. ΔP, ΔR, and $\Delta F_{0.5}$ represent the difference in precision, recall, and F-score compared to the reconstruction input. $\bar{T}$ is the average number of tracksters in the dataset and $\bar{T}_{out}$ is the number of tracksters after the clustering step.*

Algorithm	Dataset	2 pions: $\bar{T} = 28.1$				10 particles: $\bar{T} = 96.8$			
	Params.	$\bar{T}_{out}$	ΔP	ΔR	$\Delta F_{0.5}$	$\bar{T}_{out}$	ΔP	ΔR	$\Delta F_{0.5}$
K-means	$k \in \{4, 8\}$	9	-0.016	0.132	0.083	19.5	-0.061	0.141	0.071
GMM	$k \in \{3, 4\}$	9	-0.033	0.144	0.082	47.05	-0.027	0.067	0.0577
DBSCAN	$\eta \in \{9, 7\}$	11.1	-0.037	0.162	0.086	51.9	-0.062	0.105	0.038
Meanshift	$h \in \{11, 9\}$	9.2	-0.039	0.169	**0.09**	37.1	-0.046	0.126	**0.075**
Agglom.	$d \in \{18, 17\}$	9.9	-0.016	0.109	0.082	32.9	-0.042	0.094	0.066

Table 6.2: LEARNED MODEL RESULTS. *Comparison in the $\Delta F_{0.5}$ improvement of clustering performance on two pions, ten particles, and single pion with pile-up datasets using the pairwise, graph node classification, and graph link prediction approach.*

Model	2 pions			10 particles			Pion in 200PU		
	e_m	r_n	$\Delta F_{0.5}$	e_m	r_n	$\Delta F_{0.5}$	e_m	r_n	$\Delta F_{0.5}$
Naive (best)	10	10	0.170	10	10	0.106	5	5	0.046
MLP (pairwise)	10	30	0.228	10	25	0.151	5	15	0.083
GNN (node classification)	30	$\varnothing$	0.230	10	20	**0.168**	5	15	0.087
GNN (link prediction)	10	50	**0.235**	10	30	0.151	5	10	**0.100**

Table 6.3: INFERENCE TIME OF THE TESTED MODELS. *This table reports the model inference times tested on 8000 events of pions in a pile-up of 200 using a PyTorch backend. The tests were conducted on a system with a ten-core Intel Xeon Gold 6140 CPU and an NVIDIA V100 GPU. The times describe the entire processing loop, including data transfer to and from the GPU. The CPU test utilized all ten cores. Best-performing batch sizes in multiples of two between 8 and 512 were selected in each test. Sample refers to a single feature vector describing a trackster or pair of tracksters.*

Model	Parameters	Time per 100 000 samples		Time per 1000 events	
		CPU	GPU	CPU	GPU
MLP	49 535	0.86 s	0.66 s	0.78 s	0.60 s
GNN (node classification)	47 952	1.42 s	0.27 s	1.35 s	0.25 s
GNN (link prediction)	34 544	3.07 s	1.08 s	1.31 s	0.46 s

Chapter 7

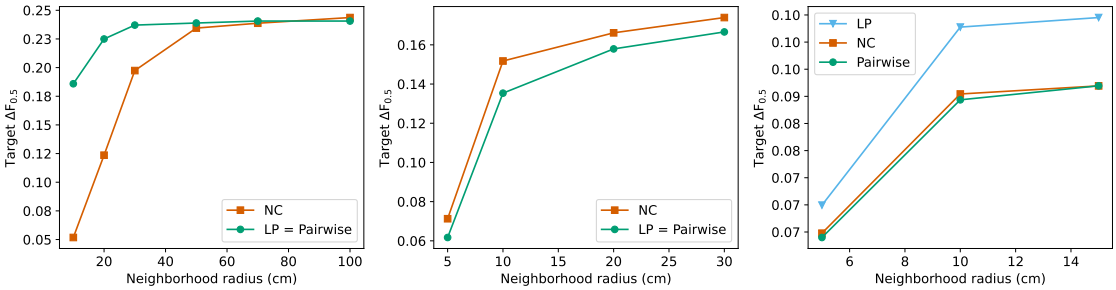
Discussion

This chapter reviews the experiments and results from Chapter 6, discusses the experiment and approach shortcomings, and points out future work directions. The results of experiments on standard MLP models evaluating pairs of tracksters described in Section 6.3 show that learned methods significantly outperform classical clustering methods based on the spatial disposition of the point cloud tracksters. The best clustering results obtained by the Meanshift algorithm (Cheng, 1995), see Table 6.1, were outperformed by the MLP model with a mean $\Delta F_{0.5} = 0.138$ for the two pions case, and $\Delta F_{0.5} = 0.076$ in the ten-particles case, see Table 6.2. With respect to the first research question of comparing classical and learned approaches, the results strongly suggest that learned approaches are viable candidates for improving particle shower reconstruction via calorimetric clustering beyond the classical methods.

Section 6.4 returns to the second research question and evaluates graph neural network approaches in two problem formulations from related work applicable to calorimetric clustering. Intending to extend the scope from an isolated trackster pair towards evaluating a trackster within its environment, it compares the performance of networks based on the ParticleNet (Qu and Gouskos, 2020) architecture adapted to node classification and link prediction task formulations. The results summarized in Table 6.2 suggest that including the trackster neighborhood within the point cloud enables the learned models to infer more information about the reconstructed shower, improving reconstruction performance across all datasets. Both node classification and link prediction formulation benefit the reconstruction performance, the more suitable formulation depending on the dataset.

7.1 Approach considerations

The outputs provided by the pairwise, graph node classification, and graph link prediction approaches differ in event representation complexity, post-processing, and target reconstruction improvement. In this section, we review how the formulation affects the reconstruction process and discuss the benefit and drawbacks of individual approaches. The targeted improvement across the approaches is similar; see Figure 7.1. However, differences in the multi-particle and pile-up cases drive the success of the models trained on these datasets. We identify two critical considerations stemming from the reconstruction approach: the selection problem and the post-processing of the predictions.



(a) Two pions dataset (b) Ten particles dataset (c) Pion in 200PU dataset

Figure 7.1: APPROACH COMPARISON: TARGET IMPROVEMENT. *Improvement in the target $\Delta F_{0.5}$ between the pairwise, node classification (NC), and link prediction (LP) approaches. In the two pions (a) and ten particles case (b), the target of link prediction and the pairwise approach is the same. The link prediction approach permits connections to multiple high-energy tracksters, resulting in a higher target score in the pile-up case (c).*

7.1.1 Selection problem

The high-energy tracksters are selected for smoothing based on the minimum selection energy threshold e_m and shared energy with a simtrackster requirement in the pile-up case. Suppose a simtrackster in a multi-particle event (including two particle cases) is too fragmented, and all of its reconstructed tracksters are below the minimum selection threshold. In that case, none of them will be selected for smoothing. If any of these tracksters is in a neighborhood of a high-energy trackster from a different particle, it might be connected to it, given that it surpasses the minimum threshold. However, the score would have been higher towards the tracksters from its true simtrackster, which were neglected. This effect is less likely in the foreground-background setting, which might explain the better performance of the node classification approach on the multi-particle dataset.

7.1.2 Post-processing of node classification

Other than the node-classification case with two particles, reconstructing the event given the model predictions is not trivial. Multi-particle and pile-up scenarios require post-processing of the overlapping foreground classes in reconstruction. We chose to connect predicted groups of tracksters (super-tracksters) formed by the foreground classes that share at least 50% of their energy (in either direction). If two overlapping super-tracksters share less than 50% of their energy, the conflicting tracksters are assigned to the higher-energetic super-trackster. While this heuristic provides competitive results, the reconstruction could be improved by introducing another layer of a classifier to recognize which overlapping classes should be merged. This issue could be addressed by allowing trackster overlaps in linking and annotating the overlapping classes with respective trackster multiplicities. Improving the conflict resolution process has a substantial impact on the performance of the node classification approach.

7.1.3 Post-processing of link prediction

In the pairwise case, trackster t is connected to the high-energy trackster t' with the highest predicted score $p(t, t')$, from among all the pairs in which t is considered. A single connection per t

is a design choice, as in the pairwise case, the model is unaware of all the other candidate pairs. However, in the link prediction task, the model is processing the whole event at once and might be able to learn to assign t to the correct high-energy trackster. Only considering the best candidate produces the same reconstruction target as the pairwise approach. Conversely, enabling t to connect to multiple high-energy tracksters might improve the reconstruction target, such as the pile-up case shown in Figure 7.1(c). In the two-pion case, considering multiple connections does not have an effect, whereas, in the multi-particle case, it performs worse and is therefore not applied.

Link-prediction in pile-up bias

However, the higher link prediction target score in the pile-up dataset might be due to leaking information about the simtrackster. In pile-up, only high-energy tracksters that share at least half of their energy with a simtrackster are considered for smoothing. Including all these tracksters in a single graph introduces a bias, as high-energy tracksters in the graph are very likely to be from the same particle. Allowing reconstructed tracksters to connect to multiple high-energy tracksters merges the entire sub-graph into a single super-tracksters, which results in better reconstruction performance on the dataset but might not transfer to the production setting.

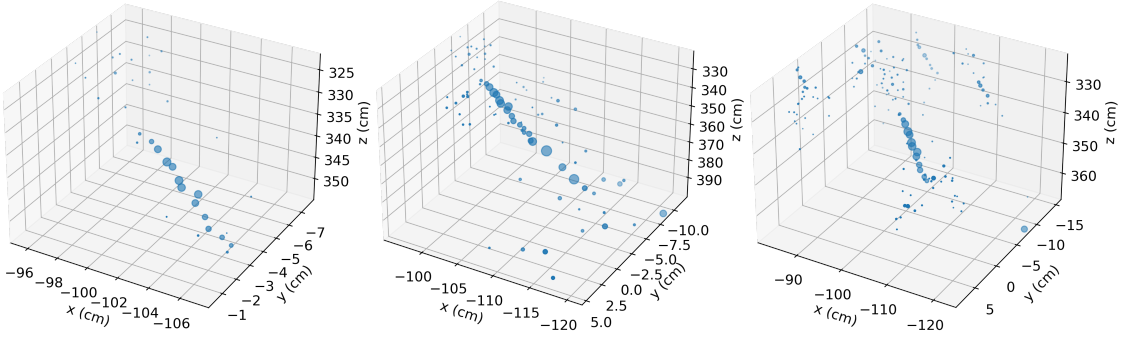
7.1.4 Representation complexity

We describe representation complexity by the number of inputs that must be processed to apply trackster smoothing to all high-energy tracksters in an event. Let H denote the number of high-energy tracksters in the event, C denote the average number of (candidate) tracksters in the neighborhood of a high-energy trackster, and f the number of features describing each trackster. Then, the number of inputs per processed event in the pairwise case can be expressed as $2fHC$. The node classification removes the redundancy in the features of the high-energy tracksters, reducing the number of inputs to $H(f + Cf)$. Link prediction provides the most efficient representation of the event by reducing the redundancy in candidate tracksters shared by multiple high-energy tracksters. The number of inputs can then be expressed as $Hf + Cfm_c^{-1}$, where m_c is the mean number of high-energy tracksters that contain the candidate trackster c in their neighborhood belongs. This number is bounded from above by $T \cdot f$, where T is the total number of tracksters in the event.

Example This effect is most visible in the multi-particle setting, where 1000 ten-particle events produce roughly 2.58 million trackster pairs (with $2f$ features per pair) and 2.62 million nodes in 40 thousand graphs in the node classification setting and only about 250 thousand nodes in 1000 graphs in the link prediction settings (with f features per node). As with the first two approaches, the link prediction approach will still predict about 2.5 million edges, using significantly fewer inputs and providing the lowest representation complexity. The difference between the graph formulations in the two-particle setting is less significant than in the multi-particle case. Node classification may be set to predict the foreground class of the event only, which, with no radius constraints, also bounds the number of inputs from above by $T \cdot f$.

7.1.5 Evaluation gap

We evaluated our experiments on adapted evaluation metrics that enabled us to compare different approaches without fully integrating them into the reconstruction pipeline. First, the set of events we used for the evaluation is relatively small, typically 20 events. The low number of events was



(a) Original trackster
Precision: 0.92 | Recall: 0.12

(b) Smoothed trackster
Precision: 0.99 | Recall: 0.74

(c) Wrongly connected tracksters
Precision: 0.37 | Recall: 0.13

Figure 7.2: INTUITION BEHIND THE EVALUATION METRICS. *Display of a high-energy pion trackster selected for smoothing in the high-pileup dataset. (a) shows the selected trackster, (b) shows the output of the smoothing – connecting to all trackster from the same simtrackster within $r_n = 15\,\mathrm{cm}$ neighborhood. (b) shows the result of connecting to the other tracksters (negative edges) in the $r_n = 15\,\mathrm{cm}$ neighborhood.*

selected due to the high-computational complexity of the B^3 metrics – growing quadratically in the number of layer-clusters, and due to the input ROOT files typically containing 20 events per file. Secondly, the extent of how well these metrics translate to the production environment is unclear. Extrinsically, we evaluated the functionality of the metrics on a set of events, observing that reconnecting with tracksters from the same particle shower (simtrackster) is increasing the recall, see Figure 7.2(b) and connecting with tracksters from other simtracksters is decreasing the precision, see Figure 7.2(c), matching the intuition behind the metrics.

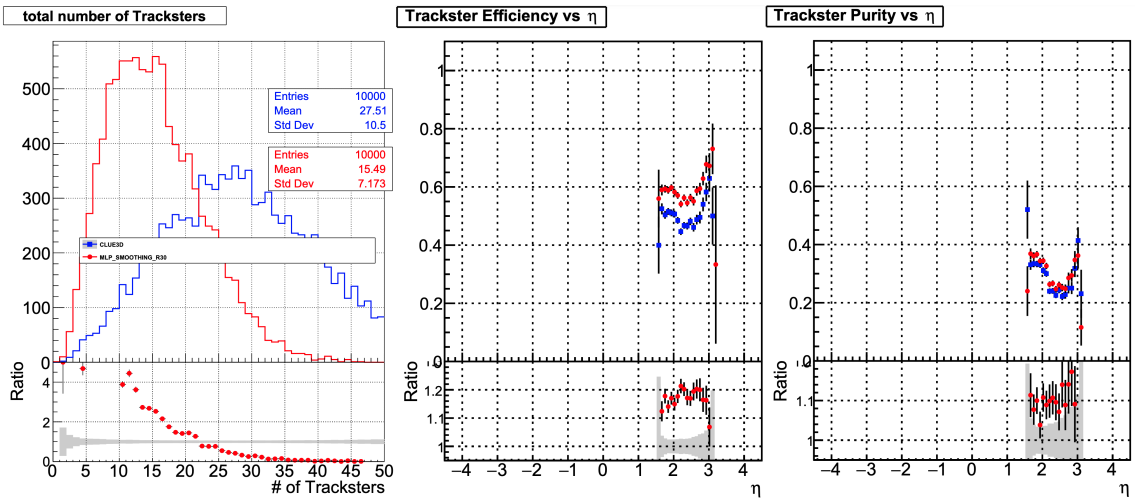
Evaluation in production environment

To test the portability to the production setting, we exported an MLP model trained on the two-pions case into ONNX and implemented the construction of the pairwise features in CMSSW[1]. Figure 7.3 shows a reduction in the mean number of tracksters from 27.51 to 15.49, driving an increase in efficiency, by reconstructing larger parts of the simtracksters, and purity, by reducing the number of small, impure tracksters. The results suggest that the model can reconnect the CLUE3D tracksters deposited by the same simulated particle and that the improvements on B^3 metrics, particularly recall, translate to improvements in efficiency in the production pipeline.

Nevertheless, further examination using other datasets and models is needed to establish the link between the metrics. Moreover, integration with the successive parts of the reconstruction pipeline, particularly linking and particle identification, is crucial for evaluating the benefit of trackster smoothing. However, porting the graph models, particularly their dynamic layers, into CMSSW is difficult due to the limited support of PyTorch Geometric modules in ONNX and the need to re-implement the graph formation process into C++. Furthermore, producing pile-up events is not trivial and requires execution on WLCG[2]. Therefore, further evaluation in the production environment is not in the scope of this work.

[1]Source code: https://github.com/edcuba/cmssw/blob/CMSSW_12_6_0_pre3_MLP_smoothing_with_ntuplizer/RecoHGCal/TICL/plugins/SmoothingAlgoByMLP.cc

[2]The Worldwide LHC Computing Grid: https://wlcg.web.cern.ch



(a) Effect on the number of tracksters (b) Effect on efficiency (higher is better) (c) Effect on purity (higher is better)

Figure 7.3: EVALUATION IN THE PRODUCTION ENVIRONMENT. *The MLP smoothing model applied in production of* 10 000 *two-close-by-pion events with the shower energy between* (10 *and* 600 GeV). *The smoothed trackster are selected by* $e_m = 10\,\text{GeV}$, *and neighborhoods with* $r_n = 30\,\text{cm}$ *are considered. Figure (a) compares the number of tracksters in the input event (produced by CLUE3D, in blue), and the nubmer of tracksters in the smoothing output (in red). Figures (b) and (c) show the efficiency and purity of the input and the smoothing output given the* η *of the trackster, the ratio between them is highlighted in the bottom part of the figure.*

7.2 Model and training considerations

Beyond the approach considerations, we discuss the complexity and training considerations of the learned models. Although the link prediction approach requires processing the least amount of inputs, dynamic graph convolution layers in the graph approach are more computationally expensive than the dense layers in the feed-forward neural networks. On the other hand, convolutional layers have fewer parameters than dense layers.

7.2.1 Dataset size

A sufficient amount of available data supported the two-pion and ten-particle cases. However, pre-processing the datasets was a significant bottleneck for experimentation, as preparing a full-size dataset for training in a particular e_m and r_n configuration for a specific task formulation could take one or two days – with several iterations throughout the development of the work. The pile-up datasets suffered from a low signal-to-data ratio, as a single simtrackster reconstruction in a pile-up of 200 takes up $\approx$5 MB, compared to $\approx$50 kB per a two-pion event. Therefore, generating samples from 8000 single-pion pile-up events requires processing 40 GB of data, leading to smaller datasets. Additionally, issues in pile-up datasets production significantly slowed down the experiments.

7.2.2 Model complexity

While the graph-based approaches provide a more compact representation of the input data, the dynamic graph convolutions and edge prediction represent additional layers of complexity for inference. The MLP model does not use batch normalization, and the inference can be parallelized by using large batch sizes (such as 256 or 512). The effect of batch size dramatically influences the inference time. However, it might impact the prediction quality in graph networks due to batch normalization layers in the EdgeConv blocks. The impact of batch size on the prediction quality of the graph approaches was not evaluated.

For execution on the CPU, the pairwise approach is the best-performing option, both in terms of the sample (trackster or trackster pair feature vector) and event throughput. The node classification approach performs best on a GPU, requiring only $0.25\,\mathrm{s}$ to infer predictions for reconstructing 1000 pions in a pile-up environment. On the other hand, the link-prediction approach requires $0.46\,\mathrm{s}$ per 1000 pion events, which is likely to be driven by the additional complexity of constructing trackster pairs in the network.

Training convergence

The smaller pile-up datasets limited the size of the considered models regarding the number of learnable parameters. However, the graph models, mainly the link prediction case, were susceptible to the model configuration, and many training attempts did not converge. Specifically, using Layer Normalization before the first EdgeConv block, extending the dimensions of the first blocks, and increasing the number of EdgeConv blocks often resulted in failed training.

Hyperparameter optimization

This work focuses on identifying feasible methods and problem formulations rather than fine-tuning a particular approach. Due to the vast search space, we strived to balance the conditions in dataset size and model parameters across the approaches. However, the performance of the individual data-driven models could be further improved by hyper-parameter optimization (Wulff et al., 2023), extending the dataset size and allowing for more parameters.

7.2.3 Graph construction

The edges between the tracksters in the point clouds were constructed dynamically, based on k-nearest neighbors in the feature space. Constructing the nearest neighbors in high-dimensional spaces might be counter-productive and contribute to the abovementioned training convergence issues. Constructing the edges of the first EdgeConv layer in the coordinate space (polar or Cartesian) such as proposed in ParticleNet by Qu and Gouskos (2020), applying a spatial transformation of the features such as in the original DGCNN design (Wang et al., 2019), or projecting the input features into the coordinate and feature spaces separately, as in GravNet Qasim et al. (2019) would likely be beneficial for the training stability and prediction performance. These adaptations should be considered when fine-tuning a task-specific network in future work.

Using a pre-constructed static graph, as suggested by Ju et al. (2020), was also considered. However, early trials did not produce competitive results to the dynamic approach with the same choice of k. A fixed graph will likely require more nearest neighbors, as the connections are not updated in each step. Stacking several EdgeConv blocks allows networks with lower k to reach more distant neighbors. While the performance impact of using the dynamically constructed edges was initially a concern, results shown in Table 6.3 suggest that this is not a problem, even with 64-dimensional feature spaces.

7.3 Alternative approaches

Beyond architectures considered in this work, alternatives such as transformer models or interaction networks could be considered. Transformer models, applied to a similar related task of jet tagging by Qu et al. (2022), were shown to outperform ParticleNet. However, due to significantly more trainable parameters, an order of magnitude larger datasets than those used in this work are required to take advantage of the architecture fully. On the other hand, interaction networks, such as proposed by Moreno et al. (2020) in JEDI-net, could be used as a simple extension of the pairwise approach, introducing a context of other pairs in the batch without performing the convolution operations. Furthermore, representation learning methods such as graph auto-encoders (Kipf and Welling, 2016) could be explored to use the layer-cluster-level information to describe the tracksters.

Additionally, the current trackster representation neglects the timing information. Exploring the propagation of particle showers through time could provide a significant clue for distinguishing tracksters produced by different particles and guide the reconstruction of individual simtracksters. Sparse methods for processing 3D video, such as spatiotemporal convolutional neural networks (Choy et al., 2019) could be a feasible approach to include the timing information directly.

Lastly, other than the model architecture, alternative loss functions, such as object condensation loss (Kieseler, 2020) could be considered to formulate the calorimetric clustering problem directly on multiple particles as a learning problem and avoid the difficulties in applying the model predictions to the reconstruction.

Chapter 8

Conclusion

The goal of this work was to identify effective AI approaches for improving particle shower reconstructions in the calorimetric clustering setting. Based on experiments conducted on datasets of two pions, ten particles, and a single pion in a pile-up environment, evaluating the quality of event reconstruction using B^3 metrics described in Section 5.1, it can be concluded that machine-learned approaches can significantly improve the reconstruction quality beyond the baselines established by classical clustering methods.

The pairwise MLP-based approach provides competitive reconstruction quality at the cost of quadratic growth in the number of tracksters and a higher event run-time. While the MLP model achieved competitive results, graph neural networks provide a more compact representation of the data and show better run-time performance and reconstruction quality. A reconstruction quality and run-time performance comparison between the pairwise, graph node classification, and graph link prediction shows that the graph-based methods are more effective and efficient in reconstructing single particle showers under pile-up conditions, which is the primary use case in the production setting. Compared to the pairwise approach, graph models can process entire events or trackster neighborhoods at once, considering the context information in the predictions. Furthermore, the more concise graph-based data representation leads to better scalability of the models.

A GravNet-inspired node classification approach with an adapted energy-weighted mean squared error loss function delivers the best reconstruction results in multi-particle datasets. Additionally, the more straightforward inference mechanism compared to the link prediction provides a higher throughput on pions in a pile-up environment. However, the binary node classification setting exhibits a high graph-node redundancy in multi-particle events.

Like the node classification, the link prediction models use a DGCNN-like architecture and dynamically updated edge convolutions to encode node information in the context of their neighborhoods into a latent space. However, instead of directly inferring predictions for nodes, the link prediction models combine the latent node features to form edge representations and predict the probabilities of the edge's existence. Due to the high negative edge ratio, a Quality Focal Loss function is used in training, reaching the best reconstruction performance on the pile-up dataset.

8.1 Future work

Due to representing the entire event in a single graph, the link-prediction results in a pile-up environment might be biased by leaking information about the simtrackster. This effect needs to be further investigated in the production setting.

Furthermore, while the meta-task of improving the reconstruction given the B^3 metrics translates to improvements in efficiency and purity in the production environment, as discussed in Sec-

tion 7.1.5, a more detailed evaluation and integration with the rest of the reconstruction pipeline are needed. Additionally, larger datasets and extensive hyperparameter optimization would be required to establish a fair comparison between the models.

Finally, while dynamic convolutional layers with k-nearest neighbors in the feature space produce promising results, separating the coordinate space, at least in the first EdgeConv block, as discussed in Section 7.2.3, would likely lead to better training stability, run-time, and physics results. Therefore, it should be considered in future work and production deployment.

8.2 Final remarks

Machine-learning-based AI approaches can be successfully deployed to improve the calorimetric reconstructions, providing a significant benefit over classical clustering methods. The experimental results suggest that learned models can improve the clustering recall while maintaining the precision of the clusters. Depending on the problem formulation, the link-prediction problem formulation is computationally best suited for processing events with a high number of tracksters with overlapping neighborhoods, while the node classification is suitable for reconstructions formulated as recovering individual particles from noise.

Regardless of the formulation, graph neural networks have the potential to achieve better clustering quality, run-time characteristics on GPU, and more compact data representation leading to better scalability toward large data volumes.

Appendix A

Attachments

A.1 Photon shower fragmentation analysis

Figures A.1, A.2, and A.3 show the photon case of the exploratory data analysis described in Section 4.3. Figure A.4 shows the same-particle trackster distances for photons, as discussed for pions in Section 5.3.1.

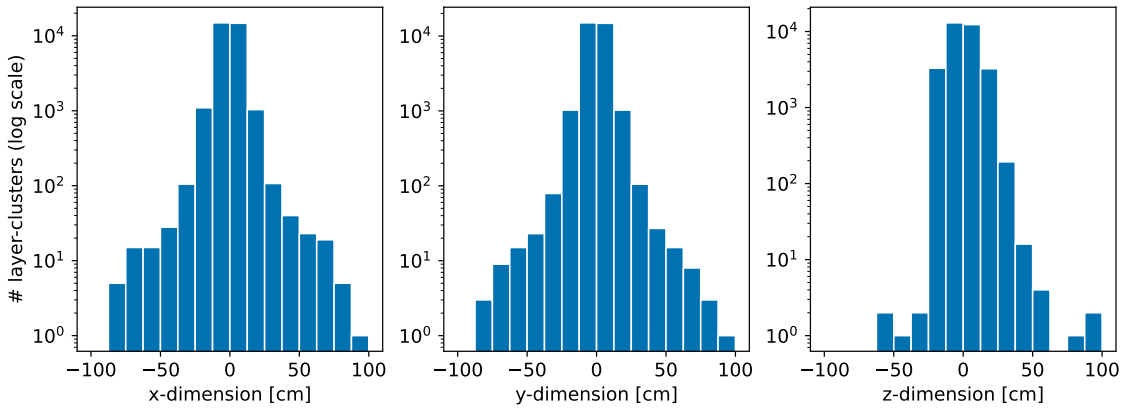
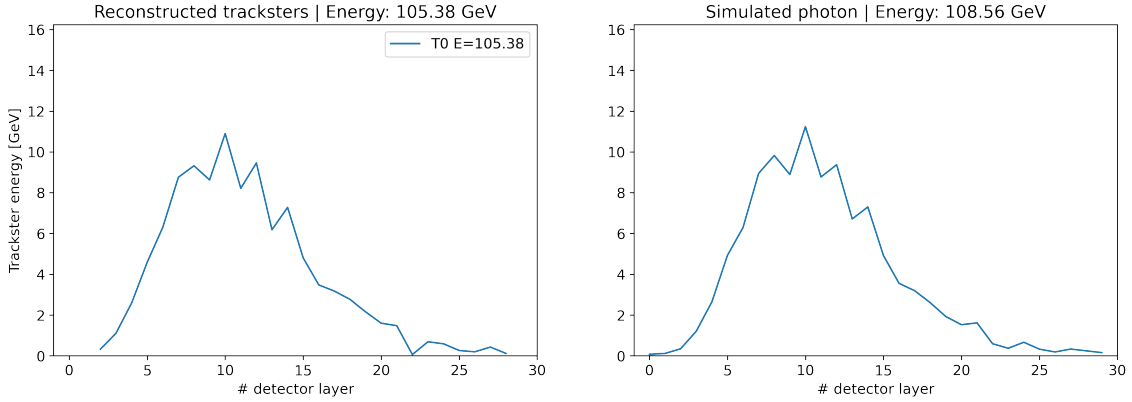
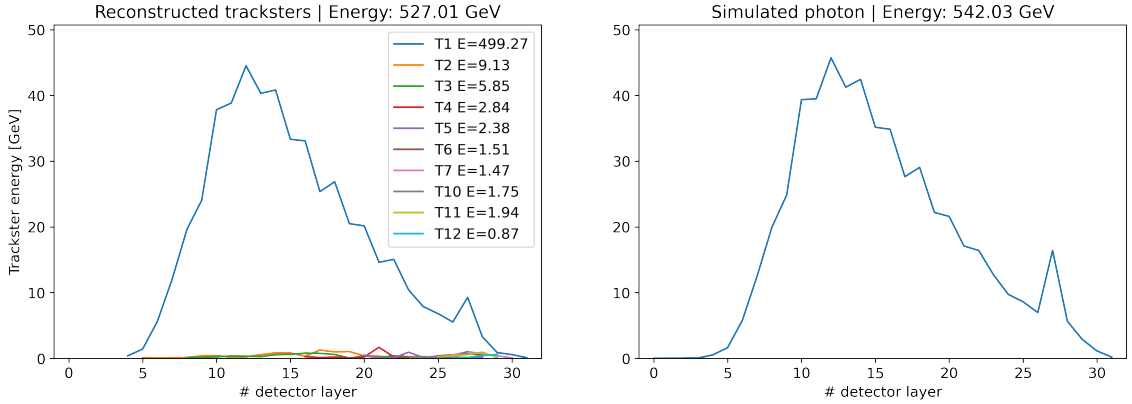


Figure A.1: LAYER-CLUSTER DISTRIBUTION IN PHOTON SIMTRACKSTERS. *This figure describes the distribution of layer-clusters with respect to the simtrackster barycenters of 100 simulated photons along x, y, and z axis. The number of layer-clusters is shown in the logarithmic scale. Deviations along the x and y axes are symmetric, with most of the layer-clusters concentrated around the barycenter, while the z-axis shows a longer tail.*



(a) Lower energy photon shower (108.56 GeV)



(b) Higher energy photon shower (542.03 GeV)

Figure A.2: LONGITUDINAL ENERGY PROFILES OF PHOTON SHOWERS. *This figure captures the energy fragmentation of photon showers along the detector layers. The plots are computed by summing up layer-cluster energies of the reconstructed photon tracksters (left) associated with the photon simtrackster (right) in each detector layer. Figure (a) shows a relatively low-energetic photon shower with no fragmentation. Figure (b) shows a higher-energetic photon shower. The reconstruction of the high-energetic shower is fragmented into a single high-energy trackster and multiple low-energy tracksters, with an overlap with the main trackster on the z-axis.*

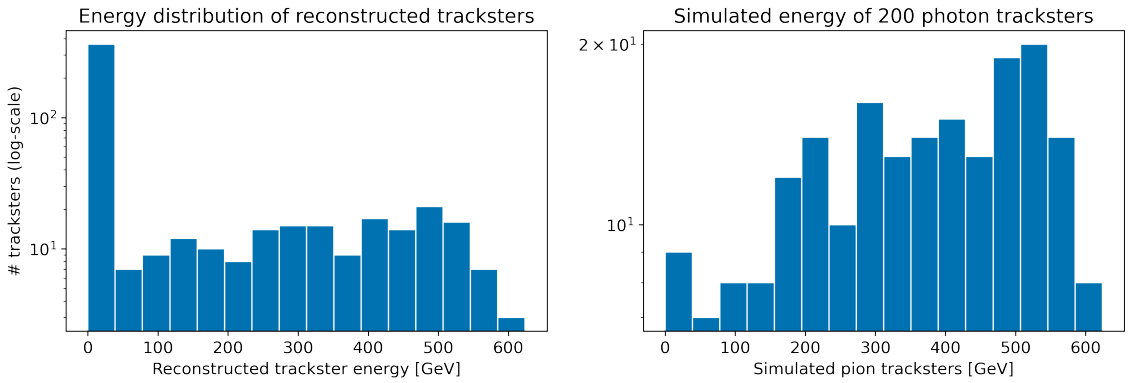


Figure A.3: ENERGY DISTRIBUTION OF PHOTON TRACKSTERS. *This figure shows the energy distribution of reconstructed photon tracksters and their corresponding simtracksters. Reconstructed tracksters show a peak in the low-energy range caused by the low-energetic trackster fragments.*

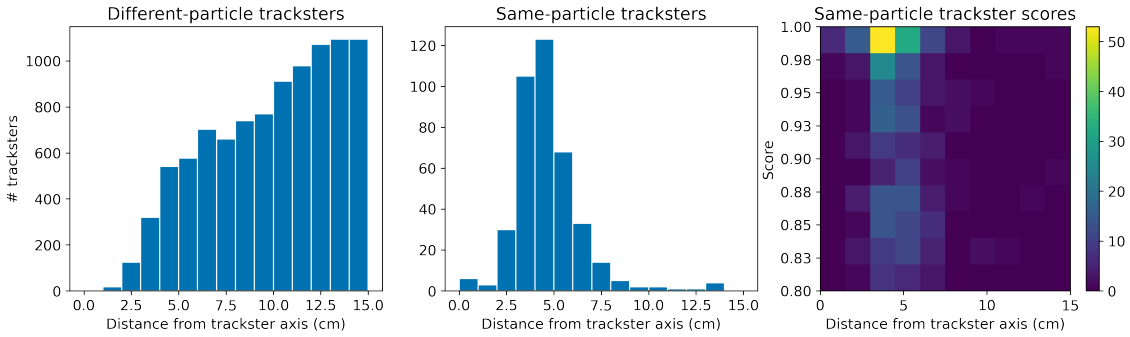


Figure A.4: NEIGHBORHOOD OF PHOTON TRACKSTERS. *This figure shows the distances between a high-energy photon trackster in a 200-pile-up environment and other tracksters in the neighborhood. The neighborhood is defined by the distance from the trackster axis. The first two figures compare the distance to the tracksters of the same simulated particle and the distance to tracksters of other simulated particles. The right-most figure relates the distance and the recoToSim associator score of the tracksters.*

A.2 Model and dataset details

Table A.1 provides supplemental information to models and dataset discussed in Section 6.

Table A.1: MODEL AND DATASET SETUP IN EXPERIMENTS. *In all approaches, 10% of the training samples were kept for validation. ParticleNet refers to adaptations of the architecture suggested by Qu and Gouskos (2020), as described in Section 6.4. A decision threshold of 0.5 is applied for computing the validation ROC AUC and the ratio of positive and negative samples (Pos. / Neg.) in the datasets.*

Dataset	2 pions	10 particles	Pion in 200 PU
MLP			
Architecture	LayerNorm, 63×256, Sigmoid, 256×128, Sigmoid, 128×1		
Events	20 000	1000	8000
Selection (e_m, r_n)	(10 GeV, 50 cm)	(10 GeV, 30 cm)	(5 GeV, 15 cm)
Pairs (Pos. / Neg.)	2 119 563 (0.57)	2 582 842 (0.17)	729 995 (0.42)
Training epochs	50	100	100
Validation ROC AUC	0.937	0.928	0.928
GNN (node classification)			
Architecture	ParticleNet (64^2)	ParticleNet (64^3)	ParticleNet (64^2)
Events	20 000	1000	8000
Selection (e_m, r_n)	(10 GeV, $\varnothing$)	(10 GeV, 30 cm)	(5 GeV, 15 cm)
Graphs	49 767	39 548	28 123
Nodes (Pos. / Neg.)	1 515 444 (0.56)	2 622 390 (0.18)	758 118 (0.45)
Training epochs	100	100	100
Validation ROC AUC	0.963	0.924	0.936
GNN (link prediction)			
Architecture	ParticleNet (32×64)	ParticleNet (64^3)	ParticleNet (32×64)
Events	20 000	5000	8000
Selection (e_m, r_n)	(10 GeV, $\varnothing$)	(10 GeV, 30 cm)	(5 GeV, 15 cm)
Graphs	19 968	5000	7569
Nodes	563 041	1 272 368	341 008
Edges (Pos. / Neg.)	2 421 912 (0.55)	13 283 054 (0.18)	729 995 (0.42)
Training epochs	100	100	100
Validation ROC AUC	0.959	0.943	0.933

List of Figures

List of Tables

Bibliography

Achlioptas, P., Diamanti, O., Mitliagkas, I., and Guibas, L. (2018). Learning representations and generative models for 3D point clouds. In Dy, J. and Krause, A., editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 40–49. PMLR.

Agostinelli, S. et al. (2003). GEANT4–a simulation toolkit. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 506(3):250–303.

Amigó, E., Gonzalo, J., Artiles, J., and Verdejo, F. (2009). A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*, 12(5):613–613.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bagga, A. and Baldwin, B. (1998). Algorithms for scoring coreference chains. In *The first international conference on language resources and evaluation workshop on linguistics coreference*, volume 1, pages 563–566. Citeseer.

Battaglia, P., Pascanu, R., Lai, M., Rezende, D. J., and Kavukcuoglu, K. (2016). Interaction networks for learning about objects, relations and physics. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 4509–4517, Red Hook, NY, USA. Curran Associates Inc.

Blomer, J., Ganis, G., Hardi, N., and Popescu, R. (2017). Delivering LHC Software to HPC Compute Elements with CernVM-FS. In Kunkel, J. M., Yokota, R., Taufer, M., and Shalf, J., editors, *High Performance Computing*, pages 724–730, Cham. Springer International Publishing.

Bradley, A. P. (1997). The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159.

Brüning, O. and Rossi, L. (2015). *The High Luminosity Large Hadron Collider*. WORLD SCIENTIFIC.

Cheng, Y. (1995). Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799.

Choy, C., Gwak, J., and Savarese, S. (2019). 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3070–3079.

CMS Collaboration (2009). Particle-Flow Event Reconstruction in CMS and Performance for Jets, Taus, and MET. Technical report, CERN, Geneva.

CMS Collaboration (2017). The Phase-2 Upgrade of the CMS Endcap Calorimeter. Technical report, CERN, Geneva.

CMS Collaboration (2021). The Phase-2 Upgrade of the CMS Data Acquisition and High Level Trigger. Technical report, CERN, Geneva.

CMS Collaboration (2022). The TICL (v4) reconstruction at the CMS Phase-2 High Granularity Calorimeter Endcap. Technical report, CERN.

CMS Offline Software and Computing (2022). CMS Phase-2 Computing Model: Update Document. Technical report, CERN, Geneva.

Comaniciu, D. and Meer, P. (2002). Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619.

Duarte, J. and Vlimant, J.-R. (2022a). *Graph Neural Networks for Particle Tracking and Reconstruction*, chapter Chapter 12, pages 387–436. WORLD SCIENTIFIC.

Duarte, J. and Vlimant, J.-R. (2022b). Graph neural networks for particle tracking and reconstruction. In *Artificial Intelligence for High Energy Physics*, pages 387–436. WORLD SCIENTIFIC.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press.

Fabjan, C. W. and Gianotti, F. (2003). Calorimetry for particle physics. *Rev. Mod. Phys.*, 75:1243–1286.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, ICML'17, page 1263–1272. JMLR.org.

Hamilton, W. L. (2020). Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(3):1–159.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 448–456. JMLR.org.

Jaderberg, M., Simonyan, K., Zisserman, A., and Kavukcuoglu, K. (2015). Spatial transformer networks. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Ju, X. et al. (2020). Graph neural networks for particle reconstruction in high energy physics detectors. In *33rd Annual Conference on Neural Information Processing Systems*.

Kaiser, M. (2008). Mean clustering coefficients: the role of isolated nodes and leafs on clustering measures for small-world networks. *New Journal of Physics*, 10(8):083042.

Kieseler, J. (2020). Object condensation: one-stage grid-free multi-object reconstruction in physics detectors, graph, and image data. *The European Physical Journal C*, 80(9).

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kipf, T. N. and Welling, M. (2016). Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.

Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Kriege, N. M., Johansson, F. D., and Morris, C. (2020). A survey on graph kernels. *Applied Network Science*, 5(1).

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Li, C., Gunther, M., and Boult, T. E. (2018a). Eclipse: Ensembles of centroids leveraging iteratively processed spatial eclipse clustering. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 131–140.

Li, X., Lv, C., Wang, W., Li, G., Yang, L., and Yang, J. (2023). Generalized focal loss: Towards efficient representation learning for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3139–3153.

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., and Chen, B. (2018b). PointCNN: Convolution On X-Transformed Points. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Liben-Nowell, D. and Kleinberg, J. (2003). The link prediction problem for social networks. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03, page 556–559, New York, NY, USA. Association for Computing Machinery.

Lin, T., Goyal, P., Girshick, R., He, K., and Dollar, P. (2017). Focal loss for dense object detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, Los Alamitos, CA, USA. IEEE Computer Society.

Lobanov, A. (2019). The CMS HGCAL detector for HL-LHC upgrade. *PoS*, ICHEP2018:086.

Lopienska, E. (2022). The CERN accelerator complex, layout in 2022. Complexe des accélérateurs du CERN en janvier 2022. Technical report, CERN. General Photo.

Loshchilov, I. and Hutter, F. (2017). SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations*.

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press.

Martelli, A. (2017). The CMS HGCAL detector for HL-LHC upgrade. In *5th Large Hadron Collider Physics Conference*.

Maturana, D. and Scherer, S. (2015). VoxNet: A 3D Convolutional Neural Network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 922–928.

McInnes, L., Healy, J., and Astels, S. (2017). hdbscan: Hierarchical density based clustering. *The Journal of Open Source Software*, 2(11):205.

Meilă, M. (2007). Comparing clusterings — an information based distance. *Journal of Multivariate Analysis*, 98(5):873–895.

Mikuni, V. and Canelli, F. (2021). Point cloud transformers applied to collider physics. *Machine Learning: Science and Technology*, 2(3):035027.

Miniere, X., Pinçon, J.-L., and Lefeuvre, F. (1996). A neural network approach to the classification of electron and proton whistlers. *Journal of Atmospheric and Terrestrial Physics*, 58(7):911–924. Geomagnetic Storms: Their Origin, Mechanism and Ionospheric/Atmospheric Effects.

Moreno, E. A., Cerri, O., Duarte, J. M., Newman, H. B., Nguyen, T. Q., Periwal, A., Pierini, M., Serikova, A., Spiropulu, M., and Vlimant, J.-R. (2020). JEDI-net: a jet identification algorithm based on interaction networks. *The European Physical Journal C*, 80(1).

Nandi, A. (2022). New techniques for reconstruction in the CMS High Granularity Calorimeter. Master's thesis, RWTH Aachen University.

Niedermayer, G. P. (2017). Investigations of Calorimeter Clustering in ATLAS using Machine Learning. Master's thesis, University of Ottawa.

Ochando, C. (2017). HGCAL: A High-Granularity Calorimeter for the endcaps of CMS at HL-LHC. *J. Phys.: Conf. Ser.*, 928:012025. 4 p.

Pantaleo, F. (2017). *New Track Seeding Techniques for the CMS Experiment*. PhD thesis, "Universität Hamburg".

Pantaleo, F. and Rovere, M. (2022). The Iterative Clustering framework for the CMS HGCAL Reconstruction. Technical report, CERN, Geneva.

Pata, J., Duarte, J., Vlimant, J.-R., Pierini, M., and Spiropulu, M. (2021). MLPF: efficient machine-learned particle-flow reconstruction using graph neural networks. *The European Physical Journal C*, 81(5).

Pilato, A. D., Chen, Z., Pantaleo, F., and Rovere, M. (2020). Reconstruction in an imaging calorimeter for HL-LHC. *Journal of Instrumentation*, 15(06):C06023–C06023.

Pomarol, A. (2012). Beyond the Standard Model. In *2010 European School of High Energy Physics*, pages 115–151.

Qasim, S. R., Kieseler, J., Iiyama, Y., and Pierini, M. (2019). Learning representations of irregular particle-detector geometry with distance-weighted graph networks. *The European Physical Journal C*, 79(608).

Qi, C. R., Su, H., Kaichun, M., and Guibas, L. J. (2017a). PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85.

Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017b). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 5105–5114, Red Hook, NY, USA. Curran Associates Inc.

Qu, H. and Gouskos, L. (2020). Jet tagging via particle clouds. *Phys. Rev. D*, 101:056019.

Qu, H., Li, C., and Qian, S. (2022). Particle transformer for jet tagging. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S., editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 18281–18292. PMLR.

Rapacz, K. (2017). Calorimeter endcap (CE) parameter drawings. Technical Report CMS-CE-ES-0017, CERN.

Rasmussen, C. E. (1999). The infinite gaussian mixture model. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, page 554–560, Cambridge, MA, USA. MIT Press.

Rovere, M., Chen, Z., Di Pilato, A., Pantaleo, F., and Seez, C. (2020). CLUE: A fast parallel clustering algorithm for high granularity calorimeters in high-energy physics. *Frontiers in Big Data*, 3.

Rubner, Y., Tomasi, C., and Guibas, L. J. (2000). The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121.

Sakuma, T. and McCauley, T. (2014). Detector and Event Visualization with SketchUp at the CMS Experiment. *Journal of Physics: Conference Series*, 513(2):022032.

Thomas, H., Qi, C. R., Deschaud, J., Marcotegui, B., Goulette, F., and Guibas, L. (2019). KPConv: Flexible and Deformable Convolution for Point Clouds. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6410–6419, Los Alamitos, CA, USA. IEEE Computer Society.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. (2018). Graph attention networks. *International Conference on Learning Representations*.

Wang, Y., Sun, Y., Liu, Z., Sarma, S. E., Bronstein, M. M., and Solomon, J. M. (2019). Dynamic Graph CNN for Learning on Point Clouds. *ACM Trans. Graph.*, 38(5).

Ward, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. (2021). A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015). 3D ShapeNets: A deep representation for volumetric shapes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, Los Alamitos, CA, USA. IEEE Computer Society.

Wulff, E., Girone, M., and Pata, J. (2023). Hyperparameter optimization of data-driven AI models on HPC systems. *J. Phys. Conf. Ser.*, 2438(1):012092.

Xu, D. and Tian, Y. (2015). A comprehensive survey of clustering algorithms. *Annals of Data Science*, 2(2):165–193.

Xu, W., Liu, X., and Gong, Y. (2003). Document clustering based on non-negative matrix factorization. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR '03, page 267–273, New York, NY, USA. Association for Computing Machinery.

Zhang, M. and Chen, Y. (2018). Link prediction based on graph neural networks. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Zhang, M., Li, P., Xia, Y., Wang, K., and Jin, L. (2021). Labeling trick: A theory of using graph neural networks for multi-node representation learning. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*, volume 34, pages 9061–9073. Curran Associates, Inc.

Zhao, H., Jiang, L., Jia, J., Torr, P., and Koltun, V. (2021). Point transformer. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 16239–16248.