أكاديمية كاوست
KAUST ACADEMY

جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

Sheaf Neural Networks for Pattern Recognition in Particle Detectors

Dimah Alotaibi, Felice Pantaleo

dimah.alotaibi@kaust.edu.sa, felice.pantaleo@cern.ch

Abstract

Modern particle detectors generate enormous amounts of high-dimensional data by recording the energy deposited by particles traversing the detector. A fundamental challenge is identifying which detector signals originate from the same particle, particularly in the high-density collision environment expected at the High-Luminosity Large Hadron Collider (HL-LHC). This problem can be formulated as a graph-based pattern recognition task, where reconstructed detector objects are linked to recover complete particle showers. This project investigates the use of Sheaf Neural Networks (SNNs) for the Trackster Linking task in the CMS High-Granularity Calorimeter (HGCAL). By replacing uniform graph message passing with learned edge-specific transformations, SNNs provide a more flexible mechanism for information exchange between neighboring Tracksters, particularly in heterophilic graph regions. The proposed model was evaluated against a baseline Graph Neural Network under identical training conditions and demonstrated improved class separation while maintaining strong edge classification performance, highlighting the potential of sheaf-based message passing for particle reconstruction.

Date: August 24, 2026

Contents

1 Introduction **2**

1.1 Problem Statement . . . 2

1.2 Key Assumptions and Limitations . . . 2

1.3 Summary of the Proposed System . . . 2

2 Background **3**

2.1 CERN Experimental Environment . . . 3

2.2 Graph Neural Networks (GNN) . . . 4

2.3 Dataset . . . 6

3 Methodology **7**

3.1 Dataset Generation and Graph Construction Pipeline . . . 7

3.2 Graph Construction . . . 7

3.3 Sheaf Neural Network Architecture . . . 8

3.4 Restriction Map Parameterization . . . 8

3.5 Experimental Configuration . . . 9

3.6 Training Procedure . . . 10

3.7 Evaluation . . . 10

4 Results **10**

4.1 Overall Classification Performance . . . 11

4.2 Receiver Operating Characteristic Analysis . . . 11

4.3 Prediction Score Distribution . . . 12

4.4 Performance Across Pseudorapidity . . . 13

4.5 Comparison with the Baseline Graph Neural Network . . . 14

4.6 Discussion . . . 15

4.7 Limitations and Future Work . . . 15

4.8 Conclusion . . . 16

References **16**

1 Introduction

The High-Luminosity Large Hadron Collider (HL-LHC) [1] is a major upgrade of the Large Hadron Collider designed to increase the instantaneous luminosity by approximately an order of magnitude while maintaining proton–proton collisions at a center-of-mass energy of 14 TeV. Although this significantly enhances the physics potential of the experiment, it also results in severe pileup, where nearly 200 proton–proton interactions may occur during a single bunch crossing.

To operate efficiently in this challenging environment, the CMS High-Granularity Calorimeter (HGCAL) [2] has been designed with unprecedented spatial and temporal granularity. The detector records millions of energy deposits from overlapping particle showers, producing large, high-dimensional datasets. Reconstructing the trajectories of the original particles from these detector signals has therefore become a challenging large-scale pattern recognition problem that requires advanced machine learning techniques.

1.1 Problem Statement

Within the CMS reconstruction chain, energy deposits are first grouped into three-dimensional clusters known as *Tracksters*, which represent candidate particle showers. The Trackster Linking task aims to determine whether neighboring Tracksters originate from the same underlying particle, enabling complete particle showers to be reconstructed.

Graph Neural Networks (GNNs) [3] have become an effective solution for this task because they naturally model Tracksters as nodes and their candidate connections as graph edges. However, conventional message-passing GNNs typically apply the same aggregation mechanism across all graph edges. In dense detector environments, neighboring Tracksters frequently originate from different particles, making this uniform message-passing strategy less effective and increasing the difficulty of distinguishing true particle connections from background-induced links.

1.2 Key Assumptions and Limitations

This work focuses on two important limitations commonly encountered by conventional Graph Neural Networks when applied to detector data:

- **Oversmoothing:** As additional message-passing layers are stacked, node representations gradually become more similar [4]. This reduces the ability of the network to distinguish different particle showers and can degrade classification performance in dense detector environments.
- **Heterophily:** Many Graph Neural Networks implicitly assume *homophily*, where neighboring nodes are expected to have similar representations. However, detector graphs are often highly heterophilic, since neighboring Tracksters may originate from different particles. This violates the homophily assumption and limits the effectiveness of standard message passing [5].

1.3 Summary of the Proposed System

To address these limitations, this work investigates a Topological Deep Learning approach based on Sheaf Neural Networks (SNNs) [6]. Unlike conventional Graph Neural Networks that apply uniform message passing across all graph edges, Sheaf Neural Networks learn edge-specific restriction maps that adapt information exchange between neighboring Tracksters.

The proposed model embeds Trackster features into sheaf feature spaces and performs sheaf diffusion through learned restriction maps while preserving the existing Trackster Linking

pipeline. To evaluate the impact of adaptive message passing, the proposed Sheaf Neural Network is compared with a baseline Graph Neural Network using identical datasets, training settings, and evaluation procedures.

2 Background

This chapter introduces the background concepts required to understand the Trackster Linking problem and the proposed Sheaf Neural Network approach. It first presents the CERN experimental environment, including the Large Hadron Collider (LHC), the High-Luminosity LHC (HL-LHC), the CMS detector, and the Tracksters Iterative Clustering (TICL) framework. The chapter then reviews Graph Neural Networks (GNNs) and their limitations before introducing Sheaf Neural Networks (SNNs), which form the basis of the proposed approach.

2.1 CERN Experimental Environment

This section provides an overview of the CERN experimental environment and the detector reconstruction framework used throughout this project.[1]

2.1.1 Large Hadron Collider (LHC)

The Large Hadron Collider (LHC) [7] is the world's largest and most powerful particle accelerator. Located at CERN near Geneva, Switzerland, it consists of a 27 km circular tunnel in which two counter-rotating proton beams are accelerated to nearly the speed of light before colliding at four interaction points occupied by the ATLAS, CMS, ALICE, and LHCb experiments.

Proton bunches cross approximately 40 million times per second (40 MHz), producing an enormous number of particle interactions. These collisions allow physicists to study the fundamental constituents of matter, perform precision measurements of the Standard Model, and search for physics beyond the Standard Model.

During each bunch crossing, multiple proton–proton interactions occur simultaneously. While one interaction typically contains the physics process of interest, the remaining interactions contribute additional detector activity known as *pileup*, significantly increasing the complexity of event reconstruction.

2.1.2 High-Luminosity Large Hadron Collider (HL-LHC)

The High-Luminosity Large Hadron Collider (HL-LHC) [1] is the next major upgrade of the LHC. Although it will continue operating at a center-of-mass energy of 14 TeV, it will increase the instantaneous luminosity by approximately an order of magnitude, enabling the collection of substantially larger datasets throughout its operational lifetime.

This increase in luminosity comes at the cost of significantly higher event complexity. The HL-LHC is expected to produce up to 200 simultaneous proton–proton interactions during a single bunch crossing, dramatically increasing detector occupancy, data volume, and computational requirements for event reconstruction. Consequently, advanced reconstruction algorithms and machine learning techniques have become essential components of the CMS reconstruction chain.

2.1.3 Compact Muon Solenoid (CMS)

The Compact Muon Solenoid (CMS) [8] is one of the two general-purpose detectors operating at the LHC. It is designed to measure the trajectories, energies, and identities of particles produced

[1] General descriptions of the CERN experimental environment are based on introductory lectures and educational material provided during the CERN Summer Student Programme. Technical details are referenced from the corresponding official publications.

in proton–proton collisions, enabling a broad range of physics analyses.

The CMS detector consists of multiple sub-detectors that measure complementary particle properties, including the silicon tracker, electromagnetic and hadronic calorimeters, and muon detection system. Information collected from these detector components is combined using the Particle Flow (PF) reconstruction algorithm, which reconstructs individual particles by integrating measurements from all detector subsystems.

To maintain reconstruction performance under HL-LHC conditions, CMS is upgrading its endcap calorimeters with the High-Granularity Calorimeter (HGCAL) [2]. HGCAL provides unprecedented spatial and longitudinal segmentation, producing detailed three-dimensional measurements of particle showers while introducing new computational challenges for event reconstruction.

2.1.4 Tracksters Iterative CLustering (TICL)

The Tracksters Iterative CLustering (TICL) framework is the reconstruction framework developed for the CMS High-Granularity Calorimeter. Its objective is to reconstruct particle showers by grouping detector energy deposits into three-dimensional objects called *Tracksters*.

The reconstruction pipeline consists of multiple stages, including two-dimensional clustering, three-dimensional clustering, particle identification, Trackster linking, and finally integration into the Particle Flow (PF) reconstruction chain. Because TICL is modular, individual reconstruction components can be replaced or extended without modifying the complete reconstruction workflow.

This project focuses on the **Trackster Linking** stage, where Graph Neural Networks are used to determine whether neighboring Tracksters originate from the same underlying particle.

2.2 Graph Neural Networks (GNN)

Graph Neural Networks (GNNs) [3] are deep learning models designed to learn from graph-structured data by representing objects as nodes and their relationships as edges. A graph is formally defined as $G = (V, E)$, where V is the set of nodes and E is the set of edges connecting them. Each node is associated with a feature vector, forming the node feature matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$.

Unlike conventional neural networks, GNNs exploit the graph topology through an iterative *message passing* process. During each layer, a node aggregates information from its neighboring nodes and updates its representation according to

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

where $\mathcal{N}(v)$ denotes the neighborhood of node v. Different GNN architectures differ primarily in how they define the aggregation and update functions.

2.2.1 Graph Convolutional Networks (GCN)

Graph Convolutional Networks (GCNs) [9] extend the concept of convolution to graph-structured data by aggregating neighboring node features using a normalized adjacency matrix. The propagation rule of a GCN layer is given by

$$\mathbf{H}^{(l+1)} = \sigma\left(\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\mathbf{H}^{(l)}\mathbf{W}^{(l)}\right), \tag{2}$$

where $\mathbf{A} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix with self-loops, $\mathbf{D}$ is the corresponding degree matrix, $\mathbf{W}^{(l)}$ is a learnable weight matrix, and $\sigma(\cdot)$ denotes a non-linear activation function.

GCNs have demonstrated strong performance across many graph-learning tasks. However, they assume *homophily*, where neighboring nodes tend to share similar characteristics. Their

performance often degrades on heterophilic graphs, and deeper GCNs suffer from *oversmoothing*, where node representations gradually become indistinguishable [4].

2.2.2 GraphSAGE

GraphSAGE (Sample and Aggregate) [10] extends the message-passing framework by sampling a fixed number of neighboring nodes instead of aggregating information from the entire neighborhood. This sampling strategy significantly improves scalability and enables inductive learning, allowing the model to generalize to previously unseen nodes. Different aggregation functions, including mean, max pooling, and LSTM aggregators, can be employed depending on the application.

2.2.3 Graph Attention Networks (GAT)

Graph Attention Networks (GATs) [11] improve message passing by learning the relative importance of neighboring nodes through an attention mechanism rather than relying solely on graph connectivity. The attention coefficient between neighboring nodes is computed as

$$\alpha_{ij} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^\top [\mathbf{W}\mathbf{h}_i \| \mathbf{W}\mathbf{h}_j]\right)\right)}{\sum_{k \in \mathcal{N}(i)} \exp\left(\text{LeakyReLU}\left(\mathbf{a}^\top [\mathbf{W}\mathbf{h}_i \| \mathbf{W}\mathbf{h}_k]\right)\right)}, \tag{3}$$

where $\mathbf{a}$ and $\mathbf{W}$ are learnable parameters. Multi-head attention is typically employed to improve representation quality and training stability. Although GAT learns different attention weights for different neighbors, it still relies on a shared message-passing mechanism across graph edges.

2.2.4 Graph Isomorphism Networks (GIN)

Graph Isomorphism Networks (GINs) [12] were developed to maximize the expressive power of message-passing Graph Neural Networks. By combining sum aggregation with a Multi-Layer Perceptron (MLP), GIN achieves expressive power comparable to the Weisfeiler-Lehman graph isomorphism test, making it one of the most discriminative GNN architectures for graph classification tasks.

Although GCN, GraphSAGE, GAT, and GIN differ in their aggregation strategies, they all rely on message passing to exchange information between neighboring nodes. In detector graphs, however, neighboring Tracksters frequently originate from different particles, producing highly heterophilic graph structures. This motivates more flexible message-passing mechanisms capable of adapting information exchange to individual graph edges, leading to the development of Sheaf Neural Networks.

2.2.5 Sheaf Neural Networks (SNN)

Conventional Graph Neural Networks assume that the same message-passing operation can be applied uniformly across every edge in a graph. While this assumption is effective for homophilic graphs, it becomes less suitable for heterophilic graphs, where neighboring nodes may represent fundamentally different entities. In detector reconstruction, adjacent Tracksters often originate from different particles, making adaptive information exchange an important requirement.

Sheaf Neural Networks (SNNs) [5, 6] extend the message-passing framework using concepts from algebraic topology. Instead of associating each node with a single feature vector, SNNs assign a vector space, known as a *stalk*, to every node and edge in the graph. Information is exchanged through learnable linear transformations called *restriction maps*,which map node representations into compatible edge spaces, as illustrated in Figure 1.

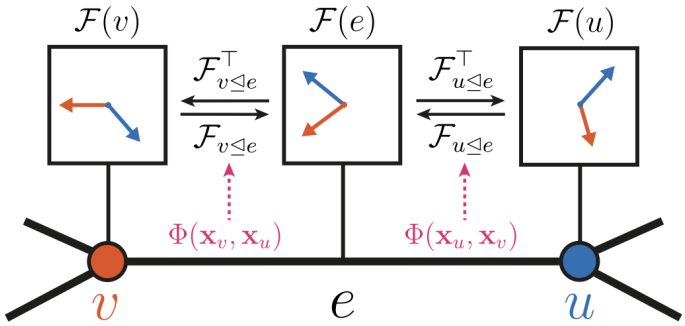


Figure 1 Illustration of a cellular sheaf defined on a graph, adapted from [5].

Formally, a cellular sheaf $\mathcal{F}$ defined on a graph $G = (V, E)$ consists of

- a vector space $\mathcal{F}(v)$ assigned to every node $v \in V$,
- a vector space $\mathcal{F}(e)$ assigned to every edge $e \in E$, and
- a learnable linear restriction map $\mathcal{F}_{v \trianglelefteq e} : \mathcal{F}(v) \to \mathcal{F}(e)$ for every incident node-edge pair.

Rather than directly comparing neighboring node features, the restriction maps first transform node representations into a shared edge space, where their agreement or disagreement can be evaluated. This allows message passing to adapt to the local relationship between neighboring nodes instead of relying on a single global aggregation rule.

The consistency of neighboring node representations is captured through the *sheaf Laplacian*, defined as

$$L_{\mathcal{F}}(x)_v = \sum_{u:\, v, u \trianglelefteq e} \mathcal{F}_{v \trianglelefteq e}^{\top} \left(\mathcal{F}_{v \trianglelefteq e} x_v - \mathcal{F}_{u \trianglelefteq e} x_u \right). \tag{4}$$

Using the normalized sheaf Laplacian

$$\Delta_{\mathcal{F}} = D^{-1/2} L_{\mathcal{F}} D^{-1/2}, \tag{5}$$

feature propagation is performed through a sheaf diffusion process

$$\mathbf{H}^{(l+1)} = \sigma\left((\mathbf{I} - \Delta_{\mathcal{F}}) \mathbf{H}^{(l)} \mathbf{W}^{(l)} \right). \tag{6}$$

Unlike conventional Graph Neural Networks, the restriction maps are learned during training, enabling edge-specific message passing that adapts information exchange according to the relationship between neighboring Tracksters. This additional flexibility makes Sheaf Neural Networks particularly well suited for heterophilic graph-learning problems such as Trackster Linking, where neighboring detector objects do not necessarily belong to the same underlying particle.

2.3 Dataset

The dataset used in this project consists of graph representations generated from simulated CMS High-Granularity Calorimeter (HGCAL) events. Each graph corresponds to a collision event, where reconstructed Tracksters are represented as graph nodes and candidate links between neighboring Tracksters are represented as graph edges.

Each node is described by reconstructed detector features, including spatial coordinates, energy, timing information, principal component analysis (PCA) variables, and particle identification

features. Edge features characterize the geometric and physical relationship between neighboring Tracksters using quantities such as spatial distance, angular separation, and energy-related measurements.

Ground-truth labels are obtained from detector simulation by determining whether two connected Tracksters originate from the same simulated particle. Consequently, Trackster Linking is formulated as a binary edge-classification problem, where the objective is to predict whether each candidate edge corresponds to a true particle connection.

3 Methodology

3.1 Dataset Generation and Graph Construction Pipeline

The CMS Trackster Linking dataset is distributed as ROOT files containing reconstructed detector information for simulated HGCAL collision events. Before training, these events are converted into graph representations compatible with the PyTorch Geometric framework through a dedicated preprocessing pipeline.

The preprocessing pipeline reads the ROOT files in parallel using `uproot` and converts each collision event into an independent `torch_geometric.data.Data` object. Node and edge features described in Section 2.3 are extracted from the reconstructed detector information, while ground-truth edge labels are assigned from the detector simulation.

Feature normalization is then performed using a global Max-Abs scaling strategy computed from the training dataset. For each feature column j, the normalized value is computed as

$$x_{i,j}^{\text{scaled}} = \frac{x_{i,j}}{\max_k |x_{k,j}|}, \tag{7}$$

where $x_{i,j}$ denotes the original feature value and the denominator corresponds to the maximum absolute value observed for that feature across the training set. This normalization maps each feature to a comparable numerical range, improving optimization stability during training.

Finally, the processed graphs are organized into training and testing datasets and stored in PyTorch Geometric format for subsequent model development and evaluation.

3.2 Graph Construction

Each collision event is represented as an undirected graph

$$G = (V, E),$$

where each node corresponds to a reconstructed *Trackster*, representing a three-dimensional particle shower candidate, and each edge represents a candidate connection between two neighboring Tracksters.

Each node is described by a feature vector containing reconstructed detector information, including spatial coordinates, deposited energy, timing information, principal component analysis (PCA) features, and additional detector-specific quantities. Similarly, each edge is associated with a set of geometric and physical features describing the relationship between the connected Tracksters.

Ground-truth edge labels are obtained from simulation by determining whether the connected Tracksters originate from the same simulated particle. Consequently, the Trackster Linking task is formulated as a binary edge-classification problem. During preprocessing, the pileup masking strategy adopted in the original Trackster Linking framework is also applied to suppress edges associated with pileup interactions.

3.3 Sheaf Neural Network Architecture

The proposed model extends the existing Trackster Linking Graph Neural Network by replacing the conventional graph message-passing operator with a Sheaf Neural Network (SNN). Rather than directly exchanging node features between neighboring Tracksters, the proposed architecture first embeds node representations into sheaf feature spaces and applies learned edge-specific restriction maps that adapt information exchange during message passing.

The overall architecture consists of four main stages: feature embedding, global attention, sheaf message passing, and edge classification. First, an embedding network projects the original Trackster features into a higher-dimensional latent space. Next, multiple sheaf message-passing layers iteratively propagate information using learned restriction maps and sheaf diffusion. Finally, an edge classification network predicts whether each candidate edge connects Tracksters originating from the same particle.

To ensure a fair comparison with the baseline Graph Neural Network, the overall training pipeline, optimizer, loss function, and evaluation procedure were kept unchanged. Consequently, any performance differences can be attributed solely to the replacement of the message-passing mechanism.

Figure 2 illustrates the proposed Sheaf Neural Network architecture. Panel (a) presents the overall processing pipeline, while panel (b) shows the sheaf message-passing mechanism based on learned restriction maps used for edge classification.

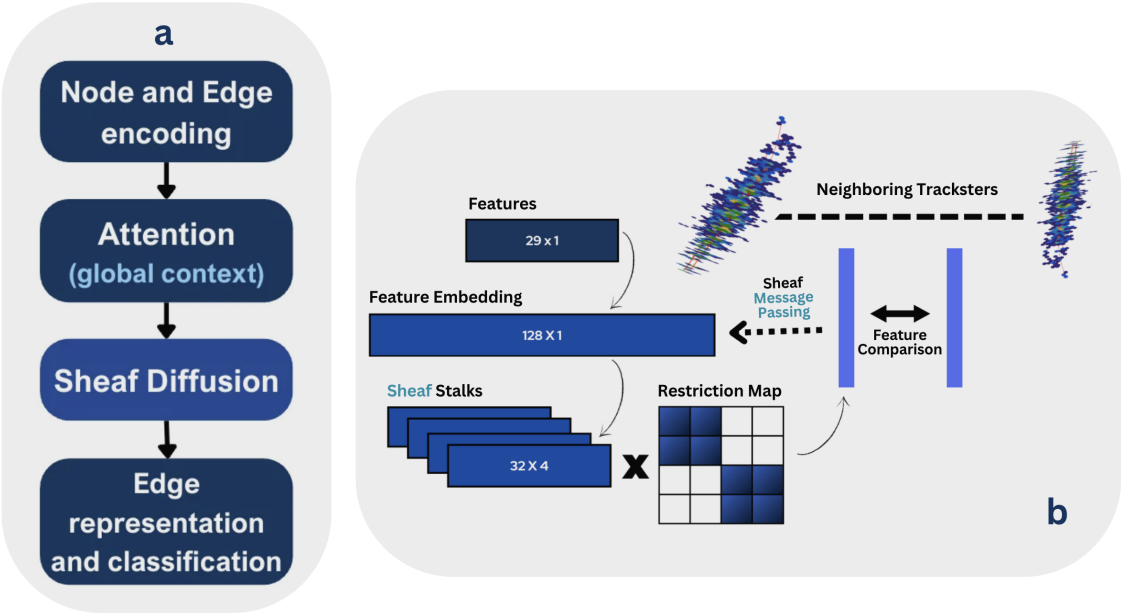


Figure 2 Overview of the proposed Sheaf Neural Network architecture. (a) Overall processing pipeline, consisting of node and edge encoding, attention, sheaf diffusion, and edge classification. (b) Illustration of the sheaf message-passing mechanism, where Trackster features are embedded into sheaf feature spaces and transformed through learned restriction maps before edge classification.

3.4 Restriction Map Parameterization

A key component of the proposed Sheaf Neural Network is the restriction map, which defines how information is transformed between neighboring Tracksters during message passing. Unlike conventional Graph Neural Networks, which apply the same message-passing operation across all graph edges, restriction maps are learned individually for each edge, enabling information

exchange to adapt to the local graph structure.

Two restriction map parameterizations were implemented and evaluated in this work:

- **Orthogonal restriction maps**, which constrain the learned transformations to orthogonal matrices, preserving vector norms while allowing rotations of the feature space.
- **Block-diagonal restriction maps**, which partition the feature space into multiple independent subspaces. This reduces the number of learnable parameters while allowing different groups of features to learn specialized transformations.

Both parameterizations were integrated into the Trackster Linking framework using the same training and evaluation pipeline. Unless otherwise stated, the experimental results presented in this report correspond to the block-diagonal parameterization.

3.5 Experimental Configuration

Two experimental configurations were used in this work. The first was designed to provide a controlled comparison between the baseline Graph Neural Network (GNN) and the proposed Sheaf Neural Network (SNN). Both models were trained under identical conditions using a subset of 100 simulated collision events, ensuring that any performance differences resulted solely from the message-passing architecture.

To evaluate the proposed architecture under a more realistic training scenario, a second configuration with increased model capacity was used to train the SNN on a larger dataset consisting of 5000 simulated collision events. This configuration produced the final performance results presented in Chapter 4.

Table 1 Experimental configuration used for the baseline GNN and SNN comparison (100-event dataset).

Parameter	Value
Node features	29
Edge features	12
Hidden dimension	32
Edge hidden dimension	16
Stalk dimension (SNN only)	4
Block size	2
Number of layers	2
Message-passing iterations	2
Attention heads	4
Dropout	0.0
Weighted aggregation	Enabled
Learning rate scheduler	Cosine Annealing
Optimizer	AdamW
Learning rate	1×10^{-3}
Weight decay	0.01
Epochs	150
Loss function	Combined (Focal + Contrastive)

Table 2 Experimental configuration used for the final SNN evaluation (5000-event dataset).

Parameter	Value
Node features	29
Edge features	12
Hidden dimension	128
Edge hidden dimension	64
Stalk dimension	4
Block size	2
Number of layers	3
Message-passing iterations	3
Attention heads	8
Dropout	0.3
Weighted aggregation	Enabled
Optimizer	AdamW
Learning rate scheduler	Cosine Annealing
Learning rate	2×10^{-4}
Weight decay	0.01
Loss function	Combined (Focal + Contrastive)

3.6 Training Procedure

The models were optimized using AdamW together with a cosine annealing learning-rate scheduler. The hyperparameters for each experimental configuration are summarized in Tables 1 and 2. To address the class imbalance inherent in the Trackster Linking task, training employed a combined loss consisting of Focal Loss for edge classification and Contrastive Loss to encourage discriminative node representations.

3.7 Evaluation

Model performance was evaluated using the validation framework provided by the CMS Trackster Linking pipeline. During validation, the optimal classification threshold was selected by maximizing the weighted validation F1-score.

The trained models were evaluated using standard binary classification metrics, including Accuracy, Precision, Recall, F1-score, and the Area Under the Receiver Operating Characteristic Curve (ROC AUC). In addition, energy-weighted ROC AUC was computed to account for the physical importance of high-energy particle showers.

To further analyze the behavior of the proposed model, Receiver Operating Characteristic (ROC) curves, prediction score distributions, and comparisons with the baseline Graph Neural Network were generated. These complementary analyses provide insight into both the overall classification performance and the discriminative capability of the learned edge representations.

4 Results

This chapter presents the experimental results obtained using the proposed Sheaf Neural Network (SNN) for the Trackster Linking task. Unless otherwise stated, the reported results correspond to the final SNN configuration trained on the 5000-event dataset using the block-diagonal restriction map parameterization described in Chapter 3.

The evaluation begins with the overall edge-classification performance, followed by an analysis of the Receiver Operating Characteristic (ROC) curves and prediction score distributions. Finally, the proposed approach is compared with the baseline Graph Neural Network (GNN) to assess the impact of incorporating sheaf-based message passing into the Trackster Linking framework.

This chapter presents the experimental evaluation of the proposed Sheaf Neural Network within the CMS Trackster Linking framework. Since large-scale training and integration are part of the ongoing collaboration, the experiments were conducted on representative subsets of the available dataset. These experiments verify the correctness of the implementation and evaluate whether the proposed architecture can learn meaningful edge representations before large-scale training.

4.1 Overall Classification Performance

Table 3 summarizes the performance of the proposed Sheaf Neural Network on the Trackster Linking task. The model achieved strong classification performance across all evaluation metrics, indicating its ability to accurately distinguish true Trackster connections from incorrect candidate links.

Table 3 Overall edge-classification performance of the proposed Sheaf Neural Network.

Metric	Performance
Accuracy	91.50%
Precision	93.79%
Recall	96.99%
F1-score	95.36%
ROC AUC	87.75%
Energy-weighted ROC AUC	91.64%

The model achieved an accuracy of 91.50%, demonstrating reliable overall classification performance. A precision of 93.79% indicates that the majority of predicted links correspond to genuine particle connections, while the high recall of 96.99% shows that most true Trackster links were successfully identified.

The resulting F1-score of 95.36% reflects a strong balance between precision and recall. In addition, the energy-weighted ROC AUC exceeds the standard ROC AUC, suggesting that the model performs particularly well on higher-energy particle showers, which contribute more significantly to the reconstruction process.

4.2 Receiver Operating Characteristic Analysis

While threshold-dependent metrics such as accuracy and F1-score summarize the performance at a single operating point, the Receiver Operating Characteristic (ROC) curve provides a threshold-independent evaluation of the classifier. The ROC curve illustrates the trade-off between the True Positive Rate (TPR) and the False Positive Rate (FPR) across all possible decision thresholds, while the Area Under the Curve (ROC AUC) summarizes the overall discriminative capability of the model.

Figure 3 presents the ROC curve of the proposed Sheaf Neural Network. The model achieved a ROC AUC of 87.75%, indicating strong discriminative capability. When edge contributions are weighted by detector energy, the ROC AUC increases to 91.64%, demonstrating improved discrimination for higher-energy particle showers.

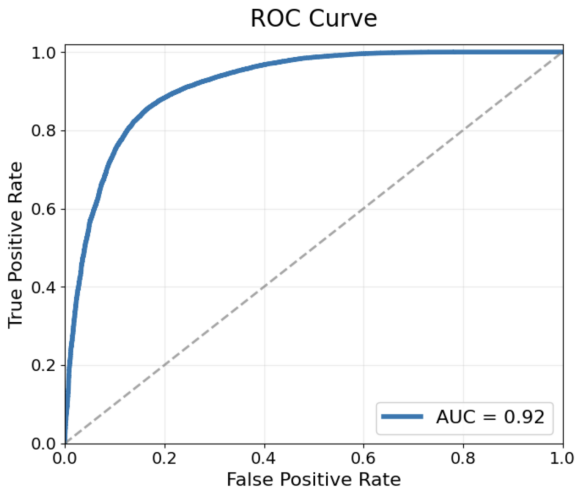


Figure 3 Receiver Operating Characteristic (ROC) curve of the proposed Sheaf Neural Network on the Trackster Linking task.

Overall, the ROC analysis confirms that the proposed Sheaf Neural Network maintains robust classification performance over different operating thresholds and is capable of effectively separating true Trackster links from background connections.

4.3 Prediction Score Distribution

To further investigate the behavior of the proposed Sheaf Neural Network, the distribution of prediction scores assigned to positive and negative edge candidates was analyzed. Unlike summary metrics, prediction score distributions provide insight into the confidence of the classifier and the degree of separation achieved between the two classes.

Figure 4 presents the prediction score distributions for positive and negative Trackster links. Positive edges are concentrated at high prediction scores, whereas negative edges occupy substantially lower score ranges. Although a limited overlap is observed around the classification threshold, the two distributions remain well separated, indicating that the proposed model learns discriminative edge representations and assigns high confidence to the majority of correctly classified samples.

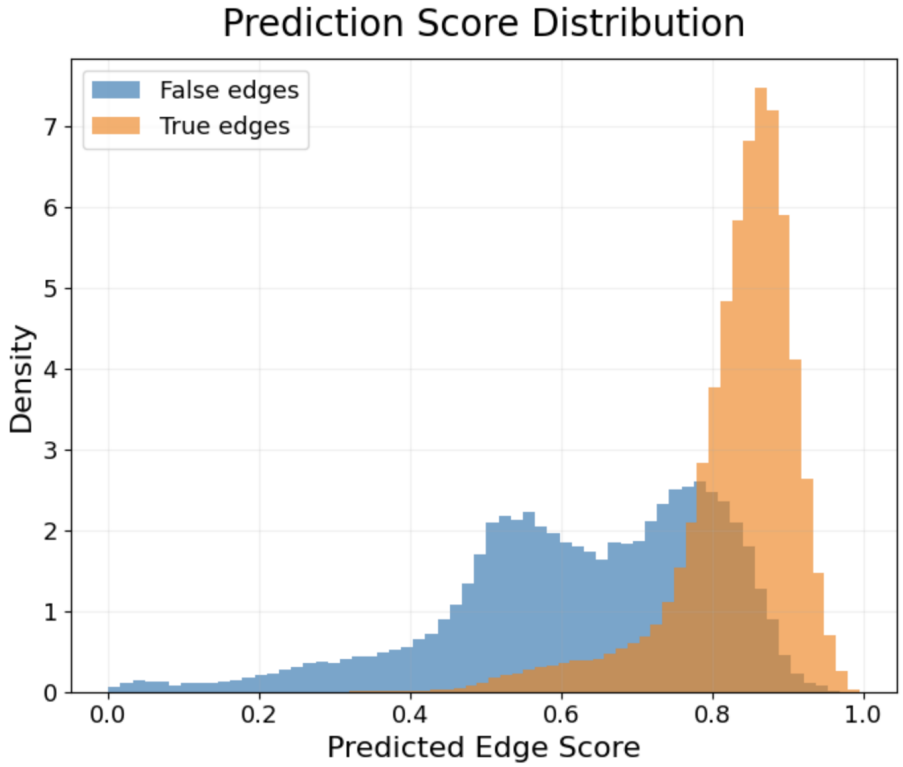


Figure 4 Prediction score distributions for positive and negative Trackster links.

4.4 Performance Across Pseudorapidity

To further evaluate the robustness of the proposed Sheaf Neural Network, the model performance was analyzed as a function of the Trackster pseudorapidity (η). Unlike the overall metrics presented previously, this analysis investigates whether the model maintains consistent performance across different regions of the High-Granularity Calorimeter (HGCAL).

Figure 5 presents the energy-weighted Accuracy, Precision, Recall, and F1-score computed for different pseudorapidity intervals. The evaluation follows the standard CMS Trackster Linking procedure, where each edge contributes according to its associated detector energy.

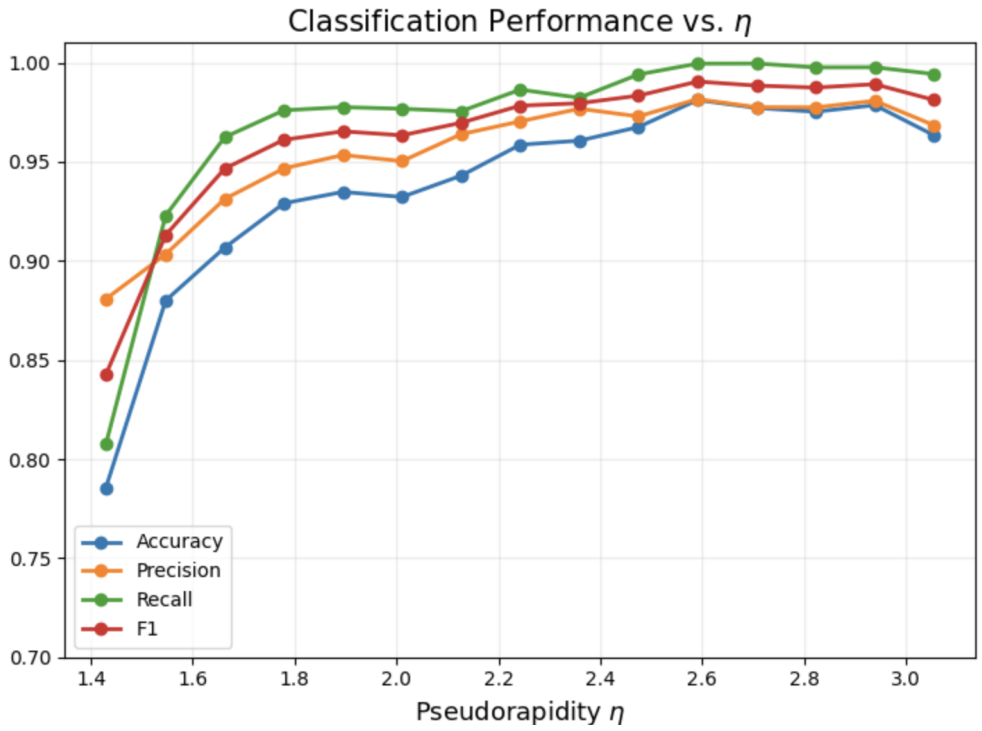


Figure 5 Energy-weighted Accuracy, Precision, Recall, and F1-score of the proposed Sheaf Neural Network as a function of Trackster pseudorapidity (η).

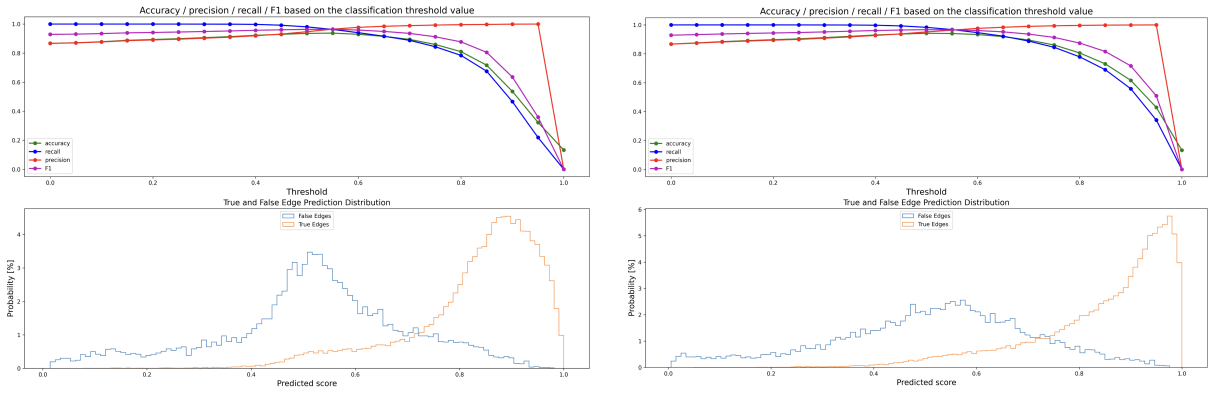
As shown in Figure 5, all four performance metrics improve as the pseudorapidity increases from approximately $\eta = 1.4$ before reaching a stable plateau above $\eta \approx 2.5$. Throughout the evaluated detector region, the proposed model consistently achieves high classification performance, with the energy-weighted F1-score remaining close to 0.99 in the highest pseudorapidity intervals.

The absence of a significant performance degradation across the detector acceptance indicates that the proposed Sheaf Neural Network generalizes well over different detector geometries. These results suggest that the learned sheaf-based message-passing mechanism remains effective despite the varying detector granularity and particle shower characteristics encountered at different pseudorapidities.

4.5 Comparison with the Baseline Graph Neural Network

To evaluate the proposed sheaf-based message-passing mechanism, the Sheaf Neural Network (SNN) was compared with the baseline Graph Neural Network (GNN) using the experimental configuration described in Section 3.5. Both models were trained on the same subset of 100 simulated collision events for 150 epochs, ensuring that the message-passing architecture was the only difference between the two models.

Figure 6 compares the threshold-dependent classification metrics and the prediction score distributions obtained by both models. In both cases, the networks successfully separate true and false Trackster links. However, the proposed SNN produces a clearer separation between the two classes, with true edges concentrated closer to prediction scores of one and false edges remaining closer to zero.



(a) Baseline GNN **(b)** Proposed SNN

Figure 6 Comparison of the prediction score distributions and threshold-dependent classification metrics obtained using the baseline Graph Neural Network (GNN) and the proposed Sheaf Neural Network (SNN) on the 100-event comparison dataset after 150 training epochs.

The threshold analysis also shows that the SNN maintains a more favorable balance between precision and recall over a broad range of decision thresholds, resulting in consistently high F1-score values. These observations indicate that replacing conventional message passing with sheaf-based message passing improves the discriminative capability of the learned edge representations.

Although this comparison was performed on a representative subset of 100 simulated events, it demonstrates that the proposed architecture successfully learns the Trackster Linking task and motivates further evaluation on the larger 5000-event dataset.

4.6 Discussion

The experimental results demonstrate that the proposed Sheaf Neural Network can be successfully integrated into the CMS Trackster Linking framework and effectively learns the Trackster Linking task. The model achieves high classification accuracy, a strong balance between precision and recall, and robust separation between true and false Trackster connections.

The prediction score distributions indicate good separation between positive and negative edges, while the comparison with the baseline Graph Neural Network suggests that replacing conventional message passing with sheaf-based message passing improves the discriminative capability of the learned edge representations without altering the overall Trackster Linking framework.

The primary objective of this work was to implement and validate the integration of Sheaf Neural Networks into the existing CMS reconstruction pipeline rather than to establish the final performance of the model on the complete CMS dataset. The successful implementation and experimental evaluation presented in this work demonstrate that the proposed architecture functions correctly and provides a solid foundation for future large-scale studies.

Overall, these results indicate that Sheaf Neural Networks are a promising extension to the Trackster Linking framework and represent a viable direction for improving graph-based particle reconstruction in future CMS developments.

4.7 Limitations and Future Work

This work focused on implementing and validating Sheaf Neural Networks within the CMS Trackster Linking framework. Due to the limited duration of the internship, the experimental evaluation was conducted on a representative subset of 100 simulated collision events. While this

subset was sufficient to verify the correctness of the implementation and demonstrate that the proposed architecture successfully learns the Trackster Linking task, it does not fully represent the scale and complexity of the complete CMS dataset.

A larger training configuration for a dataset of 5000 simulated collision events was also developed during this work. However, the computational resources and training time required for large-scale experiments exceeded the scope of the internship. Consequently, a comprehensive evaluation on the full dataset remains part of future work.

Several additional research directions can also be explored. Different restriction map parameterizations, such as orthogonal and block-diagonal mappings, could be evaluated more extensively to better understand their influence on learning and computational efficiency. Furthermore, investigating alternative sheaf architectures, hyperparameter configurations, and training strategies may further improve the performance of the proposed model.

Finally, the integration presented in this work establishes a foundation for future studies investigating topological deep learning methods for particle reconstruction within the CMS experiment.

4.8 Conclusion

This project investigated the application of Sheaf Neural Networks to the Trackster Linking problem within the CMS experiment at CERN. Building upon the existing Graph Neural Network framework, a Sheaf Neural Network architecture was successfully integrated into the Trackster Linking pipeline while preserving the original preprocessing, training, and evaluation procedures.

The proposed implementation introduces learnable restriction maps that enable adaptive message passing between neighboring Tracksters. Two restriction map parameterizations, namely orthogonal and block-diagonal mappings, were implemented within the framework, with the experimental evaluation presented in this report focusing on the block-diagonal approach.

Experimental results obtained on a representative subset of simulated collision events demonstrate that the proposed architecture successfully learns the Trackster Linking task and achieves strong edge-classification performance. Furthermore, the comparison with the baseline Graph Neural Network indicates that the sheaf-based message-passing mechanism provides competitive performance while offering a more flexible framework for modeling complex relationships between neighboring detector objects.

Overall, this work demonstrates the feasibility of incorporating topological deep learning techniques into the CMS Trackster Linking framework and provides a foundation for future large-scale studies of Sheaf Neural Networks for particle reconstruction in high-energy physics.

References

[1] G Apollinari et al. *High Luminosity Large Hadron Collider HL-LHC*. en. 2015. DOI: 10.5170/CERN-2015-005.1. URL: https://cds.cern.ch/record/2120673.

[2] CMS Collaboration. *The Phase-2 Upgrade of the CMS Endcap Calorimeter*. Tech. rep. CERN-LHCC-2017-023, CMS-TDR-019. CERN, 2017.

[3] Franco Scarselli et al. "The Graph Neural Network Model". In: *IEEE Transactions on Neural Networks* 20.1 (2009), pp. 61–80. DOI: 10.1109/TNN.2008.2005605.

[4] Qimai Li, Zhichao Han, and Xiao-Ming Wu. "Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning". In: *AAAI*. 2018.

[5] Cristian Bodnar et al. *Neural Sheaf Diffusion: A Topological Perspective on Heterophily and Oversmoothing in GNNs*. 2023. arXiv: 2202.04579 [cs.LG]. URL: https://arxiv.org/abs/2202.04579.

[6] Cristian Bodnar et al. *Neural Sheaf Diffusion: A Topological Perspective on Heterophily and Oversmoothing in GNNs*. 2023. arXiv: 2202.04579 [cs.LG]. URL: https://arxiv.org/abs/2202.04579.

[7] Lyndon Evans and Philip Bryant. "LHC Machine". In: *Journal of Instrumentation* 3 (2008), S08001.

[8] CMS Collaboration. "The CMS Experiment at the CERN LHC". In: *Journal of Instrumentation* 3 (2008), S08004.

[9] Thomas N. Kipf and Max Welling. *Semi-Supervised Classification with Graph Convolutional Networks*. 2017. arXiv: 1609.02907 [cs.LG]. URL: https://arxiv.org/abs/1609.02907.

[10] William L. Hamilton, Rex Ying, and Jure Leskovec. *Inductive Representation Learning on Large Graphs*. 2018. arXiv: 1706.02216 [cs.SI]. URL: https://arxiv.org/abs/1706.02216.

[11] Petar Veličković et al. *Graph Attention Networks*. 2018. arXiv: 1710.10903 [stat.ML]. URL: https://arxiv.org/abs/1710.10903.

[12] Keyulu Xu et al. "How Powerful are Graph Neural Networks?" In: *International Conference on Learning Representations (ICLR)*. 2019.