Graph-based Deep Learning Methods for Particle Reconstruction at the LFHCal

Selma Mazioud
Advisor: Professor Helen Caines

1 Introduction

Problem I investigate graph-based deep learning methods for clustering calorimeter data (i.e. particle reconstruction), in preparation for the upcoming Electron Ion Collider (EIC). This experiment aims to study the properties of Quark-Gluon plasma, a state of matter in which gluons and quarks are freely interacting and that was one of the first form of matters to ever exist after the Big Bang, and that we can recreate in high energy collisions. We are at a critical juncture where mathematical modeling and simulation are essential for understanding how detectors respond to particles passing through them. The complexity of the detector geometry and the high amount of data it collects make traditional methods obsolete as they tend to not be scalable. I had approached this problem using simpler machine learning methods like k-means and would be interested in studying how they compare to graph-based methods. This problem is particularly interesting because the data that we study does not have an inherent graph structure—rather, graph construction will be the most physics-informed step of the proposed methods. The overall goal of my project is to encourage the study of graph-based deep learning methods and their applications to high-energy physics problems, from particle tracking and jet tagging to reconstruction.

Particle reconstruction A cluster is built by collecting the energy deposits left by a particle shower in the detector, where each cluster represents a single particle or overlapping particles. Since the Molière radius of the showers (radius in which 90% of the shower energy is contained) is usually bigger than the surface area of a single tower, one must combine information from multiple towers to reconstruct the energy deposited as it is spread across multiple towers [1]. While showers produced by single electromagnetic particles are relatively easy to reconstruct, overlapping particle showers (as it is the case in jets) pose a more significant problem. The goal is then to absorb as much of the deposited energy as possible during the clusterization process. More information about the LFHCal can be found in the appendix.

Method overview In this work, we implement a dynamic graph neural network that uses a trainable adjacency matrix to embed node features into a highly expressive latent space. The model iteratively re-embeds and constructs k-nearest neighbor graphs, before passing the final graph embedding through a node classifier that will determine which cluster each hit belongs to. We include our code in https://github.com/selma-m/DLG-for-Particle-Reconstruction.

Summary of the results Although the performance of the method is not as good as hoped and stays below traditional k-means performance, our implementation sheds light on valuable insights into the limitations of the approach, and provided new avenues of research that I am eager to explore next semester.

2 Related Work

More approaches (including Kalman filters and Interaction Networks) are presented in the appendix 5.

Dynamic Graph CNN EdgeConv is a module that captures local neighborhood information, leverages affinity in the feature space, and is permutation invariant. The graph is dynamically updated after each layer of the network. Initially, a graph G is constructed as a k-nearest neighbors (k-NN) graph of the input X in $\mathbb{R}^F$ (where F is the feature dimensionality) that includes self-loops. The edge features $e_{ij} = h_\Theta(x_i, x_j)$ are functions of the nodes they connect, where the function has learnable parameters Θ. In the paper, they adopt an asymmetric edge function $h_\Theta(x_i, x_j) = \bar{h}_\Theta(x_i, x_i - x_j)$, which combines global shape structure with local neighborhood information and is implemented as a shared MLP. Specifically, they set $e'_{ijm} = \text{ReLU}(\theta_m \cdot (x_j - x_i) + \phi_m \cdot x_i)$ where $\Theta = (\theta_1, ..., \theta_M, \phi_1, ...\phi_M)$ and m represents the layer. The new node embedding is then $x'_i = \square_{j:(i,j)\in E} \bar{h}_\Theta(x_i, x_i - x_j)$ where $\square$ is an aggregation function (here set to be the maximum). After each edge convolution, we re-run k-NN in the feature space produced by the previous layer. A limitation of this method is the initial computation of edge features, which doesn't always come with the input data. This step is not obvious and could be computationally expensive.

GravNet and GarNet GravNet and GarNet [6] are two proposed architectures that aim to learn the geometry of the data via trainable adjacency matrices. Dense neural networks are constructed to learn a representation of vertex features. For GravNet, the input of the network is interpreted as the coordinates in the learned representation space S, so the graph is constructed in the following way: the nodes are the registered hits, and the edges are drawn by associated nearest neighbors based on Euclidean distance in the learned space S. On the other hand, GarNet sees the output of the MLP as a set of distances between vertices and a set of S aggregators. In the graph, we connect each vertex to a set of S aggregators, where the learned distance is the edge weights. The next step consists in collecting information by utilizing a function of distance called potential by updating feature vectors and aggregating the obtained information. The reconstruction loss used is weighted by the energy at each node. GravNet and GarNet both assume that every event has the same number of particles, which is not always the case.

3 Method

Model We will use a Dynamic Graph CNN approach based on edge convolutions inspired by [7] and [6]. Initially, we define our nodes to be the recorded hits, with node features $v_i = (x_i, y_i, z_i, ix, iy, iz, E_i)$ where x_i, y_i, z_i are the Cartesian coordinates of the hit, ix, iy, iz are the tower index, and E_i is its energy.

The model we implement stacks blocks of dense layers and GravNet layers, as shown in 2. The GravNet module leverages multiple dense layers to re-embed the input graph into a latent space before performing k-nearest neighbors to build an updated graph (see 1). We do this iteratively, which means that at each step, we incorporate local and global neighborhood information into the structure of the graph. Finally, we pass our final graph embedding through a classifier which will output, for each node, a distribution over k clusters.

Initially, my model included a k-means differentiable layer that aimed to reconstruct the number of clusters since the events are generated by a varying number of particles. However, this turned out to be challenging to integrate into the training pipeline. We therefore sample a fixed amount of clusters in each sample ($k \in \{5, 10, 15\}$. Hypothesizing that the model might struggle with the z-direction, as the trajectories of the particles are particularly complex, we also apply the model on z-faces, where we consider each z "slice" as an independent model. We show these results in the appendix 5 (Table 2 and Figure 10).

My model extends the GravNet method previously described beyond binary classification and uses a modified loss function. Additionally, we provide more interpretability to the model by using a diffusion geometry-based dimensionality reduction technique (PHATE [4]) to visualize the embeddings created by the model.

Loss function We use CrossEntropy loss.

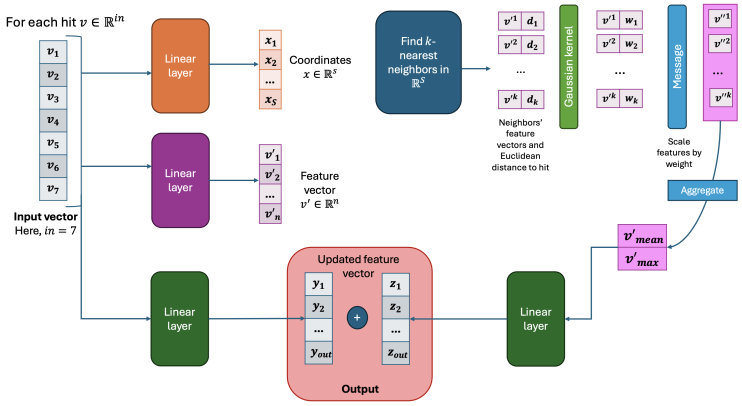

Figure 1: GravNet module.

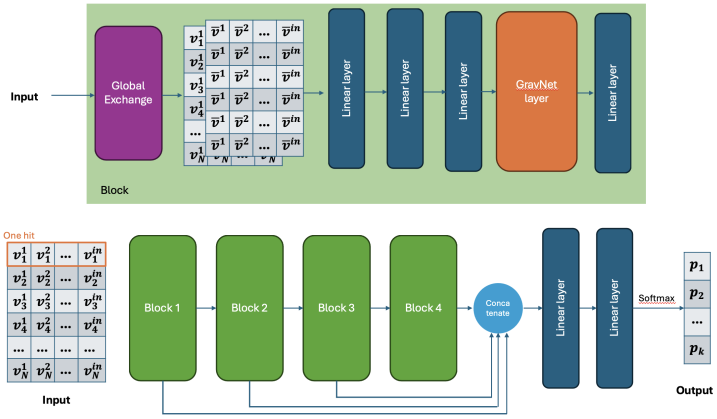

Figure 2: Module architecture.

4 Experiments

4.1 Dataset

I base my analysis on a toy dataset [5], since we don't have real data yet (the experiments are not set to start until a few years). The data used are simulations provided by the ePIC collaboration that reflect the real geometry of the detector. About 1,000 events were simulated using a merged "particle gun" with neutrons, protons, and pions fired from the interaction point that is 3.58 m from the surface of the LFHCal. An important note to make is that the simulations were completed without the particle additionally interacting with the electromagnetic calorimeter and with each other. In other words, single particles were shot at a time and the result was aggregated for higher multiplicity. Nevertheless, the particle multiplicities and energies in these events are set to reflect semi-realistic conditions for ePIC data-collecting, averaging about 20 particles per event. More visuals are included in the Appendix 5 (Figure 7 and 8).

4.2 Training details

- We train our models for 20 epochs using the Adam optimizer, set to a learning rate of 0.001. We perform a train/test split of 80/20.
- We set the seed to 42, 29, and 0, in each run of the model.
- We run our experiments on Yale HPC CPU clusters.
- Data processing. The cluster assignment numbers in the dataset are randomly generated; we map them to integers from 0 to $k-1$, where k is the number of clusters. Additionally,

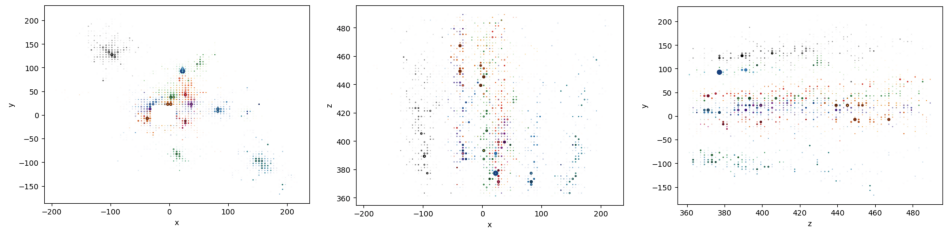

Figure 3: A typical event seen in two dimensions, in the xy (front view), xz (top view), and zy (lateral views) planes. We color the hits by true cluster and the opacity and size of the points is proportional to its energy. The rectangular hole is due to the geometry of the detector. Note that a typical event has $3000 - 4000$ hits, with $15 - 20$ particles involved in leaving energy deposits. Here, the number of clusters is $k = 19$.

	Test accuracy	Homogeneity	Completeness	V-Score
$k = 2, n_{\text{neighb}} = 11$	$59.61 \pm 0.87\%$	0.036 ± 0.013	0.042 ± 0.015	0.039 ± 0.013
$k = 2, n_{\text{neighb}} = 30$	59.460.92	0.033 ± 0.014	0.036 ± 0.014	0.034 ± 0.014
$k = 5, n_{\text{neighb}} = 11$	$23.35 \pm 0.65\%$	0.053 ± 0.031	0.025 ± 0.003	0.013 ± 0.002
$k = 5, n_{\text{neighb}} = 30$	$23.24 \pm 0.82\%$	0.012 ± 0.00	0.024 ± 0.001	0.017 ± 0.00
$k = 10, n_{\text{neighb}} = 11$	$9.77 \pm 0.26\%$	0.00 ± 0.00	0.678 ± 0.455	0.002 ± 0.003
$k = 10, n_{\text{neighb}} = 30$	$9.99 \pm 0.59\%$	0.00 ± 0.00	0.352 ± 0.459	0.00 ± 0.00
$k = 15, n_{\text{neighb}} = 11$	$10.07 \pm 0.67\%$	0.040 ± 0.00	0.058 ± 0.006	0.047 ± 0.00
$k = 15, n_{\text{neighb}} = 30$	$9.91 \pm 0.396\%$	0.046 ± 0.00	0.061 ± 0.00	0.052 ± 0.00
k-means method		0.904	0.896	0.900

Table 1: Test accuracy, homogeneity, completeness, and Validity Scores. The results are averaged over three runs.

because the dataset has a varying number of cluster per event, for our analysis, we randomly choose 2,5, 10, and 15 clusters per event and disregard the rest.

4.3 Results

Comparison with baseline We compare our method to simply performing k-means clustering on the raw data, inputting x, y, z coordinates and hit energies.

Analysis of results. The results obtained (as shown in Table 1) show that the model does not outperform vanilla k-means clustering. This suggests that the models don't learn the right laws of physics; no graph is explicitly constructed to integrate domain knowledge. A closer look at the generated embeddings in Figure 5 reveals a lack of clear structure by cluster. This observation is critical because it indicates that the model's internal representation of the data does not effectively separate the underlying classes, leading to suboptimal clustering performance. The embeddings, ideally, should exhibit a discernible pattern where points belonging to the same cluster are grouped together, but this is not evident in the current results.

One of the key challenges identified is the absence of an explicitly constructed graph that integrates domain knowledge into the learning process. The dynamic graph neural network, while flexible, might not inherently capture the specific physical relationships and interactions present in the data unless adequately guided by domain-specific information. This contrasts with k-means clustering, which, although simpler, can sometimes better leverage the inherent structure of the data when such relationships are not explicitly modeled.

The model does not seem to be very sensitive to hyper-parameter choice. However, I noticed that the results varied greatly depending on the number of GravNet layers I stacked. I found less layers (≤ 2) provides better results than more (≥ 4). Moreover, the results varied depending on the number of neighbors allowed in $k-$NN; we vary that number from 11 to 30.

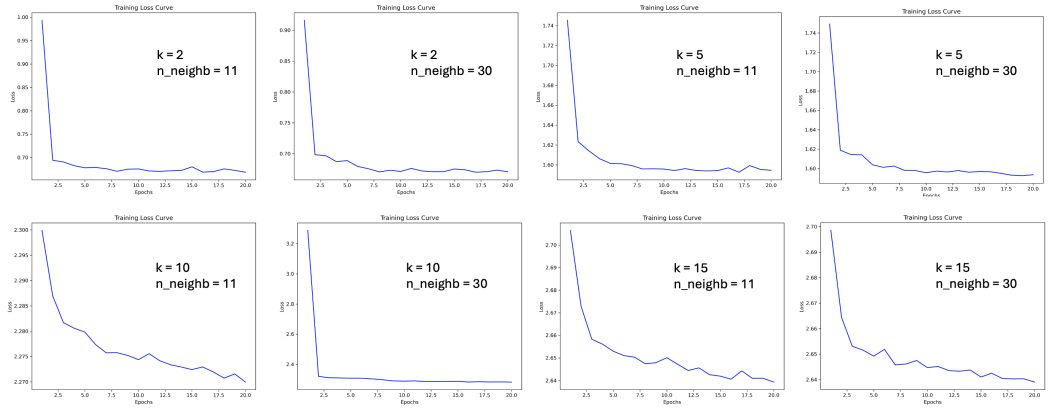

Figure 4: Training curves for our models. Each model was trained for 20 epochs.

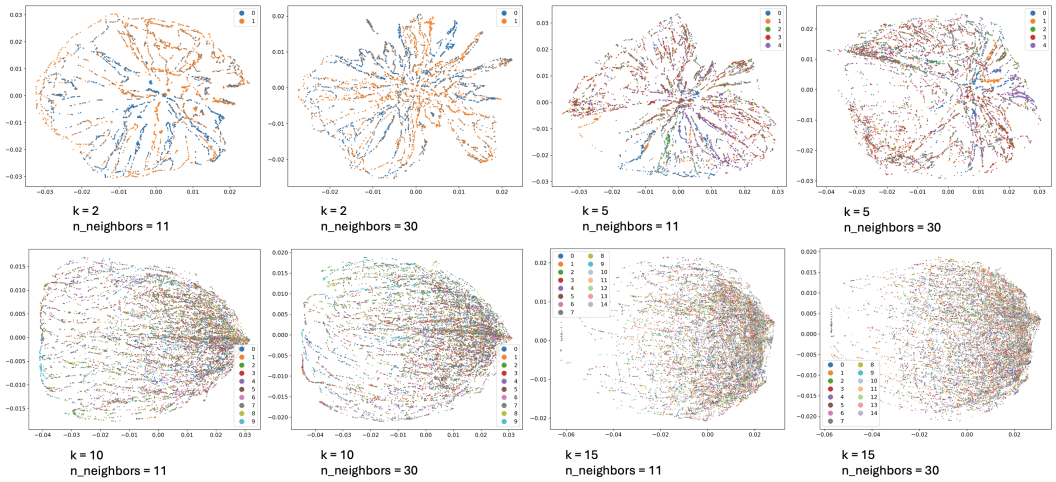

Figure 5: Generated embeddings visualized using PHATE. We visualize the embeddings of one event; each point represents a recorded hit.

5 Conclusion

In this study, we investigated graph-based deep learning methods for clustering calorimeter data in preparation for the upcoming Electron Ion Collider (EIC). Our dynamic graph neural network, which leverages a trainable adjacency matrix, was evaluated against traditional k-means clustering. Although our model did not outperform the k-means baseline, our investigation provided valuable insights into the limitations and potential areas for improvement in graph-based approaches.

Our results indicate that while the models do not yet learn the complex physics necessary for accurate particle reconstruction, they offer a promising avenue for integrating domain-specific knowledge into the clustering process. Specifically, the sensitivity of the model's performance to the number of GravNet layers and the number of neighbors in k-NN suggests that further tuning and optimization of these hyperparameters could enhance the model's capabilities.

Future research should focus on improving the integration of domain knowledge within the graph construction process and exploring more sophisticated architectural variations. Additionally, expanding the dataset to include a more representative variety of particle interactions and incorporating more realistic simulations will be crucial for advancing the practical applicability of these models.

Ultimately, while our current implementation falls short in terms of raw performance, it opens the door to new research directions and underscores the potential of graph-based methods in particle

physics. The continued exploration of these methods may lead to more robust and scalable solutions for high-energy physics challenges.

References

[1] F. Bock et al. "Design and simulated performance of calorimetry systems for the ECCE detector at the electron ion collider". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 1055 (Oct. 2023), p. 168464. ISSN: 0168-9002. DOI: 10.1016/j.nima.2023.168464. URL: http://dx.doi.org/10.1016/j.nima.2023.168464.

[2] Gage DeZoort et al. "Charged Particle Tracking via Edge-Classifying Interaction Networks". In: *Computing and Software for Big Science* 5.1 (Nov. 2021). ISSN: 2510-2044. DOI: 10.1007/s41781-021-00073-z. URL: http://dx.doi.org/10.1007/s41781-021-00073-z.

[3] R. Frühwirth. "Application of Kalman filtering to track and vertex fitting". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 262.2 (1987), pp. 444–450. ISSN: 0168-9002. DOI: https://doi.org/10.1016/0168-9002(87)90887-4. URL: https://www.sciencedirect.com/science/article/pii/0168900287908874.

[4] Kevin R. Moon et al. "Visualizing structure and transitions in high-dimensional biological data". In: *Nat Biotechnol* 37.12 (2019), pp. 1482–1492.

[5] Daniel Murnane et al. *HGS-HIRe Power Week - an ePIC competition*. Accessed: YYYY-MM-DD. 2023. URL: https://kaggle.com/competitions/hgs-hire-power-week-an-epic-competition.

[6] Shah Rukh Qasim et al. "Learning representations of irregular particle-detector geometry with distance-weighted graph networks". In: *The European Physical Journal C* 79.7 (July 2019). ISSN: 1434-6052. DOI: 10.1140/epjc/s10052-019-7113-9. URL: http://dx.doi.org/10.1140/epjc/s10052-019-7113-9.

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

Appendix

The LFHCal

The Longitudinally separated Forward Hadronic Calorimeter (LFHCal) The LFHCal (shown in Figure 6) is positioned at $z = 3.28$ m from the center of the detector. It is made up of two half disks on wheels, each of radius 2.6 m. Most of the calorimeter is made up of 8-tower modules stacked in a lego-like manner for alignment and stability. Each tower has an active depth of $\Delta z = 1.4$ m, consists of 70 layers of alternating 1.6 cm absorber and 0.4 cm scintillator material and has transverse dimensions of 5×5 cm^2. Within a tower, ten consecutive fibers are read out by a single SiPM (a Silicon PhotoMultiplier which transforms photon signals it absorbs into a current pulse made up of electrons). The detector consists of 63,320 readout channels grouped in 9,040 read-out towers. It measures the energy of hadronic particles created in the collision in the forward (hadron-going) direction.

More related work

Traditional approaches In particle physics, traditional clustering methods, such as combinatorial search algorithms and Kalman filters, have long been employed to group and analyze complex event data from detectors [3]. Combinatorial search techniques systematically explore the possible combinations of detected particle tracks to identify clusters corresponding to specific particle interactions. Meanwhile, Kalman filters are utilized to refine these track estimates by incorporating measurement uncertainties and temporal dynamics, enabling more accurate predictions of particle trajectories. However, these traditional methods often struggle with scalability, particularly as the volume and complexity of data increase in modern experiments. The computational overhead of combinatorial searches can become prohibitive with larger datasets, while Kalman filters may require significant tuning and computational resources as the number of particles in an event grows. This limitation

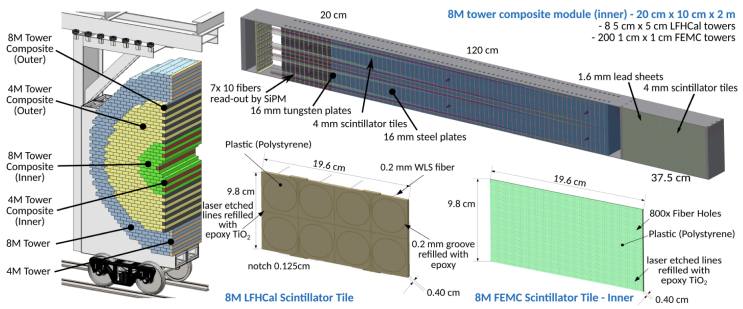

Figure 6: Design pictures of the forward calorimeter assembly (left), 8-tower module design (top right) and single scintillator plates of the LFHCAL (bottom middle) and FEMC (bottom right) for an 8M tower module with embedded wavelength shifting fibers. [1]

	event	hit_number	N	E	T	ix	iy	iz	posx	posy	posz	clusterID
0	90000	0	2257	1.755586	15.13	63	53	5	12.62984	-17.425	457.27997	9.054685e+18
1	90000	1	2257	1.158431	15.75	63	53	6	12.62984	-17.425	489.27997	9.054685e+18
2	90000	2	2257	0.995533	15.14	63	52	5	12.62984	-12.575	457.27997	9.054685e+18
3	90000	3	2257	0.578184	18.26	61	59	4	12.58656	12.575	441.27997	9.054685e+18
4	90000	4	2257	0.450340	16.18	62	53	6	17.54328	-17.425	489.27997	9.054685e+18

Figure 7: Head of the toy dataset. `event` is the event's unique ID, `hit_number` is the recorded cell's ID, `E` is the energy deposited in the cell in GeV, `T` is the time passed since the start of the event, `ix, iy, iz` are the module index of the pixel, `posx,posy,posz` are the pixel x, y, z positions, and `clusterID` is a hashed identifier for the cluster.

in scalability highlights the need for innovative approaches, motivating the exploration of machine learning techniques that can handle vast amounts of data more efficiently and adaptively, thereby enhancing the clustering process in particle physics.

Interaction Networks Interaction Networks (IN) have been applied to particle tracking [2] and could similarly be applied to particle reconstruction. We represent particle trajectories as edges in the graph. Given a set of hypothesized edges, we design an edge-classifying GNN that finds the real edges. This method can be subdivided into three distinct steps: 1) graph construction, 2) edge classification via the calculation of edge weights, and 3) track building leveraging the determined edge weights. We represent the hits as nodes with spatial features and attribute to the edges some geometric features. The training target is a vector $y \in \mathbb{R}^{n_{\text{edges}}}$, where $y_e = 1$ when e connects two hits associated with the same particle. During the graph construction step, a truth filter (e.g. noise filter, same-layer filter) is applied to the hits to output a set of nodes, and edge assignment algorithms (geometric, geometric and pre-clustering, or geometric and data-driven) are deployed. The graph construction is evaluated by efficiency $\left(\frac{N_{\text{true}}^{\text{reconstructed}}}{N_{\text{true}}^{\text{possible}}}\right)$ and purity $\left(\frac{N_{\text{true}}^{\text{reconstructed}}}{N_{\text{total}}^{\text{reconstructed}}}\right)$. This method requires that the original input graph, for which the construction is not always obvious. We now consider approaches that do not require an input graph.

Additional results

We include some experiments where the dataset is further split and we consider each "slice" in the z direction as an independent event. The model is trained for 400 epochs, but the loss does not seem to converge (see Figure 9). The results, shown in Table 2, were slightly better, but still not quite satisfying. We visualize the graph embedding in Figure 10.

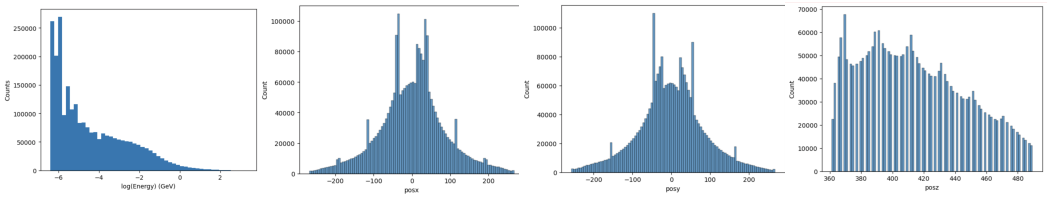

Figure 8: Energy and x,y,z coordinate distributions across events. The energy distribution is a long-tail, with most of the hits having a tiny amount of energy, and a few hits having a huge amount of energy. Note that the x and y axes are bounded by $(-270, 270)$, while the z axis is bounded by $(360, 490)$.

	Test accuracy	Homogeneity	Completeness	V-Score
Z-faces method	$18.33 \pm 2.64\%$	0.048 ± 0.01	0.060 ± 0.01	0.053 ± 0.01

Table 2: Test accuracy, homogeneity, completeness, and Validity Scores. The results are averaged over three runs.

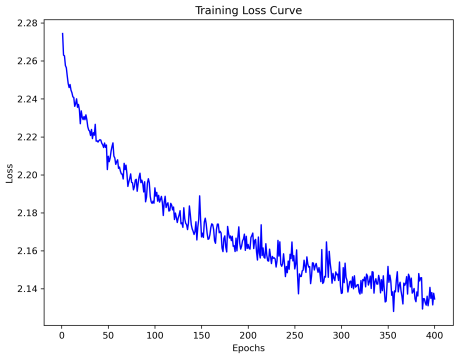

Figure 9: Training loss for 400 epochs. The loss does not really converge yet.

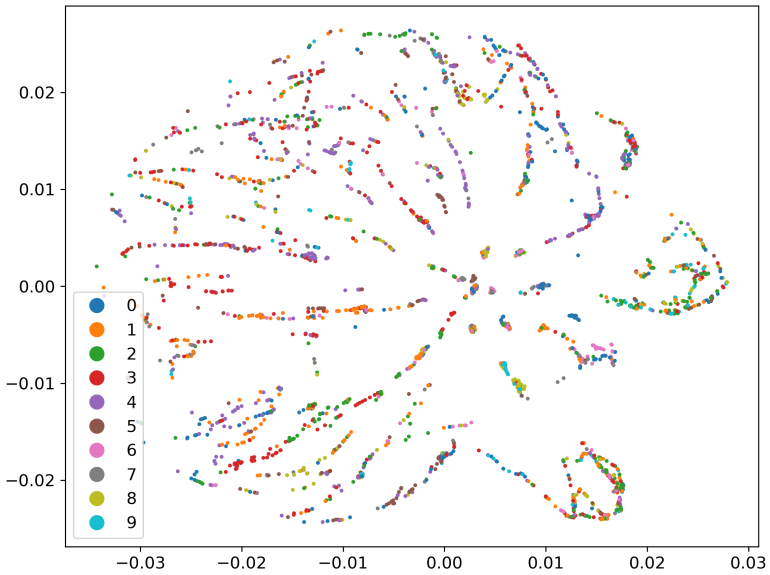

Figure 10: Graph embeddings visualized using PHATE.