

Segmentation of EM showers for neutrino experiments with deep graph neural networks

V. Belavin,^a E. Trofimova,^{a,b,1} A. Ustyuzhanin^a

^aLaboratory of methods for Big Data Analysis, National Research University Higher School of Economics, Pokrovsky Boulevard 11, Russia

^bSkolkovo Institute of Science and Technology, Bolshoy Boulevard 30, bld. 1, Russia

E-mail: etrofimova@hse.ru

ABSTRACT: We introduce a first-ever algorithm for the reconstruction of multiple showers from the data collected with electromagnetic (EM) sampling calorimeters. Such detectors are widely used in High Energy Physics to measure the energy and kinematics of in-going particles. In this work, we consider the case when many electrons pass through an Emulsion Cloud Chamber (ECC) brick, initiating electron-induced electromagnetic showers, which can be the case with long exposure times or large input particle flux. For example, SHiP experiment is planning to use emulsion detectors for dark matter search and neutrino physics investigation. The expected full flux of SHiP experiment is about 10^{20} particles over five years. To reduce the cost of the experiment associated with the replacement of the ECC brick and off-line data taking (emulsion scanning), it is decided to increase exposure time. Thus, we expect to observe a lot of overlapping showers, which turn EM showers reconstruction into a challenging point cloud segmentation problem. Our reconstruction pipeline consists of a Graph Neural Network that predicts an adjacency matrix and a clustering algorithm. We propose a new layer type (EmulsionConv) that takes into account geometrical properties of shower development in ECC brick. For the clustering of overlapping showers, we use a modified hierarchical density-based clustering algorithm. Our method does not use any prior information about the incoming particles and identifies up to 87% of electromagnetic showers in emulsion detectors. The achieved energy resolution over 16,577 showers is $\frac{\sigma_E}{E} = (0.095 \pm 0.005) + \frac{(0.134 \pm 0.011)}{\sqrt{E}}$. The main test bench for the algorithm for reconstructing electromagnetic showers is going to be SND@LHC.

¹Corresponding author.

Contents

1	Introduction	1
2	Related work	3
3	Dataset description	5
4	Reconstruction algorithm	5
4.1	Graph construction	6
4.2	Edge classification	8
4.2.1	Graph convolution block	8
4.2.2	Binary classification block	8
4.2.3	Solving an issue of slow receptive field growth: EmulsionConv	9
4.2.4	Summarised architecture	10
4.3	Showers clusterization	10
4.4	Clusters classification	13
4.5	Kinematic reconstruction	13
5	Experiments and results	14
5.1	Architecture evaluation	15
5.2	Clusterization	16
5.3	Classification of clusters	17
5.4	Energy reconstruction	17
6	Conclusion and perspectives	18
A	Tracklet pairs energy and likeliness estimates	20
B	Grid search of optimal parameters	21
C	Different multiplicities	21

1 Introduction

Electromagnetic (EM) showers are produced by interactions of incoming particle decay products with the photographic plates of emulsion cloud chamber (ECC) bricks [1]. The ECC brick has a modular structure made of a sequence of lead plates interleaved with emulsion films (figure 1). It combines the capability of high-precision tracking of nuclear emulsion films and the large stopping power from the passive material [2]. EM showers reconstruction algorithm is needed to accurately

estimate the decay point, full momentum, and energy of the incoming particle within the brick from the tracking data collected with ECC brick.

The ECC has been used in the Oscillation Project with Emulsion-Tracking Apparatus (OPERA) experiment. OPERA collected data for five years, from 2008 to 2013, and discovered muon to tau neutrino oscillations in appearance mode [3]. One of the future experiments, SHiP [4], is planning to follow the same principle and a similar design as the OPERA experiment. An expected full flux of the particles passing through SHiP detectors will be about 2×10^{20} protons over five years [5]. That is about 50-300 showers per brick.

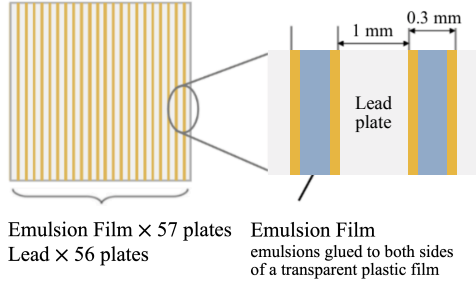


Figure 1: Sectional emulsion brick. The brick consists of 56 lead plates and 57 plastic plates with nuclear photographic emulsions glued on both sides [6].

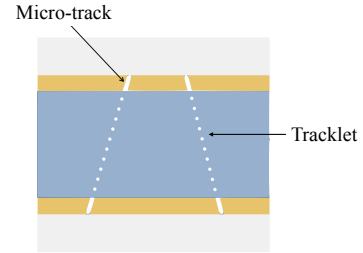


Figure 2: Micro-track and tracklet in emulsion film definition.

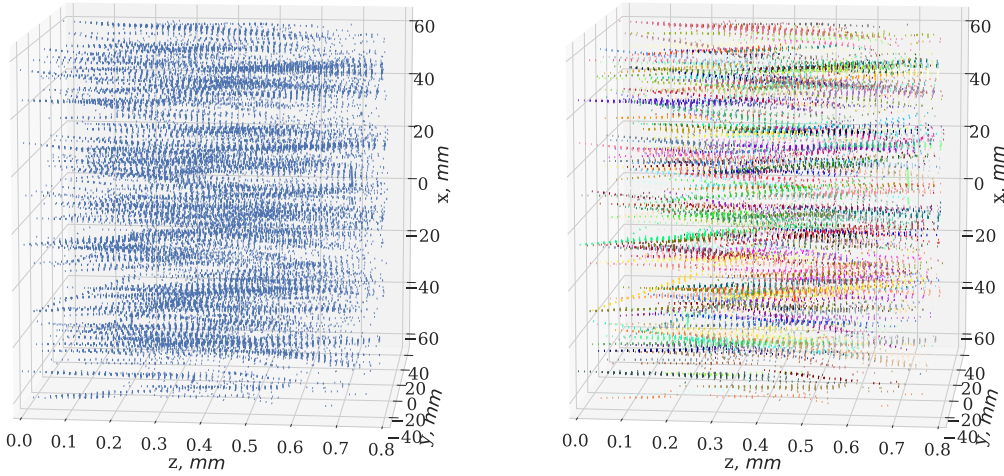
Reconstruction of electromagnetic showers starts with the off-line data taking, performed by fully automated optical microscopes [7, 8]. The scanning system provides three-dimensional spatial information of particle tracks by identifying micro-tracks in emulsion films and connecting them across the plastic base to form tracklets (figures 2). Due to the long exposure time (in SHiP ECC bricks are planned to be replaced twice a year [9]), showers could overlap. These overlaps make it difficult to correctly determine a mapping between tracklets and showers and, consequently, to recover the properties of the initial particle.

A rapidly emerging field of Deep Learning known as Graph Neural Networks (GNN) [10–15] provide an effective approach to analyse unordered data with an inner structure that can be expressed as a graph. As a result, there is a growing interest in the application of this type of networks in high energy physics (HEP) problems. In our work, we use GNN to predict an adjacency matrix for the clustering algorithm. The key motivation for using GNN is the highly structured nature of the data associated with the EM shower development in the detector.

In this paper, we (1) propose a new type of layer (EmulsionConv) for GNNs that utilizes prior knowledge of the physical problem, (2) develop an adapted version of the HDBSCAN algorithm [16], (3) we validate our pipeline on the semantic segmentation of overlapping showers, i.e., assigning each track shower label (figure 3).

This paper is structured as follows. In section 2 we outline the literature about EM showers reconstruction algorithms and Graph Neural Networks applications for the calorimetry and tracking. In section 3, we introduce the dataset used to perform experiments. In section 4, we describe an algorithm for showers segmentation. In section 5 we present experimental results that demonstrate

the practical viability of the proposed method. ¹



(a) Unclustered simulated showers: all tracklets with the same color. (b) Clustered showers: tracklets are colored according to the Monte-Carlo ground truth shower label.

Figure 3: EM showers in an emulsion brick.

2 Related work

Several problems can compromise the reconstruction of electromagnetic showers in the SHiP and SND@CERN experiments. One group of problems is connected with background noise rejection. Another one, that we address in this work is EM showers overlapping. The largest source of background is expected from pass-through muons. Other possible sources of noise are instrumental background, cosmic rays, EM showers induced by the deep inelastic scattering of muons and coherent neutral current neutrino scattering, with the production of single neutral pion [17]. Some of these problems were already addressed; for example, in [18, 19] authors propose algorithms for instrumental background rejection, in [2] π^0/e -separation is studied, and in [20] active muon shield is proposed to deflect the muons out of the acceptance of the spectrometer. An efficient algorithm for electron showers separation and reconstruction would also allow to further improve algorithms for background EM showers rejection. We discuss some works on EM showers reconstruction in more detail below.

To the best of our knowledge, there was no prior work that addressed the problem of multiple showers reconstruction. The most closely related works are focused on the separation of two electromagnetic or electromagnetic-hadronic showers for highly granular silicon calorimeter [21]. Another work [22] provides an algorithm for the separation of up to three showers in the cellular gams-type calorimeter. In contrast, we tackle the problem of the separation of the variable number of showers inside the sampling emulsion calorimeter.

¹The code for this work is available in [23] under MIT licence

In [2], the algorithms for electron shower reconstruction have been developed and are applied to study the electron/pion separation and the shower energy estimation in an emulsion brick. The algorithm iteratively matches each tracklet to tracklets in the downstream films based on specified angular and position requirements. Extrapolation of the tracklet candidate is allowed at most for three films.

In [18], authors solve the problem of shower reconstruction in the presence of a dominated instrumental background due to the ambient radioactivity and cosmic rays. The algorithm for one shower reconstruction is based on prior knowledge about the initial point and shower direction, and utilized Boosted Decision Trees from TMVA [24] to classify all tracklets as a signal or background. For energy reconstruction, a linear regression on the number of selected tracklets is applied. The achieved energy resolution is $\frac{\sigma_E}{E} = (0.28 \pm 0.09) + \frac{(0.09 \pm 0.04)}{\sqrt{E}}$. In [19], authors also address the problem of electron shower identification in the presence of an instrumental background that is orders of magnitude larger than the signal. They propose and study new topological variables that are used in Random Forest for efficient background rejection. They assume many showers in brick with fixed energy on the level of 6GeV but do not address the problem of showers overlap and focus solely on the identification of segments of the showers in an environment with high noise density.

Similarly to [18], [25] presents an algorithm for background classification that does not rely on the information of the shower origin. It also utilizes Boosted Decision Trees, followed by the Conditional Random Field model for pattern identification. The achieved energy resolution is 0.27. This approach is similar to ours in the sense of the absence of prior information about showers origin. However, the authors are not solving the problem of showers semantic segmentation.

There are also many works that propose algorithms for electromagnetic shower reconstruction with deep learning techniques. For example, [26–28] use convolutional neural networks for shower reconstruction, as well as for the classification of the type of input particle.

In [14] and [15], authors justify the use of GNNs for particle tracking and reconstruction in HEP. The authors describe how HEP reconstruction tasks could be reformulated into problems involving graphs. For example, track search could be formulated as a classification of edges of the graph and jet tagging as a regression of graph characteristics. The authors also note the practical advantages of using GNNs. The computational performance of many established reconstruction algorithms does not scale well with the increasing collision complexity of physical events, while GNNs can scale better.

In [29], authors apply edge classification with clustering post-processing for the task of particle tracking. Their work is similar to ours in the sense that they are using GNN to classify edges for the identification of the doublets and triplets. Track labelling task is solved with Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [30], using predicted edges scores. We show that a naive implementation of their approach, demonstrated on a 10 layers detector, does not function well when applied to 56 layers, likely due to vanishing gradients over too many message-passing steps, the issue that we study in section 5.1. We propose a heuristic procedure for graph construction and a new layer type for GNN for the fast growth of the receptive field.

And in [31], the authors demonstrate that Graph Neural Networks have promising results for tracking and calorimetry in high energy physics. They use GNN to classify edges in a graph and combine it with a connecting-the-dot algorithm for tracking problems and calorimeter cluster problems for single-particle samples. In comparison, we are solving a clusterization problem for

multiple particle samples with a much larger diversity of clusters and possible overlaps.

This work aims to recover multiple showers in a sampling emulsion calorimeter with GNNs while achieving the same energy resolution as one shower reconstruction algorithm for the same task. In our approach, we do not use information about shower origin or direction that authors of [18] inferred from Changeable Sheets (CS) doublets [32], a pair of nuclear emulsion films attached to the downstream side of the ECC brick.

3 Dataset description

Tracklets, reconstructed by automatic scanning system, are represented by tracklet position (x , y , z coordinates) and its slope ($\theta_x = p_x/p_z$ and $\theta_y = p_y/p_z$, where $\vec{p} = (p_x, p_y, p_z)$ is the particle momentum). Tracklets exist only inside the brick emulsion. That leads to the following constraints on tracklets coordinates: $x \in [-62500\mu\text{m}, 62500\mu\text{m}]$, $y \in [-49500\mu\text{m}, 49500\mu\text{m}]$; $z = 1293k\mu\text{m}$, $k \in \{0, \dots, 56\}$ (figures 4, 5). We also preselect tracklets in accordance with visibility conditions as in [33]. We discard tracklets produced by low momentum particles (less than 30 MeV) and with high angle of traversal of the emulsion film ($\sqrt{(\arctan \theta_x)^2 + (\arctan \theta_y)^2} > 1$ rad), because they could not be reconstructed by the scanning microscope. Our target variables are energy, momentum, and direction of the initial electrons. To sum up, the data has the following format: one matrix where each row contains tracklet information ($\{x, y, z, \theta_x, \theta_y, \text{shower_id}\}$) and another matrix where each row contains shower information ($\{\text{shower_id}, x^{\text{init}}, y^{\text{init}}, z^{\text{init}}, \theta_x^{\text{init}}, \theta_y^{\text{init}}, E_{\text{true}}\}$).

To assess the performance of our algorithm, we have generated 16,577 electron-induced showers using FairShip framework [34][35]. We modify FairBoxGenerator class from FairROOT to better match one-dimensional distributions of energies and polar angles of electrons to the ones that we expect to observe in neutrino interactions in SHiP. To simulate energy, we have used gamma distribution with parameters $\alpha = 1.4, \beta = 0.5$. To simulate polar angle, we have used log-normal distribution with parameters $\nu = 0, 3, \sigma = 0.7$ to sample pseudorapidity (η), which is connected with polar angle (θ) by a simple relation: $\theta = 2 \arctan(\exp(-\eta))$. This leads to the energy distributed from 0 to 40 GeV with a peak at 6.4 GeV (figure 6).

For our experiments, we aggregate showers into a dataset with a variable multiplicity of 50-300 showers per brick that serves as a proxy for a realistic scenario, where we expect a random number of showers per brick. The resulting dataset consists of 100 emulsion bricks EM showers data with 50-300 showers in each brick (figure 7). In this work, we consider electromagnetic showers data to be cleaned from the background tracklets.

4 Reconstruction algorithm

Our algorithm for EM showers reconstruction comprises five steps. **First**, we heuristically construct a directed graph, where each tracklet is assigned with a vertex. **Second**, we predict the probability for each edge to connect a pair of tracklets from the same shower with a neural network. **Third**, we use transformed probabilities from the previous step as weights in clusterization. **Forth**, because cluster assignment is ambiguous, we also introduce a simple boosting tree classifier to select clusters (that we also denote in the text as “reconstructed showers”) for the subsequent kinematic reconstruction.

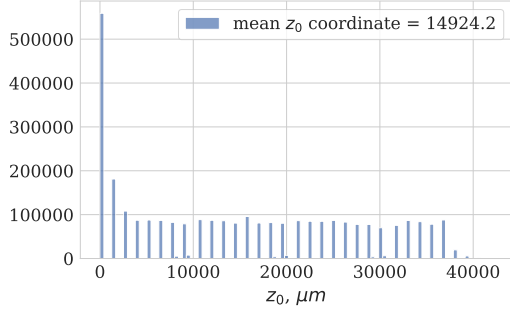


Figure 4: Distribution of the showers z^{init} positions.

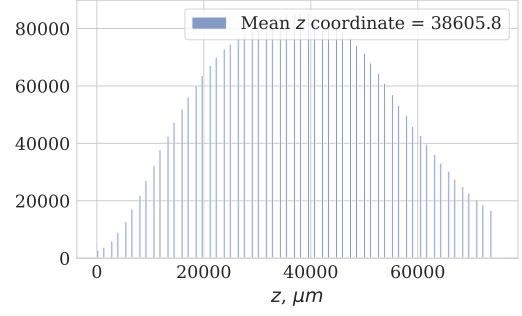


Figure 5: Distribution of all tracklets z coordinates.

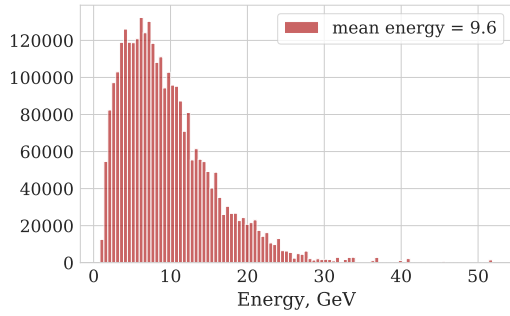


Figure 6: Distribution of the showers true energy.

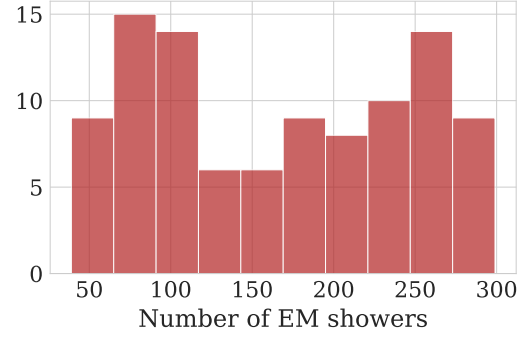


Figure 7: Distribution of the number of shower per brick (realistic case).

And, **fifth**, for reconstructed showers, we estimate kinematic variables such as the position of decay, direction and energy of the primary particle that initiated the shower.

4.1 Graph construction

We build a directed graph with vertices representing tracklets. To decide whether to connect two vertices with an edge or not, we introduce a distance metric defined on pairs of tracklets that we call “integral distance”. Integral distance is equal to the area formed by the union of the extrapolations of tracklets (figure 8).

If we assume that the one tracklet is described by the parameters $x_1, y_1, z_1, \theta_{x_1}, \theta_{y_1}$, and the other one by $x_2, y_2, z_2, \theta_{x_2}, \theta_{y_2}$, then this distance is expressed by the following integral, which can be evaluated in closed form:

$$\text{IntDist} = \int_{z_2}^{z_1} \left((z(\theta_{x_2} - \theta_{x_1}) - (x_1 - x_2 + \theta_{x_2}(z_2 - z_1)))^2 \right)^{\frac{1}{2}} dz + \int_{z_2}^{z_1} \left((z(\theta_{y_2} - \theta_{y_1}) - (y_1 - y_2 + \theta_{y_2}(z_2 - z_1)))^2 \right)^{\frac{1}{2}} dz \quad (4.1)$$

The integral distance combines the angle difference and impact parameter difference into one scalar metric by calculating area projections on xz and yz planes.

The edge is directed from a tracklet with a smaller z coordinate to a tracklet with a larger z coordinate. An edge could connect tracklets not only from two successive plates but from any pair of plates because the particle could pass several layers of the detector without leaving any reconstructible tracklets. For each tracklet, only 10 outgoing and 10 incoming edges with the smallest value of the IntDist are left. We have chosen the number of incoming and outgoing edges by assessing the segmentation of the ground truth showers within the constructed graph. I.e. for each ground truth shower, we calculate the number of clusters (connected components in the graph) and the relative size of the largest clusters to the total number of tracklets in the original shower. In figure 9 we plot the 80th percentile of the number of clusters. In the ideal case, one ground truth shower should be associated with exactly one cluster. So we have to choose the number of edges that minimize this metric. On the other hand, we can tolerate situations when one ground truth shower is associated with one large cluster and several tiny ones. Thus, to distinguish situations when we have several clusters of approximately equal size (in terms of the number of tracklets) and situations when we have one large cluster, we also calculate the relative size of the largest cluster. In figure 10 we plot the 20th percentile of the relative size of the largest cluster to the total number of tracklets in the original shower as the function of the number of edges. As one can notice, when the number of edges is larger than 5, we observe neither an improvement in the number of clusters nor a significant increase in the relative size of the largest cluster. Considering this and considering computational demands during neural network training with dense graphs, we have decided to keep the number of incoming and outgoing edges equal to 10.

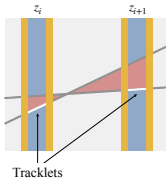


Figure 8: Integral distance (red area).

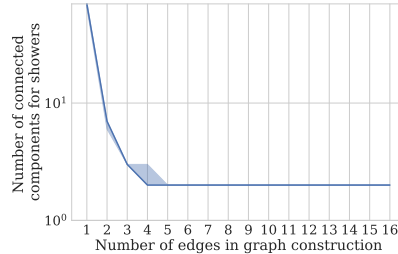


Figure 9: 80th percentile of number of clusters per shower as a function of number of edges.

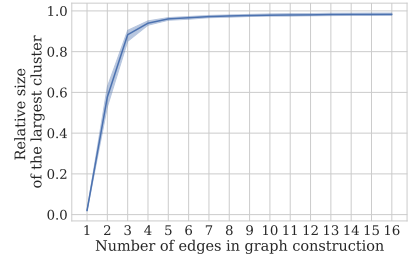


Figure 10: 20th percentile of the relative size of the largest clusters as a function of number of edges.

To further lower computational costs and reduce the size of the neural network used, we perform feature engineering and combine features that describe vertices and edges. In clustering algorithms, the choice of the feature space and the measure of the distance between objects is of critical importance. From first principles of the proposed clustering algorithm, good features should be approximately equal for tracklets from the same shower and take as distinct values as possible on tracklets from separate showers. Experimentally, the following vertex features, presenting the azimuthal angle, cartesian x , y coordinates projections on the z -axis and their combinations, were found to improved the algorithm's quality over the raw data case:

1. initial features: $x, y, z, \theta_x, \theta_y$

2. trigonometric features: $\phi = \arctan\left(\frac{y}{x}\right)$, $\frac{\sqrt{x^2+y^2}}{z}$, $\frac{x}{z}$, $\frac{y}{z}$, $\frac{(\sin(\phi)+\cos(\phi))}{\phi}$

Edge features include:

1. IntDist (eq. 4.1),

2. IP (impact parameter) projections on the X and Y axes for tracklet pairs,

$$\text{IP projection on the X axis} = \frac{x_1 - x_2 - (z_1 - z_2) \cdot \theta_{x_i}}{z_1 - z_2}, \quad (4.2)$$

$$\text{IP projection on the Y axis} = \frac{y_1 - y_2 - (z_1 - z_2) \cdot \theta_{y_i}}{z_1 - z_2}, \quad (4.3)$$

where $i \in 1, 2$

3. tracklet pairs energy and likeliness estimates (eq. A.2).

As a result, 10 and 7 features are used for vertex and edge description respectively.

4.2 Edge classification

We use two neural networks to predict the probability that edge connects tracklets from the same shower: a Graph Convolutional Network (GCN) that predicts embeddings of the vertices and a Fully Connected Neural Network (FCNN) that predicts classification scores for edges.

4.2.1 Graph convolution block

To predict meaningful embeddings of vertices, we are using Graph Convolution Network. GCN is a special type of neural network that generalizes convolution operations on regular N-dimensional grids to the unstructured grids described by a graph. The proposed GCN includes two components: an encoder for input graph transformation in the latent representations of each vertex and each edge and a module that performs message-passing for latent features updating.

In particular, we use EdgeConv layer. EdgeConv is proposed in [36] for the segmentation of 3D clouds. The key idea is to modify a way to compute messages. They propose to use relative information about vertices in the message-passing step, i.e. using the following formula to compute messages: $m_{vv'} = M(h_v, h_{v'} - h_v, e_{vv'})$, where h_v and $e_{vv'}$ are the latent representation of vertex and edge, respectively, and M is a differentiable function, for example, neural network.

4.2.2 Binary classification block

A Fully Connected Neural Network takes the vertices embeddings from the GCN as an input. Then, it predicts the probability for each edge to connect tracklets from the same shower.

The distribution of edges classes is highly imbalanced (approximately 10:1). Thus, we are using focal loss [37] during training:

$$\text{FL}(w_{vv'}, y) = - (y(1 - w_{vv'})^\gamma \log(w_{vv'}) + (1 - y)w_{vv'}^\gamma \log(1 - w_{vv'})), \quad (4.4)$$

where $w_{vv'}$ is the output of the model estimating the probability for the class $y = 1$ and γ is the focusing parameter (we choose $\gamma = 3.0$).

4.2.3 Solving an issue of slow receptive field growth: EmulsionConv

Tracklets that are closer to the shower origin, i.e. produced in the early stage of shower development by hard electrons and photons, contain more information about a shower than tracklets produced in the late stages by soft particles. Thus, we need to encourage awareness of the neural network of early-stage tracklets when this network makes predictions about late-stage tracklets. In other words, we need to ensure the fast growth of the receptive field of neural network [38, 39].

In EdgeConv, messages propagation from vertex v to some vertex v' takes as many steps as the length of the shortest path between v and v' . It would take 56 (equal to the number of emulsion layers in the detector) message-passing steps for the network from the last message-passing step to consider early-stage tracklets when making predictions for late-stage tracklets. More generally, without additional tricks, the receptive field of GCNs grows linearly with the number of convolution blocks, which, in combination with vanishing gradient problem [40, 41], leads to the abuse of shallow networks that can not properly propagate information in large graphs. We propose the EmulsionConv layer in which we modify the algorithm to collect messages and update hidden representation vectors (h_v) of vertices for each emulsion layer separately. EmulsionConv aims to solve the problem of the slow growth of the receptive field and computation burden of such inefficient updates in vanilla GCN, by exploiting the tree-like structure of electromagnetic showers. First, the proposed layer splits edges into 56 groups, grouping edges $e_{vv'}$ by the z coordinate of the tracklet associated with vertex v' . Second, EmulsionConv performs the full message-passing within one group before proceeding to the next group. Because of the lesser computational parallelism and inability to fully utilize GPU parallelization, every single step of EmulsionConv takes more time than one step of EdgeConv. However, in one step of EmulsionConv, we gain a forward in z receptive field of size 56, which is impossible to achieve with the same number of EdgeConv layers. The output of EmulsionConv at vertex v is an average of messages passed, and updated vertex embeddings for each vertex $v \in N(v')$. The EmulsionConv layer algorithm is summarized in Algorithm 1. We parameterize the message updating function M with a linear layer followed by ReLU activation function.

Algorithm 1 EmulsionConv algorithm

Require: graph $(V, E) = \{v_h, e_{vv'}\}$; M, U – neural networks

Ensure: updated graph $(V, E) = \{v_h, e_{vv'}\}$

- 1: Group pairs of vertices $g_k = \{(vv') \mid z_{v'} = z_k \wedge \exists e_{vv'}\}$ based on 56 unique z_k .
 - 2: **for** $k \in [1, \dots, 56]$ **do**
 - 3: **for** each (vv') in g_k **do**
 - 4: $m_{vv'} = M(h_v, h_{v'} - h_v, e_{vv'})$
 - 5: **end for**
 - 6: $m_{v'} = \text{sum}\{m_{vv'}\}_{v \in N(v')}$
 - 7: $h_{v'} \leftarrow \frac{h_{v'} + m_{v'}}{2}$
 - 8: **end for**
-

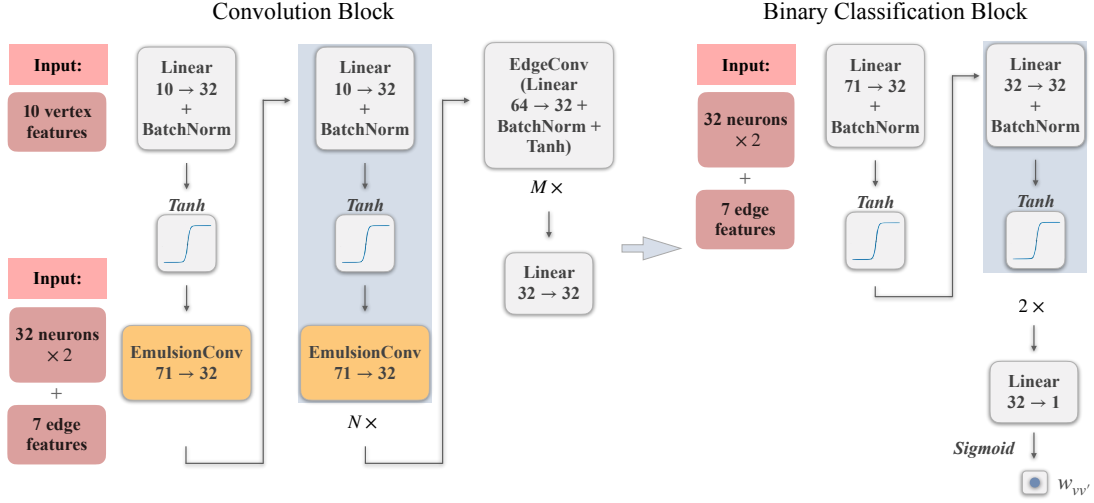


Figure 11: Edge classification neural network architecture.

4.2.4 Summarised architecture

The graph convolution and binary classification blocks are illustrated in figure 11. The first block is composed of N layers of newly proposed EmulsionConv and M layers of EdgeConv. We will choose specific N and M in section 5. Next, the output of the graph convolution block is further passed to the binary classification block, where we concatenate the embedding of vertices and the corresponding edge features. Finally, the binary classifies predicts an edge probability $w_{vv'}$.

4.3 Showers clusterization

For a final separation of the showers, there is a need for an algorithm that can operate with large sparse graphs and that avoid breaking showers during the clustering. To perform clustering, we need to introduce a distance between vertices, close to zero for edges that connect vertices from the same shower and large for those edges that connect vertices from different showers. By default, if there were no edge between two vertices after graph construction, we assume that $d_{vv'} = +\infty$. For all other pairs of vertices, we define distance as a function of $w_{vv'}$:

$$d_{vv'} = \begin{cases} \frac{\arctanh(1 - w_{vv'})}{w_{vv'}}, & \frac{\arctanh(1 - w_{vv'})}{w_{vv'}} < threshold \\ +\infty, & \frac{\arctanh(1 - w_{vv'})}{w_{vv'}} > threshold \end{cases} \quad (4.5)$$

From now on, we will define the shower-candidate as a “cluster” and the ground-truth shower as a “shower”. We introduce a modified version of the HDBSCAN [16] clustering algorithm. We call it an edge weight-based spatial clustering algorithm for graph structures (EWSCAM).

EWSCAM takes as input a graph $\{v, d_{vv'}\}$, the two hyperparameters k (minimum samples core) and \min_{cl} (minimum cluster size) and produces a cluster hierarchy. Following the original

HDBSCAN algorithm, we are transforming the space, defining *mutual reachability distance* between pairs of vertices as follows:

$$d_{vv'}^{\text{mreach}_k} = \max(\text{core}_k(v), \text{core}_k(v'), d_{vv'}), \quad (4.6)$$

where $\text{core}_k(v)$ is a distance from the vertex to the k -nearest neighbour. $\text{core}_k(v)$ shows how dense or sparse the area around the vertex.

We are using the graph $\{v, d_{vv'}^{\text{mreach}_k}\}$ to construct a minimum spanning tree (MST) with the Kruskal algorithm [42], that at each step adds the lowest-weighted edge. MST and the order in which each edge was added define a hierarchical tree structure (dendrogram) of vertices.

The next step is to condense the dendrogram of vertices into clusters. One way to look at this procedure is from a top-down point of view. We are starting from the root of the dendrogram; if both children at the current level have more than min_{cl} vertices, **then** we consider that cluster splits into two *different* clusters. If only one child has more than min_{cl} vertices, then the cluster remains the same, but with fewer vertices. Finally, if both children have less than min_{cl} vertices, then the cluster disappears at this level.

After we have defined the hierarchical tree structure of clusters, we need to extract clusters, or, in other words, choose levels in the dendrogram of clusters where we stop the splitting procedure. We want to choose the most persistent clusters. First, we are defining λ_{birth} , which is equal to the inverse of the mutual reachability distance when the cluster splits off from its parent. Then, *stability* is defined as a sum over inversed weights of edges that were attached to the cluster **within its path in the dendrogram**:

$$stability = \sum_{\{vv'\}} \frac{1}{d_{vv'}^{\text{mreach}_k} - \lambda_{birth}} \quad (4.7)$$

After computation of stabilities, we are going from bottom to top and pruning **the** dendrogram. Suppose the sum of the stabilities of children is greater than the parent cluster stability; in that case, parent vertex stability is set to be equal to the sum of the stabilities of the children. If, on the other hand, **the stability of the parent is greater than the stability summed over its children**, then we prune children. **The EWSCAM algorithm is sketched in Algorithm 2.**

The main difference between our algorithm and HDBSCAN is that in HDBSCAN, authors use the Prim algorithm for Minimum Spanning Tree (MST) construction, whereas in EWSCAM, the Kruskal algorithm is used. Both algorithms produce correct MST but connect vertices in a different order, leading to different linkage and condensed trees. A condensed tree is obtained by assessing the stability of each cluster in relation to the clusters into which it is divided. In our case, it is very important not to over-cluster the showers. **In our experiments, we observe that EWSCAM decreases the number of broken showers (a more formal definition of the broken shower would be given in 4.4) by about four times.**

Algorithm 2 EWSCAM algorithm

Require: graph $\{v, d_{vv'}\}$

k – minimum samples core, parameter for mutual reachability distance computation

\min_{cl} – regulates the minimum cluster size at splitting

Ensure: a dendrogram D with cluster

- 1: Compute mutual reachability distance and construct new graph $\{v, d_{vv'}^{mreach_k}\}$
 - 2: Construct minimum spanning tree (MST), i.e. tree with the lowest sum of $d_{vv'}^{mreach_k}$
 - 3: $S = \{\{v_i\}\}_{i=1}^N$ – initialize set of clusters with each cluster including one vertex
 - 4: Initialize empty dendrogram D
 - 5: **for** each edges $e_{vv'}$ in T_S in increasing order of $d_{vv'}^{mreach_k}$ **do**
 - 6: cluster _{v} \leftarrow cluster from S , that contains vertex v
 - 7: cluster _{v'} \leftarrow cluster from S , that contains vertex v'
 - 8: **if** $\text{len}(\text{cluster}_v) > \min_{cl}$ AND $\text{len}(\text{cluster}_{v'}) > \min_{cl}$ **then**
 - 9: create new cluster cluster _{vv'}
 - 10: set cluster _{v} and cluster _{v'} to be children of cluster _{vv'}
 - 11: add cluster _{vv'} to S , add cluster _{vv'} to D
 - 12: delete cluster _{v} and cluster _{v'} from S
 - 13: **end if**
 - 14: **if** $\text{len}(\text{cluster}_v) > \min_{cl}$ AND $\text{len}(\text{cluster}_{v'}) < \min_{cl}$ **then**
 - 15: append vertices from cluster _{v'} to cluster _{v}
 - 16: delete cluster _{v'} from S
 - 17: **end if**
 - 18: **if** $\text{len}(\text{cluster}_v) < \min_{cl}$ AND $\text{len}(\text{cluster}_{v'}) > \min_{cl}$ **then**
 - 19: append vertices from cluster _{v} to cluster _{v'}
 - 20: delete cluster _{v} from S
 - 21: **end if**
 - 22: **if** $\text{len}(\text{cluster}_v) < \min_{cl}$ AND $\text{len}(\text{cluster}_{v'}) < \min_{cl}$ **then**
 - 23: create new cluster cluster _{vv'}
 - 24: add cluster _{vv'} to S
 - 25: **if** $\text{len}(\text{cluster}_{vv'}) > \min_{cl}$ **then**
 - 26: add cluster _{vv'} to D
 - 27: **end if**
 - 28: delete cluster _{v} and cluster _{v'} from S
 - 29: **end if**
 - 30: **end for**
 - 31: For each cluster in D calculate stability
 - 32: Recursively prune D from bottom to top
 - 33: Return D
-

4.4 Clusters classification

To assess the quality of the EM showers separation, we define recovered, broken, lost, and stuck showers as follows:

- a shower is considered to be *recovered* if one cluster contains more than 90% of its tracklets and it is not broken or lost;
- a shower is considered to be *broken* if the ratio of sizes of the largest cluster, i.e. the cluster containing the maximum number of tracklets, to the second largest cluster less than 2;
- a shower is considered to be *lost* if less than 10% of its tracklets are within all clusters;
- a shower is considered to be *stuck* if it does not fall into any of the above-listed categories (Figure ??).

We estimate the recovered shower's characteristics (decay position of the initial particle and its momentum) by analysing the associated cluster with $> 90\%$ of tracklets. Because these definitions are based on the Monte-Carlo ground truth information, to use our pipeline on real data, we also need a classifier to decide if the cluster produced by the clustering algorithm is recovered or not.

To separate recovered showers from other types of clusters, we use an XGBoost Classifier [43]. Figure 12 shows a cone constructed as follows: the vertex corresponds to the estimated position of the initial particle, the axis corresponds to the estimated direction of the particle (see section 4.5 for a definition). Classifier takes three input variables: the numbers of tracklets in a cone with radii of 10 mrad, 30 mrad, and 50 mrad. The model predicts the probability that the shower is recovered. We have used an XGBoost classifier because tree algorithms provide the best performance for tabular data.

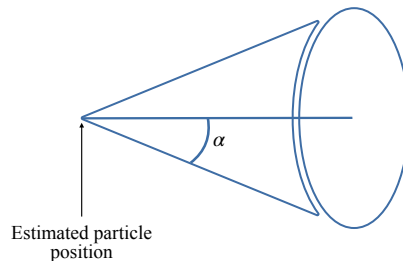


Figure 12: Cone construction for the recovered shower. Cone apex is the estimated particle position, and cone direction is the estimated particle direction.

4.5 Kinematic reconstruction

After the reconstruction of the EM showers, we analyze the physical properties of the successfully reconstructed showers. Figure 13 summarizes successive steps of showers reconstruction, followed by the assessment of the particles' position, direction and energy.

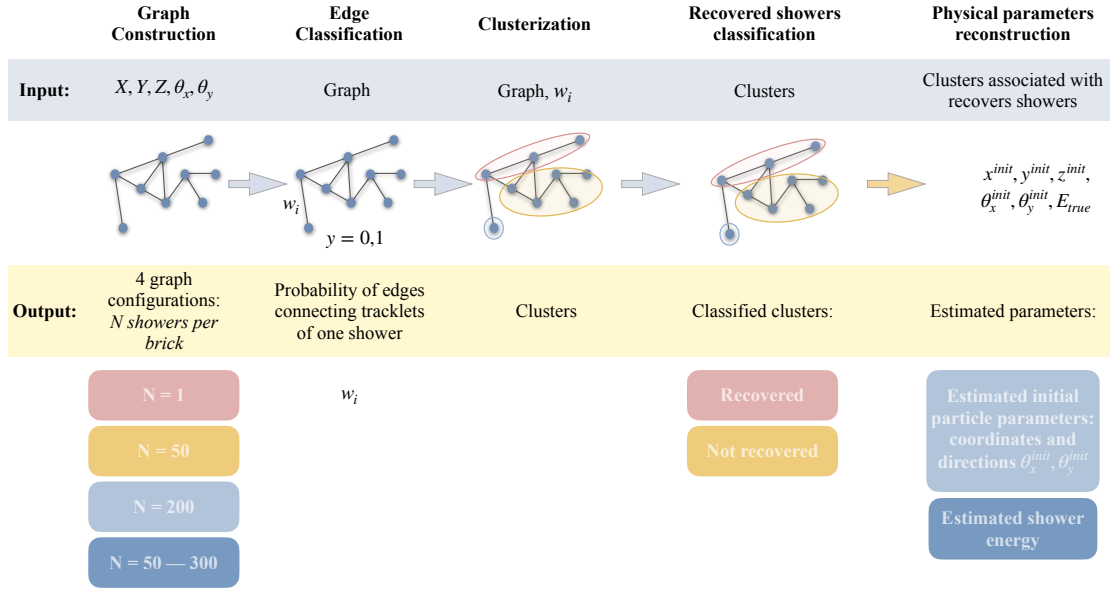


Figure 13: Overall pipeline of the experiments.

The initial particles coordinates ($x^{init}, y^{init}, z^{init}$) are estimated from the median position of the first three tracklets in the cluster along the z axis. The directions θ_x^{init} and θ_y^{init} are inferred by fitting $z_i = \theta_x^{init} x_i + \text{const}$ and $z_i = \theta_y^{init} y_i + \text{const}$ to the first 20 tracklets along the z axis. We estimate the accuracy of the initial particle coordinates and direction reconstruction by computing the mean absolute error between the true and reconstructed values.

We use linear regression to reconstruct the energy because, in general, the response of the electromagnetic calorimeter is linear w.r.t. the energy of the incoming particle [44]. However, a linear dependency of the true energy is corrupted with a high level of noise. Thus, we estimate the energy of the shower with a linear regression trained with the Huber loss [45] to be more robust to outliers. As a predictor, we use the number tracklets in the recovered shower (N_{tr}):

$$E_{rec} = p_0 + p_1 N_{tr} \quad (4.8)$$

We assess the quality of energy reconstruction for recovered showers with energy resolution defined as the standard deviation of the difference between the true and reconstructed energy divided by the true energy.

5 Experiments and results

For all experiments, we split the dataset into ten folds. Six folds are used as a train, three folds as a validation, and one fold as a test. First, we use the train part of the dataset to fit the network and perform a hyperparameter search for clusterization on it. Next, we use the validation part of the dataset for (1) early stopping during the network training, (2) fitting the clusters classifier, (3) fitting

linear regression for energy reconstruction. Finally, we use the test part of the dataset to assess the performance of the whole pipeline (figure 13).

After that, we “rotate” the dataset (i.e. [shift ten folds of the dataset with the offset of one](#)) and repeat experiments. We “rotate” the dataset ten times so that every part is used as a test only once. [The average and standard deviation over these ten measurements are reported in the plots and tables below.](#)

5.1 Architecture evaluation

For the ablation study, to show that the EmulsionConv layer indeed improves edge classification performance in comparison with EdgeConv, we compare six architectures: (1) 8 layers of EmulsionConv (“pure emulsion”), (2) 8 layers of EdgeConv (“pure edge 8”), (3) 56 layers of EdgeConv (“pure edge 56”), (4) 4 layer of EmulsionConv and 4 layers of EdgeConv (“mix equal”), (5) 3 layer of EmulsionConv and 5 layers of EdgeConv (“mix edge”), (6) 5 layer of EmulsionConv and 3 layers of EdgeConv (“mix emulsion”).

We use ROC-AUC (area under the receiver operating characteristic curve, eq. 5.1) metric [46] as a proxy metric to validate different architectures, because [it](#) measures [the](#) quality of ranking, i.e. ensures that probabilities for edges that connect tracklets from different showers are lower than the probabilities for edges that connect tracklets from the same shower. These probabilities are used as edge weights in section 4.3; thus, the ROC-AUC metric indirectly measures the quality of the downstream clusterization.

$$\text{ROC} - \text{AUC} = \frac{\sum_{i=1}^N \sum_{j=1}^N H[y_i - y_j] H[w_i - w_j]}{\sum_{i=1}^N \sum_{j=1}^N H[y_i - y_j]}, \quad (5.1)$$

where H is the heaviside function, $y \in \{0, 1\}$ is a binary label, and $w \in [0, 1]$ is the edge probability predicted by the model.

We train neural networks for 4000 epochs with early stopping with a patience parameter of 100. For optimization we use [the](#) Adam algorithm [47] with [a](#) learning rate of 10^{-3} .

As one can see from figure 14, the best results are achieved with networks either composed of a mix of EmulsionConv and EdgeConv or purely composed of EmulsionConv. Whereas the quality of networks entirely composed of only EdgeConv layers shows a statistically [significant](#) lower performance. Moreover, stacking 56 EdgeConv layers to increase receptive field and expressiveness of the network does not improve [the](#) results. We suggest that it may be due to the overfitting caused by a significant increase in the number of parameters, oversmoothing [48], or vanishing gradients [49]. The combination of EmulsionConv and EdgeConv has higher stability during training and a higher value of validation ROC-AUC (figure 14). [We scrutinize the ROC-AUC score on the validation part of the dataset to assess](#) the possible degradation of quality with the increase of the multiplicity (figure 15). We observe a mild, approximately linear degradation of the network ability to classify edges. For the experiments in the next sections, we are going to use three best networks: (1) 3 layer of EmulsionConv and 5 layers of EdgeConv (“mix emulsion”) (“mix edge”), (2) 4 layer of EmulsionConv and 4 layers of EdgeConv (“mix equal”), (3) 5 layer of EmulsionConv and 3 layers of EdgeConv (“mix emulsion”).

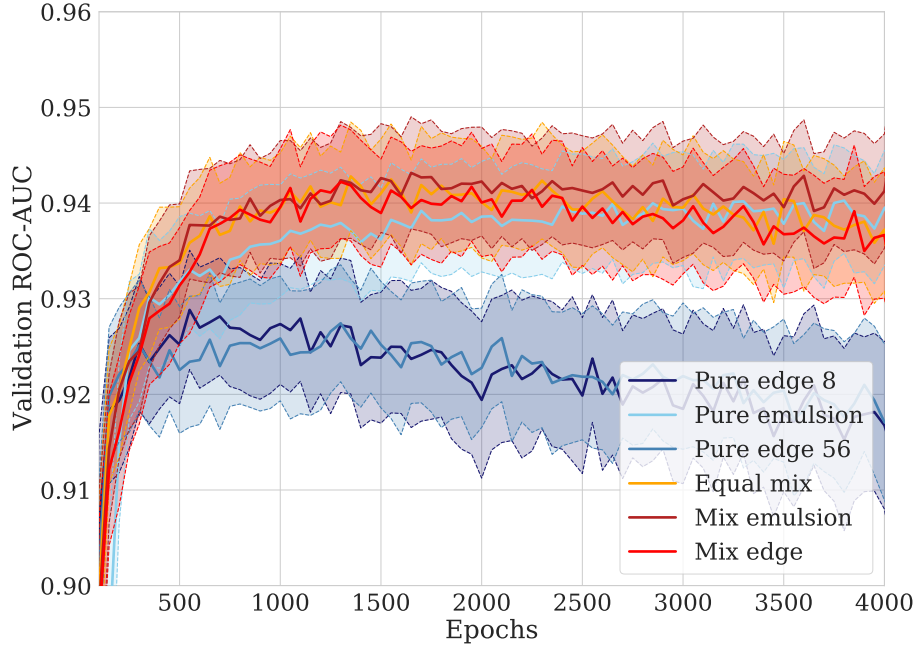


Figure 14: 10-folds cross-validation training curves for six networks.

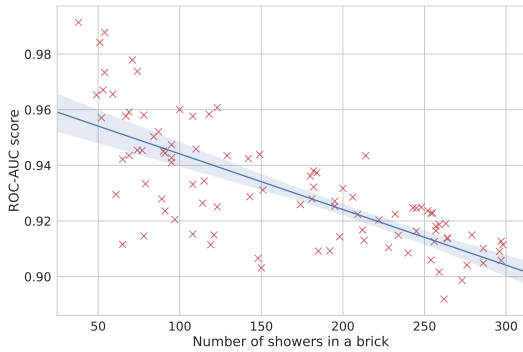


Figure 15: ROC-AUC per brick for the realistic case of the variable number of showers.

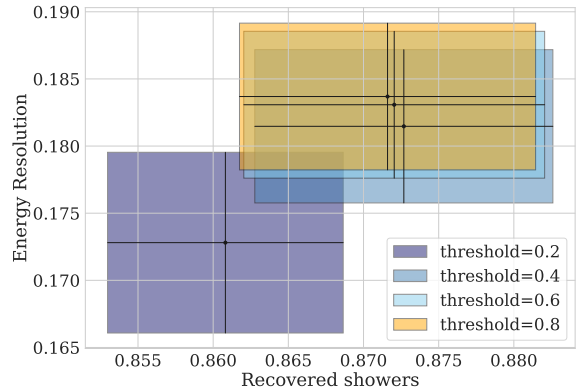


Figure 16: Fraction of recovered showers and energy resolution trade-off for the “mix edge” network for various values of the threshold.

5.2 Clusterization

We optimize the hyperparameters of the EWSCAM algorithm by a grid search by maximizing the percentage of recovered showers. We use the training part of the dataset that we also used for GNN training to find the clustering hyperparameters. The optimal values for minimum samples core (k), threshold, and minimum cluster size (\min_{cl}) are 2, 0.6, and 30 (figure 16, see appendix B for more information on grid search results), respectively. The quality metrics for EWSCAM and HDBSCAN are reported in Table 1; EWSCAM recovers up to 87% of EM showers, depending on the GNN architecture used. These results take into account variability introduced by different

random seeds used for network initialization and dataset shuffling. In our case, where there is a high number of overlapping showers per brick, we highlight the recovered showers metric as the most relevant one.

Table 1: Comparison of the performance of the clustering algorithms reported using 10-fold cross-validation on the dataset composed of 50-300 showers per brick.

	EWSCAM			HDBSCAN
Network Metric	Mix Emulsion	Equal Mix	Mix Edge	Mix Edge
Recovered Showers, %	85.98 ± 3.46	85.97 ± 3.06	86.55 ± 2.21	69.18 ± 5.99
Stuck Showers, %	10.38 ± 4.38	10.53 ± 3.71	9.76 ± 2.74	16.49 ± 9.06
Broken Showers, %	3.19 ± 0.95	3.17 ± 0.73	3.24 ± 0.73	14.18 ± 4.23
Lost Showers, %	0.45 ± 0.29	0.32 ± 0.17	0.44 ± 0.34	0.15 ± 0.31
For Recovered Showers Only				
$MAE_x, \mu m$	154.01 ± 12.77	153.72 ± 12.13	154.63 ± 13.80	146.81 ± 13.95
$MAE_y, \mu m$	151.06 ± 10.21	147.01 ± 12.95	156.47 ± 13.16	147.01 ± 12.95
$MAE_z, \mu m$	802.55 ± 50.39	809.87 ± 73.17	823.49 ± 74.28	724.76 ± 58.80
$MAE_{\theta_x}, \times 10^{-4}$	85.46 ± 4.48	87.04 ± 3.37	86.4 ± 3.81	87.8 ± 4.43
$MAE_{\theta_y}, \times 10^{-4}$	85.7 ± 3.83	86.8 ± 3.30	86.50 ± 4.71	87.80 ± 6.05

5.3 Classification of clusters

For binary classification of the clusters, we train an XGBoost classifier with 300 trees and \max_{depth} of 9 on the validation part of the data and evaluate on the test part. Figures 17 and 18 show the averages and standard deviations for receiver operating characteristic (ROC) and precision-recall (Pr-R) curves on the 10-fold cross-validation. ROC curve illustrates the diagnostic capabilities of the binary classifier by plotting the true positive rate (TPR) versus the false positive rate (FPR) at various threshold settings. The precision-recall curve shows the trade-off between precision and recall, i.e. TPR, for different thresholds. The area under the ROC curve (ROC-AUC) and average precision score (PR-AUC) are equal to 80% and 96%, correspondingly.

5.4 Energy reconstruction

We estimate particle energy as a function of the number of tracklets associated with the shower using Huber regression (figure 19). The regressor is trained on the validation part of the data and evaluated on the test part of the data. We also estimate 95% confidence intervals for coefficients: $p_0 = 0.52 \pm 0.02$ [GeV], $p_1 = 0.0346 \pm 0.0001$ [GeV]. We tried to capture nonlinear dependence by fitting gradient boosting. However, we did not observe statistically significant quality improvement, so we decided to use a more robust and interpretable linear model.

We estimate energy resolution as a function of energy by splitting all showers, sorted by true energy, into ten baskets with an equal number of showers in each. For each basket, we fit a Gaussian

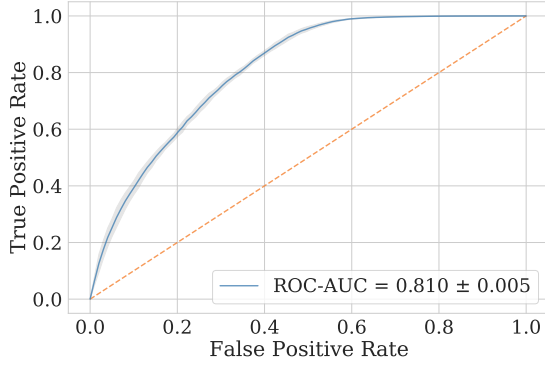


Figure 17: 10-fold cross validated receiver operating characteristic curve. Shaded region corresponds to 1σ confidence intervals.

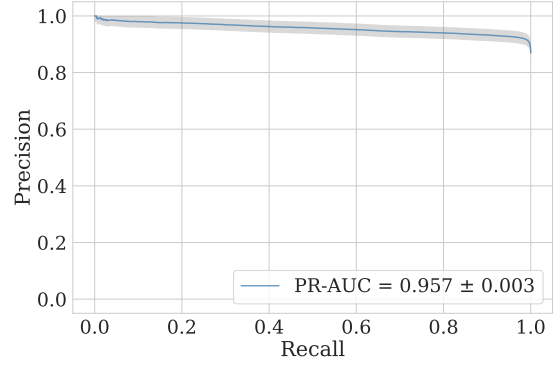


Figure 18: 10-fold cross validated precision-recall curve. Shaded region corresponds to 1σ confidence intervals.

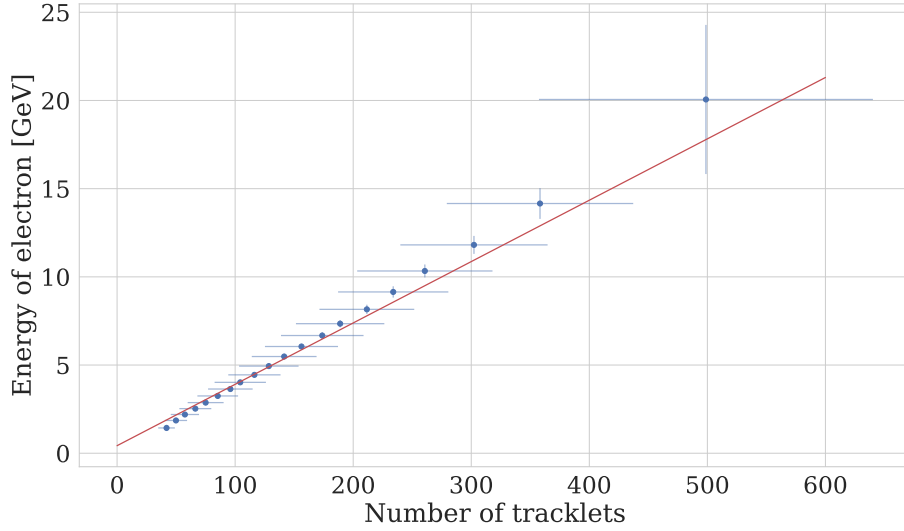


Figure 19: Visualization of the linear regression for energy reconstruction.

distribution for $\Delta E/E$ and plot the mean energy of the basket versus $\frac{\sigma_E}{E}$ (figure 20). We also fit energy resolution as a function of energy using the following relationship:

$$\frac{\sigma_E}{E} = A + \frac{B}{\sqrt{E}} \quad (5.2)$$

As we can see, with the increase of ground truth energy, resolution steadily improves as we would expect. Energy resolution for the network with 4 layers of EmulsionConv and 4 layers of EdgeConv is equal to $\frac{\sigma_E}{E} = (0.095 \pm 0.005) + \frac{(0.134 \pm 0.011)}{\sqrt{E}}$.

6 Conclusion and perspectives

We propose the first-ever algorithm that can reconstruct multiple showers in ECC brick. Our key contribution is a new layer type that can be used in end-to-end graph deep learning pipelines.

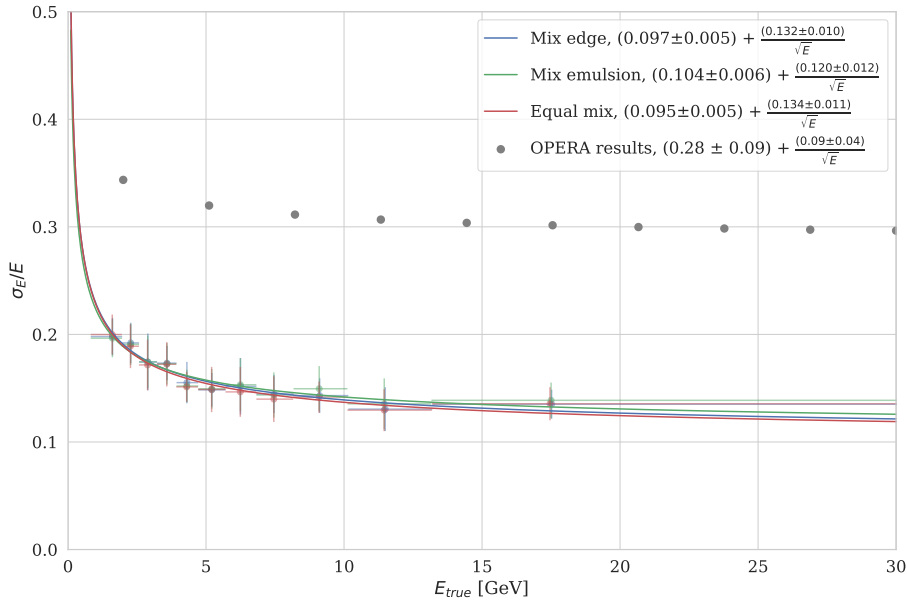


Figure 20: Energy resolution as a function of true energy. The solid line corresponds to fitted parameterized energy resolution.

We observe a statistically significant performance boost in comparison with the state-of-the-art EdgeConv layer [36] for the same number of graph convolution blocks on the problem of showers semantic segmentation on the dataset generated with FairSHIP software [34]. Furthermore, experiments have shown that the algorithm can detect up to 87% of showers with an achieved energy resolution of $\frac{\sigma_E}{E} = (0.095 \pm 0.005) + \frac{(0.134 \pm 0.011)}{\sqrt{E}}$. These results are obtained for the clusters that contain more than 90% of tracklets of the original showers (also called in the paper “recovered showers”). Thus, varying this definition, we can simultaneously change the percentage of recovered showers and energy resolution. For future work, it is of interest to investigate other possible definitions for the “recovered showers”, considering physical constraints on the required efficiency of the reconstruction and energy resolution.

Currents results in terms of energy resolution are on par with prior works on EM showers reconstruction while outperforming them in two key aspects:

- it is capable of solving shower semantic segmentation problem and separating showers in cases of overlaps;
- it does not use any prior information on showers origin, which simplifies the analysis pipeline and reduces the costs of experimental setup by, for example, allowing neglecting Changeable Sheets that were used in OPERA [18] to estimate shower origin position.

We believe that our approach can be of interest for other physical experiments that use sampling calorimeters or detectors that have similar data representation, i.e. 3D point clouds. One of the principal test benches for the proposed EM showers reconstruction algorithm could be the SND@LHC [50]. SND@LHC is a proposed, compact and self-contained experiment for measure-

ments with neutrinos produced at the Large Hadron Collider in the as yet unexplored region of pseudo-rapidity $7.2 < \eta < 8.6$.

We speculate that those possible uses are not limited to the sampling calorimeters and can be used to analyse tracklets data from Time Projection Chamber [51], and Silicon Tracker [52]. In [53], authors successfully apply hits classification by graph edges labelling for neutrino physics experiments in Liquid Argon Time Projection Chamber detectors. In [54], GNN is used to track particles and extract tracklets parameters. Our algorithm solves a more general form of a multiple showers reconstruction problem. For future work, we are going to investigate usage perspectives for other detector types.

Acknowledgments

We would like to express our appreciation to Giovanni De Lellis, Denis Derkach and Fedor Ratnikov for the invaluable comments and support.

The reported study utilized the supercomputer resources of the National Research University Higher School of Economics. The research leading to these results has received funding from the Russian Science Foundation under grant agreement n° 19-71-30020.

A Tracklet pairs energy and likeliness estimates

The energy and likeliness features are estimated with Molière's formulas of multiple scattering [55]. The formulas of multiple scattering states that for tracklets pairs with the parameters $x_i, y_i, z_i, \hat{\theta}_{x_i} = \arctan(\theta_{x_i}), \hat{\theta}_{y_i} = \arctan(\theta_{y_i})$, where $i = 1, 2$, difference in the spatial angle ($\Delta\hat{\theta} = \left((\Delta\hat{\theta}_x)^2 - (\Delta\hat{\theta}_y)^2 \right)^{1/2}$) and change in Z coordinate (Δz) could be described by the following distribution:

$$P(\Delta z, \Delta\hat{\theta}) = \frac{2\Delta\hat{\theta}}{\langle\hat{\theta}^2\rangle} \exp\left(-\frac{\Delta\hat{\theta}^2}{\langle\hat{\theta}^2\rangle}\right), \quad \langle\hat{\theta}^2\rangle = \hat{\theta}_s^2 \Delta z = \left(\frac{E_s}{\beta E}\right)^2 \frac{\Delta z}{X_0}, \quad (\text{A.1})$$

where $E_s = 21$ MeV - critical energy, $X_0 = 5000$ mm - radiation length [56], β - relative to the speed of light object velocity.

The energy and likeliness estimates features are found by maximizing following likeliness function:

$$P(\Delta z, \Delta\hat{\theta})Q(\Delta z, \hat{\theta}_x)Q(\Delta z, \hat{\theta}_y)S(\Delta z, x)S(\Delta z, y) \rightarrow \max_E, \quad (\text{A.2})$$

where Q, S - changes in spatial angle projections (i.e., $\Delta\hat{\theta}_x, \Delta\hat{\theta}_y$) and changes in spatial deviation ($\Delta x, \Delta y$), correspondingly. Q, S follow Gaussian distribution.

Table 2: Recovered showers 10-folds cross-validation results on the test data parts in a dependence on min samples core and threshold parameters.

Parameters		Networks		
min samples core	threshold	Mix Emulsion	Equal Mix	Mix Edge
2	0.2	0.853 ± 0.031	0.845 ± 0.031	0.862 ± 0.016
	0.4	0.859 ± 0.035	0.853 ± 0.036	0.868 ± 0.020
	0.6	0.859 ± 0.035	0.857 ± 0.036	0.868 ± 0.020
	0.8	0.859 ± 0.035	0.857 ± 0.036	0.867 ± 0.020
3	0.2	0.848 ± 0.032	0.840 ± 0.033	0.862 ± 0.014
	0.4	0.856 ± 0.038	0.854 ± 0.037	0.864 ± 0.019
	0.6	0.856 ± 0.038	0.854 ± 0.037	0.867 ± 0.020
	0.8	0.856 ± 0.038	0.854 ± 0.037	0.867 ± 0.020
4	0.2	0.844 ± 0.034	0.837 ± 0.033	0.857 ± 0.016
	0.4	0.851 ± 0.038	0.851 ± 0.035	0.866 ± 0.019
	0.6	0.852 ± 0.038	0.851 ± 0.036	0.866 ± 0.019
	0.8	0.852 ± 0.038	0.852 ± 0.035	0.866 ± 0.019
5	0.2	0.843 ± 0.034	0.835 ± 0.036	0.854 ± 0.017
	0.4	0.846 ± 0.036	0.848 ± 0.034	0.862 ± 0.017
	0.6	0.847 ± 0.037	0.848 ± 0.034	0.862 ± 0.017
	0.8	0.847 ± 0.037	0.848 ± 0.034	0.862 ± 0.017

B Grid search of optimal parameters

In tables 2, 3 we present the grid-search of the EWSCAM algorithm parameters. We show 10-fold cross-validation results for three neural networks configurations: mix emulsion network, consisting of 5 layers of EmulsionConv and 3 layers of EdgeConv, equal mix network, containing 4 EmulsionConv and 4 EdgeConv layers, and mix edge network, that includes 3 layers of EmulsionConv and 5 layers of EdgeConv. The best parameters are shown in table 4.

C Different multiplicities

In this section, we report the performance of our algorithm applied for two datasets with fixed multiplicity profiles: 50 showers per brick and 200 showers per brick. We also assess how the performance changes when the network trained on one multiplicity profile is applied to the dataset with a different multiplicity profile.

First, we train the “mix emulsion” network on the datasets with fixed and variable multiplicities and evaluate this network on a dataset with a variable number of showers per brick (figure 21). As one can notice, networks trained on datasets with fixed multiplicities perform worse on a dataset with variable multiplicity profile; in other words, they overfit to a constant number of showers in a brick.

Table 3: Average per brick energy resolution 10-folds cross-validation results on the test data parts in a dependence on min samples core and threshold parameters.

Parameters		Networks		
min samples core	threshold	Mix Emulsion	Equal Mix	Mix Edge
2	0.2	0.182 \pm 0.008	0.175 \pm 0.008	0.176 \pm 0.013
	0.4	0.184 \pm 0.009	0.178 \pm 0.008	0.182 \pm 0.011
	0.6	0.185 \pm 0.009	0.179 \pm 0.009	0.183 \pm 0.011
	0.8	0.185 \pm 0.009	0.179 \pm 0.008	0.183 \pm 0.011
3	0.2	0.184 \pm 0.008	0.179 \pm 0.008	0.179 \pm 0.014
	0.4	0.188 \pm 0.009	0.183 \pm 0.108	0.186 \pm 0.011
	0.6	0.188 \pm 0.009	0.183 \pm 0.008	0.187 \pm 0.010
	0.8	0.188 \pm 0.009	0.183 \pm 0.008	0.187 \pm 0.010
4	0.2	0.186 \pm 0.008	0.180 \pm 0.007	0.182 \pm 0.012
	0.4	0.189 \pm 0.009	0.185 \pm 0.008	0.190 \pm 0.009
	0.6	0.189 \pm 0.009	0.184 \pm 0.008	0.190 \pm 0.009
	0.8	0.189 \pm 0.009	0.184 \pm 0.008	0.190 \pm 0.009
5	0.2	0.188 \pm 0.006	0.182 \pm 0.007	0.183 \pm 0.012
	0.4	0.191 \pm 0.009	0.186 \pm 0.008	0.192 \pm 0.008
	0.6	0.191 \pm 0.009	0.186 \pm 0.007	0.192 \pm 0.008
	0.8	0.190 \pm 0.009	0.186 \pm 0.007	0.192 \pm 0.008

Table 4: Resulting clustering parameters.

Network	Parameters	
	min samples core	threshold
Mix emulsion	2	0.4
Mix equal	2	0.6
Mix edge	2	0.4

We also report recovered showers, stuck showers, broken showers, lost showers, and energy resolution for each dataset applying networks trained on different multiplicity profiles (tables 5, 6, 7, figures 22, 23, 24). The provided results are achieved with the minimal sample core and threshold clustering parameters equal to 2 and 0.6, correspondingly.

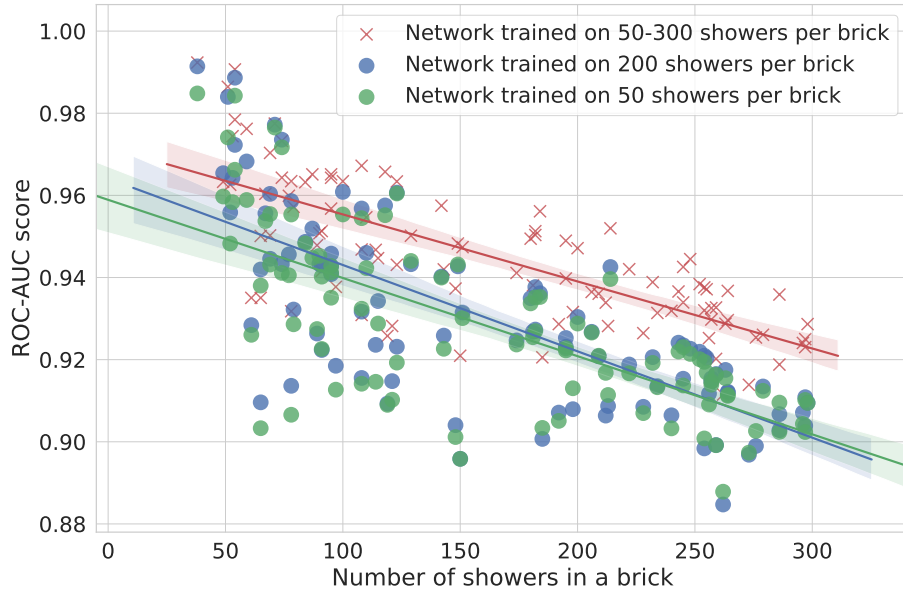


Figure 21: ROC-AUC scores per brick. Networks are trained on datasets with different multiplicities and evaluated on a dataset with variable multiplicity 50-300.

Table 5: Comparison of the performance of the clustering algorithms reported using 3-fold cross-validation on the dataset composed of 50-300 showers per brick if trained on a dataset with different multiplicity.

Network Metric	50	200	50-300
Recovered Showers, %	82.10 ± 2.46	82.20 ± 1.91	86.78 ± 2.15
Stuck Showers, %	14.86 ± 2.75	15.03 ± 2.21	2.19 ± 0.32
Broken Showers, %	2.68 ± 0.03	2.49 ± 0.37	4.03 ± 0.86
Lost Showers, %	0.36 ± 0.08	0.28 ± 0.07	0.36 ± 0.14

Table 6: Comparison of the performance of the clustering algorithms reported using 3-fold cross-validation on the dataset composed of 50 showers per brick if trained on a dataset with different multiplicity.

Network Metric	50	200	50-300
Recovered Showers, %	86.00 ± 2.53	88.99 ± 0.68	91.52 ± 0.06
Stuck Showers, %	12.1 ± 2.25	9.05 ± 0.63	5.53 ± 0.23
Broken Showers, %	1.26 ± 0.19	1.25 ± 0.06	2.00 ± 0.32
Lost Showers, %	0.64 ± 0.16	0.71 ± 0.04	0.36 ± 0.08

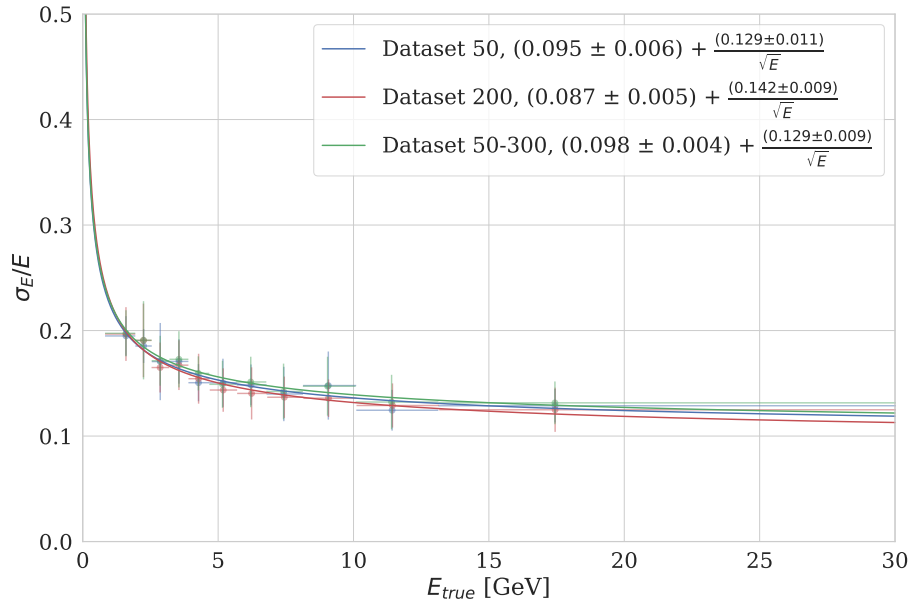


Figure 22: Energy resolution on the test dataset with 50-300 showers per brick depending on the train dataset number of showers.

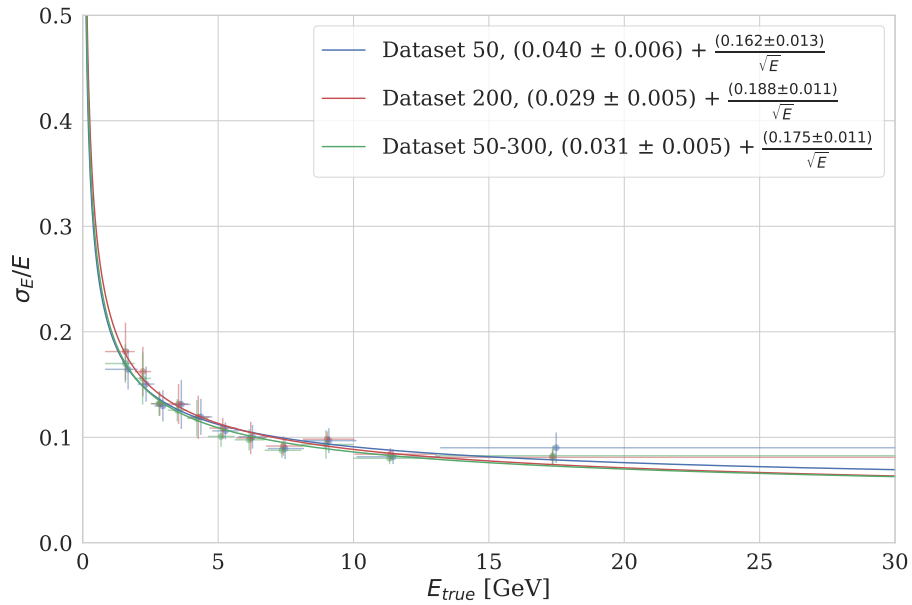


Figure 23: Energy resolution on the test dataset with 50 showers per brick depending on the train dataset number of showers.

Table 7: Comparison of the performance of the clustering algorithms reported using 3-fold cross-validation on the dataset composed of 200 showers per brick if trained on a dataset with different multiplicity.

Metric \ Network	50	200	50-300
Recovered Showers, %	75.24 ± 0.89	78.48 ± 0.25	83.26 ± 2.17
Stuck Showers, %	21.93 ± 0.89	18.63 ± 0.13	13.08 ± 2.42
Broken Showers, %	1.43 ± 0.15	1.49 ± 0.11	2.19 ± 0.32
Lost Showers, %	1.40 ± 0.24	1.39 ± 0.08	1.47 ± 0.13

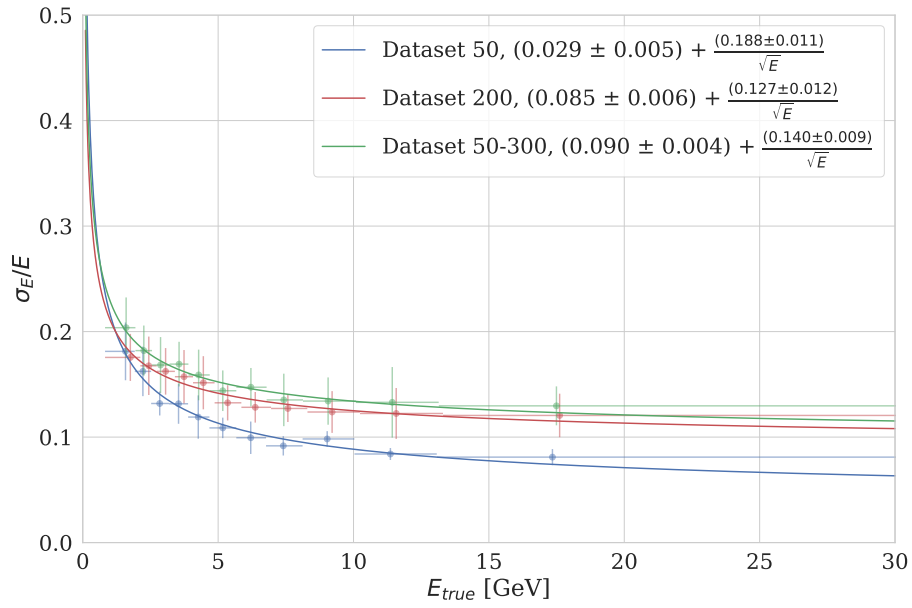


Figure 24: Energy resolution on the test dataset with 200 showers per brick depending on the train dataset number of showers.

References

- [1] R. Acquafredda et al., *The OPERA experiment in the CERN to Gran Sasso neutrino beam*, *JINST* **4** (2009) P04018.
- [2] L. Arrabito, D. Autiero, C. Bozza, S. Buontempo, Y. Caffari, L. Consiglio, M. Cozzi, N. D'Ambrosio, G. De Lellis and M. De Serio, *Electron/pion separation with an Emulsion Cloud Chamber by using a Neural Network*, *Journal of Instrumentation* **2** (2007) P02001.
- [3] N. Agafonova et al., *Final Results of the OPERA Experiment on $\nu\tau$ Appearance in the CNGS Neutrino Beam*, *PRL* **120** (2018) 211801.
- [4] W. M. Bonivento, *The SHiP experiment at CERN*, *Journal of Physics: Conference Series* **878** (2017) 1 012059.
- [5] O. Lantwin, *Search for new physics with the SHiP experiment at CERN*, *PoS EPS-HEP2017* **304** (2017) 7 [hep-ex/1710.03277].
- [6] S. Dmitrievsky, *Status of the OPERA Neutrino Oscillation Experiment*, *Acta Physica Polonica B.* **41** (2010)
- [7] L. Arrabito et al., *Hardware performance of a scanning system for high speed analysis of nuclear emulsions*, *Nucl. Instrum. Meth. A568* (2006) 578–587, [physics/0604043].
- [8] N. Armenise et al., *High-speed particle tracking in nuclear emulsion by last-generation automatic microscopes*, *Nucl. Instrum. Meth. A551* (2005) 261–270.
- [9] R. Jacobsson et al., *SHiP Experiment - Progress Report*, CERN-SHiP-NOTE-2018-001 (2019), url: <https://cds.cern.ch/record/2650966?ln=en>, accessed: 2021-09-12.
- [10] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang and P. S. Yu, *A Comprehensive Survey on Graph Neural Networks*, " in *IEEE Transactions on Neural Networks and Learning Systems* (2020) 2978386.
- [11] K. Xu, W. Hu, J. Leskovec, St. Jegelka, *How Powerful are Graph Neural Networks?*, *CoRR* (2018) [cs.LG/1810.006].
- [12] J. Zhou, G. Cui, Zh. Zhang, Ch. Yang, Zh. Liu, L. Wang, Ch. Li, M. Sun , *Graph Neural Networks: A Review of Methods and Applications* *CoRR* (2018) [cs.LG/1812.08434].
- [13] HEP ML Community, *A Living Review of Machine Learning for Particle Physics* url: <https://iml-wg.github.io/HEPML-LivingReview/>
- [14] J. Duarte, J.-R. Vlimant, *Graph Neural Networks for Particle Tracking and Reconstruction* (2020) [hep-ph/2012.01249].
- [15] J. Shlomi, P. Battaglia, J.-R. Vlimant, *Graph neural networks in particle physics*, *Machine Learning: Science and Technology*, Vol. 2, p. 021001 (2021).
- [16] L. McInnes and J. Healy, *Accelerated Hierarchical Density Clustering*, *2017 IEEE International Conference on Data Mining Workshops (ICDMW)* (2017) pp. 33-42 [stat.ML/1705.07321].
- [17] Ahdida, C., Akmete, A. et al. *Sensitivity of the SHiP experiment to light dark matter*, *J. High Energ. Phys.* **2021**, 199 (2021)
- [18] B. Hosseini, *Search for Tau Neutrinos in the $\tau \rightarrow e$ Decay Channel in the OPERA Experiment* (2015).
- [19] M. De Luca, *Electron identification in the Emulsion detector of SHiP experiment using machine learning techniques* (2021).

- [20] A. Akmete et al., *The active muon shield in the SHiP experiment*, *Journal of Instrumentation* 12 (5) (2017) P05011.
- [21] K. Shpak, *Separation of the overlapping electromagnetic showers in the cellular GAMS-type calorimeters*, *International Workshop on Future Linear Collider (LCWS2017)*, (2018) [physics.ins-det/].
- [22] A.A. Lednev, *Separation of the overlapping electromagnetic showers in the cellular GAMS type calorimeters*, *IFVE-93-153*, (1993).
- [23] E. Trofimova, *ketrint/em_showers_segmentation* url: <http://doi.org/10.5281/zenodo.5708308>.
- [24] CERN *The Toolkit for Multivariate Data Analysis with ROOT (TMVA)*, CERN-OPEN-2007-007 [physics/0703039].
- [25] A. Ustyuzhanin, S. Shirobokov, V. Belavin and A. Filatov, *Machine-Learning techniques for electromagnetic showers identification in OPERA datasets*, *ACAT 2017 conference proceedings* (2017).
- [26] L. Oliveir, B. Nachmana, M. Paganini, *Electromagnetic showers beyond shower shapes*, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, Volume 951, (2020) [hep-ex/1806.05667].
- [27] Y. Verma, S. Jena, *Shower Identification in Calorimeter using Deep Learning*, (2021) [physics.data-an/2103.16247].
- [28] A. Aurisano et al. *A convolutional neural network neutrino event classifier*, *Journal of Instrumentation* 11 (09) (2016) [hep-ex/1604.01444]
- [29] N. Choma et al., *Track Seeding and Labelling with Embedded-space Graph Neural Networks* (2020) [physics.ins-det/2007.00149].
- [30] F. Pedregosa et al., *Scikit-learn: Machine Learning in Python*, *Journal of Machine Learning Research* 12 (2011), pp. 25–2830.
- [31] X. Ju et al., *Graph Neural Networks for Particle Reconstruction in High Energy Physics detectors* (2020) [physics.ins-det/2003.11603].
- [32] A. Anokhina et al., *Emulsion sheet doublets as interface trackers for the OPERA experiment*, *Journal of Instrumentation* (3) (2008) P07005.
- [33] M. De Luca *Electron identification in the Emulsion detector of SHiP experiment using machine learning techniques* (2021) url: <https://cds.cern.ch/record/2750060>, accessed:2021-09-14.
- [34] GitHub, Inc. *Fairship* url: <https://github.com/ShipSoft/FairShip>, accessed: 2019-11-07.
- [35] V. Belavin, E. Trofimova, *Simulated EM showers data*. Zenodo url: <https://doi.org/10.5281/zenodo.5570901> (2021) [Data set]
- [36] Y. Wang, Y. Sun, Z. Liu, S.E. Sarma, M.M. Bronstein and J.M. Solomon, *Dynamic Graph CNN for Learning on Point Clouds*, *ACM Transactions on Graphics (TOG)* 38 (2019), [cs.CV/1801.079].
- [37] T.-Y. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, *Focal Loss for Dense Object Detection*, *Facebook AI Research (FAIR)* (2018) [cs.CV/1708.02002v2].
- [38] A. Araujo, W. Norris, J. Sim, *Computing Receptive Fields of Convolutional Neural Networks*, *Distill* 11 (2019).
- [39] W. Luo, Y. Li, R. Urtasun, R. Zemel, *Understanding the effective receptive field in deep convolutional neural networks*, *Advances in neural information processing systems* (2016) [cs.CV/1701.04128].

- [40] Z. Liu, Ch. Chen, L. Li et al., *Geniepath: Graph neural networks with adaptive receptive paths*, *Proceedings of the AAAI Conference on Artificial Intelligence* **33** (2019) [cs.LG/1802.00910].
- [41] G. Li, M. Muller, Al. Thabet, B. Ghanem, *Deepgcns: Can GCNs go as deep as CNNs?*, *Proceedings of the IEEE International Conference on Computer Vision* (2019) [cs.CV/1904.03751].
- [42] J. B. Kruskal, *On the shortest spanning subtree of a graph and the traveling salesman problem*, *Proceedings of the American Mathematical Society*. 7 (1) (1956) S0002-9939-1956-0078686-7.
- [43] GitHub, Inc. *XGBoost* url: <https://github.com/dmlc/xgboost>, accessed: 2019-12-17.
- [44] P.A. Zyla et al., *Particle Data Group*, *Prog. Theor. Exp. Phys.* 2020, 083C01 (2020).
- [45] P. J. Huber, *Robust Estimation of a Location Parameter*, *Annals of Mathematical Statistics* 35 (1) (1964) p. 73–101 1177703732.
- [46] T. Fawcett, *An introduction to ROC analysis*, *Pattern Recognition Letters* **26** (2006) j.patrec.2005.10.010.
- [47] P. Kingma Diederik, B. Jimmy, *Adam: A Method for Stochastic Optimization* (2014) [cs.LG/1412.6980].
- [48] L. Zhao and L. Akoglu, *PairNorm: Tackling Oversmoothing in GNNs*, *Thirty-eighth International Conference on Machine Learning (ICML2020)* (2020).
- [49] G. Li et al., *DeepGCNs: Can GCNs Go as Deep as CNNs?* (2019) [cs.CV/1904.03751].
- [50] C. Ahdida, R. Albanese, A. Alexandrov et al., *SND@LHC - Scattering and Neutrino Detector at the LHC*, CERN, Geneva, CERN-LHCC-2021-003. LHCC-P-016, (2021) url: <https://cds.cern.ch/record/2750060>, accessed: 2021-03-13.
- [51] R. Acciarri et al., *Summary of the Second Workshop on Liquid Argon Time Projection Chamber Research and Development in the United States*, *Journal of Instrumentation* 10 (7) (2015) [physics.ins-det/1504.05608].
- [52] M. Tobin, *The LHCb Silicon Tracker*, *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* (2016), p. 174-180 j.nima.2005.03.113.
- [53] J. Hewes et al., *Graph Neural Network for Object Reconstruction in Liquid Argon Time Projection Chambers* (2021) [hep-ex/1504.2103.06233].
- [54] S. Thais, G. DeZoort, *Instance Segmentation GNNs for One-Shot Conformal Tracking at the LHC* (2021) [cs.CV/2103.06509].
- [55] H A. Bethe, *Molier's Theory of Multiple Scattering* (1953) p. 1256—1266 [hep-ph/1204.3675].
- [56] De Angelis, A., Pimenta, M., *Introduction to Particle and Astroparticle Physics, Undergraduate Lecture Notes in Physics* (2018) 978-3-319-78181-5