Artificial Intelligence techniques for space experiments

M.Bossa , F.Cuna, F.Gargano, M.N.Mazziotta

Istituto Nazionale di Fisica Nucleare

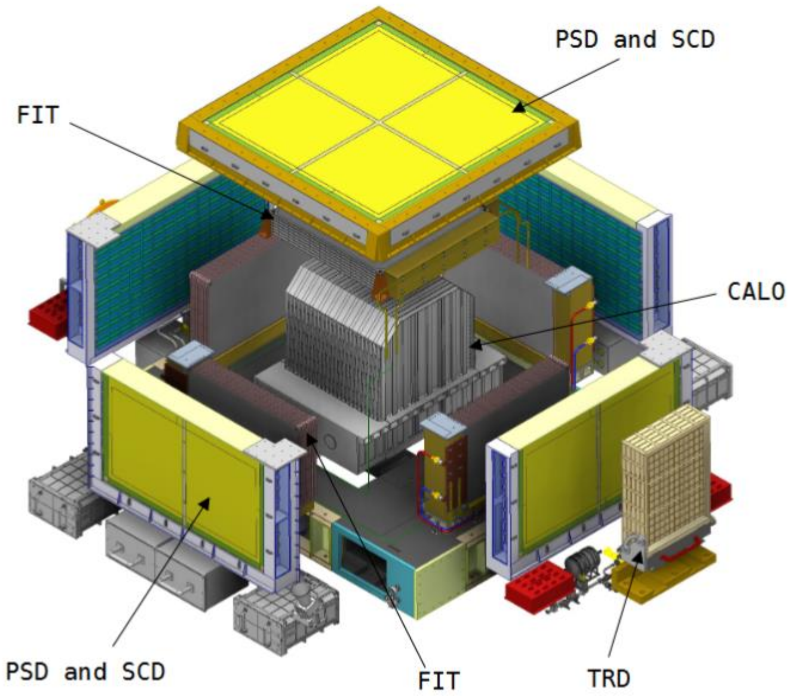
Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

Outline

Our work aims to develop robust AI- driven algorithms for space experiments capable of handling real-world experimental data.

1. The HERD experiment
2. Data simulation
3. Tracks reconstruction with Graph Neural Networks (GNNs)
4. Discrimination between hadronic and electromagnetic showers in a calorimeter with Transformers

HERD as proof of concept for AI in space experiment



- HERD is a future space mission for:
  - the direct detection of cosmic rays up to 1 PeV
  - the measurement of the electron and positron flux up to several tens of TeV

- It consists of homogeneous, isotropic, and deeply segmented 3D calorimeter, surrounded by multiple sub-detectors for charge, timing, and tracking measurements.

Fiber Tracker (FIT)

Particle tracking and charge measurement.

It consists of 5 tracking sectors made of seven X-Y double layers of scintillating fibers

Calorimeter (CALO)

Particle energy measurement

Discrimination of electrons from protons and nuclei

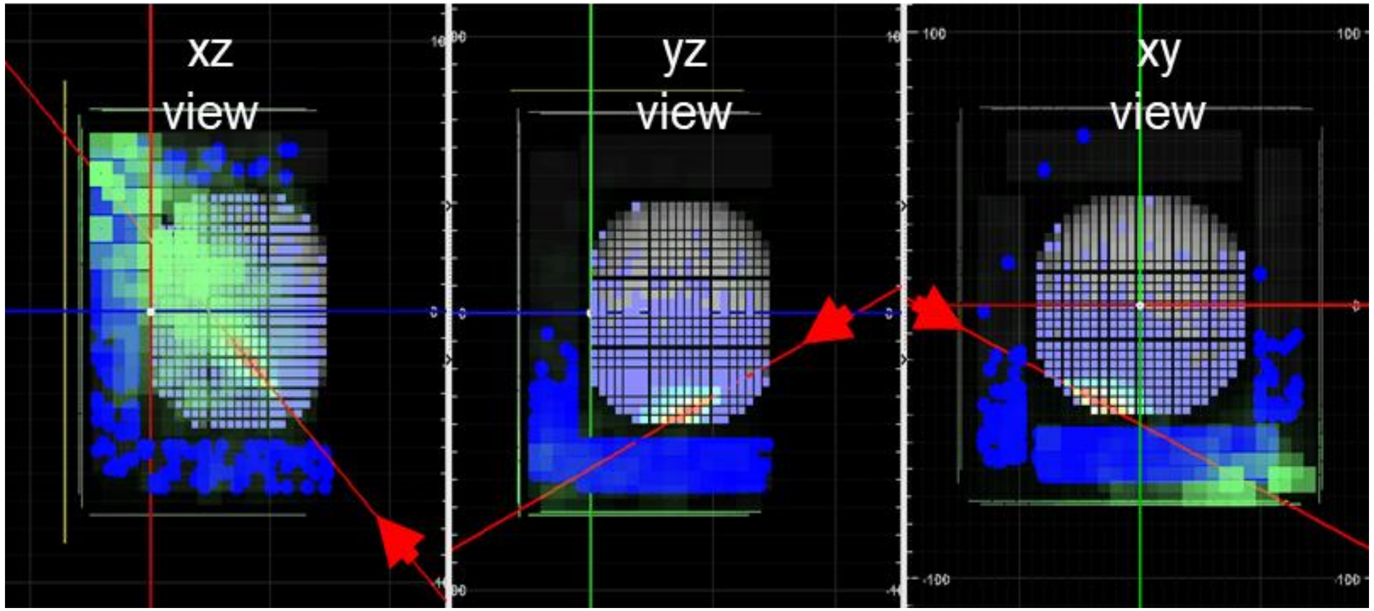
It consists of 7500 LYSO cubic scintillating crystals of 3 cm side

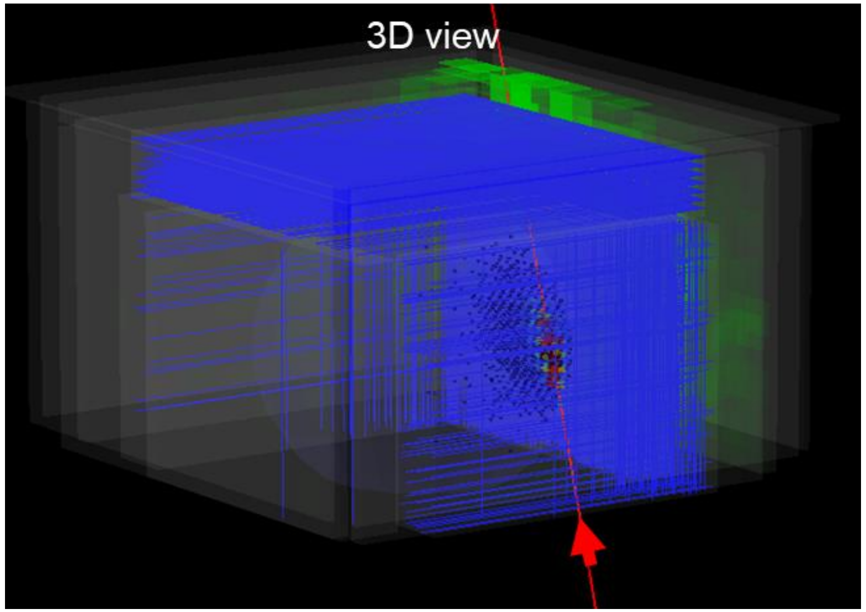
Data simulation

- The full dataset is generated using the custom HerdSoftware simulation framework
- It includes simulated events of electrons and protons, both generated according to a power-law energy spectrum E^{-1}, spanning an energy range from 1 GeV to 1 PeV
- Tracks are distributed within a spherical region surrounding the HERD detector.

Challenge

Backscattering tracks originating from the calorimeter, which interfere with the correct identification of the primary particle's trajectory

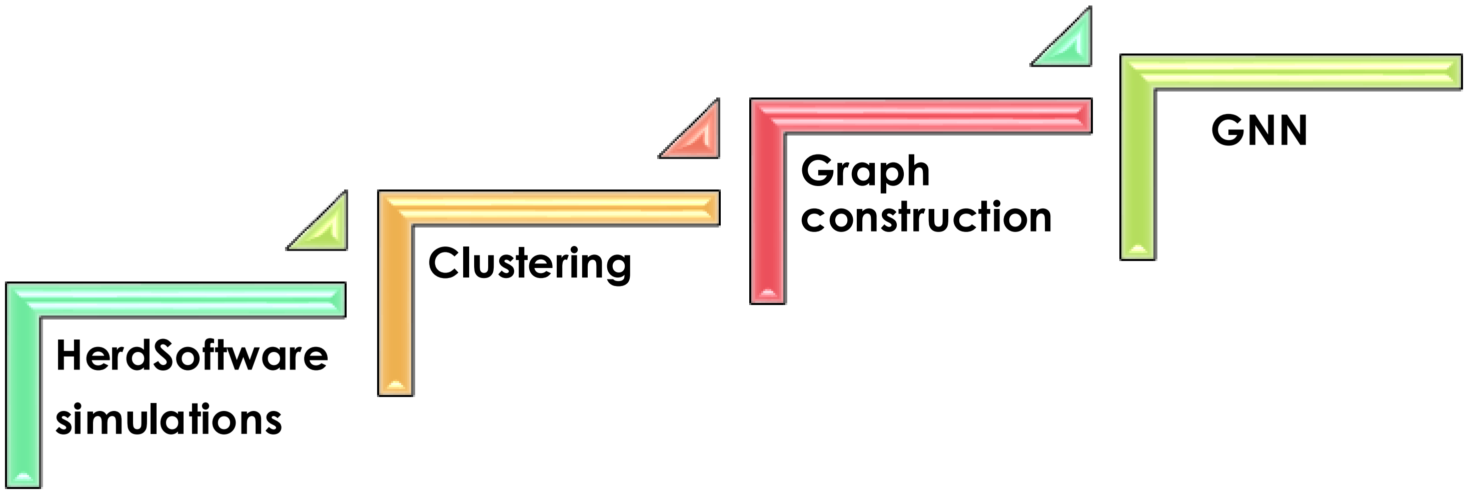




Solution methodology

Graph neural network for nodes classification. Tracking data can be represented as graphs, which is a collection of nodes and links. Nodes are the hits inside the tracker. Links are the interlayer connections.

Graph Neural Network for tracks reconstruction: pipeline



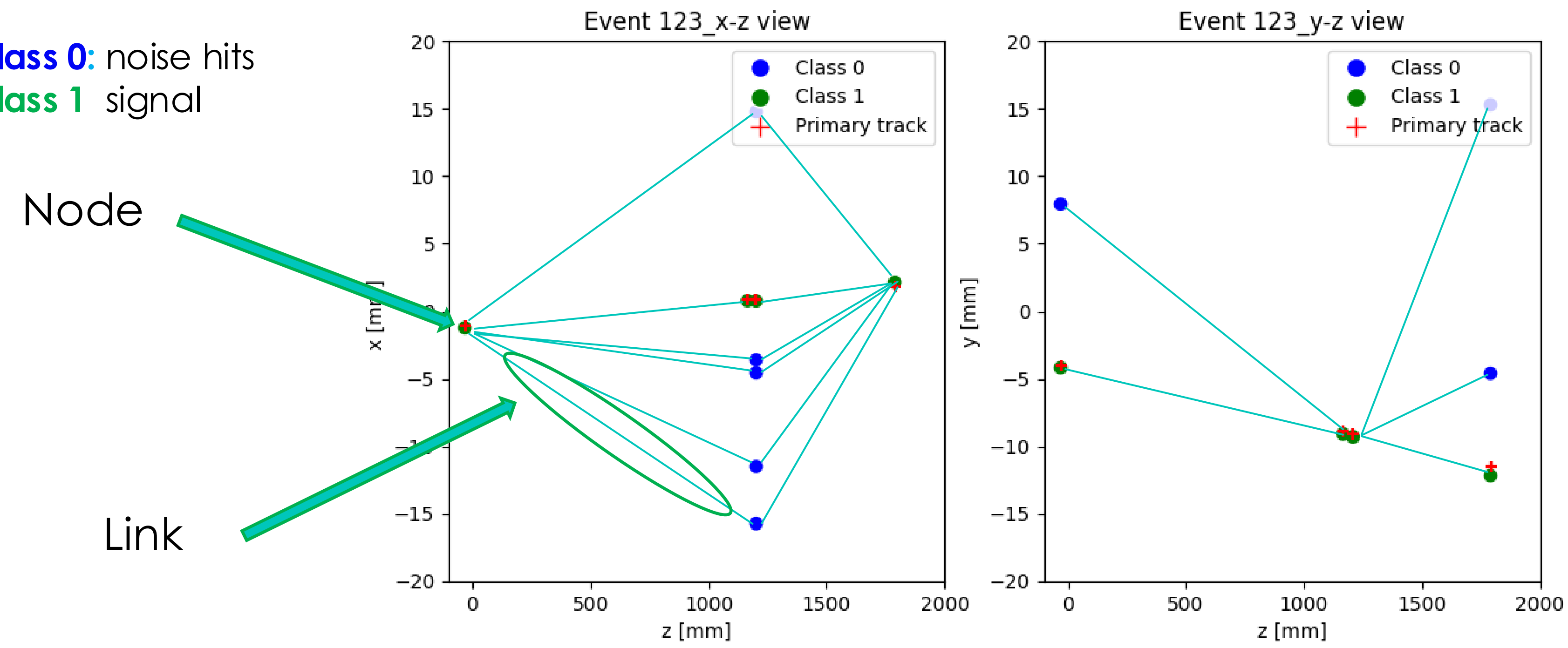
- Simulation of the data set
- Clustering
- Graph construction
- GNN-based tracking algorithm, which consists of a GNN classification of backscattering clusters from signal clusters and a final linear fit to retrieve the track parameter (angular coefficient and intercept)

The hits inside each side and layers of the FIT are clustered by applying a traditional clustering algorithm, where neighboring silicon strips with activated signals are grouped together and the barycenter of charge is calculated.

Starting by the clusters, data have been organized in a graph format, where nodes are represented by the clusters position and links by the inter layer connections between clusters.

The graphs are undirected to increase generality, as this approach captures a wider range of potential track interactions

Class 0: noise hits
Class 1 signal



Graph Neural Network for tracks reconstruction: GNN algorithm

- SageConv architecture
- 18 layers
- Mean aggregation function
- 128 hidden size
- Adam optimizer
- Binary cross entropy loss function

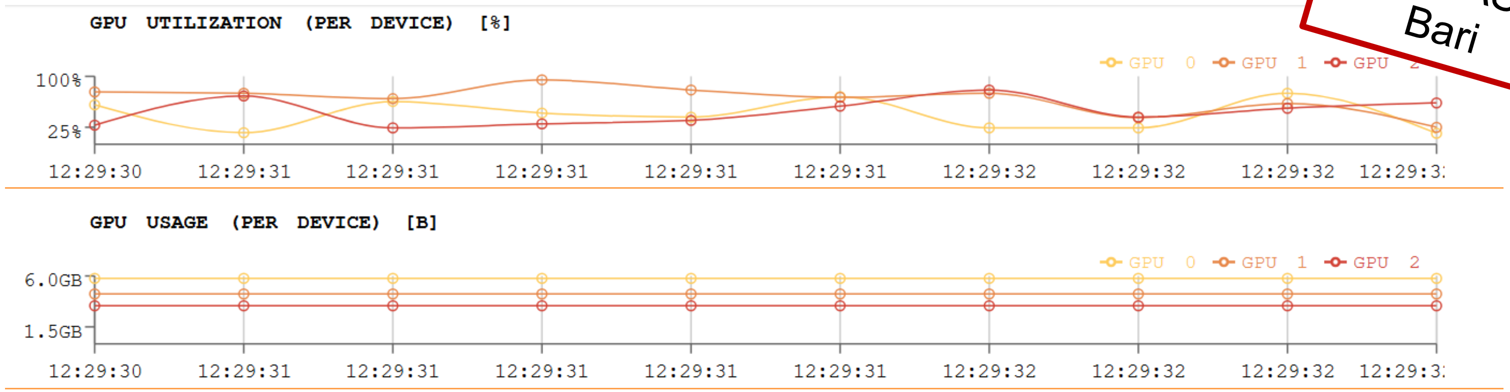
4,5 million simulated track data
75% train-15% validation-10% test

Distributed training

To enhance the time consuming we performed the distributing training by using:

- the JupyterLab instance with 3 A100 NVIDIA GPUs
- the Leonardo Hub instance with 4 A100 NVIDIA

RECAS-Bari

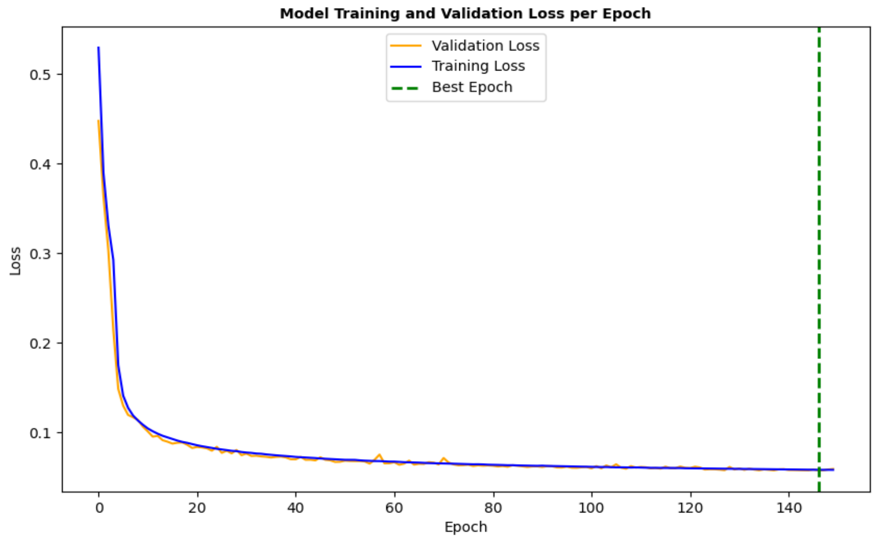


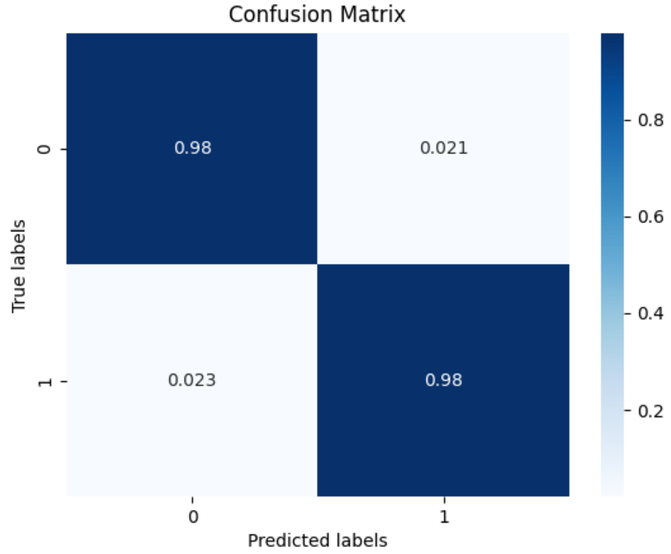
Leonardo-HUB

```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 530.30.02              Driver Version: 530.30.02    CUDA Version: 12.1     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                  Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf            Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  NVIDIA A100-SXM-64GB            On | 00000000:1D:00.0 Off |                    0 |
| N/A   44C    P0              83W / 459W |   7771MiB / 65536MiB |      1%      Default |
|                                         |                      |             Disabled |
+-----------------------------------------+----------------------+----------------------+
|   1  NVIDIA A100-SXM-64GB            On | 00000000:56:00.0 Off |                    0 |
| N/A   45C    P0              86W / 461W |   3999MiB / 65536MiB |     70%      Default |
|                                         |                      |             Disabled |
+-----------------------------------------+----------------------+----------------------+
|   2  NVIDIA A100-SXM-64GB            On | 00000000:8F:00.0 Off |                    0 |
| N/A   45C    P0              88W / 458W |   3957MiB / 65536MiB |     86%      Default |
|                                         |                      |             Disabled |
+-----------------------------------------+----------------------+----------------------+
|   3  NVIDIA A100-SXM-64GB            On | 00000000:C8:00.0 Off |                    0 |
| N/A   44C    P0              82W / 457W |   4103MiB / 65536MiB |     10%      Default |
|                                         |                      |             Disabled |
+-----------------------------------------+----------------------+----------------------+

+---------------------------------------------------------------------------------------+
| Processes:                                                                            |
|  GPU   GI   CI        PID   Type   Process name                            GPU Memory |
|        ID   ID                                                             Usage      |
|=======================================================================================|
+---------------------------------------------------------------------------------------+
```

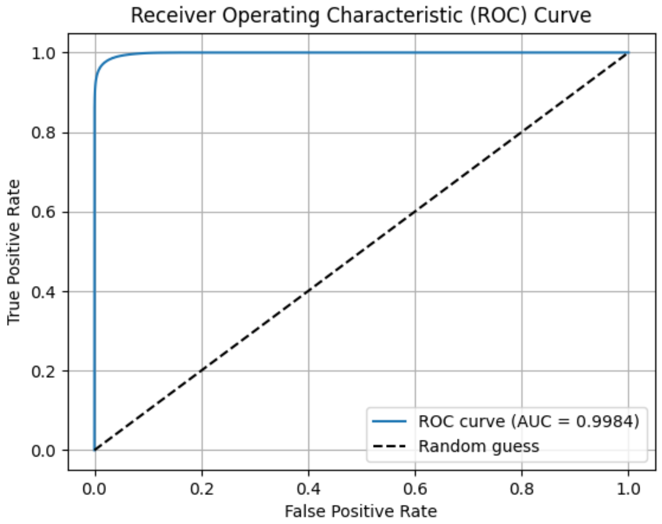
Graph Neural Network for tracks reconstruction: results



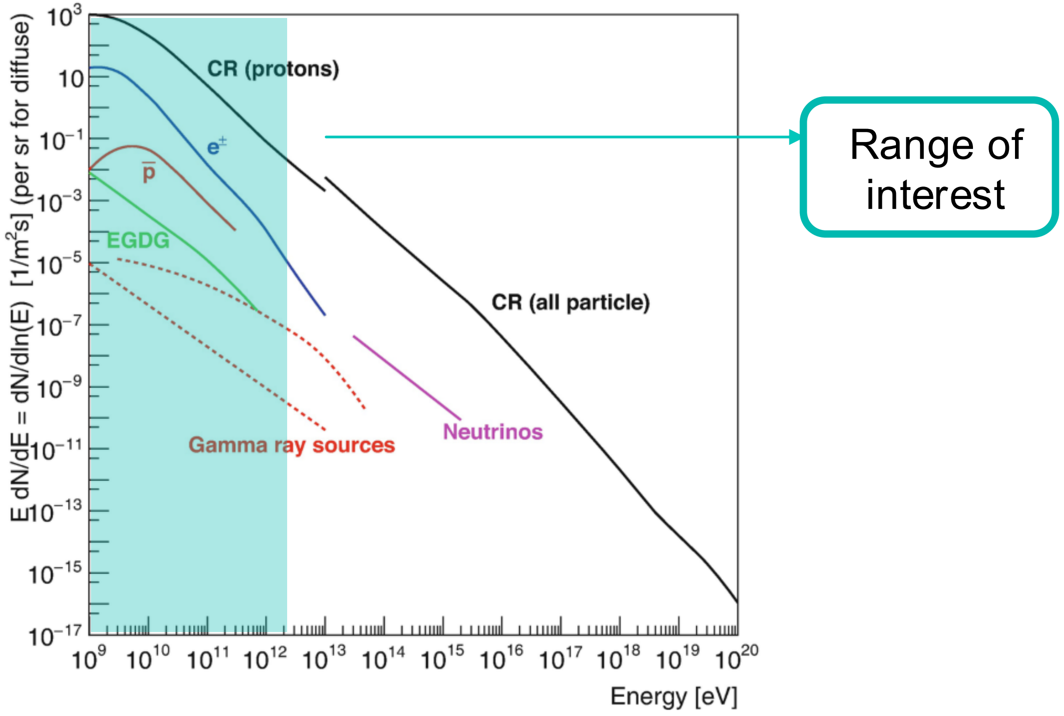


Paper submitted at PDP Conference!!! https://ieeexplore.ieee.org/xpl/conhome/10974743/proceeding

Metrics	Values
Accuracy	97.80%
Recall	97.65%
Precision	97.81%
F1-score	97.73%
ROC AUC	99.84%



Challenge



Cosmic rays (CRs) are high-energy protons and nuclei (plus a minority electron component, less than 1%) produced in astrophysical environments, filling the galactic space and arriving on Earth.

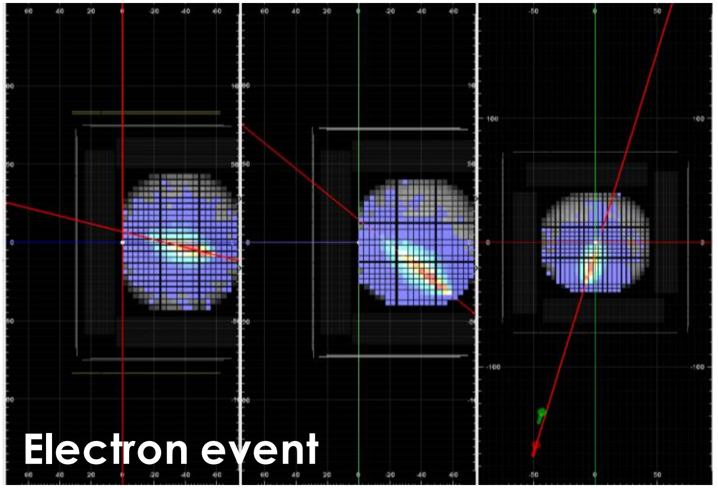
The abundance of cosmic ray electrons above a few GeV is significantly lower-by a factor of $10^{-3} \sim 10^{-2}$ -compared to that of cosmic ray protons. This makes precise measurement of the electron spectrum a challenging task.

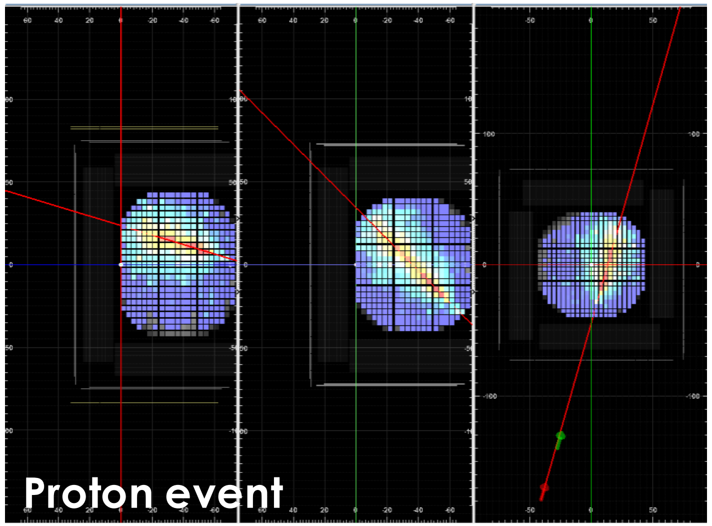
Discrimination between hadronic and electromagnetic showers in a calorimeter with Transformers

Different particle types create distinct shower patterns:

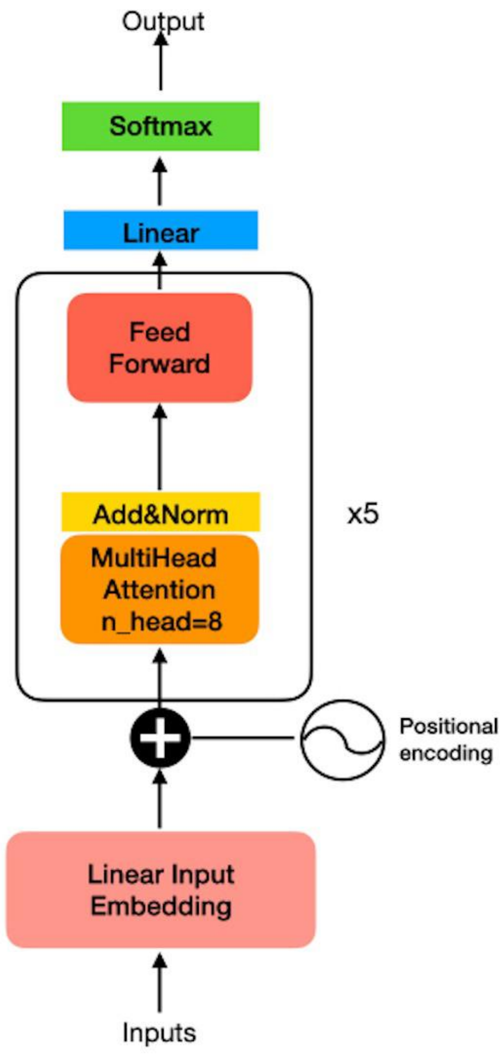
Solution methodology

Novel approach for classifying electron- and proton-induced events consists of a transformer model





Discrimination between hadronic and electromagnetic showers in a calorimeter with Transformers: pipeline



Features: energy deposition and the X, Y, Z coordinates of the activated pixel.

- ❑ **Embedding Layer**: the input features are transformed through a linear embedding layer, which maps each input vector to a vector of larger size.
- ❑ **Positional Encoding**: computes a fixed encoding based on sine and cosine functions, which is then added to the embedded input.
- ❑ **Transformer Encoder**: consists of a stack of five identical layers. Each layer includes two sub-layers: the first is a multi-head self-attention mechanism that learns dependencies between sequence elements, while the second is a position-wise fully connected feed-forward network that allows the model to capture complex, high dimensional relationships inherent in physical events.
- ❑ **Add&Norm**: includes residual connections and layer normalization after each sub-layer
- ❑ **Classification Layer**: the encoder's output is aggregated by computing the mean across the sequence dimension, producing a single feature vector.
- ❑ **Linear & Softmax**: this vector is fed to a fully connected layer, which outputs a final prediction for the binary classification.

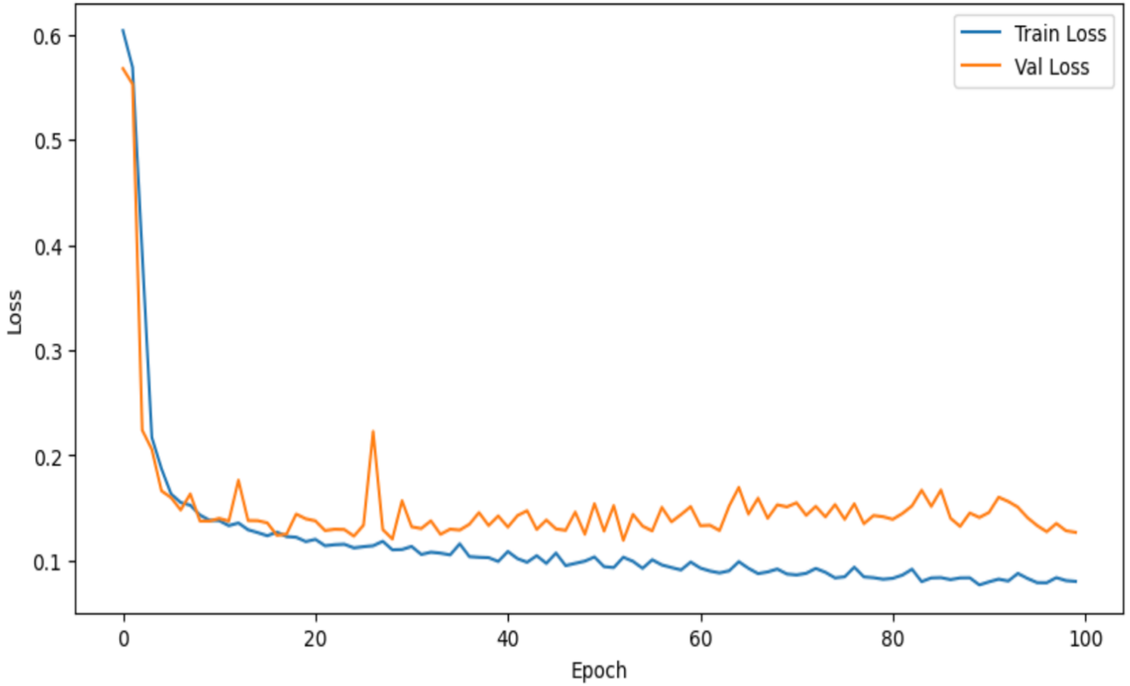
Discrimination between hadronic and electromagnetic showers in a calorimeter with Transformers: results

20000 events:

- 10000 electrons
- 10000 protons

80%-20% train-test

Model was trained for 100 epochs using the Adam optimizer to minimize the binary cross-entropy loss.



Metrics	Values
Accuracy	94.55%
Recall	96.81%
Precision	92.58%

Paper submitted at PDP Conference!!! https://ieeexplore.ieee.org/xpl/conhome/10974743/proceeding

The observed overfitting and fluctuations in the curves are due to the small size of the dataset.

Discrimination between hadronic and electromagnetic showers in a calorimeter: XGBoost

XGBoost (Extreme Gradient Boosting) is a powerful, tree-based machine learning algorithm that excels in classification tasks.

We leveraged XGBoost on a dataset of **high-level features** extracted from calorimeter simulations. These features encapsulate the fundamental differences in how **EM** and **hadronic showers** propagate through the detector:

1. **Lateral Moment of the Shower (until the 4th order):** * This describes the **transverse spread** of the shower. EM showers tend to have smaller lateral moments (more concentrated), while hadronic showers are generally wider.
    - Higher-order moments capture finer details of the distribution's shape, providing more discriminative power.

2. **Longitudinal Moment of the Shower (until the 4th order):** * This characterizes the **depth of penetration and development** of the shower within the calorimeter. EM showers typically deposit most of their energy earlier, leading to different longitudinal moments compared to hadronic showers, which develop more slowly and irregularly.
    - Similar to lateral moments, higher orders provide a more nuanced description of the energy deposition along the detector's depth.

3. **Longitudinal Profile of the Shower:**
    - This refers to the energy deposited along the shower's axis during its development, offers a direct and detailed representation of how the shower evolves in depth

XGBoost Algorithm: high level features

- **Lateral moment of the shower**

$$M_{lateral}(n) = \frac{\sum_i E_i \cdot d_i{}^n}{\sum_i E_i},$$

Where E_i is the energy deposited in the i-th pixel, d_i is the distance of the i-th activated pixel from a reference axis, which in our case is the shower axis.

- **Longitudinal moment of the shower**

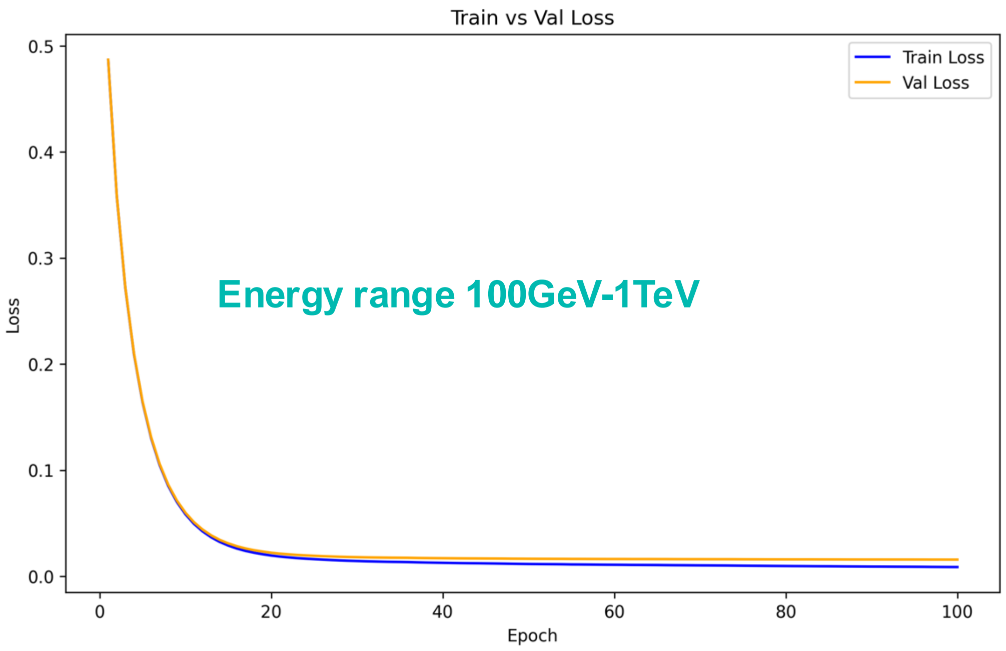
$$M_{long}(n) = \frac{\sum_i E_i\big(\bar{d} \cdot (\bar{x} - x_0)\big)^{n}}{\sum_i E_i},$$

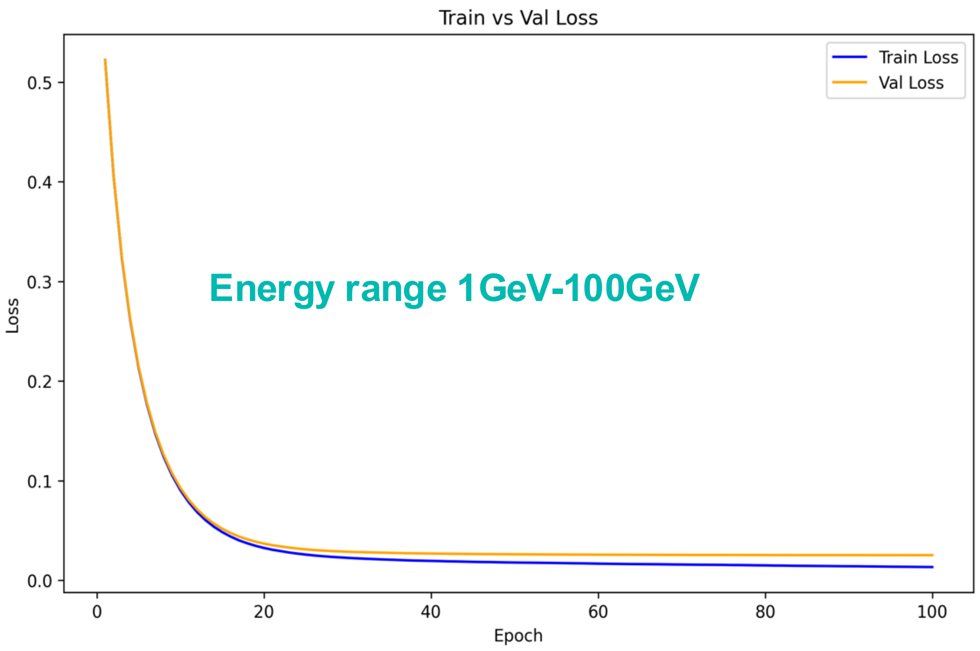
where $\bar{d}$ is the directional vector, which in our case has been chosen as the primary direction, x_i is the i-th activated pixel, x_0 is a reference point, that corresponds to the starting point of the shower.

- **Longitudinal profile of the shower**

Each hit, within a fixed radius, is projected onto the shower axis. The axis is split into 10 equal-length segments. In each segment, the deposited energy from hits is summed. The energy is then normalized by the segment length, yielding the energy density along the shower axis.

XGBoost Algorithm: results





- Accuracy: 99.50 %
- Precision: 99.53 %
- Recall: 99.47 %

- Accuracy: 99.19 %
- Precision: 99.16 %
- Recall: 99.21 %

Conclusions and next steps: GNN

The GNN shows promising performance for tracking tasks compared to traditional analytical approaches, both in terms of accuracy and time efficiency.
We need more simulated data to properly evaluate the performances of the model!!!

Spoiler alert: the perfect classifier? It doesn't exist!

The algorithm's validity is limited beyond a certain energy range!
Between 100 TeV and 1 PeV, the information from the FIT alone is insufficient to distinguish between backscattering and primary tracks.

Conclusion and next steps: Transformer

Two distinct approaches for handling two different tasks in space experiments:

- charge particle track reconstruction
- discrimination between hadronic and electromagnetic showers in calorimeters.

The results obtained are promising, indicating that the AI algorithm can be effectively applied to real-world experiments.

Next steps

What about the GNN???

The GNN does not work for proton in high energy range from TeV to PeV. The performance are quite low.
So we need to:

1) Find the precise validity limits
2) Incorporate additional data from other subdetector (heterogeneous graph neural network and more sophisticated data preprocessing)
3) Obtain a bigger dataset to properly train the model

What about the transformer???

1) Transformer architecture is under testing on a bigger dataset with Optuna for a proper hyperparameter optimization
2) Different energy ranges will be analyzed

Thank you for your attention

Backup slides

Graph neural networks

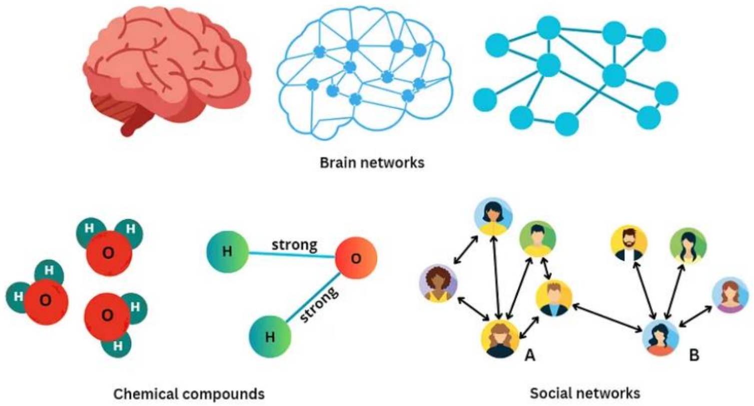
A graph represents the relations (edges or links) between a collection of entities (nodes).
Graph Neural Networks (GNNs) are a class of deep learning models that are designed to operate on graph-structured data.
They have shown remarkable success in tasks such as node classification, link prediction, and graph classification.
The key idea behind GNNs is to learn representations for nodes and edges in a graph by aggregating information from their local neighborhood.

A GNN consists of a number of layers, each of which updates the representation of each node based on its local neighborhood.

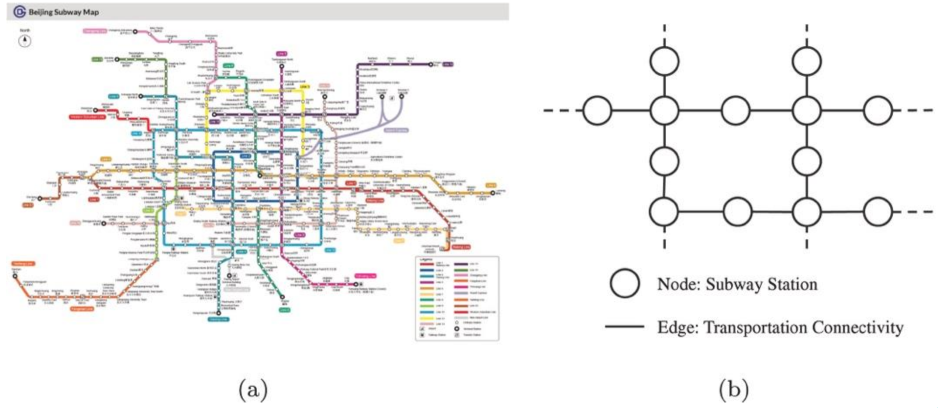
The representation of each node is typically a low-dimensional vector that encodes the node's properties and its relationships with other nodes.

The layers of a GNN are designed to capture increasingly complex features of the graph by aggregating information from the neighborhood of each node.

The key component of a GNN layer is the aggregation function, which takes as input the representations of a node's neighbors and produces a new representation for the node.



- *A Gentle Introduction to Graph Neural Networks*, https://distill.pub/2021/gnn-intro/
- Hands-On Graph Neural Networks Using Python, M.Labonne, Packt Publishing Ltd.

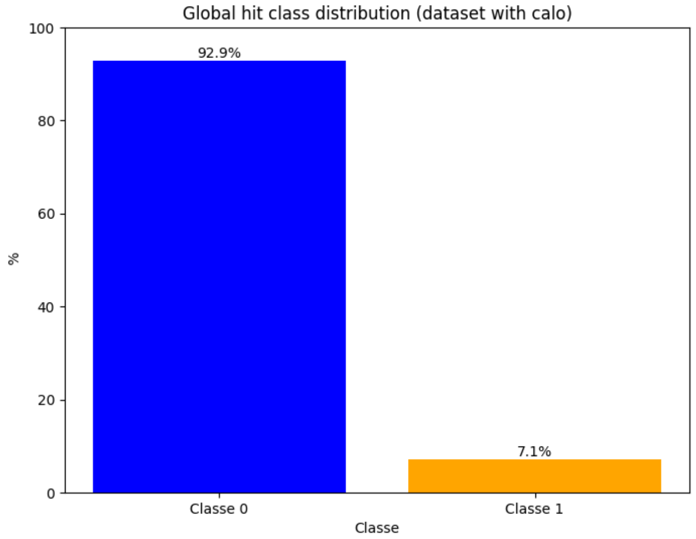


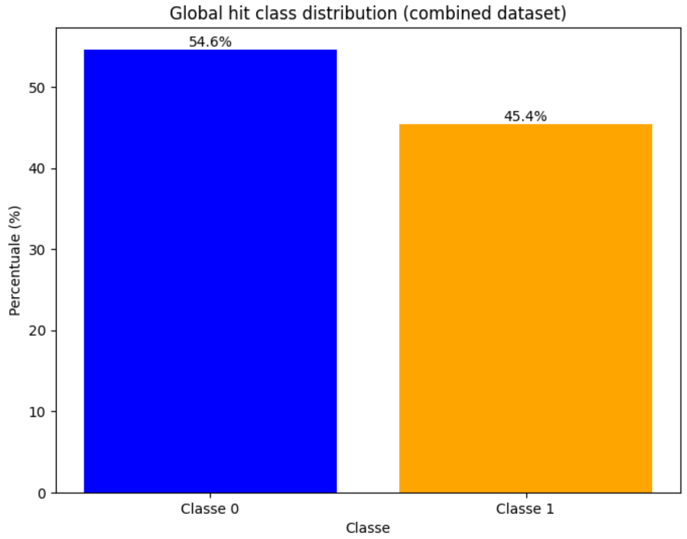
Balancing FIT dataset

Data are highly imbalanced, since there are many bacskattering tracks originating from the calorimeter, which interfere with the correct identification of the primary particle's trajectory

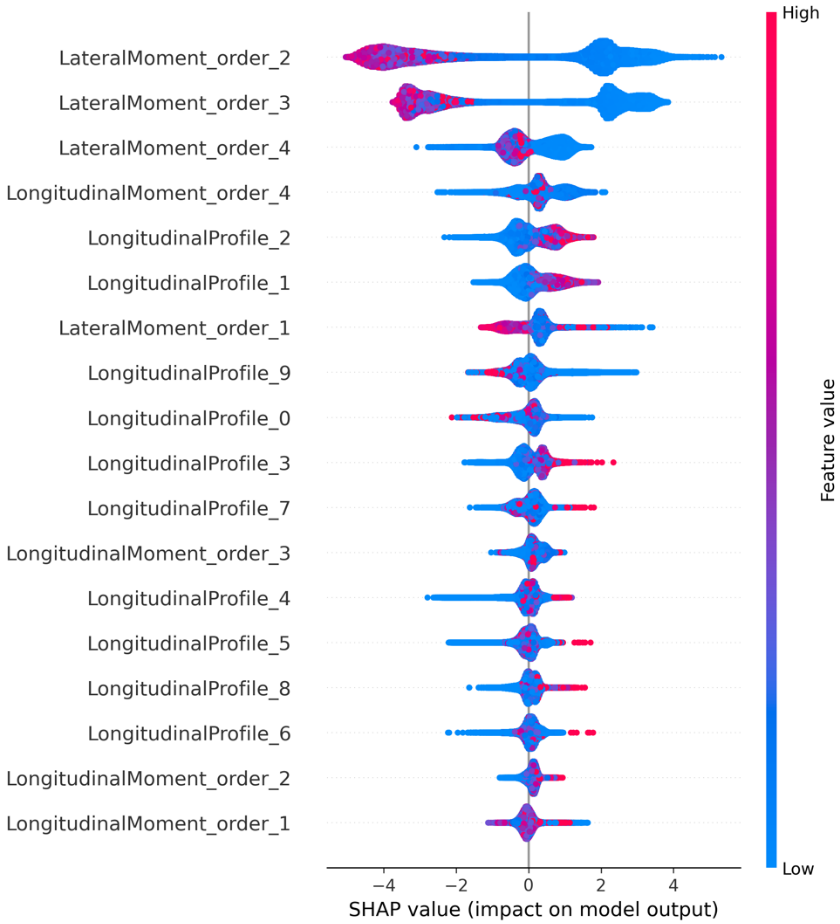
To mitigate the imbalance:

1. Clustering algorithm
2. Cut of noise clusters “far away” from the primary track
3. Adding simulated events without calorimeter

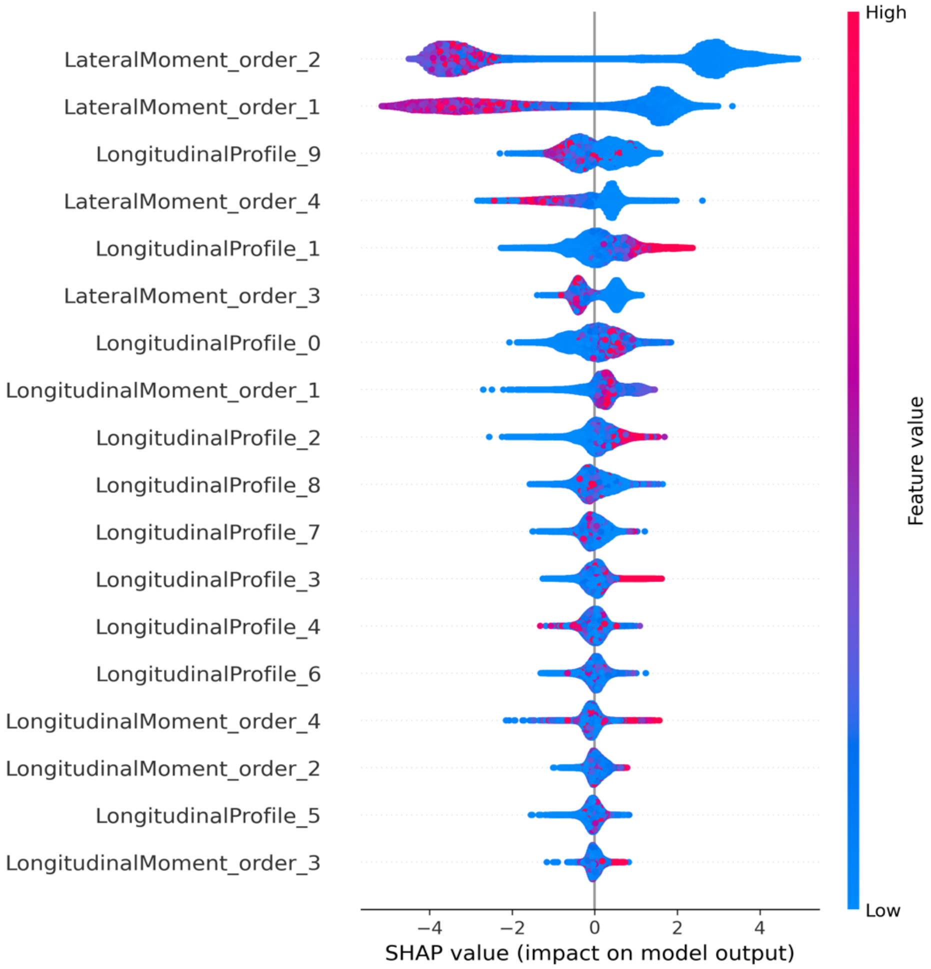




XAI Techniques for XGBoost



High energy range



Low energy range