Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
ICSC Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

Graph neural network for track reconstruction in space experiments

Dr. Federica Cuna and dr. Fabio Gargano

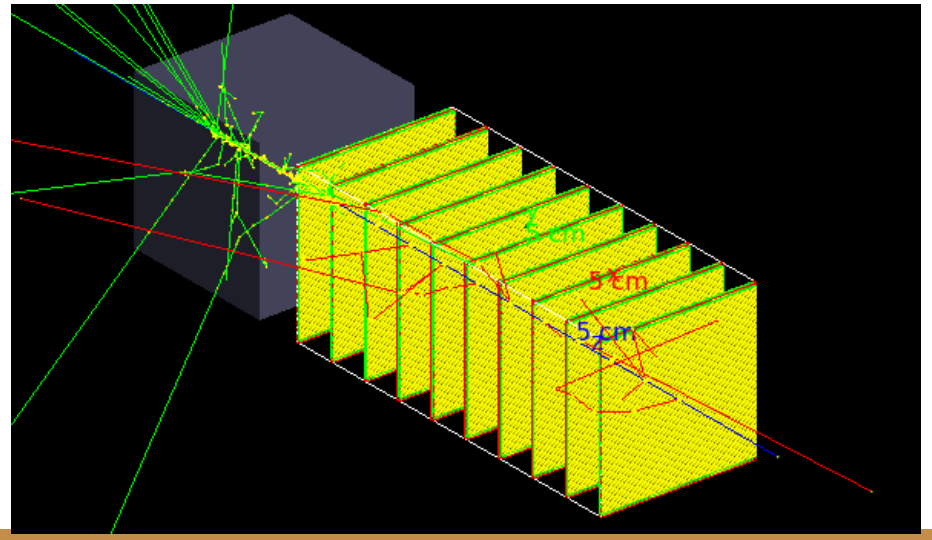
Spoke 3 Technical Workshop, Trieste October 9 / 11, 2023

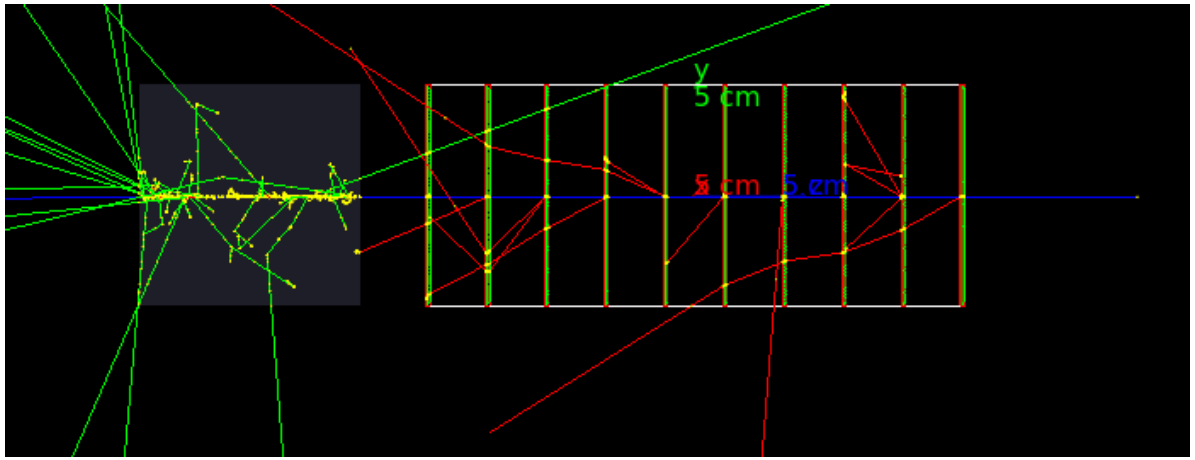
Scientific rationale

Development of tracking algorithm with deep learning techniques

A range of models inspired by computer vision applications were investigated, which operated on data from tracking detectors in a format resembling images [*A deep learning method for the trajectory reconstruction of cosmic rays with the DAMPE mission, Andrii Tykhonov et al, Astroparticle Physics 146, April 2023, 102795*].
Although these approaches demonstrated potential, image-based methods encountered difficulties in adapting to the scale of realistic data, primarily due to the *high dimensionality and sparsity of the data*.
Tracking data are naturally represented as graph by identifying hits as nodes and tracks segments as (in general) directed edges. This leads to the investigation of the **geometric deep learning approach**.

We implemented an algorithm which exploits the potentials of the **Graph Neural Networks (GNN)**, a subset of GDL algorithm, for the task of track reconstruction in a toy model of space experiment.

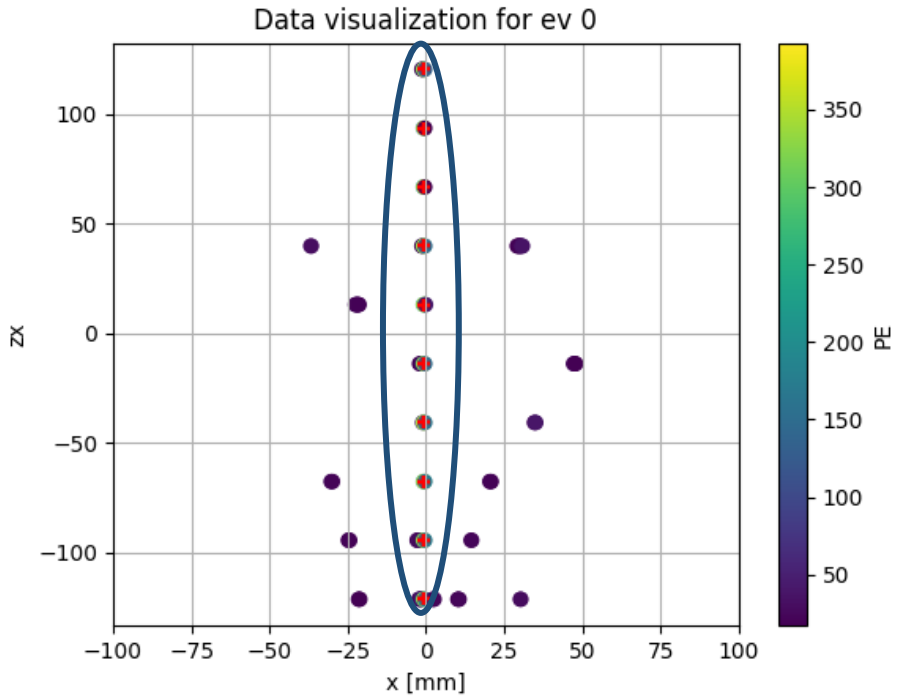


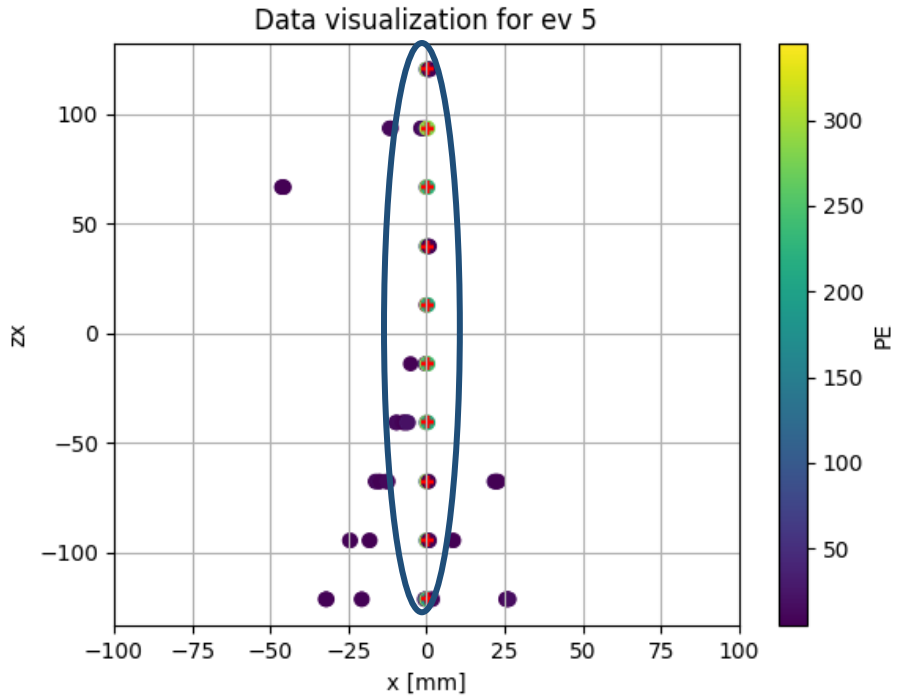


Event display

Each event contains the hits corresponding to the primary track and other hits which correspond to the backscattering tracks.

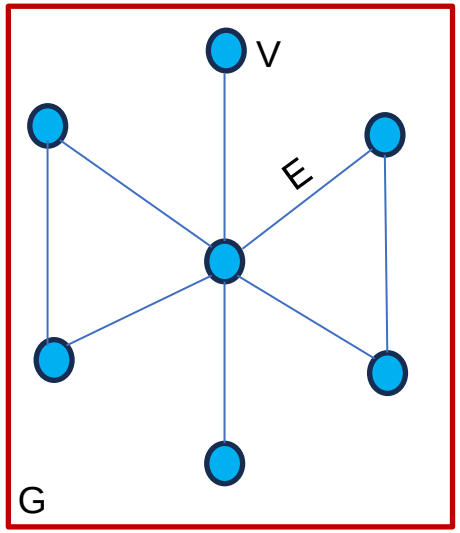
Milestone: distinguish between primary and backscattering tracks.

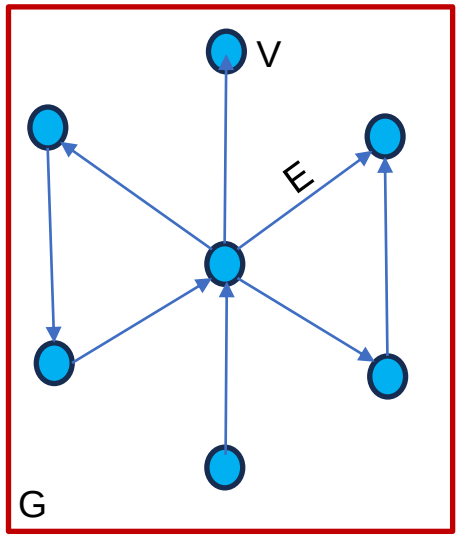


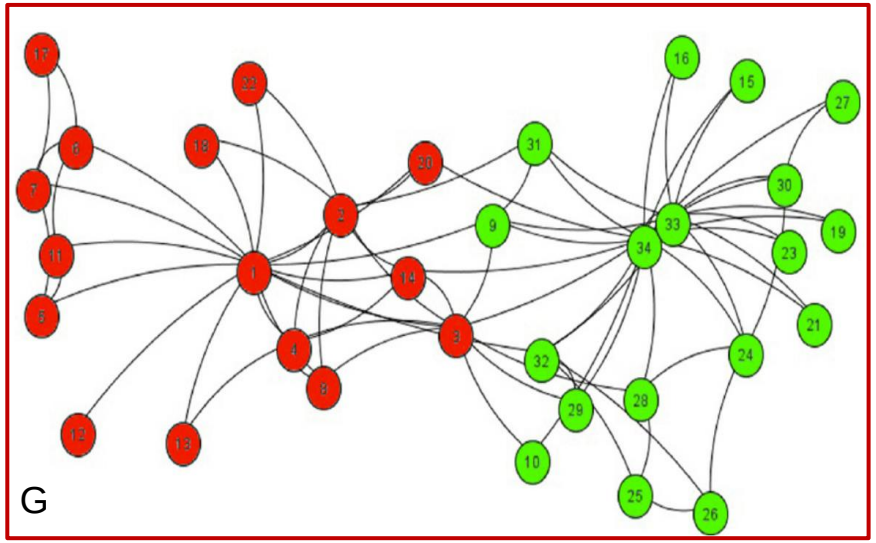


Technical Objectives, Methodologies and Solutions: graph neural network (GNN)

A graph represents the relations (edges or links) between a collection of entities (nodes).







Graph neural networks are a type of neural network that is designed to process graph-structured data.

The architecture of graph neural networks

Each node in the graph has a feature vector representing its attributes (categorical or continuous).

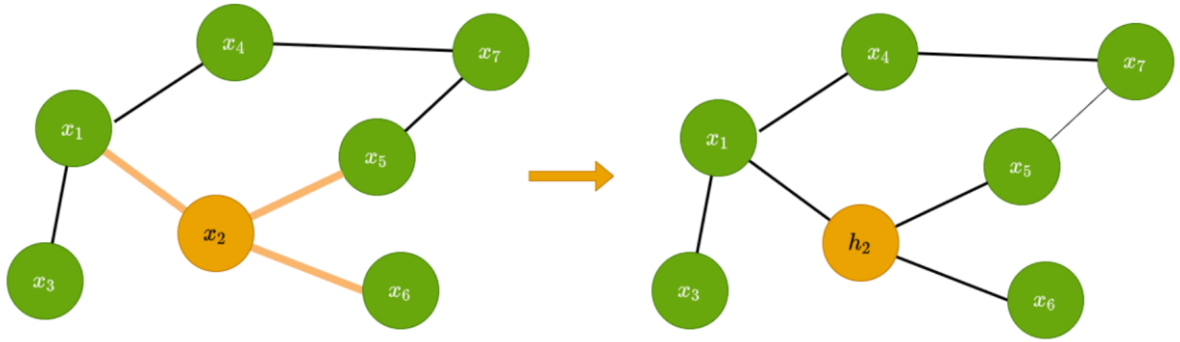
Edges between nodes can also have associated features, capturing more nuanced information about the relationships between nodes.

The core idea of GNNs is to iteratively update the node representations by **passing and aggregating messages from neighboring nodes**.

This process, called **message passing**, involves computing a message vector for each neighboring node using a message function.

The message function takes the features of the sender node, receiver node and edge (if present) as input and produces a message vector.

This message-passing process is performed through multiple layers or “hops” to capture higher-order relationships within the graph.



$$h_2 = g(x_1, x_5, x_6)$$

GNN applications:

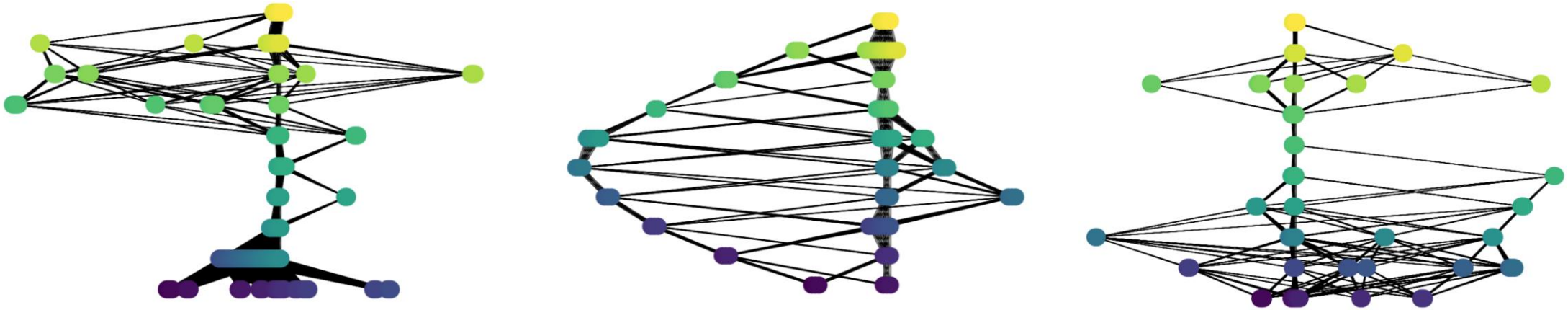
- Link prediction
- Node classification
- Graph classification

- *A Gentle Introduction to Graph Neural Networks,* https://distill.pub/2021/gnn-intro/
- *Introducing TensorFlow Graph Neural Networkshttps://blog.tensorflow.org/2021/11/introducing-tensorflow-gnn.html*
- *TensorFlow-GNN: An End-To-End Guide For Graph Neural Networks,* https://towardsdatascience.com/tensorflow-gnn-an-end-to-end-guide-for-graph-neural-networks-a66bfd237c8c

Accomplished Work, Results: GNN for node task classification

We developed a graph neural network for node classification, by using GNN-Tensorflow library.

Nodes are the hits inside the tracking detector and edges are the inter-layer hit connection.



The software development has been accomplished by using an in-local JupyterHub with GPU.

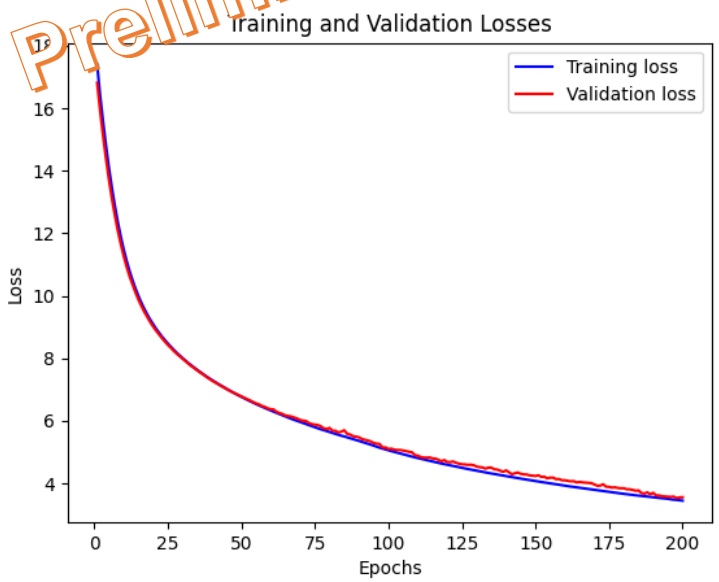
The GPU for the JupyterHub instance is a partitioned GPU created from a 40GB Nvidia A100.

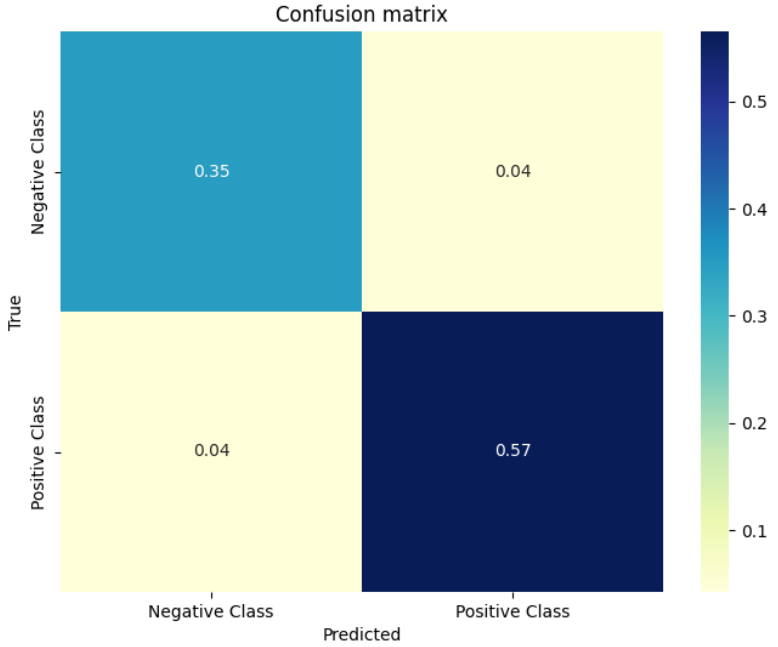
In details, from a single A100, 7 GPU were generated, each with 5 GB of dedicated memory.

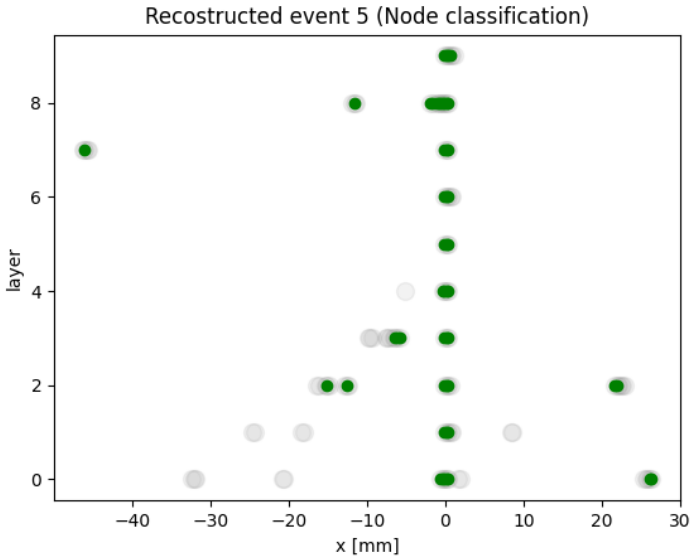
The computing power of each partitioned GPU is also 1/7 of an A100

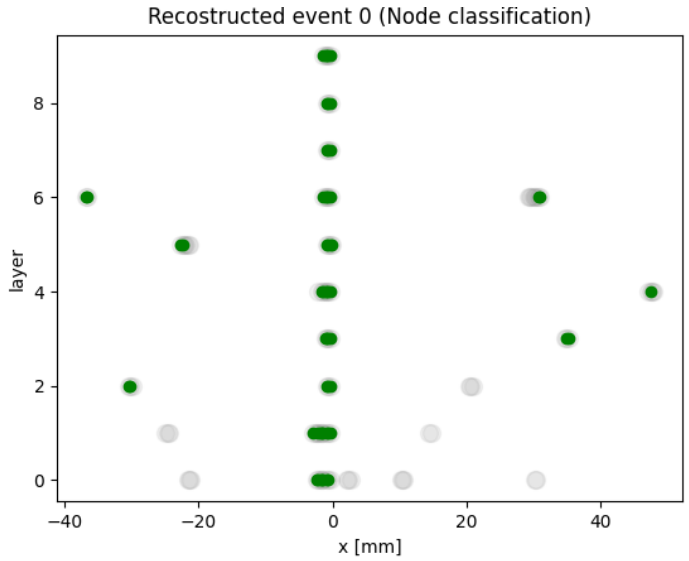
Accomplished Work, Results: GNN performances

Preliminary









The algorithm seems promising, but it need to be optimized and tested on larger dataset.

KPIs/milestones/timescale

- Improving the GNN tracking algorithm (*by the end of the year*): adding link prediction, using PyTorch libraries…
- Testing the algorithm on big datasets, by using docker container implementation (*by the end of the year/first months of 2024* - **April 2024 checkpoint**)
- Adapt the algorithm to in orbit or future space experiments, like DAMPE and HERD (*2024/2025*)

Useful tools

- GitHub/GitLab to store and share code
- JupyterHub with GPU
- Docker registry

Note

As low priority task, we plan to develop AI algorithms for particle identification, which will exploit the information of the different subdetectors of space experiments *(2024/2025).*

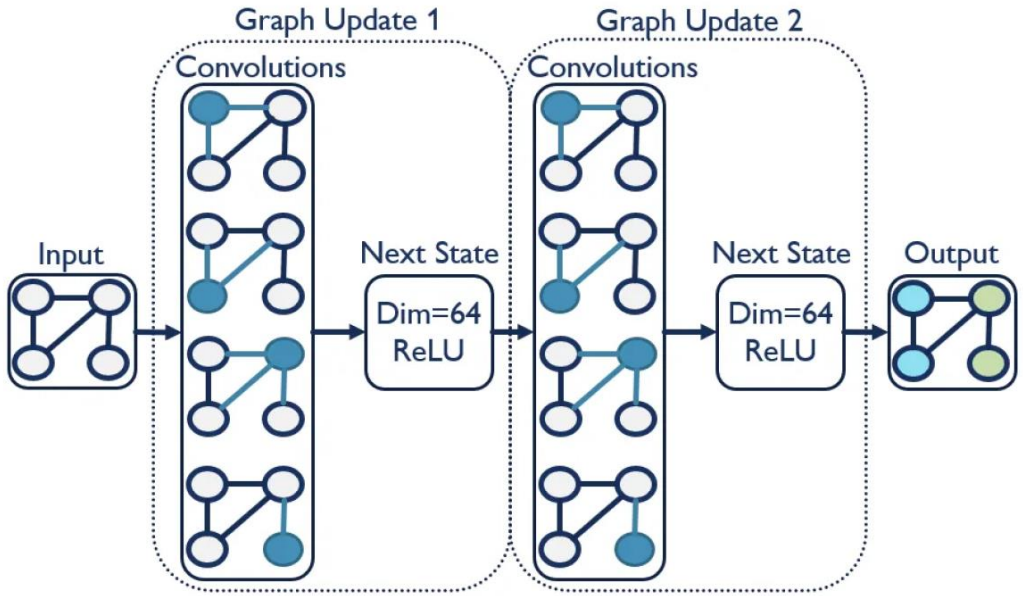
Thank you for your attention!

BACKUP

GNN IMPLEMENTATION

Node model (Graph convolutional neural network)

- Features for the nodes and edges are initialized, by creating a dense layer for each feature, with ‘Relu’ activation function.
- The initialized graph features are fed to a function called GraphUpdate.
- We set 10 iterations and in each iteration the graph undergoes an update process: nodes of the graph are updated by using information aggregated from their neighbors.
- With each ‘graph_updates’ loop, the previous graph becomes the input for the new ‘GraphUpdate’ along with a dense layer.
- At the end of the update iterations, an output probability is calculated using a fully connected layer with a "sigmoid" activation.



Toy model for space experiments

The toy model for a space experiment consists of 10 fibers tracking layers and a calorimeter.
The model has been simulated with Geant4 toolkit.

Geant4 simulation settings details:

- 10 tracking fibers layers
- Tracker gap 25 mm
- Fiber radius 0.25 mm
- Fiber length 10 cm
- Number of fiber per view 2*192
- Calorimeter length in z 10 cm
- Calorimeter pixel size 1 cm
- Calorimeter gap 3 cm

- Particle Carbon ion
- Energy 50 GeV
- Pos shape Rectangle
- Position at center (0,0,1)
- Position halfx 0.1 cm
- Position halfy 0.1 cm
- Direction (0,0,-1)

- Number of events 100k