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

# *Artificial intelligence for tracking in space experiments*

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

**Spoke 3 General Meeting,** Elba 5-9 / 05, 2024

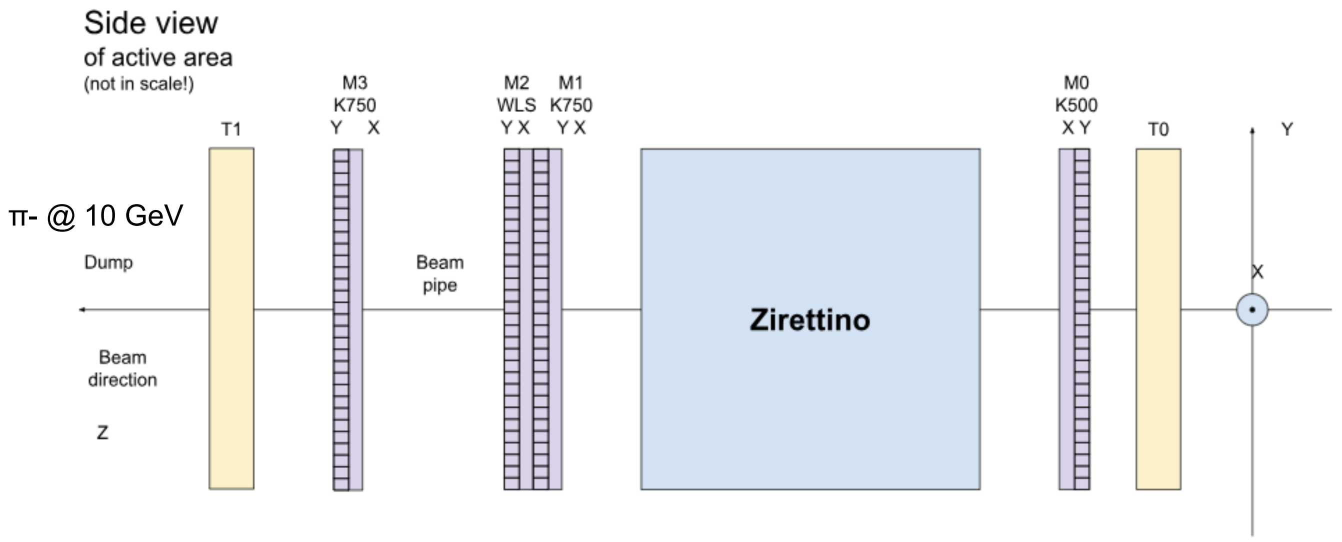
# Scientific Rationale: AI for tracking in space experiments

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 model of space experiment.




## Beam test set up at Cern T10

The set up has been simulated by using Geant4 toolkit.
Zirettino has not been included for now.
A beam of π- of 10 GeV with inclined tracks of 0.5 deg has been simulated.
M0,M1,M2,M3 are fiber tracking layers.
M2 consists of WLS with a 3 mm thick LYSO crystal in between.
The simulation includes the LYSO crystal but considers the fiber as the scintillating ones.

Random noise hits have been added to simulate properly the experimental response of each tracking layer.

# Technical Objectives, Methodologies and Solutions: Graph neural networks

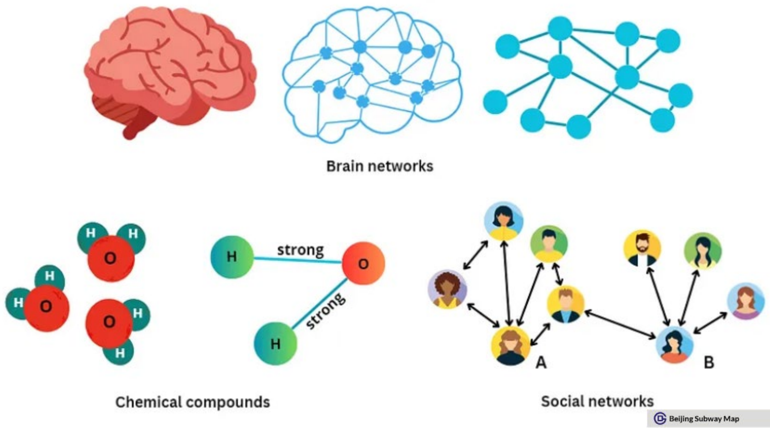
A graph represents the relations (edges or links) between a collection of entities (nodes).
Graph Neural Networks (GNNs) are a class of deep learning models that are designed to operate on graph-structured data.
They have shown remarkable success in tasks such as node classification, link prediction, 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.

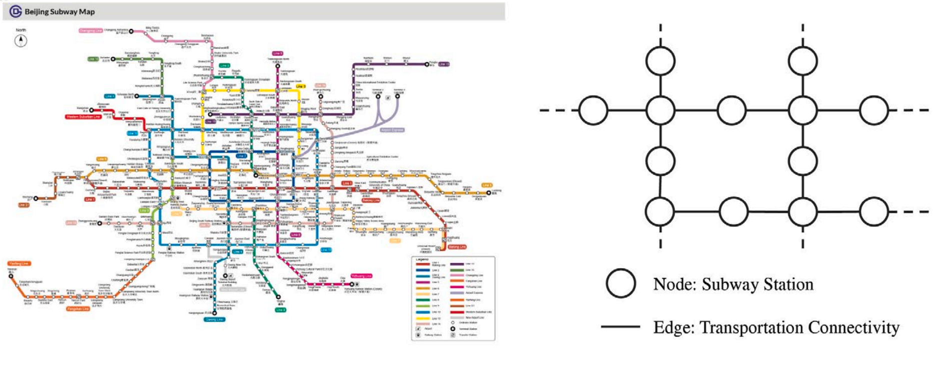
The representation of each node is typically a low-dimensional vector that encodes the node’s properties and its relationships with other nodes.

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/](https://distill.pub/2021/gnn-intro/)
- Hands-On Graph Neural Networks Using Python, M.Labonne, Packt Publishing Ltd.



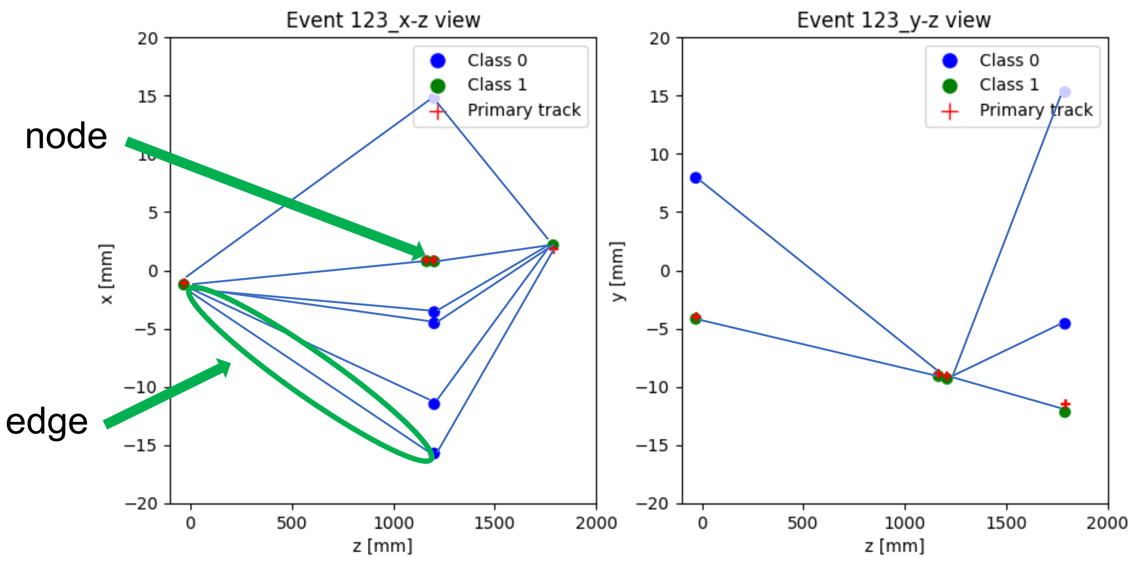
(a) (b)

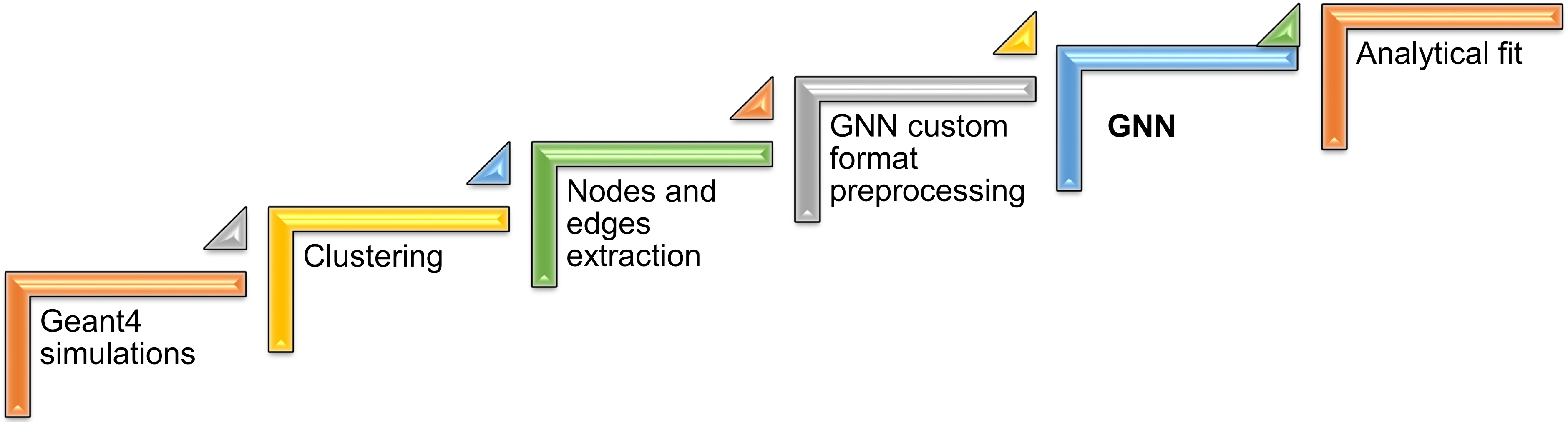
# Accomplished Work, Results

Graphs are a natural way to represent tracks!

We developed graph neural networks for node classification, by using PyTorch geometric library.

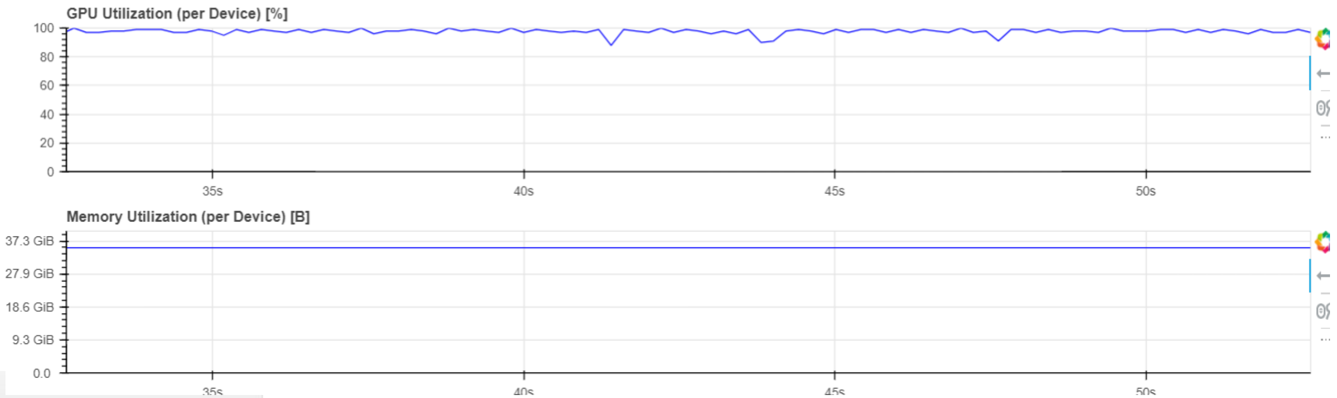
Nodes are the hits inside the tracking detector and edges are the inter-layer hit connection.



## Software development tools

- 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.
- Tests on the GPU utilization have been accomplished by using a in-local JupyterLab instance with 40GB Nvidia A100-GPU.
- Training of the GNNs has been accomplished by using a in-local GPUs cluster. The available GPUs are NVIDIA A100 40GB and NVIDIA V100 32GB




+ ADD JOB

| SUCCESS 0 | FAILURE 0 | FRESH 6 | RUNNING 4 | QUEUED 2 | IDLE 0 |
|---|---|---|---|---|---|

| JOB | NEXT RUN | STATUS | STATE | ACTIONS |
|---|---|---|---|---|
| federicacuna-SageConv-long-optim0-2024-3-21-9-17-48-17830 | in ~365 days | fresh | 1 running | |
| federicacuna-SageConv-long-optim1-2024-3-21-9-17-48-18015 | in ~365 days | fresh | 1 running | |

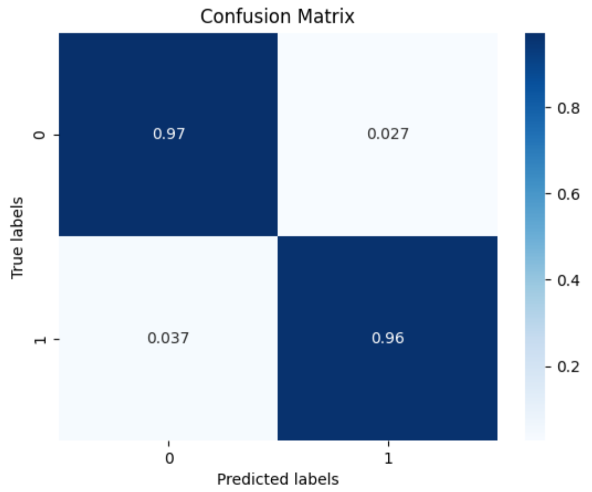
*Thanks to dr G.Donvito and dr G.Vino from Recas-Bari.*

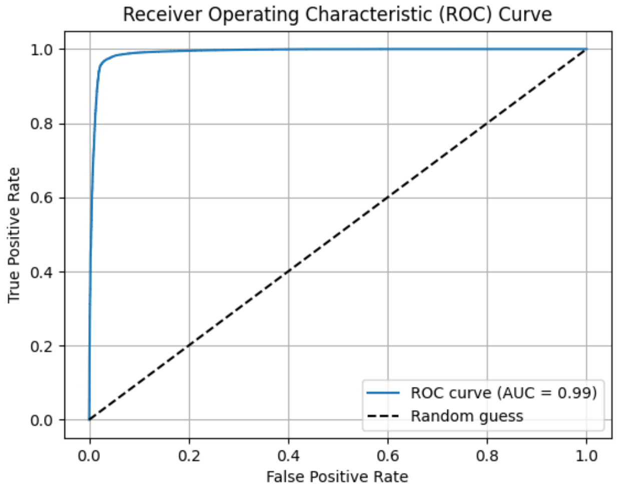
# The SageConv algorithm

The SageConv architecture is a variant of the GraphSAGE architecture.
It uses a more expressive convolutional operator, which allows it to capture more complex features.
The SageConv aggregation function takes into account the degrees of the nodes in the neighborhood.
SageConv uses the average of the representations of the neighbors, normalized by the degree of each neighbor, as the aggregate representation. This allows it to capture more fine-grained information about the structure of the graph. It also uses skip connections to facilitate gradient flow during training. Specifically, the output of each layer is combined with the input representation of the node. The concatenated vector is then passed through a fully connected layer to produce the final output of the layer.

| CV 5 fold | accuracy | recall | precision |
|---|---|---|---|
| train data | 0.972±0.007 | 0.968 ± 0.009 | 0.9902 ± 0.0014 |
| validation data | 0.972±0.007 | 0.968 ± 0.009 | 0.9862±0.0024 |

- 2,6 million event
- 500 epochs,
- lr =0.0001,
- aggregation function=mean

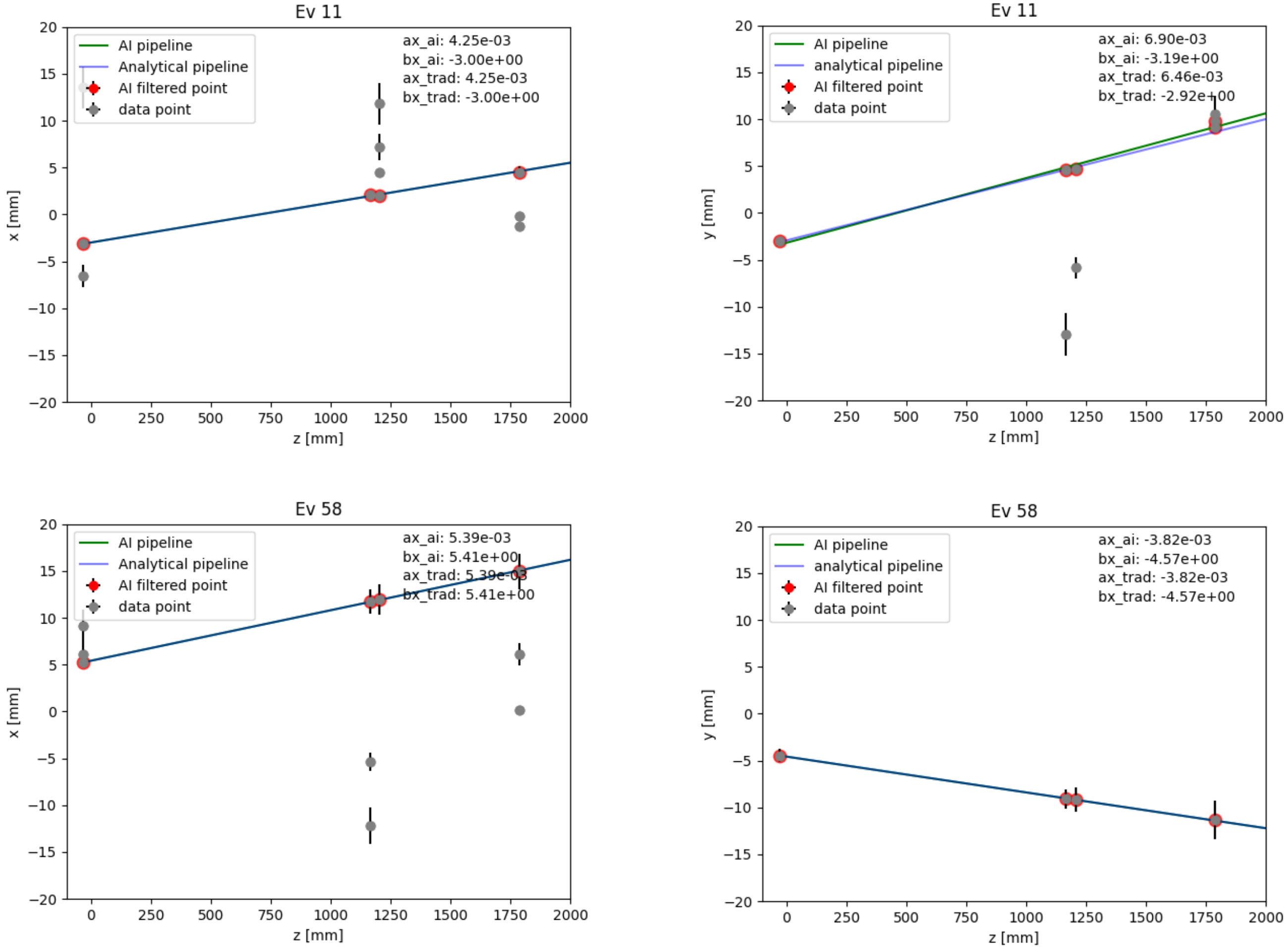



Performances on test data
accuracy 0.9665
recall 0.9631
precision 0.9873

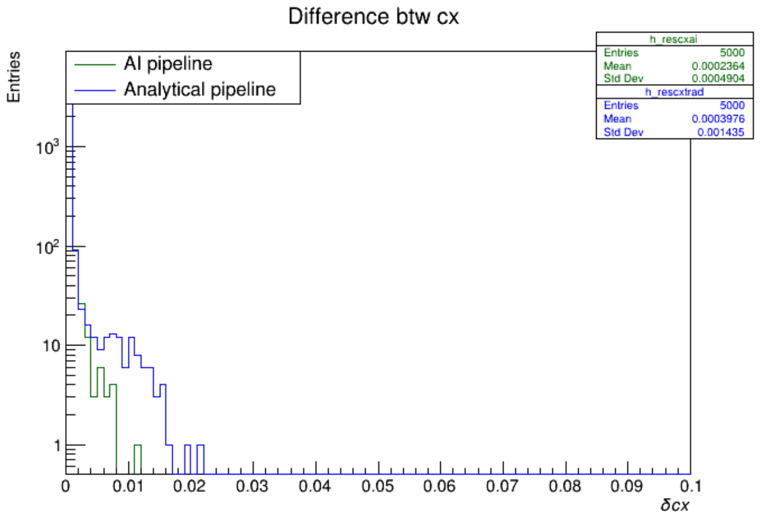
## The SageConv algorithm: comparison with traditional pipeline

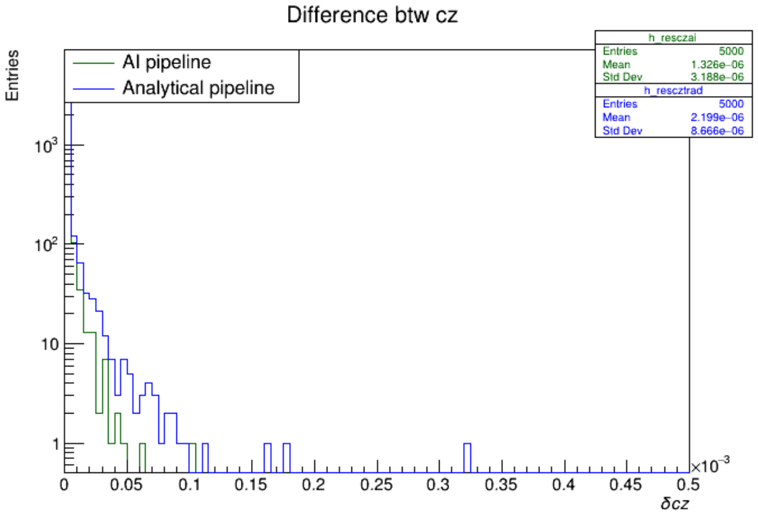
The traditional tracking pipeline minimizes the chi-square between the hits inside each events, then fit the track with a linear function.
The AI pipeline performs the selection of good hits and fit the track with the same linear function.

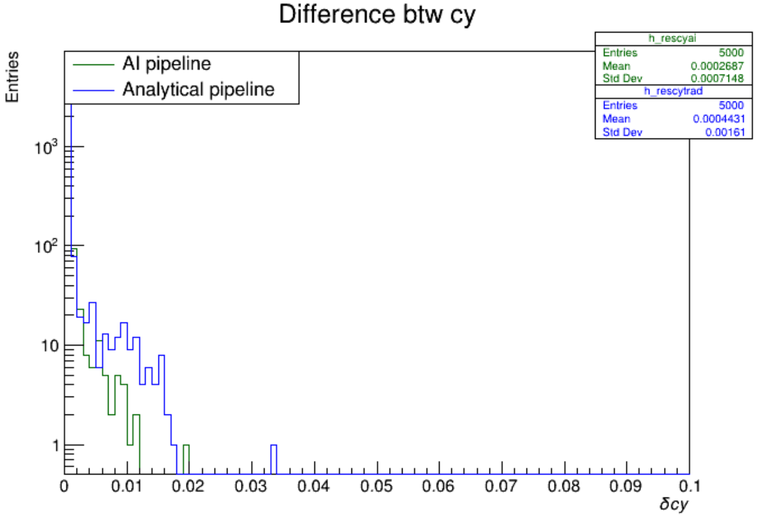


# The SageConv algorithm: comparison with traditional pipeline

To evaluate the AI pipeline, we compare the differences between the monte carlo director cosines and the ones evaluated by applying the AI and traditional pipeline



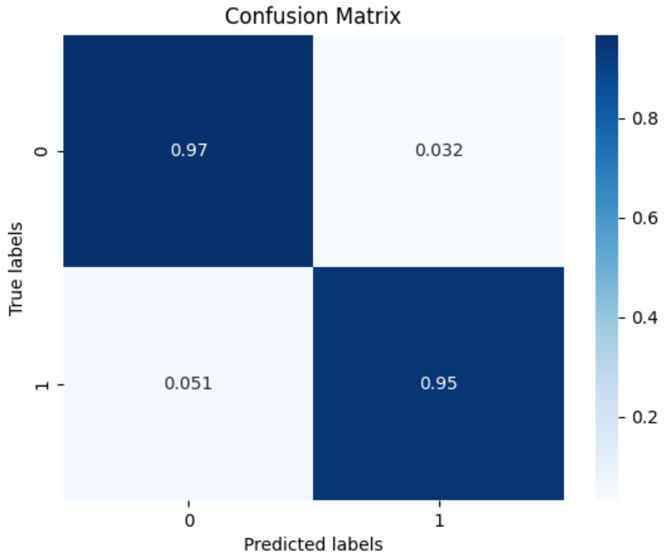

To process 5000 tracks the analytical pipeline takes 9 min and 15 s, the AI pipeline takes 114 ms!

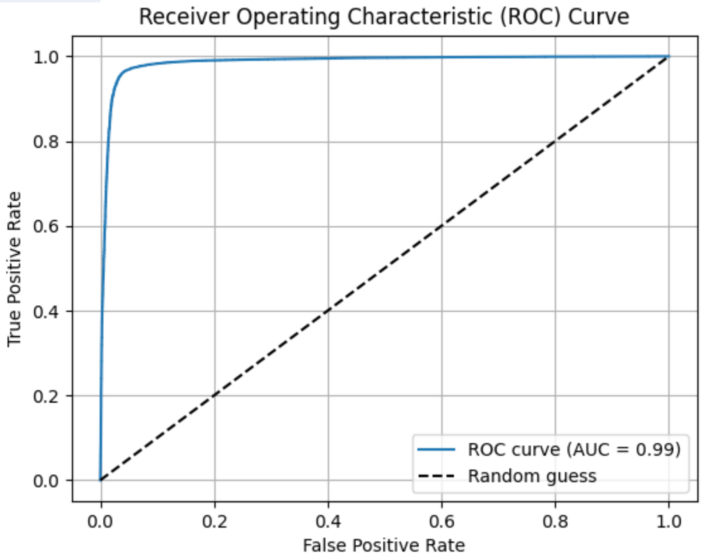
# GAT algorithm

Graph Attention Networks (GATs) are a variant of Graph Neural Networks (GNNs) that leverage attention mechanisms for feature learning on graphs.
In standard GNNs, such as Graph Convolutional Networks (GCNs), the feature update of a node is typically the average of the features of its neighbors. This approach does not differentiate between the contributions of different neighbors.
GATs, on the other hand, assign an attention coefficient to each neighbor, indicating the importance of that neighbor's features for the feature update of the node. These coefficients are computed using a shared self-attention mechanism, which calculates an attention score for each pair of nodes. The scores are then normalized across each node's neighborhood using a SoftMax function.

| CV | accuracy | recall | precision |
|---|---|---|---|
| train data | 0.941± 0.024 | 0.926 ± 0.033 | 0.9860 ± 0.0038 |
| validation data | 0.941 ± 0.024 | 0.926 ± 0.033 | 0.9859 ± 0.0038 |

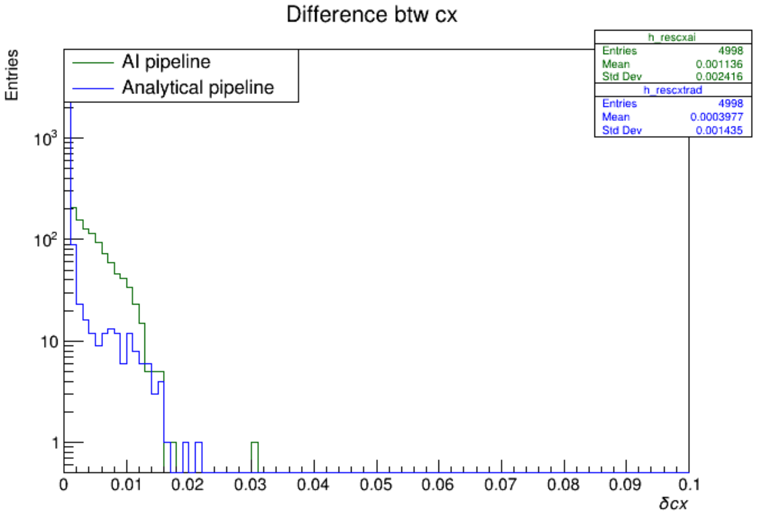
600 epochs
2,6 million events
learning rate: 1e-4

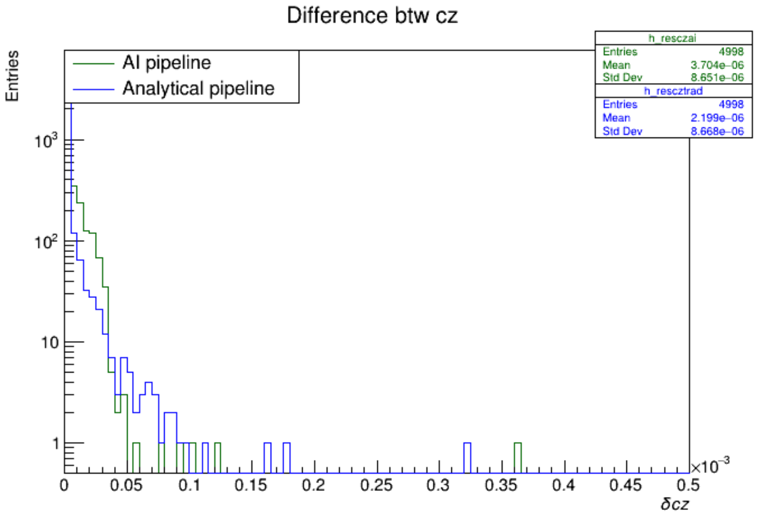


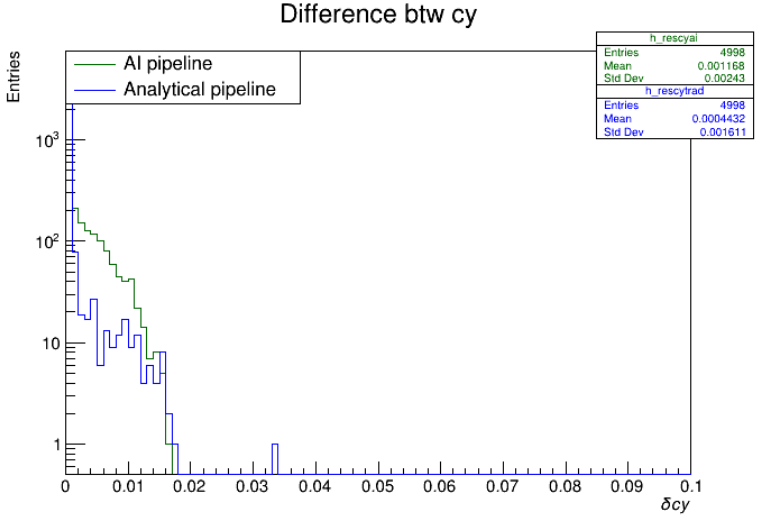




Performances on test data
Accuracy: 0.9557
Recall: 0.9493
Precision: 0.9849

## The GAT algorithm: comparison with traditional pipeline



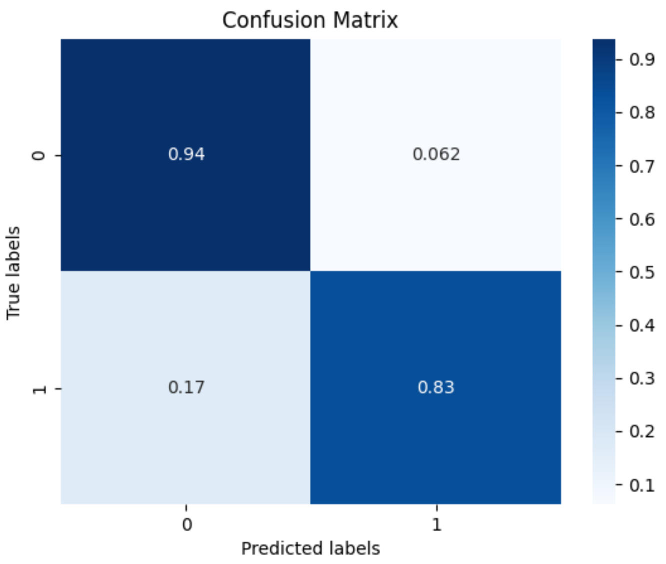



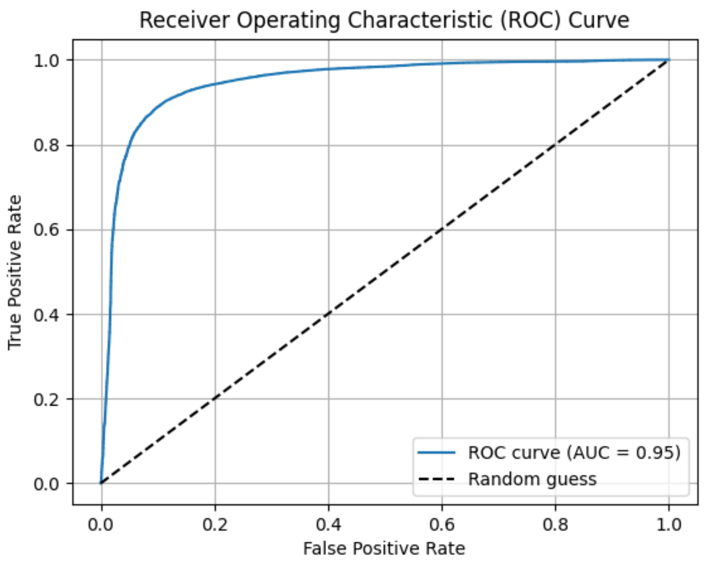




The Gat performances are similar in terms of time consumption to the SageConv ones.
Anyway, for two events, it does not recognize good hits (false negatives).

# GCN algorithm

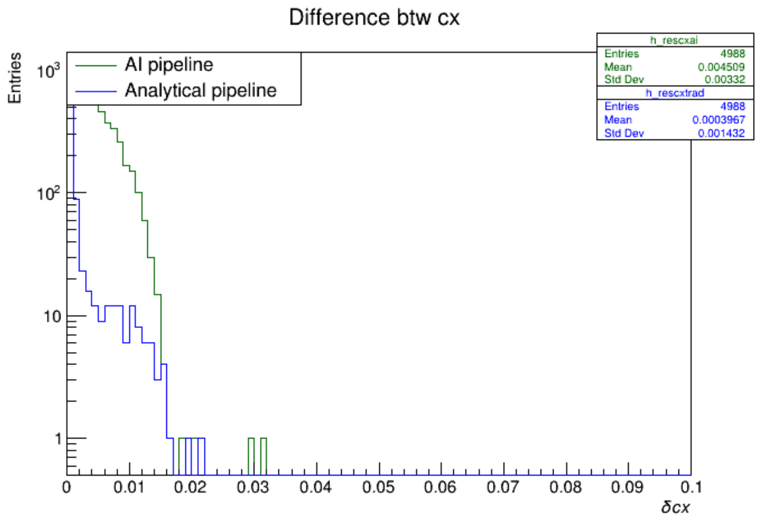
The general idea of GCN is to apply convolution over a graph. Instead of having a 2-D array as input as in the classical CNN algorithm, GCN takes a graph as an input

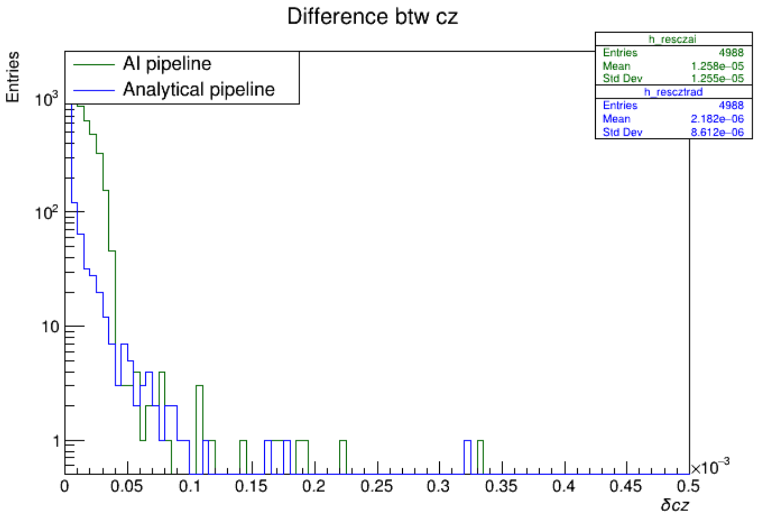


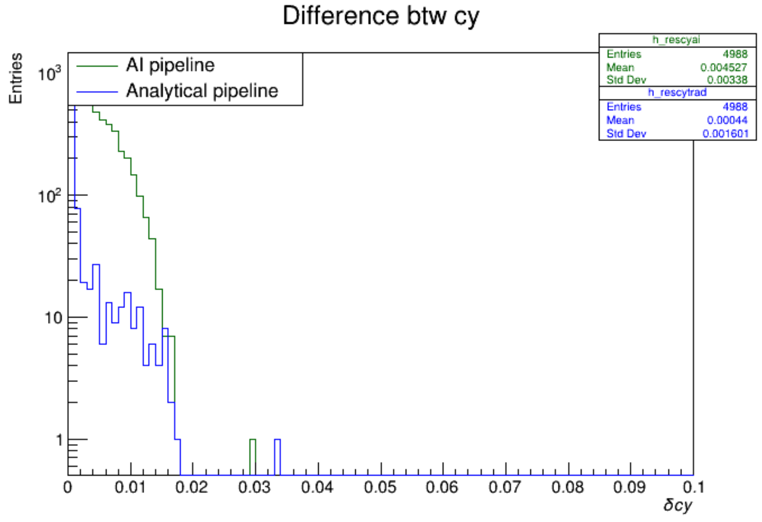




Algorithm performances:
1500 epochs
2 million events
lr 5e-4
Accuracy: 0,8662
Recall: 0,8326
Precision: 0,9663

## The GCN algorithm: comparison with traditional pipeline

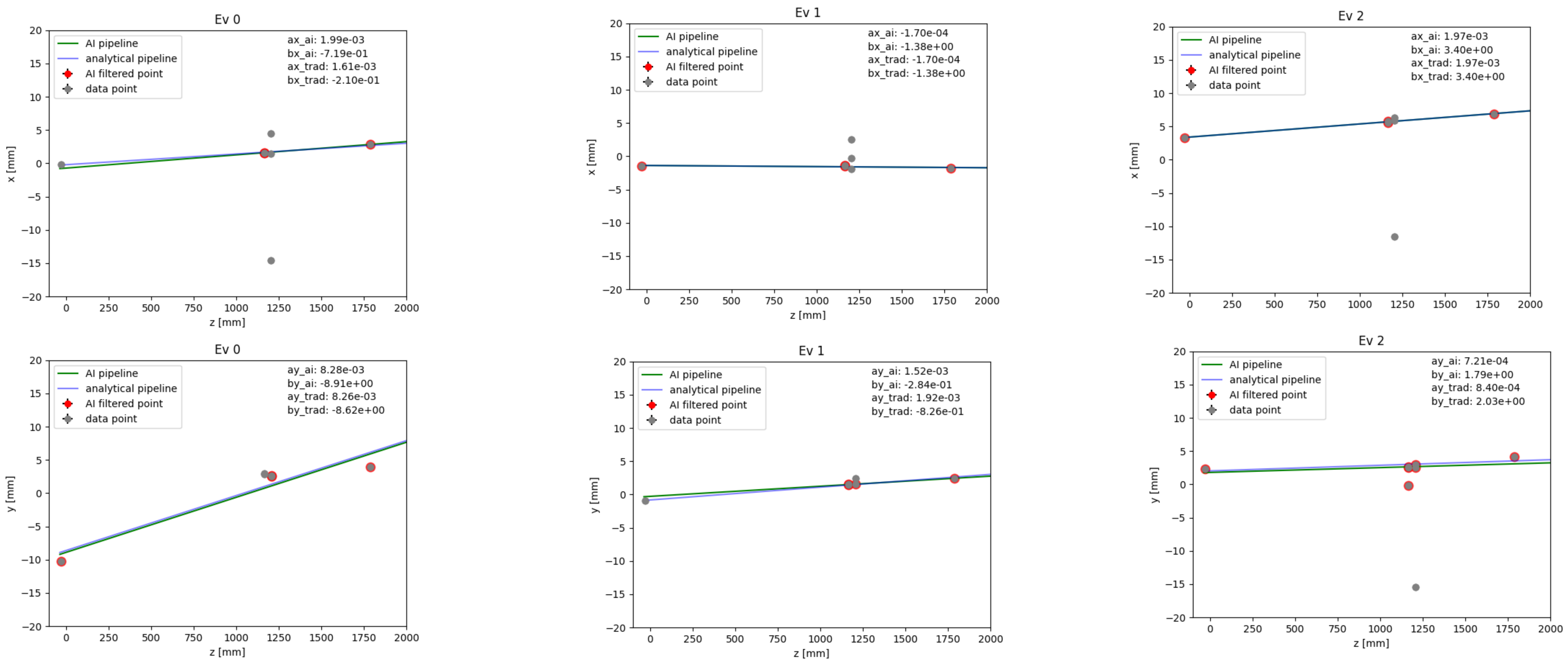


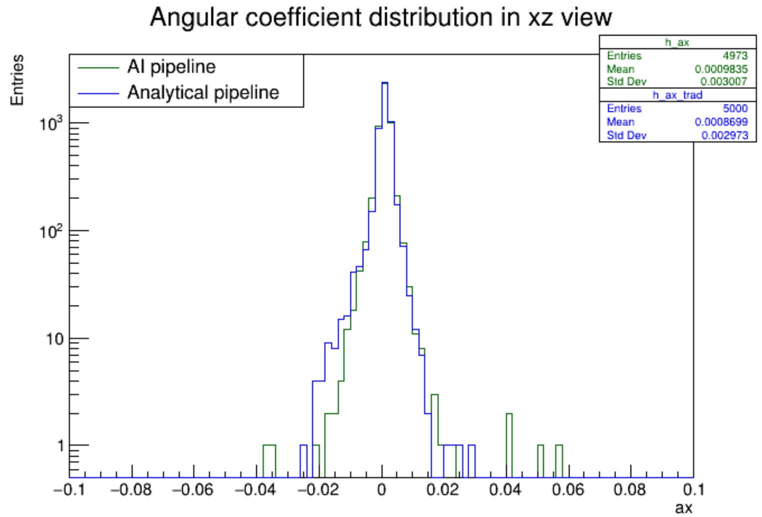

The GCN performances are similar in terms of time consumption to the SageConv ones.
Anyway, for 11 events, it does not recognize good hits (false negatives).

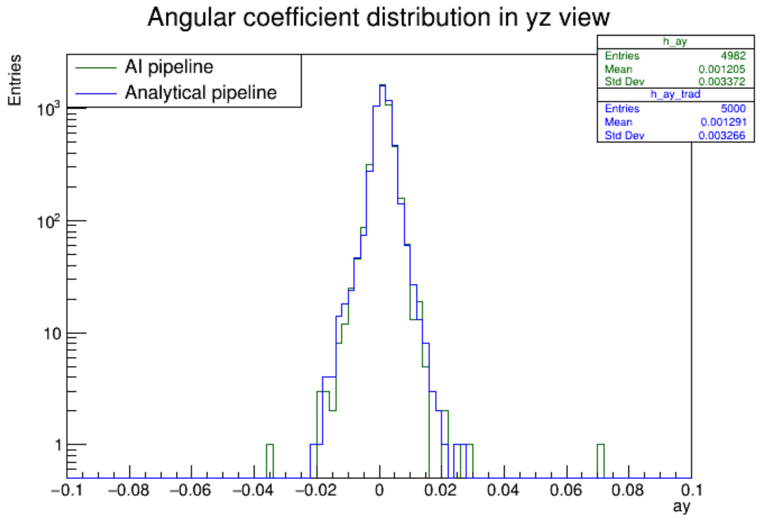
# Beam test data
# SageConv algorithm

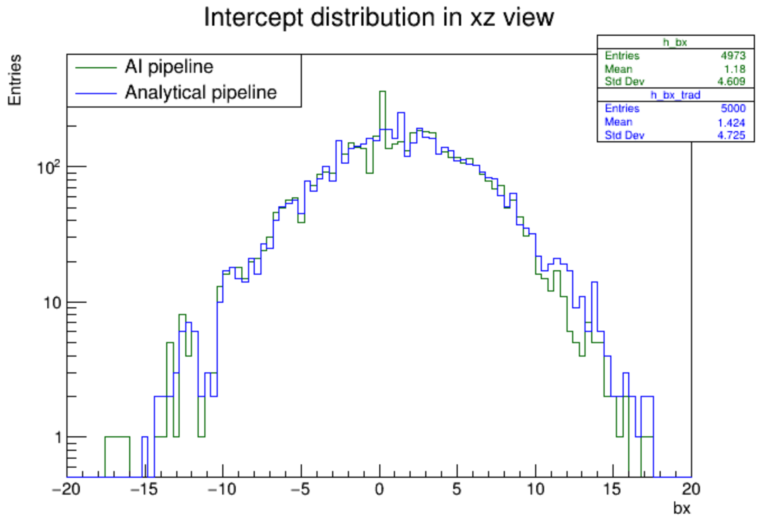


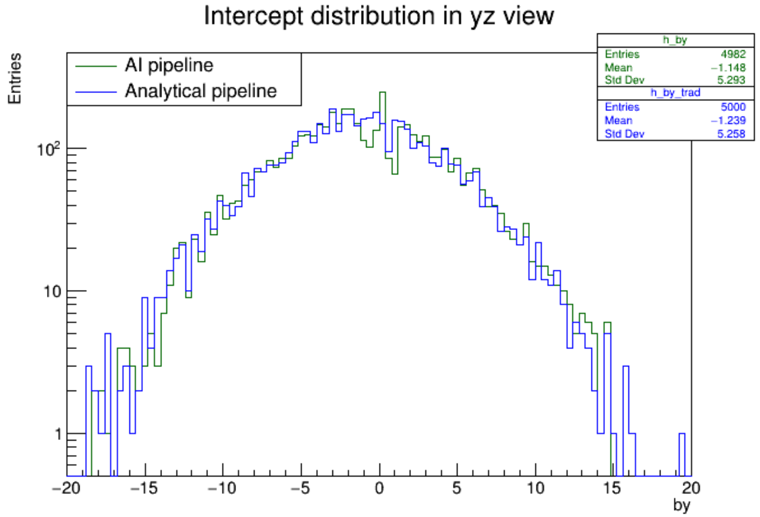
Finanziato dall'Unione europea
NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani
PIANO NAZIONALE DI RIPRESA E RESILIENZA
ICSC
Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

# Track fit parameters distributions



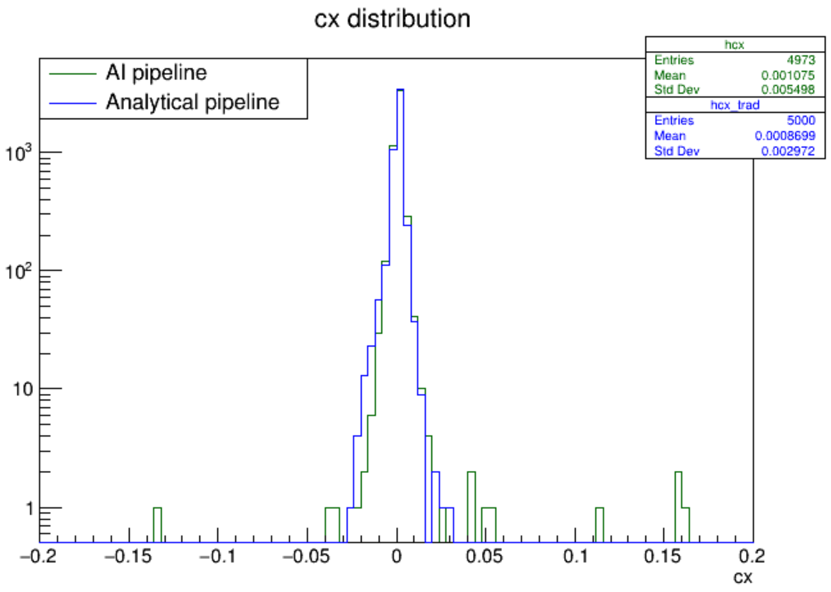
Finanziato dall'Unione europea
NextGenerationEU

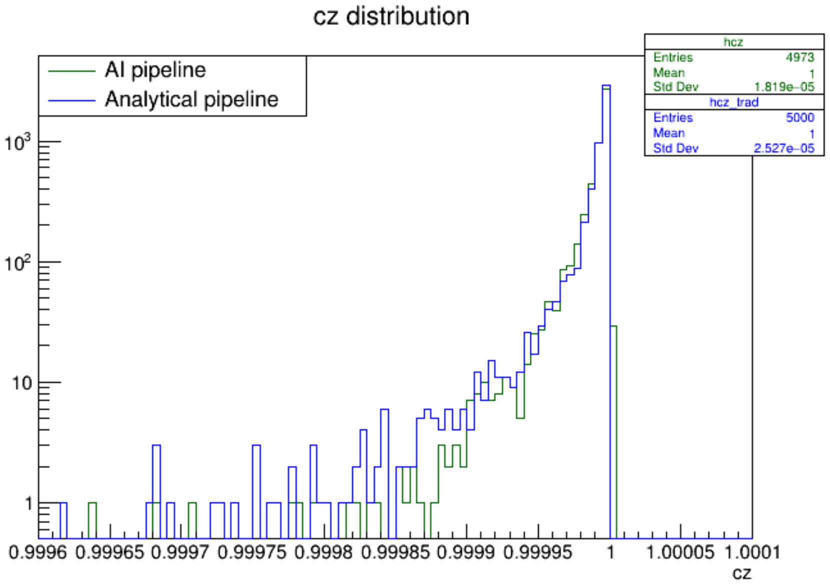
Ministero dell'Università e della Ricerca

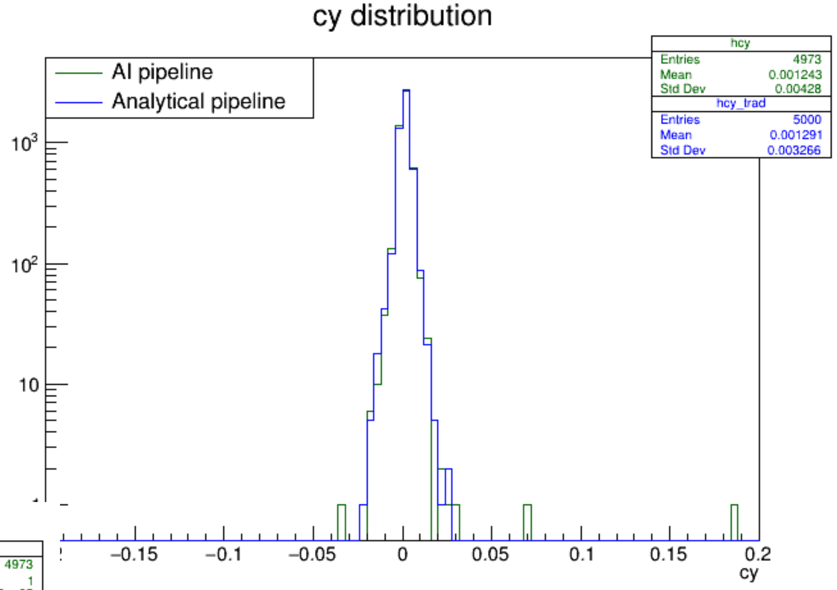
Italiadomani
PIANO NAZIONALE DI RIPRESA E RESILIENZA

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

# Cosine directors distributions



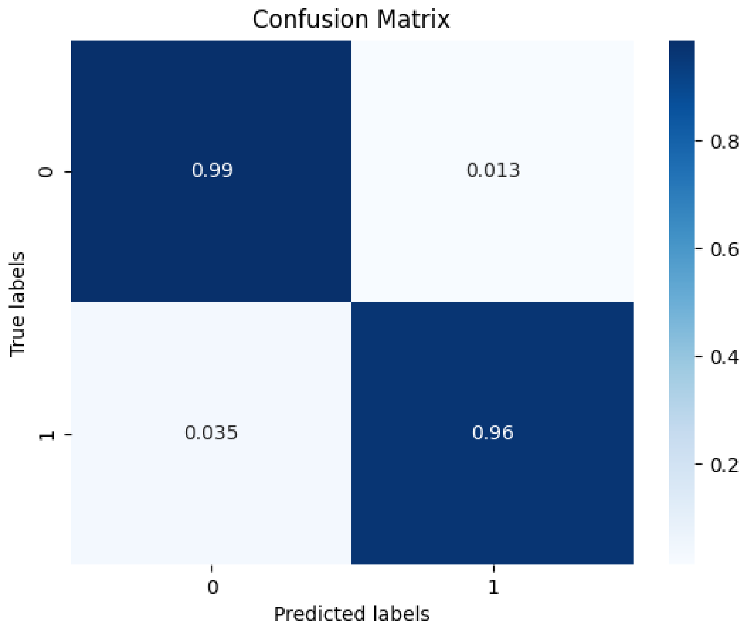



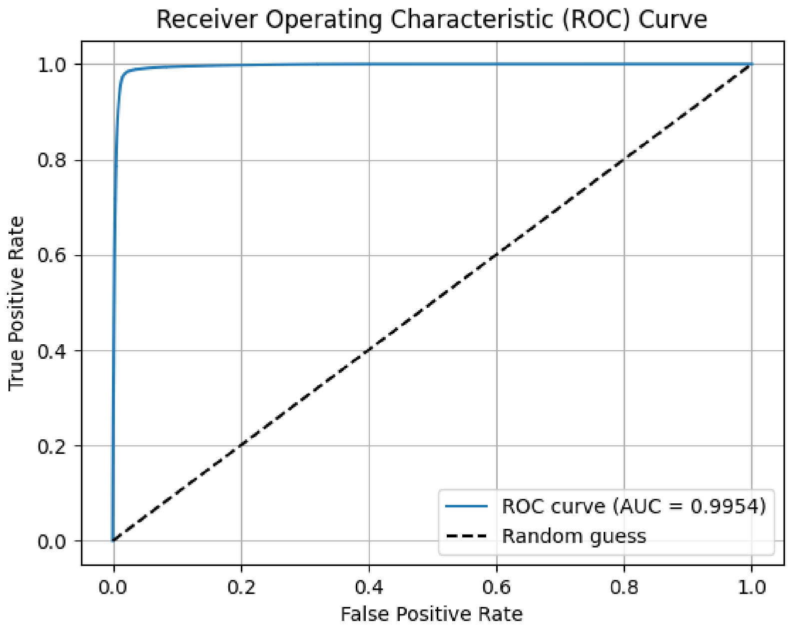



## What happen in a high noise environment?

SageConv algorithm:
- 4 million event,
- 550 epochs,
- lr =0.0001,
- 7 GNN layer
- aggregation function=mean

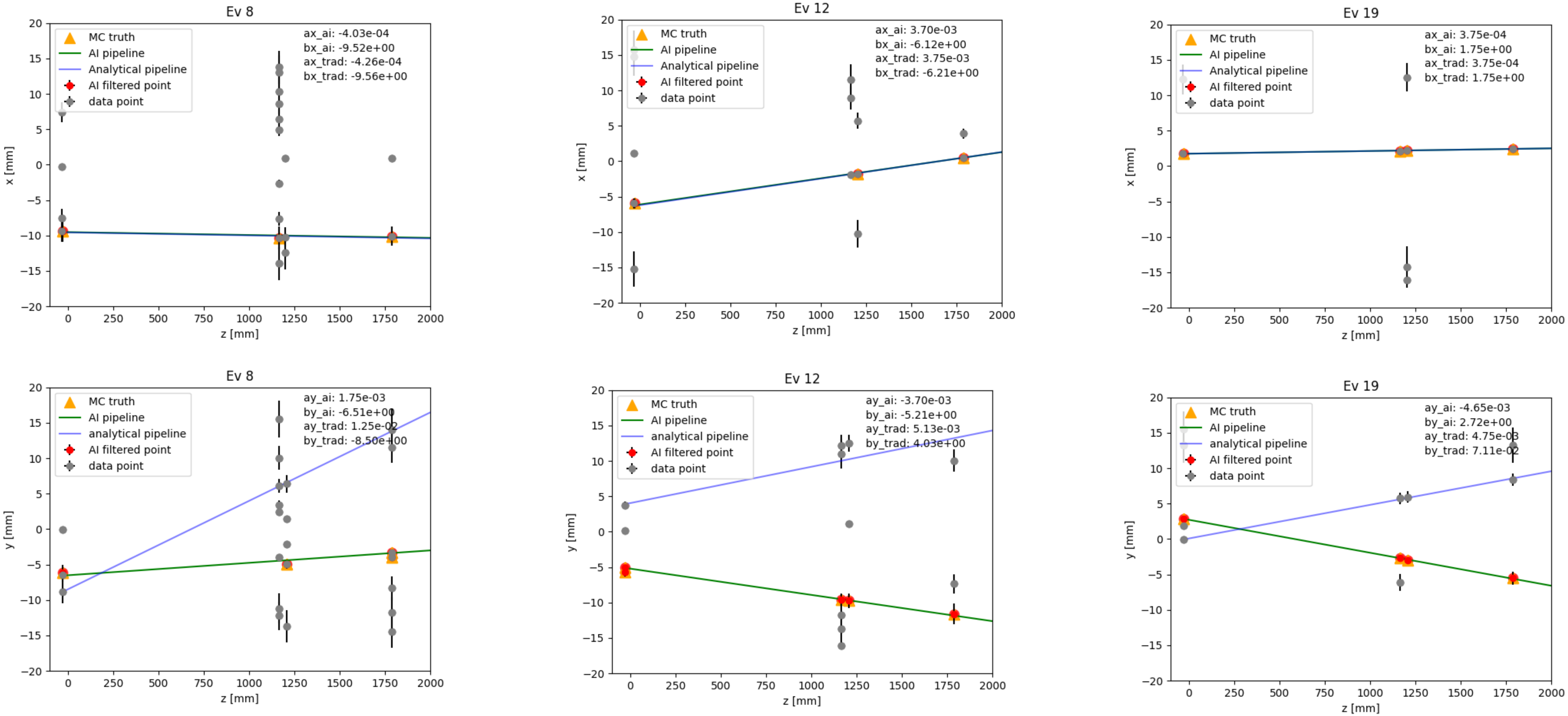
Balanced and normalized dataset



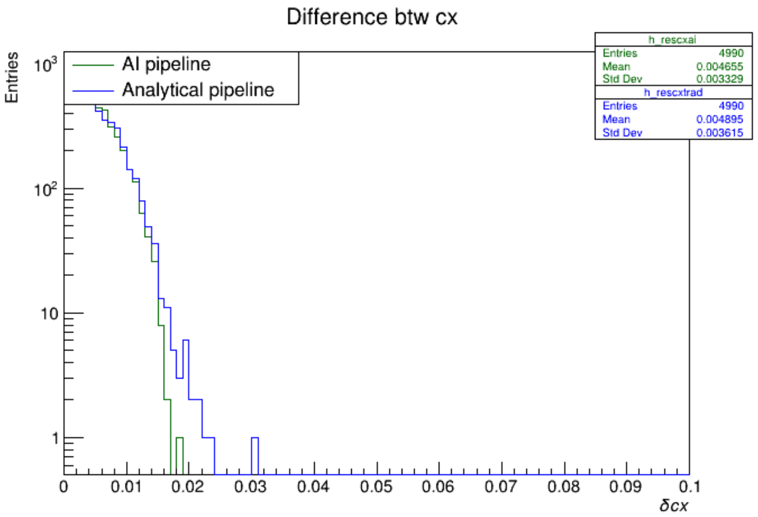



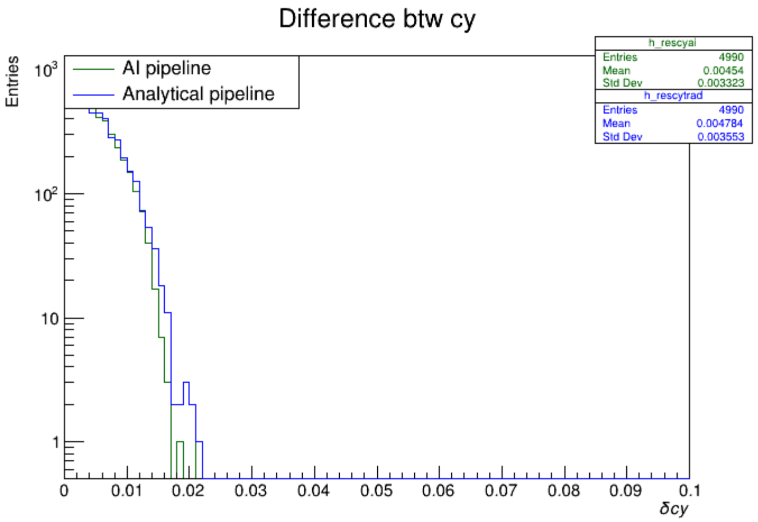

Accuracy: 0,9761
Recall: 0,9649
Precision: 0,9851

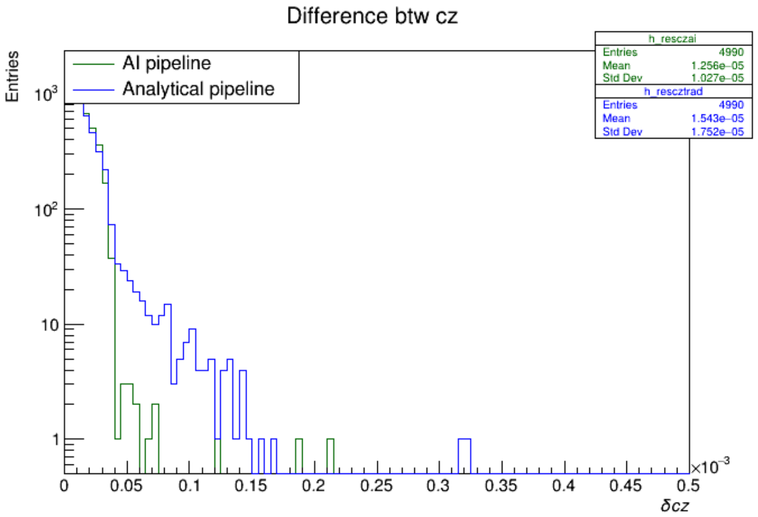
# Event display



## The SageConv algorithm: comparison with traditional pipeline




To process 5000 tracks the analytical pipeline takes 102 min the AI pipeline takes 112 ms

# Timescale, Milestones and KPIs

| Timescale | Milestones | KPI |
|---|---|---|
| May-September | Optimization of the GNN algorithms | Link github |
| September –December | Preliminary analysis and gnn development on more complex tracking data | Link github |
| January-? | Development of a preliminary unified AI architecture for tracker and calorimeter (see M.Bossa talk) in space experiment | Link github |

# Next Steps and Expected Results

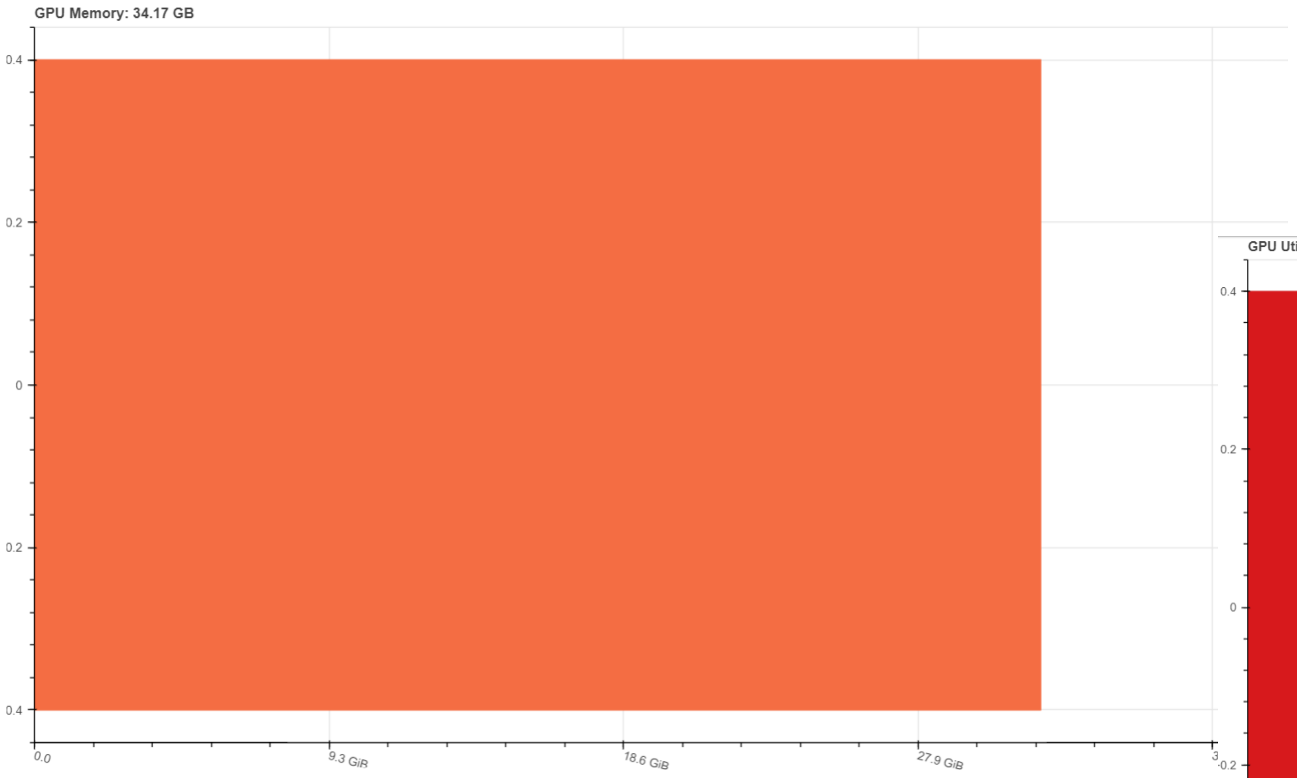
The GNN shows promising performances on tracking purposes over the analytical/traditional approach. All the software development has been saved on https://github.com/federicacuna/TB_Sept_2023_ml, https://github.com/federicacuna/TestBeam_T10_2023 , https://github.com/federicacuna/nuses_Ml (access on request).

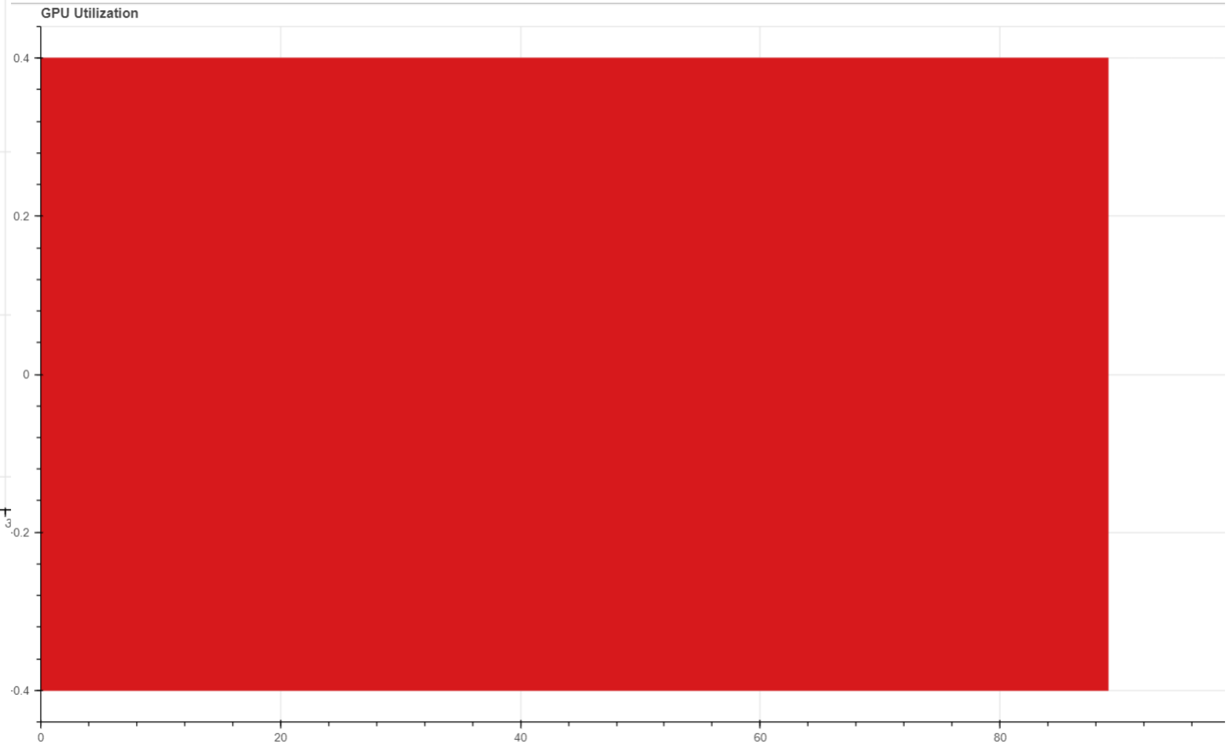
There is room for improvement, by looking on more complex problem with more sophisticated tracker, where we can investigate more elaborated GNN architecture like the interaction networks, which can reconstruct multiple tracks inside the tracker.

Next step:

- Optimization of the GNN algorithm
- Preliminary analysis and GNN development on more complex tracking data
- Development of a preliminary unified AI architecture for tracker and calorimeter (see M.Bossa talk) in space experiment

# Thank you!


GPU Memory: 34.17 GB
0.4
0.2
0
0.2
0.4
0.0
9.3 GiB
18.6 GiB
27.9 GiB


GPU Utilization
0.4
0.2
0
-0.2
-0.4
0
20
40
60
80