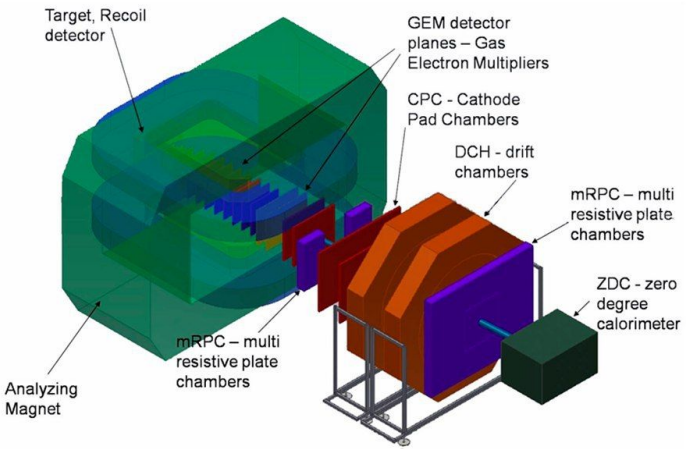
Global strategy of tracking on the basis of Graph Neural Network for BES-III CGEM inner detector

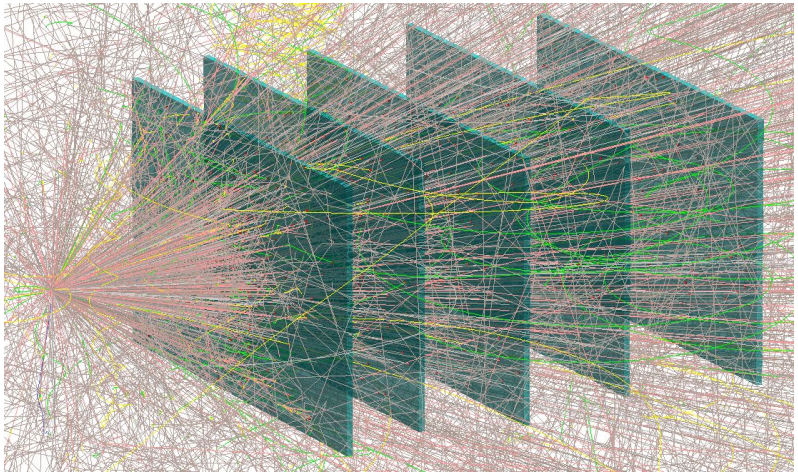
Egor Shchavelev
Saint Petersburg State University

Tracking for BESIII CGEM

- The purpose of this study was to summarize the algorithms and programs developed on the basis of deep learning methods for tracking events obtained in the fixed target BM@N JINR experiment for the track reconstruction with the data of the tracking detector of the BES-III collider experiment in Beijing IHEP. Our study is not intended to interfere in any way with the existing BES-III tracking project, but just shows the prospects for using ML in tracking



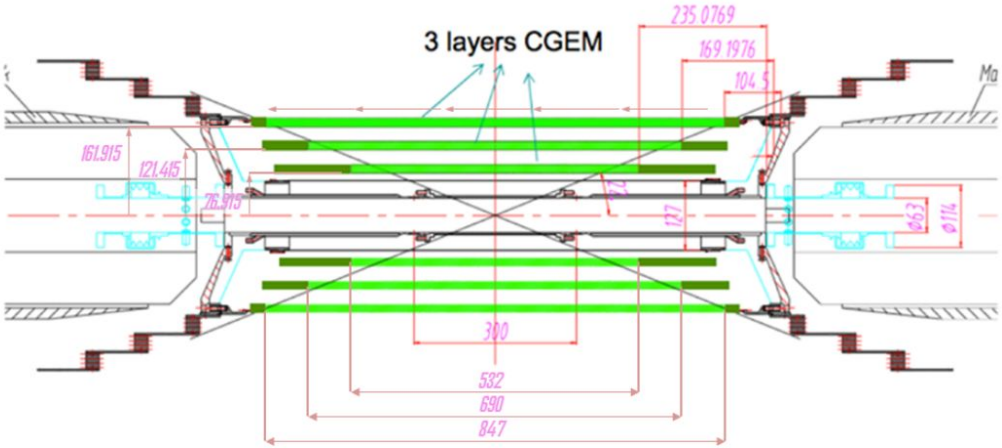
BM@N configuration



Au+Au BM@N MC simulated event

Tracking for BESIII CGEM

- BESIII CGEM is used as a demonstrator for the novel tracking approach
- BESIII CGEM Monte-Carlo simulation was used followed by a simple clustering algorithm to reconstruct CGEM hits

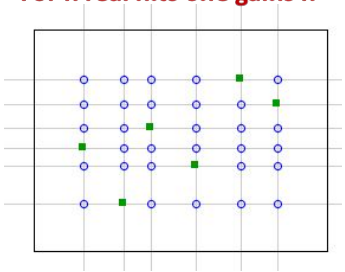


Problems of microstrip gaseous chambers

Problems of CGEM at BES-III and of GEM at BM@N are the same:

The main shortcoming is the appearance of fake hits caused by extra spurious strip crossings

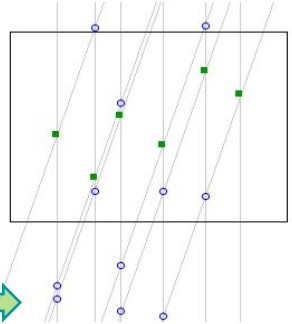
For n real hits one gains n^2- n fakes

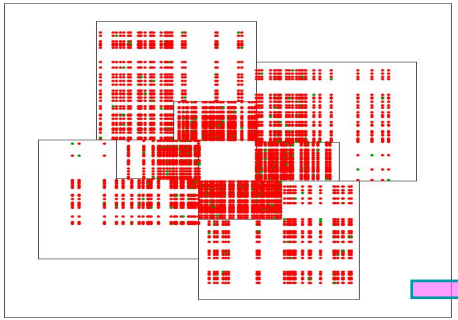


■ - **Real hit (electron avalanche center)**

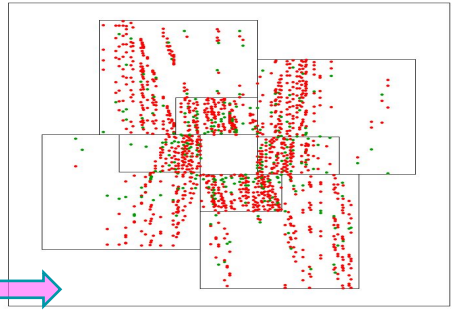
○ - **Spurious crossing**

One of ways to decrease the fake number is to rotate strips of one layer on a **small angle** (5-15 degrees) in respect to another layer



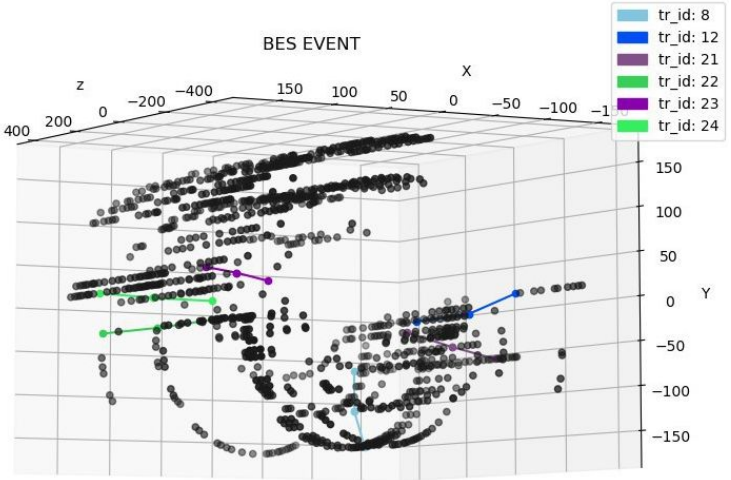


*(BM@N)Angle between strips **90 degrees**, UrQMD event Au-Au, 4 A·GeV*



*(BM@N)Angle between strips **15 degrees**, UrQMD event Au-Au, 4 A·GeV*

● - hit ● - fake



Although small angle between layers removes a lot of fakes, pretty much of them are still left

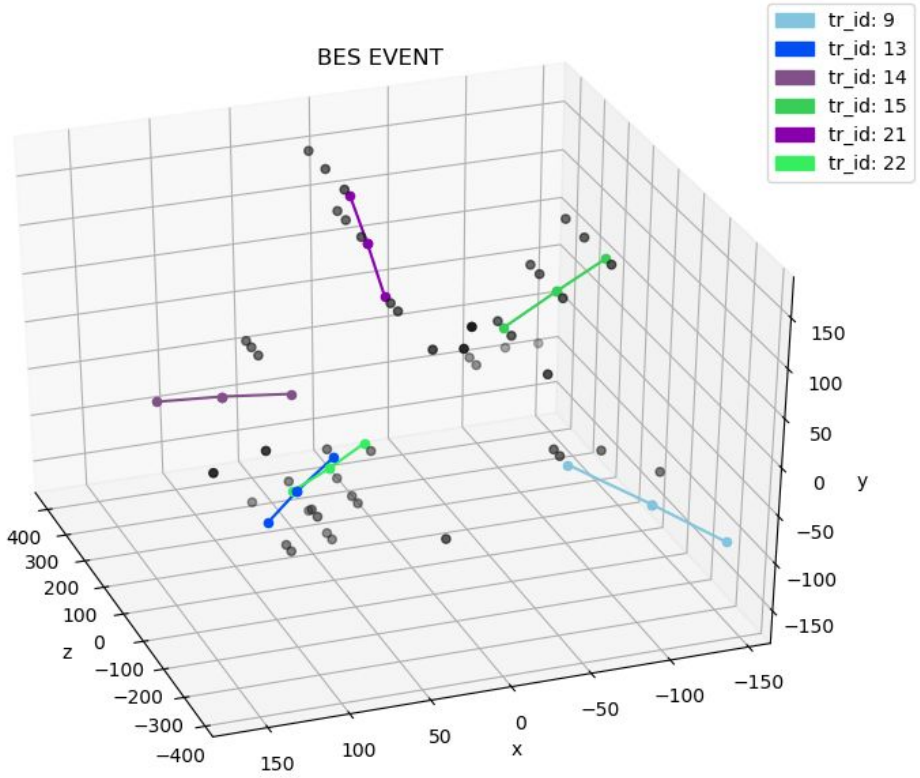
Why neural networks?

- Modern experiments produce vast amount of data in real time
- Kalman filter based tracking approaches can not handle such giant data volumes in a reasonable time

- Modern Neural Network based approaches are coming to rescue -- they can be easily run in parallel on GPU achieving comparable results in terms of purity and efficiency

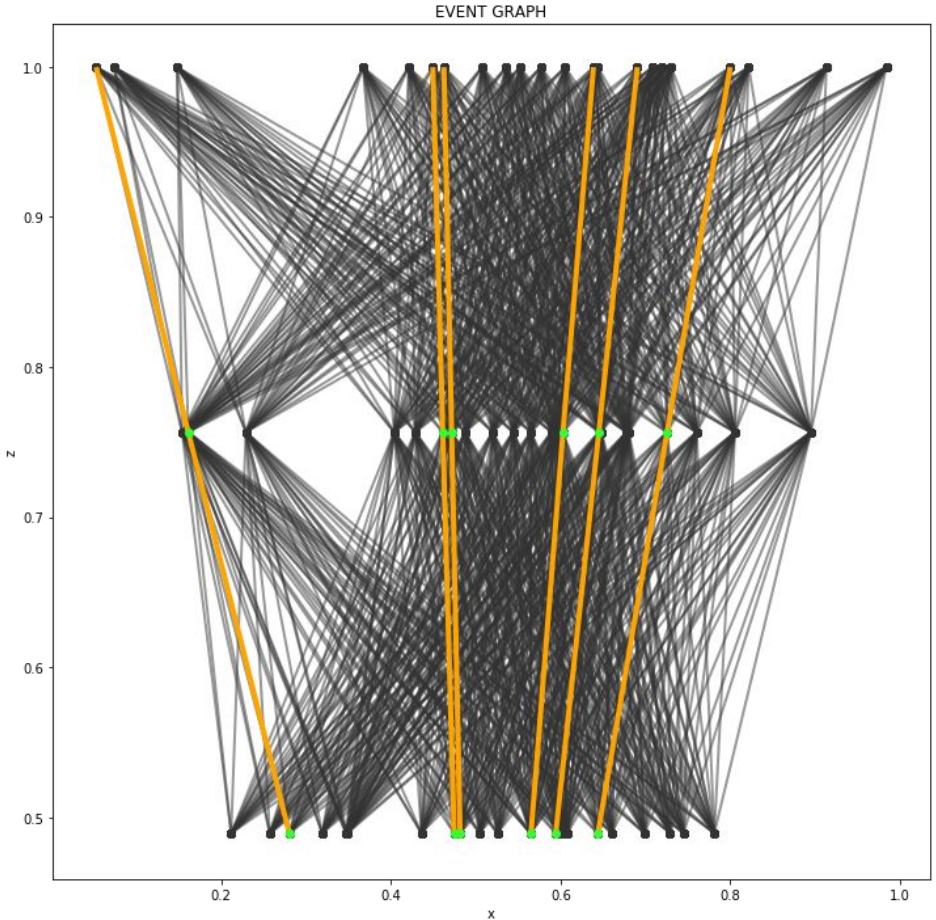
Event as a graph conception

- Consider following BES-III event
- Hits are represented as nodes of the graph;
- Nodes are fully connected between adjacent layers;
- Hits features (X, Y, Z) are being propagated inside the Graph Neural Network (GNN) through their connections to the other hits;



Event as a graph conception

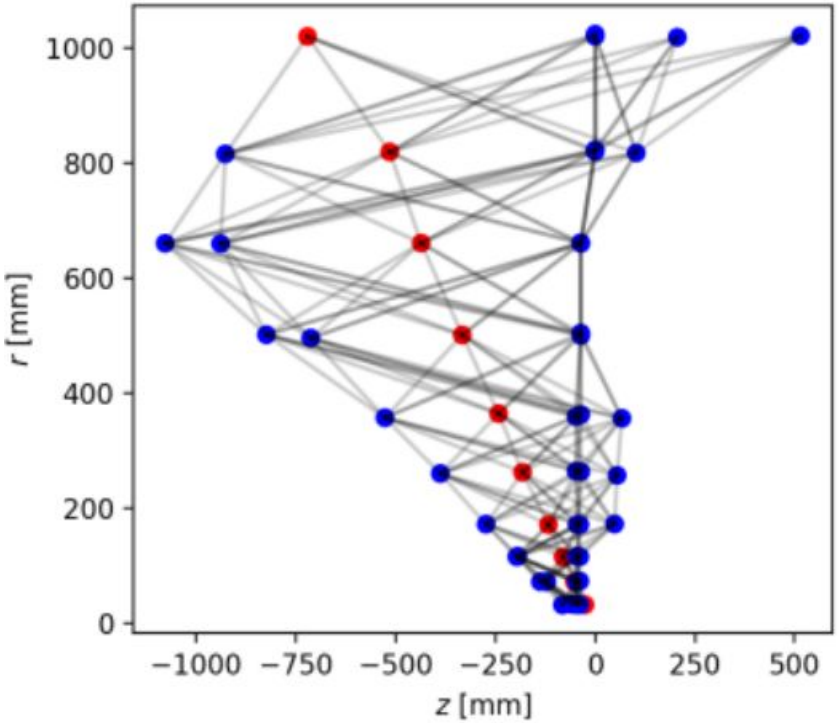
- Hits are represented as nodes of the graph;
- Nodes are fully connected between adjacent layers;
- Hits features (X, Y, Z) are being propagated inside the Graph Neural Network (GNN) through their connections to the other hits;



Graph Neural Network

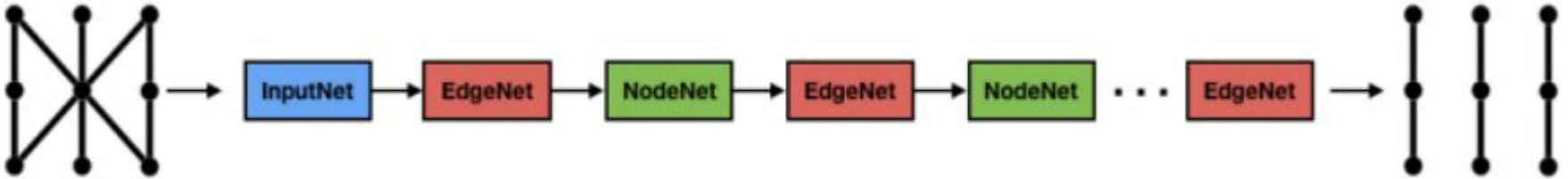
- Graph Neural Network approach for tracking was initially presented by HEP.TrkX project at LHC and based on Interaction Networks approach adapted for particle tracking;

- We introduced derivative graph as a preprocessing step to deal up with noise hits (fakes) produced by CGEM detector
- Note: LHC detectors do not produce any fake hits (they are pixel-based)



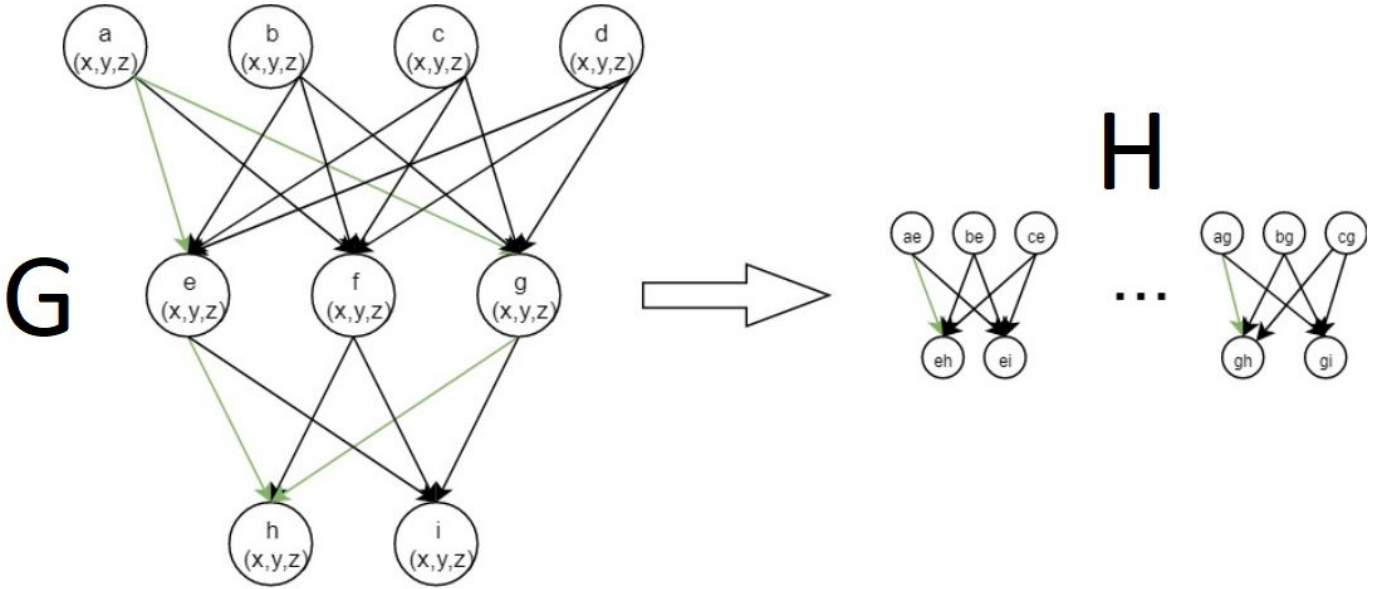
Graph Neural Network

- Graph Neural Network consists of 3 main parts: Input network, Edge network and Node network
- Edge network is the MLP with 2 layers. For each edge, it selects the associated nodes' features, then applies network layers with sigmoid activation.
- Node network is the MLP with 2 layers and Tanh activations. It computes new node features on the graph. For each node, it aggregates the neighbor node features (separately on the input and output side), and combines them with the node's previous features in a fully-connected network to compute the new features.
- Node and Edge network can be stacked and iterated over



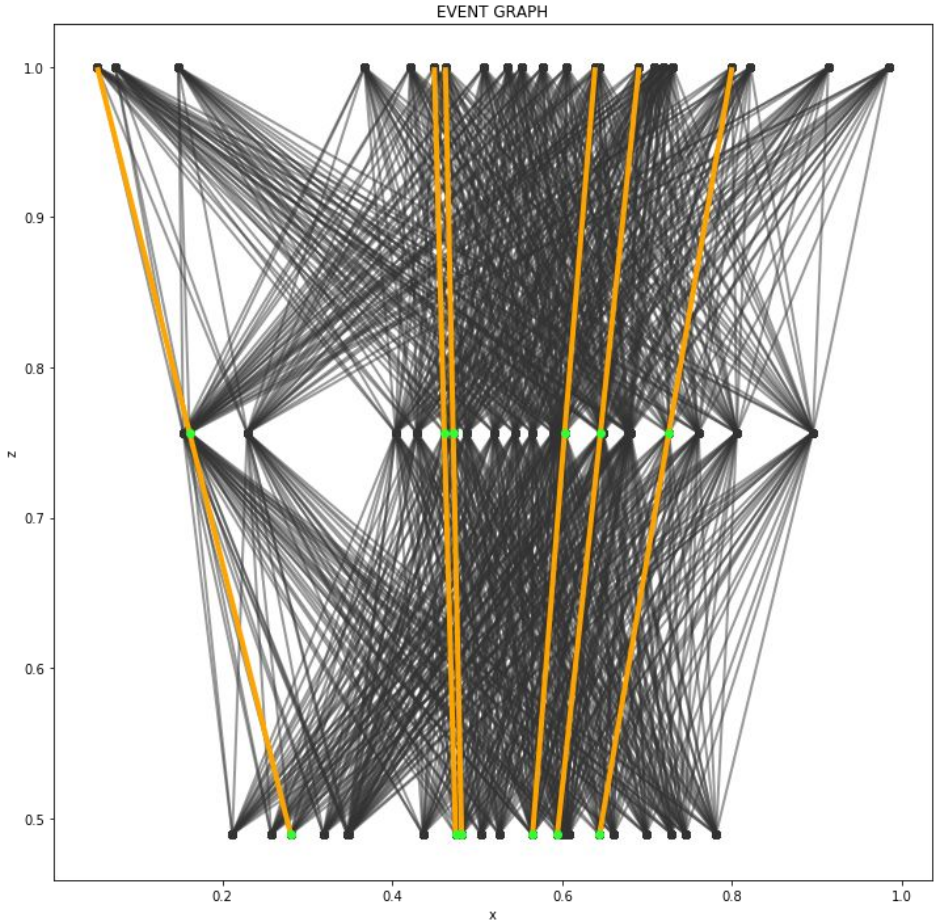
A derivative of the ordered graph

- We take graph G and compute the ‘derivative’ graph H such as:
  - Interpret nodes of graph G as edges of the graph H
  - Edges of the graph G become as nodes of the graph H
  - One edge in the derivative graph is an ordered path of length 2 of the G graph
  - In fact, the edge of the derivative graph is a whole track of the event



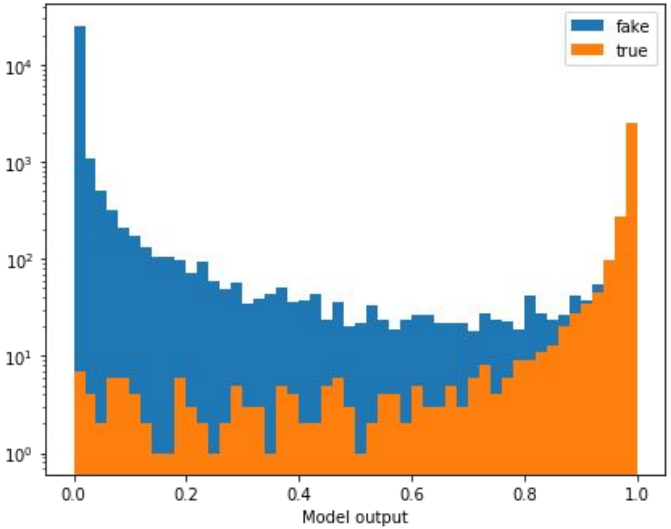
Graph Neural Network

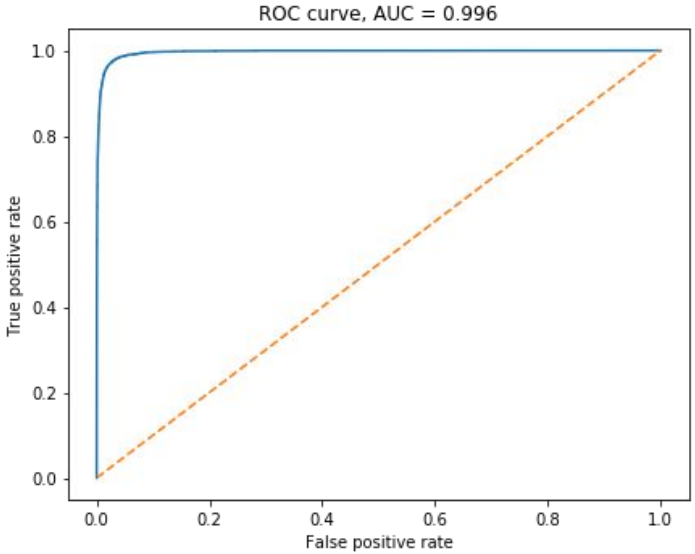
- The goal of the Graph Neural Network to recover the true segments of the graph G (marked orange)
- That is, GNN is labeling every segment with the weight W in [0,1] interval
- We consider a segment as a true segment if W > 0.5



Results(efficiency)

- Recall (percent of the found real segments)
  - 98.0385%
- Precision (percent of the found true segments interpreted as true segments)
  - 86.5048%





Results (speed)

- Constructing derivative of the graph (preprocessing step):
  - ~20 events per second (single threaded, CPU, raw python 3.7, Intel 7700K)
  - ~580 events per second (multi threaded, CPU, C++, Intel Xeon E3-1230)
- Inference of the GNN:
  - ~300 events per second (single GPU Nvidia 1080 Ti)

- A preprocessing step could be greatly faster if rewritten in C++ (x30 speed-up!)
- GNN inference could be easily run in a multi-GPU environment

Conclusion

- Although the GNN approach demonstrate promising results it is a bit overkill for a detector with 3 stations, it can make much more profit on the more complex detectors;
- Current approach demonstrates potential in terms of speed: current implementation written in pure python has an obvious bottleneck in the preprocessing. C++ preprocessing speeds up the tracking process at least by 30 times now;
- GNN model could be optimized for the BES-III detector in terms of GPU memory consumption and GPU speed as well.