Tracking for BM@N GEM detector on the basis of graph neural network

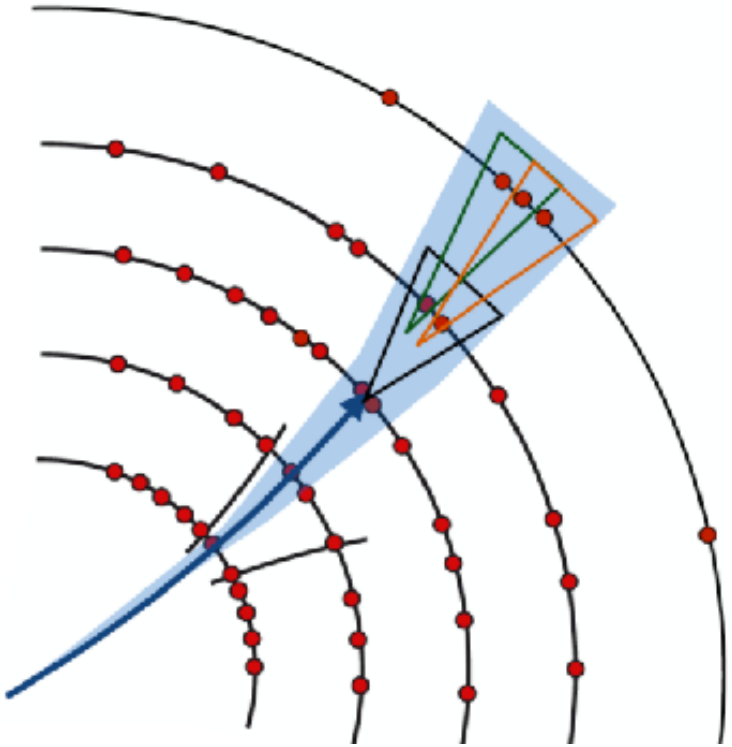
Egor Shchavelev
Pavel Goncharov, Gennady Ososkov, Dmitriy Baranov
Saint Petersburg State University
egor.schavelev@gmail.com

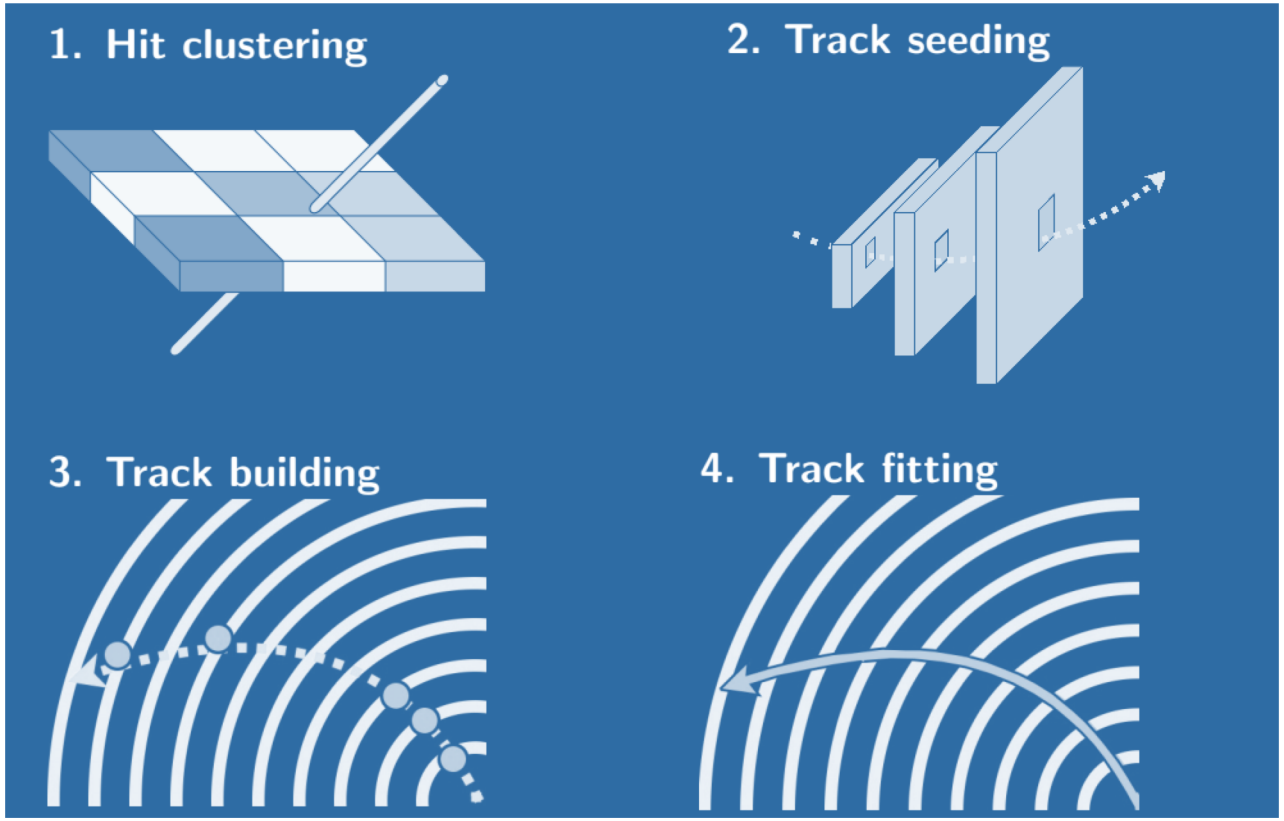
Plan

- What is tracking overall;
- What is BM@N;
- Event as a graph conception;
- Previous work;
- Improved approach;
- Results for the GNN for the GEM-detector.

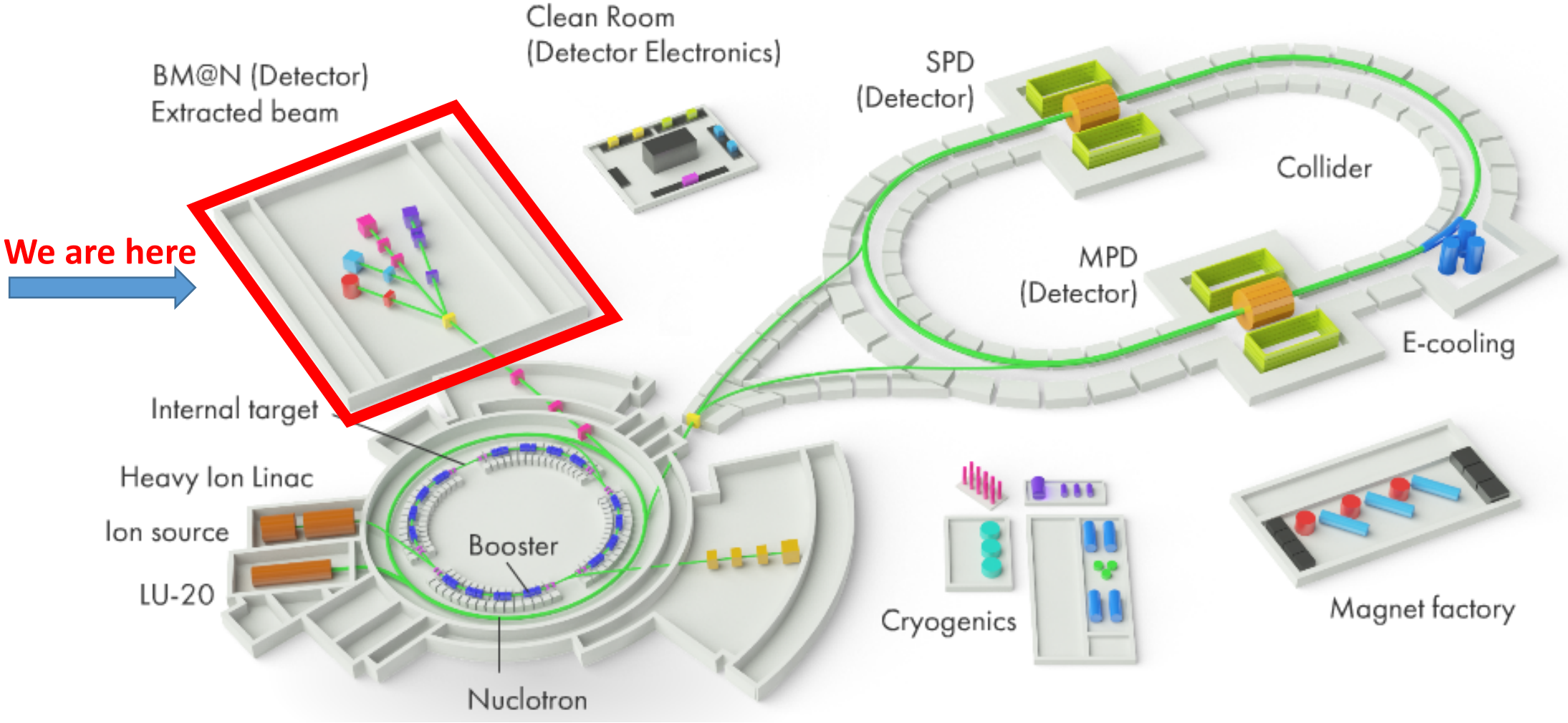
What is tracking?

Tracking or track finding is a process of **reconstruction the particle's trajectories** in high-energy physics detector by connecting the points – hits – that each particle leaves passing through detector's planes. Tracking includes track seeding and track building phases.



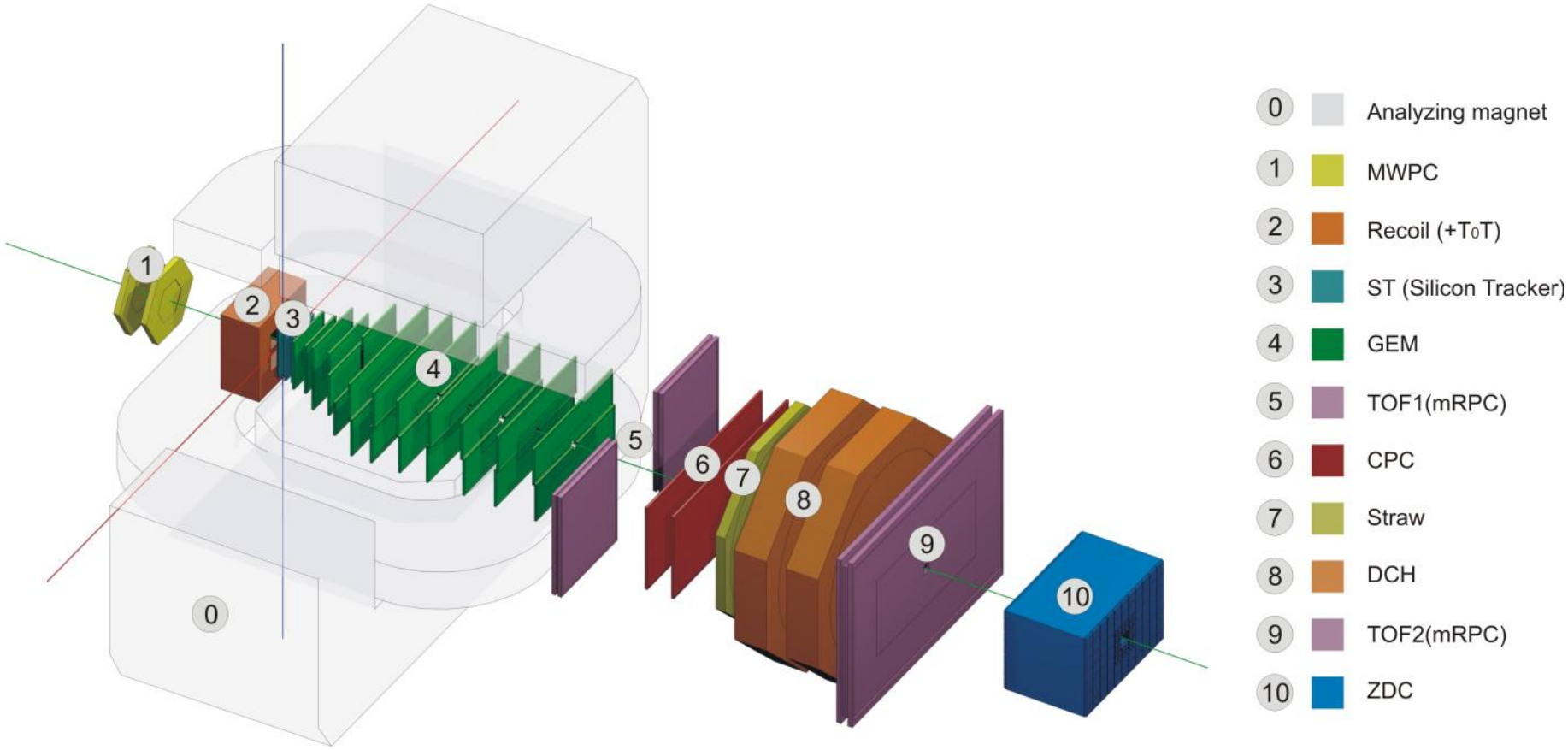


NICA-MPD-SPD-BM@N



General view of the NICA complex with the experiments MPD, SPD, BM@N

Baryonic Matter at Nuclotron (BM@N)

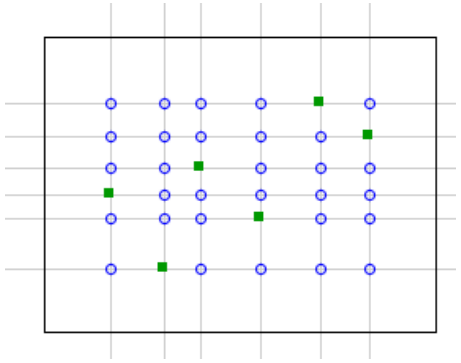


- Our problem is to reconstruct tracks registered by the GEM vertex detector with 6 GEM-stations (RUN 6, spring 2017) inside the magnet.
- All data for further study was simulated in the BmnRoot framework with LAQGSM generator.

Problems of microstrip gaseous chambers

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

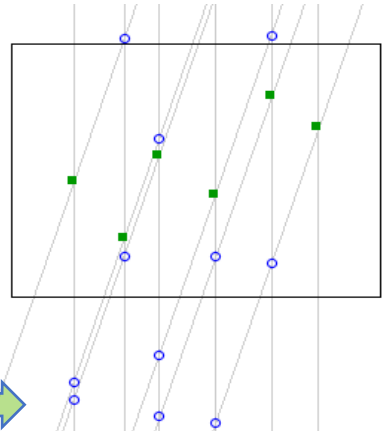
For n real hits one gains n^2 - n fakes

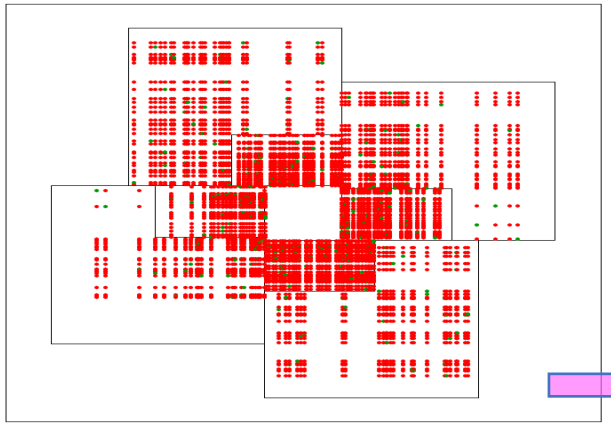


- **Real hit (electron avalanche center)**

- **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

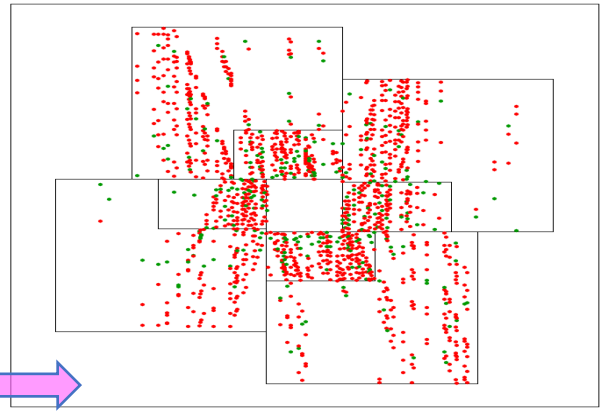




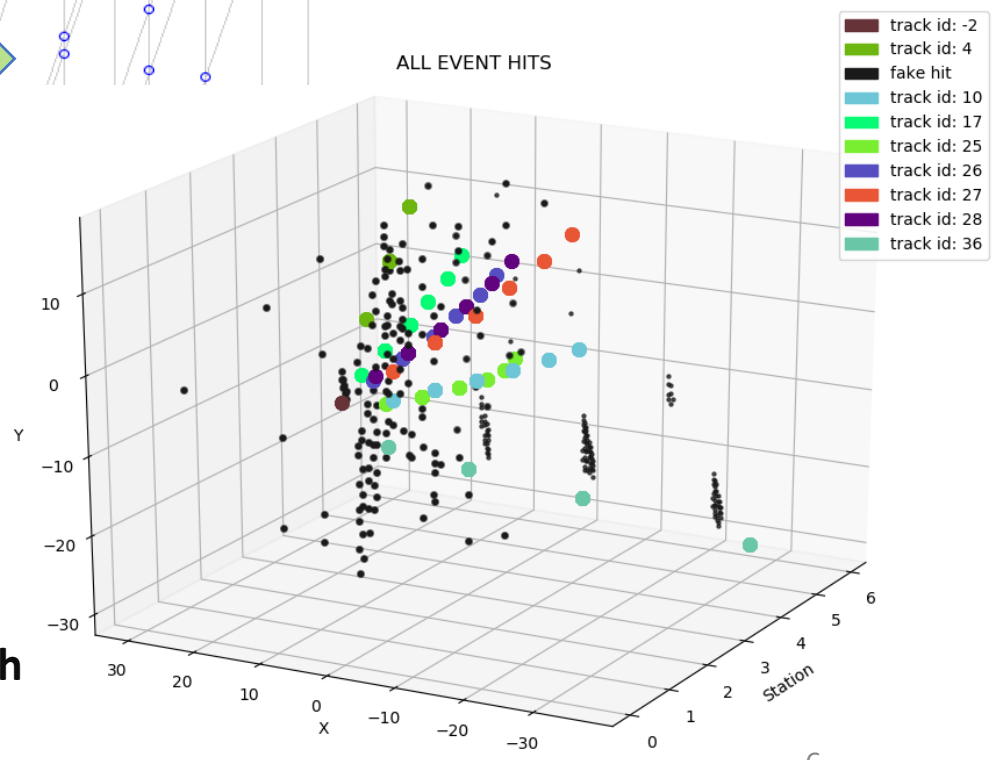
*Angle between strips **90 degrees**, UrQMD event Au-Au, 4 A·GeV*

- hit

- fake



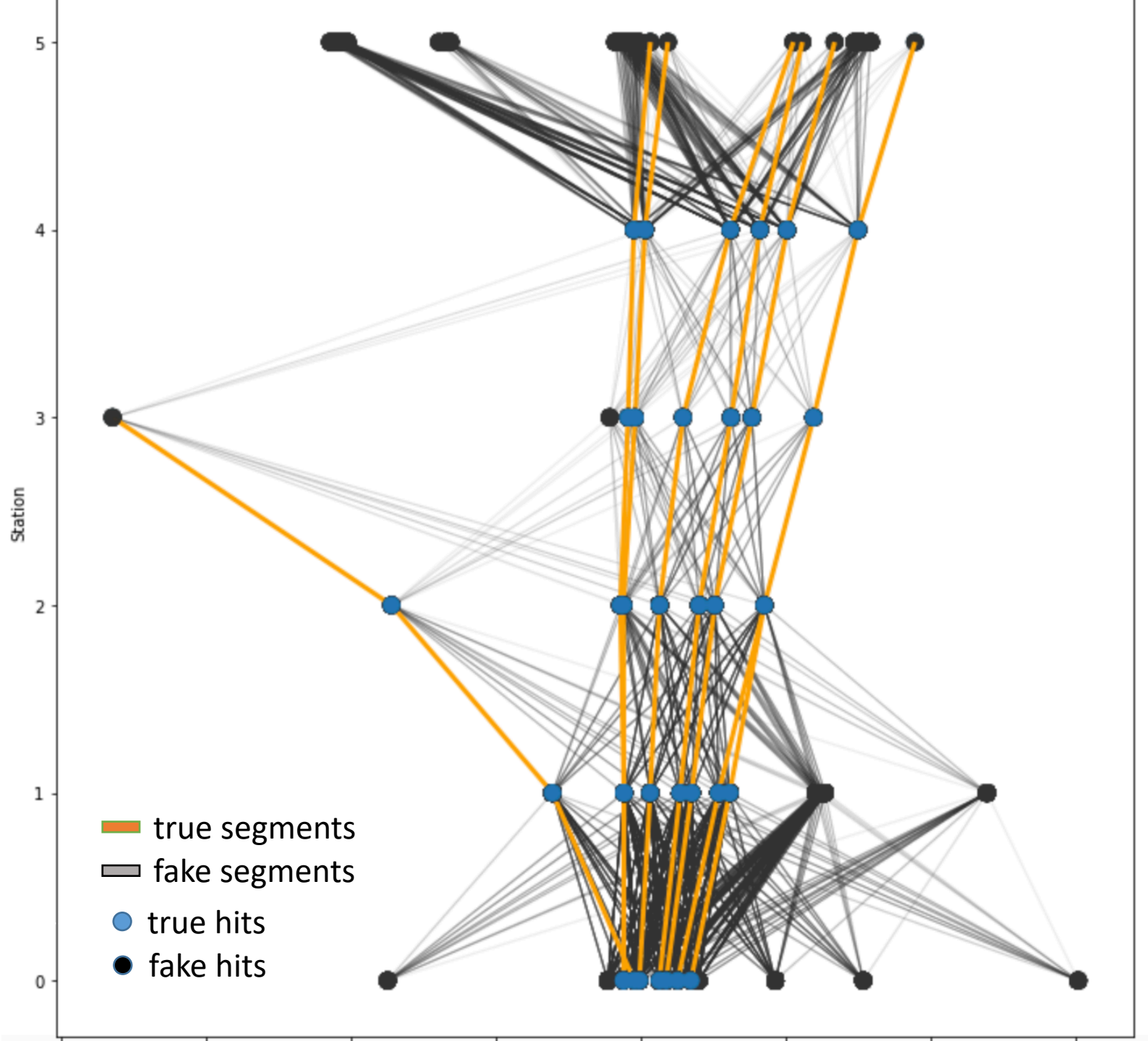
*Angle between strips **15 degrees**, UrQMD event Au-Au, 4 A·GeV*



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

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;

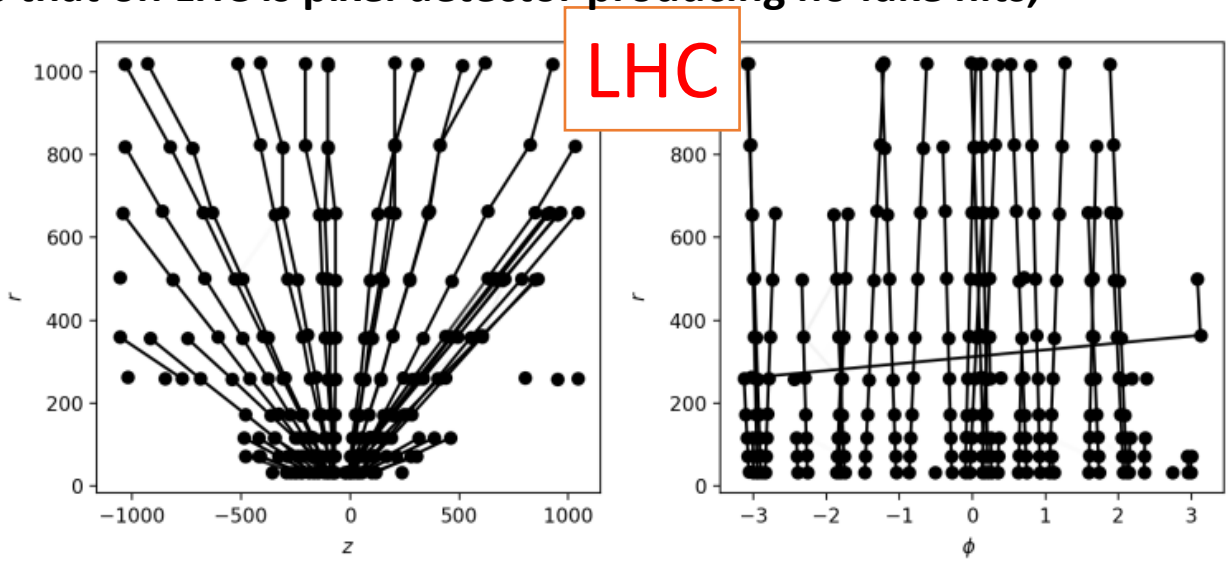


Graph neural network approach at LHC vs GEM

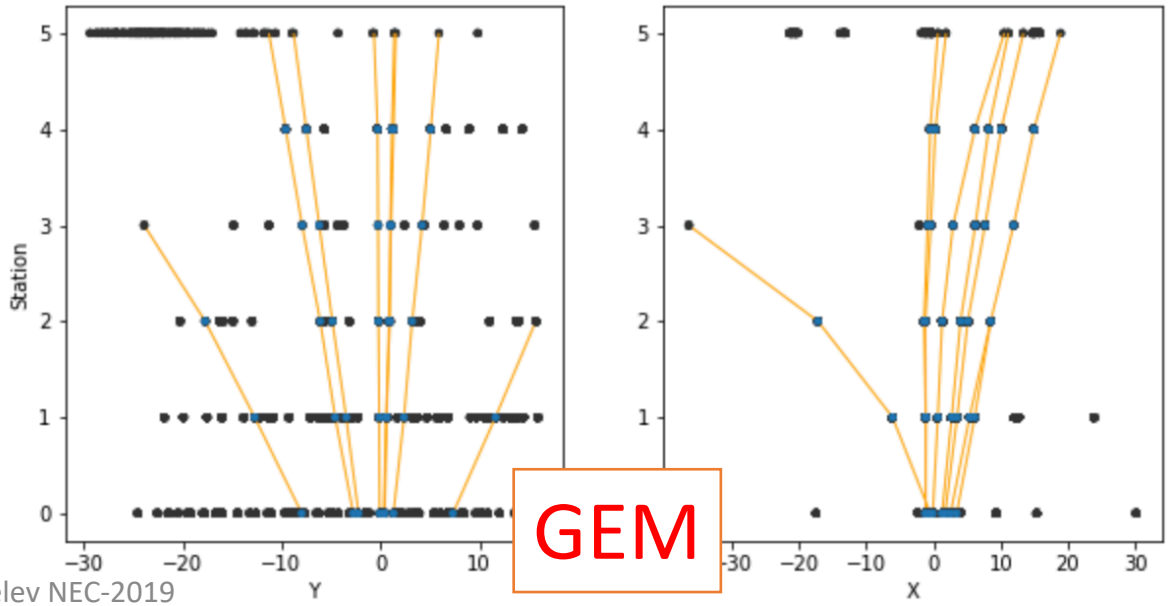
The main difference between the LHC and GEM data is that on LHC is pixel detector producing no fake hits, but in GEM - the majority of hits are fakes

- Model performs binary classification of segments between layers
- ‘1’ is for the true segment and ‘0’ is for the fake segment
- Perfect accuracy and results and achieves 99% accuracy, 98.7% recall (on LHC data)

- BUT their **fake-to-real** ratios (about $\mathbf{O}(n)$) for their pixel detector are far from ($\mathbf{O}(n^2)$) for the GEM detectors

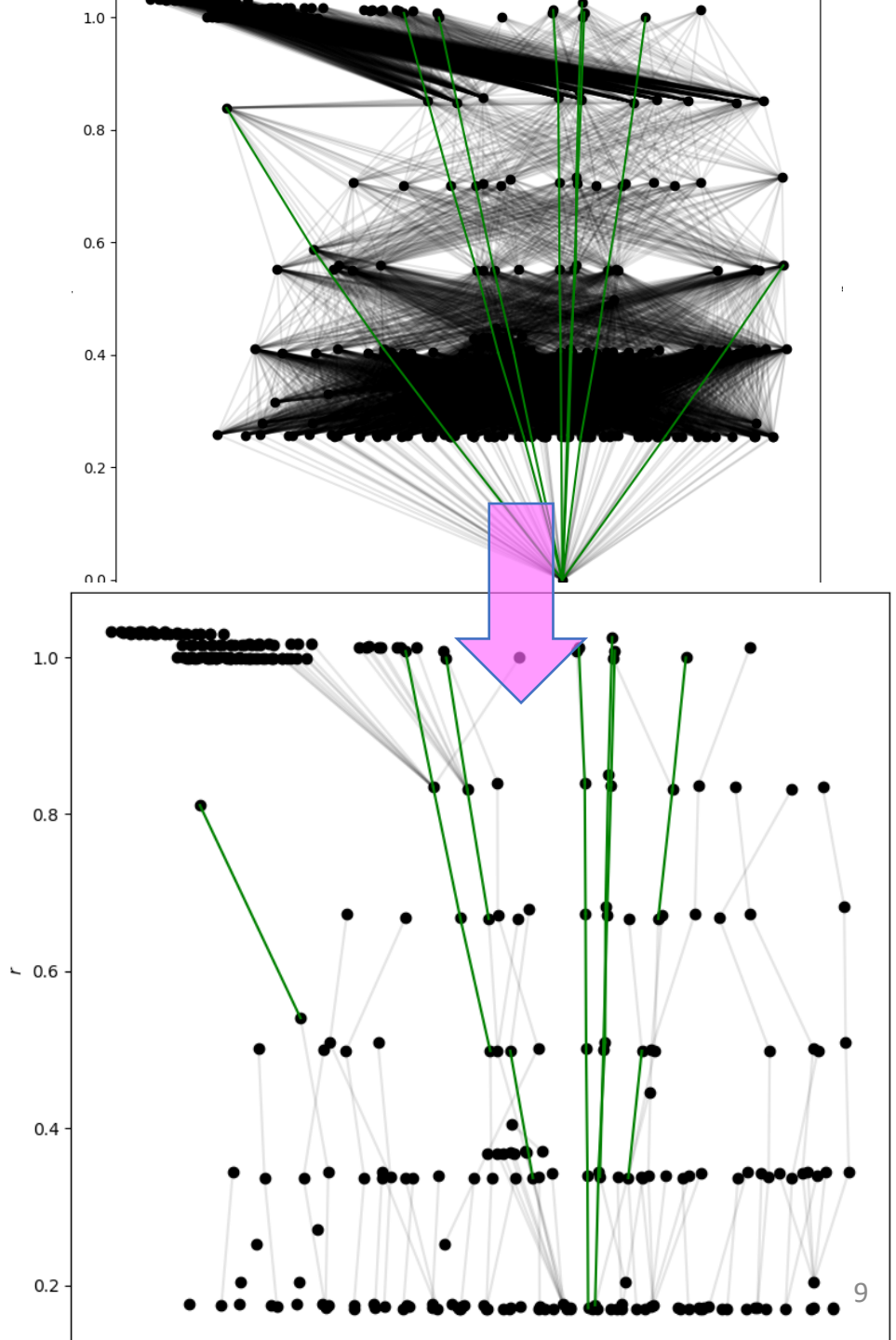


Graphs with the hits and true track trajectories for both detectors



Previous work

- Straightforward adapting the same method is not working at all
- A huge amount of fakes is the root cause for network ambiguity
- We dropped bunch of connections by some basic criteria (like a distance between the hits)
- We introduced the Minimum Branching Tree (MBT) approach and reduced the edge count by the 12 times and successfully trained the GNN on a such data
- But MBT preprocessing loses about 22% of the true segments
- Interesting but too harsh, we can not afford such inaccuracies



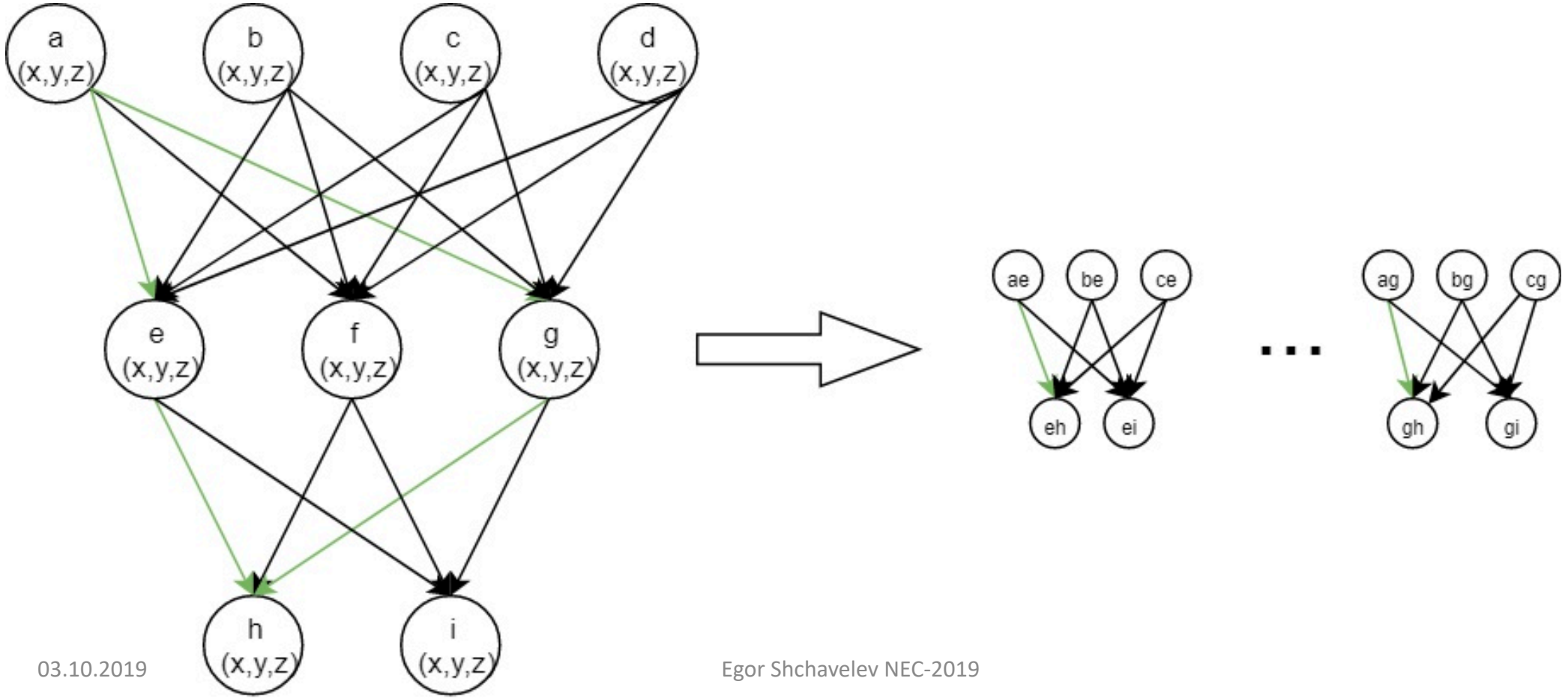
Line graph approach

- Given a graph *G* as the initial graph**.**
- **Line graph $L(G)$** represents adjacencies between edges of the initial graph. The directed version of such transformation is called **line digraph**.
- The key idea is to construct a new graph where:
  - **Edges** of the initial graph turn into the **nodes** of the new graph;
  - **Nodes** of the initial graph turn into the **edges** of the new graph;
  - Each **edge** in the $L(G)$ represents a **length-two directed path** in *G.*

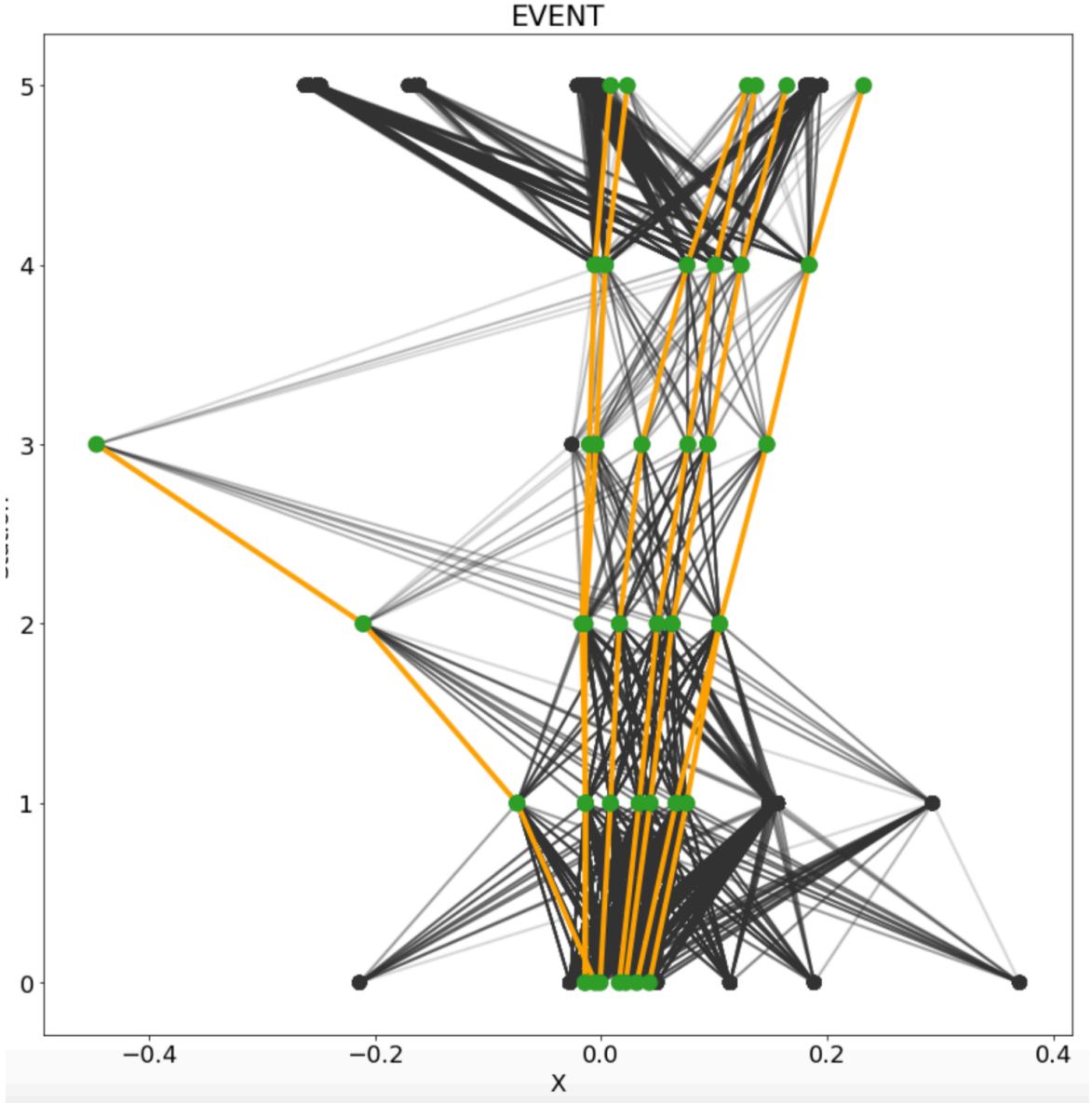
- Such line digraph can be interpreted as a **derivative** of the initial graph.

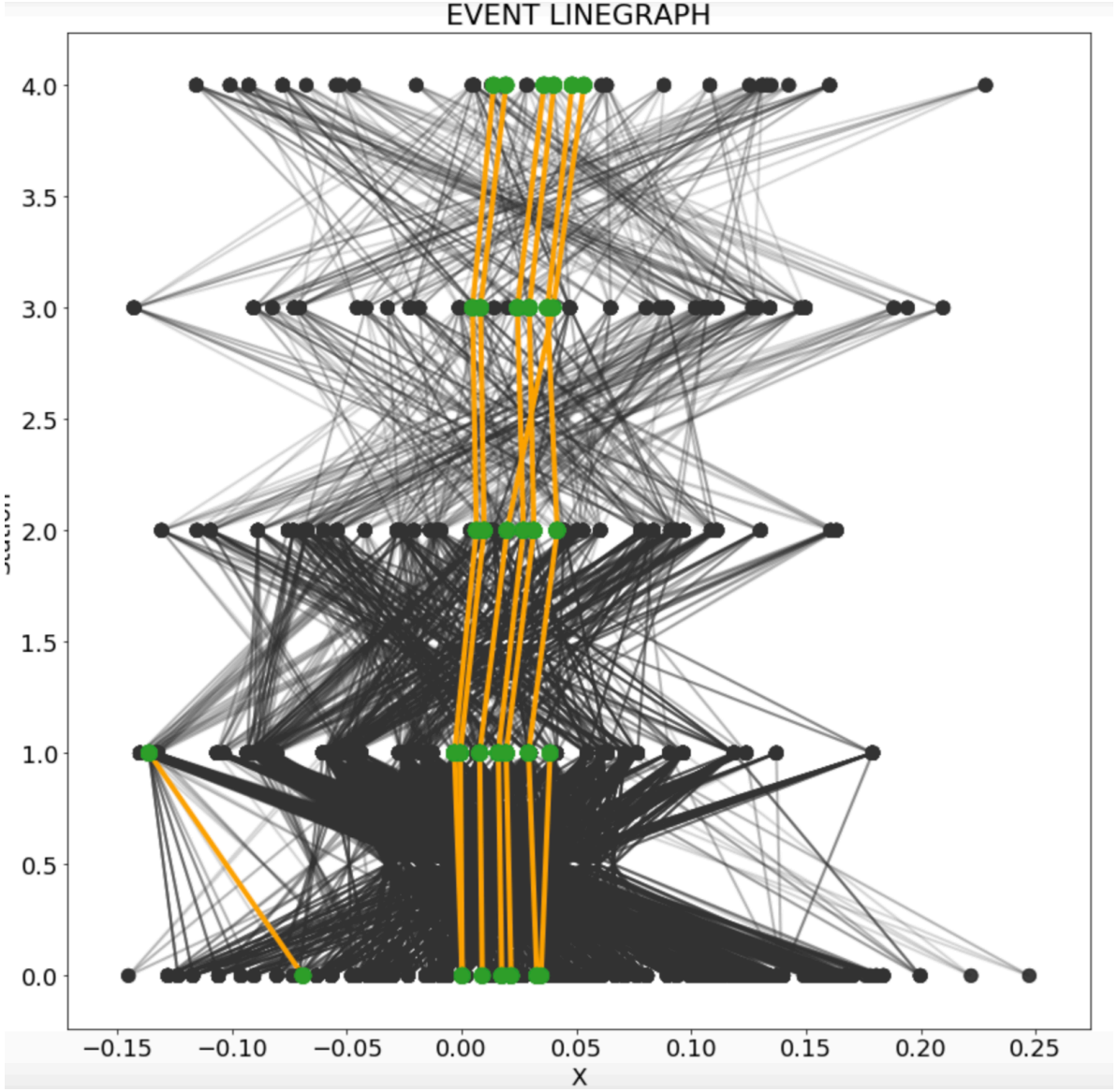
Line graph approach

- Such wonderful approach also leads to the 2 great advantages:
  - Each edge of the $\boldsymbol{L(G)}$ represents the second derivative of the initial path coordinates which helps to drop much more fake segments in a result.
  - The each node of the $\boldsymbol{L(G)}$ carries the 6 coordinates features instead of just 3 from the basic approach.



Line graph of the single event





Preprocessing of the line graph event data

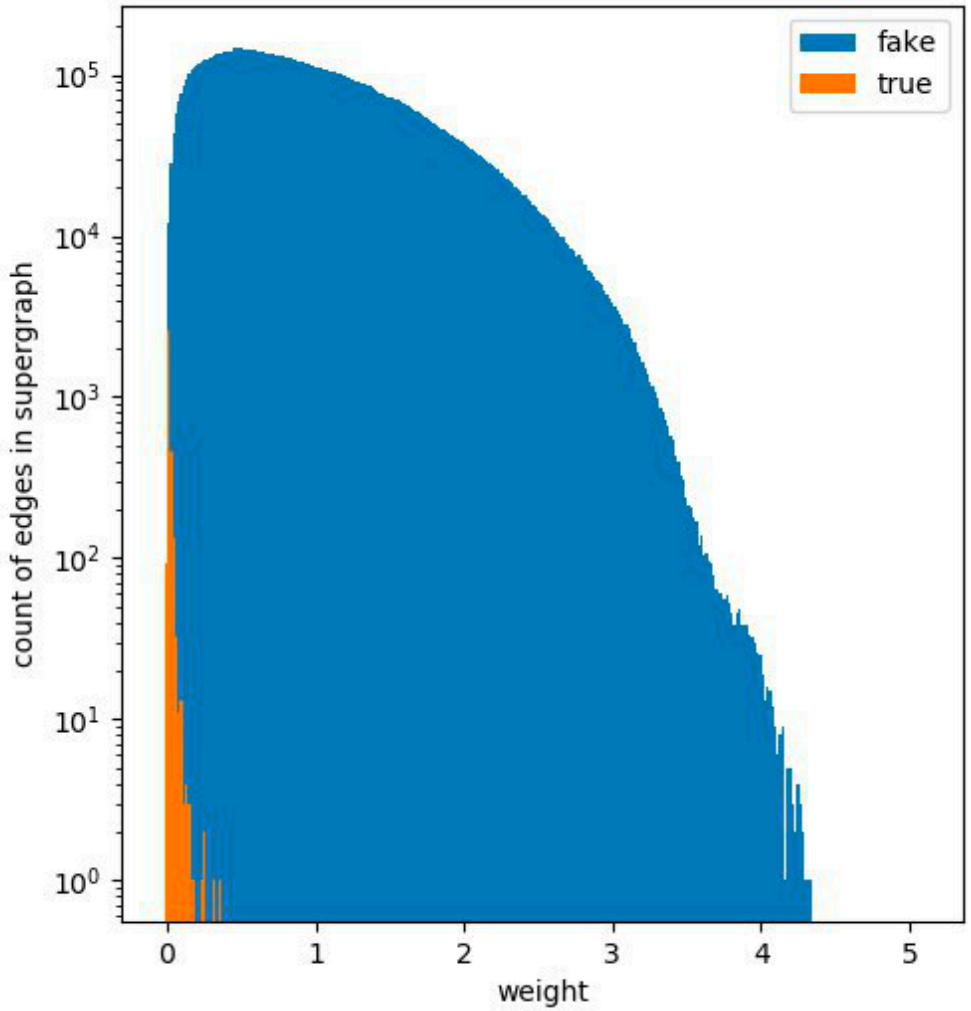
- In the line graph representation we can easily drop some nodes and segments because of their ‘derivativeness’ nature.

- We calculating the edge weight as $w = \|d^2X, d^2Y\|$, where d^2 interpreted as the 2-nd order difference in a single coordinate between **three forward hits**.

- Also we have the first derivative values in the nodes: $d^1X = x_2 - x_1$; $d^1Y = y_2 - y_1$, where d^1 is the difference in a single coordinate of the two forward hits

- After the preprocessing of the 20k events, we are reducing the nodes count by 5 times and edges count more than 4 times achieving the 100% purity of the segments.

Statistics of the line graph edges weights



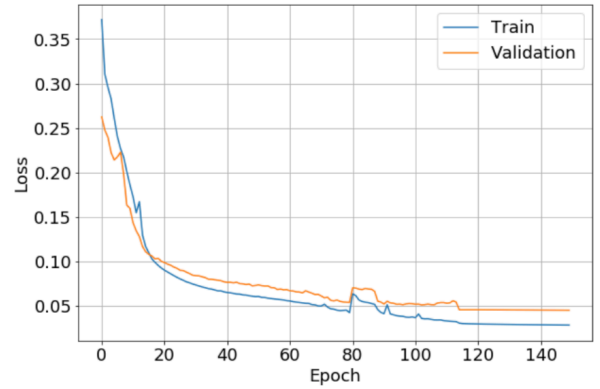
Training setup

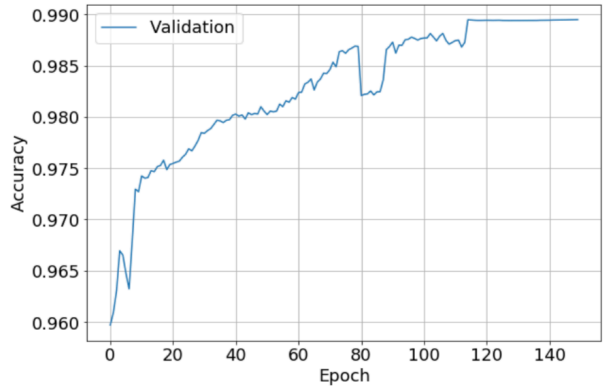
- The GNN gets the feature matrix as an input (line graph nodes with the 5 features: x_1, x_0, y_1, y_0, z; where the index ‘1’ means the end of the edge of the initial graph, and the ‘0’ means the start of the edge, z is the station number), and a connectiveness matrix of the line graph.
- Trained on the 16k events;
- Validated on the 3k events;
- Batch size equals to 1;
- Epoch count was set to 150;
- GNN hidden dim was set to 128;
- Edge-Node network iterations count is equal to 2;
- Training runs about ~10 hours on a single NVIDIA GTX970.
- Training speed is about 60 events/sec.

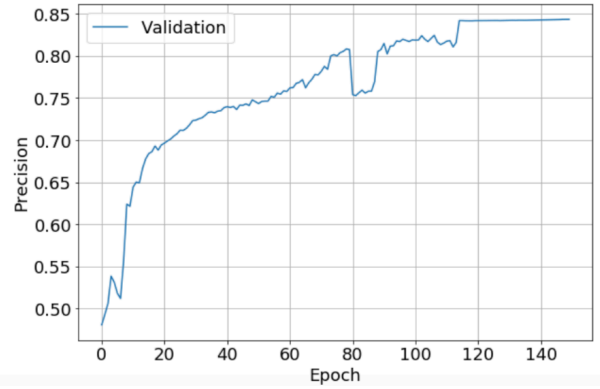
GNN training results

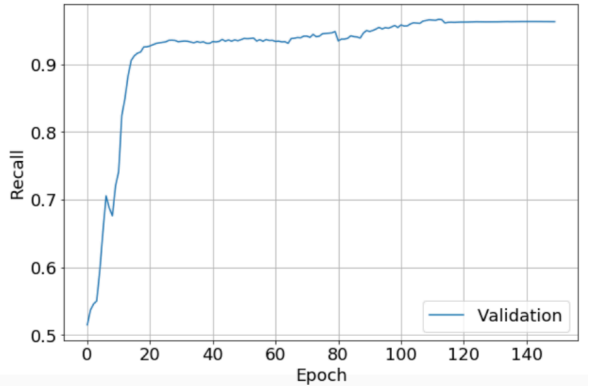
- The model achieves on the line graph data:
  - 99% accuracy
  - 96% recall
  - 85% precision

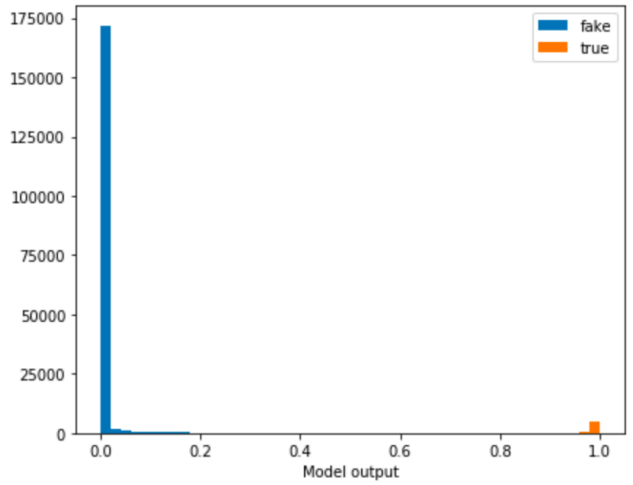
- After the backprocessing to the initial event representation, the model achieves:
  - Hit accuracy: **94%**
    - (amount of correctly predicted tracks' hits)
  - Track accuracy: **87%**
    - (amount of total tracks found without a single gap)

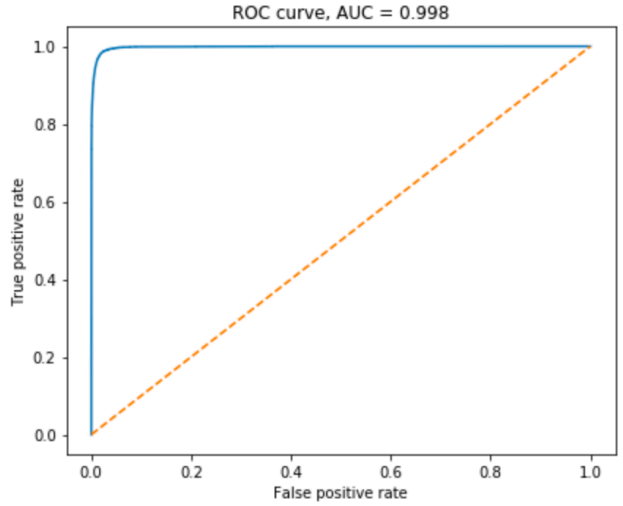




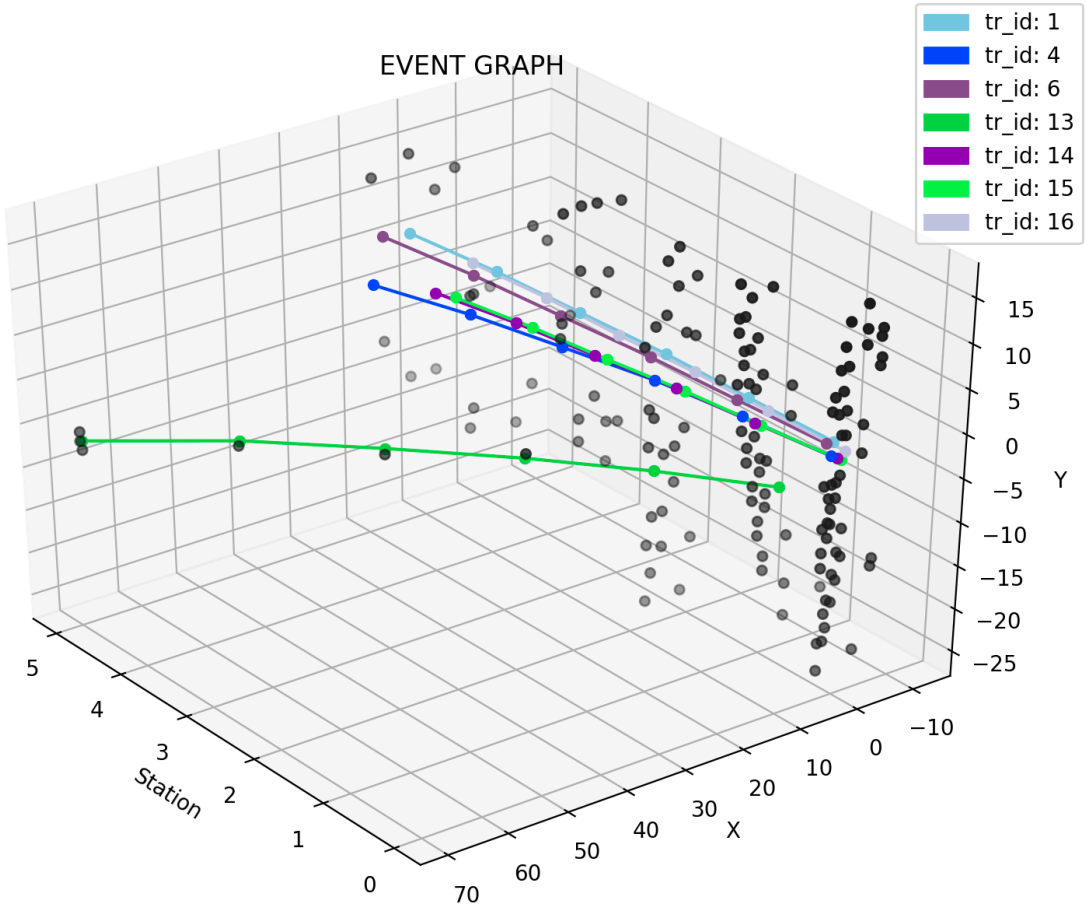


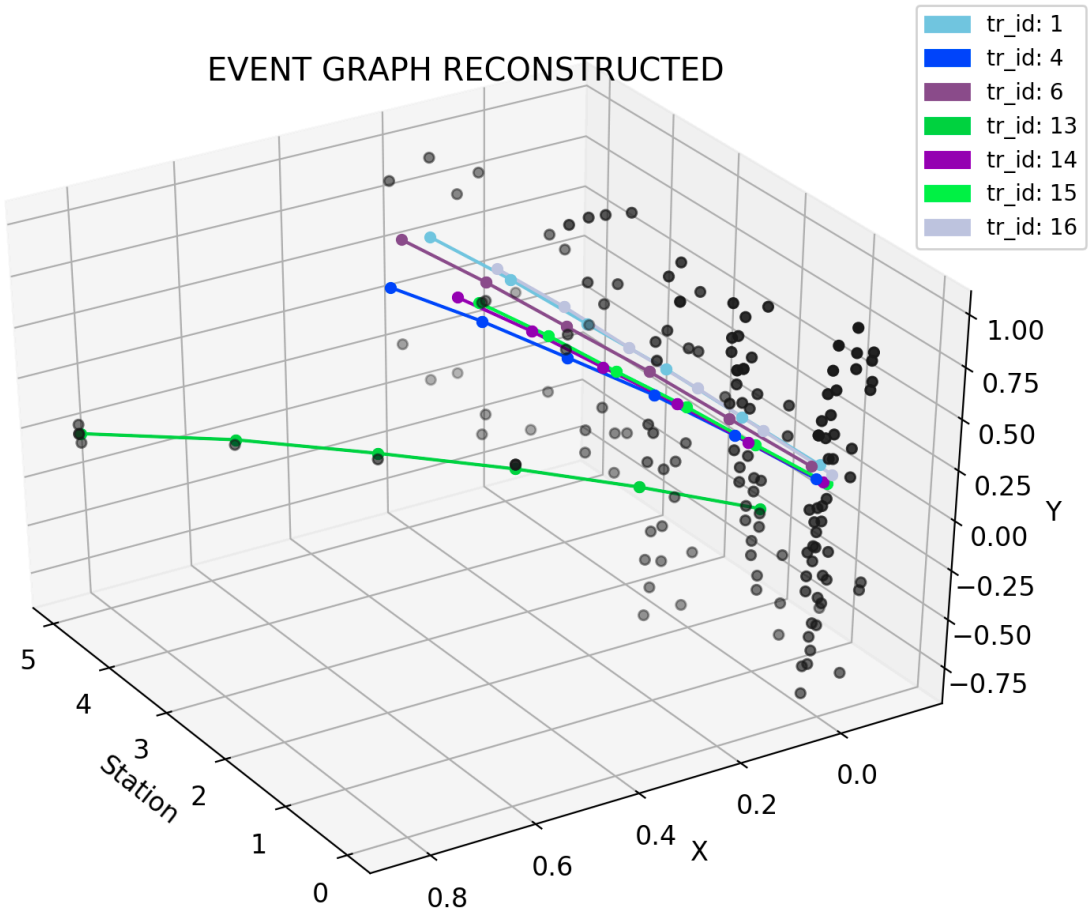






Network output example





Conclusion

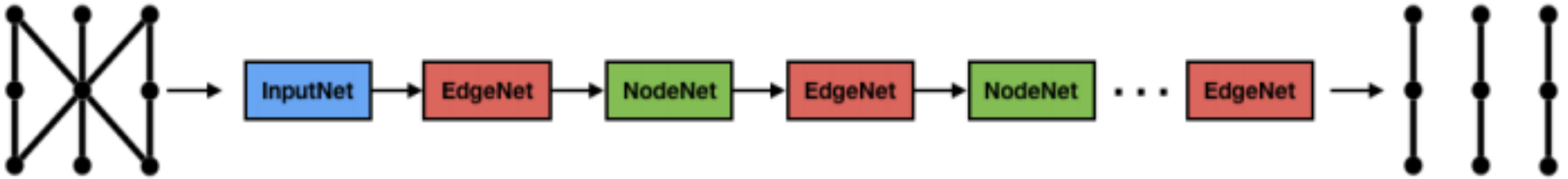
- Previous approach was too inaccurate due to lack of feature information.
- Completely new preprocessing event graph representation proposed.
- Drastically reduced the number of fake nodes and segments without lose of true segments.
- Added features to the GNN.
- GNN recall increased more than 25% comparing with the previous approach (71% -> 96%).
- The new method does not depend on track lengths or fakes ratio.
- But it struggles with the events with the huge number of hits.

Tracking for BM@N GEM detector on the basis of graph neural network

Egor Shchavelev
Pavel Goncharov, Gennady Ososkov, Dmitriy Baranov
Saint Petersburg State University
egor.schavelev@gmail.com

GNN inside. Input network.

- The GNN consists of three main parts – Input Network, Node Network, Edge Network.
- Event-graph is being interpreted as 4 matrices:
  - **X** – matrix of the node features ($N \times M$) where N is the count of nodes and M is the count of feature nodes. In our situation, we use the **hit coordinates as features**, so $\boldsymbol{M = 3}$;
  - **Ri** – matrix of the edges which are ending in the corresponging nodes with the size $N \times E$ (E is the count of edges). In this matrix $\mathrm{Ri}[i, j] = \mathbf{1}$ **if the edge** with the index $\boldsymbol{j}$ **ends on the node** with the index $\boldsymbol{i}$ and 0 otherwise;
  - **Ro** – matrix of the edges which are starting in the concrete nodes. Has the same properties as Ri but it is for output edges;
  - **Y** – neural network labels with the size of ($1 \times E$). $\mathrm{Y}[j] = \mathbf{1}$ if the edge with the index j belongs to the **real track** and **0** otherwise.
- Then, we have the 'Input network'. It is an MLP with 1 layer and Tanh activation function. The X matrix is applying to the Input network. The output of the Input network is moved to the 'Edge-Node' Network iterations.



GNN Inside. Edge network. Node network.

- **Edge network** is the MLP with 2 layers. The activation function between layers is the same Tanh, but the activation of the output layer is the **Sigmoid function** which predicts the probability that concrete **edge is the true edge**.
- Edge network is a network which computes weights for edges of the graph.
  For each edge, **it selects** the associated nodes' **features** (from multiplying input data by Ri and Ro matrices) and 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**.

- This networks can be applied one after another as the iteration cycle. **Count of the iterations** is one of the **hyperparameters** of the full neural network.