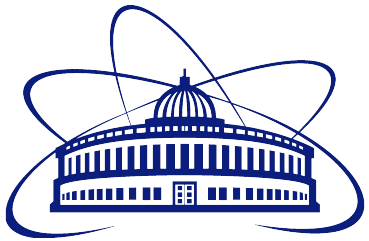
The current progress in the development of a GNN mode for the reconstruction of particle tracks detected at the MPD experiment of the NICA project

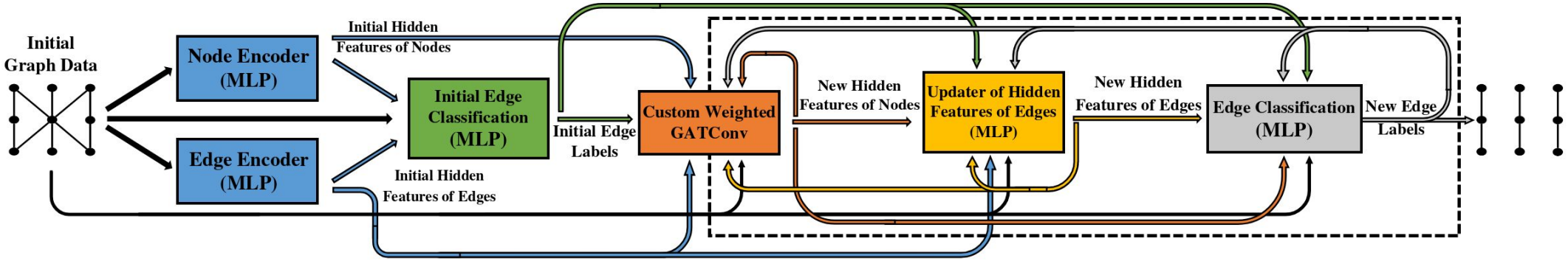
Yauheni Talochka

JINR

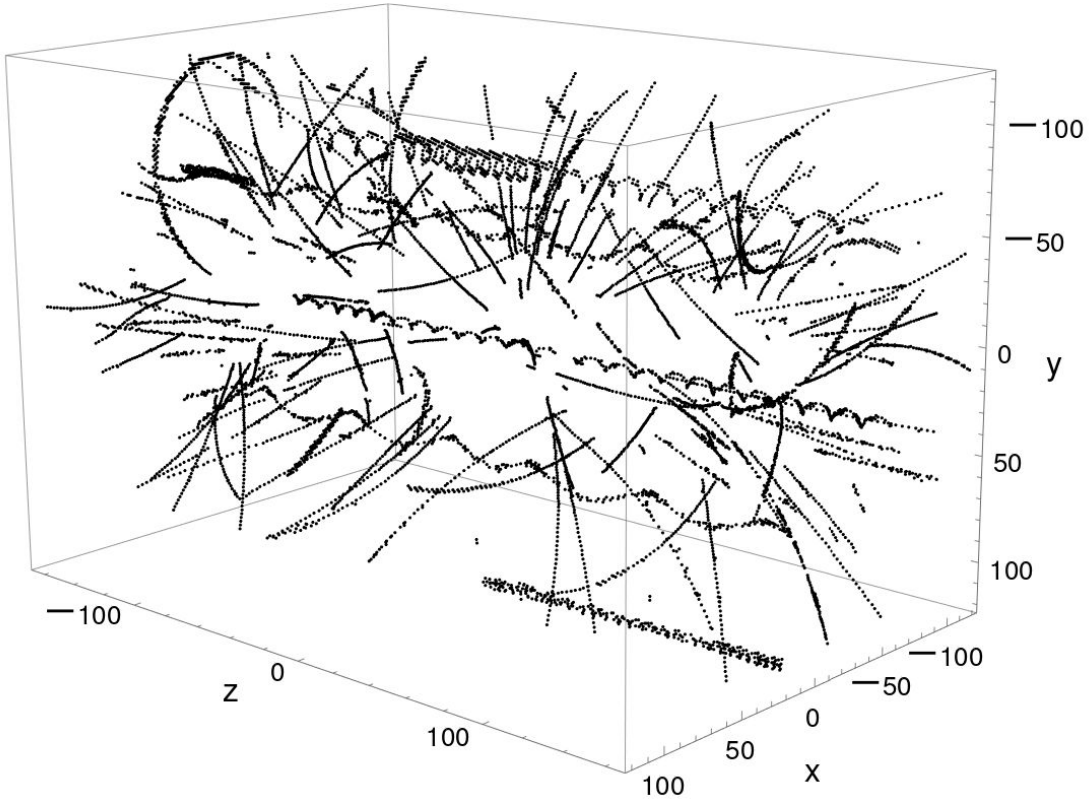
January 22, 2025



GNN model & dataset (100 MPD events)



single MPD event



The detector volume is divided into $n_\varphi \cdot n_\eta$ segments, where $n_\varphi = 4$ and $n_\eta = 2$, η is the pseudorapidity.

The node features are $\{r, \varphi, z\}$,
the edge features are $\{\Delta\theta, \Delta\varphi, \Delta\varrho, \Delta z\}$.

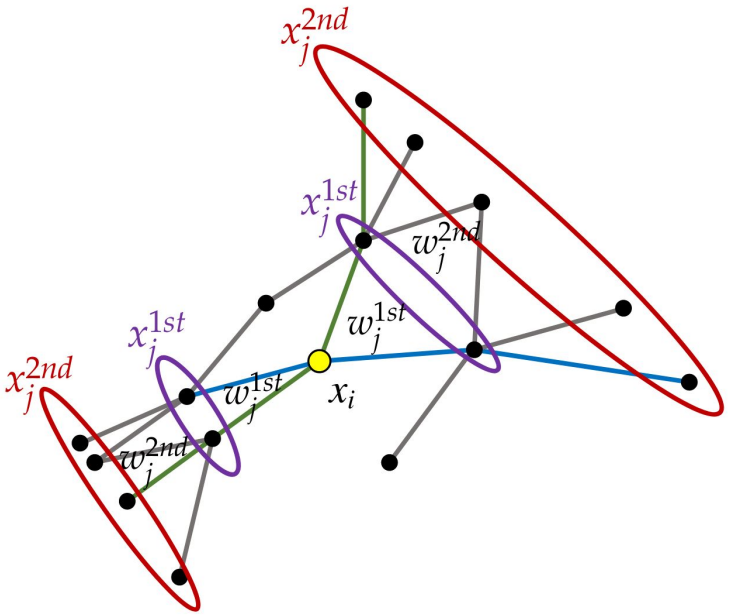
Here, r, φ and z are the cylindrical coordinates of a hit, θ is its polar angle, $\varrho^2 = \varphi^2 + \theta^2$.

Hits related to particles with $p_t < 200$ MeV are considered as noise, otherwise as useful information.

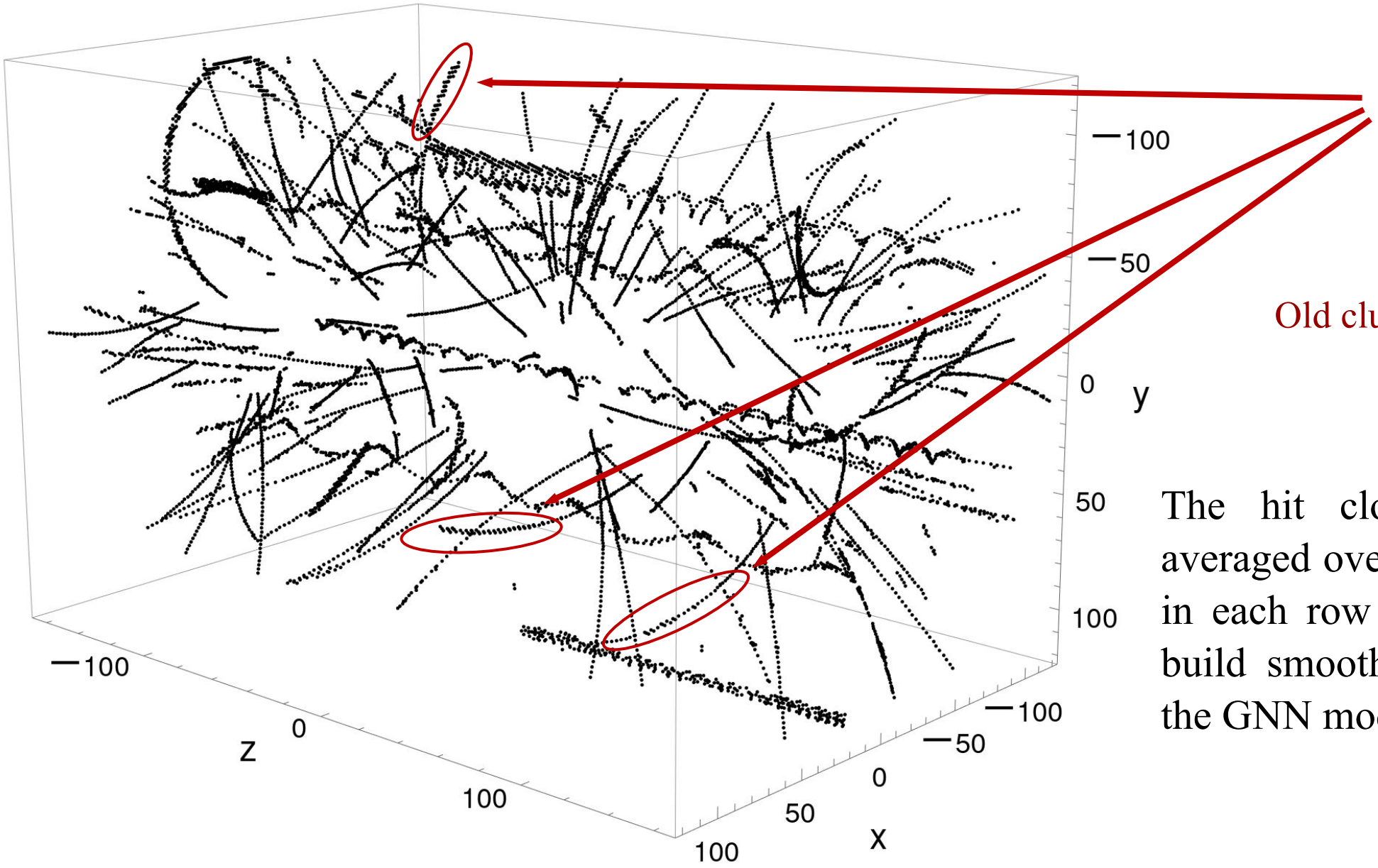
CustomWeightedGATConv

- The two-step aggregation was implemented to take into account the features of incoming and outgoing surrounding nodes in two coordination spheres.
- The edge labels at the current step are used for the attention mechanism.

```python
class CustomWeightedGATConv(MessagePassing):
    def __init__(self, node_hidden_dim, hidden_dims, output_dim, edge_hidden_dim, node_feature_dim,
                 activation=torch.nn.Tanh(), end_activation=None, dropout=torch.nn.Dropout(0.1)):
        super(CustomWeightedGATConv, self).__init__(aggr='add')
        self.mlp = CustomMLP(5 * node_hidden_dim + node_feature_dim + 4 * edge_hidden_dim, hidden_dims, output_dim, activation, end_activation, dropout)

    def forward(self, x, edge_index, edge_attr, edge_weight, initial_x):
        reversed_edge_index = edge_index.flip(0)

        first_step_incoming = self.propagate(edge_index, x=x, edge_attr=edge_attr, edge_weight=edge_weight, step=1)
        first_step_outgoing = self.propagate(reversed_edge_index, x=x, edge_attr=edge_attr, edge_weight=edge_weight, step=1)

        second_step_incoming = self.propagate(edge_index, x=first_step_incoming, edge_attr=edge_attr, edge_weight=edge_weight, step=2)
        second_step_outgoing = self.propagate(reversed_edge_index, x=first_step_outgoing, edge_attr=edge_attr, edge_weight=edge_weight, step=2)

        return self.final_update(initial_x=initial_x, x=x,
                                 first_step_incoming=first_step_incoming,
                                 first_step_outgoing=first_step_outgoing,
                                 second_step_incoming=second_step_incoming,
                                 second_step_outgoing=second_step_outgoing)

    def message(self, x_j, edge_attr, edge_weight, step):
        if step == 1:
            return edge_weight.view(-1, 1) * torch.cat([x_j, edge_attr], dim=-1)
        elif step == 2:
            return edge_weight.view(-1, 1) * x_j

    def update(self, aggr_out):
        return aggr_out

    def final_update(self, initial_x, x, first_step_incoming, first_step_outgoing, second_step_incoming, second_step_outgoing):
        out = torch.cat([initial_x, x, first_step_incoming, first_step_outgoing, second_step_incoming, second_step_outgoing], dim=-1)

        return self.mlp(out)
```



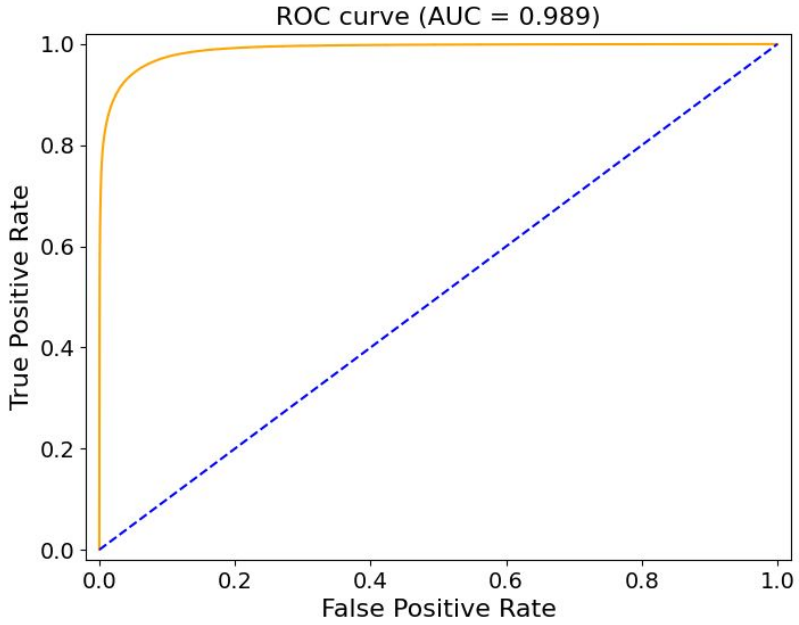
Dataset

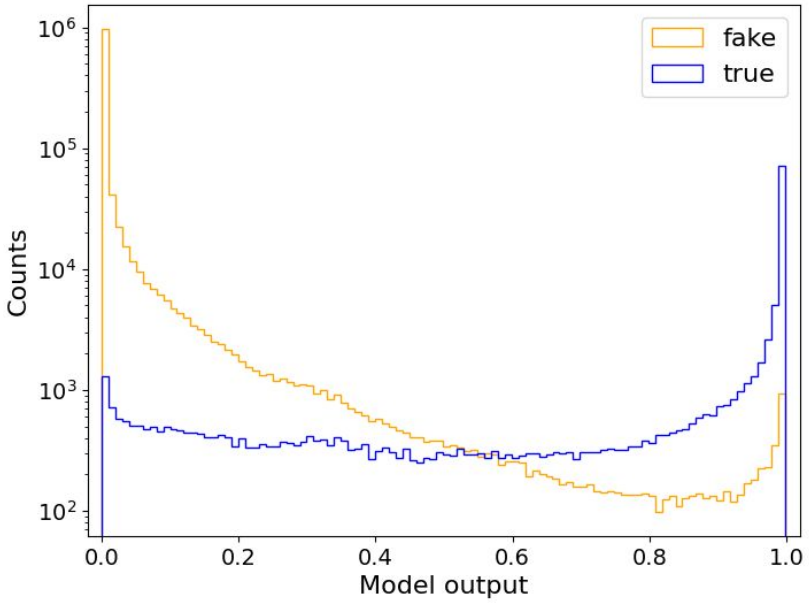


Old clustering algorithm!

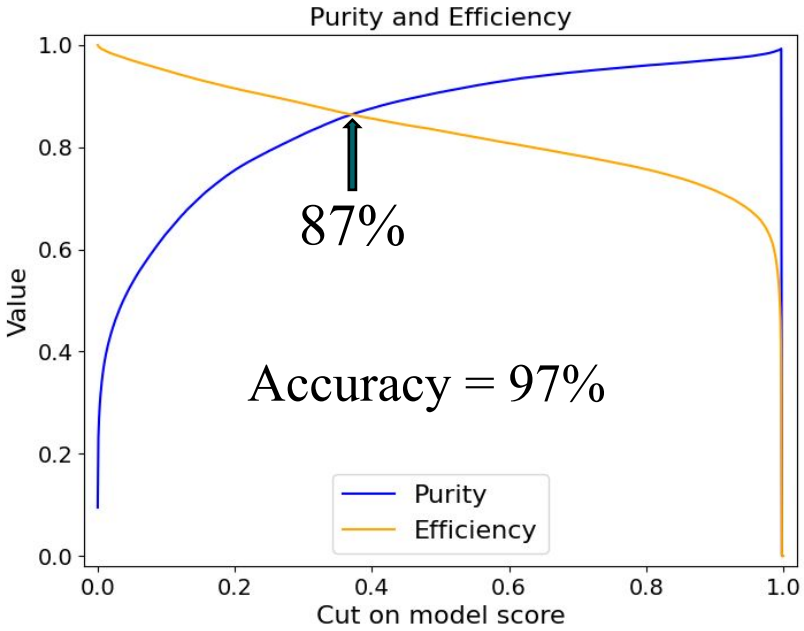
The hit closest to the point averaged over the φ angle is used in each row for each particle to build smooth tracks for training the GNN model.

Results

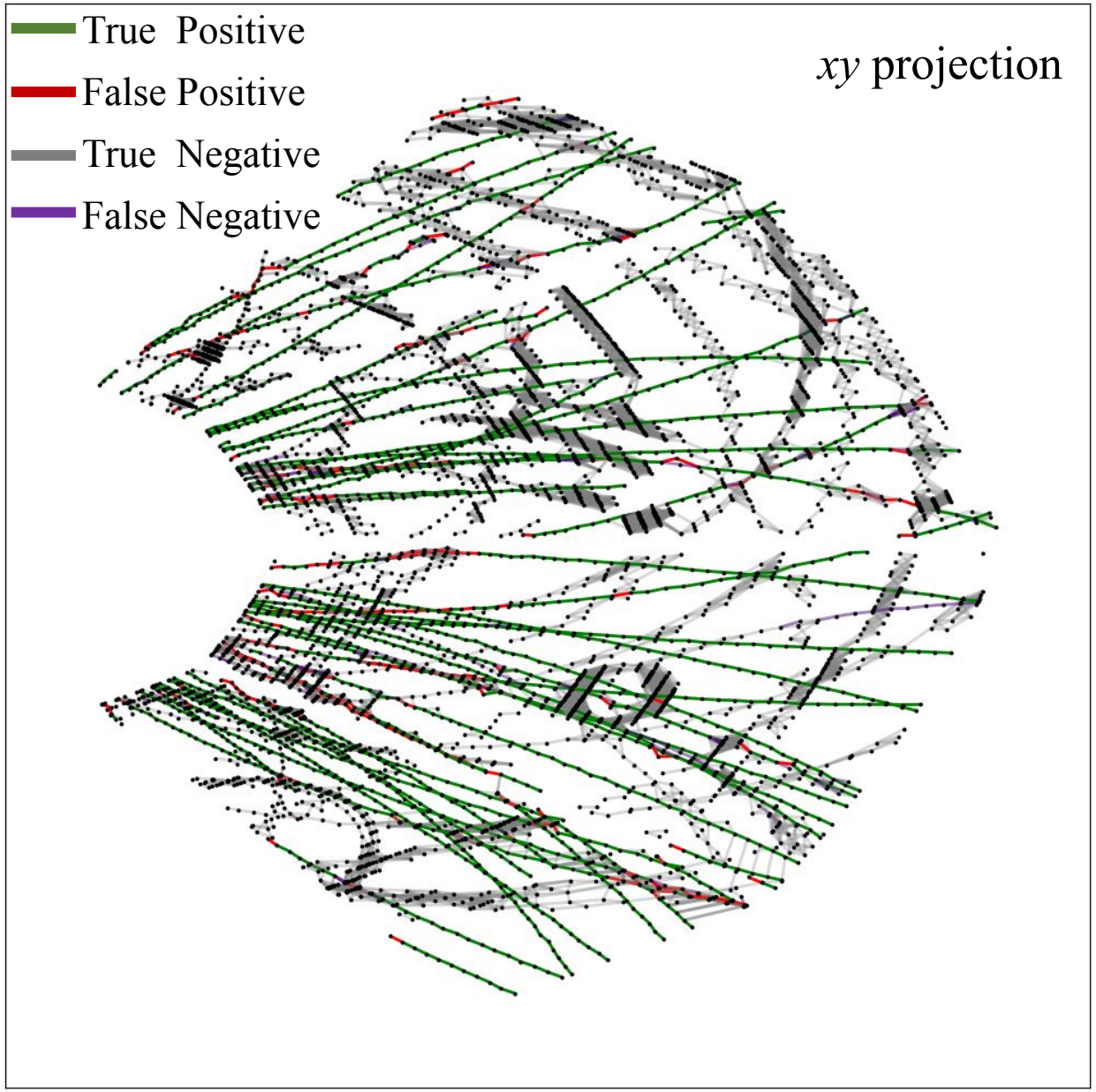




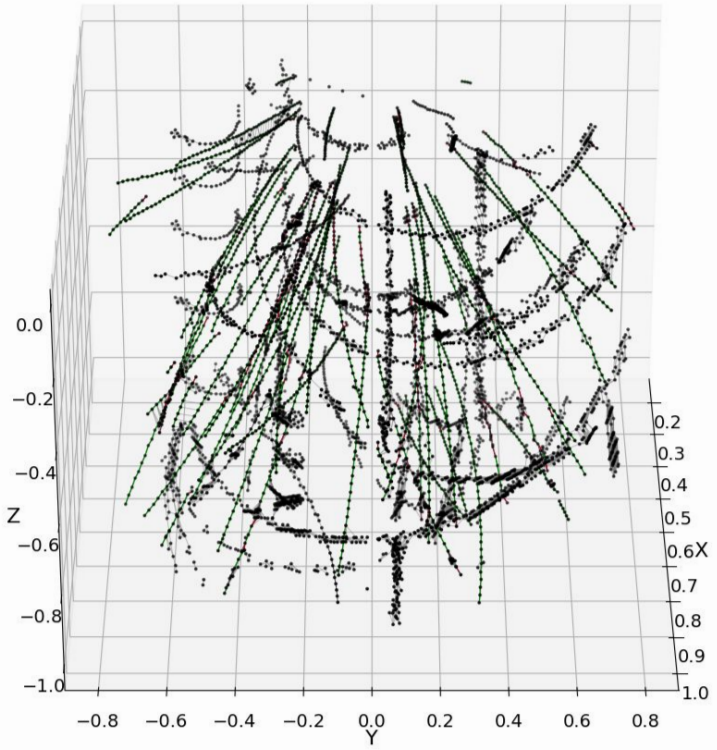
Epoch number = 70
Event number = 100



Results



3D view



Thank you for attention!