

# Particle tracking with graph neural networks

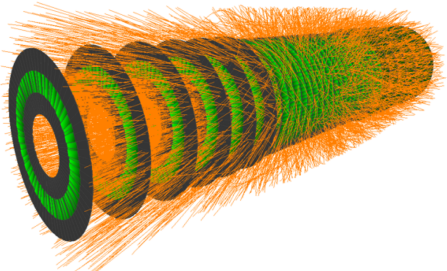
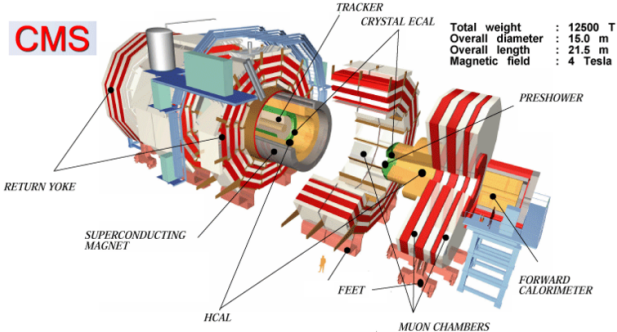
Shulgin Egor\*, Ratnikov Fedor

"Mathematical methods of pattern recognition" Conference  
Data Mining Section  
Moscow, Russia

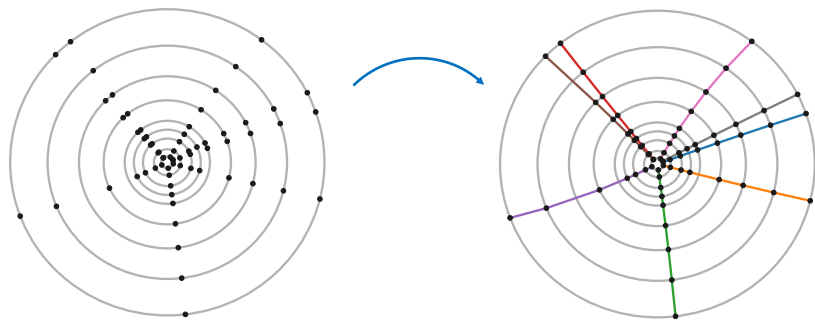
November 27, 2019

# Introduction to the problem

**CMS**



## Particle tracking problem



### Machine learning approach proposal

Treat this problem as a clusterization task.

**Object space:** hits  $\{x_1, \dots, x_n\} \in X \subset \mathbb{R}^3$

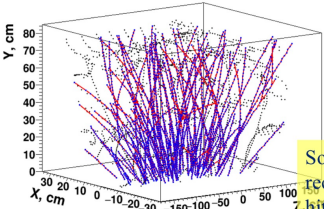
**Clusters:** tracks labels  $Y \subset \mathbb{N}$

**Required:** reconstruct mapping  $f : X \rightarrow Y$

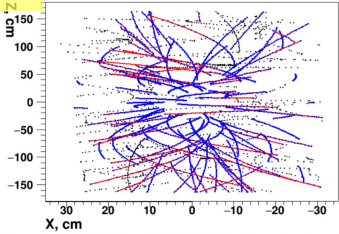
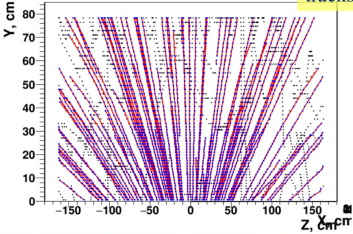
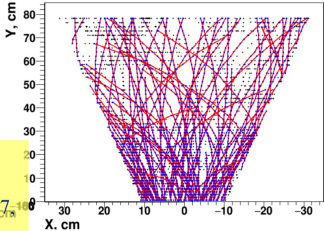
# Real data example



## Track reconstruction



Some stats:  
rec. points = 4867,  
hits on tracks = 3127,  
tracks = 102

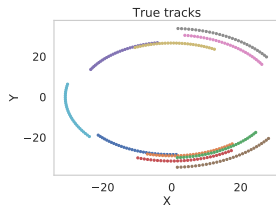
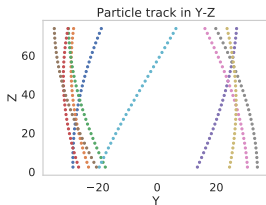
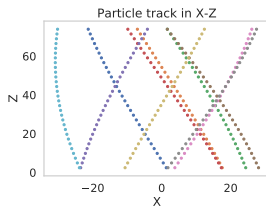
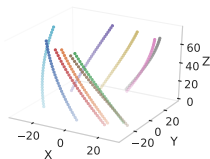


## Synthetic data

$$\begin{cases} t \sim \exp(\lambda) \\ x = x_0 + r \cdot \cos(a \cdot t + \phi_0) & + \mathcal{N}(\mu_x, \sigma_x^2) \\ y = y_0 + r \cdot \sin(a \cdot t + \phi_0) & + \mathcal{N}(\mu_y, \sigma_y^2) \\ z = z_0 + b \cdot t & + \mathcal{N}(\mu_z, \sigma_z^2) \end{cases}$$

# Synthetic data

$$\begin{cases} t \sim \exp(\lambda) \\ x = x_0 + r \cdot \cos(a \cdot t + \phi_0) + \mathcal{N}(\mu_x, \sigma_x^2) \\ y = y_0 + r \cdot \sin(a \cdot t + \phi_0) + \mathcal{N}(\mu_y, \sigma_y^2) \\ z = z_0 + b \cdot t + \mathcal{N}(\mu_z, \sigma_z^2) \end{cases}$$



# TrackML metric

Featured Prediction Competition

## TrackML Particle Tracking Challenge

High Energy Physics particle tracking in CERN detectors

\$25,000  
Prize Money

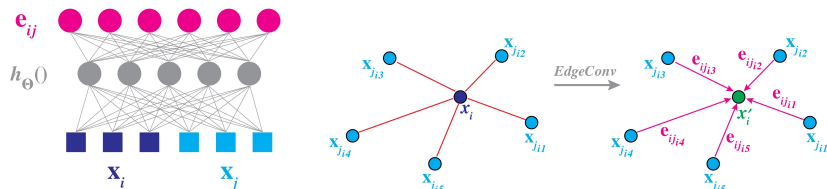
CERN · 651 teams · a year ago

[Overview](#) [Data](#) [Notebooks](#) [Discussion](#) [Leaderboard](#) [Rules](#) [Join Competition](#)

## Very short description:

It is the intersection between the reconstructed tracks and the ground truth particles, normalized to one for each event, and averaged on the events of the test set.

# Edge Convolution



**Point cloud:**  $X = \{x_1, \dots, x_n\} \subset \mathbb{R}^F$

$F$  represents the feature dimensionality of a given layer.

**Edge features:**  $e_{ij} = h_{\Theta}(x_i, x_j)$ , where

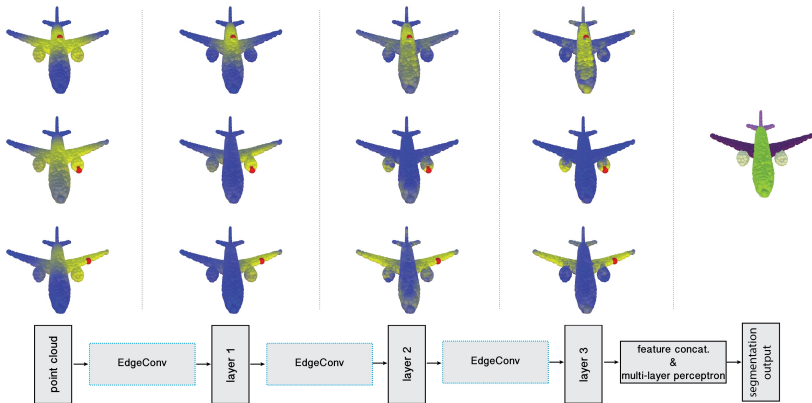
$h_{\Theta} : \mathbb{R}^F \times \mathbb{R}^F \rightarrow \mathbb{R}^{F'}$  – nonlinear function with a set of learnable parameters  $\Theta$ . The output of **EdgeConv** at the  $i$ -th vertex:

$$x'_i = \square_{j:(i,j) \in \mathcal{E}} h_{\Theta}(x_i, x_j)$$

$\mathcal{E}$  – edges of the  $k$ -nearest neighbor ( $k$ -NN) graph of  $X$  in  $\mathbb{R}^F$



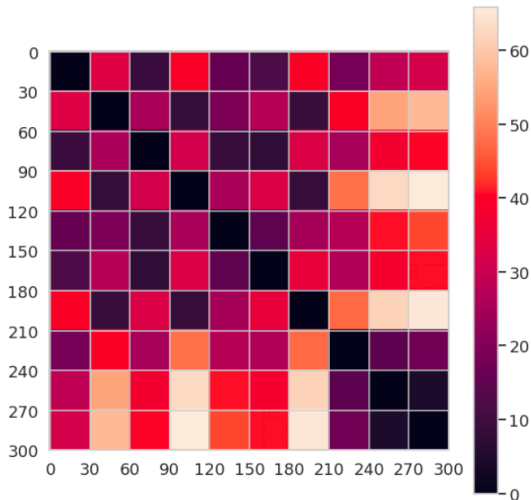
# Dynamic Graph CNN<sup>1</sup>



<sup>1</sup>Wang et al. Dynamic Graph CNN for Learning on Point Clouds, 2019

## Proposed approach for tracking

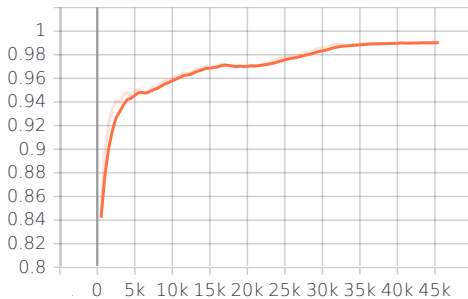
- ▶ DGCNN
- ▶ Calculation of the matrix of pairwise distances



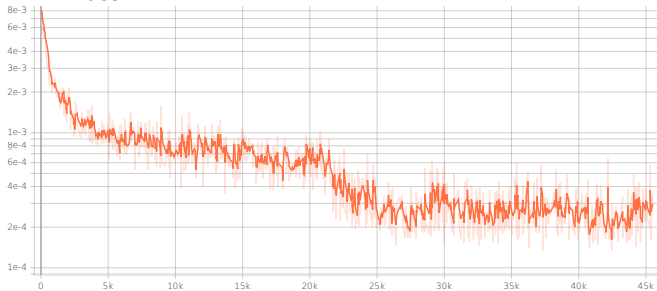
- ▶ Hierarchical agglomerative clustering

# Model training

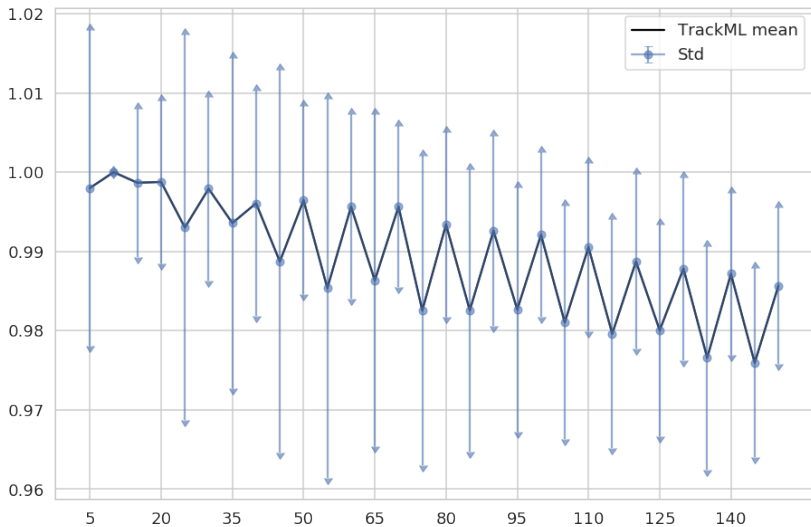
## Mean metric value on validation sample



## Loss



# Metric value dependence on the number of tracks



Thank you for your attention.

Any questions?