LOOT: Novel end-to-end trainable convolutional neural network for particle track reconstruction

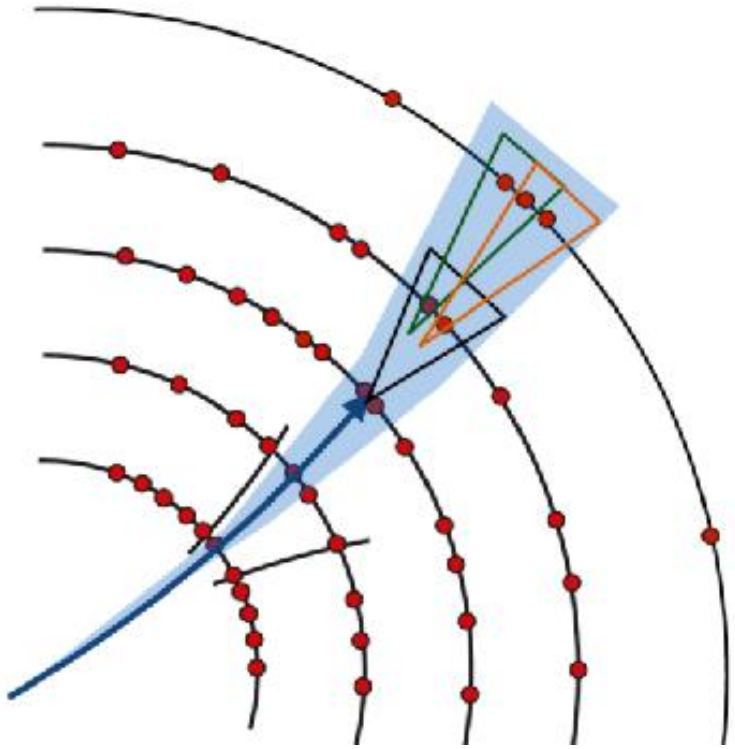
Pavel Goncharov, Gennady Ososkov, Dmitriy Baranov

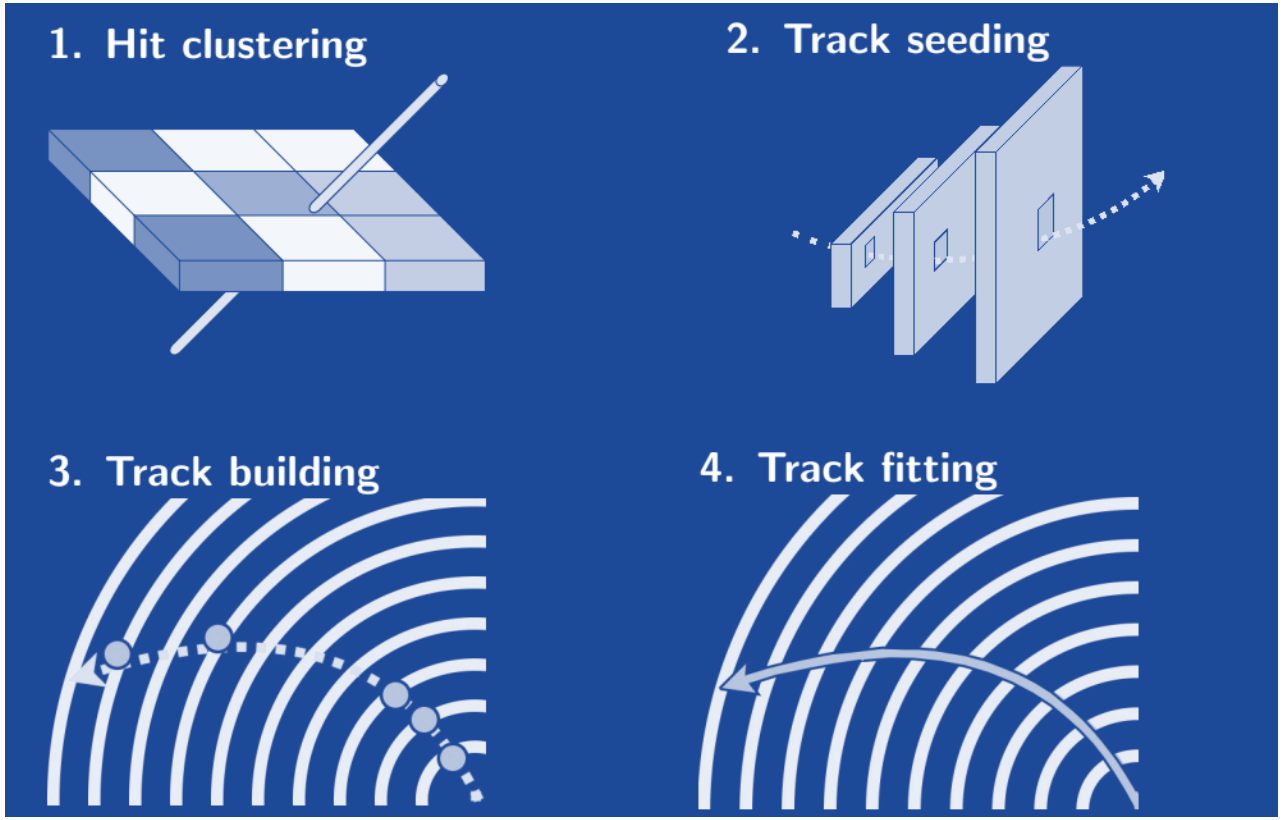
Dubna State University

kaliostrogoblin3@gmail.com

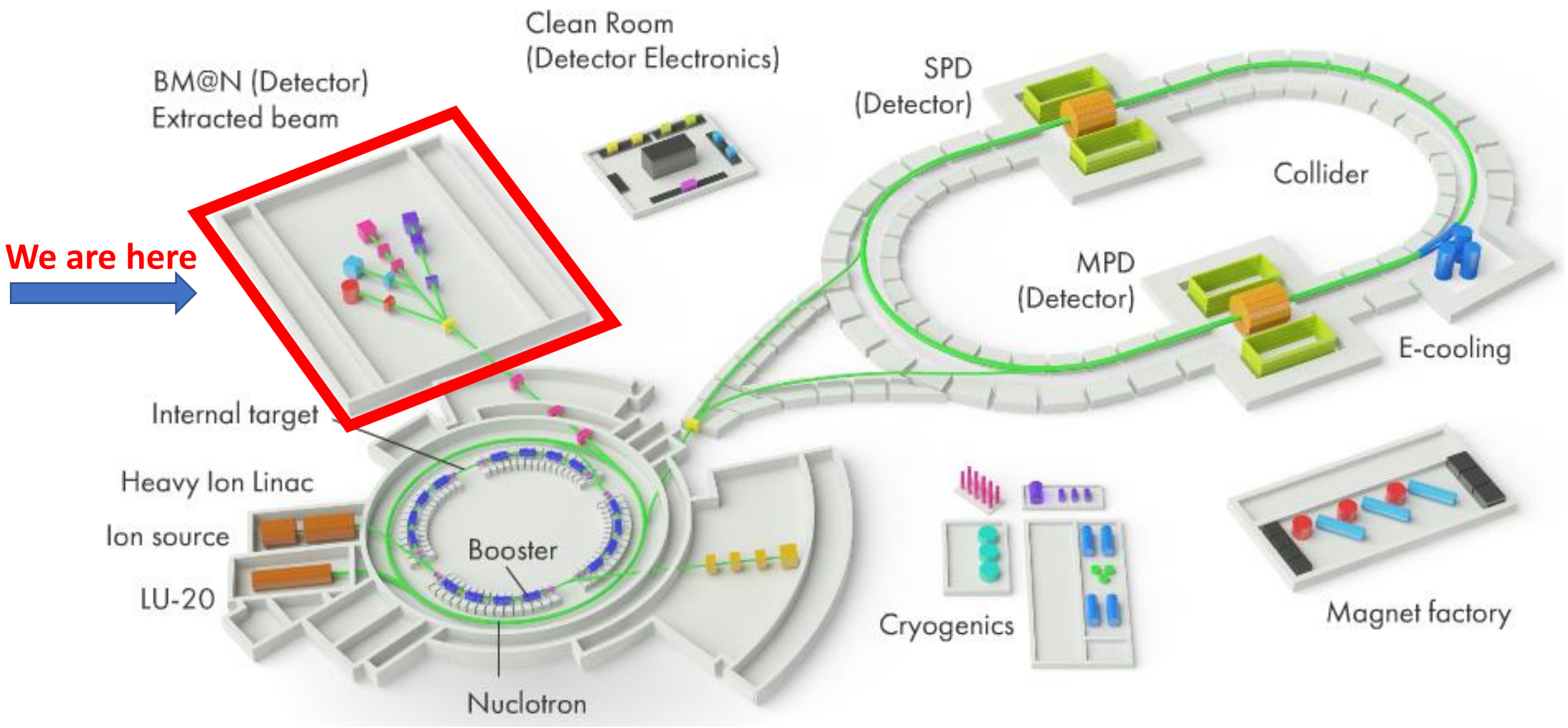
What is tracking?

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.



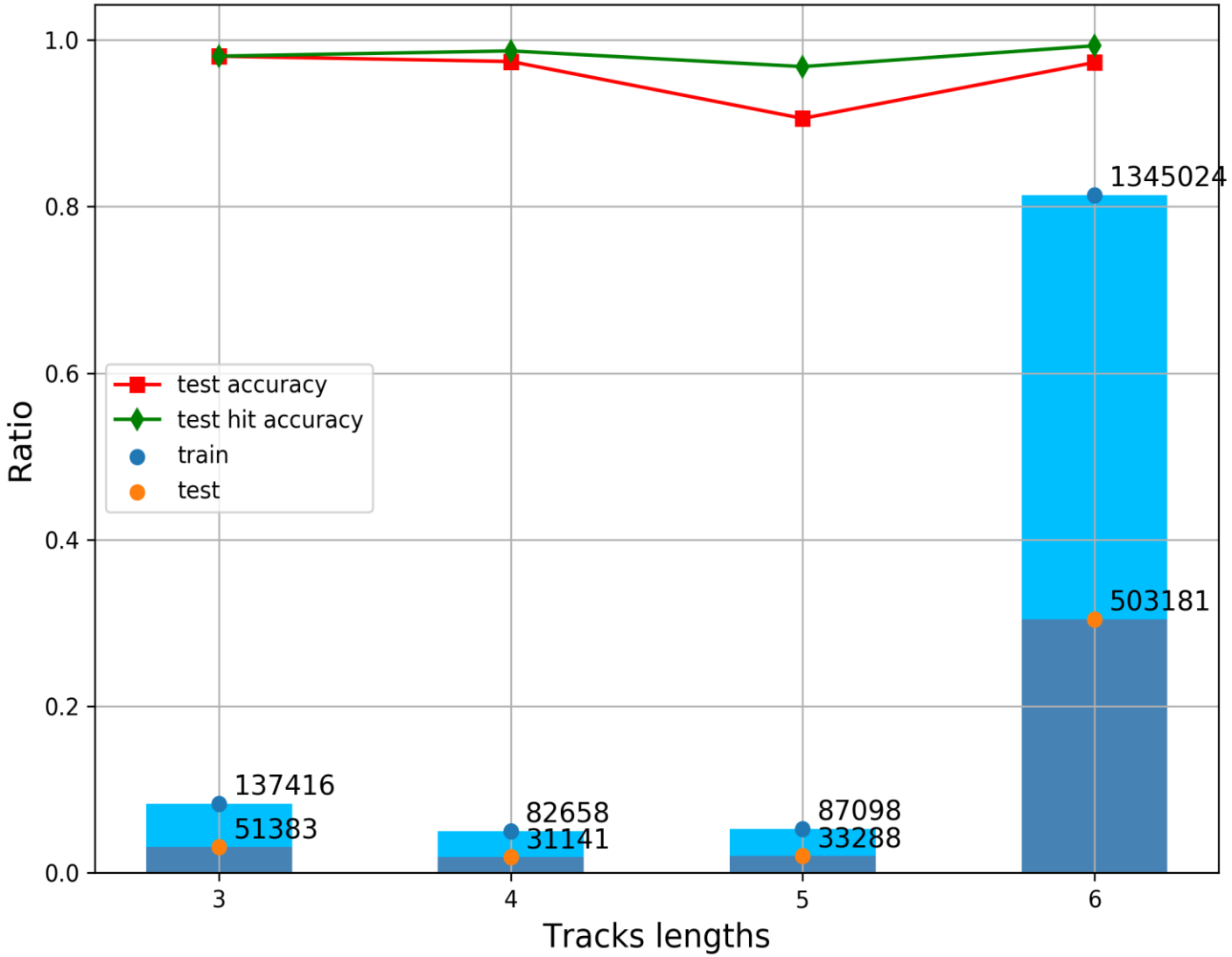


NICA-MPD-SPD-BM@N



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

Tracking with the TrackNETv2



Our last model achieved successful results on test dataset of Monte-Carlo simulation. The plot illustrates track accuracy and hit accuracy depending on the length of track.

Track accuracy – ratio of tracks the model recognized without any mistake.

Inference speed: 12K tracks/sec on Intel Core i3-4005U @1.70 ГГц

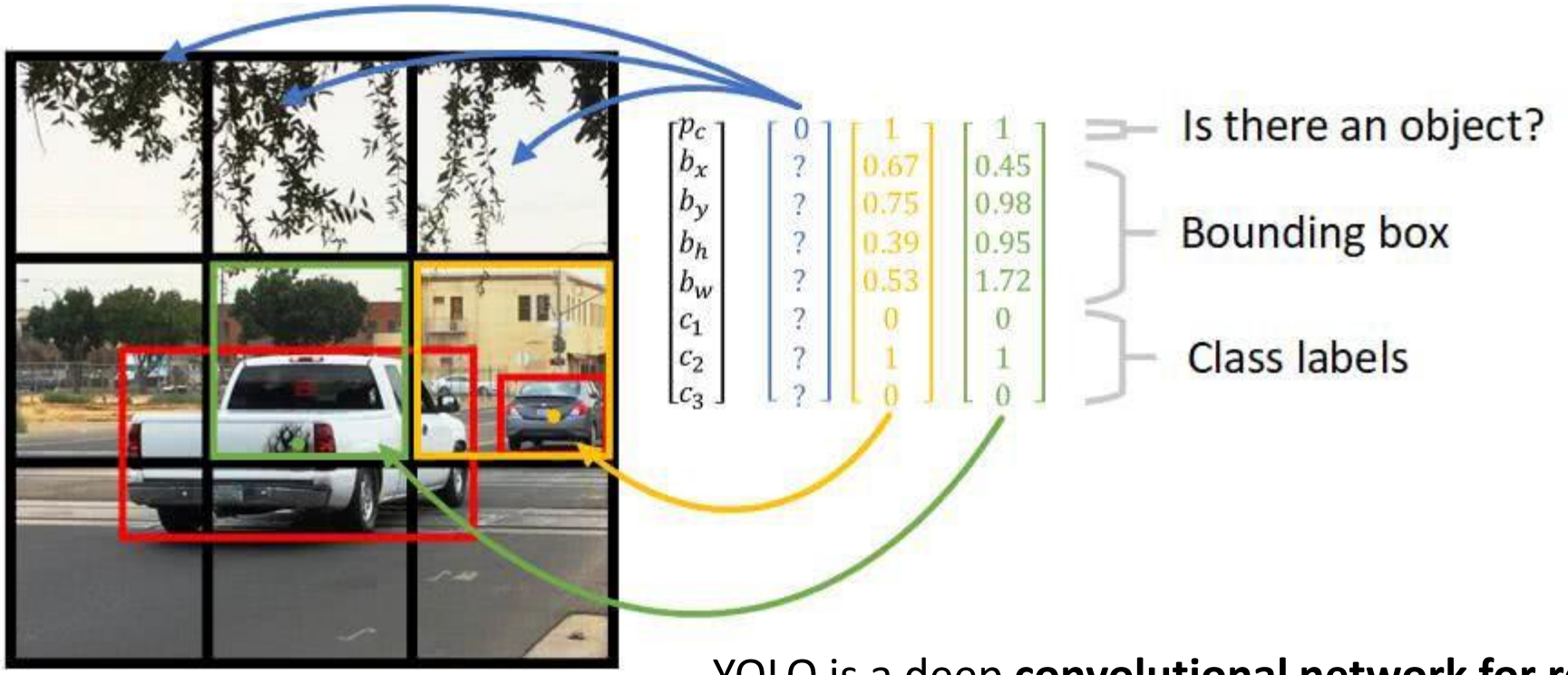
Mean track accuracy: ~96%

Problems of modern tracking

- Actually **it is not modern at all** – physicists use methods which were developed in past century
- **Almost all tracking methods are sequential** in their nature (e.g. Kalman filter)
- Many of algorithms for track reconstruction **work with tracks separately** – how to put the whole event as input into a recognition system?
- How to avoid **combinatorial problems and reduce memory usage**?
- **Number of tracks varies**
- **Lengths of tracks are different**

Deep learning opens a great opportunity to solve such a complex task as track reconstruction. But how to deal with the statements above? Are there any examples in related areas?

YOLO: You Only Look Once

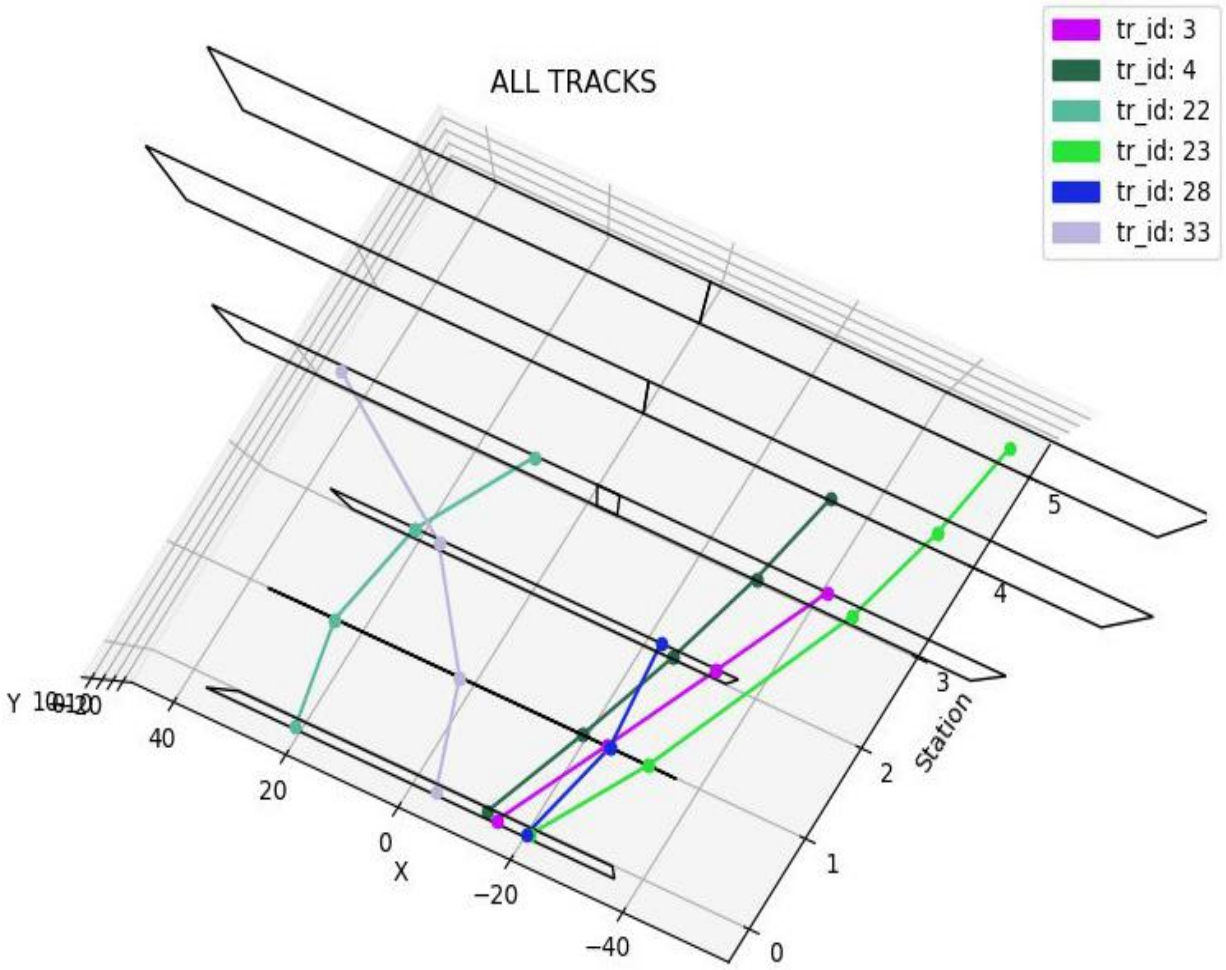


What if to use the main idea – prediction for cells?

YOLO is a deep **convolutional network for real-time object detection**. It

- splits input image on parts using a regular grid;
- for each block predicts the probability of the object existence inside;
- the center of the bounding box area;
- the size of the bounding box;
- class label for the object inside the grid cell;

Event as image. Stations like color



- Images have 3d format: **Height+Width+RGB**
- Data from each station – **sparse matrix of zeros and ones**, where ones indicate hits appearance
- **Events have 3d format too: Height+Width+Stations**

Our main idea is to use **OZ dimension instead of RGB channels** – it's a **radically new approach**.

Height and Width are the sizes of the largest station (most often the last).

Target explanation. Tracks behind bars

Detector's width

1

(0, 0) 0	(0, 1) 0	(0, 2) 0
(1, 0) 0	(1, 1) 1	(1, 2) 0
(2, 0) 0	(2, 1) 0	(2, 2) 0

Detector's height

Hits on the first station
(no fakes)

Y-coordinate shift from the starting point

Detector's width

(0, 0) 0	(0, 1) 0	(0, 2) 0
(1, 0) 0	(1, 1) 0.2	(1, 2) 0
(2, 0) 0	(2, 1) 0	(2, 2) 0

(0, 0) 0		
(1, 0) 0	(1, 1) 2.7	(1, 2) 0
(2, 0) 0	(2, 1) 0	(2, 2) 0

2

Detector's height

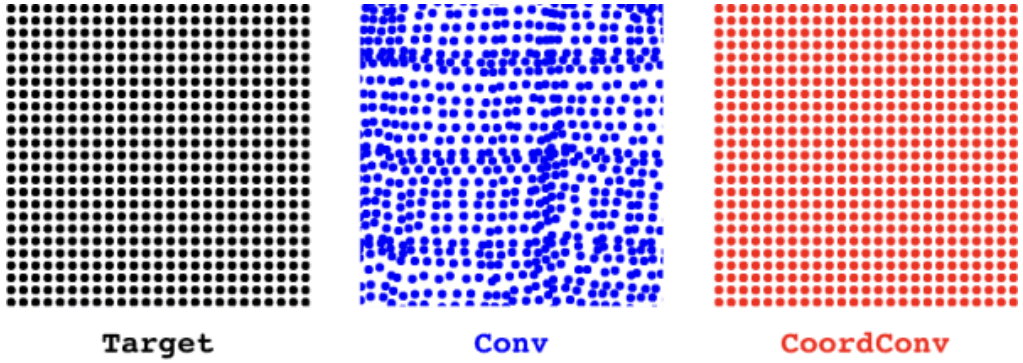
X-coordinate shift from the starting point

OUTPUT:

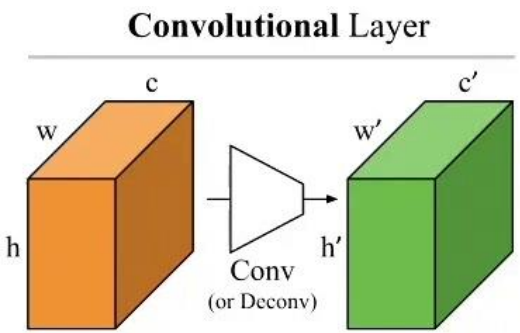
- (1) Mask for the first station – which of hits on the first station are **beginning of true tracks**
- (2) **Coordinate shifts** (x and y) from the previous hit of track **to obtain the current hit**
- **In total:** 1 mask + (n_stations – 1) * 2 shifts – multiplying by two, because 2 coordinate shifts
- **Result target dimension:** width × height × **n_stations * 2 – 1**, e.g. for 6 stations – 11 "rgb" channels

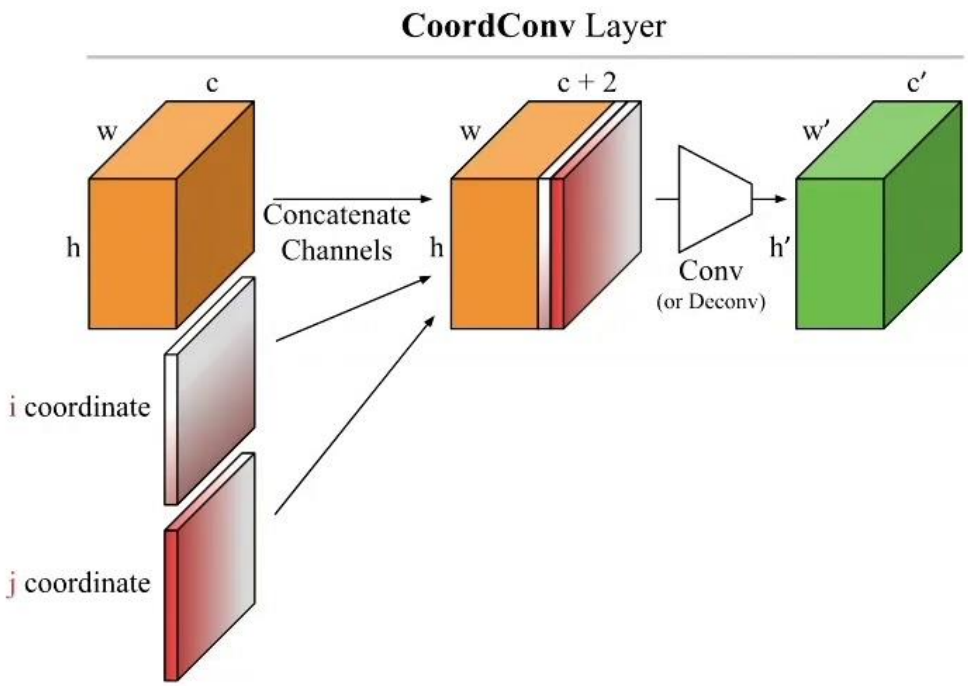
Coordinate convolution

https://arxiv.org/pdf/1807.03247.pdf



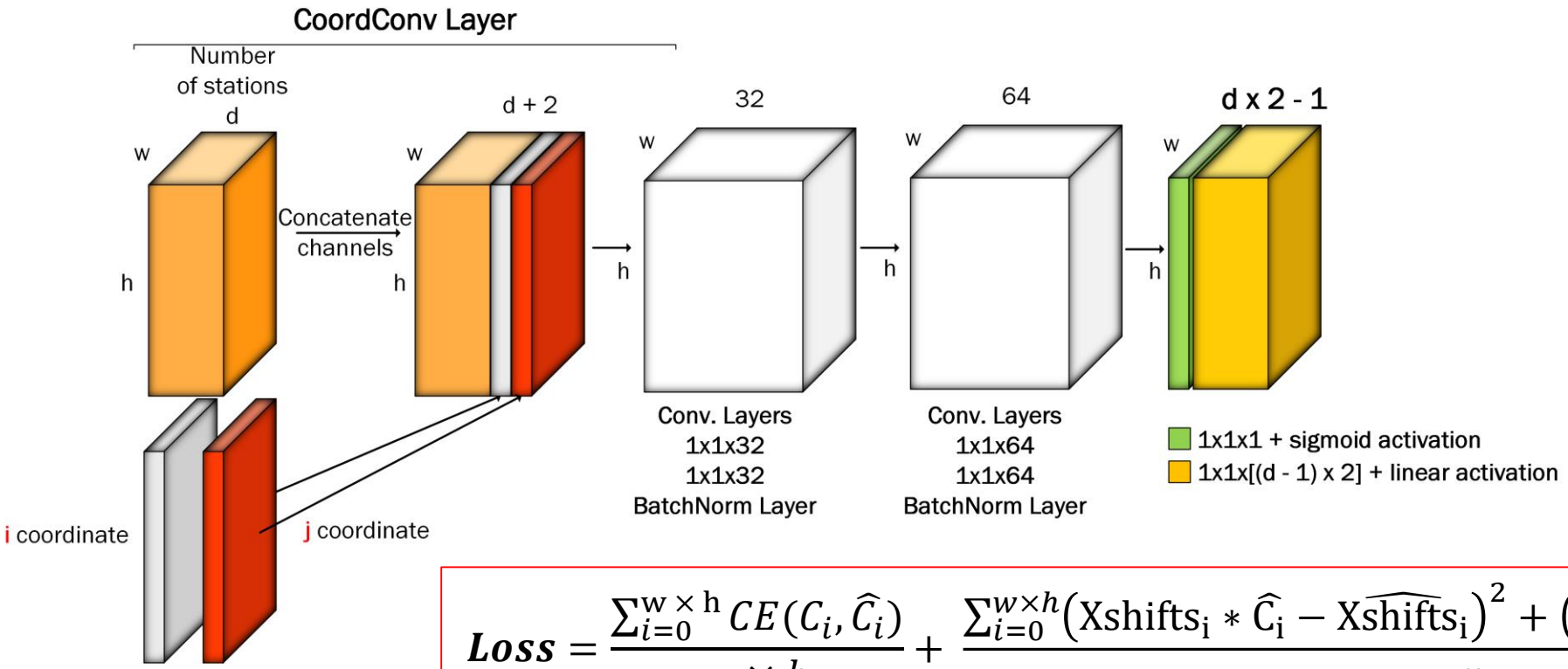
Coordinate transform problem: prediction of the coordinate grid is extremely difficult for the common **convolutional neural networks – they cannot learn coordinates from data samples**





The idea is to add coordinate channels to the input of deep neural network

Look Once On Tracks (LOOT) model and objective



$$\boldsymbol{Loss} = \frac{\sum_{i=0}^{\mathrm{w}\times\mathrm{h}} CE(C_i, \widehat{C_i})}{w \times h} + \frac{\sum_{i=0}^{w\times h}\left(\mathrm{Xshifts_i} * \widehat{\mathrm{C_i}} - \widehat{\mathrm{Xshifts_i}}\right)^2 + \left(\mathrm{Yshifts_i} * \widehat{\mathrm{C_i}} - \widehat{\mathrm{Yshifts_i}}\right)^2}{nz}$$

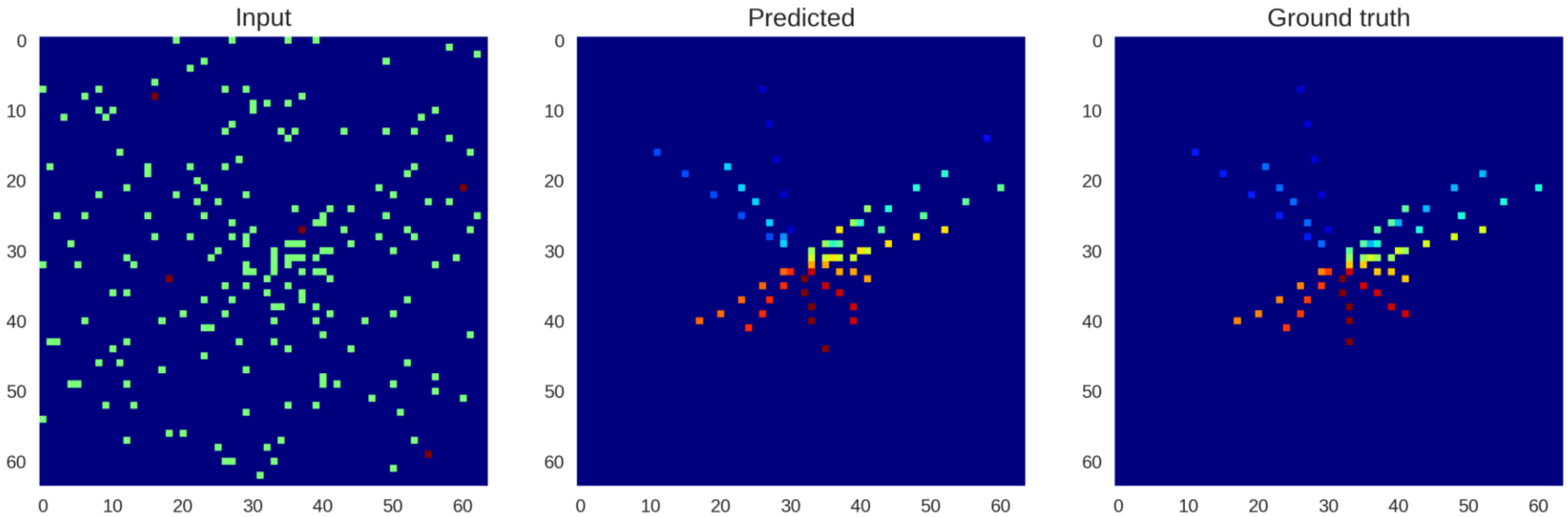
- $CE(C_i, \widehat{C_i})$ – binary cross-entropy between the predicted confidence scores – C_i, and the corresponding labels of a true track presence – $\widehat{C_i}$,
- $w \times h$ is the total number of grid cells,
- nz – the number of nonzero elements (avoid dividing by $w \times h$ because of a strong sparsity);
- multiply the shifts by the first station hits matrix to omit the cells with fakes, they do not have to take part in shifts calculation.

LOOT results on the toy dataset

Dataset:

- 160K events
- No magnetic field (straight tracks)
- Fake hits – uniform noise (twice greater than tracks)
- No. tracks – from 10 to 100 per event

# of tracks in event	10 tracks	50 tracks	100 tracks
Hit efficiency	97.37%	96.28%	95.39%
Efficiency	97.68%	96.60%	95.35%



Cross-entropy failure. Dice loss

Adapting LOOT to the Monte-Carlo simulation data for the BM@N experiment we faced with the problem, when **all output confidences (the mask for the starting hits of tracks) become equal to zero**. It happened because the **number of empty cells is much greater ($\boldsymbol{n^2}$)** than for the true tracks.

How we solved: replaced cross-entropy loss to the Dice loss function which is very common loss in the segmentation problems.

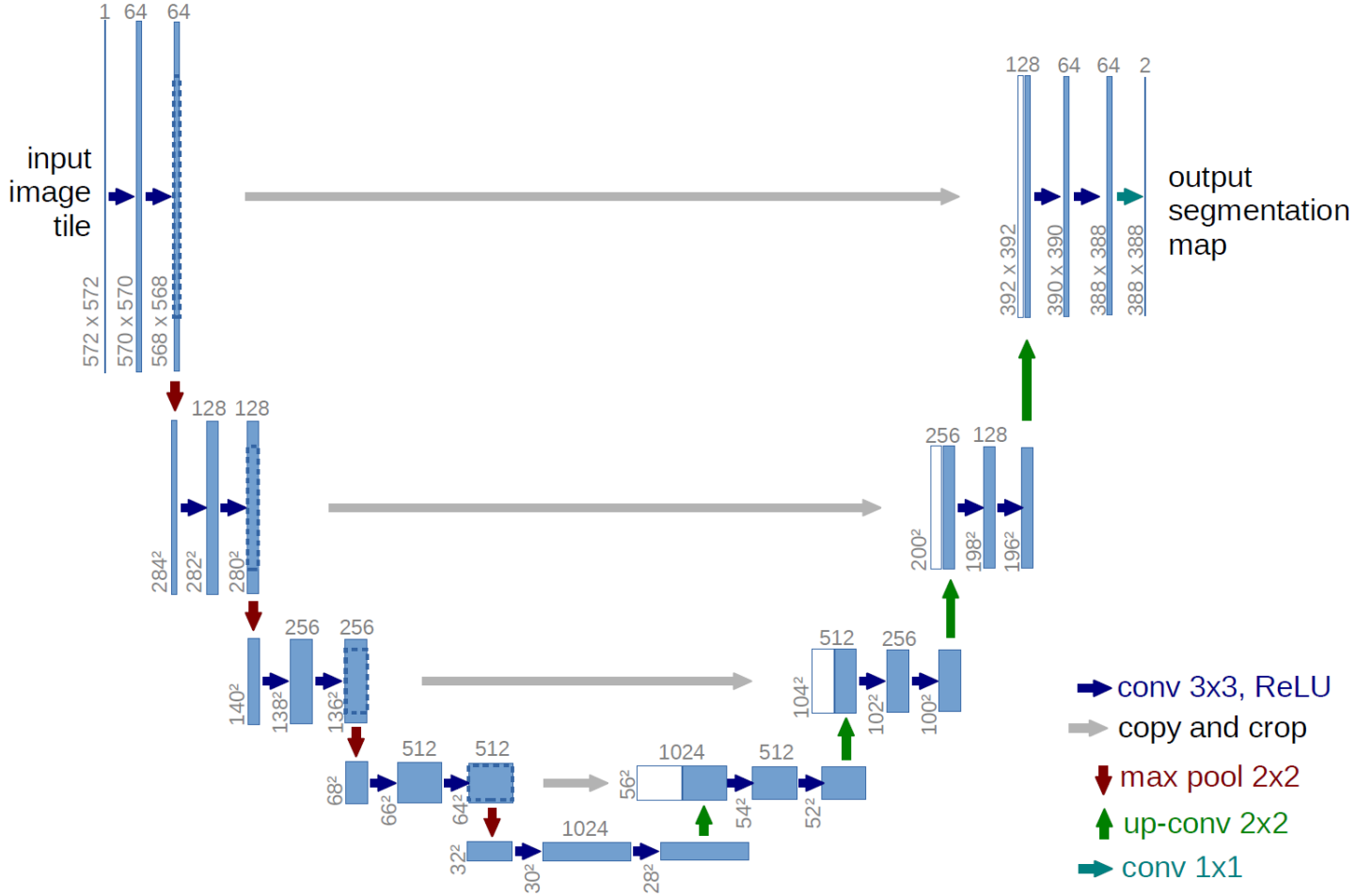
$$DiceLoss(p, \hat{p}) = 1 - \frac{2p\hat{p} + 1}{p + \hat{p} + 1}$$

where $p \in \{0,1\}$ and $0 \leq \hat{p} \leq 1$.

Cross-entropy failure. U-net architecture

U-Net is a convolutional neural network that was developed for biomedical image segmentation.

- Network consists of **a contracting path and an expansive path**, which gives it the **u-shaped architecture**.
- **During the contraction, the spatial information is reduced** while feature information is increased.
- The **expansive pathway combines** the feature and spatial information through a sequence of up-convolutions and concatenations with **high-resolution features from the contracting path**.



We used U-net with some small modifications as the LOOT base network preserving the output layer from the original LOOT.

U-LOOT results

Dataset:

- LAQGSM generator
- 4 GeV carbon-carbon interactions
- 166K train and 62K test events
- Only events where all tracks have number of hits equals to number of stations

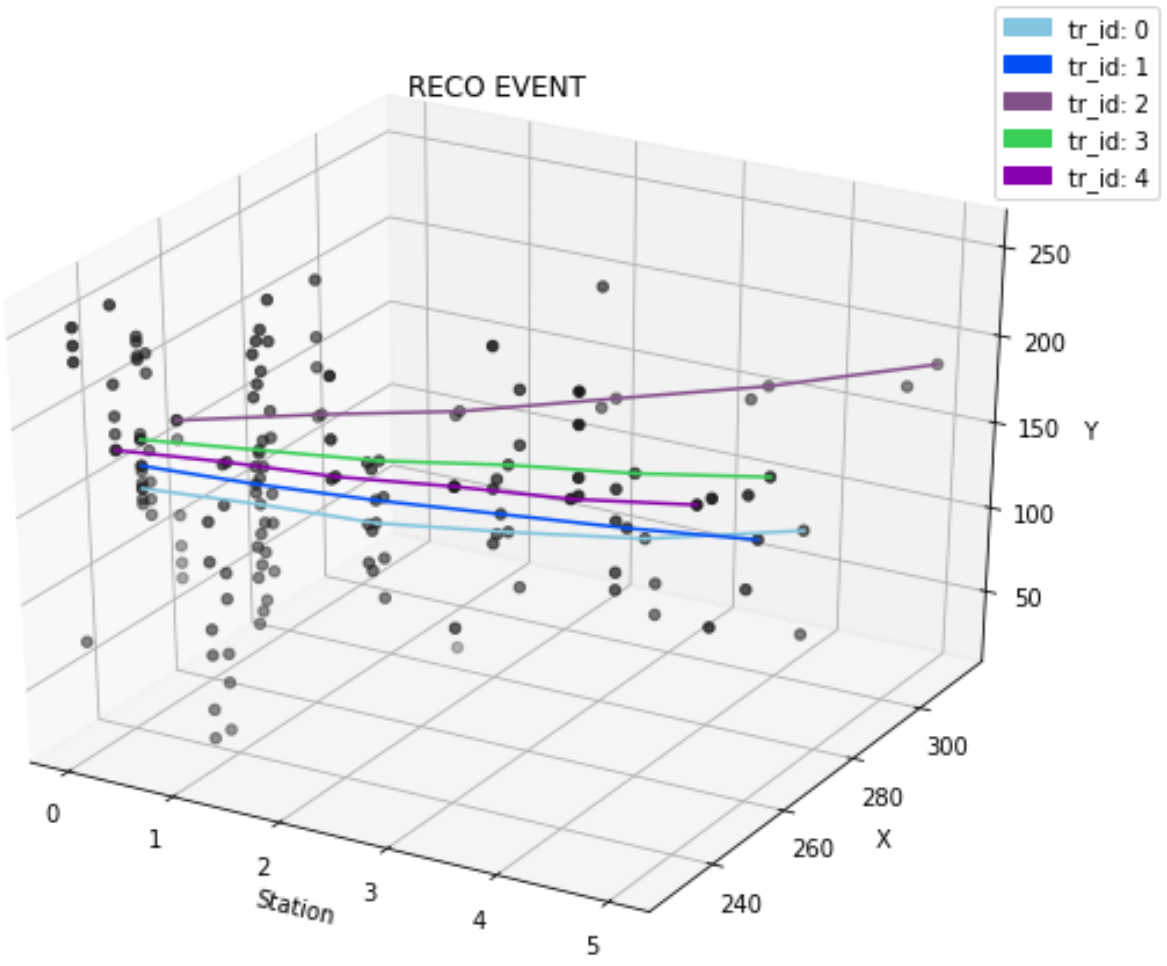
Training setup:

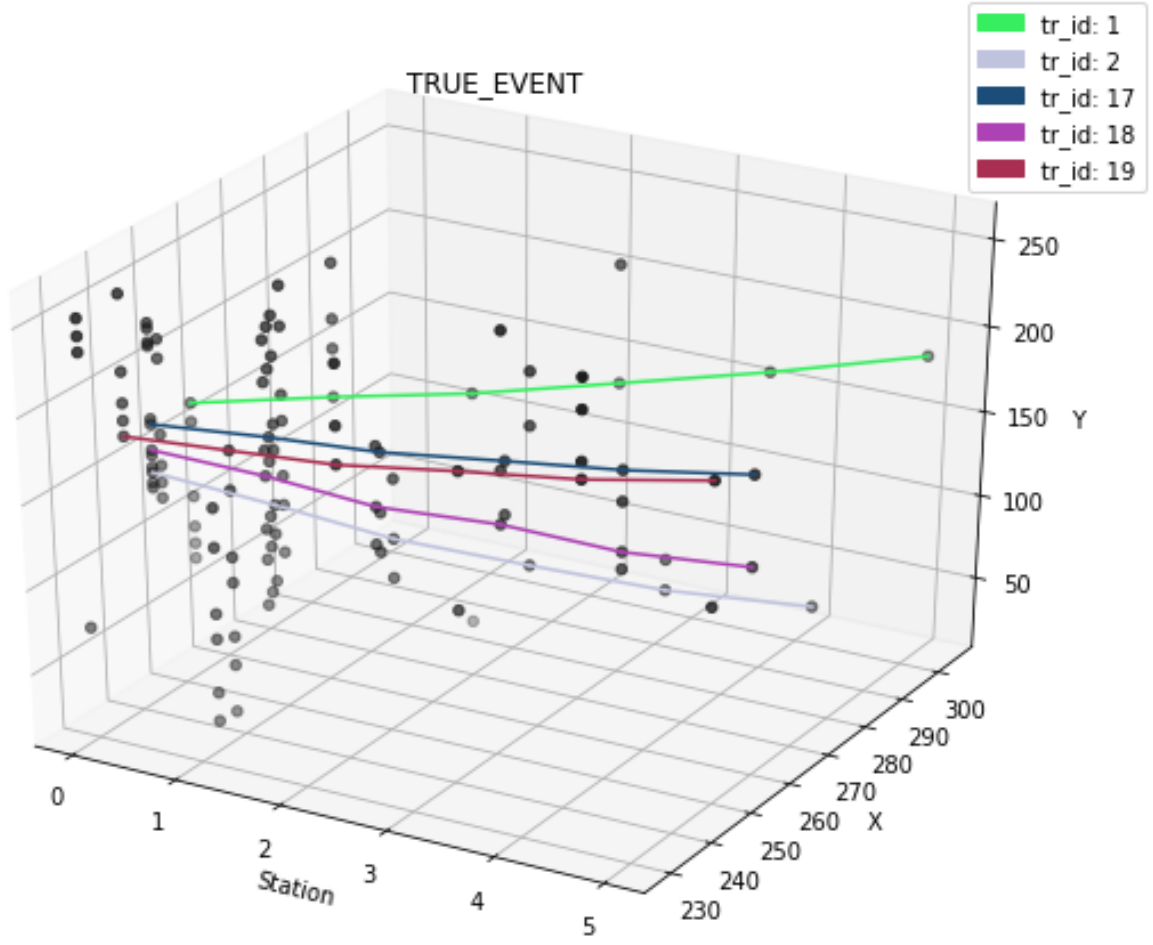
- 50 epochs;
- stochastic gradient descent with momentum;
- learning rate – 0.1, momentum – 0.9;
- cyclic learning rate update;
- batch size – 32

- **Recall** expresses the ability to find all true tracks in a dataset
- **Precision** expresses the proportion of data, our model says was true, actually were true tracks

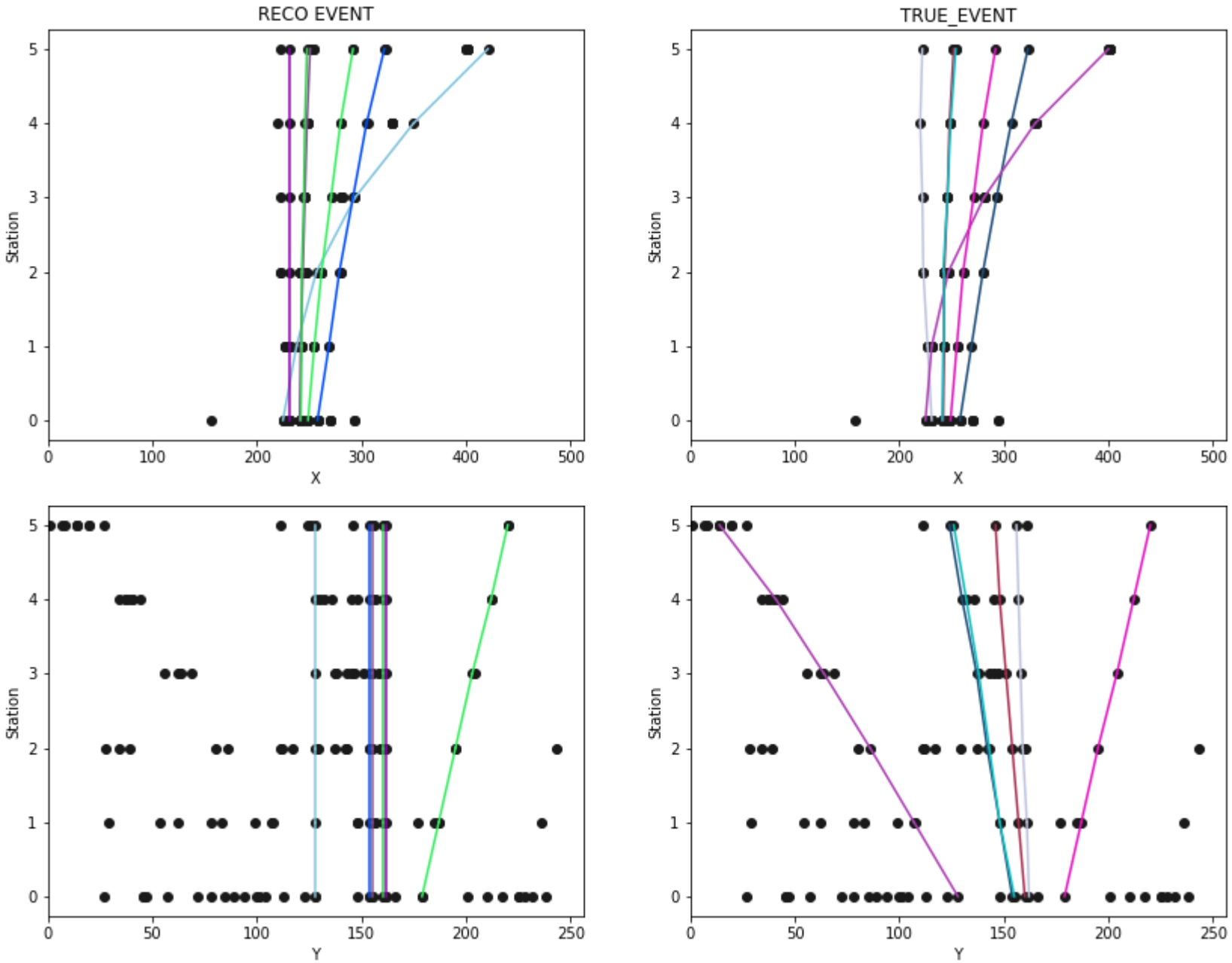
Precision	0.9719
Recall	0.9690
Shifts MSE	0.0152
Loss	0.0281
Processing speed (event/sec), batch size – 16	150

LOOT results visualization





LOOT problems with Y-shifts



OX projection

OY projection

- LOOT suffers from **very high error on OY** axis.
- It **occurred because of using mean-squared error** as shifts loss.
- **X shifts orders of magnitude greater than Y shifts**, so model learns to reduce OX error while preserving the OY error constant.

Conclusion and Outlook

We have introduced the radically new approach to the problem of tracking and have presented the LOOT model and its modification – U-LOOT, which is

- fully end-to-end trainable;
- consumes the whole event at time;
- doesn’t depend on the number of fakes and tracks;
- memory cheaper than the graph approaches;
- greatly drops out fake hits.

Now we are going to

- reduce the OY error by utilizing weighted shifts loss;
- improve the model to work on tracks with different lengths;
- try to predict track momentum instead of shifts;
- vertex prediction;
- expand the model’s powers to solve tracking in a collider environment.

LOOT: Novel end-to-end trainable convolutional neural network for particle track reconstruction

Pavel Goncharov, Gennady Ososkov, Dmitriy Baranov

Dubna State University

kaliostrogoblin3@gmail.com