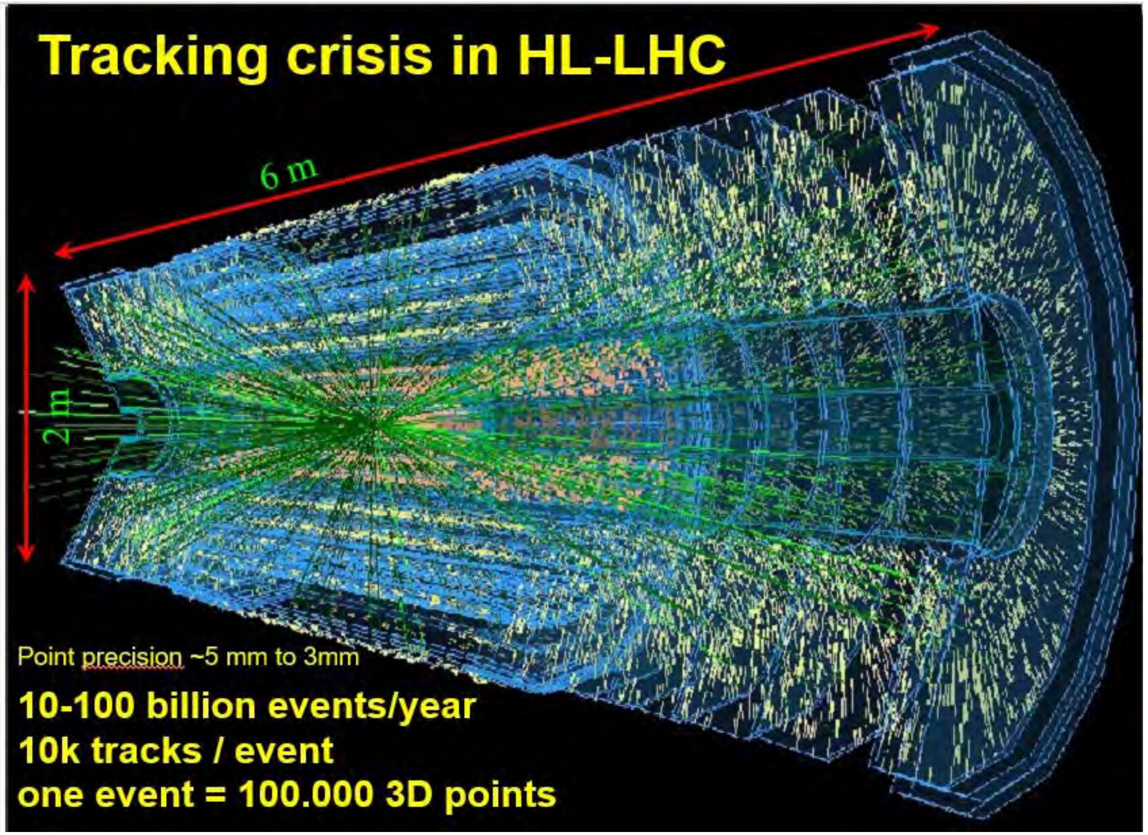
Particle track reconstruction at scale: online tracking with PyTorch

Rusov D., Goncharov P., Nikolskaia A., Ososkov G., Zhemchugov A.

The study was carried out with the financial support of the Russian Science Foundation No. 22-12-00109

Problem Statement

- **Particle track reconstruction in dense environments such** as the Run-4 detectors of the CERN High Luminosity Large Hadron Collider (HL-LHC) as well as MPD NICA is a **challenging pattern recognition problem**.
- To achieve such high luminosity, the particles are not accelerated individually, but in bunches, so that the moments of collisions occur so close to each other that the event tracks overlap strongly.
- For the SPD experiment, in which events are expected to arrive with a frequency of 3 MHz, the data acquisition is supposed to be performed in time slices, during one time slice up to 40 events with overlapping tracks may appear.
- Of the entire stream of events, only a few percent are of interest to physicists.


Tracking crisis in HL-LHC
6 m
2 m
Point precision ~5 mm to 3mm
10-100 billion events/year
10k tracks / event
one event = 100.000 3D points

- Therefore, it is **necessary to develop an intelligent online filter to sift out uninteresting events.**

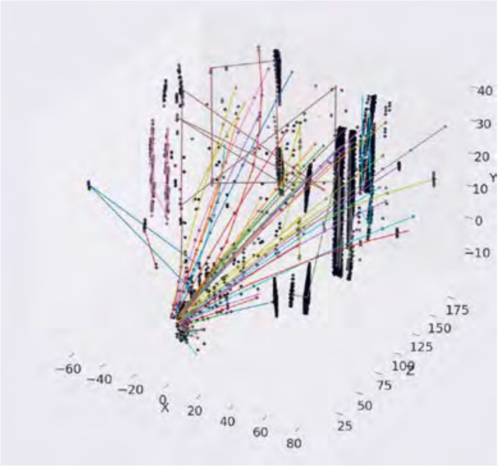
Deep Learning Comes to the Rescue

Deep learning algorithms bring a lot of potential to the tracking problem, due to

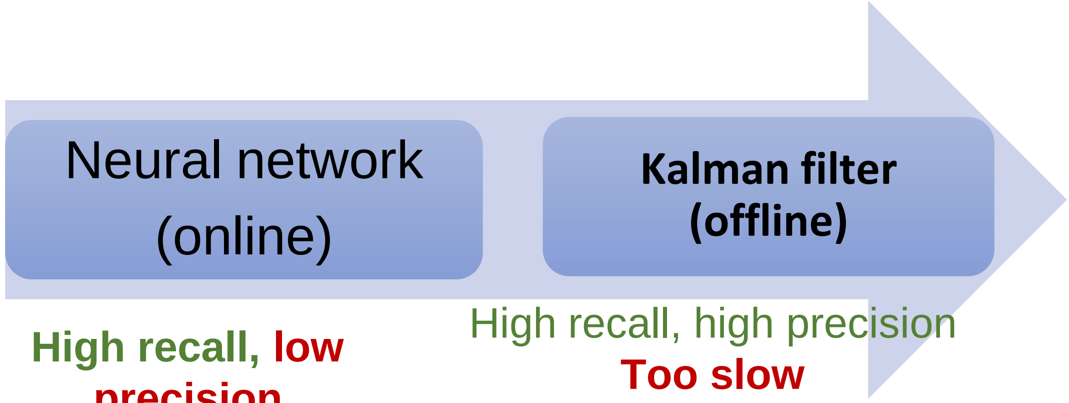
- their capability to model complex non-linear data dependencies,
- learn effective representations of high-dimensional data through training
- parallelize easily on high-throughput architectures such as GPUs

Reconstructed event

What we propose for track building?



Spatial points: hits of event

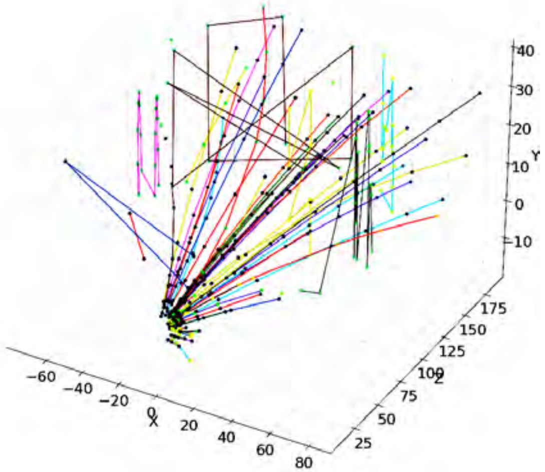


Neural network (online)

Kalman filter (offline)

High recall, low precision
Very fast

High recall, high precision
Too slow

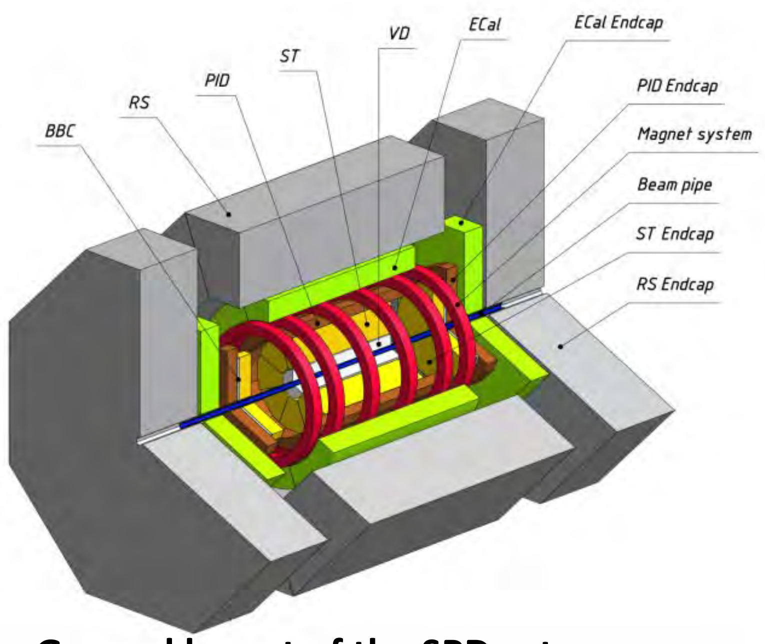


Grouped hits: tracks

SPD Experiment

SPD (Spin Physics Detector) – is a future experiment of NICA facility in Dubna. The main goal of the experiment is to test the foundations of quantum chromodynamics (QCD).
Event data from SPD comes in the form of time slices with a length of 10 ms and about 40 events in time slice will be produced.
On average there are **200 tracks per time slice and 1100 hits (real and fake) per station**.

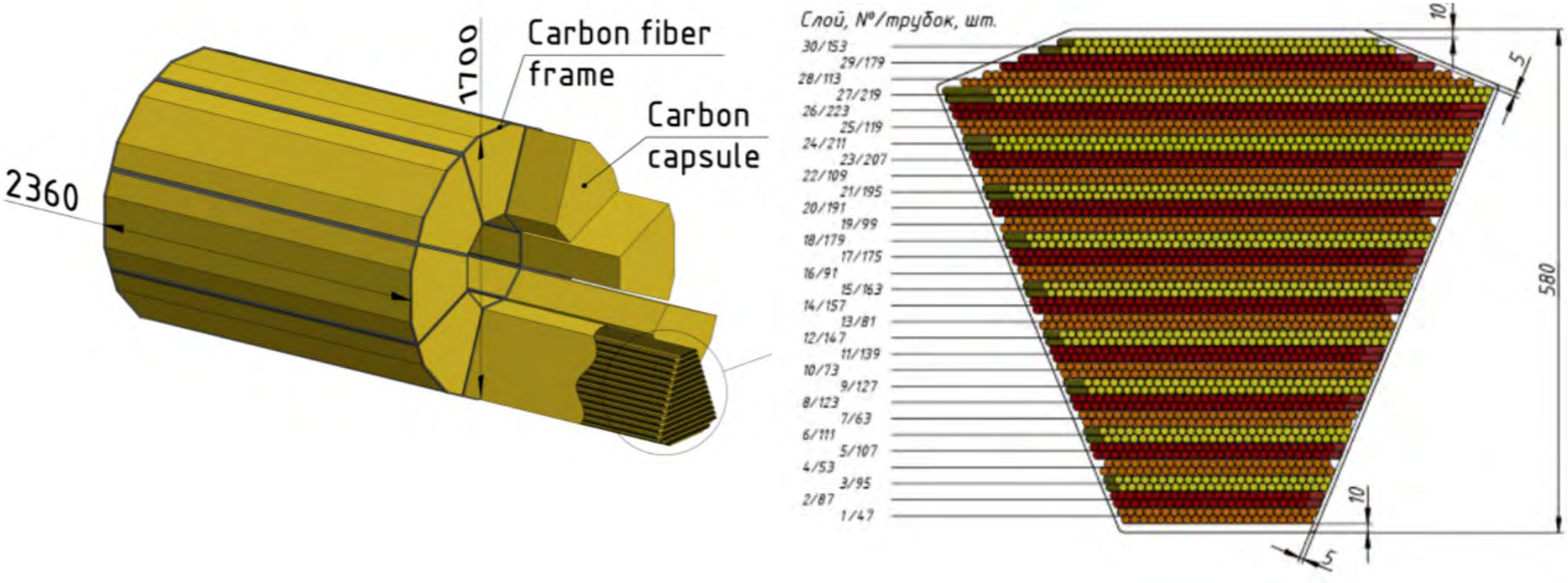
As an online filter, the algorithm must allow processing over 1000 events per second.



General layout of the SPD setup

(x2 zoom)
Without cover

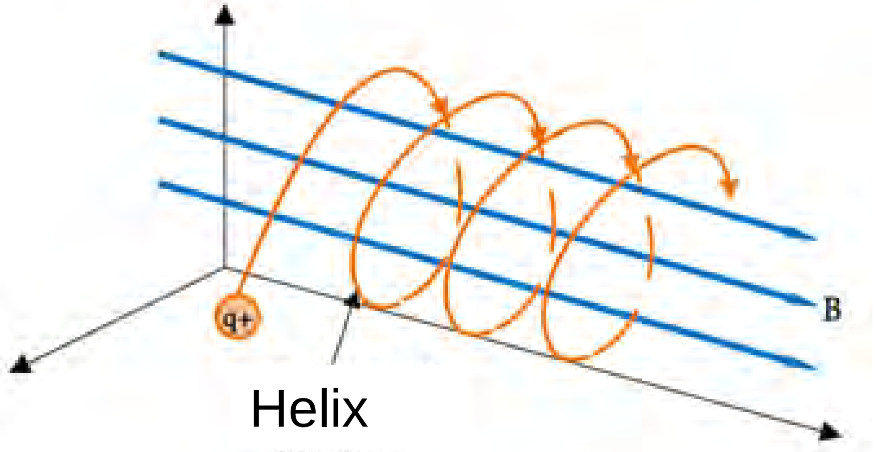
Layer, № / tubes, pcs



External view of the SPD straw detector (left) and its module in section (right)

Events Data Generator

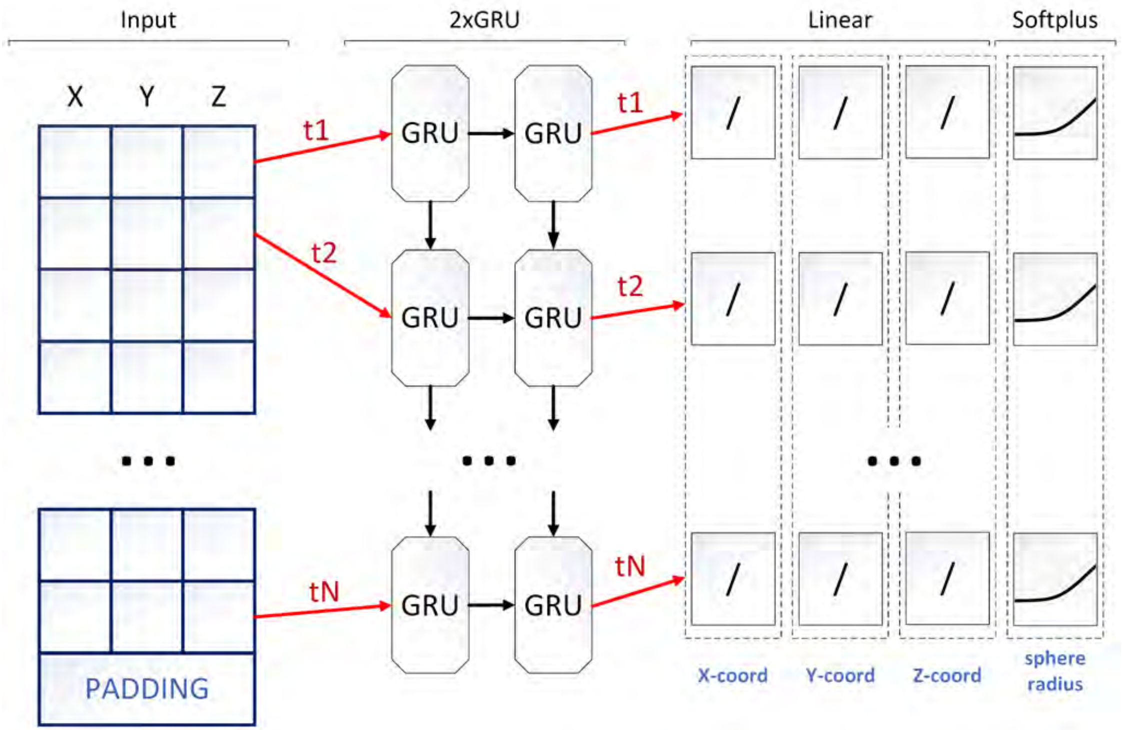
- Generator is written as a simple Python program.
- Multiplicity in each event is given by a random number from **1 to 10 tracks per event**.
- The transverse momentum of a particle is a random number with a uniform distribution in the range of values from 100 to 1000 MeV/s.
- Vertex coordinates are also random in the volume corresponding to the area of particle collisions.
- The particle trajectory is represented by a selection of points on a segment of a helix with a helix pitch $h = \frac{2\pi}{B}|\frac{m}{q}|vcos\alpha$ and radius $R = \frac{1}{B}|\frac{m}{q}|vsin\alpha$.
- Detector configuration with 35 stations is considered.
- Detector inefficiency is modeled by the probability of being “missed” for each hit independently. The detector efficiency values of 99% and 98% were used.



Helix

Local Tracking with TrackNET. Model overview

TrackNET model



How the model works?

- TrackNET is a model for local track reconstruction.
- Locality – one particular track-candidate during the prediction phase.
- The model predicts the center and radius of the sphere where to search for the next hit.
- All hits are placed in the spatial search index.
- Only K nearest to the center of sphere hits are checked (setting K=1 leads to linear computational complexity).
- Candidate tracks are extended by hits that fall into sphere.
- Extended track-candidates are fed back to the model input.

Pros:

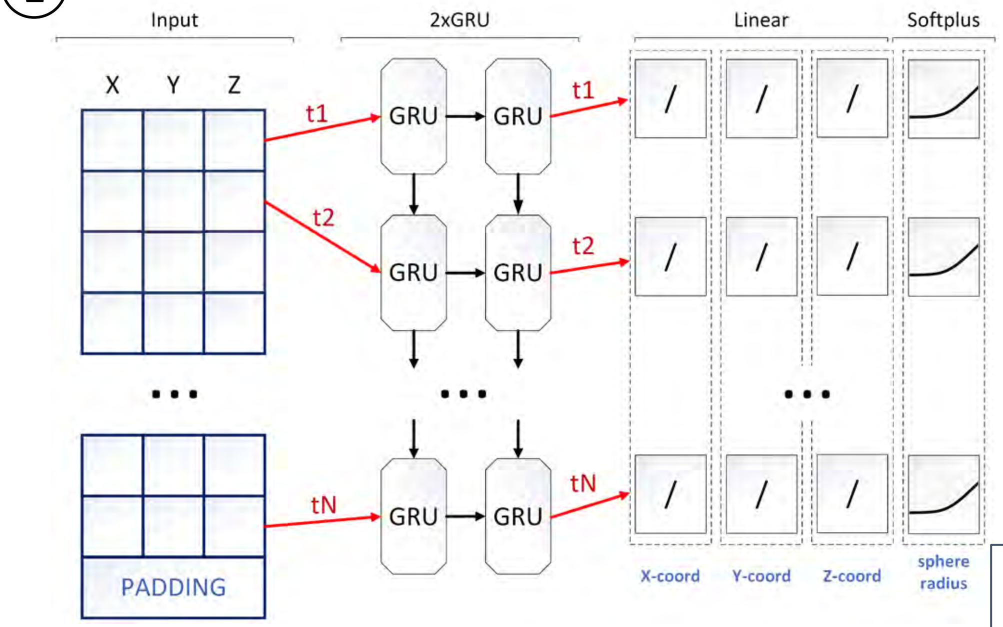
- Fast
- Lightweight
- No problems with memory consumption
- Each track can be processed separately in parallel

Cons:

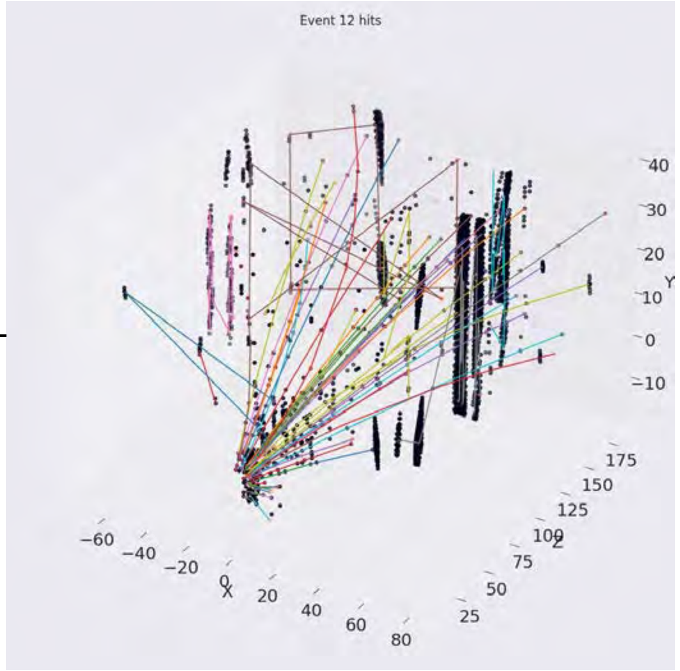
- lot of false positives or so-called ghosts, because of its local nature of prediction

Local Tracking with TrackNET. Inference

1



2 Store all event hits in the index



SEARCH INDEX

PyTorch Faiss

3 Sphere predictions (x, y, z, r)

4 Sphere centers (x, y, z)

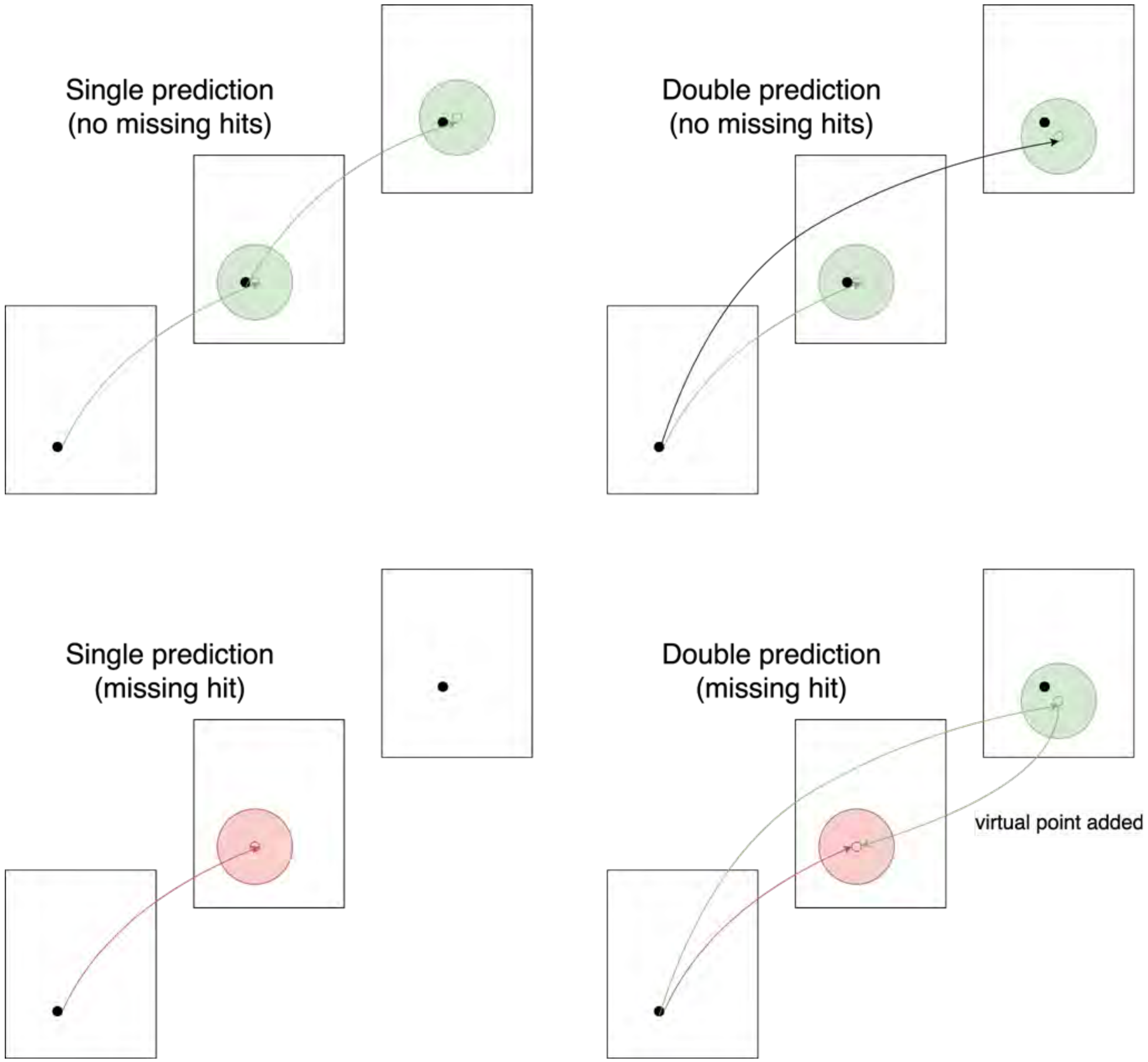
5 1 nearest hit for each sphere center

As a result
- $O(n)$ complexity depends on number of stations
- Start predictions from the 1st hit – much less ghosts

6 Check sphere attendance

7 Prolong candidates and pass them to the model

Dealing with detector inefficiency



Step ahead TrackNET:

- The network contains all necessary information on how to extrapolate a track-candidate more than one hit ahead.
- Predicts two steps ahead at once.
- If hit is not found on the first sphere, the second sphere can be checked for the hit's presence.
- If there is a hit in the second sphere, track-candidate is prolonged with a virtual point – first sphere center.
- Accuracy of the first prediction based on the single hit is lower, but model uncertainty as the sphere radius is larger, that allows model to find the next track hits.
- Virtual point placed instead of missing second hit can confuse the model for the second prediction, so tracks without hits at first sphere are saved and prolonged later with virtual point and second sphere hit together.

Testing Results. Metrics

Testing setup:

- 200 000 events (5 000 time slices) for time slice evaluation
- Xeon(R) Gold 6148 CPU @ 2.40GHz
- NVIDIA Tesla V100 32GB
- **No tracks with less than 4 hits**

Used metrics:

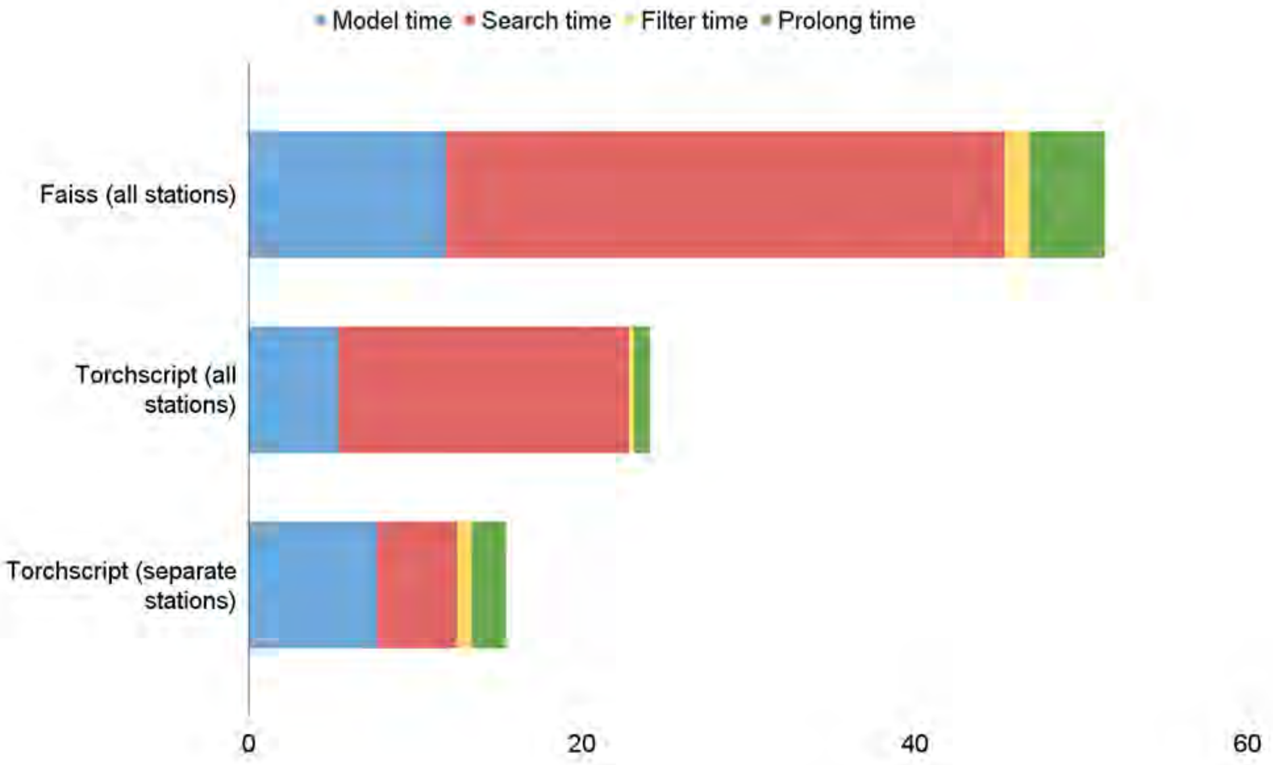
- $recall = \frac{N_{true}^{rec}}{N_{in}}$
- $precision = \frac{N_{true}^{rec}}{N^{rec}}$

- N_{true}^{rec} - no. real tracks that the network found
- N_{in} - no. all real tracks known from Monte-Carlo
- N^{rec} - no. all reconstructed tracks

	TrackNET (100% efficiency)	**Step ahead TrackNET (99% efficiency)**	**Step ahead TrackNET (98% efficiency)**
Track efficiency (recall) (%)	96,54	95,48	93,87
Track purity (precision) (%)	94,75	94,74	93,46
Processing speed (time slice / sec)	63,378	34,534	34,849
Processing speed (events / sec)	2549,52	1381,39	1393,98

Testing Results. Time measurement

- Main requirement for the approach in terms of speed is to be able to process **at least 1000 events per second**.
- It could be done by optimization of time slice processing pipeline.
- The most time consuming stages of inference are TrackNET model execution and spatial search for the nearest event hit.
- The use of sophisticated algorithms of spatial search requires significant index building time, so in case of continuous event flow brute search approach fits better.
- Spatial search was previously implemented with faiss, but showed high increase of the search time with search index size growth.
- PyTorch implementation of euclidean distance can provide much more efficient processing of large batches using GPU.
- To speed up the model inference and use the model in experiment software package, model was converted to TorchScript format.
- Reducing the index size by splitting the event hits by station can also speedup the pipeline.



Conclusion and outlook

- Applying the TrackNET model to SPD simulation data showed promising results in terms of track efficiency and purity.

- Step ahead TrackNET model can’t find track with two missing hits in a row. Such situation can be considered as very rare, since the required detector efficiency should not be less than 98% (about 1% of tracks with two missing hits in a row in this case).

- More complex simulation of SPD events is required. During the data acquisition from the straw tracker, special transformations are needed to obtain hits as spatial points. There is two main approaches: handling the left-right uncertainty and converting data to hits, or changing the model architecture to work with the signals right from the straw tracker without hit construction.

- Event disentangling is needed after the track building. It can be done by event vertex approximation and further clustering.

Particle track reconstruction at scale: online tracking with PyTorch

Rusov D., Goncharov P., Nikolskaia A., Ososkov G., Zhemchugov A.

The study was carried out with the financial support of the Russian Science Foundation No. 22-12-00109