TrackNET: Deep Learning-Based Track Recognition in Pixel and Strip-Based Particle Detectors

Pavel Goncharov (pgoncharov13@gmail.com)
Daniil Rusov, Anastasiya Nikolskaya, Gennady Ososkov, Alexey Zhemchugov

Conference on Computing in High Energy and Nuclear Physics
23 October, 2024

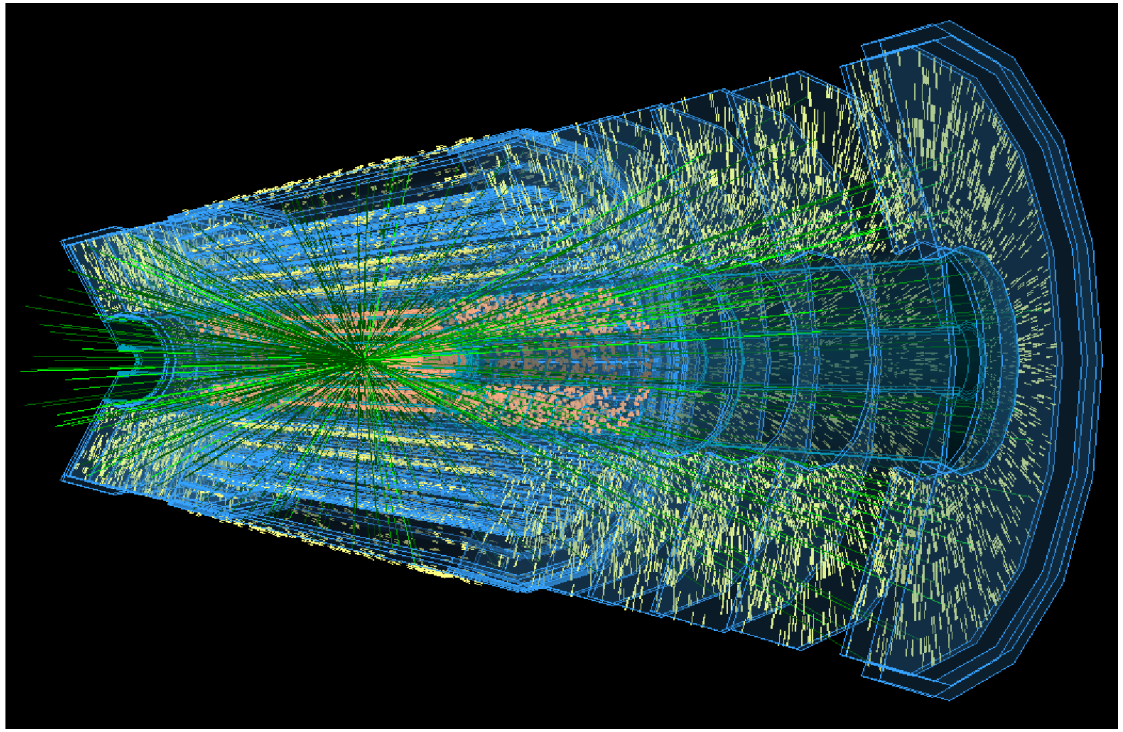
Problem Statement: The Need for Advanced Tracking Methods

Unprecedent scale of modern experiments:

- Up to 200 simultaneous proton-proton interactions is expected at High Luminosity Large Hadron Collider
- 200 particle tracks on average, 40K of tracks considering pile-up
- Traditional tracking methods struggle with dense, overlapping particle tracks due to computational complexity and time constraints

Deep Learning for Efficient Track Reconstruction:

- DL models can handle high-dimensional data and complex spatial correlations between tracks
- Coulomb scattering and inhomogeneous magnetic field effects could be learned from training data
- Effective parallelization using GPUs out of the box
- TrackML Challenge was launched to explore new scalable approaches for particles tracking



https://webific.ific.uv.es/web/en/content/taking-lhc-higher-luminosity

Deep learning-based methods have a potential to cope with immense data volumes in modern experiments

Classification of Track Reconstruction Methods

Local Tracking

Works with parts of event data (hits, track segments, detector parts).

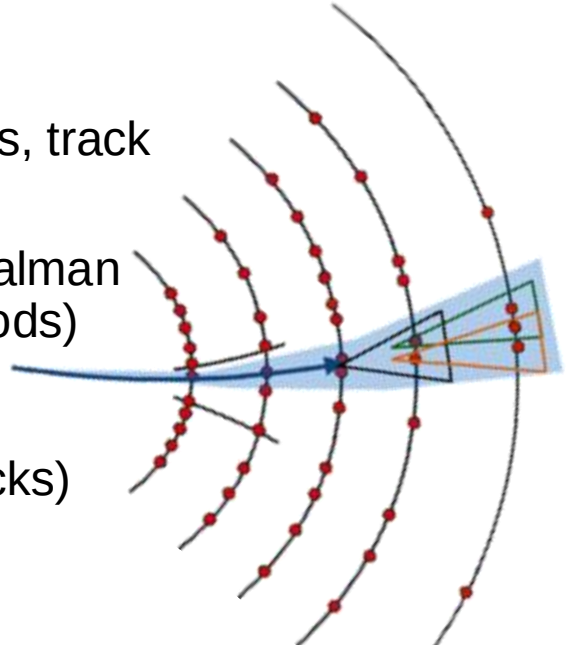
Examples: Cellular Automaton, Kalman filter (stands apart, bunch of methods)

Pros:

- High parallelism (individual tracks)
- Lightweight and fast
- Low memory use

Cons:

- Requires post-processing for full event reconstruction
- Prone to false positives (due to lack of full event view)



CNNs on FPGAs for Track Reconstruction

Global Tracking

Uses full event data for track reconstruction.

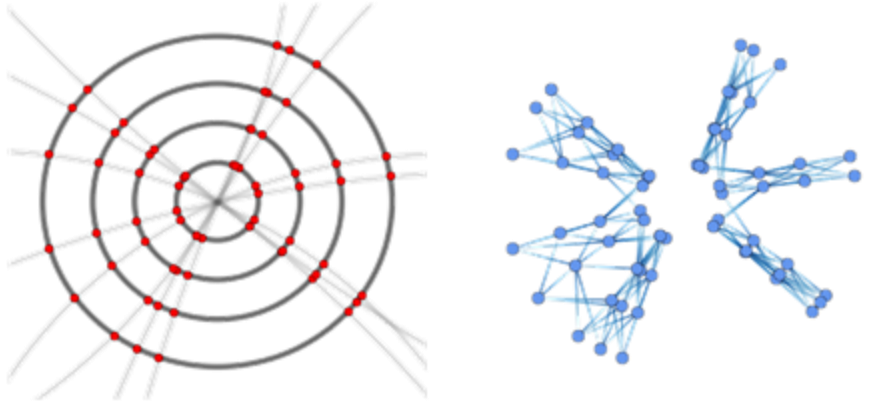
Examples: Graph Neural Networks, Hopfield network, Point Cloud Processing.

Pros:

- Higher quality metrics, fewer false positives
- Event-level parallelism possible

Cons:

- High memory requirements (entire event as input)



Graph Neural Networks in Particle Physics

Hybrid Tracking Methods

Hybrid Tracking

Combines local and global tracking methods.

Stages

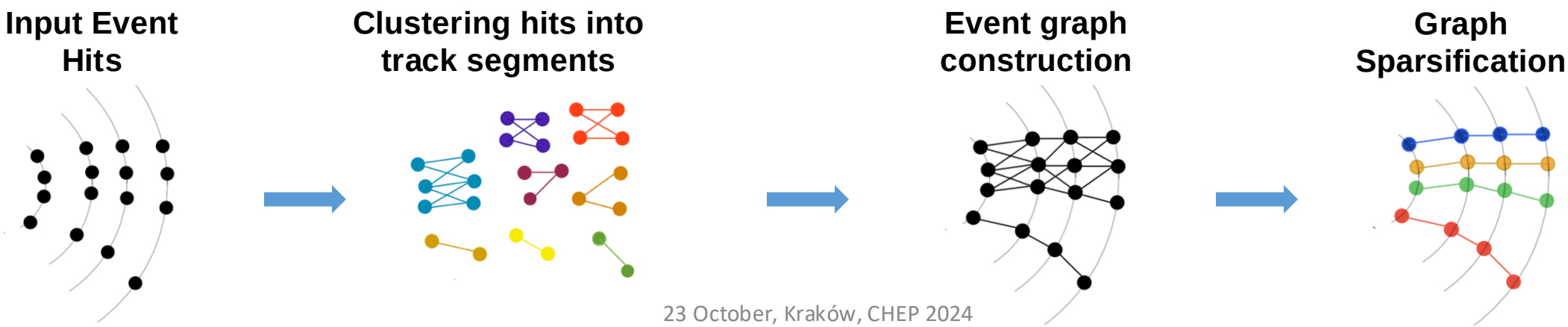
1. Track seeding, track candidates or event graph building. Main goal: high recall while reducing the number of false positives as much as possible.
2. Tracks selection either by various fitting criteria or ranking candidates using machine learning methods (e.g. graph sparsification, candidates' classification). Main goal: increase precision without recall dropping.

Pros:

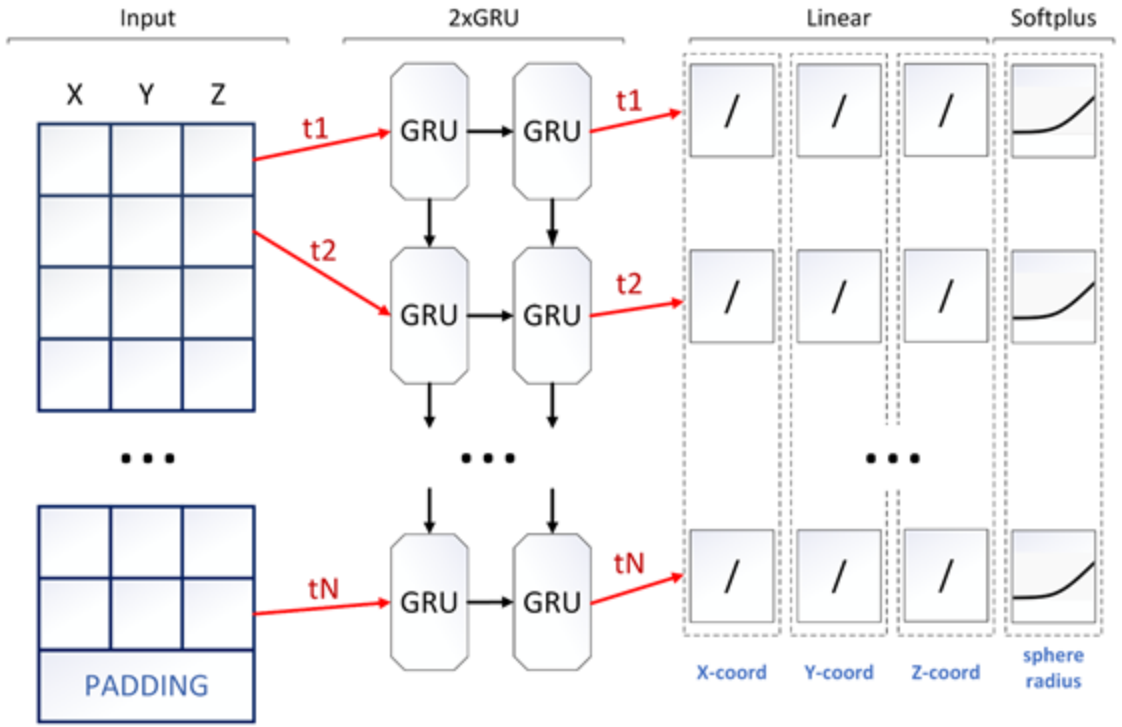
- Achieves both high performance and efficiency.

Cons:

- Errors depend on multiple models.



TrackNET as Local Tracking Method



Model Architecture

How the model works?

- Locality – one track-candidate during the inference
- The model predicts center and radius of the sphere where to search for the next hit
- All event 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:

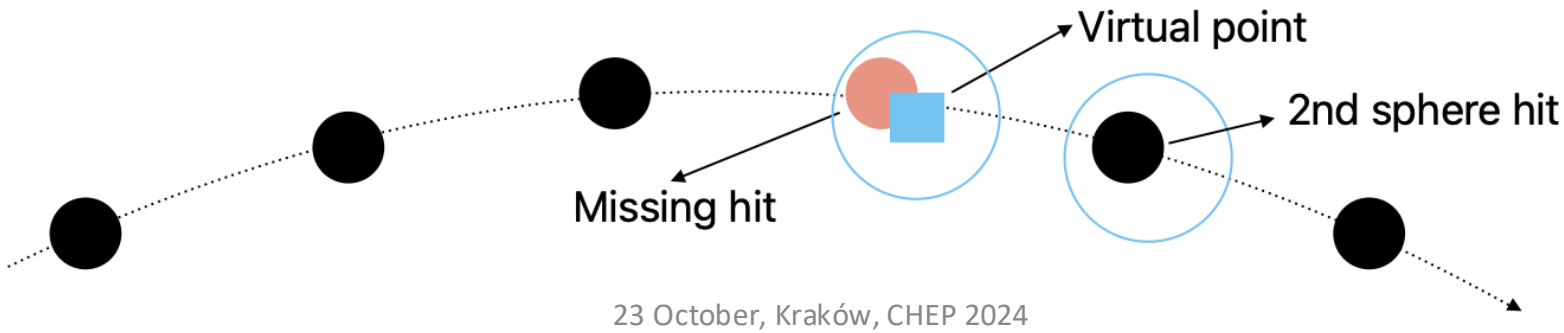
- Fast and Extremely Lightweight
- Few hyperparameters to tune – loss weights and K
- No need for seeding – prediction starts from single hit

Cons:

- Big number of false positives, because of its local nature of prediction
- Sequential inference (station by station)

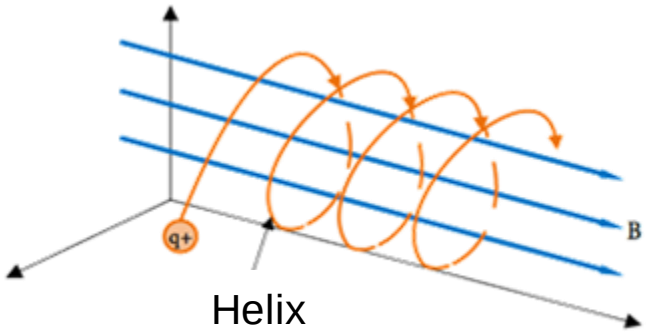
StepAhead TrackNET: Dealing with Detector Inefficiency

- The network is designed to predict the continuation of a track even when multiple hits are missing.
- It predicts two steps ahead simultaneously (covering two spheres of potential hit locations).
- If no hit is found in the 1st sphere, the 2nd sphere is checked.
- When a hit is in the 2nd sphere, the track is extended using a **virtual point** at the center of the 1st sphere.
- While predictions based on the first sphere are less accurate (due to larger uncertainty), this broader search radius helps locate the next track hits.
- However, using a virtual point in place of a missing hit **for the very first prediction** (first hit and virtual point) can introduce confusion. To mitigate this, track candidates without hits in the first sphere are temporarily saved and extended later using both the virtual point and the hit from the second sphere.

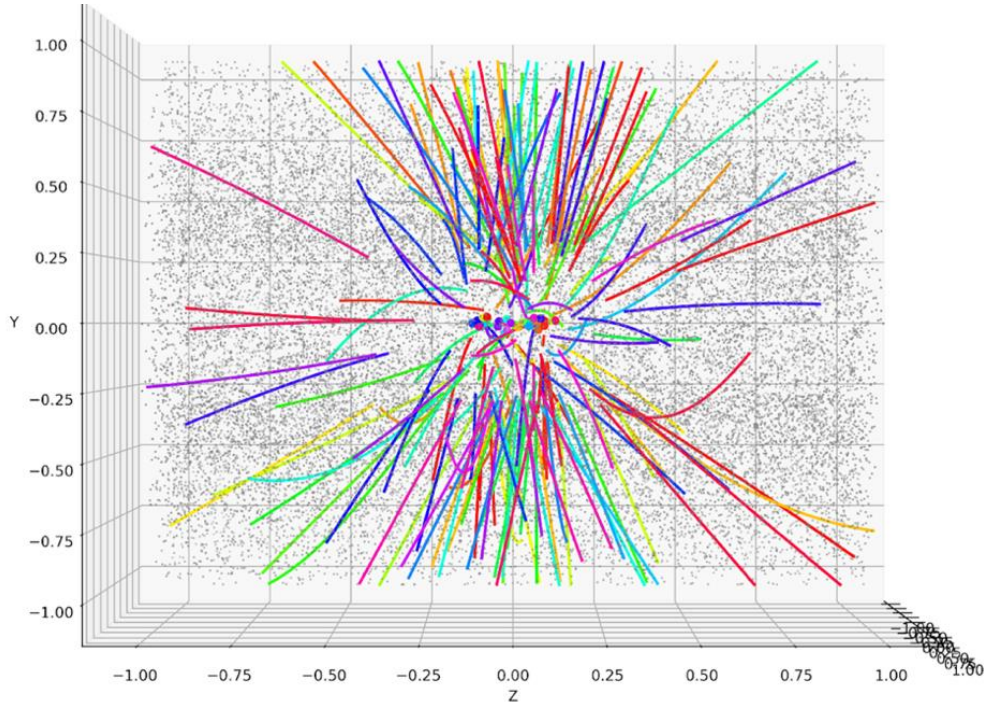


Simulated Toy Data

- Python script generates events with 1-10 random tracks.
- Transverse momentum: 100-1000 MeV/c (uniform).
- Random vertex coordinates within the collision area.
- Trajectories follow a helical path, defined by the pitch and radius equations.
- Simulated detector with 35 stations.
- Fake and noise hits simulated using randomly sampled points in detector space.
- Pile-up modeled by creating time slices of 40 events.



Helix



Statistics

- Around 200 tracks per time slice.
- ~1,100 hits per station (total: ~38,200 hits).
- 82.26% of hits are fake.

Toy Dataset Results

Testing setup:

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

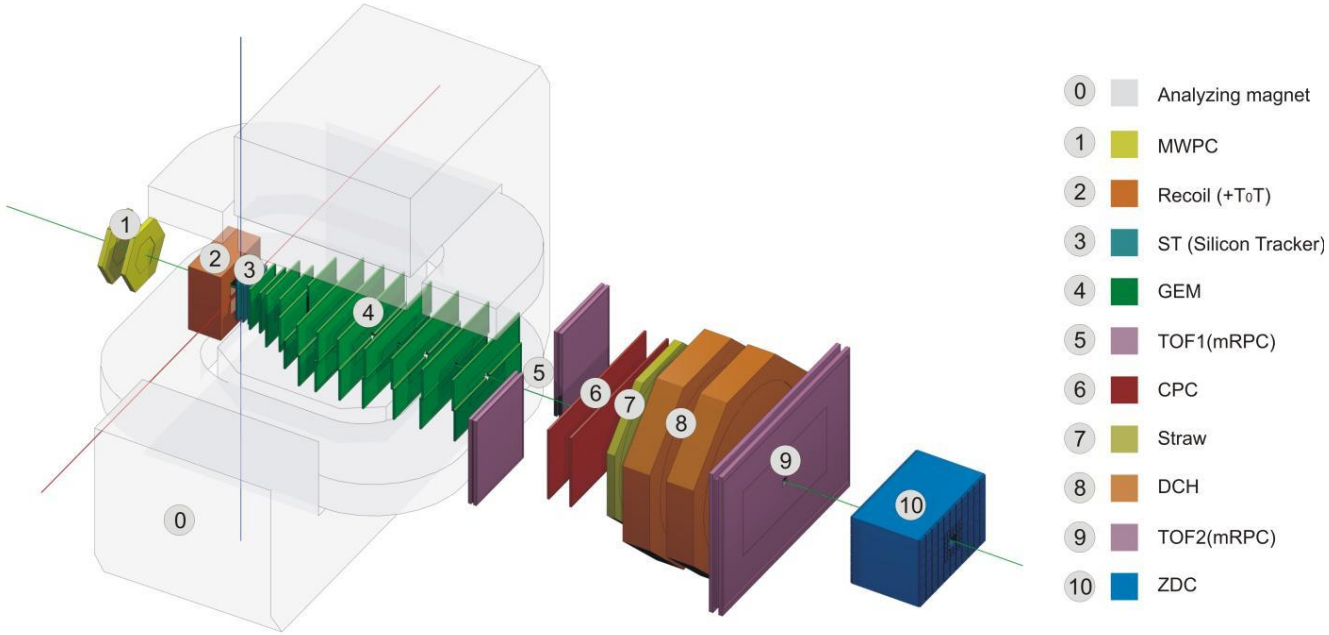
Used Metrics:

Recall = $\frac{N_{true}^{rec}}{Ntrue}$; Precision = $\frac{N_{true}^{rec}}{Nrec}$

- N_{true}^{rec} - No. of correctly reconstructed true tracks.
- $Ntrue$ – No. of correctly reconstructed true tracks.
- $Nrec$ – Total number of reconstructed tracks.

Model	**TrackNET**	**StepAhead TrackNET**	
Detector Efficiency (%)	**100**	**99**	**98**
Recall (%)	96,54	95,48	93,87
Precision (%)	94,75	94,74	93,46
Speed (time slice / sec)	63,378	34,534	34,849
Speed (events / sec)	2549,52	1381,39	1393,98

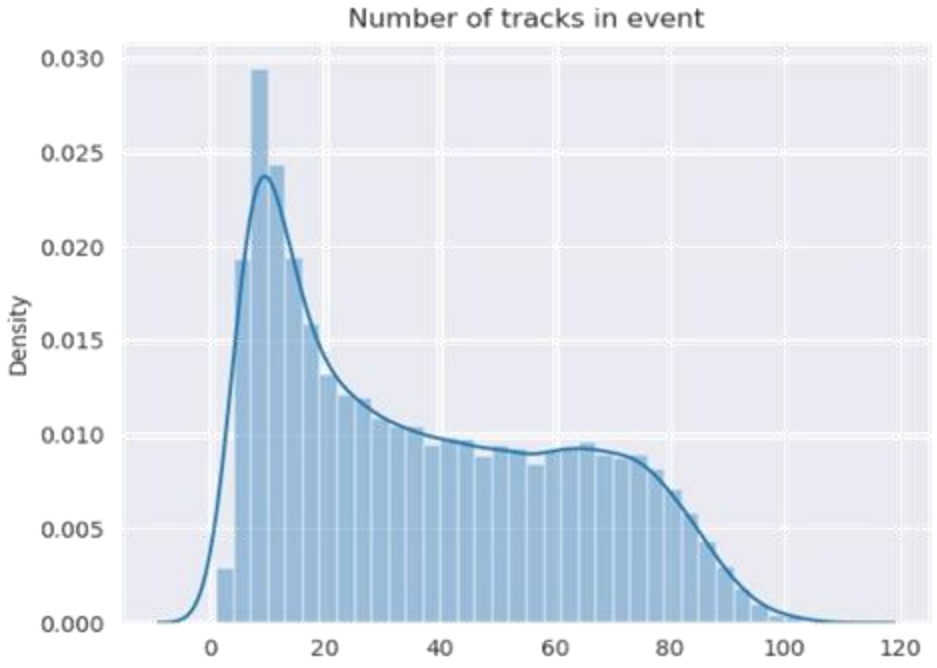
Simulation of BM@N Run 7 Data



- BM@N is a fix target experiment at NICA, JINR
- Tracking detector consists of the two main parts: silicon tracker (ST) and GEM detector
- ST has 3 stations, GEM – 6 stations

Monte-Carlo Simulation:

- 1m Ar+Pb events with 3.2 GeV (LAQGSM generator)
- Multiplicity up to 100, mean – 37 tracks
- Number of hits can be more than 500 per station
- ~68% of all hits are fakes (strip detectors specifics)



BM@N Run 7 Simulation Results

Testing setup:

- Two test sets – without missing hits ("**easy**"), and with inefficiency ("**hard**"), 25K events each
- Xeon(R) Gold 6148 CPU @ 2.40GHz + NVIDIA Tesla V100
- **Tracks with less than 4 hits and spinning tracks are removed**

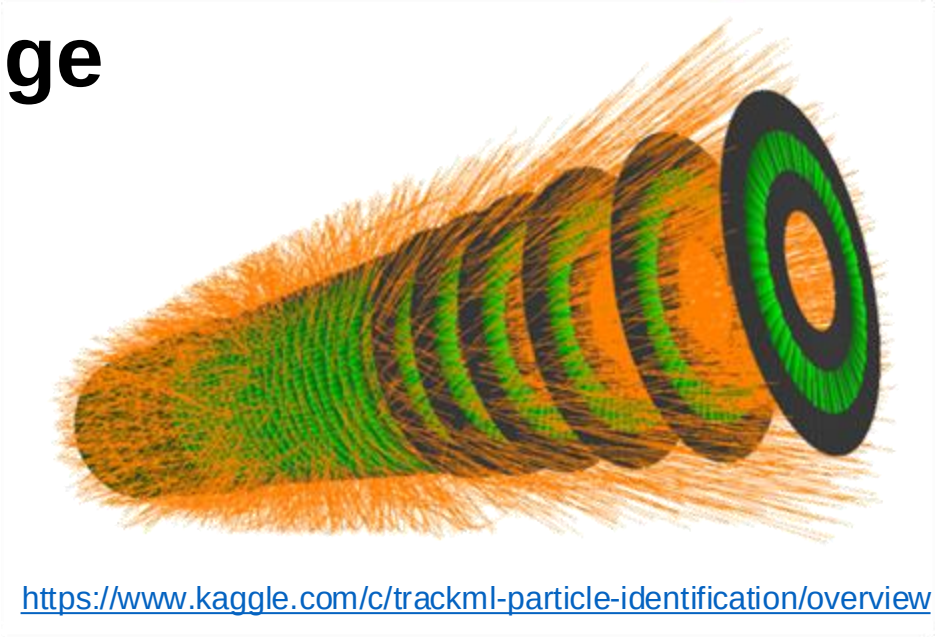
Details:

- **Easy data:** TrackNET outputs were fed to Graph Neural Network
- **Hard data:** Optimized inference with Torchscript and KNN search on GPU
- After graduation I lost access to the data, so the work is unfinished

Model	**Two-stage tracking (TrackNET + GNN)**	**Step Ahead TrackNET (optimized inference)**
Dataset	easy	**hard**
Recall (%)	94,70	88,23
Precision (%)	77,06	0,16
Speed (events / sec)	7,5	48,16

TrackML Particle Tracking Challenge

- 100 GB of simulated data encompassing around 10,000 events
- 10000 tracks per event on average
- Each track has about 10 hits – 100000 signals in one event
- Straight-line tracks (high momentum) are rare and has more weight in competition scoring function



https://www.kaggle.com/c/trackml-particle-identification/overview

Noticeable participants:

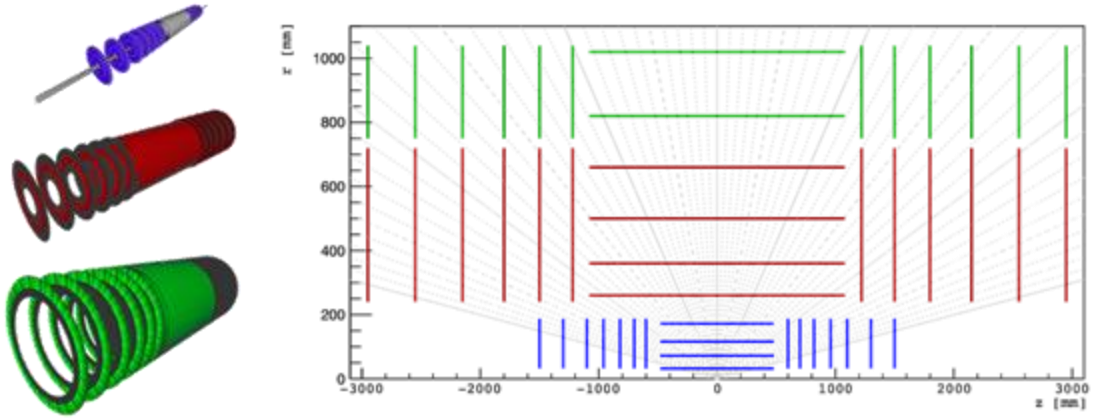
- 1st: top-quarks – Logistic regression for pairs and triplets, helix extrapolation (8 min/event).
- 2nd: outrunner – Dense NN for pair prediction, circle fitting (3+ hrs/event).
- 3rd: Sergey Gorbunov – Triplet seeds, helix fit with magnetic field estimation (0.56 sec/event).
- 9th: CPMP – DBSCAN clustering, filtered by module frequency (10 hrs/event, 30,000+ DBSCAN runs).
- 12th: Finnies – DBSCAN seeding, **LSTM for predicting next 5 hits** (slow, no speed given).

Most of the solutions repeat the classical pipeline for tracking – seeding followed by trajectory fitting.

TrackNET Training Overview on TrackML Dataset

Space shrinking/compression:

- for external detectors, the distance between layers is ~2 times larger than for internal ones. Shrink the space of external detectors by factor two.



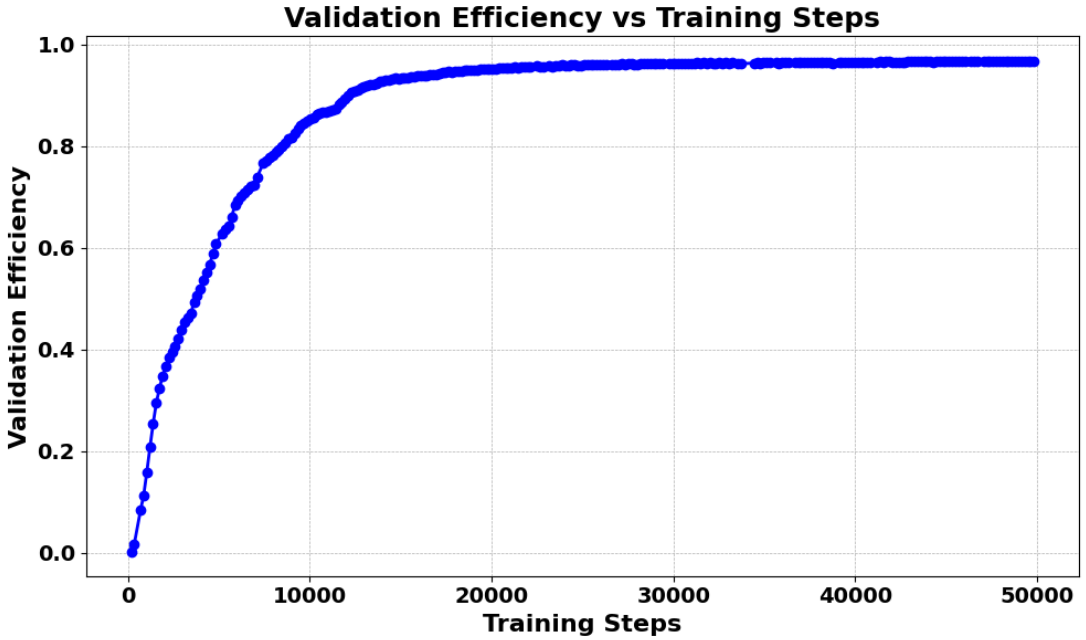
[The Tracking Machine Learning challenge: Accuracy phase]

Multiple hits on the same layer during training due to modules intersections:

- take the closest one (least r in cylindrical coordinates)

Picking seeds:

- taking all hits from the innermost layers



Training

- 10 mln tracks, 300 epochs
- 15 hours on single Nvidia V100 32GB
- weight in TrackNET loss alpha = 0.9999 because of unnormalized coords and large detector

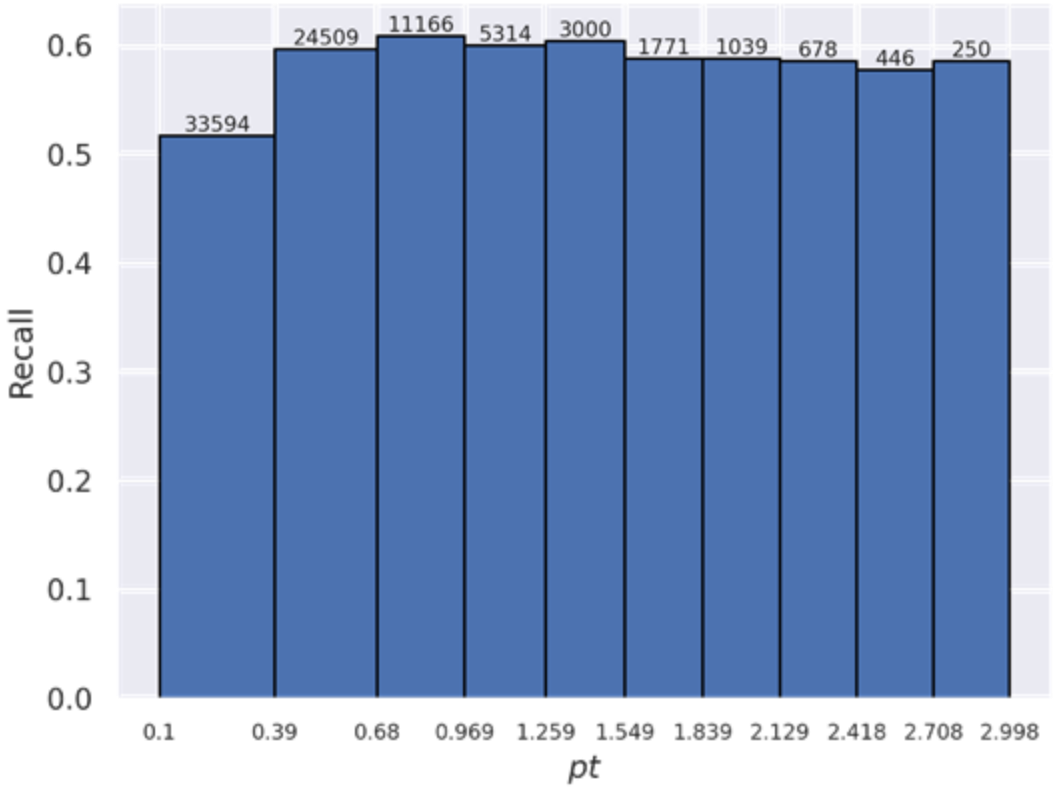
TrackML Evaluation Results (work in progress)

Testing setup:

- Xeon(R) Gold 6148 CPU @ 2.40GHz (only CPU, optimization of memory operations is required)
- **No data reduction** based either on particle momentum (no pt cut) or number of events (original pile-up)
- No specific tuning for the TrackML scoring metric – **considering all tracks with equal importance**
- Following TrackML metric, a track is considered fully reconstructed if >50% of hits were recognized correctly
- In case of duplicate track candidates, **only one is included in the final metrics**

K searched hits	1	2
Recall (%)	35,15	56,41
Precision (%)	33,47	1,01
Event processing time (sec)	6,4289	45,2144

Statistics on 10 events (100K tracks)



Notes:

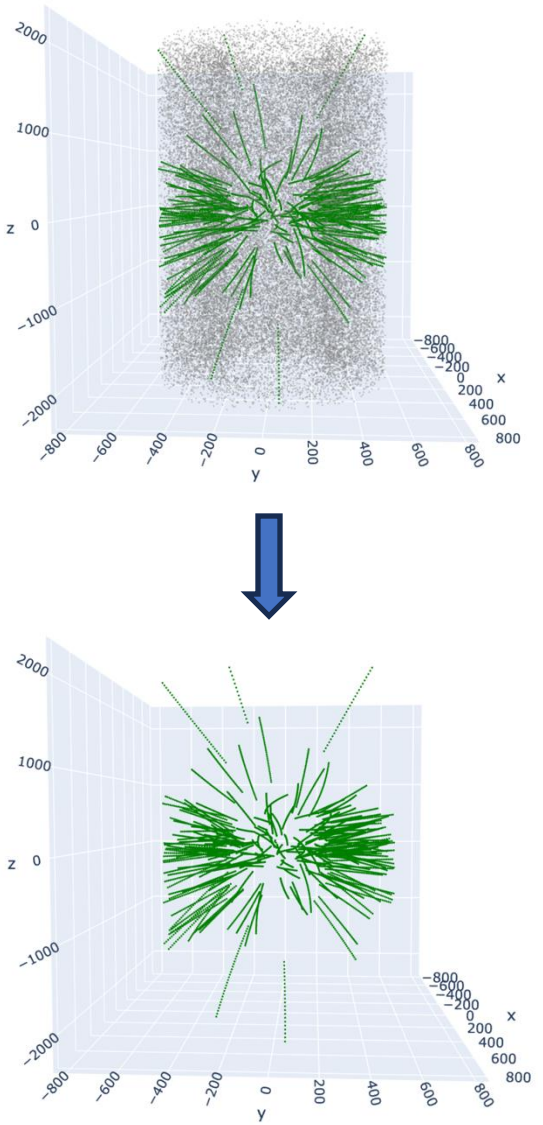
- Building precision vs pt plot requires momentum estimation for the track-candidates
- I've found a bug in preprocessing, didn't have enough time to retrain the model, but the results may be better

Conclusion

- The TrackNET model demonstrates high performance even in challenging, noisy environments (e.g., handling fake hits in microstrip detectors).
- Due to the model's local nature, a second stage of track-candidate selection is necessary to further improve precision.
- The model's first application to the TrackML dataset has been successfully conducted.
- Further work is needed to enhance performance: tuning to the TrackML scoring metric, applying data reductions, and balancing training samples based on relevance.
- Without any optimizations, the model processes an average TrackML event in approximately 7 seconds on a single CPU (K=1).
- The code of TrackNET application to TrackML data will be open-source soon.

One end-to-end model ~~to rule them all~~ for tracking?

- Graph-based event representation is not compact; segments increase rapidly with pile-up.
- Point-cloud representation offers better scalability, allowing models to learn signal relations (e.g., attention).
- Point-clouds enable unsupervised pretraining on experimental data (e.g., predicting hits in selected areas like masked language modelling).
- Using bare hit coordinates causes information loss; it's better to work with raw data (e.g. fakes after signal digitalization in microstrip detectors).
- Transformer models are data-agnostic, allowing direct use of raw detector data.
- Idea: Divide the detector space into voxels, process each voxel with an encoder (e.g., PointNet), then apply multi-head attention on patch representations for tracking.
- Potential training objectives: instance segmentation, track parameter prediction, or clustering via learned hit vector representations.
- Example: Point Cloud Transformers have been successfully applied for signal segmentation.



TrackNET: Deep Learning-Based Track Recognition in Pixel and Strip-Based Particle Detectors

Pavel Goncharov (pgoncharov13@gmail.com)
Daniil Rusov, Anastasiya Nikolskaya, Gennady Ososkov, Alexey Zhemchugov

Conference on Computing in High Energy and Nuclear Physics
23 October, 2024