Track Reconstruction on the TrackML Detector with Monte Carlo Tree Search

M. Zhao, J. Wagner, H. Gray

Department of Physics, University of California, Berkeley

Berkeley Physics

Abstract

In high energy experiments, track reconstruction is the computational problem of reconstructing charged particle trajectories from hits left in a detector. As accelerators are upgraded, the integration of machine learning techniques into track reconstruction algorithms is an active research area. We would like an algorithm that preserves the track following structure of the Combinatorial Kalman Filter in conjunction with neural network components. To that end, we are developing a novel track reconstruction algorithm based on the Monte Carlo Tree Search, which has found success in parsing similarly intractable search spaces. This summer we developed and trained the neural network components for an implementation on the TrackML detector, which would allow us to benchmark performance against existing algorithms.

Track Reconstruction

In modern detectors, the first observation of particles emanating from a collision vertex is when they pass through sensitive silicon layers and deposit energy. The spatial locations of these interactions are detector hits. To convert this into useful information, the hits need to be associated to particle trajectories which is the process of track reconstruction.

Track parameters define the track's shape and orientation. A strong magnetic field is present within the detector, so the curvature yields information about the particles' charge and momentum. The tracks are also used as input into higher level analyses such as associating particles to calorimeter measurements.

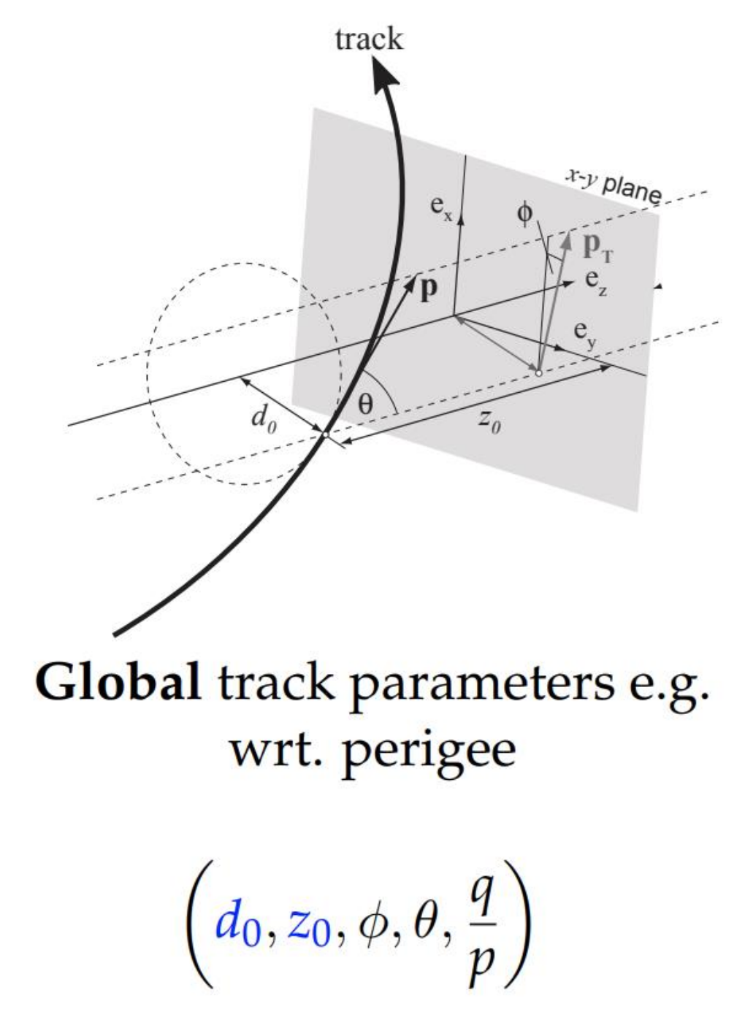

track
x-y plane
p
d_0
z_0
θ
Global track parameters e.g. wrt. perigee
$\left(d_0, z_0, \phi, \theta, \frac{q}{p}\right)$

Fig 1. Track parameters defined in ATLAS [1]

Current Tracking Algorithms

The current state of the art for track reconstruction is the Combinatorial Kalman Filter (CKF). The CKF starts with track seeds and expands them into full track candidates by propagating them through the detector with Kalman filtering. If at a step multiple compatible hits are found, they are all considered simultaneously by branching.

The Kalman filters uncertainty reducing properties make the CKF very accurate. However, it is superlinear with the number of hits due to the need for branching and requires computationally expensive numerical differential equations solving for the Kalman filters.

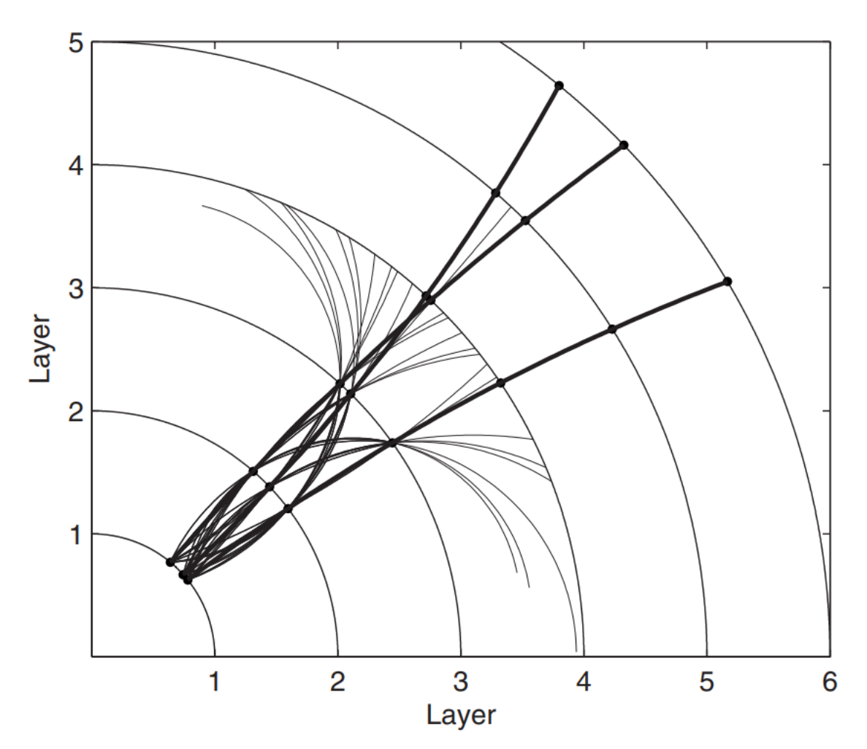


Fig 2. Multiple possible track candidates being entertained simultaneously [2]

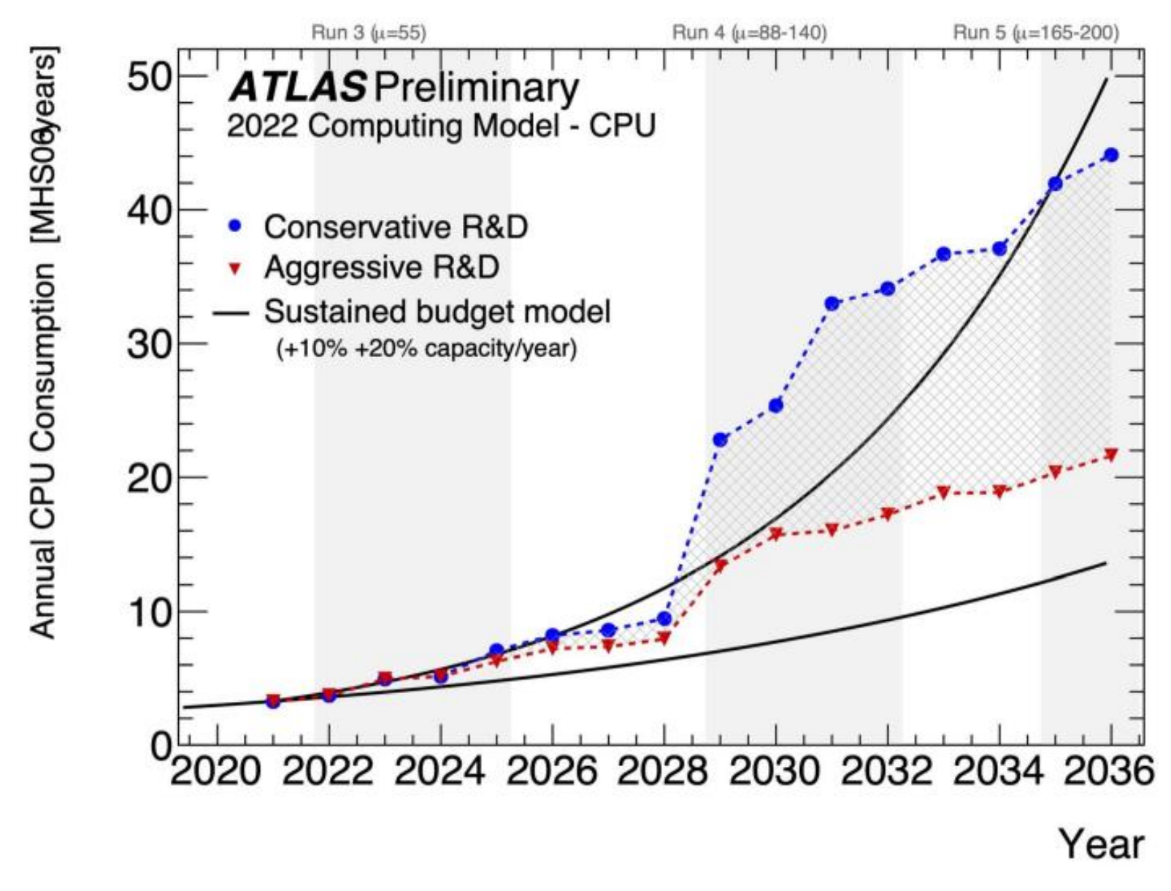


Fig 3. Projected CPU consumption for ATLAS analysis relative to projected resources [3]

GPU and TPU hardware capability has improved significantly in recent years. There are active research efforts into parallelizing components of the CKF on GPUs. Machine learning techniques are also attractive for their pattern recognition abilities. A goal of current research is a track algorithm that integrates domain knowledge with machine learning techniques.

Monte Carlo Tree Search

Our novel approach to tracking is based on the Monte Carlo Tree Search (MCTS), a heuristic search algorithm that uses random playouts to converge to optimal decision making [4]. The basic principle is that each track candidate is an agent which optimizes its own quality. Outside of tracking, MCTS has been integrated with neural networks to achieve success in playing strategy games, notably with its implementation in AlphaGo [5].

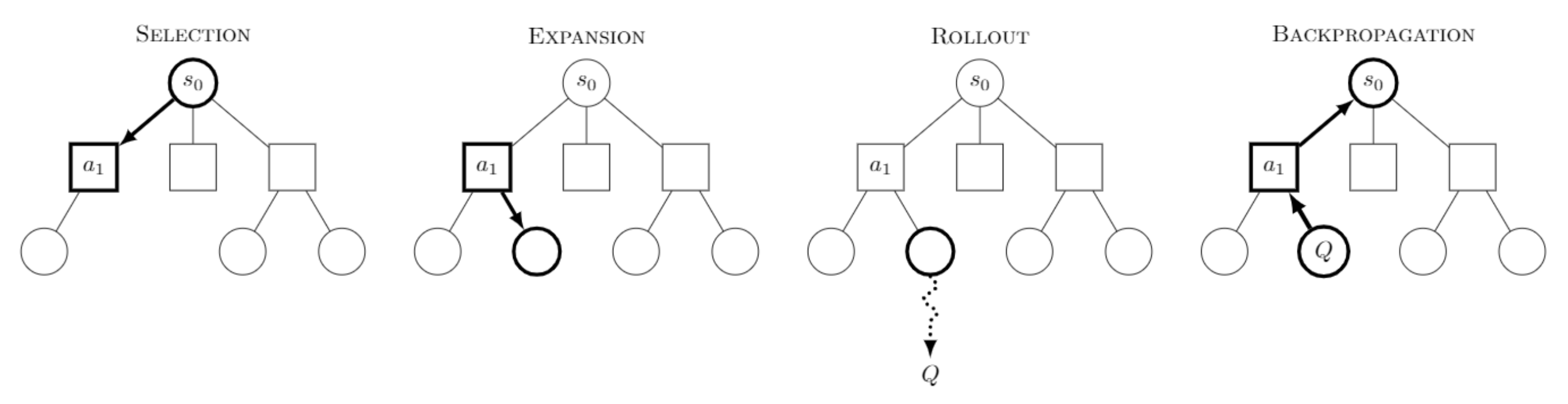


Fig 4. Visualization of the different stages of each iteration in MCTS [6]

The outline of the MCTS in our context of tracking is as follows. For each iteration:

1. **Selection:** Traverse the decision tree to find a promising unexplored hit using the current weights of the tree.
2. **Expansion:** Expand the tree at that hit to all plausible following hits. Then imbue the edges to each with a prior probability from a Policy network.
3. **Playout:** From that same hit, execute a random playout: sampling repeatedly sampling over the Policy network output until you build a complete track candidate
4. **Backpropagation:** Evaluate this completed track candidate with an Evaluation network and go back to update all visited edges with the new information.

This approach preserves the branching track following structure of the CKF, but how it chooses to traverse the tree is guided by machine learning components. The MCTS is very conservative in the variations it explores, only expanding the tree in promising direction. This allows it to parse intractable search spaces which in this case is a dense hit environment.

TrackML Detector

This algorithm has been previously demonstrated on a 9-layer telescope detector with 8 muon events. It showed high performance, but the problem is far simpler than a modern experiment [7]. To get a more accurate benchmark of the efficacy of our approach, we are seeking an implementation on the TrackML Detector. The TrackML detector is a simulated detector that was used for the TrackML challenge. It has similar geometry to the ATLAS and CMS detectors, and the associated dataset has become common for algorithms to benchmark with [8].

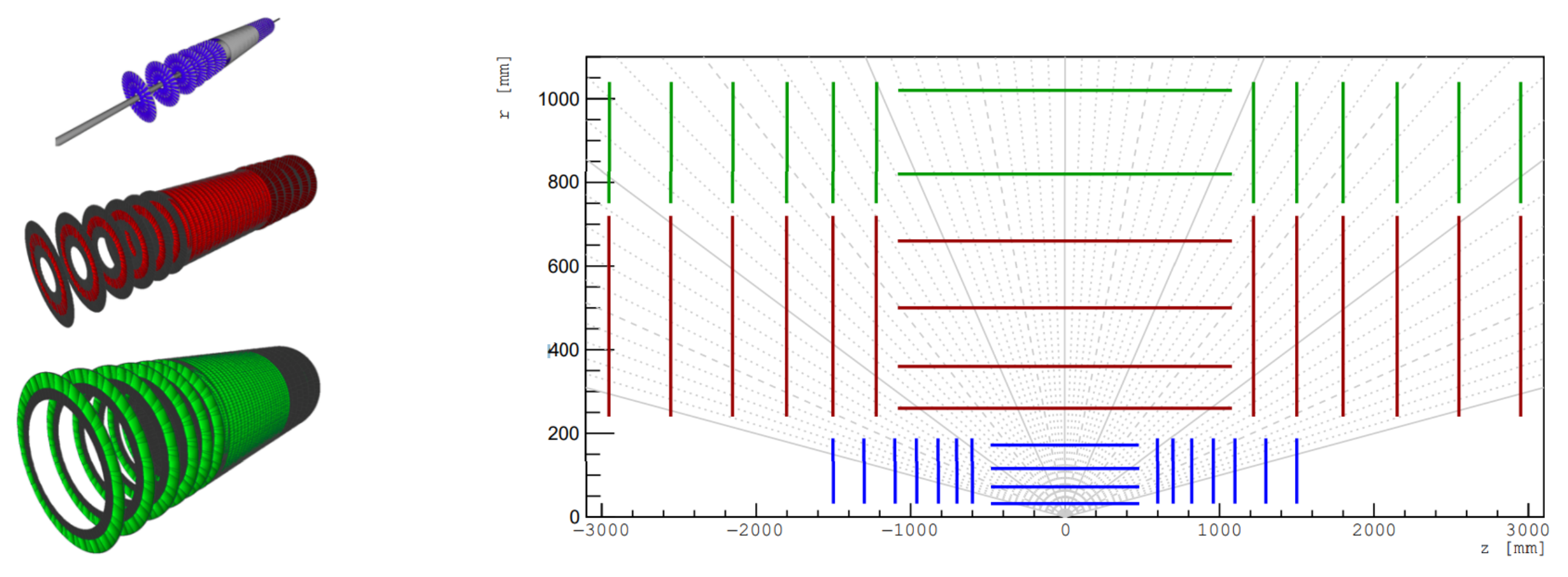
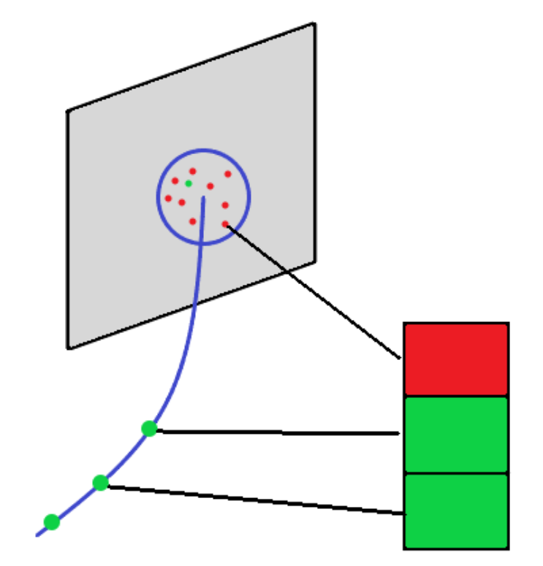


Fig 5. Layout of the TrackML detector. It includes volumes of barrel shaped layers and orthogonal endcaps like ATLAS and CMS.

Moving to the TrackML presents unique challenges, mainly for the training of the Policy and Evaluation networks. For example there is no clear hierarchy of layers, so selecting which hits should come next is a non-trivial task. We need to develop training pipelines for the two networks that produce representative training data without being combinatorially explosive.

Policy Network

The primary effort and achievement this summer was developing the training pipeline for the Policy network. The current implementation is a dense neural network that takes as input the last two hits in the current track candidate and a plausible hit. The output is a value [0,1] for how likely the plausible hit belongs to the same particle as the first two. As such, the training data needs to be triplets of hits with the first two being from the same particle and the last one plausible.



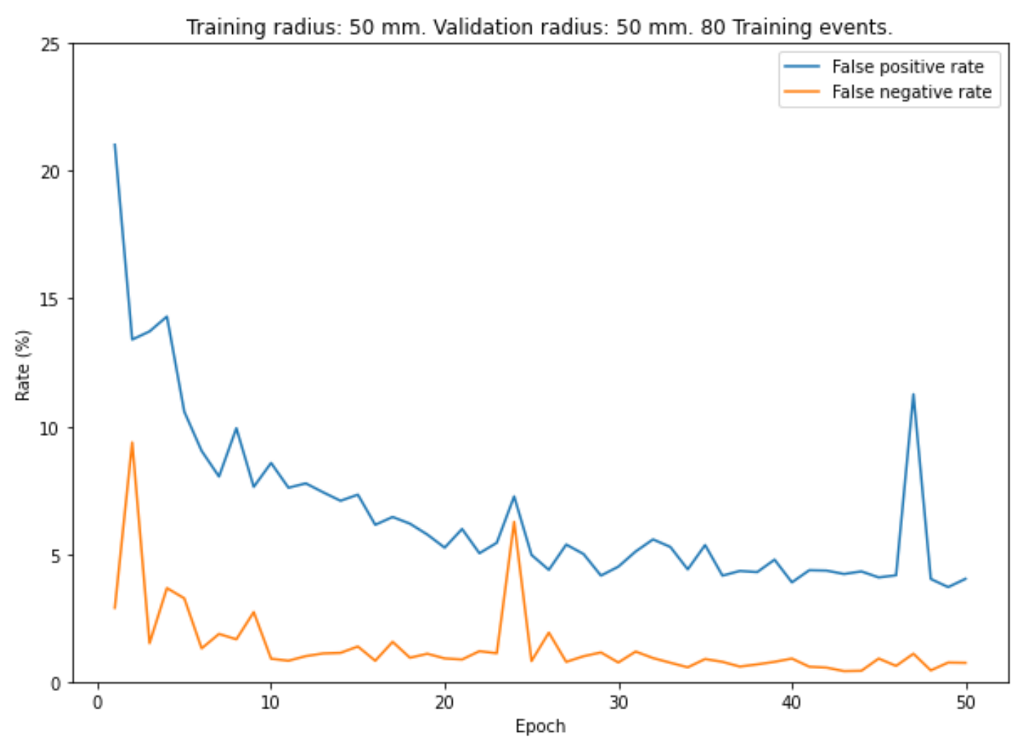

Training radius: 50 mm. Validation radius: 50 mm. 80 Training events.
False positive rate
False negative rate
Rate (%)
Epoch

Fig 6. False positive and false negative rate decrease over many epochs

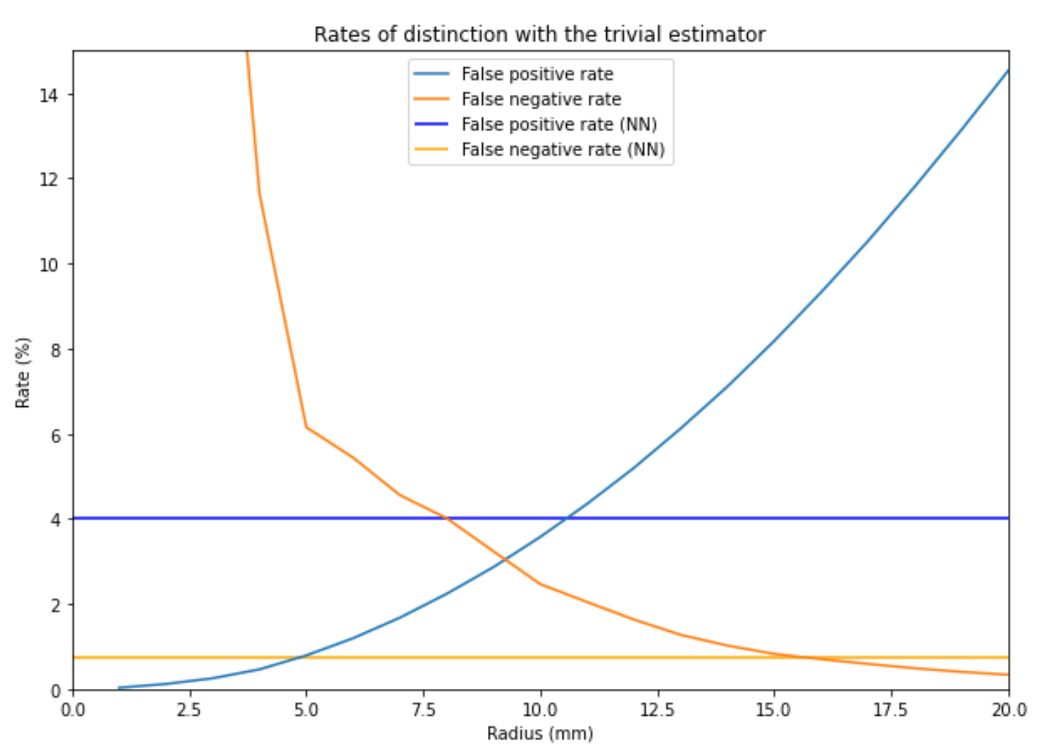


Fig 7. Performance of the neural network compared to the trivial estimator

The current training data pipeline performs simple truth tracking. At each step, we fit a helix to the last 3 hits of our track and gather all hits near the next helix intersection as plausible. If a matching hit is found among these, then it is appended to the current track and it repeats.

The total amount of data we get depends on how wide the search radius is around the helix intersection. For reasonable radii from 30-150 mm we see that training converges and the network is able to learn the problem.

We also compare the neural network performance to a trivial estimator of simply selecting hits closest to the helix intersection, and we see that the neural network performs better. This shows that the network can capture non-trivial information about the detector.

Evaluation Network

The current working implementation of the Evaluation network is a model with a transformer encoder that takes in as input a complete track candidate, and outputs the probability that each hit in the track belongs to the same particle as its neighbors.

The network is not yet trained, but the pipeline has been mostly developed. The network is to be trained on completed track candidates from the CKF output, so as to remain independent of the Policy network.

Acknowledgement

I would like to thank Johannes Wagner, Prof. Heather Gray, and the ACTS tracking researchers at Berkeley for their expertise and mentorship. I also thank MPS for supporting the project this summer. This research is performed with the computational resources of NERSC

References

[1] "ATLAS Track Reconstruction -- General Overview" *ATLAS Software Documentation*, 2021.

[2] Strandlie, A., Frühwirth, R. "Track and vertex reconstruction: From classical to adaptive methods" *Reviews of Modern Physics*, 2010.

[3] ATLAS Collaboration "ATLAS Software and Computing HL-LHC Roadmap" CERN, 2022.

[4] Coulom, R. "Efficient selectivity and backup operators in Monte-Carlo tree search" *Proceedings Computers and Games*, 2006.

[5] Silver, D., Huang, A., Maddison, C. et al. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 484–489 (2016). https://doi.org/10.1038/nature16961

[6] By Rmoss92 - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=111182752

[7] Zhao, M. "Charged Particle Track Finding with MCTS" IRIS-HEP Fellows Presentations, 2022.

[8] Amrouche, S., et al. "The Tracking Machine Learning challenge : Accuracy phase" arXiv preprint arXiv:1904.06778, 2019.