Using Machine Learning to Speed up ATLAS Tracking

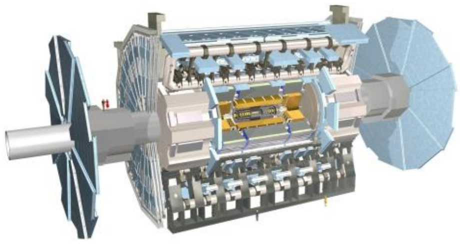
VIKTORIA ERDI-KRAUSZ

15/08/23

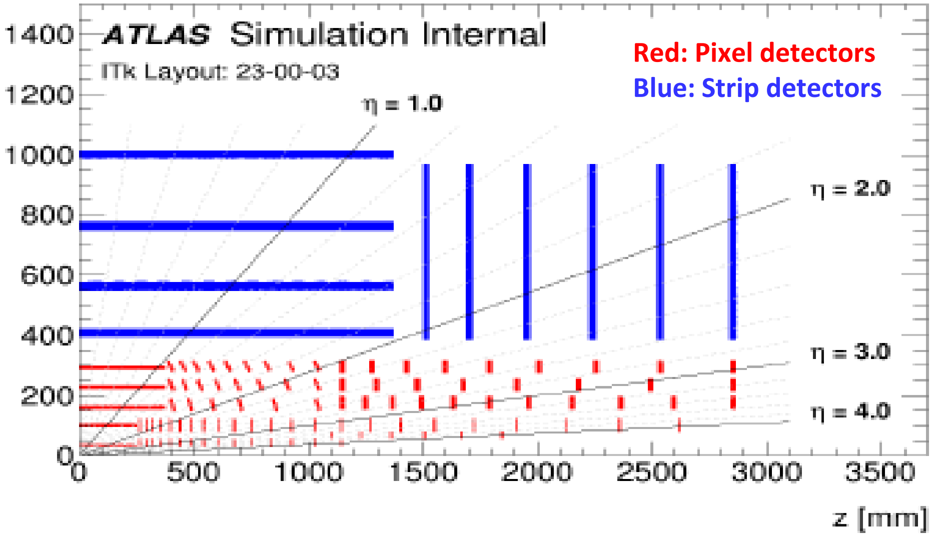
Agenda

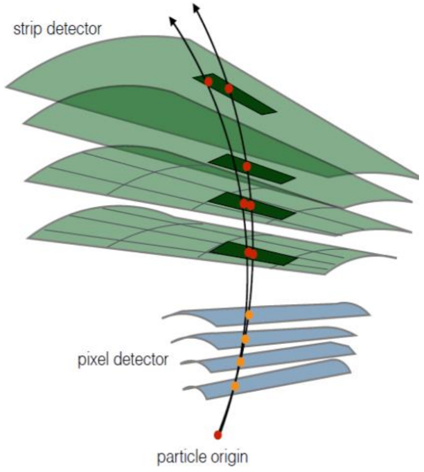
1. Background
   - ATLAS and ITk
   - FastTrackFinder
2. Project
   - Aims and Overview
   - Introduction to Machine Learning
3. Data Exploration
4. Model Development
5. Results
6. Implementations and Next Steps

Background



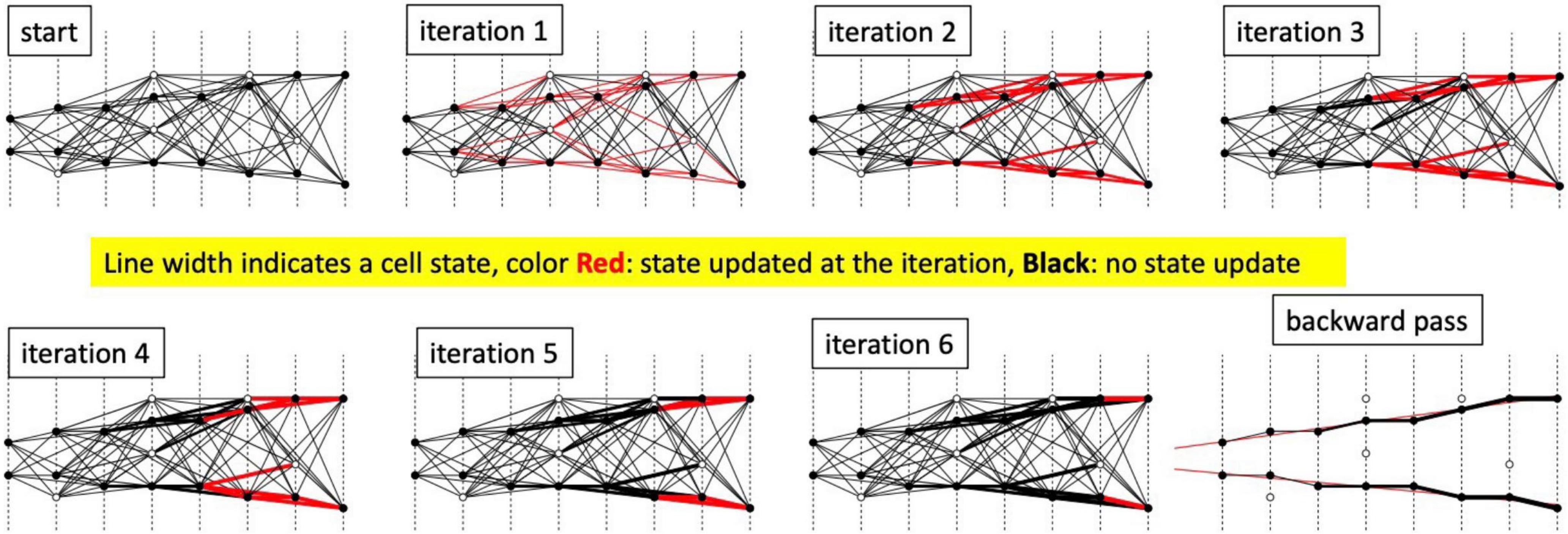
ITk Pixel Detector





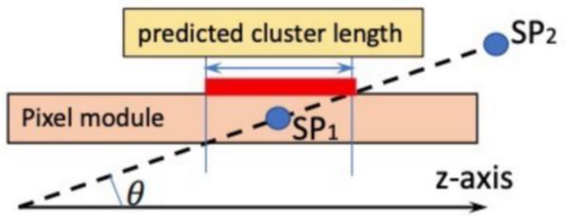
1. ATLAS inner tracker replaced by all silicon ITk detector.
   - Inclined disks
   - More forward coverage
2. Collisions occur at origin and where particles hit the detectors are connected to create tracks.
   - Tracking starts in pixel detector, later extended to strip detector.
   - We only focus on doublets in the pixel detector.

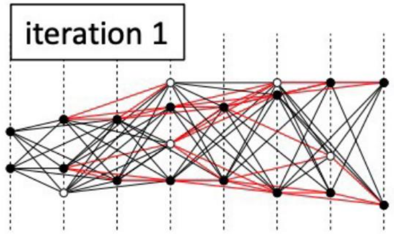
Background


start
iteration 1
iteration 2
iteration 3
Line width indicates a cell state, color **Red**: state updated at the iteration, **Black**: no state update
iteration 4
iteration 5
iteration 6
backward pass

Project Overview

1. Aims
   - Create machine learning model to speed up tracking.
2. Overview
   - Use track angle and cluster width to reduce number of possible doublets.

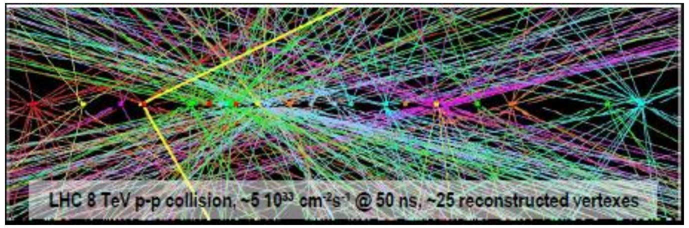


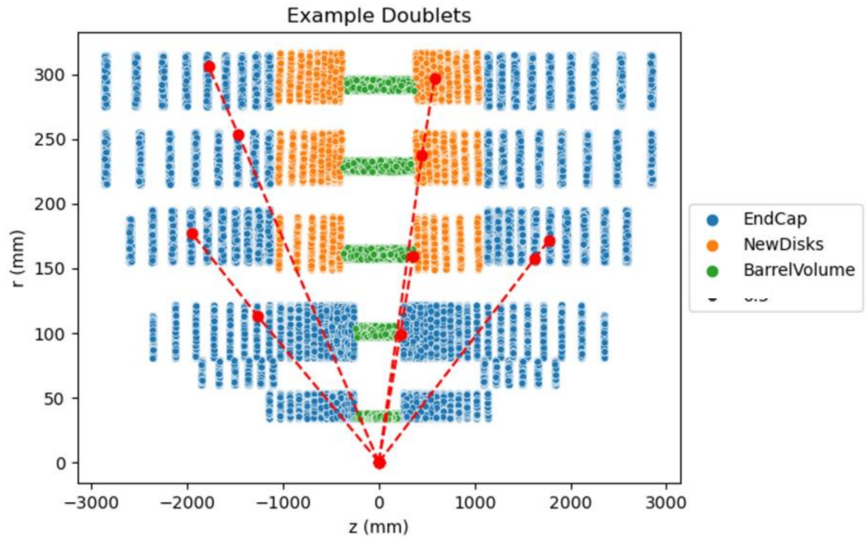


3. Random Forrest
   - Type of machine learning model which uses a number of decision trees to identify trends in dataset.

Data Exploration

1. Data used: TT bar with 200 pileup
   - Signal: any particle produced from the physics interaction.
   - Background: Low momentum tracks.
2. Issues with large data quantity – for 1 event:
   - 2380 signals
   - 10,091,380 background
3. Data must be reduced to workable volume whilst staying representative of each region.
4. Data extracted for 55 events and down sampling was implemented leaving:
   - 375,034 signals
   - 750,068 background
5. Different trends observed in each part of the detector i.e. barrel, endcap, inclined.





Model Development

1. Model was trained to maximise efficiency, not purity.
2. Efficiency (/recall) is the number of points correctly classified:

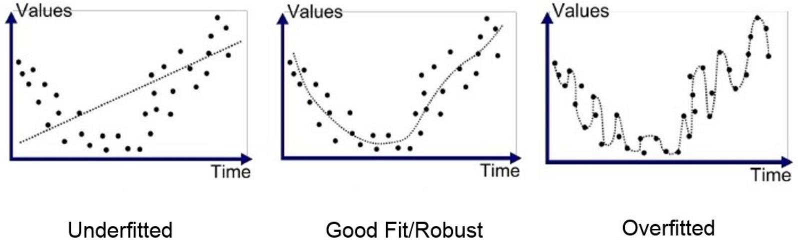
$$\text{Recall} = \frac{TP}{TP + FN}$$

   - Signal recall is the number of signals correctly identified (i.e. (**PREDICTED** signal and **IS** signal) / ((**PREDICTED** signal and **IS** signal)&(**PREDICTED** background and **IS** signal)))
   - Background recall is the number of background correctly identified (i.e. (**PREDICTED** background and **IS** background) / ((**PREDICTED** background and **IS** background)&(**PREDICTED** signal and **IS** background)))

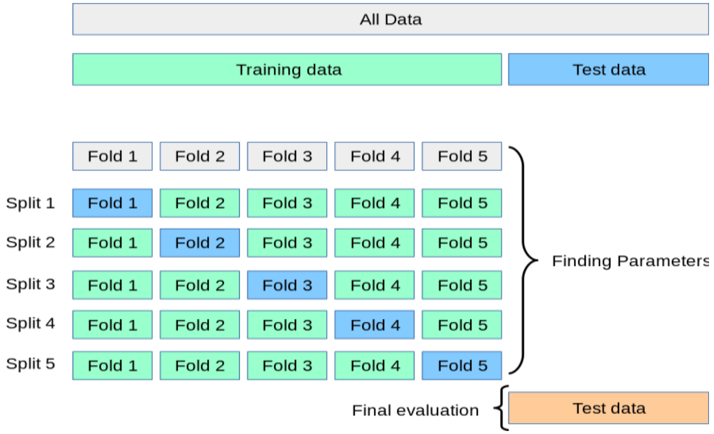
3. All three models must achieve a minimum efficiency of 95%.
4. Existing ML model for current detector has a background recall of ~40%.

Model Development - Tuning

1. Tuning a model ensures the model fits the true trend of the provided data to improve chosen metric.
   - A too simple model is defined as underfitted.
   - Overfitted model fits to the noise in the data set.

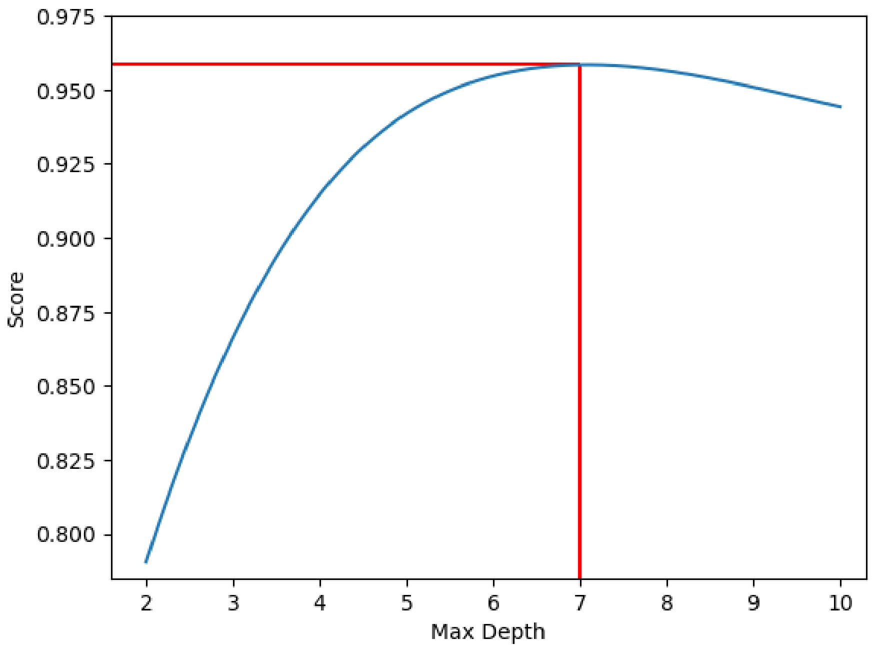


2. Cross validation can be used to reduce the chance of overfitting.
   - Split train and test data with 70-30 split.
   - 5 fold cross validation performed on training, large variation in performances suggests overfitting.



Model Development - Tuning

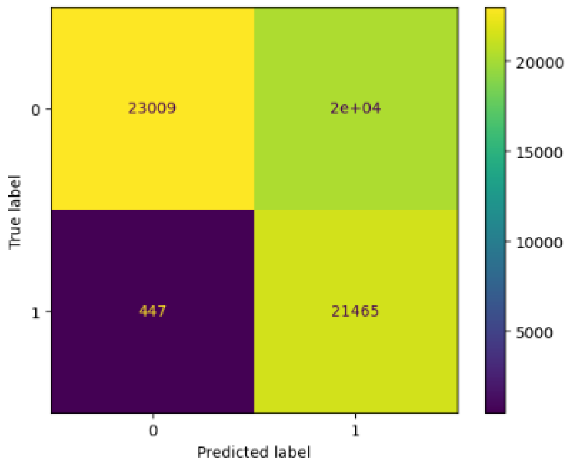
1. Tuning works by running model using range of hyperparameters and finding the optimum values (where the efficiency is the highest).
2. Example using max_depth:
   - Max_depth is the number of branches in each tree of a random forest.
   - Increases max_depth increases complexity and therefore increase in accuracy.
   - Increasing too far causes overfitting and therefore a drop in accuracy.
   - Optimum value identified as max_depth = 7



Results

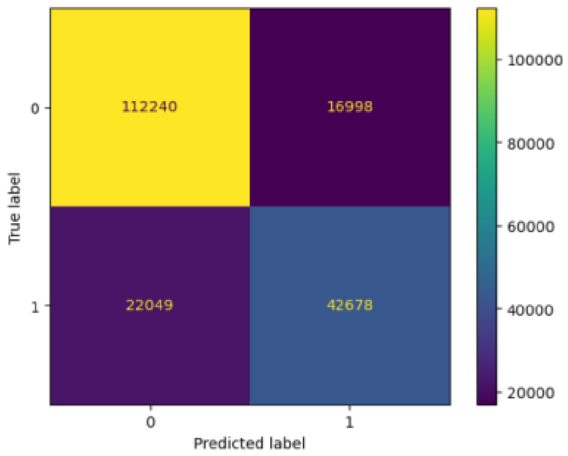
1. Achieved the desired >95% efficiency for all 3 models whilst keep accuracy high enough to reject significant amount of background.

Barrel Predictions



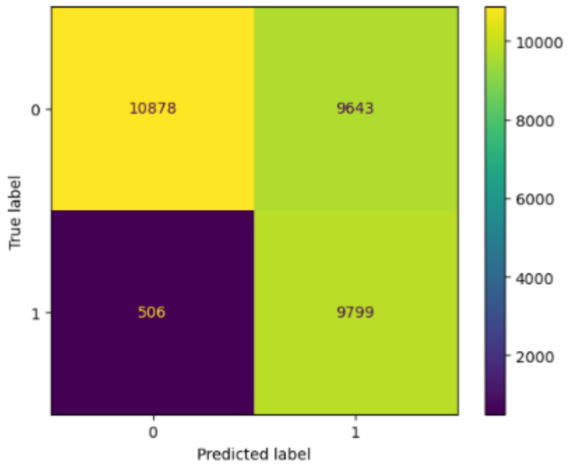
- Background Recall = 53.5%
- **Signal Recall = 98.0%**

Endcap Predictions



- Background Recall = 85.6%
- **Signal Recall = 95.8%**

Inclined Disks Predictions

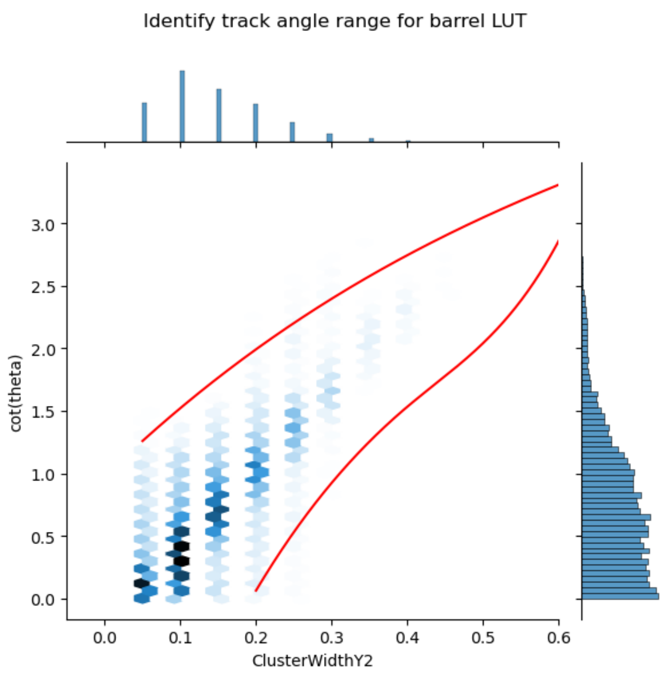


- Background Recall = 53.0%
- **Signal Recall = 95.1%**

Implementation and Next Steps

1. Using the models' predictions data is grouped with respect to cluster width and range of track angles identified where data is most probable to be signal.
   - Create LUT.
   - Causes some loss in accuracy.

2. LUT can be implemented into C++.
   - FTF should be run implementing LUT to assess the performance and how speed of program is affected.



THANK YOU!

VIKTORIA ERDI-KRAUSZ

11/08/23