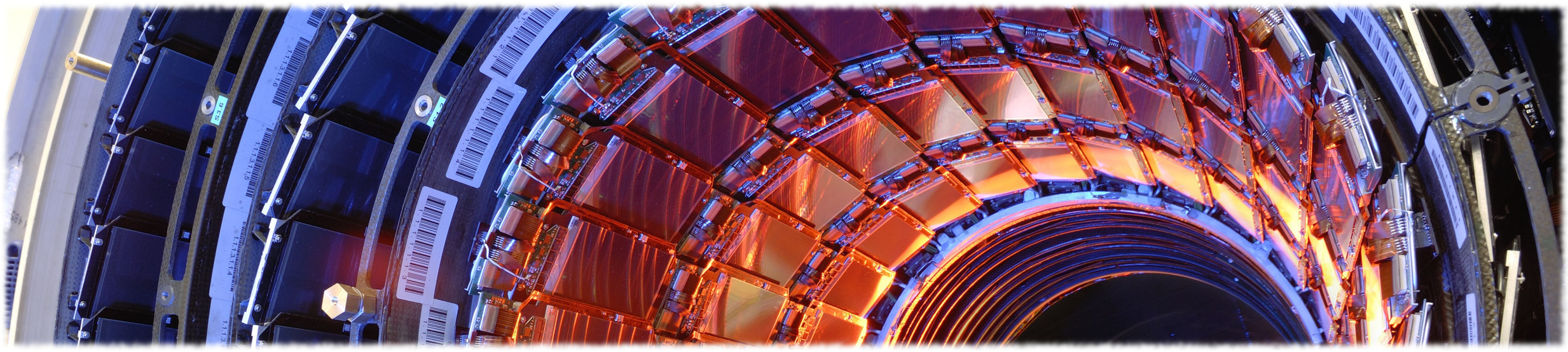
A novel deep neural network classifier

Assessing track quality in the Iterative Track Reconstruction at CMS

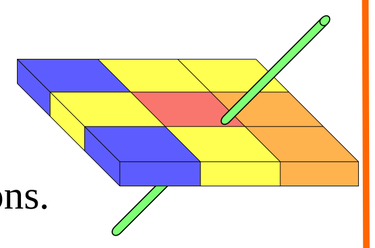


Joona Havukainen
On behalf of the CMS Collaboration
joona.havukainen@helsinki.fi

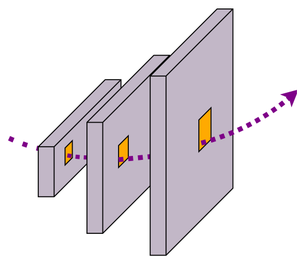
Reconstructing tracks

Track reconstruction in the CMS tracker can be split into four steps: Hit clustering, track seeding, track building and track fitting [1].

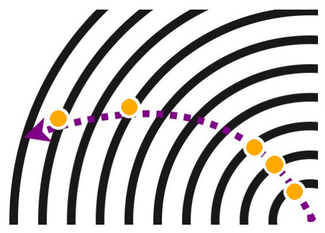
1) **Hit clustering**: Energy deposits left by the charged particles in the tracker are used to determine hit locations.



2) **Track seeding**: Hits in different layers compatible to be originating from a single particle are formed into track seeds.



3) **Track building**: Seeds are extended to include hits from outer layers calculating particle propagation using a Kalman filter.



4) **Track fitting**: A final fit is performed on the collection of hits associated to the track candidate.

After the fitting, the track candidate quality is assessed using multivariate analysis techniques on the track parameters. Poor quality tracks are rejected and good quality candidates are saved into a track collection and the hits associated to them can be masked from the remaining reconstruction.

Iterative tracking

Track reconstruction is a computationally demanding task. In order to reduce the combinatorial complexity of the problem an iterative approach has been chosen. The tracks that are easiest to find are searched in the early iterations and the signals associated to the found good quality tracks are masked from the later iterations to reduce the computational load.

This allows more resources to be spent on finding the difficult tracks. Specialized iterations can be used for example to search tracks in the densely populated regions inside highly energetic jets. The earlier a track gets reconstructed and accepted in the iterations, the more resources will be saved. The iterations, the seeds used and the targeted track types are presented in order they are applied in Table 1.

Name	Seed	Target track
Initial	Pixel triplets	Prompt
LowPtQuad	Pixel quadruplets	Prompt, low p_T
HighPtTriplet	Pixel triplets	Prompt, high p_T
LowPtTriplet	Pixel triplets	Prompt, high p_T
DetachedQuad	Pixel quadruplets	Displaced, low p_T
DetachedTriplet	Pixel triplets	Displaced, low p_T
PixelPair	Pixel pairs	Recover high p_T
MixedTriplet	Pixel+strip triplets	Displaced
PixelLess	TIB/TID/TEC strip triplets	More displaced
TobTec	TOB/TEC ring 5 strip triplets	Very displaced
JetCoreRegional	Pixel+strip pairs	Prompt, high p_T, merged pixel clusters

Table 1: The iterations in the order they are applied, the seeds used for building the tracks and the targeted track types.

Track quality estimation

An accurate method for estimating the track quality is needed both for masking the signals that are associated to reconstructed tracks and for rejecting *fake tracks*. These are tracks that are falsely reconstructed from unrelated hits or tracks that are badly reconstructed with spurious hits.

Good performance has been achieved using track variables such as the χ^2 value for the fit, the number of hits in the track and track displacement as an input to a machine learning method that is trained to tell the difference between fake and true tracks. So far the method used for this classification has been an ensemble of Boosted Decision Trees (BDT).

As the different iterations target different types of tracks, it has been necessary to have a separate classifier prepared for every iteration. This unavoidably complicates both the training process of the machine learning method and its application in the reconstruction pipeline. We propose using a Deep Neural Network (DNN) as a classifier instead and taking advantage of its large capacity to perform the classification with just a single classifier.

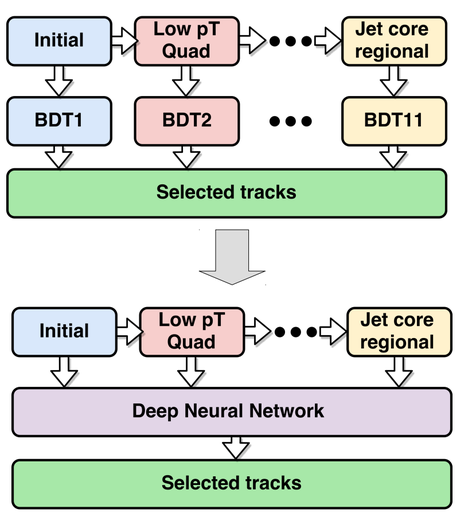


Figure 1: Using a single Deep Neural Network classifier instead of 11 different ensembles of Boosted Decision Trees simplifies both the training and the evaluation.

Deep Neural Network

The initial goal is to find a DNN using the same input variables as the current BDTs that would reach at least the same level of performance. Already with a relatively simple DNN, using four hidden dense layers (300, 150, 20, 10 neurons), the network is able reach and in some cases surpass the performance of the BDTs, as shown in Figure 1 demonstrating the efficiency and fake rate as a function of p_T for a sample of $t\bar{t}$ events with pile-up 50.

However making sure that the network is able to perform well also on exotic track types that are only present in small quantities or not at all in the training samples has proven to be difficult and is still to be understood. As the classifier decides if a reconstructed track is stored or rejected, unexpected behavior on rare track types can cause decrease in efficiency.

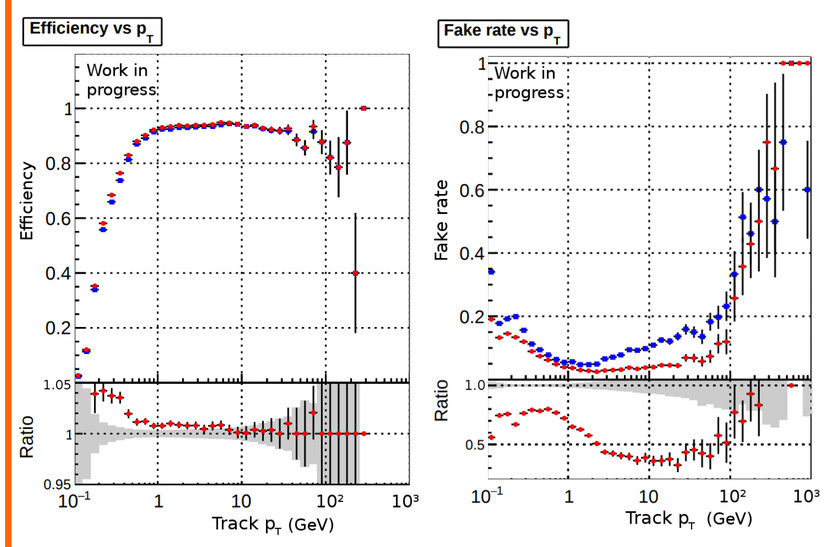


Figure 2: The DNN classifier is able to outperform the BDT classifier both in efficiency and fake rate, on a sample of $t\bar{t}$ events with pile-up 50.

Future work

- Ensuring good performance on rare tracks
  - Sample weights, network generalization
- Including hit variables to the classification
  - Hit type, layer, submodule, fit residual
- Compilation for fast evaluation when deployed
  - AOT TensorFlow, C++ TensorFlow
- Hyperparameter optimization

[1] The CMS Collaboration: Description and performance of track and primary-vertex reconstruction with the CMS tracker, CMS-TRK-11-001, arXiv:1405.6569v2, 2014