DeepCore2.0: Convolutional Neural Network for Tracking in Jets with High Transverse Momentum in CMS

Hichem Bouchamaoui on behalf of the CMS collaboration.

PRINCETON UNIVERSITY

Motivation

The tracking efficiency [1] in the core of high p_T jets is affected by cluster merging: as p_T^{jet} increases, the occupancy and number of merged clusters in **the barrel pixel detector layers (BPIX)** increase. Moreover, the increase in track density leads to a higher probability to mis-reconstruct tracks in the core of jets. To mitigate this, more candidate tracks need to be fitted to the BPIX hits to correctly reconstruct particle tracks, resulting in exponentially increasing combinatorics and more CPU time.

JetCore [2], the current tracking iteration in the core of high p_T jets, splits merged clusters associated to **AK4 [3] calo-jets** with $p_T^{jet} > 100$ GeV and uses more candidate tracks to improve tracking. However, there is small room for improvement from improved cluster splitting and using more candidate tracks uses significantly more CPU time.

DeepCore2.0 is a **Convolutional Neural Network (CNN)** based on Tensorflow [4] and Keras [5]. DeepCore2.0 using charge information from BPIX1 to BPIX4 to predict a **Track Crossing Point (TCP)** on BPIX2, resulting in better track seeds. DeepCore2.0 is an updated (Run 3 + fixes) and improved version of the **original version of DeepCore [6]** (Deepcore1.0) that was tested in Run2.

Input and Target

Merged cluster information: AK4 calo-jet with $p_T^{jet} > 100$ GeV and $\Delta R(jet, cluster) \leq 0.1$

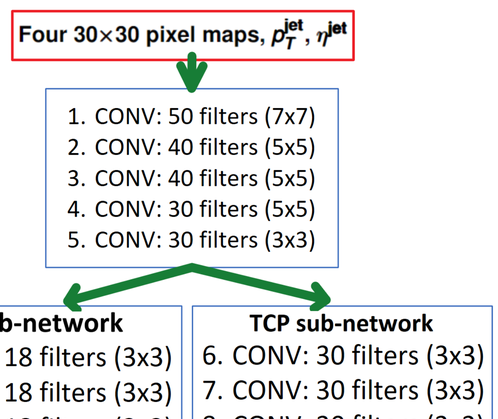
- **p_T^{jet}, η^{jet} and 30x30 x-y maps of pixel ADC counts (normalized) for BPIX 1 to 4.**

Target format: three sets of 30x30 TCP maps, three sets of 30x30 **Track Parameter (TP)** maps. The **first set of TCP/TP maps (Overlap map 1)** is used for the first particle crossing a pixel. If 2 particles cross the same pixel, **Overlap map 2** is used as well. **Overlap maps 3** is used if 3 particles cross the same pixel.

- TCP map: every BPIX2 pixel is assigned 1 if a particle crossed it, and 0 otherwise.
- TP map: $\Delta x, \Delta y, \Delta\eta, \Delta\phi, p_T$ associated to a TCP.
- The predicted TCP maps are filled with TCP scores between 0 and 1, and the predicted TP maps are filled with the corresponding track parameters.

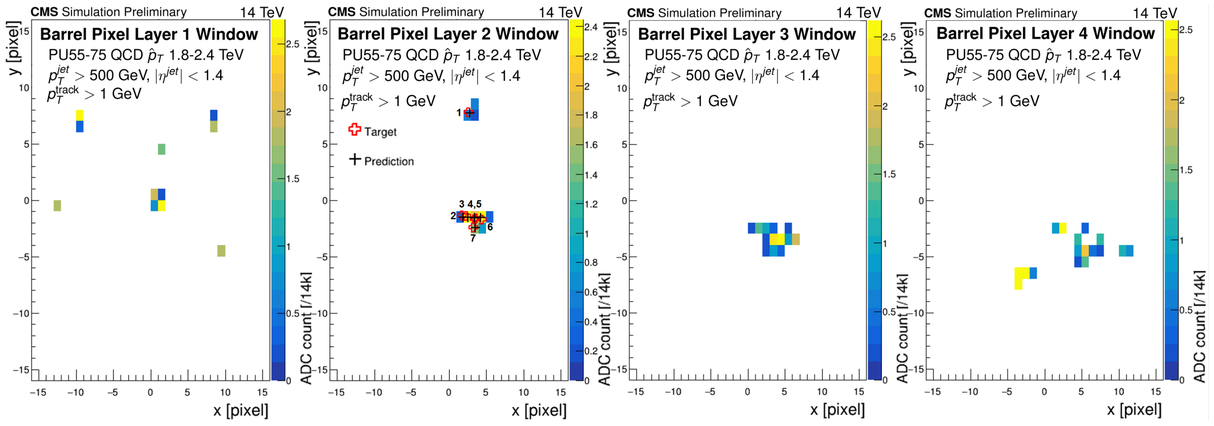
Training Details and Prediction Threshold

Epochs	Learning Rate (LR)	TCP Loss Function
1-15	10^{-4}	Weighted Binary Cross Entropy
16-20	10^{-5}	Weighted Binary Cross Entropy
21-25	10^{-5}	Weighted Binary Cross Entropy (including far pixels)
26-30	10^{-6}	Weighted Binary Cross Entropy (including far pixels)

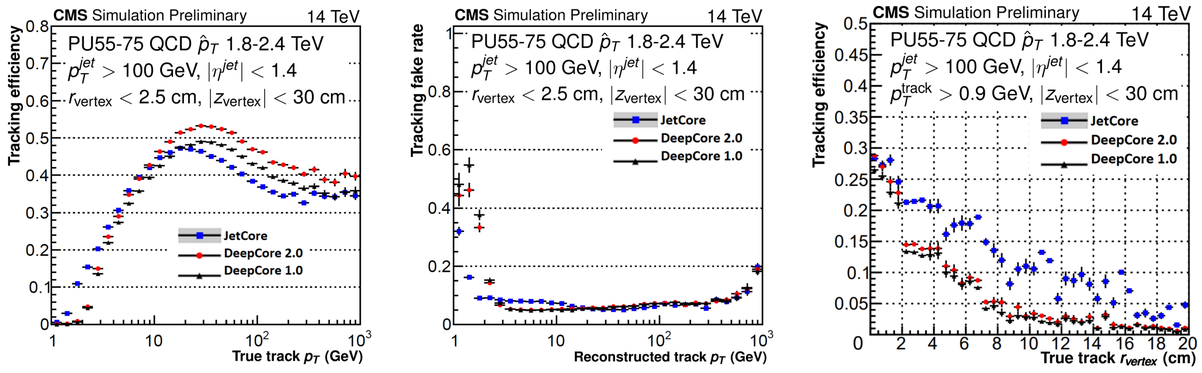


Dataset: **2M QCD** MultiJet simulated [7] events with $\sqrt{s} = 14$ TeV, PU55-75 and $1.8 < \hat{p}_T < 2.4$ TeV.
Jet selection: $p_T^{jet} > 500$ GeV, $|\eta^{jet}| < 1.4$; Simulated track selection: $p_T^{track} > 1$ GeV.
Samples: **10.5M clusters** for training with 0.2 validation and 150k clusters for testing.
Loss function: weighted Binary Cross Entropy [8] for TCP maps and Mean Square Error for TP maps.
Activation function: all ReLU [9] except Sigmoid in the last TCP CONV layer.
Optimizer: Adam [10], batch size: 64.

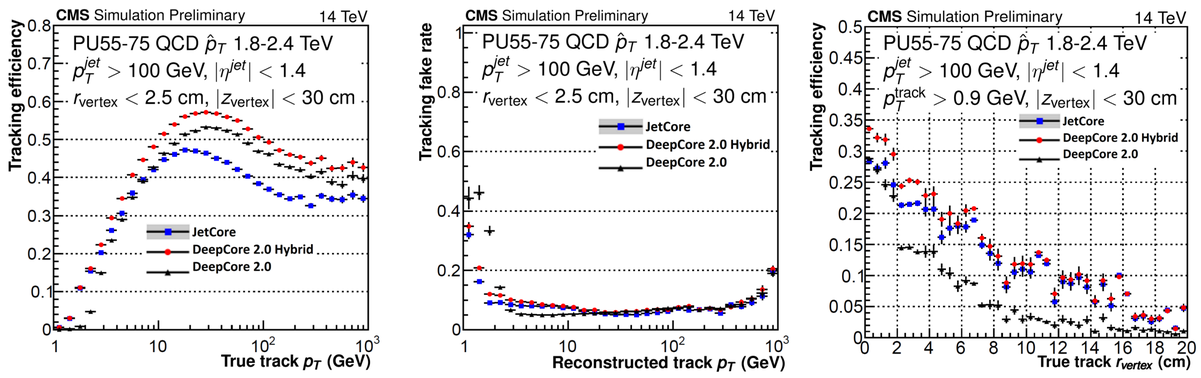
The prediction threshold depends on Overlap and sum of charge deposit on a given BPIX. The set of prediction thresholds is 0.7/0.85/1 for the first/second/third particle in Overlap and **they are raised** to 0.8/0/9/1 if BPIX1/3/4 are empty and disabled if BPIX2 is empty.



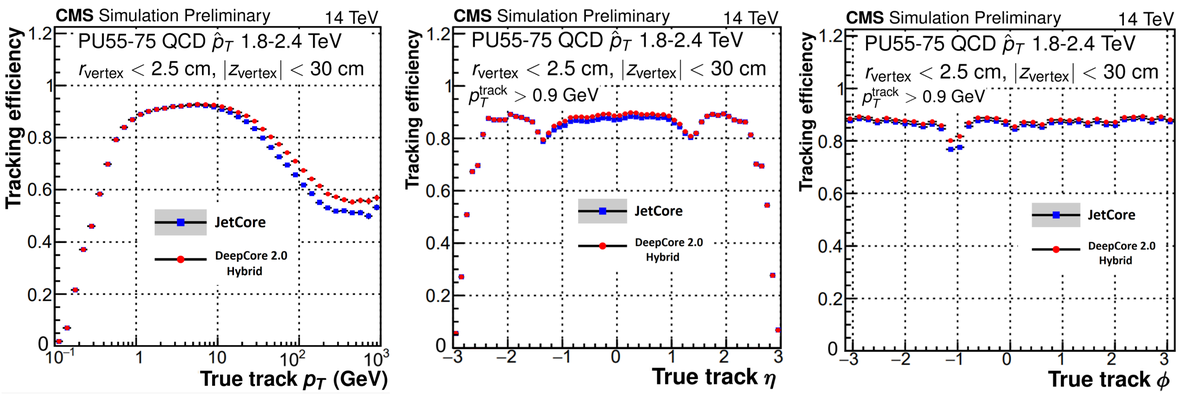
Tracking Performance: DeepCore2.0



The tracking performance is evaluated using **10k simulated QCD** events with $\sqrt{s} = 14$ TeV, PU55-75, $1.8 < \hat{p}_T < 2.4$ TeV and using the Run 3 configuration of the CMS detector.
The plots above show the **efficiency/fake rate vs p_T^{track}/production vertex radial position (r_{vertex})** for the tracking iteration dedicated to tracks in the core of high p_T jets.
True tracks come from hard-scatter processes (excludes pileup).
Despite significant improvements relative to DeepCore1.0 and JetCore in the high track p_T region, the efficiency of DeepCore2.0 for displaced track is significantly lower due to its heavy reliance on BPIX2. **DeepCore2.0-Hybrid** combines DeepCore2.0 and JetCore and uses less candidate tracks, which leads to **lower CPU time.**



Overall Tracking Performance: DeepCore2.0-Hybrid



The plots above show the **efficiency vs $p_T^{track}/\eta^{track}/\phi^{track}$ for all tracking iterations**.
DeepCore2.0-Hybrid improves the tracking efficiency of true tracks with $p_T^{track} > 200$ GeV by **over 10% relative to JetCore**, with no significant impact on the fake rate. DeepCore2.0-Hybrid improves the tracking efficiency of true tracks with $|\eta^{track}| < 1.4$ and all the ϕ range, **especially in the region $-1.6 < \eta < 0, \phi \approx -1 \pm 0.2$ [11]** where broken BPIX3/4 modules lead to a decrease in efficiency relative to the rest of the detector.

Summary and Status

DeepCore2.0-Hybrid is an optimized combination of DeepCore2.0 and JetCore with lower CPU timing and improved tracking performance. Further validation is currently ongoing. DeepCore2.0-Hybrid can be **further optimized** with an updated training and the **CPU time can be further lowered** by tuning the combination of DeepCore2.0 and JetCore.

References

[1] Description and performance of track and primary-vertex reconstruction with the CMS tracker, the CMS Collaboration. Journal of Instrumentation, vol 9 (2014), P10009.
[2] High pT jets tracking and cluster splitting, The CMS Collaboration, CMS-DP-14/032.
[3] The anti-kT jet clustering algorithm, M. Cacciari, G. P. Salam, G. Soyez, Journal of High Energy Physics, vol. 2008, no. 04 (2008), P063.
[4] Keras, F. Chollet et al., Software available from https://github.com/keras-team/keras.
[5] TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems, M. Abadi et al., Software available from tensorflow.org, e-Print: 1603.04467 [cs.DC].
[6] DeepCore: Convolutional Neural Network for high pT jet tracking, The CMS Collaboration, CMS-DP-19/007.
[7] A Brief Introduction to PYTHIA 8.1, Torbjörn Sjöstrand, Stephen Mrenna, Peter Skands. Computer Physics Communications, vol. 178, no. 11 (2008), P852–67.
[8] Scikit-learn: Machine Learning in Python, F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel et al., Journal of Machine Learning Research 12 (2011) P2825.
[9] Deep Learning using Rectified Linear Units (ReLU), A.F. Agarap, 2018.
[10] Adam: A Method for Stochastic Optimization, Diederik P. Kingma, Jimmy Lei Ba, e-Print: 1412.6980 [cs.LG].
[11] CMS Tracking Performance in 2023, The CMS Collaboration, CMS-DP-23/090.

Presented at the 6th Inter-experiment Machine Learning Workshop 2024 in Geneva, Switzerland.
More details available at CMS-DP-2024/003 and hichemb@princeton.edu.