DeepCore: Convolutional Neural Network for high $p_T$ jet tracking

Valerio Bertacchi
On behalf of CMS Collaboration

Connecting the Dots and Workshop on Intelligent Trackers
2-5 April 2019, Valencia
Outlook

• High-Pt jet tracking: motivation and strategy
• DeepCore
  – Input, target, architecture
  – Loss functions and training
  – Event Display and training performance
• Integration in CMS reconstruction
• Tracking Performance with DeepCore
Tracking in high $p_T$ jets

Why?
- Jet reconstruction with Particle Flow\(^1\)
- $b$-tagging
- Substructures

Issues
- Tracking (based on **combinatorial Kalman Filter**) inside jets starts to become inefficient over 500 GeV
- Critical step is the splitting of the clusters in the pixel detector

---

1 all the references in the backup slides
Previous solution of the seeding inefficiency

Before this work has been developed a K-means based algorithm
• Efficiency gain, but still present inefficiencies due to the splitting
• each layer analyzed separately
Strategy

Goal:
Improve the jet-tracking seeding → *Eyes on the ball*: track parameters
- Efficient clustering is an issue? Let’s skip it!
- From raw pixel info obtain directly track parameters (seed) of the tracks around the jet axis

Strategy:
- Develop a Convolutional Neural Network (CNN) to reproduce the «function»:

![Diagram showing the process](image-url)
Convolutional NN approach

- Input / Target as 2D pictures
- Each node is a pixel
- Each node looks only in a small zone of the layer (filter)
- Discovered features shared between filters \(\rightarrow\) looks for pattern

- 4 layer as «RGB channels» of same picture
- All the track seeds predicted at same time
  - Independent of the number of seeds/tracks
  - Indipendent of the dimension of the map
Input of DeepCore

• 4 pixel detector maps, merged clusters centered
  – interception of calo-jet axis with layer 1,
  – open a cone of $\Delta R = 0.1 \rightarrow$ list of merged clusters on layer 1
  – If layer 1 module is broken $\rightarrow$ list of merged clusters on layer 2
  – for each merged cluster $\rightarrow$ open a window 30x30 pixel in each layer (primary vertex-merged cluster direction)
  – Added to list of the direction the jet axis from calorimeter information
  – save x; y; charge information for each pixel in the 4 windows for each merged cluster
  – Charge info normalized to order of magnitude 1

• jet $\eta$ (for each merged cluster of the list)
• jet $p_T$ (for each merged cluster of the list)
Target of DeepCore

For each pixel in 30x30 window of layer 2:

- 1 if a track cross that pixel (Track Crossing Point), 0 otherwise
- **Track parameters** in local parametrization: \((\Delta x, \Delta y, \Delta \phi, \Delta \eta, p_T)\)
  - For the track-crossing pixels (TCP)
  - In a circle of radius 4-pixel (NCP)

**Overlap**: additional 1/0 + track parameters info if a second or third track cross the pixel

\(p_T\) **range**: tracks with \(p_T > 1\) GeV
Network architecture

Input Pixel 4-map, $\eta_{\text{jet}}, p_{T\text{jet}}$

1. CONV: 50 filters, 7x7
2. CONV: 20 filters, 5x5
3. CONV: 20 filters, 5x5
4. CONV: 18 filters, 5x5
5. CONV: 18 filters, 3x3

6. CONV: 18 filters, 3x3
7. CONV: 18 filters, 3x3
8. CONV: 18 filters, 3x3
9. CONV: 18 filters, 3x3

Activation Functions:
all ReLU except Sigmoid on last probability layer

Number of parameters: 77373
Training

Loss functions:

- Crossing Points $\rightarrow$ weighted **Binary Cross Entropy**:
  $$\frac{1}{N} \sum_{i=1}^{N} \left[ y_i^{true} \ln(y_i^{pred}) + (1 - y_i^{true}) \ln(1 - y_i^{pred}) \right]$$
  but
  - Different weights:
    - TCP$^2$-pixel counts 10
    - NCP$^3$ pixel counts 1
    - Other pixel counts 0.01

- Parameters $\rightarrow$ **mean squared error**:
  - clipped between [-5; 5]
  - Averaged between TCP and NCP only
  $$\frac{\sum_{p\in TCP,NCP} \min[(p_{pred} - p_{target})^2, 25]}{N_{TCP+NCP}}$$

Training:

- Sample composition:
  - QCD events $1.8 \text{ TeV} < \hat{p}_T < 2.4 \text{ TeV}$
  - $p_T^{jet} > 1 \text{ TeV}, \ |\eta^{jet}| < 1.4$
- Sample dimension: **22M input (2M jets)**, 2M input for validation
- Batch size: **32**
- Optimizer: **Adam**
- Learning Rate: reduced during training: $2\cdot10^{-4} \rightarrow 1\cdot10^{-4} \rightarrow \ldots \rightarrow 1\cdot10^{-7}$
- Relative weights of losses: same weight

---

2 TCP = Track Crossing Point, where the TCP map target = 1.
3 NCP = pixel inside a circle of radius 2 pixel centered in the TCP
The “Event Display” of DeepCore

R = 3 cm  
R = 6.8 cm  
R = 10.2 cm  
R = 16 cm
The “Event Display” of DeepCore

R = 3 cm
R = 6.8 cm
R = 10.2 cm
R = 16 cm
The “Event Display” of DeepCore
Performance of DeepCore training

Residual distribution - $\eta$

| CMS | Simulation Preliminary |
|-----|------------------------|
| 13 TeV |

QCD events, $(PU) = 30$
1.8 TeV < $p_T$ < 2.4 TeV
$\rho_T^{jet} > 1$ TeV, $|\eta^{jet}| < 1.4$

$\sigma_{res} = 0.014$

Residual between the seed $\eta$ parameter predicted by DeepCore and the target (simulated) track $\eta$ parameter.

Correlation between prediction of DeepCore and target parameters shown with seed $\eta$ parameter predicted against the simulated track $\eta$ parameter.
Inside CMS Reconstruction

Standard reconstruction

- Quadruplet seeds → Quadruplet tracks
- Triplet seeds → Triplet tracks
- [... seeds] → [... tracks]
- jetCore seeds → jetCore tracks

Combinatorial Kalman Filter

Reconstructed Tracks

Pixel info → cluster → cluster splitting
Inside CMS Reconstruction

Reconstruction With DeepCore

- Quadruplet seeds
- Quadruplet tracks
- Triplet seeds
- Triplet tracks
- [… seeds]
- [… tracks]

DeepCore Seeds
- jetCore tracks

JetCoreDirectSeed Generator

Combinatorial Kalman Filter

Reconstructed Tracks

Pixel info 
cluster 
cluster splitting 

Reconstructed Tracks
Inside CMS Reconstruction: DeepCore seeds

1. Loop over calorimeter-jets with $p_T > 0.3$ TeV
2. For each cluster with DeltaR(jet, cluster) < 0.1 $\rightarrow$ new direction
3. Built a NN input: four 30x30 pixel windows around the found direction + ($\eta^{\text{jet}}, p_T^{\text{jet}}$)
4. Prediction of the NN $\rightarrow$ List of DeepCore Seeds

How the seed list built from the NN output?

- Saved 5 parameters of most probable pixels
- **Most Probable** = TCP>0.85, 0.75, 0.65 in the 3 repetitions, or 0.50,0.40,0.30 if layer 2 is missing
- **Cleaning**: if two seed have x,y closer than 50 $\mu$m, $\eta$, $\phi$ closer than 0.002 rad $\rightarrow$ remove one
- Uncertainty: **the same** for all the seeds: ($\sigma_{p_T} = 0.15$ GeV, $\sigma_\eta = \sigma_\phi = 0.002$, $\sigma_{xy} = \sigma_z = 44$ $\mu$m)
DeepCore performance in CMS reconstruction

NB: in the efficiency the shared reconstructed tracks (duplicated) between various iterations are not removed.
DeepCore compared performance in CMS reconstruction

CMS  
Simulation Preliminary

Tracking Efficiency

13 TeV

QCD 1800 GeV < \hat{p}_T < 2400 GeV (no PU)
\pT^{jet}>1 TeV, |\eta^{jet}|<1.4

- Without JetCore
- Standard JetCore
- DeepCore
- MC truth seeding

Fake Rate

CMS  
Simulation Preliminary

13 TeV

QCD 1800 GeV < \hat{p}_T < 2400 GeV (no PU)
\pT^{jet}>1 TeV, |\eta^{jet}|<1.4

- Without JetCore
- Standard JetCore
- DeepCore
- MC truth seeding
DeepCore compared performance in CMS reconstruction

Almost reached ideal seeding efficiency

Avoided all the fakes produced by jetCore
DeepCore compared performance in CMS reconstruction

Timing performance (JetCore iteration only):

DeepCore Time/ev / default JetCore Time/ev = 0.15

Almost reached ideal seeding efficiency

Avoided all the fakes produced by jetCore
• The CNN approach works for tracking at seeding level

• The DeepCore implementation for CMS reconstruction gives (in jet core regions):
  – almost cancelled seeding inefficiencies
  – fake rate reduction up to 60%
  – seeding time reduced by 85%

• The result is very preliminary:
  – Barrel only → must be extended, expect higher gain in endcaps
  – Further optimization on training (LR, batch size, architecture)
  – Extension to other steps of pattern recognition
BACKUP SLIDES
Standard seeding details

Why outward?
• Pixel→Lower occupancy
• Pixel→3D measurement
• Higher efficiency (low energy tracks, energy losses, interactions with material)

Seed parameters:
• 5 track parameters
• From three 3D hits or two 3D hits+vertex constraint

Algorithm:
• For each iteration defined 2 sets of parameters
  – Seeding layer: the used detectors
  – Tracking regions: optimized set of cuts to define good combinations of hits (on $p_T$, $d_0$, $|z_0|$, charge)
• If seeding layer is a pair of detector: look for pair of hits and applied cuts of the region
• If seeding layer is a triplet/quadruplet: Cellular Automaton seeding (see Felice’s talk)

Iteration (some):
• Pixel triplets: for prompt tracks
• Pair+vertex: also from Endcap strips, for forward tracks
• Mixed triplets: strips+pixel, for displaced tracks (b tracks, photon conversion, nuclear interaction)
• Strip Pairs: weaker constraints, recover efficiency for tracks produced outside pixel detector
• jetCore: pairs+axis constraint, high $p_T$ jet tracking
Details of the training

- Ep 0-11 → 200k ev, LR=0.0002
- Ep. 11-40 → 200k ev, LR=0.0001
- Ep. 40-191 → 200k ev, LR=0.00007
- Ep 191-200 → 200k ev, LR=0.00007, new loss
- Ep. 200-233 → 200k ev, LR= 0.000001, new loss
- Ep 233-239 → 22 M events, LR= 0.000001, new loss
- Ep 239-246 → 22 M events, LR =0.0000001, new loss, epochs of 1M ev.

- Small sample (200k events)
- BCE with 0 out of NCP
- Complete Sample
- Complete BCE
Definitions

Tracking Efficiency
Number of reconstructed tracks associated to a simulated one divided by the number of simulated tracks. A reconstructed track is flagged as “associated” if the $\chi^2$ between its parameters and the simulated is lower than 25.

Fake Rate
Number of reconstructed tracks not associated to a simulated one divided by the number of reconstructed tracks.

Duplicate rate
Number of tracks duplicate divided by the number of reconstructed tracks. A reconstructed track is flagged as “duplicate” if at least another reconstructed track is associated to the same simulated track of the first one.

Track selection
The typical selection for simulated tracks used in validation: $|\eta^{\text{jet}}|<2.5$, $r_{\text{prod}}<3$ cm, $|z_{\text{prod}}|<30$ cm, $p_T>0.9$ GeV.
Technicalities and references

YOLO (You Only Look Once)
Our source of inspiration for CNN approach
References:
- https://pjreddie.com/darknet/yolo/
- arXiv:1506.02640v5
- arXiv:1612.08242v1

Adam Optimizer
Extension to stochastic gradient descent (adaptive per-parameter LR, use of the second moment of gradients)
References: https://arxiv.org/abs/1412.6980

CMS references
• Track Reconstruction: arXiv:1405.6569v2
• Particle Flow: arXiv:1706.04965v2
• b-tagging and vertexing: arXiv:1712.07158v2
• Jet substructures: arXiv:1803.06991v2

Activation functions:
• ReLU: \( f(x) = \max(0, x) \)
• Sigmoid: \( f(x) = \frac{1}{1+e^{-x}} \)

Timing Performance:
Unofficial absolute values for jetCore iteration only:
• DeepCore: 2.5 sec./ev.
• Standard JetCore: 16.6 sec./ev.