DeepJet: a deep-learned multiclass jet-tagger for slim and fat jets

Mauro Verzetti\textsuperscript{1,2}, Jan Kieseler\textsuperscript{1}, Markus Stoye\textsuperscript{3}, Huilin Qu\textsuperscript{4}, Loukas Gouskos\textsuperscript{4}
on behalf of the CMS Collaboration

\textsuperscript{1}CERN, \textsuperscript{2}FWO, \textsuperscript{3}Imperial College London, \textsuperscript{4}UC Santa Barbara
The problem
The problem
Last year development — DeepCSV

Performance of the c jet identification efficiency algorithms demonstrating the probability for b jets to be misidentified as c jet as a function of the efficiency to correctly identify c jets. The curves are obtained on simulated $tt\bar{t}$ events using jets within tracker acceptance with $p_T > 30$ GeV, b jets from gluon splitting to a pair of b quarks are considered as b jets. The lines shown are for CSVv2, DeepCSV $C_{vsB}$, c tagger $C_{vsB}$ and cMVAv2. cMVAv2 and the c tagger use also the information from the soft leptons inside jets, while CSVv2, DeepCSV do not. The irregularity observed in the ROC curve of the c tagger is caused by a sharp feature in the discriminator distribution due to jets without any selected tracks.

Dense 100 nodes x 5 layers

Output classes: b, bb, c, l

Charged (8 features) x6

Secondary Vtx (8 features) x1

Global variables (12 features)

CMS Simulation Preliminary

$\sqrt{s} = 13$ TeV, 2016

tt events
AK4jets ($p_T > 30$ GeV)

b-jet misid. probability

Output classes: b, bb, c, l

CMS Simulation Preliminary

$\sqrt{s} = 13$ TeV, 2016

tt events
AK4jets ($p_T > 30$ GeV)

udsg-jet misid. probability

arXiv:1712.07158
Last year development — DeepCSV

35.9 fb⁻¹ (13 TeV, 2016)

arXiv:1712.07158
Trying more complex architectures

• Convolutional NN successfully applied in neutrino physics and image recognition
• Some proposals to treat jets as images

Boosted W

... but
• Jets do not look like normal images!
• CMS events are way more complex and bring more information than a flat image (e.g. tracking information)
Particle-based NN architecture

- Charged (16 features) x25
  - 1x1 conv. 64/32/32/8
  - RNN 150

- Neutral (8 features) x25
  - 1x1 conv. 32/16/4
  - RNN 50

- Secondary Vtx (17 features) x4
  - 1x1 conv. 64/32/32/8
  - RNN 50

- Global variables (6 features)

- Dense
  - 200 nodes x1, 100 nodes x6

M. Verzetti (CERN and FWO)
Particle-based NN architecture

Convolutional layers progressively learn a more compact feature representation (automatic feature engineering)
Particle-based NN architecture

- Charged (16 features) x 25
- Neutral (8 features) x 25
- Secondary Vtx (17 features) x 4
- Global variables (6 features)

Convolutional layers progressively learn a more compact feature representation (automatic feature engineering)

The recurrent layers (LSTM) builds a “summary” of the information contained in each set of feature types

Dense
200 nodes x 1, 100 nodes x 6
Particle-based NN architecture

- Charged (16 features) x25
- Neutral (8 features) x25
- Secondary Vtx (17 features) x4
- Global variables (6 features)

1x1 conv. 64/32/32/8 → RNN 150
1x1 conv. 32/16/4 → RNN 50
1x1 conv. 64/32/32/8 → RNN 50

Dense
- 200 nodes x1,
- 100 nodes x6

Figure 3: Performance of the b jet identification algorithms demonstrating the probability for non-b jets to be misidentified as b jet, as a function of the efficiency to correctly identify b jets. The curves are obtained on simulated ttbar events using jets within $\text{abs}(\eta)<2.4$ and with $p_T>30$ GeV. The b jets from gluon splitting to a pair of b quarks are considered as b jets. The lines shown are for DeepCSV (retrained for the Phase 1 detector geometry), NoConv, and DeepFlavour. The NoConv algorithm serves only for comparison. The absolute performance in this figure serves as an illustration since the b jet identification efficiency depends on the $p_T$ and $\eta$ distribution of the jets in the topology as well as the amount of b jets from gluon splitting in the sample.

Figure 5: Performance of the DeepCSV (retrained for the Phase 1 detector geometry) and DeepFlavour b jet identification algorithms demonstrating the probability for non-b jets to be misidentified as b jet, as a function of the efficiency to correctly identify b jets. The curves are obtained on simulated QCD multijet events using jets within $\text{abs}(\eta)<2.4$ and with $300$ GeV $< p_T < 600$ GeV. The b jets from gluon splitting to a pair of b quarks are considered as b jets. The absolute performance in this figure serves as an illustration since the b jet identification efficiency depends on the $p_T$ and $\eta$ distribution of the jets in the topology as well as the amount of b jets from gluon splitting in the sample.

CMS-DP-2017-013
Particle-based NN architecture

Charged (16 features) x25

Neutral (8 features) x25

Secondary Vtx (17 features) x4

Global variables (6 features)

1x1 conv. 64/32/32/8 → RNN 150

1x1 conv. 32/16/4 → RNN 50

1x1 conv. 64/32/32/8 → RNN 50

Dense
200 nodes x1, 100 nodes x6

Figure 3: Performance of the b jet identification algorithms demonstrating the probability for non-b jets to be misidentified as b jet, as a function of the efficiency to correctly identify b jets. The curves are obtained on simulated ttbar events using jets within $|\eta|<2.4$ and with $p_T>30$ GeV. The b jets from gluon splitting to a pair of b quarks are considered as b jets. The lines shown are for DeepCSV (retrained for the Phase 1 detector geometry), NoConv, and DeepFlavour. The NoConv algorithm serves only for comparison. The absolute performance in this figure serves as an illustration since the b jet identification efficiency depends on the $p_T$ and $\eta$ distribution of the jets in the topology as well as the amount of b jets from gluon splitting in the sample.

Figure 5: Performance of the DeepCSV (retrained for the Phase 1 detector geometry) and DeepFlavour b jet identification algorithms demonstrating the probability for non-b jets to be misidentified as b jet, as a function of the efficiency to correctly identify b jets. The curves are obtained on simulated QCD multijet events using jets within $|\eta|<2.4$ and with $300$ GeV $< p_T < 600$ GeV. The b jets from gluon splitting to a pair of b quarks are considered as b jets. The absolute performance in this figure serves as an illustration since the b jet identification efficiency depends on the $p_T$ and $\eta$ distribution of the jets in the topology as well as the amount of b jets from gluon splitting in the sample.
Particle-based NN architecture

Charged (16 features) x25
- 1x1 conv. 64/32/32/8
- RNN 150

Neutral (8 features) x25
- 1x1 conv. 32/16/4
- RNN 50

Secondary Vtx (17 features) x4
- 1x1 conv. 64/32/32/8
- RNN 50

Global variables (6 features)

Dense
- 200 nodes x1
- 100 nodes x6

b
bb
c
uds
g

Similar performance to simpler, dedicated binary taggers, but with full multi-class power.

Significantly better performances in given regions with different quark composition

CMS-DP-2017-027
• Significantly larger amount of candidates used to accommodate for 90% of the fat jets
• Need to learn substructure from both charged and neutral candidates
• RNNs become computationally too expensive to train
• Use particle-level convolutional layers (P-CNN) where each feature is treated as a “colour”
P-CNNs

\[ z^\alpha_m = \sum_a \sum_j k^\alpha_{a,j} x_{a,(m+j-1)} \]
P-CNNs

\[ z^\alpha_m = \sum_a \sum_j k^\alpha_{a,j} x_{a,(m+j-1)} \]

- **Sweep over the elements**
- **Loop over contiguous elements of the kernel**
Multiple features ("colours") are accounted computing the transformation 

\[ Z_\alpha^m = \sum_a \sum_j k_\alpha_{a,j} \times a_{(m+j-1)} \]
P-CNNs

\[ Z_{\alpha m} = \sum_a \sum_j k_{\alpha,a,j} X_{a,(m+j-1)} \]

Different filters/kernels learn different transformations
Performance

- Flavour information largely improves jet tagging
- Large improvement w.r.t to the BDT approach
- Introduces mass sculpting, not necessarily a bad thing

CMS-DP-2017-049
Deploying the model
Two worlds colliding

Training / Analysis:

- Keras + TensorFlow
- Python-based
- Private productions
- Minimal interaction with ROOT
- Few processes, single threads
- Little memory constraints
- Expendable jobs

Production:

- Custom framework
- C++ based (speed!)
- Mostly ROOT-centric (at least I/O)
- Many processes, multiple threads
- Many other concurrent activities → memory constraints
- Processes cannot die (e.g. trigger)
Integration of DeepJet (AK4) into CMSSW. PR #19893

Tensorflow-based integration of new DeepFlavour tagger
#19893

Pablo Decm commented on 25 Jul 2017 - edited

This pull request integrates the new DeepFlavour tagger, using the library CMSSW-DNN by @riga (the required part is also included) and adds it to the standard sequences. You can find an overview of the reason and design behind this PR in this BTV WG presentation.

PAT vs reference training framework (latest version)

Here are some checks of compatibility of CMSSW pat-based discriminators computed using the producers develop for this PR with the output from the training framework (DeepJet) as 2D histograms.
Tensorflow-based integration of new DeepFlavour tagger

#19893

cmsbuild merged 150 commits into cms-sw:master from pablodecm:ceep_flavour_tf_rebased_2017_07 on 25 Jan

pablodecm commented on 25 Jul 2017

This pull request integrates the new DeepFlavour tagger, using the library CMSSW-DNN by @riga (the required part is also included) and adds it to the standard sequences. You can find an overview of the reason and design behind this PR in this BTV WG presentation.

PAT vs reference training framework (latest version)

Here are some checks of compatibility of CMSSW pat-based discriminators computed using the producers develop for this PR with the output from the training framework (DeepJet) as 2D histograms
Backend choice

✗ Interface based on TF python API:
  • Uses python C API and a pre-built TF package
  • Large overhead and no handle on memory/threading

✗ Interface based on TF C API:
  • Low level and not very convenient
  • Lots of customisations and ad-hoc handling needed

✓ Interface based on TF C++ API:
  • Access to all the needed internals for production usage with minimal need for custom code
  • Shallow interface to connect TF to the CMSSW internals (e.g. logging)

For more information look here.
Remaining issues

Multithreading:

- TF **loves** threads
- Normally a good thing, has a critical impact on memory consumption in HEP frameworks, which have their own thread schemes/pools (CMSSW uses TBB)
- Solved with the implementation of two custom sessions: **without** threading and **sharing the threading pool**

Memory footprint

- DeepJet model initially very big (~150MB)
- A careful optimisation for inference only can brought O(10-100) gain in memory reduction
- Further improvements from separating graph storage (common) and graph evaluation (one each thread)
- Exploring AOT compilation as future option
Summary

- Jet tagging is of paramount importance for the CMS Physics program

- Lots of development in the last ~1.5 years to apply modern machine learning techniques to this field
  - Large improvements in performance
  - Still some room for new developments, especially in the boosted regime

- Flavour tagging is not only fancy algorithms, but solid and performing computing infrastructures as well