Jet Energy Corrections with DNN Regression

Daniel Holmberg

CMS ML Forum 08.09.2021

Introduction

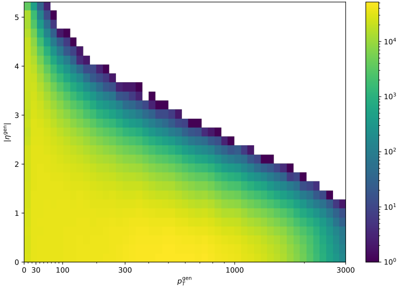
- The physical detector causes the jet transverse momentum p_T to be di erent from the true particle-level jet
- Corrected such that it agrees on average with the p_T of the particle level jet
  - Determined by using basic kinematic quantities of the jet
- Possible to include more information and get better corrections using machine learning
  - Has been done successfully for b-jets using a deep feed-forward neural network
- However, this study is about generically applicable DNN-based corrections

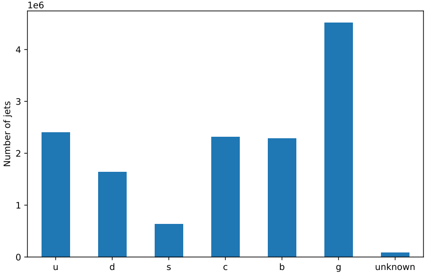
Dataset

- QCD H_T-binned samples, 2016 configuration
  - `/QCD_HT*_TuneCUETP8M1_13TeV-madgraphMLM-pythia8/RunIISummer16MiniAODv3*/MINIAODSIM`
- Custom ML JEC dataset by A. Popov (ULB)
- Forked and added SV angles for initial coordinates in ParticleNet
- Use 10M jets for training set, 2M jets for validation set and 2M jets for test set

Data distribution

- Same shape for all jet avours
- Flat in $(p_T;\ \ )$ at low p_T
- Steeply falling in p_T at high p_T
- Proportions of b, c, uds, and g jets xed as 1 : 1 : 2 : 2





Training features

- Event level
  - p_T, $\log p_T$, , , , mass, area
  - multiplicity, $p_T D$, $_2$, num pv
- Charged PF candidates
  - p_T, , , p_T, ,
  - dxy, dz, dxy signi cance, normalized ^{2}
  - num hits, num pixel hits, lost hits
  - particle id, pv association quality
- Neutral PF candidates
  - p_T, , , p_T, ,
  - particle id, hcal energy fraction
- Secondary vertices
  - p_T, , , p_T, , , mass
  - ight distance, signi cance, num tracks

Feature engineering

Create event-level features: multiplicity, $p_T D$, $_2$ that <u>helps with quark gluon discrimation</u>

Feature engineering

- Relative features for all constituents
  - $p_{T;i} = p^{pf}_{T;i} = p^{jet}_T$
  - $_i = \mathrm{sgn}(\ ^{jet})(\ ^{pf}_i \quad ^{jet})$
  - $_i = (\ ^{pf}_i \quad ^{jet} + \quad)\mathrm{mod}(2\ )$
- One hot encode categorical features
  - particle id and primary vertex association quality
  - e.g. neutral pid:

```
[1, 2, 22, 130] -> [
   [1, 0, 0, 0], [0, 1, 0, 0],
   [0, 0, 1, 0], [0, 0, 0, 1]
]
```

Target and loss

- Regression target $\hat{y} = \log(p_T^{gen}/p_T)$
  - Correction factor is thus e^{y} where y is the NN output
- MAE loss function $L = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i| I_{|\hat{y}_i|<1}$
  - The last factor rejects 0.8% of jets where the target is way o

Choice of ML models

- For every jet there are global features as well as constituents
- Jet constituents form a permutation invariant set
  - Number of constituents varies from jet to jet
  - Order doesn't matter
  -) Requires special treatment to use it for ML
- DeepSets and DynamicGraphCNN are examples of NN architectures allowing for unordered sets to be consumed
  - They have been used for jet tagging in Energy Flow Networks and ParticleNet respectively
- Modi ed versions of Deep Sets and ParticleNet are used to include all available information, both global features and constituents!

Deep Sets

- Used a JEC study with Deep Sets from 2020 as baseline
- Procedure
  - An MLP $F : x_i \to y_i$ is applied to every constituent x_i
  - Weights of the MLP are shared among all constituents
  - The learned parameters are aggregated using a permutation invariant operation
  - Here the sum over all constituents $\sum_i y_i$ is chosen
  - This is based on the theorem that any function $G(\{x_i\})$ invariant under permutations of its inputs can be represented in the form $\sum_i F(x_i)$
- Concatenate with global features and feed into MLP

Deep Sets architecture

(a) Complete network

(b) Deep Sets block

ParticleNet

- Started from H. Qu's Keras version of ParticleNet
- Edge convolution
  - Begin with coordinates in pseudorapidity-azimuth space
  - Calculate k-nearest neighboring particles for each particle using the coordinates
  - \Edge features" are constructed from the constituent features using the indices of k-nearest neighboring particles
  - Feed into shared MLP to update each particle in the graph (in practice using convolution layers)
  - Perform permutation invariant aggregation, selected mean which is used in the ParticleNet paper
  - Subsequent EdgeConv blocks use the learned feature vectors as coordinates (hence dynamic)
- Concatenate with global features and feed into MLP

ParticleNet architecture

(a) Complete network

(b) EdgeConv block

Training

- Two models are trained
  - Deep Sets with 1.47M parameters
  - ParticleNet with 1.20M parameters
- Using TensorFlow 2.4.1
- MirroredStrategy on two Nvidia GeForce RTX 3090 cards
- Adam optimizer
- Batch size 1024
- Learning rate $2 \cdot 10^{-3}$, reduced by a factor of 5 when validation loss plateaus
- Regularization through early stopping callback

E ective data pipeline

Figure: Naive and parallel data handling in TensorFlow.

Loss

- Deep Sets
  - min training loss 0.0784
  - min validation loss 0.0792
- ParticleNet
  - min training loss 0.0776
  - min validation loss 0.0785

(a) Deep Sets loss

(b) ParticleNet loss

Results

all jet response

uds jet response

gluon jet response

b jet response

c jet response

avour di erence

Summary

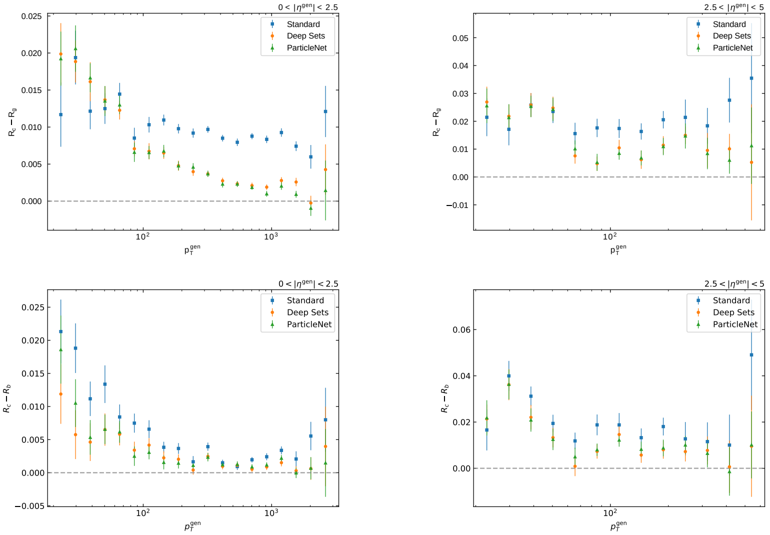
- Improved p_T resolution w.r.t standard corrections
  - 10-15% for uds jets, 10% for b & c jets and around 8% for g jets in the central region
  - 10-20% for uds jets and 5-20% for the rest of the jets in the forward region
- Reduced avour di erences
  - Factor of 3 improvement in central region and 30% in forward region
- ParticleNet vs Deep Sets
  - 270k less parameters in my ParticleNet model
  - Despite this ParticleNet achieves slightly better resolution, especially for jets with higher p_T
  - ParticleNet also has slightly less avour di erence for the response
  - However, Deep Sets has fewer GPU intense operations and is faster to train

Extra material

Residual response

Residual response

Residual response



CERN

home.cern