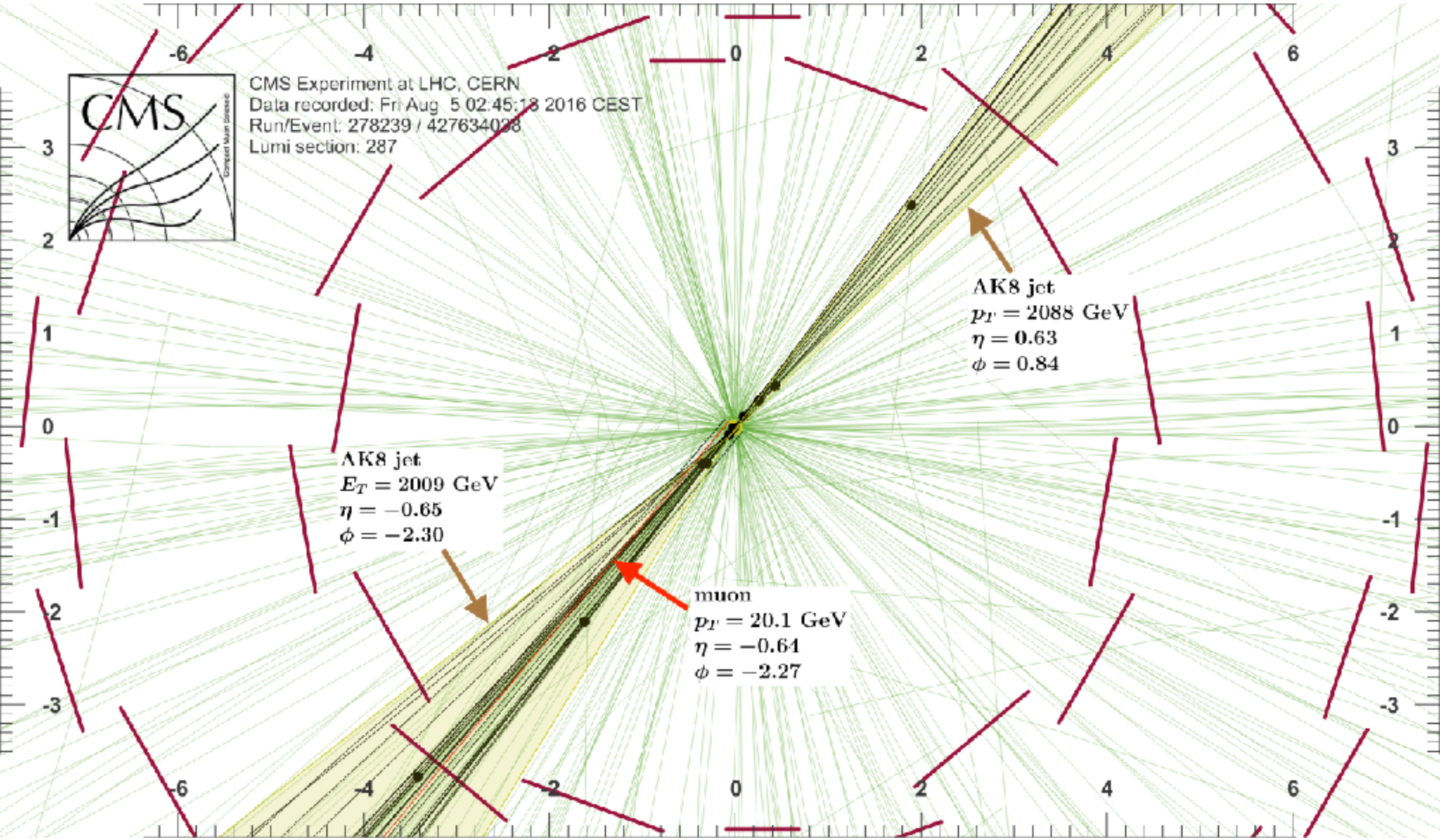
DeepJet: a deep-learned multiclass jet-tagger for slim and fat jets

Mauro Verzetti[1,2], Jan Kieseler[1], Markus Stoye[3], Huilin Qu[4], Loukas Gouskos[4]
on behalf of the CMS Collaboration

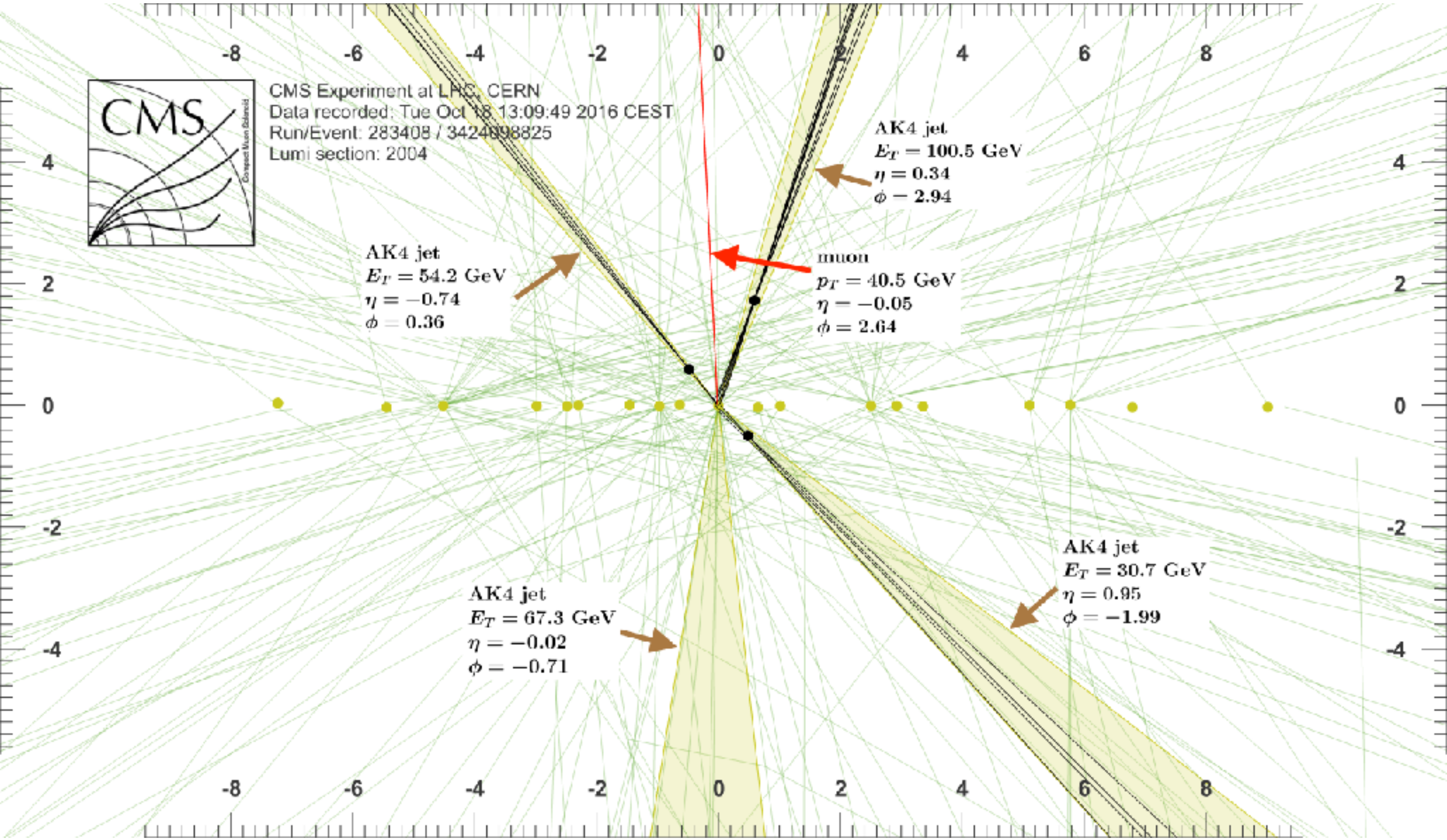
[1]CERN, [2]FWO, [3]Imperial College London, [4]UC Santa Barbara

The problem



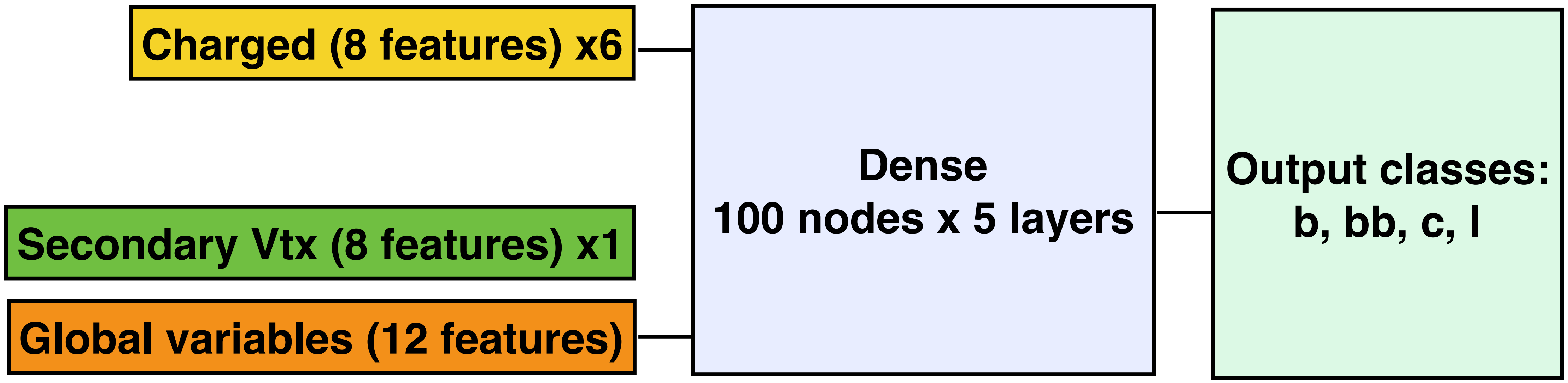
CMS Experiment at LHC, CERN
Data recorded: Fri Aug 5 02:45:18 2016 CEST
Run/Event: 278239 / 427634038
Lumi section: 287
AK8 jet
$p_T = 2088$ GeV
$\eta = 0.63$
$\phi = 0.84$
AK8 jet
$E_T = 2009$ GeV
$\eta = -0.65$
$\phi = -2.30$
muon
$p_T = 20.1$ GeV
$\eta = -0.64$
$\phi = -2.27$

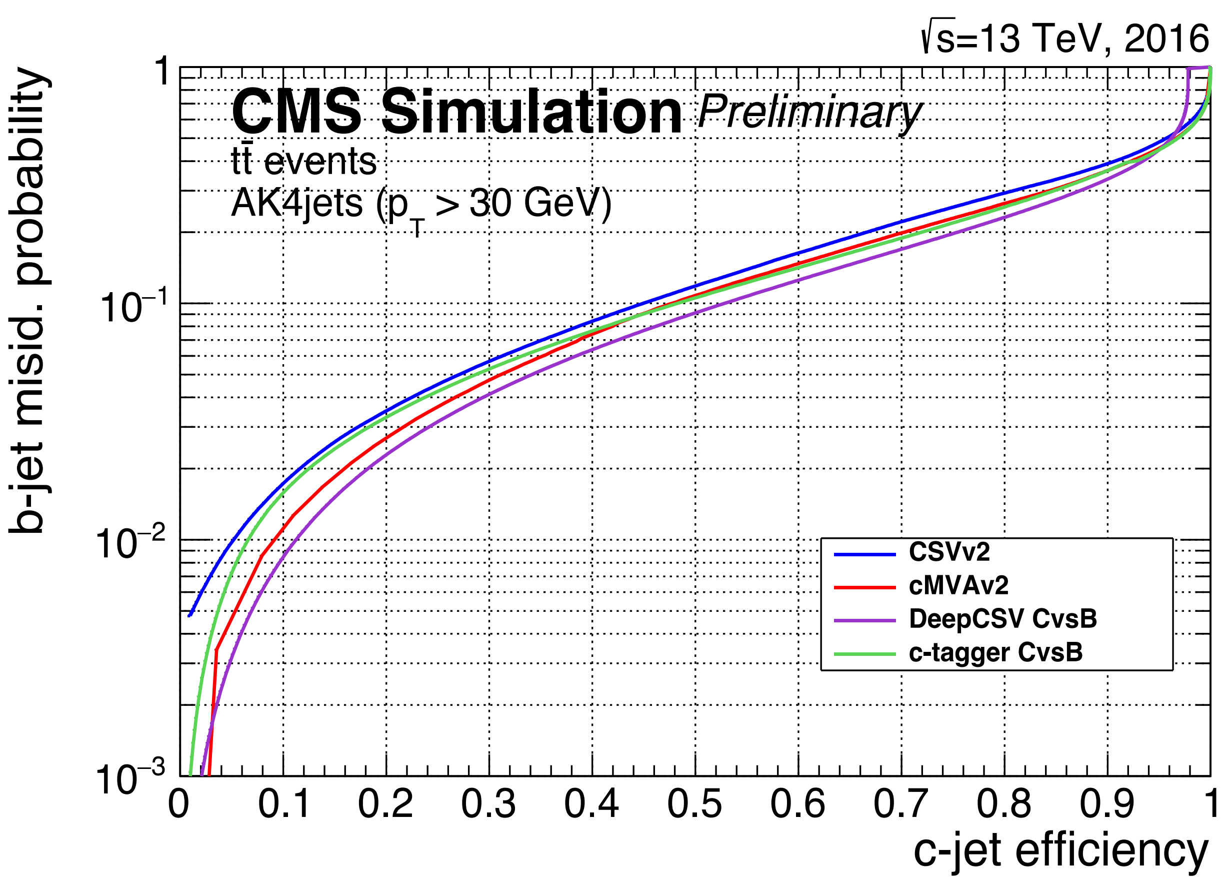
The problem

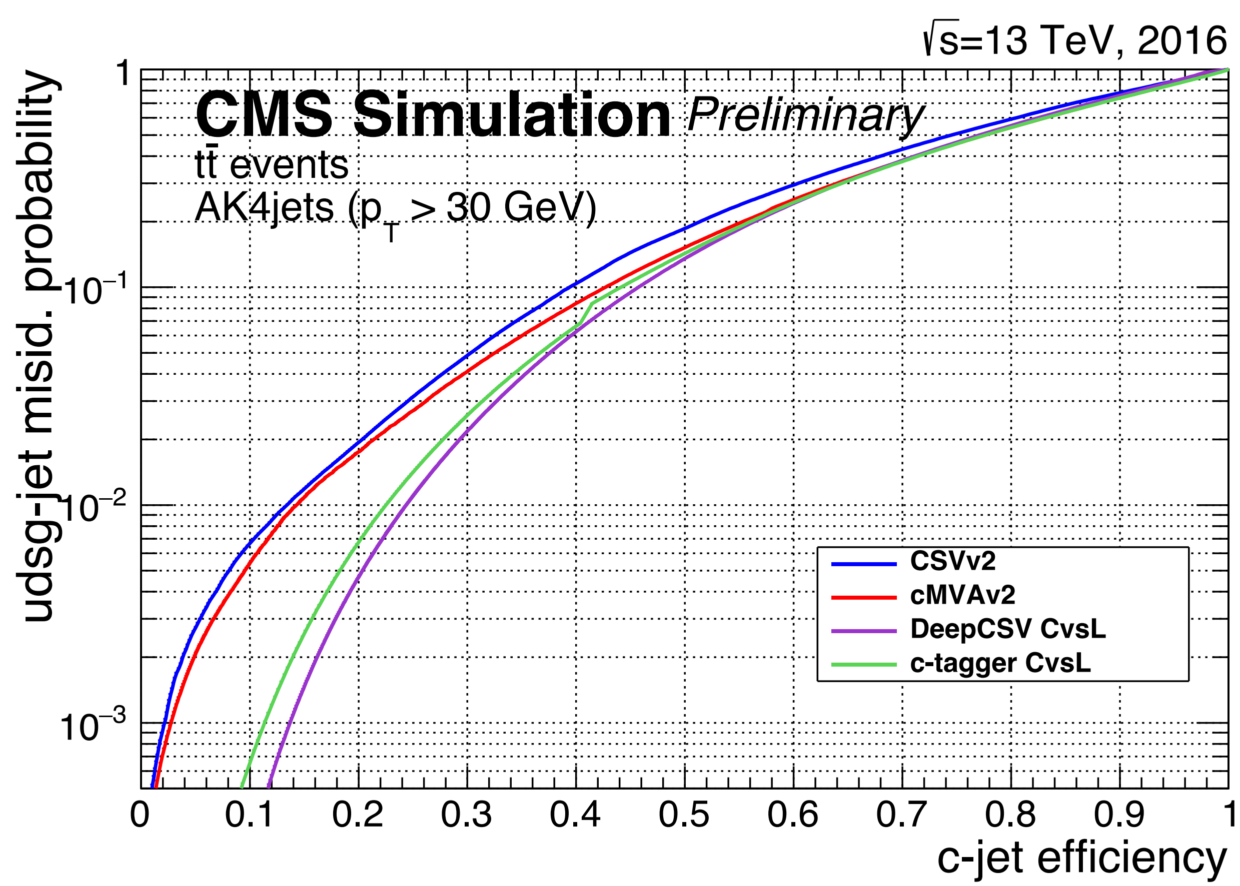


CMS Experiment at LHC, CERN
Data recorded: Tue Oct 18 13:09:49 2016 CEST
Run/Event: 283408 / 3424098825
Lumi section: 2004
AK4 jet
$E_T = 100.5$ GeV
$\eta = 0.34$
$\phi = 2.94$
AK4 jet
$E_T = 54.2$ GeV
$\eta = -0.74$
$\phi = 0.36$
muon
$p_T = 40.5$ GeV
$\eta = -0.05$
$\phi = 2.64$
AK4 jet
$E_T = 30.7$ GeV
$\eta = 0.95$
$\phi = -1.99$
AK4 jet
$E_T = 67.3$ GeV
$\eta = -0.02$
$\phi = -0.71$

Last year development — DeepCSV

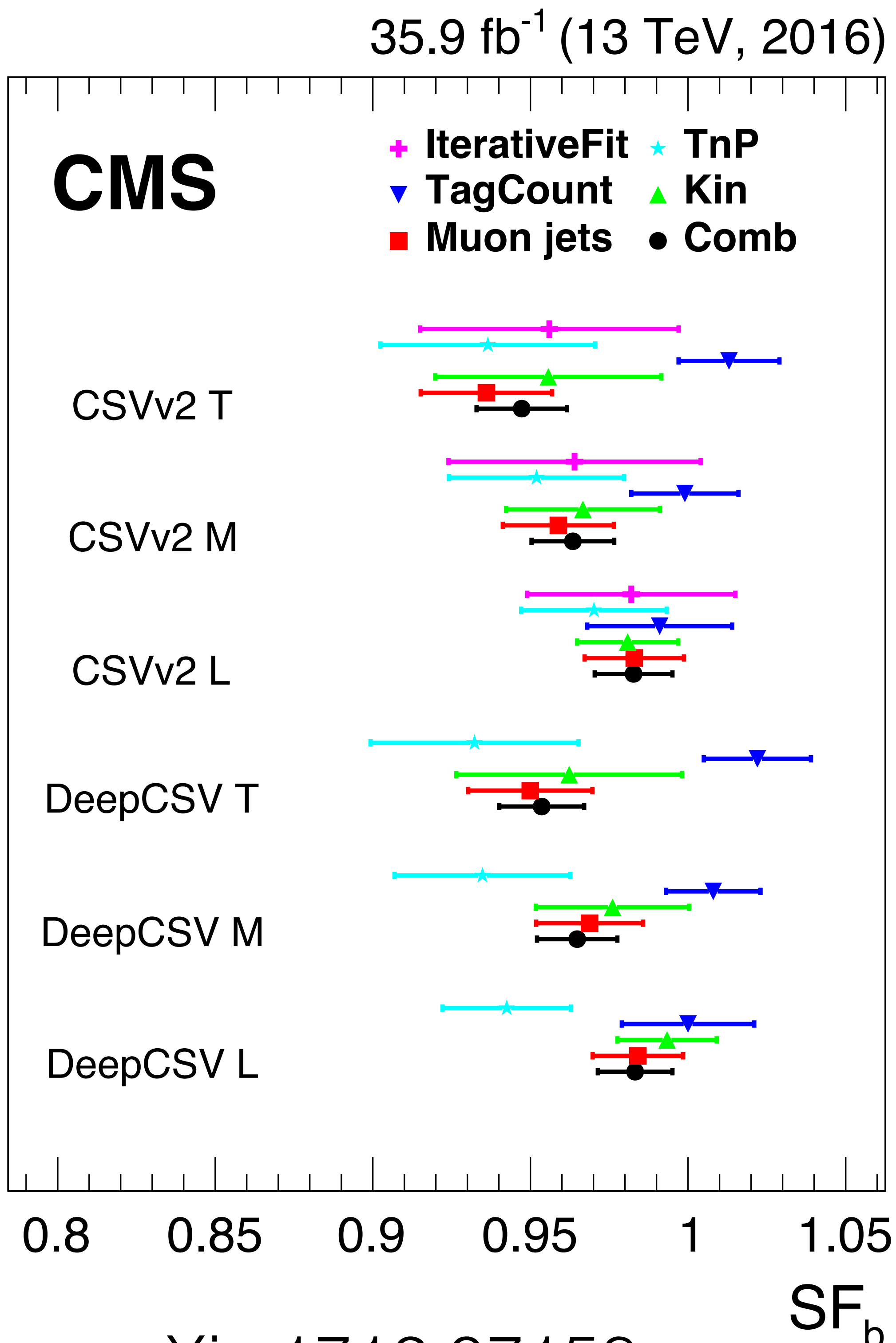


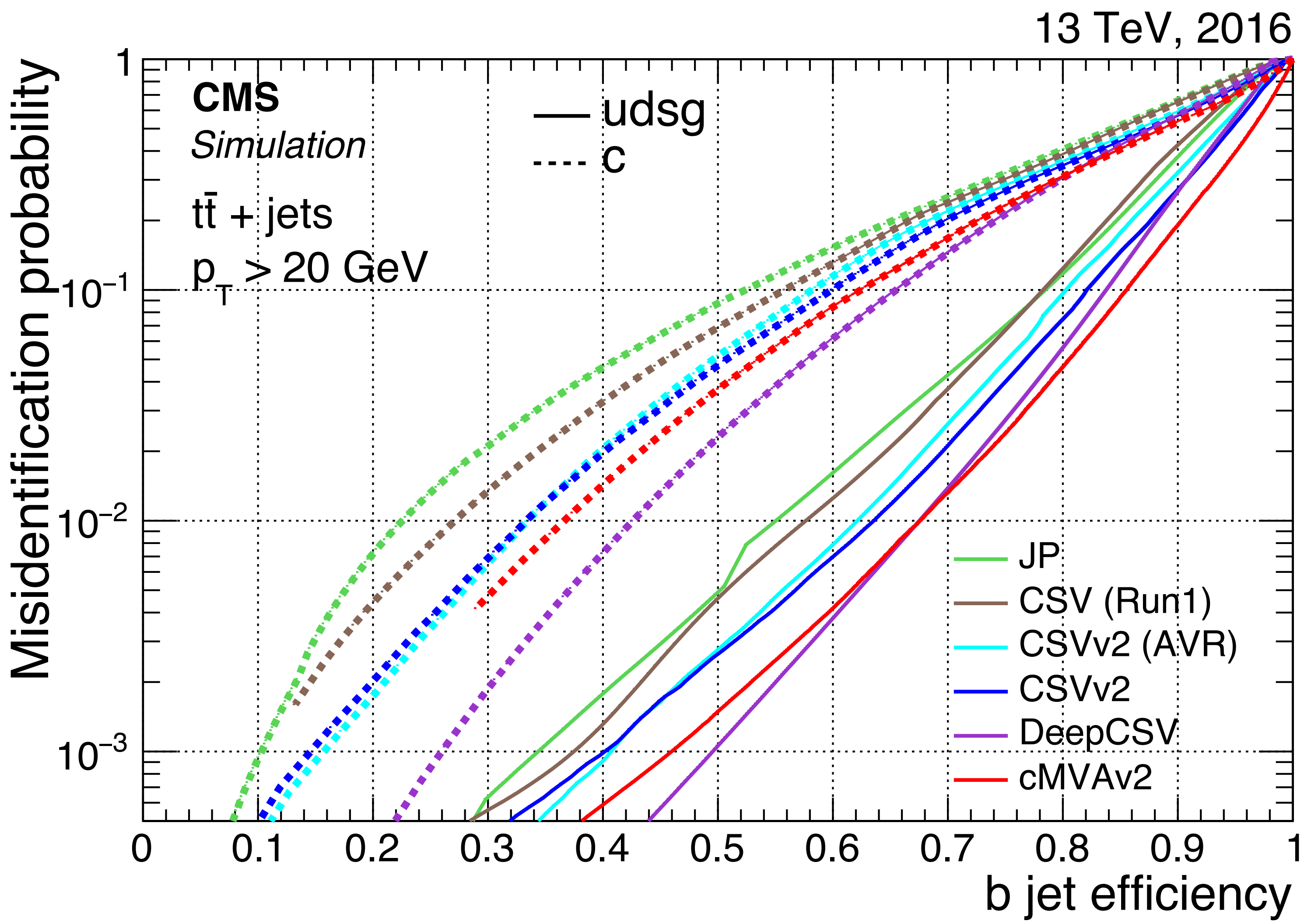




arXiv:1712.07158

Last year development — DeepCSV

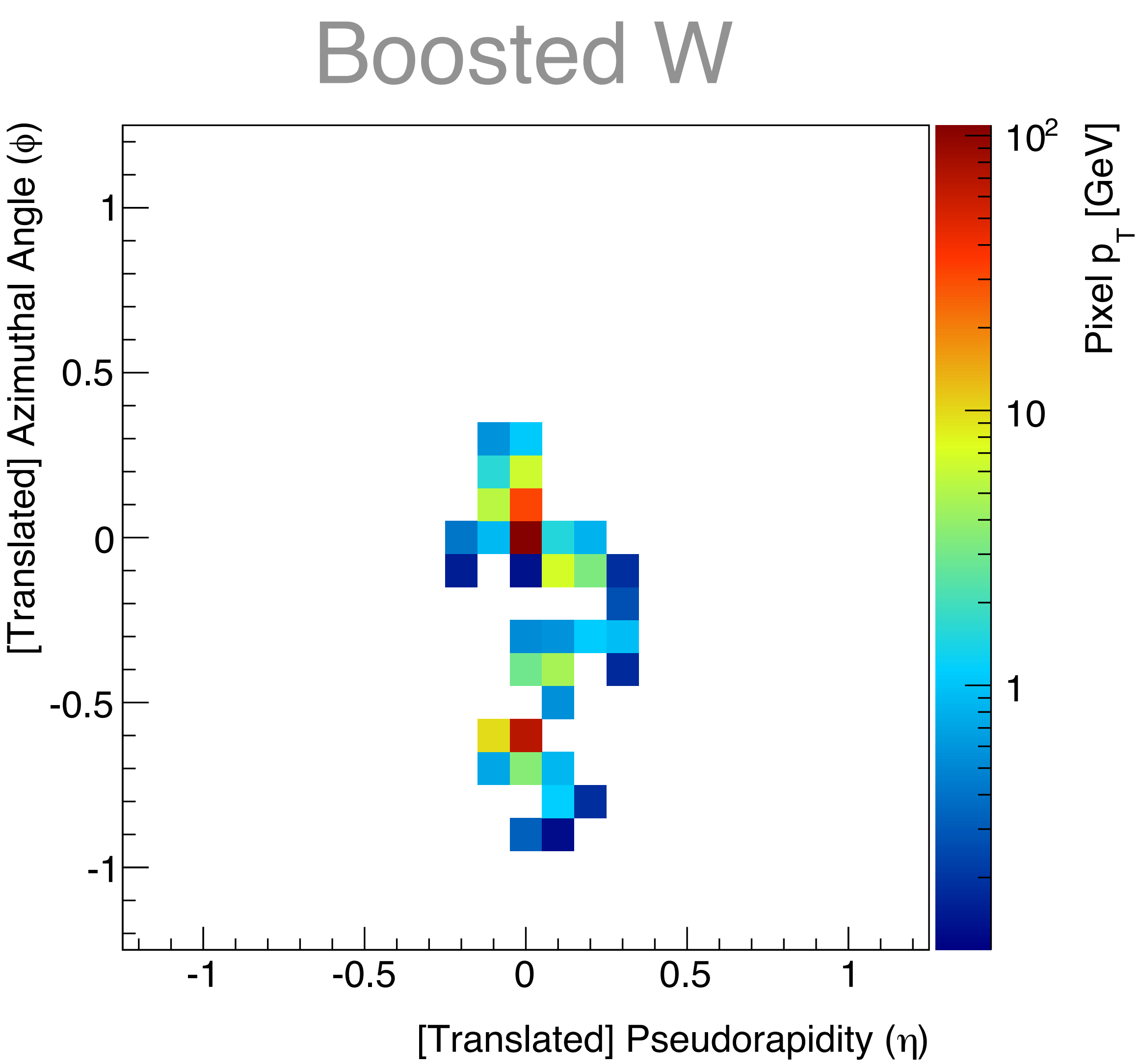


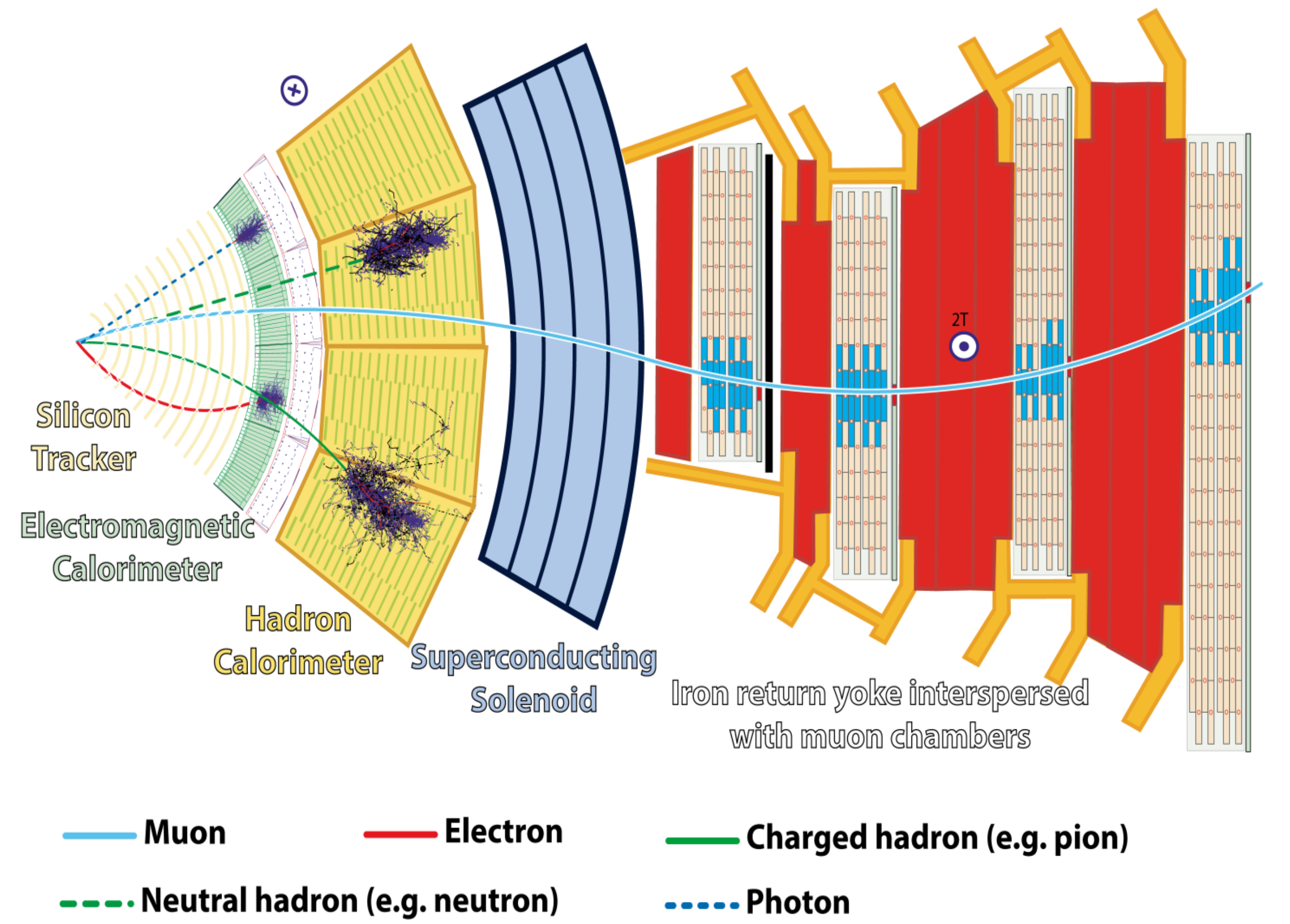


arXiv:1712.07158

Trying more complex architectures

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

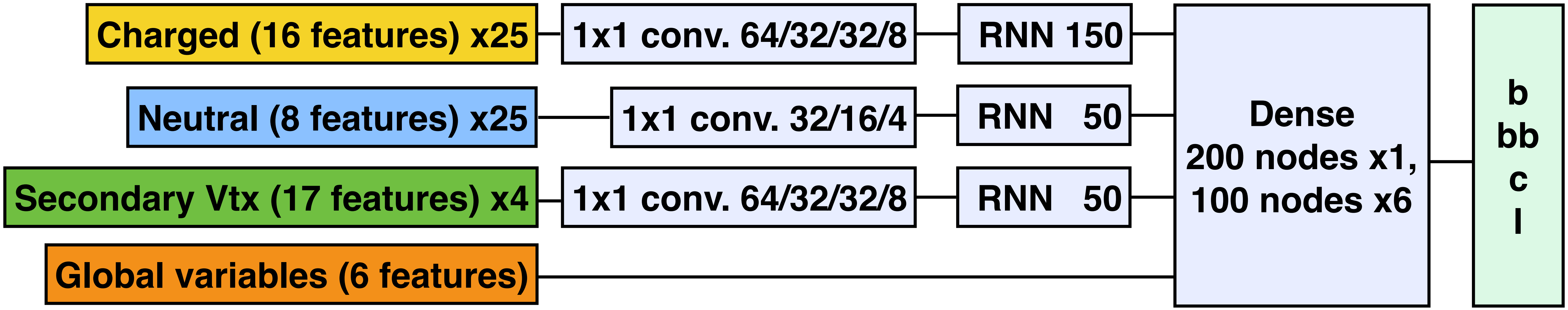




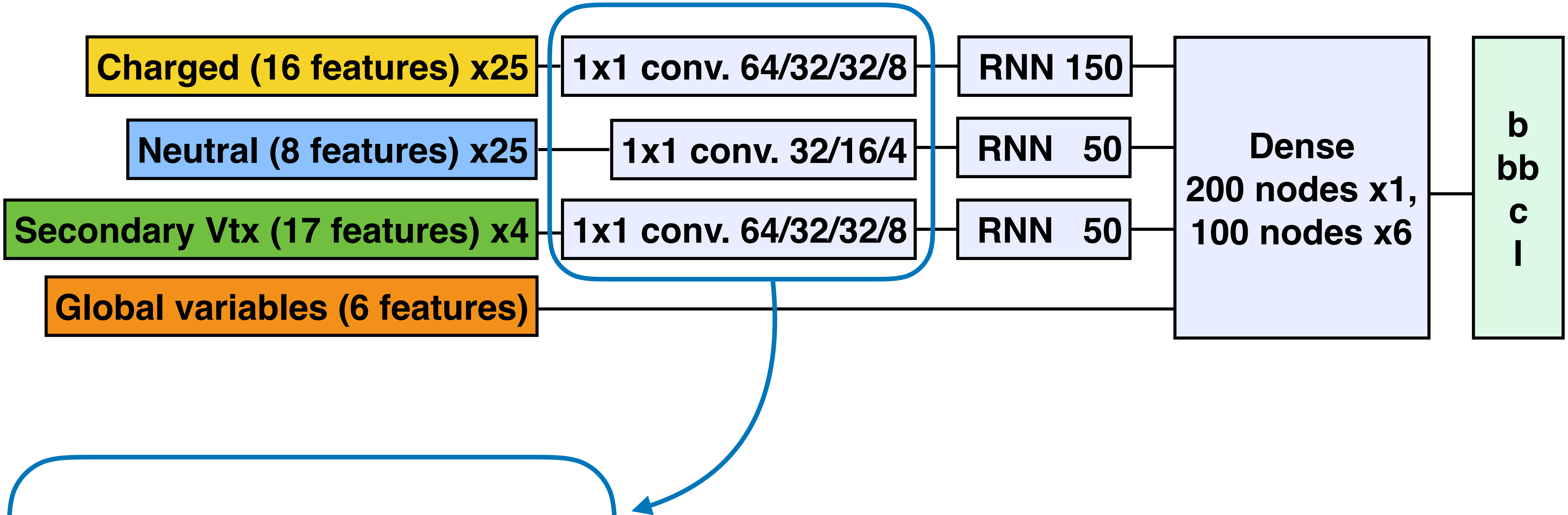
… but

- 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

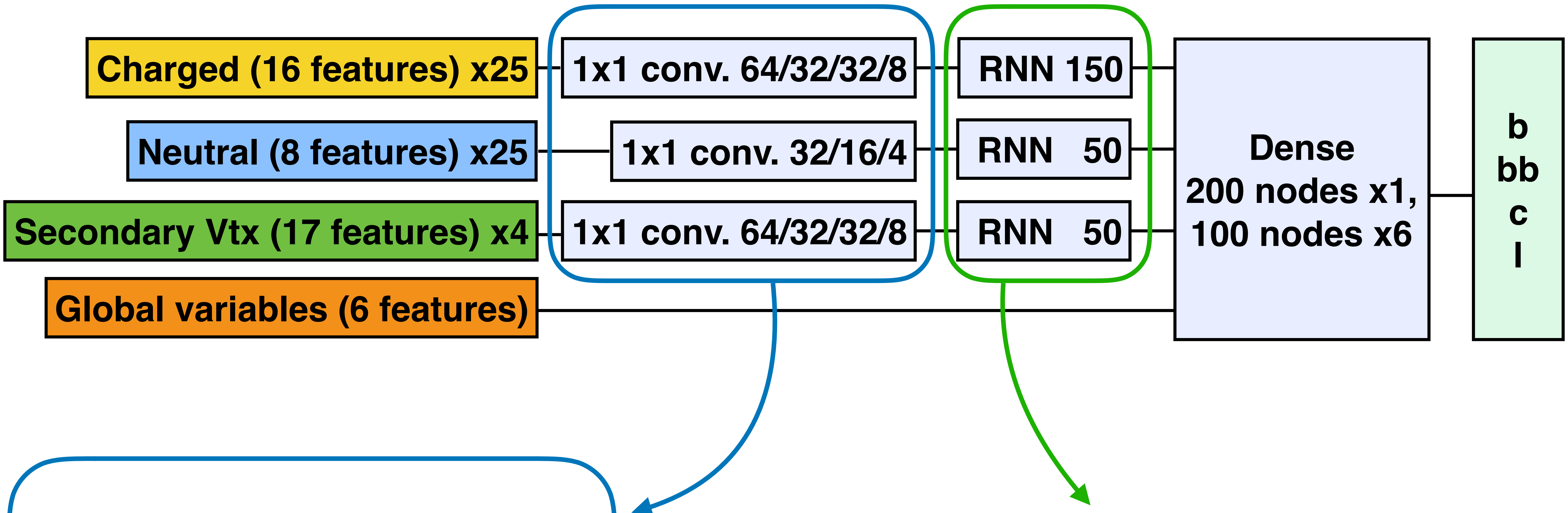


Particle-based NN architecture



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

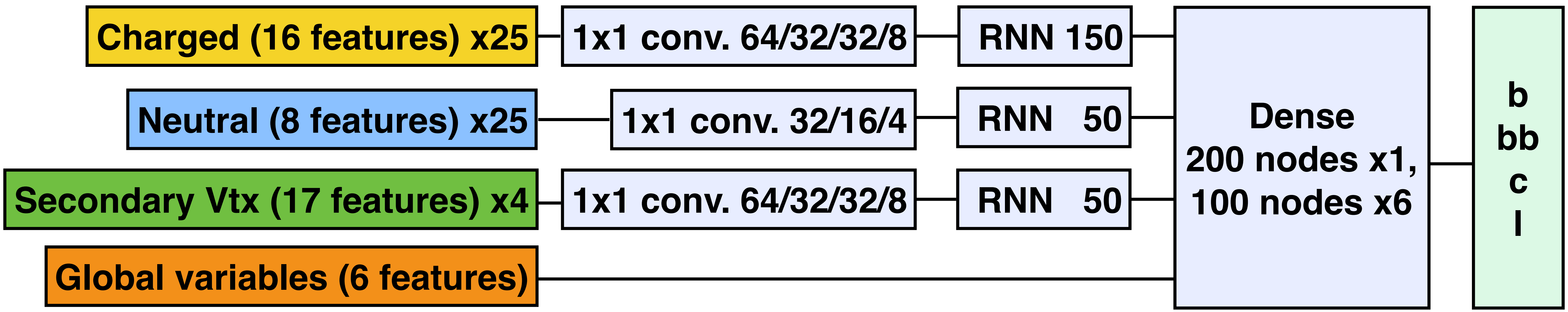
Particle-based NN architecture

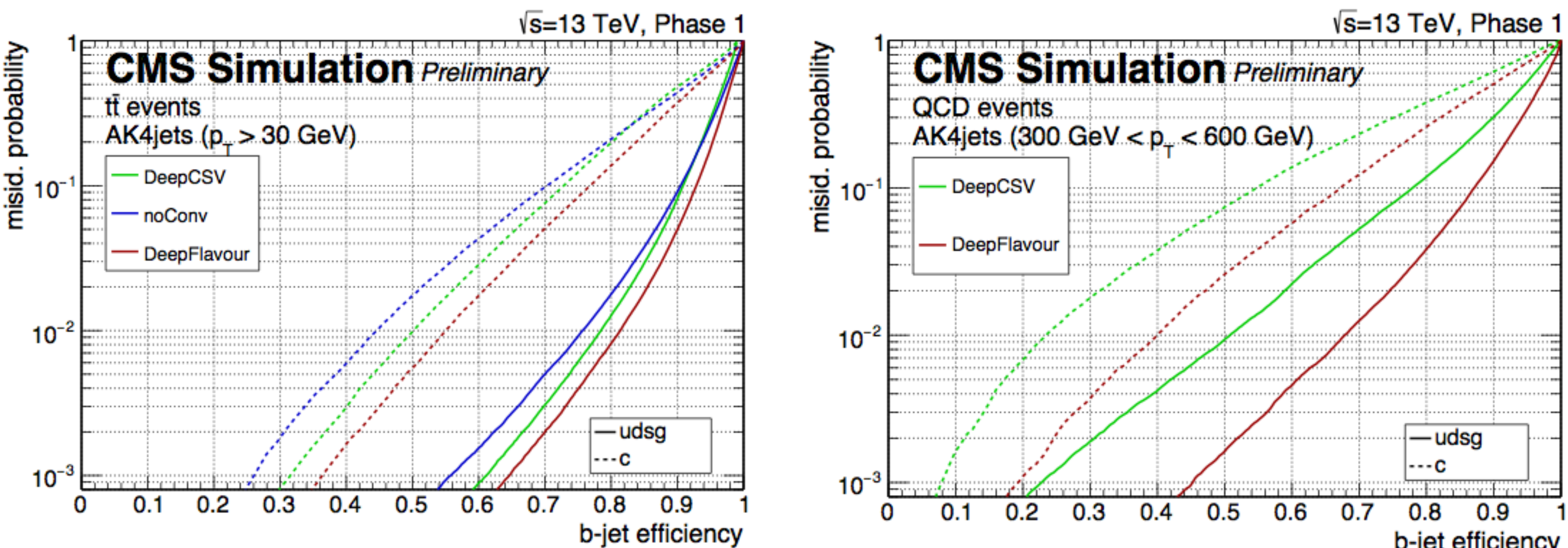


Convolutional layers progressively learn a more compact feature representation (automatic feature engineering)

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

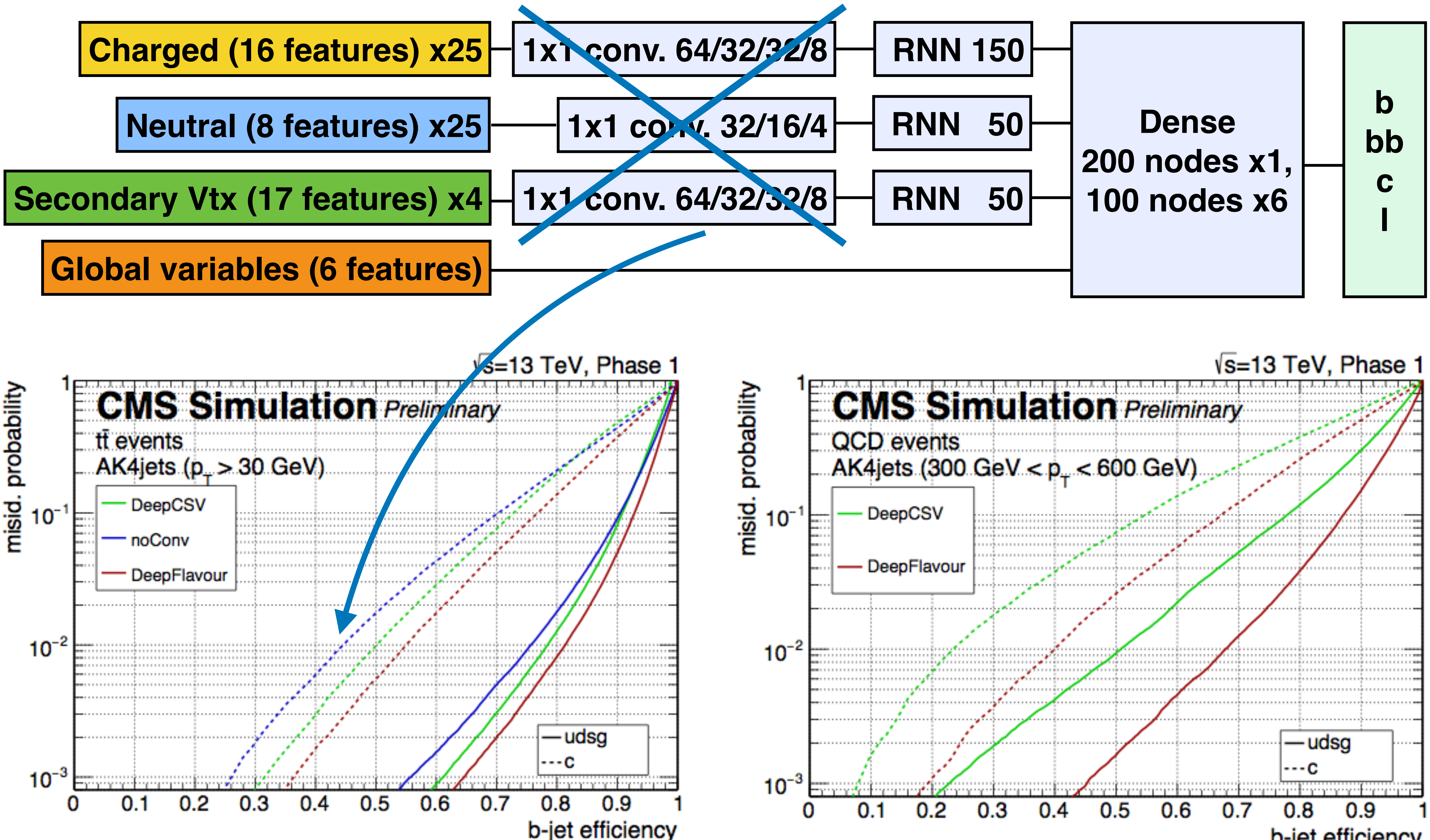
Particle-based NN architecture





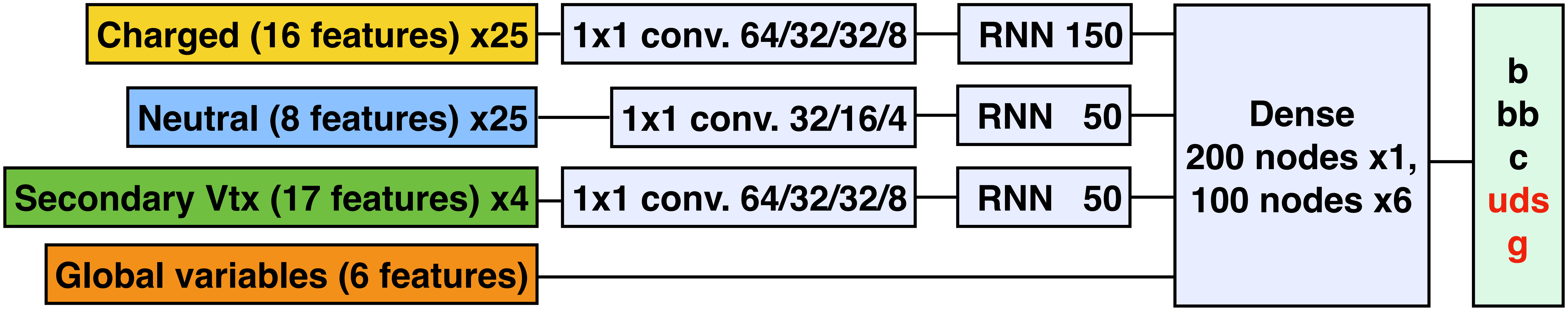
CMS-DP-2017-013

Particle-based NN architecture



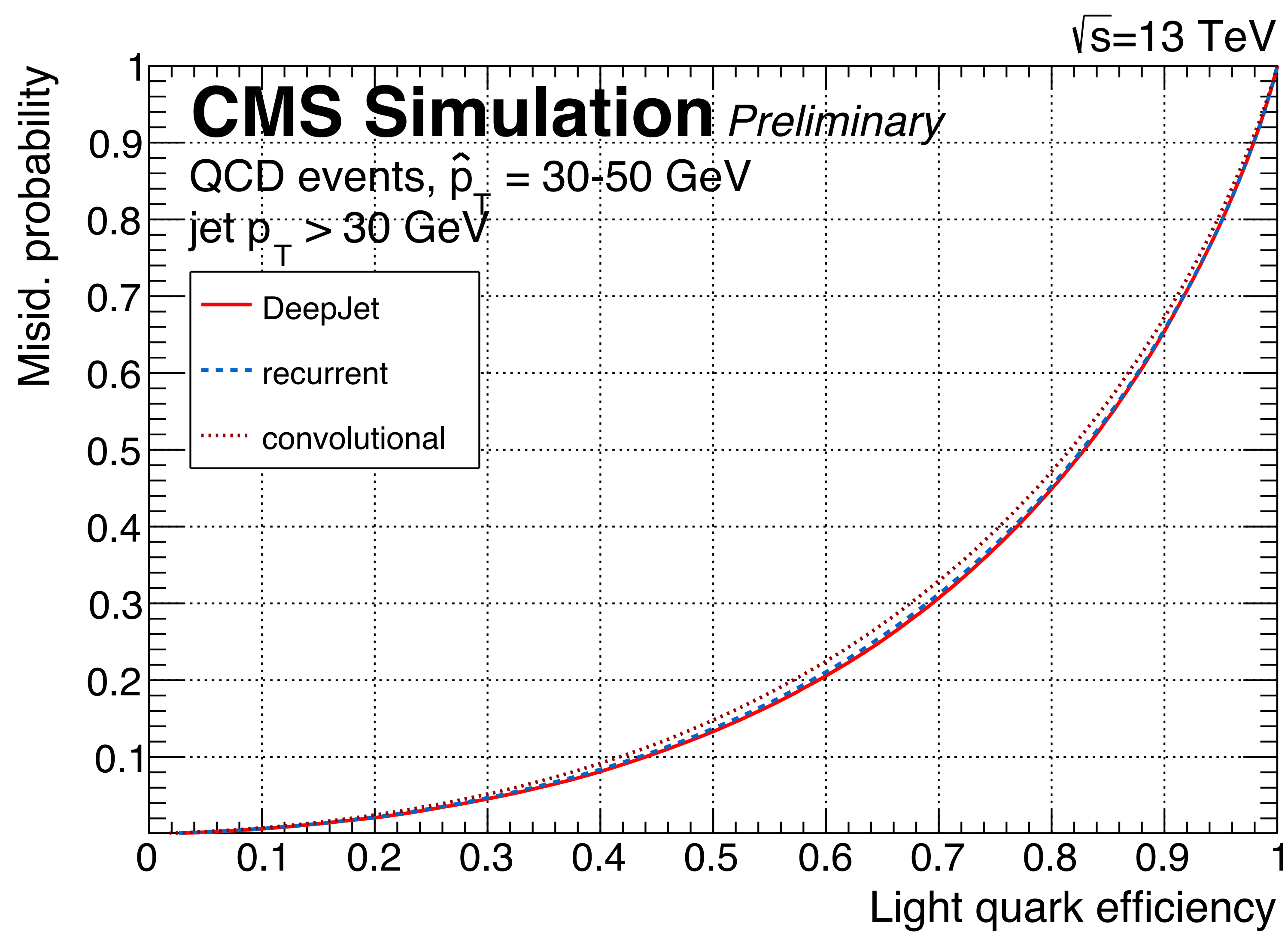
CMS-DP-2017-013

Particle-based NN architecture



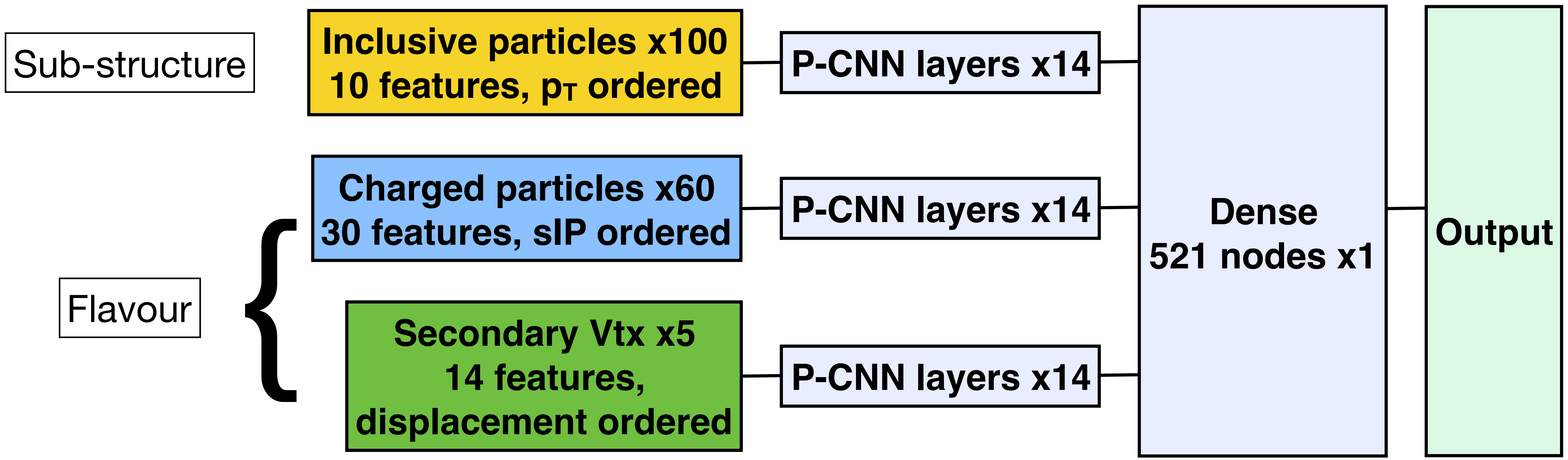
Similar performance to simpler, dedicated binary taggers, but with full multi-class power.

Significantly better performances in given regions with different quark composition



CMS-DP-2017-027

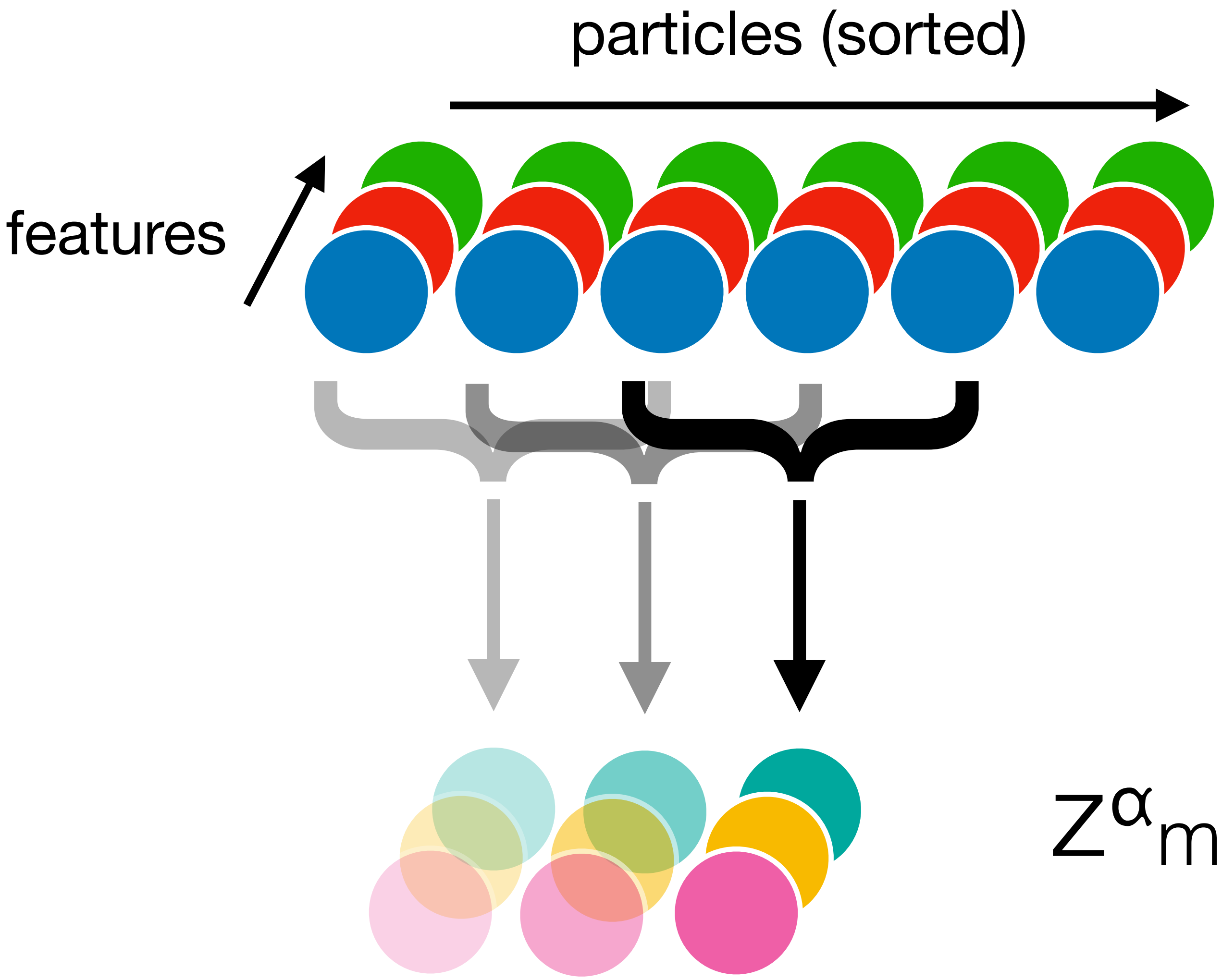
DeepJet for boosted resonances



- Significantly larger amount of candidates used to accomodate 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”

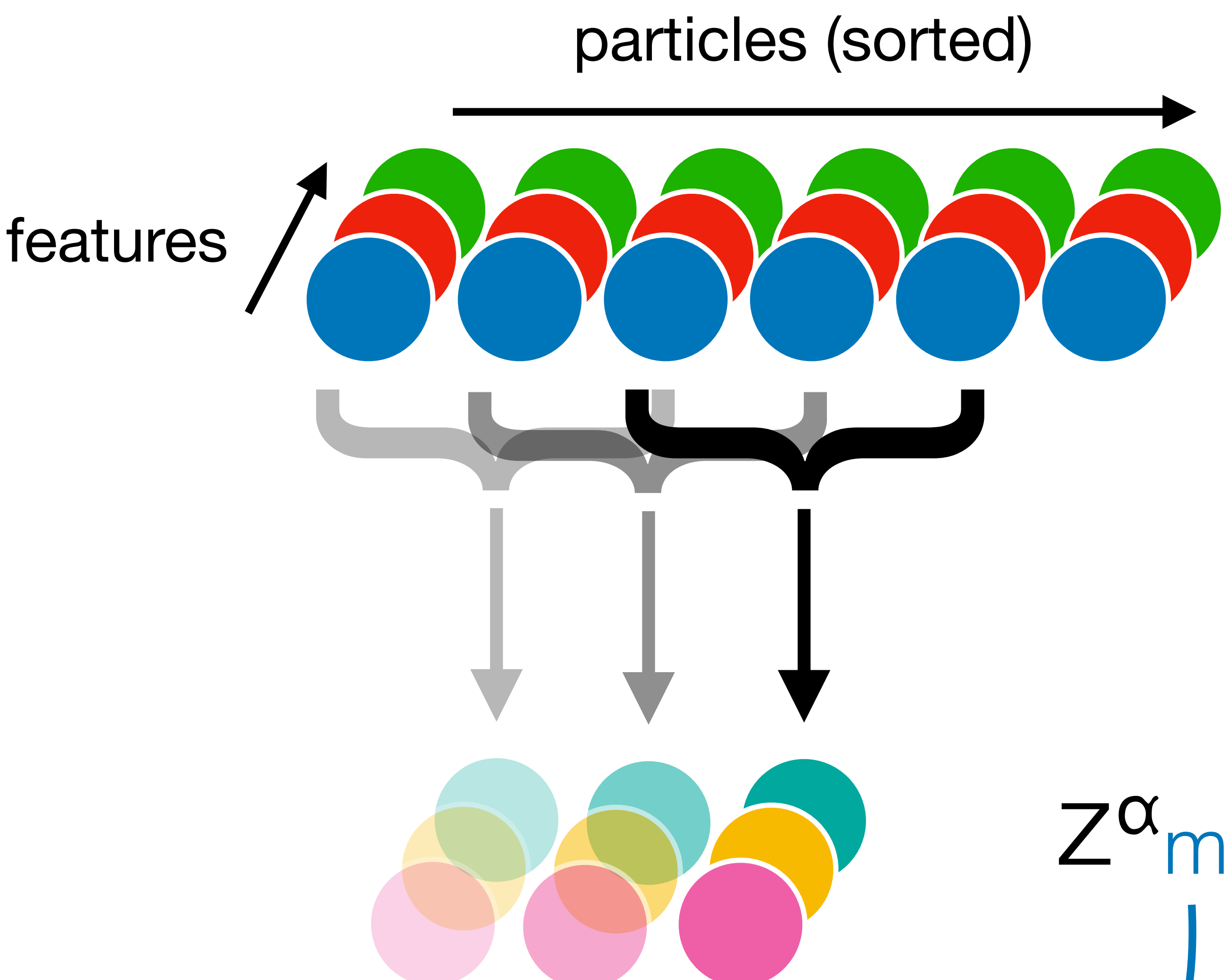
M. Verzetti (CERN and FWO)

P-CNNs



$$z^{\alpha}_{m} = \Sigma_a \Sigma_j \, k^{\alpha}_{a,j} \, x_{a,(m+j-1)}$$

P-CNNs

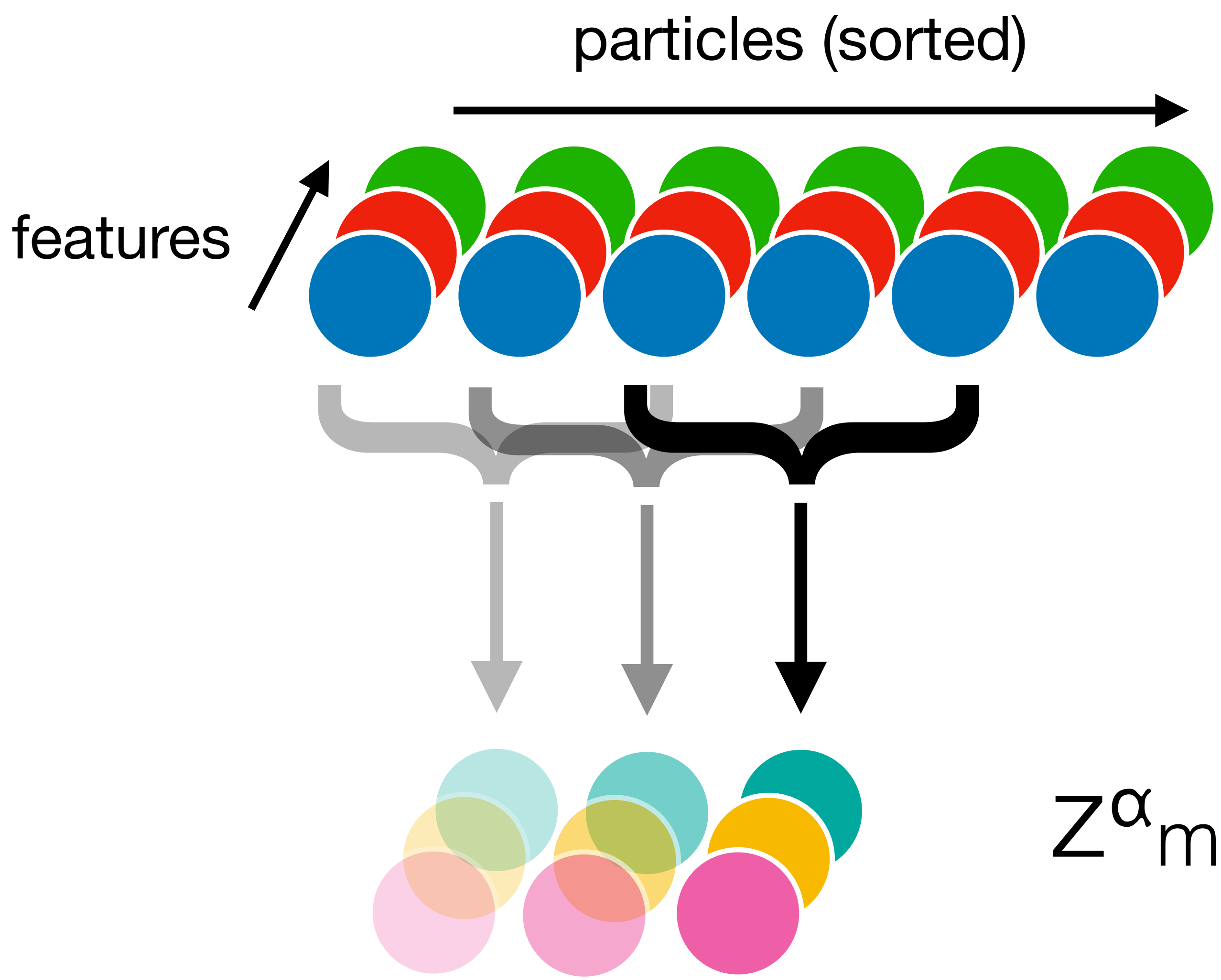


$$z^{\alpha}_{m} = \Sigma_a \Sigma_j k^{\alpha}_{a,j} \, x_{a,(m+j-1)}$$

Sweep over the elements

Loop over contiguous elements of the kernel

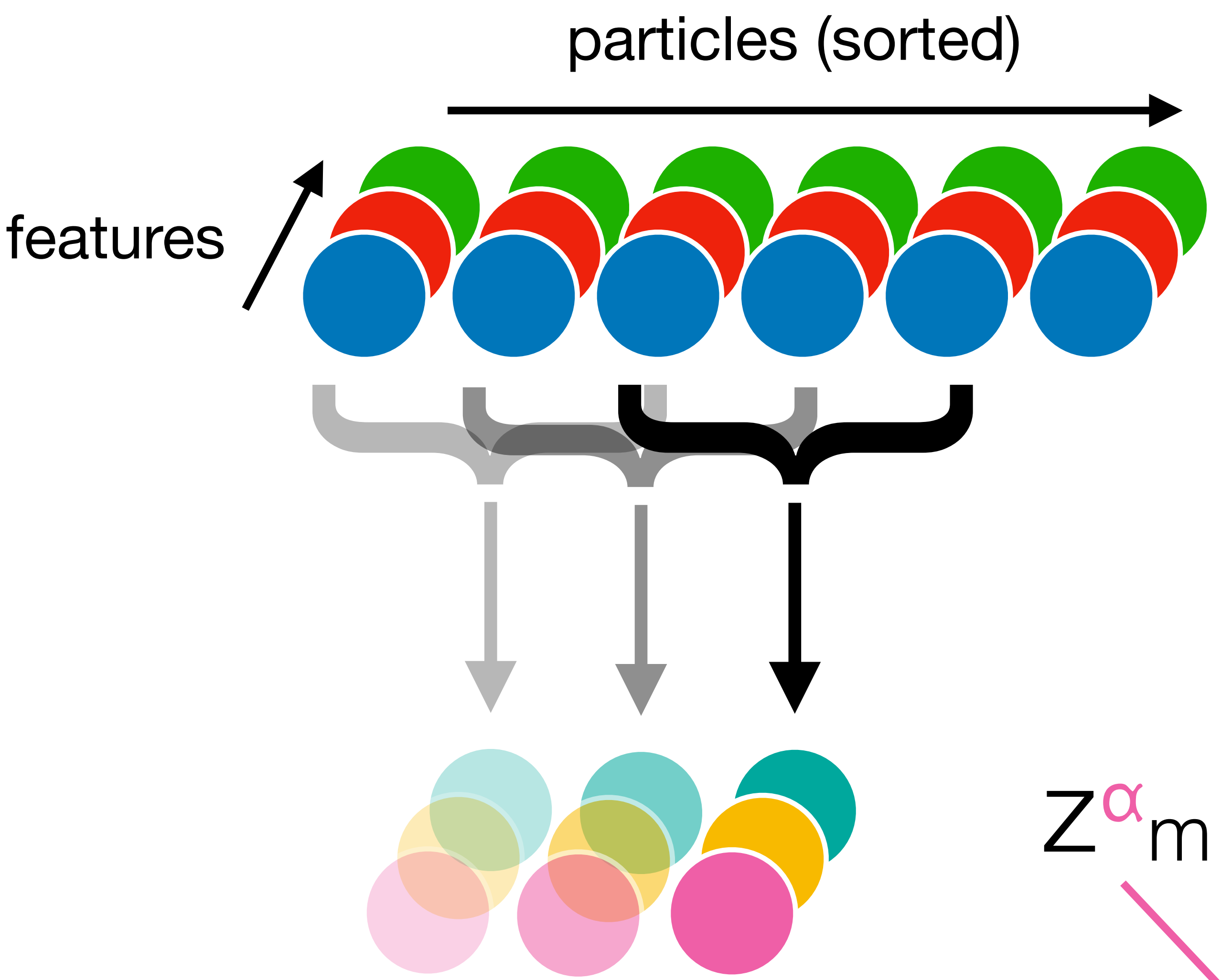
P-CNNs



$$z^{\alpha}_{m} = \sum_{a} \sum_{j} k^{\alpha}_{a,j}\, x_{a,(m+j-1)}$$

Multiple features (“colours”) are accounted computing the transformation

P-CNNs

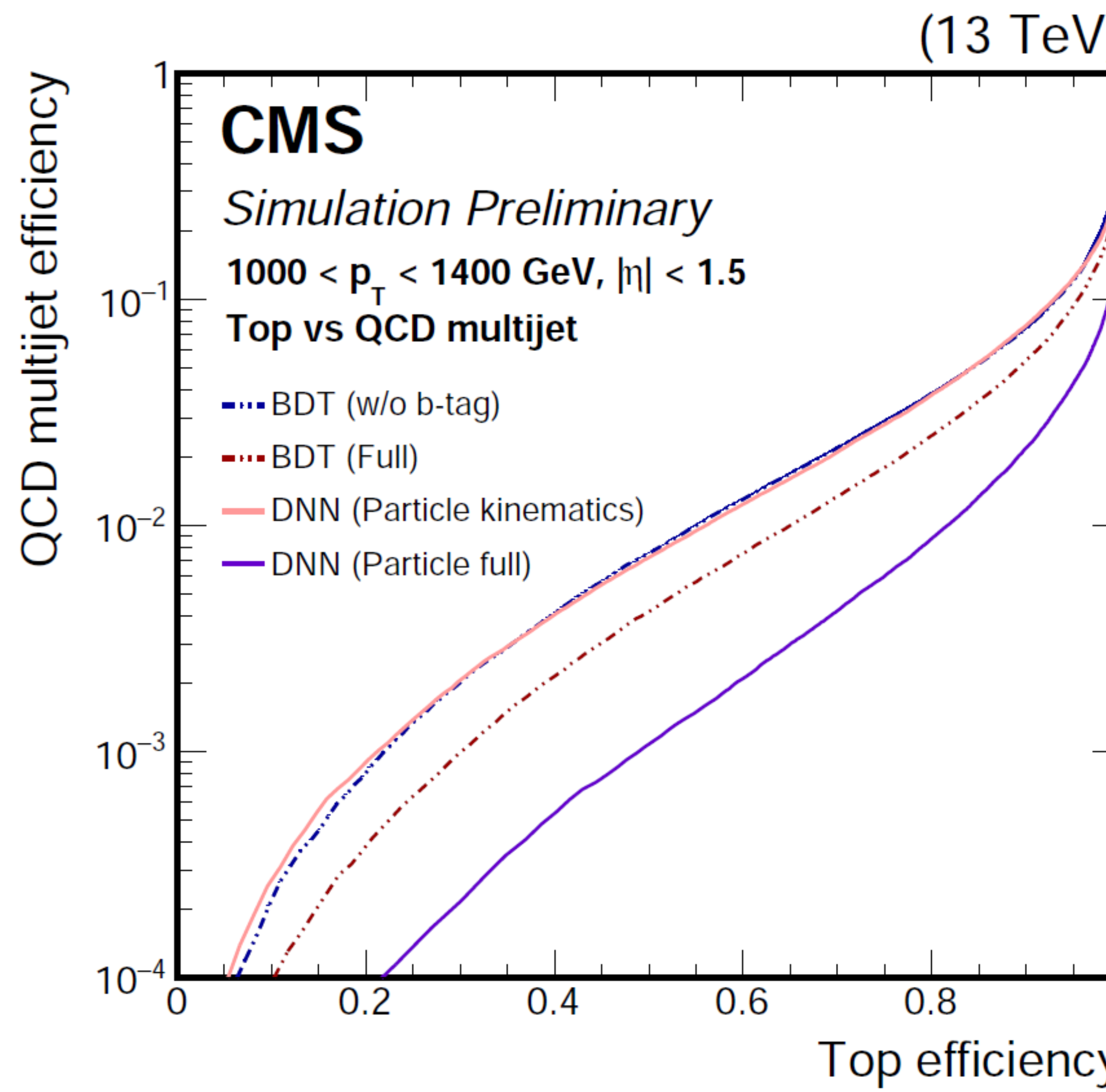


$$z^{\alpha}_{m} = \Sigma_a \Sigma_j \, k^{\alpha}_{a,j} \, x_{a,(m+j-1)}$$

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

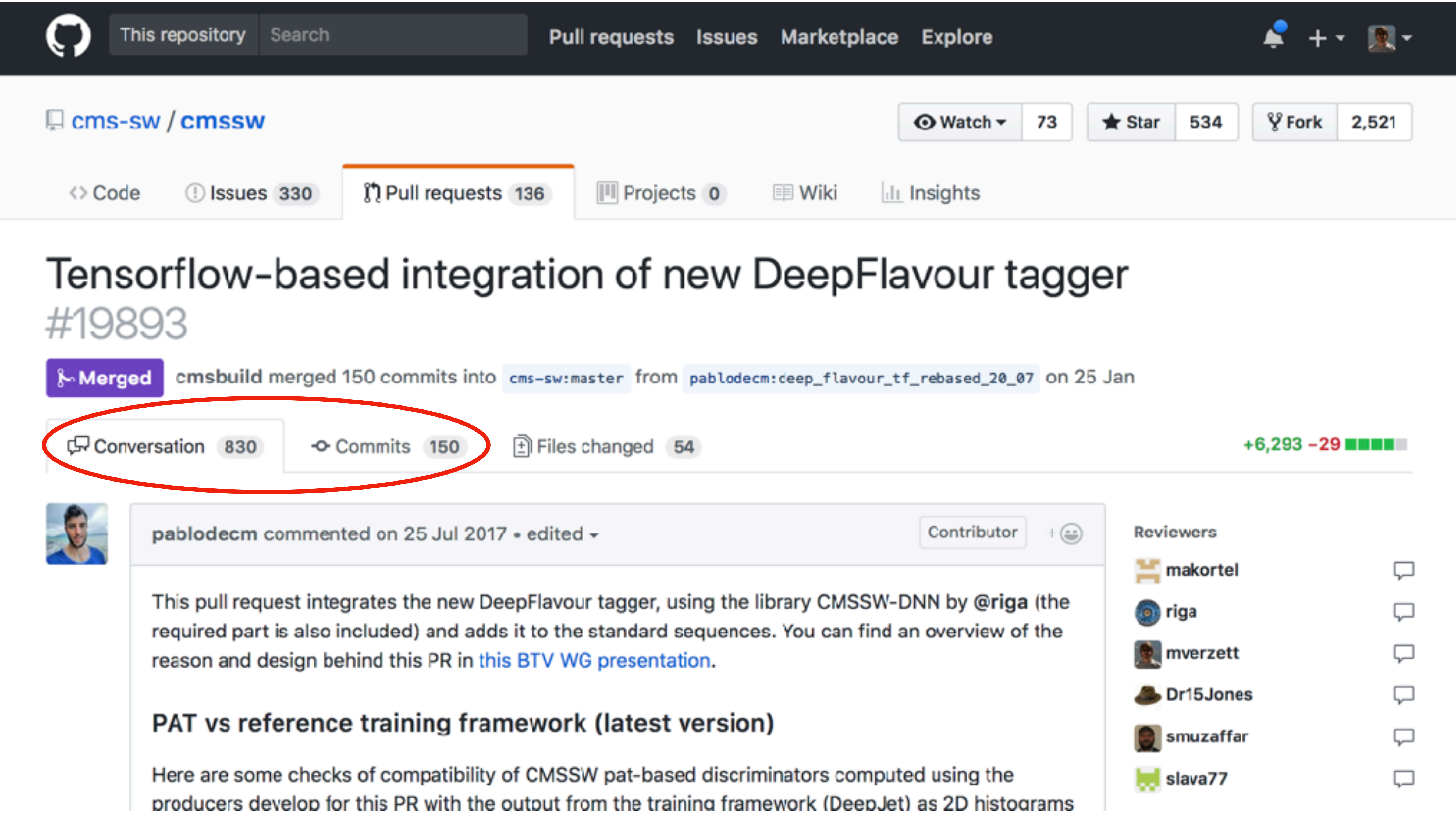
Training / Analysis:

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

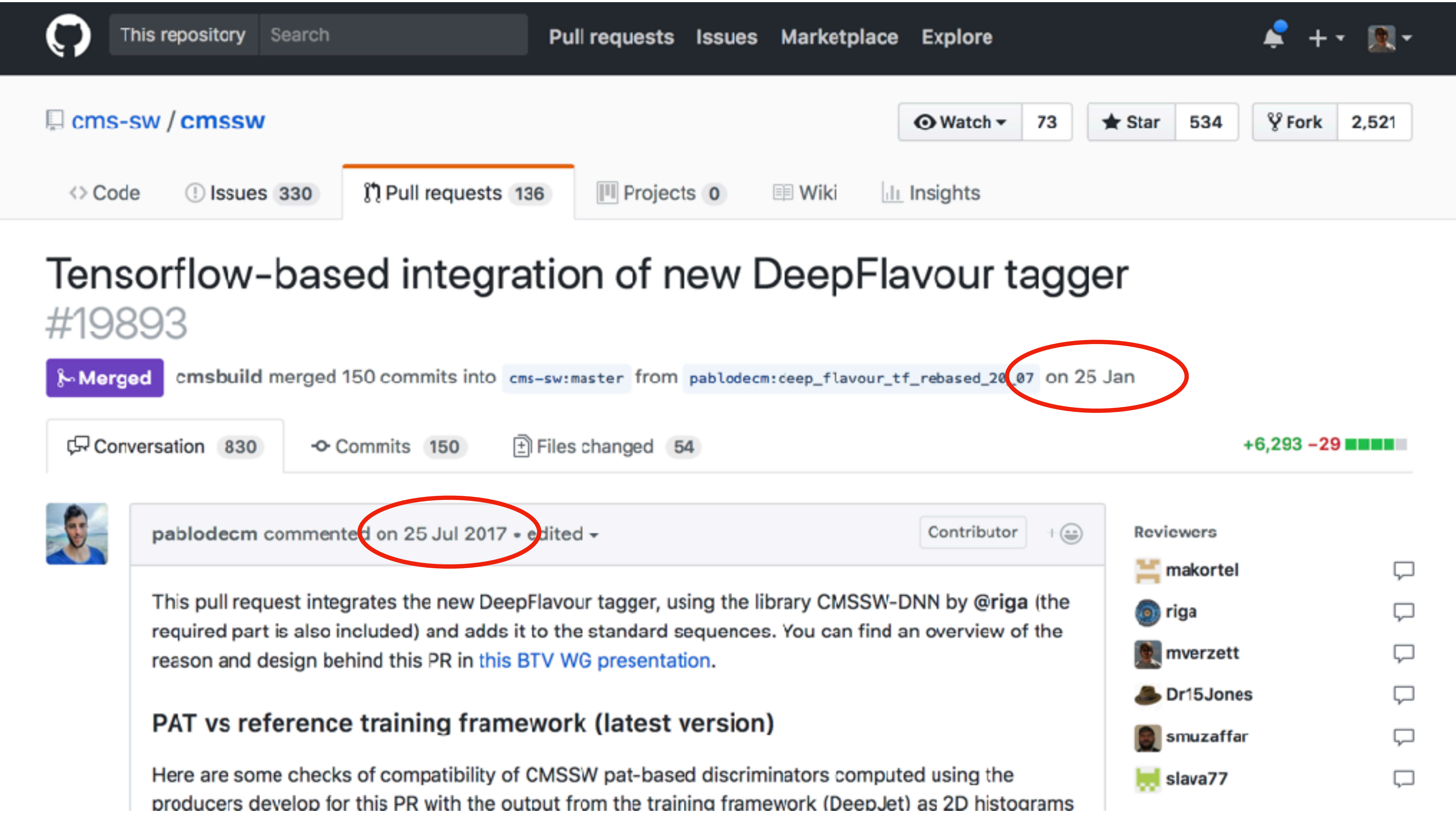
Production:

- 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



Integration of DeepJet (AK4) into CMSSW. PR #19893



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: <u>without threading</u> and <u>sharing the threading pool</u>

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