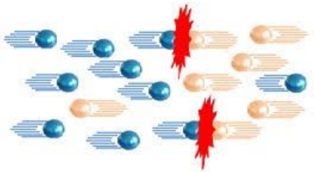
Machine Learning and b-tagging in CMS

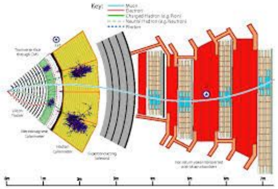
Introduction to b-tagging and exercises with DNNs

SNS - SCIENTIFIC DATA ANALYSIS SCHOOL

28/11/2019

Reminder / introduction





proton proton collision

Detector / trigger

Detector / offline reco

pass trigger

Offline reconstruction + analysis

local reco

> tracks + single det. objects

> single particles

> composite objects (jets)

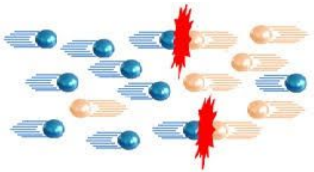
> tagging / common analysis techniques

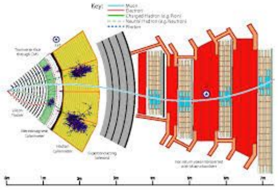
> physics analysis

Multiple levels of access to data

- Machine Learning can help at all levels
- Deep Learning can handle multiple levels

Reminder / introduction





proton proton collision

Detector / trigger

Detector / offline reco

pass trigger

Offline reconstruction + analysis

local reco

> tracks + single det. objects

> single particles

> composite objects (jets)

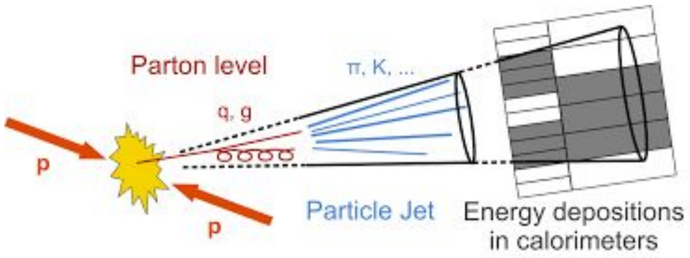
> tagging / common analysis techniques

> physics analysis

b-tagging at this level - thanks to deep learning we can use lower level information

What is (jet) b-tagging ?

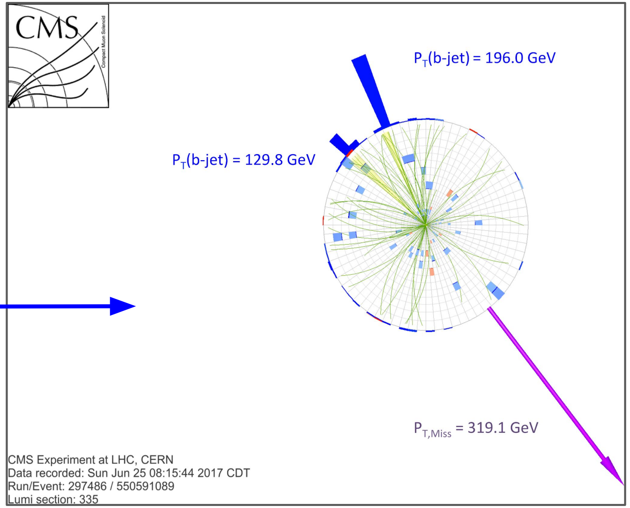
It is the identification (or "tagging") of jets originating from bottom quark



- **So what is a jet?**
  - A collection of collimated particles originating from the hadronization of a quark or a gluon
  - Clustering particles and detector signal in jets is the way we reconstruct the originating partons

- **Why b-jet tagging?**
  - Jets production is one of the most common processes at the LHC and a background for many analyses
  - b-jets production is suppressed compared to light quark/gluon jets
  - Final states with b-jets are interesting for many analyses:
    - Top quark
    - H-> bb
    - HH (bb+XX)
    - etc.

Z(vv) H(bb):
2b jets + Neutrinos



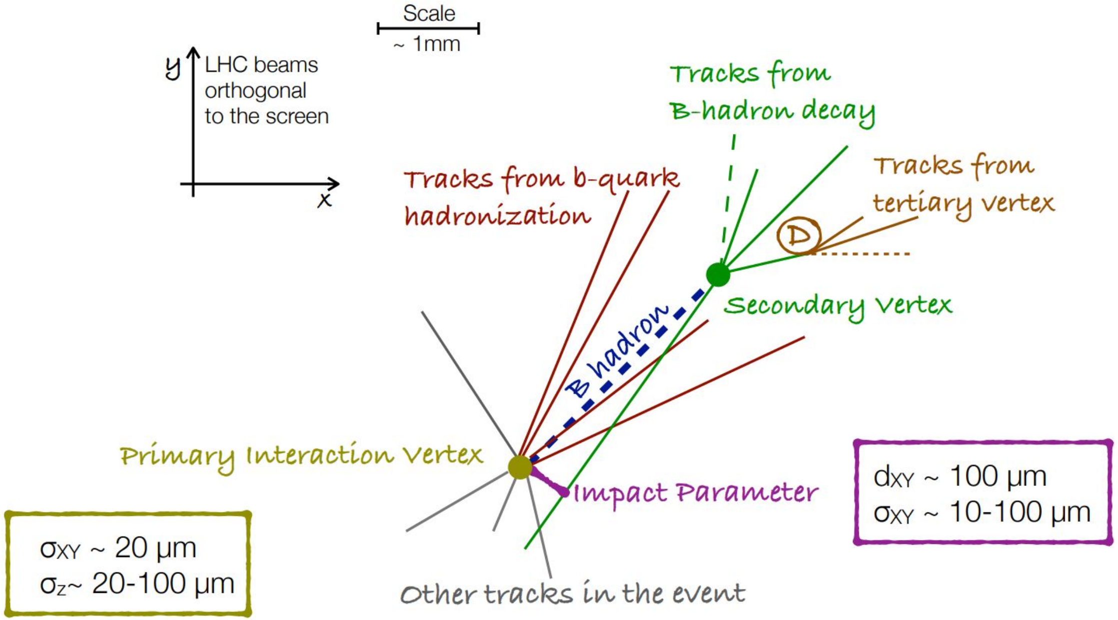
b jet properties

b-jets contain B hadrons

- **sizeable lifetime** (cτ ~ 500 μm) decay length of a few mm when boosted
  - Significant Impact Parameter (IP)
  - Secondary vertex

- Large mass (5 GeV)
- High rate of semileptonic decays (25%)
- High momentum transfer to the B hadron

b-tagging picture

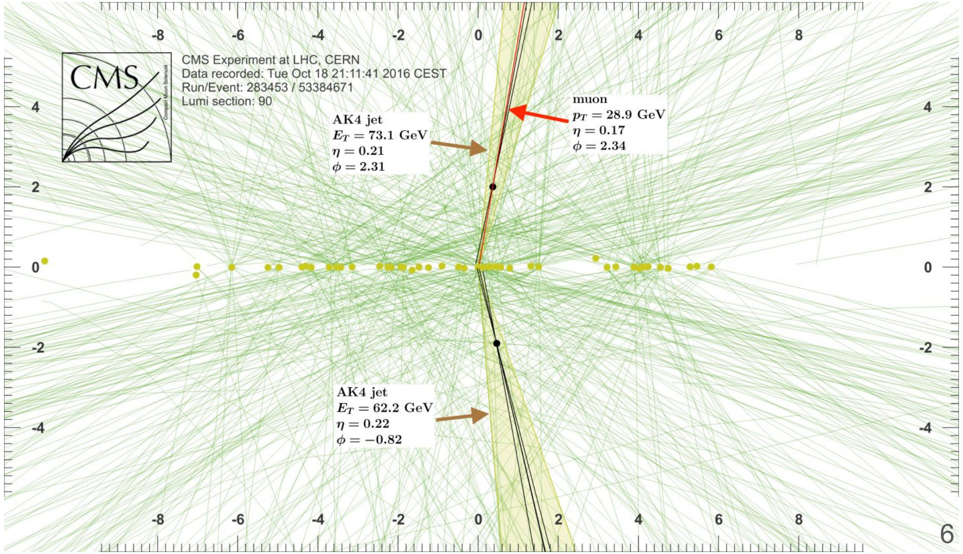


How is b-tagging done?

b-tagging relies mostly on the reconstruction of the B hadrons decay products:

- Efficient and robust tracking needed
- Displaced tracks
  - with good IP resolution
- Secondary vertex reconstruction
- The picture is not as simple as outlined

More realistic b-tagging picture



How is b-tagging done?

b-tagging relies mostly on the reconstruction of the B hadrons decay products:

- Efficient and robust tracking needed
- Displaced tracks
  - with good IP resolution
- Secondary vertex reconstruction
- The picture is not as simple as outlined

Pileup in pp collisions:

- Noisy environment
- Displaced “noise” tracks
- Critical point: jet-track association

We have to deal with:

- Uncertainty in track reconstruction
- Poor IP resolution
- SV inefficient reconstruction

b-tagging algorithms

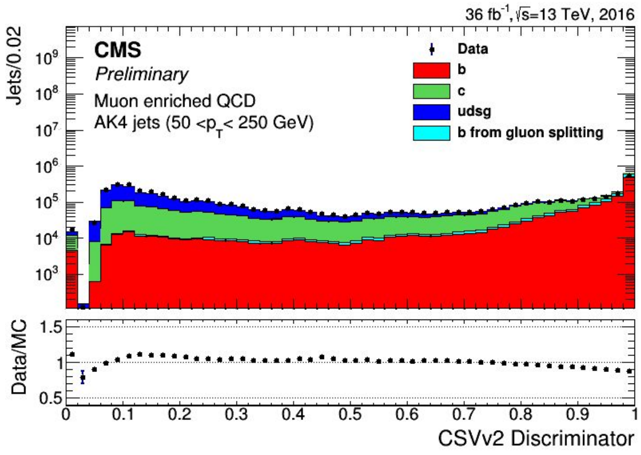
- **Can use single discriminating variables**
  - Tracks IP, Secondary vertices

- **Can combine several discriminating variables with ML**
  - ML is used to combine the information in an optimal way -> better performance
  - ML techniques are also more robust under different conditions (pileup, tracker detector, tracking etc.)

- **With Deep Learning we can also bypass some of the choices we make before optimization**
  - using lower level inputs
  - It can be more flexible and ultimately better performing

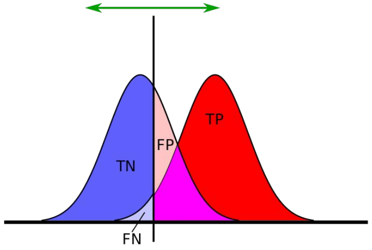
In ML b-tagging is a supervised classification problem

Example ML discriminator

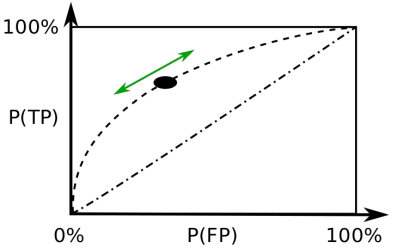


Benchmarks - some of the CMS standard algorithms

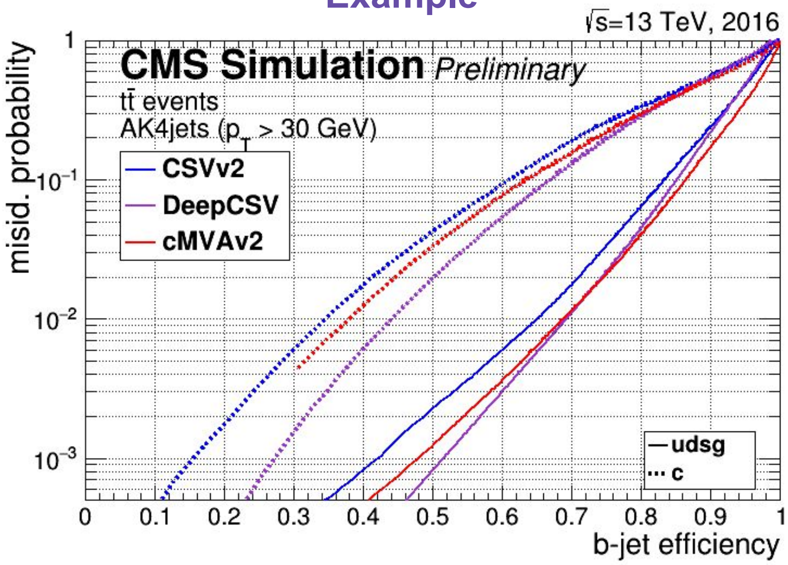
- Inputs
- Algorithm
- Performance: ROC curve



TP	FP
FN	TN



Example



ROC curve for b-tagging :

B efficiency / TP (x - axis)
vs
Mistag / FP (y - axis)

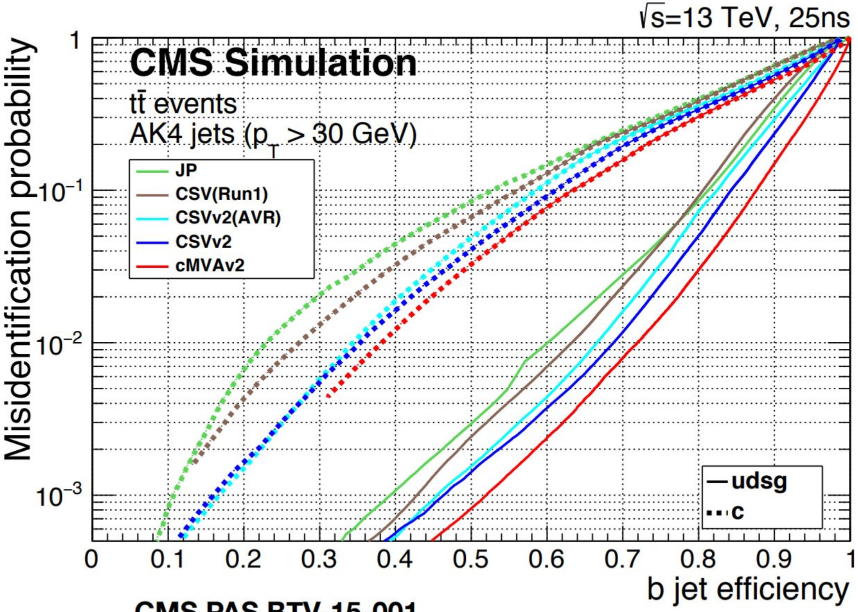
CSV (Combined secondary vertex)

CSV -> BDT or Shallow NN based

based on the combination secondary vertex and track information

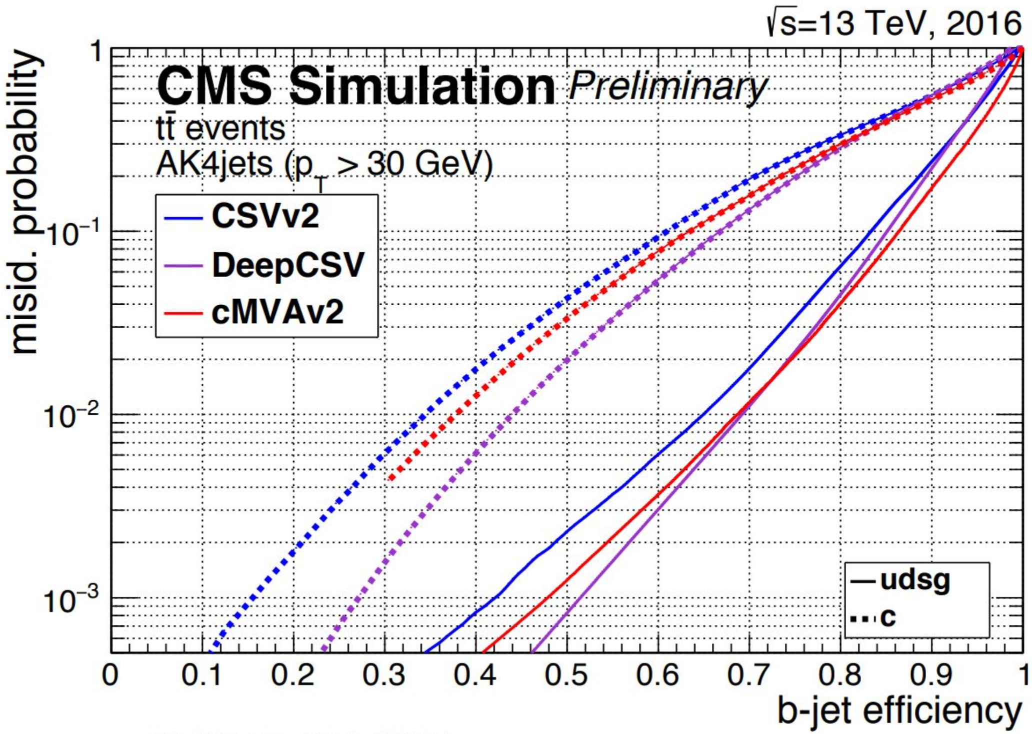
The variables used are chosen based on discriminating power / previous knowledge

- Multiple training steps
- 3 categories: vertex - no vertex - pseudovertex
- ~ 20 variables, “tagging variables”



CMS PAS BTV-15-001

DeepCSV



CMS DP-2017/005

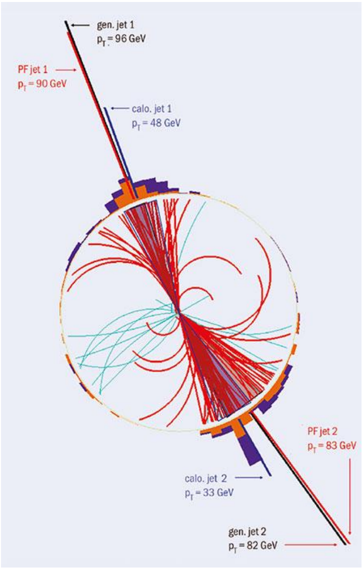
DNN based version + a few more tracks

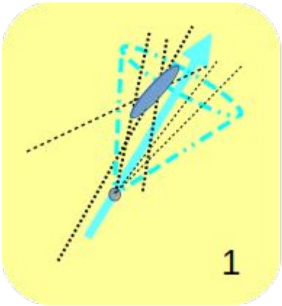
Using the same set of variables as the DeepCSV algorithm - but more charged particle tracks.

DNN based, with four hidden layer (i.e. six layers altogether) of a width of 100 nodes each.

Going deeper - lower level inputs

- Not just discriminating variables
  - Thanks to capability of DNNs one can be less picky with the input choice
  - The algorithm can be more flexible in the optimization of the input choice
- Jet fed to a DNN a set of particles
- Particles collections - each with the same features





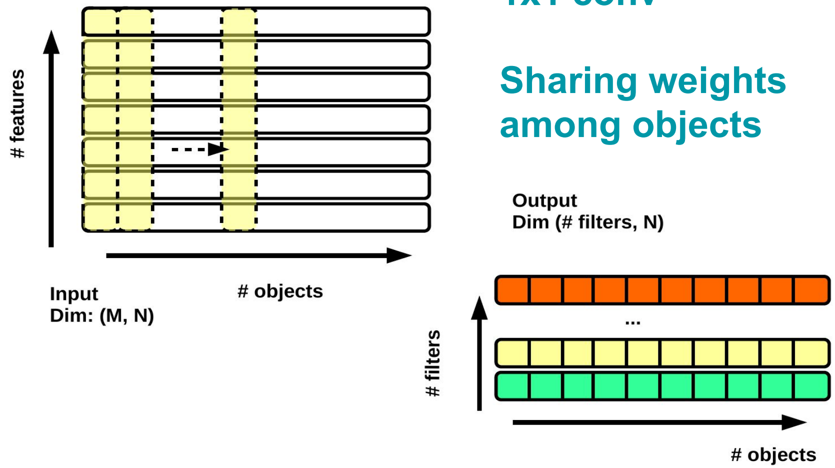
Collections: Reconstructed secondary vertices

Collections:

- **charged particles with b-tagging (not only) properties**
- **Neutral particles (?)**

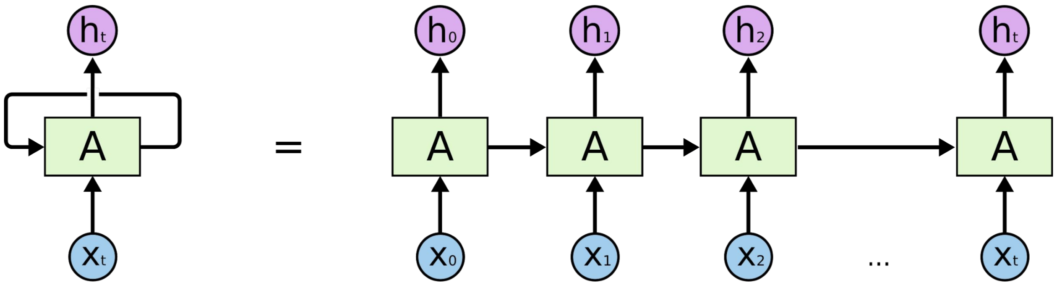
Sequence processing

- Sequence of e.g. tracks
  - Parameter sharing
    - -> conv 1x1
    - -> recurrent networks



1x1 conv

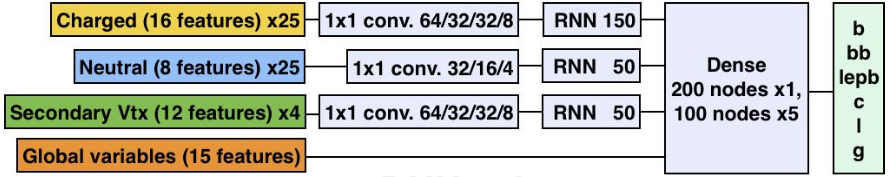
Sharing weights among objects



Recurrent node

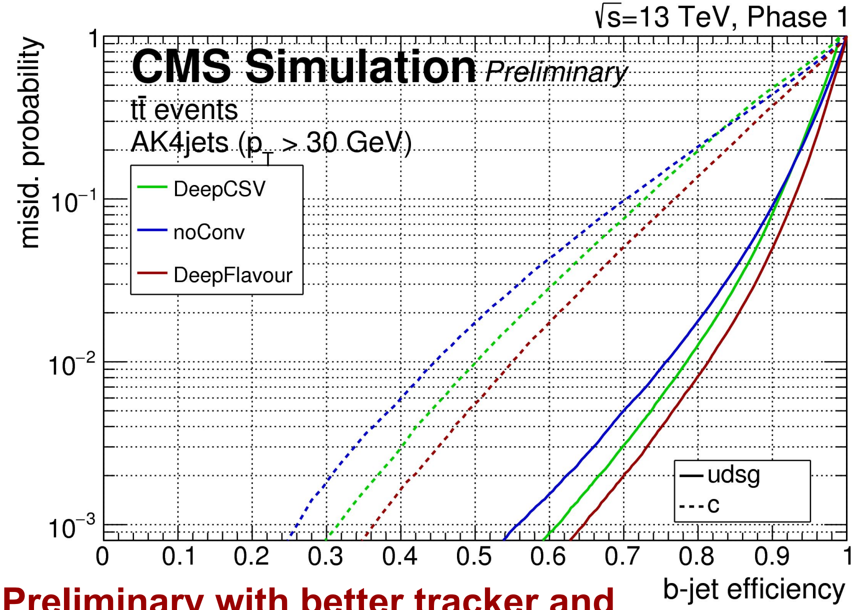
Parameter sharing across sequence

DeepJet

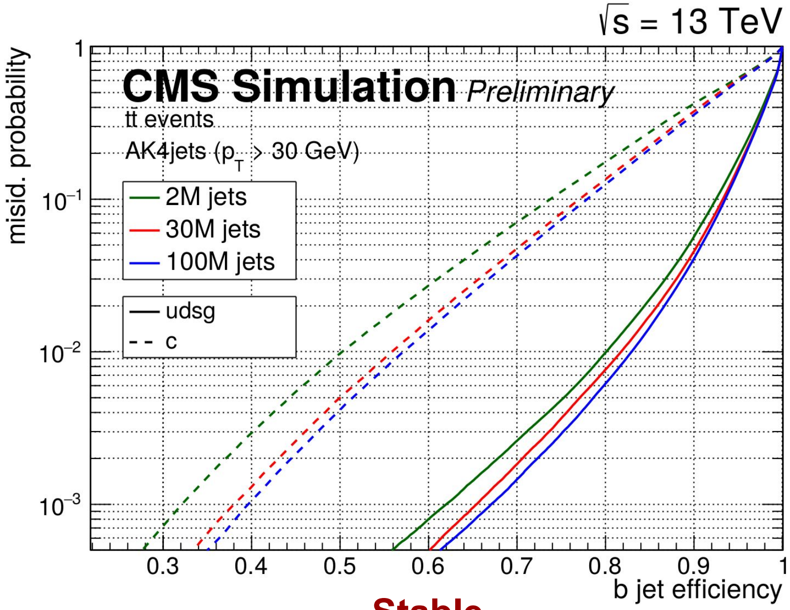


DNN scheme

Conv1D + LSTM to process collections



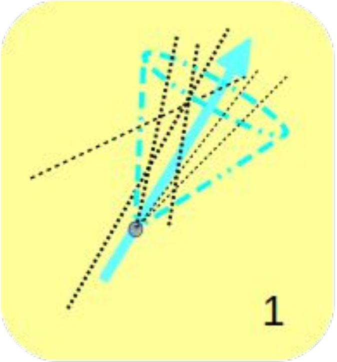
Preliminary with better tracker and compare algorithms



Stable
(sample dependence)

DeepVertex

- Going further: vertexing handled by the DNN
    - Vertices from track clusters around displaced tracks

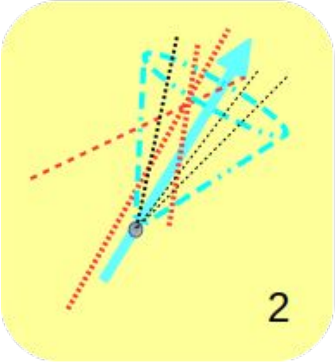


Displaced tracks

- Multiple level of sequencing

1) **Collection of displaced tracks IP**
significance based (10 per jet / or zero-pad)

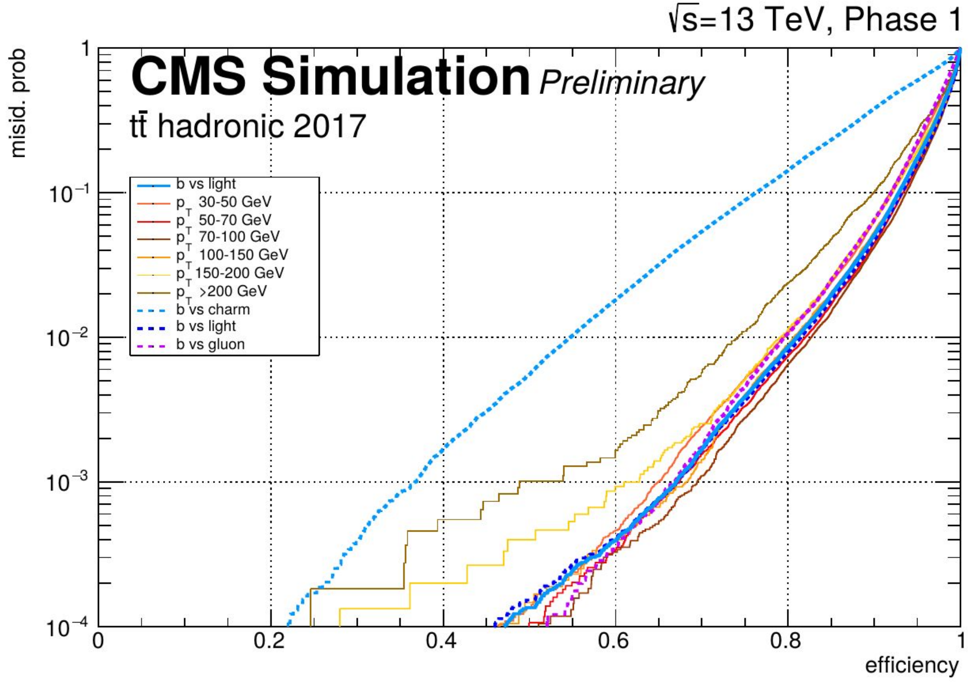
2) **A collection of neighbors for each**
PCA distance based (20X10 per jet - 20 per seed)

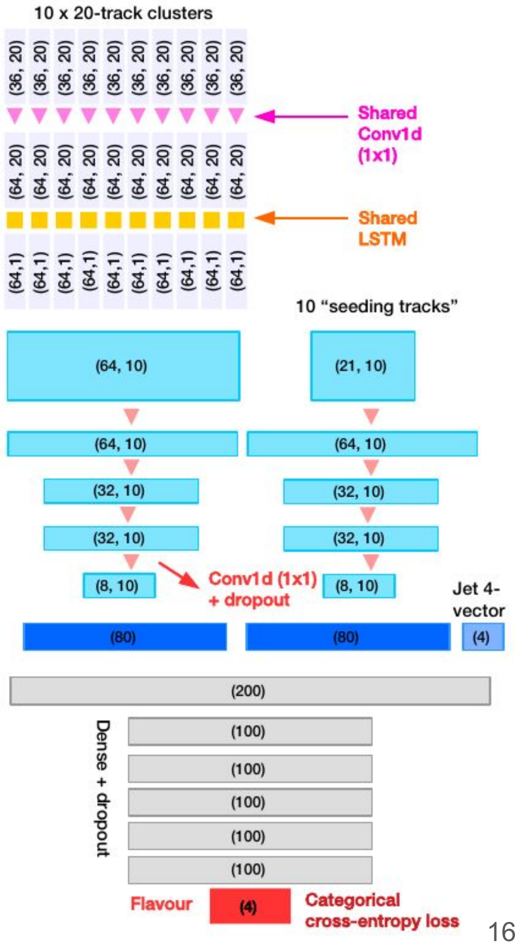


Cluster around displaced track

DeepVertex

DNN architecture and performance





The tutorial

Notebooks here

1. plot_NNinput
2. plot_seedingTrackFeatures
3. keras_DNN
4. CNN1x1_btag
5. lstm_btag

Notebook 1

* **loading the data**
* **check some of the data content and labeling**
* **plot the labeling**
* **plot the distributions per category**
* **example ROC curve**

Notebook 2

not very different from notebook 1

* **loading the data**
* **check another ntuple content**
  The 2nd ntuple contains variables per track per jet
  So it is a sequence inside a jet
* **plot the a distributions per category**

Notebook 3

- Keras user manual (https://keras.io/)

In this notebook, we will

- **Load the data from the usual file**
- **build the feature and the target array**
- **define a DNN with three layers, fixing node number, activation function, etc**
- **train the model, using Early Stopping and dynamic learning rate**
- **check training history**
- **check training performances: AUC and confusion matrix**

Notebook 4 (5)

In this notebook, we will

- **Load the data from the usual file**
- **build the feature and the target array**
- **define a "convolutional" ("recurrent") DNN**

  - **The DNN used 1x1 convolution to share parameters between object at the same level (tracks)**
  - **reference (https://keras.io/layers/convolutional/)**

  (- The DNN uses the LSTM to process the track sequence instead of 1x1 convolution
  - Recurrent layers info (https://keras.io/layers/recurrent/))

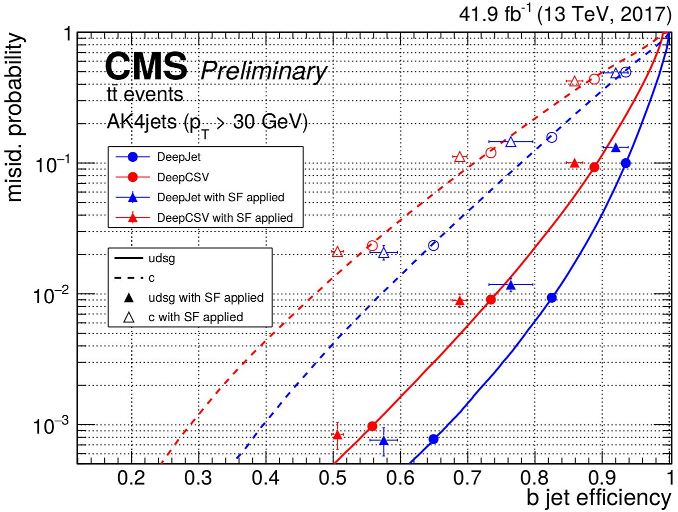
- **train the model, using Early Stopping and dynamic learning rate**
- **check training history**
- **check training performances: AUC and confusion matrix**

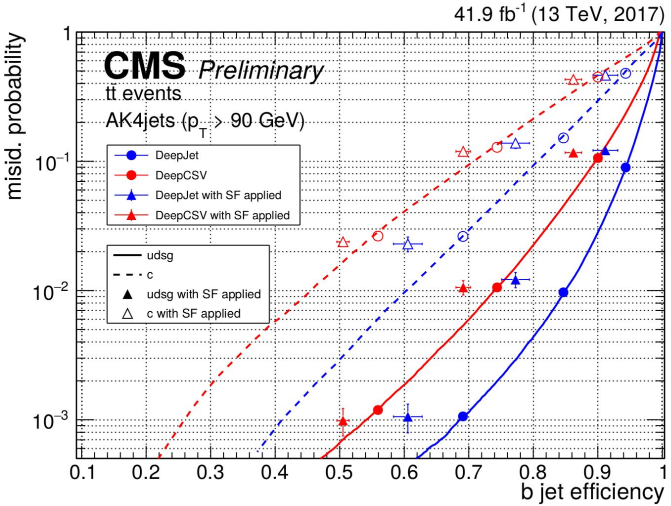
Performance in data - Scale factors

All algorithms are trained with simulation

- Accurate and up to date simulation of physics processes + detector

Performance is very similar in data - a bit worse - corrections are needed for analysis.





More material

ML with Jets in CMS:
https://indico.cern.ch/event/745718/contributions/3146638/attachments/1753044/2841151/ML4Jets2018.pdf

Today's Introduction by Andrea Rizzi