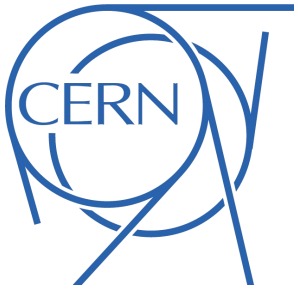


This Report is part of a project that has received funding from the **European Union's Horizon 2020 research and innovation programme under grant agreement Nº675440**

Machine learning for jet physics in CMS

Markus Stoye[1,2]

on behalf of the CMS collaboration

CERN[1], ITN aMVA4newphysics[2]

11th December 2017, machine learning for jet physics, LBLN, Berkeley

IML workshop

April 6-9 IML workshop@CERN

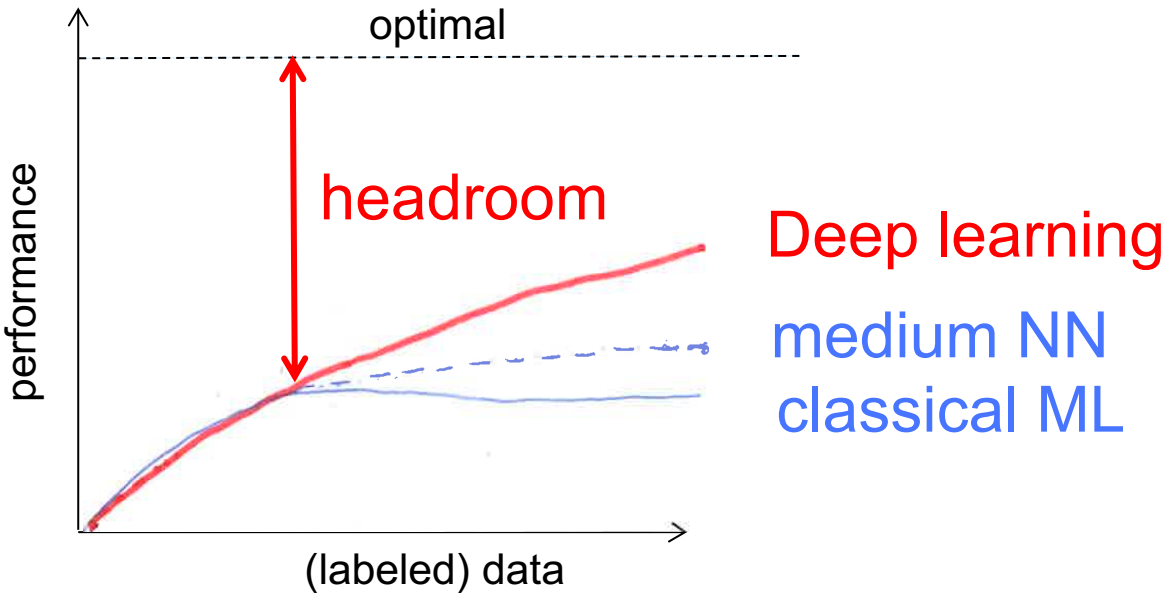
- Discussion on challenges and storing data
- Jet physics a important topic
- Last time ~150 participants

Would be happy to see some of you there

Content

- Deep learning in jets, some thoughts
- Current b-tagging
- DeepJet: physics object base neural networks at CMS for jet physics
- Shallow fat jets

How much do we win in CMS by DL?

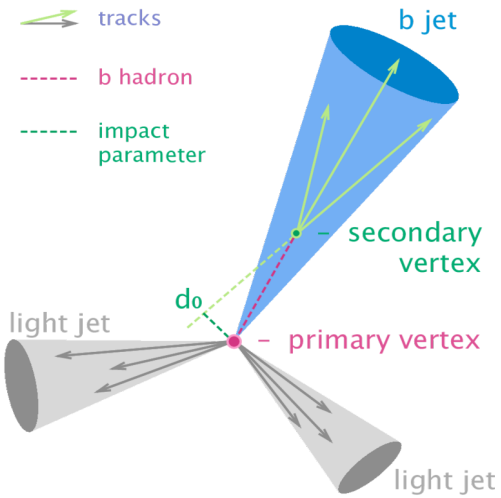


- **High** dimensional inputs with **big** dataset and a **large** **D**eep **N**eural **N**etworks (DNN) brought breakthroughs
- We have plenty of simulated labeled data
- Great opportunity ☺ *IF* there is headroom left by old methods ☹

Deep neural network for flavour tagging

Jet tagging

Task to find the particle ID of a jet, e.g. b-quark



Key features:

- Long lifetime of heavy flavor quarks
- Displaced tracks, …
- Usage of ML standard for this problem

Jet tagging: Multiclass classification

Jet flavour tagging is intrinsically a multi-class classification problem

4 exclusive ***flavour*** categories:

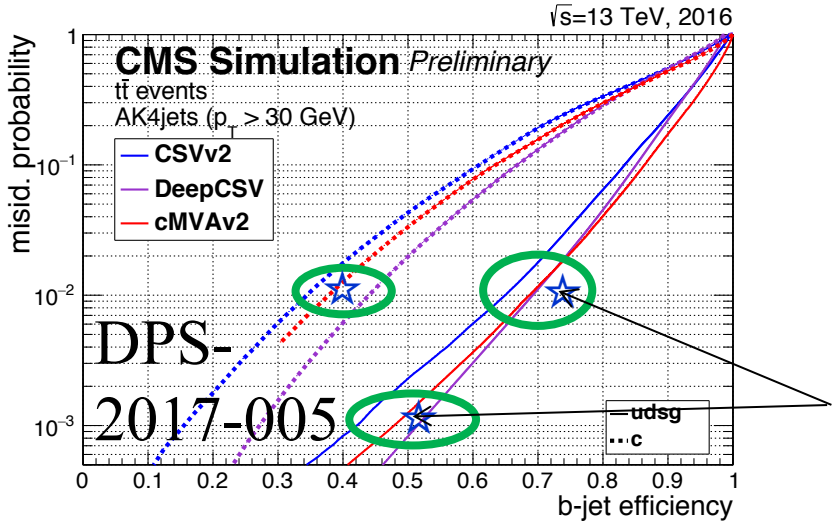
- Exactly **one b hadron** in the jet
- Exactly **one c hadron**, with no b-hadron in the jet
- **Two** or more **b hadrons** in jet
- Light quark/gluon jets (udsg)

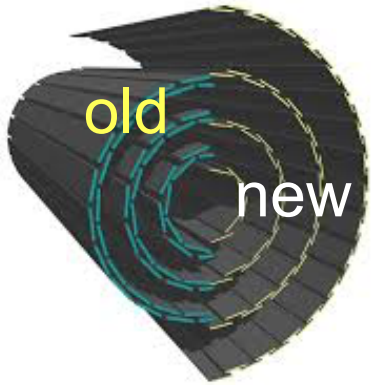
Generic jet tagging has even more classes: light quark, gluons, hadronic τ, pile up

$\rightarrow$ Using many classes is important for a robust taggers. In real data the tagger will see all possible classes

New deep learned flavour tagger

- Diverse samples
  - ➢ QCD **and tt**
- Optimal training: Large jet sample to avoid over-fitting
  - ➢ 50M jets!
- Use complete **standard CSV b-tag "Tag info"** (from ~30→60)
- Dense Deep Neural Network (Dense)

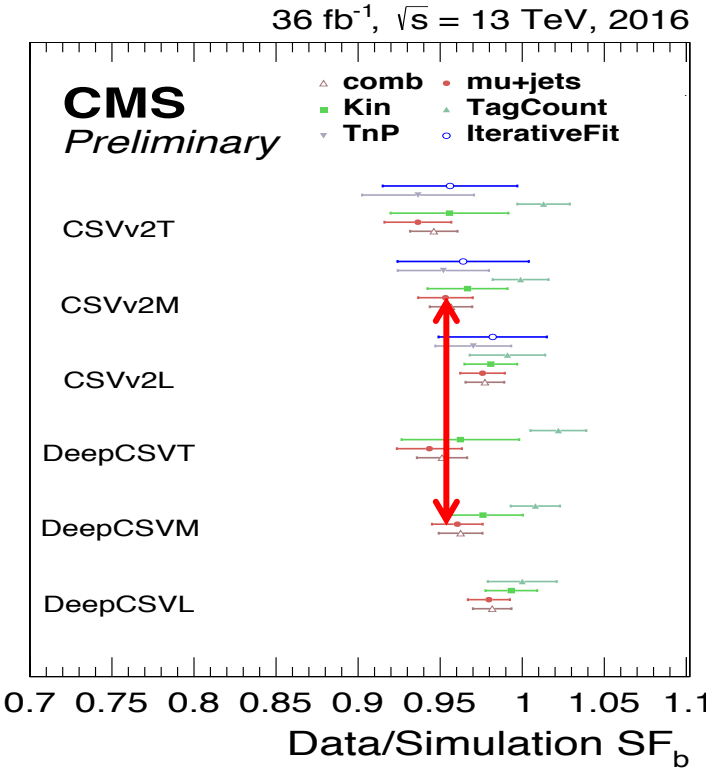




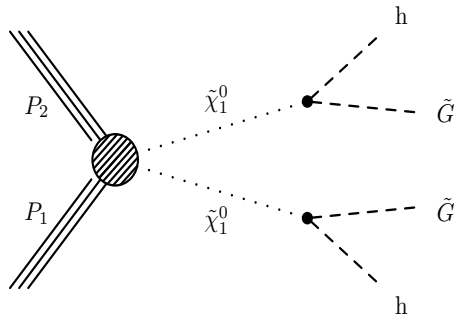
☆ Old tagger with new pixel detector in simulation

Similar impact as the new inner pixel!

Application of new tagger in data



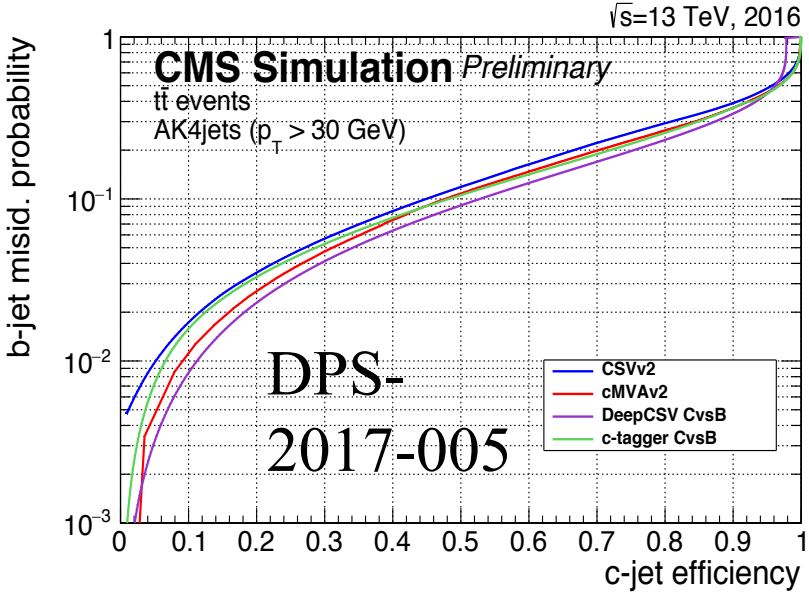
hh->bbbb and MET

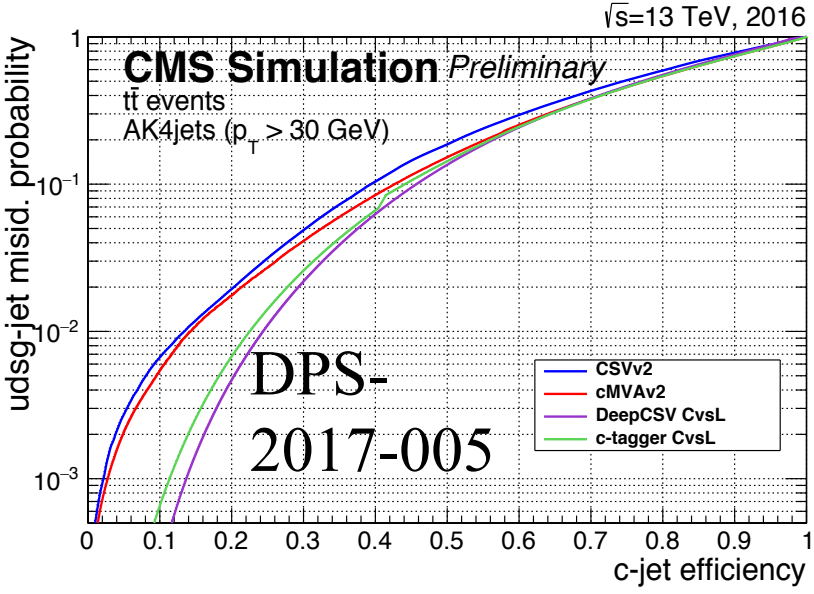


- Up to 50% more signal with 15% more bkg.
- Gained 150 GeV in m($\tilde{\chi}_1^0$)

This new flavour tagger officially ***recommended*** since 2017 in CMS!

ROC for c vs b an light



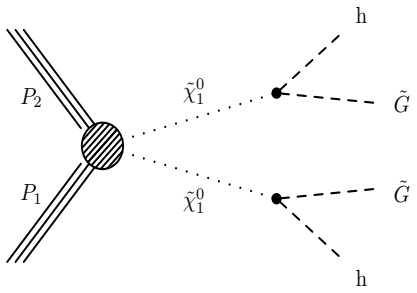


DeepCSV best c tag performance

Application in physics analysis

SUS-16-044:

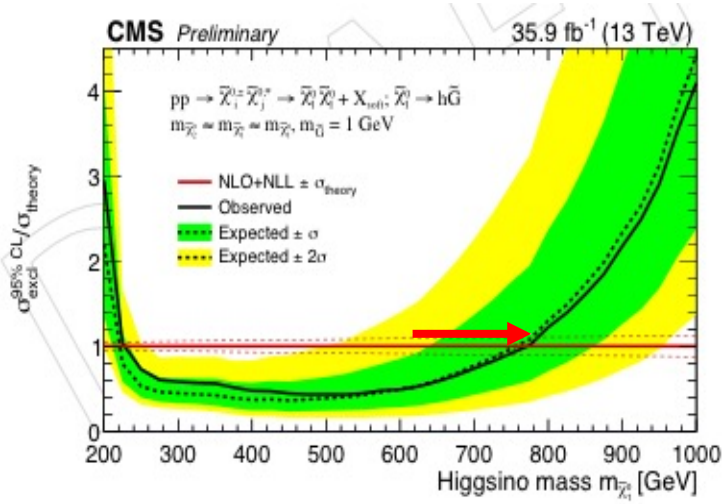
Search for events with two h->bb and MET



2b ≡ $N_{b,T} = 2$, $N_{b,M} = 2$

3b ≡ $N_{b,T} \geq 2$, $N_{b,M} = 3$, $N_{b,L} = 3$

4b ≡ $N_{b,T} \geq 2$, $N_{b,M} \geq 3$, $N_{b,L} \geq 4$



CSVv2 $\mathcal{L} = 35.9$ fb^{-1}	All SM bkg.	TChiHH (225,1)	TChiHH (700,1)
$\geq$ 2b	–	3761.5	33.7
$\geq$ 3b	–	1999.1	19.0
4b	–	860.0	9.3
Baseline, $\geq$ 2b	2600.1±101.0	75.6	7.7
Baseline, $\geq$ 3b	276.9±5.5	49.6	5.4
Baseline, 4b	72.2±4.1	30.9	3.6
Baseline, $p_T^{miss} > 300$, $\geq$ 2b	104.2±2.4	2.8	6.0
Baseline, $p_T^{miss} > 300$, $\geq$ 3b	12.9±0.8	2.4	4.2
Baseline, $p_T^{miss} > 300$, 4b	4.0±0.4	1.7	2.8

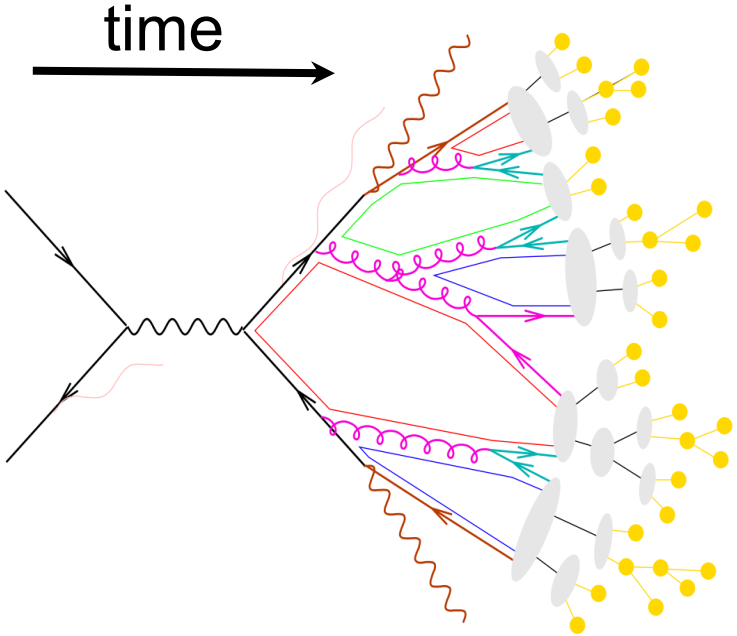
DeepCSV $\mathcal{L} = 35.9$ fb^{-1}	All SM bkg.	TChiHH (225,1)	TChiHH (700,1)
$\geq$ 2b	–	4625.6	39.7
$\geq$ 3b	–	2548.7	24.1
4b	–	1149.1	12.7
Baseline, $\geq$ 2b	3650.5±90.2	95.1	9.9
Baseline, $\geq$ 3b	385.2±9.0	68.6	7.4
Baseline, 4b	94.3±5.3	43.4	5.1
Baseline, $p_T^{miss} > 300$, $\geq$ 2b	144.8±2.8	4.0	7.7
Baseline, $p_T^{miss} > 300$, $\geq$ 3b	16.3±0.8	3.2	5.7
Baseline, $p_T^{miss} > 300$, 4b	4.6±0.4	2.5	4.0

Significant Improvement: e.g. up to ~50% more signal for 15% more bkg

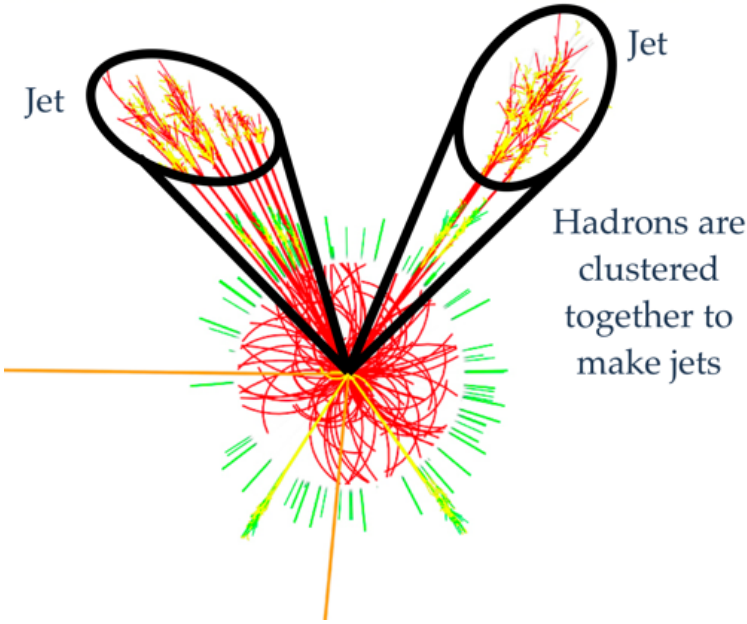
→ Significantly improved lower mass limit (150 GeV in Higgsino mass)

DeepJet
Physics object based deep learning

Collecting particles from one hard scatter particle

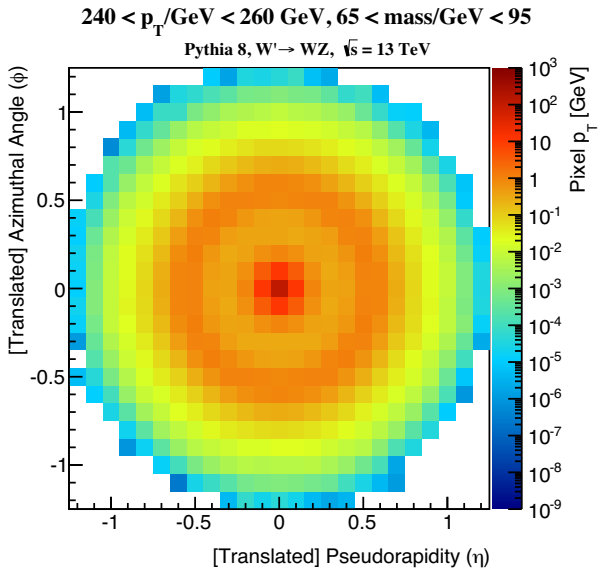


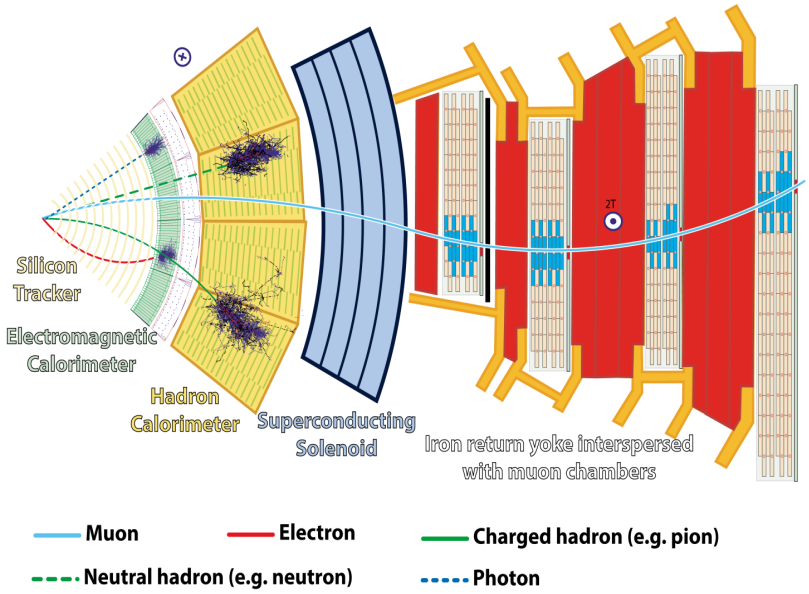
- hard scattering
- (QED) initial/final state radiation
- partonic decays, e.g. $t \to bW$
- parton shower evolution
- nonperturbative gluon splitting
- colour singlets
- colourless clusters
- cluster fission
- cluster → hadrons
- hadronic decays



- We are interested in the properties (momentum, particle Id) of the “black”; but in the detector we see the loose ends on the right.
- We use a clustering algorithm (anti-k_T) to collect particle candidates and than secondary vertices that might belong to one particle from the hard scatter.
- We remove particles coming from PU collisions (z displaced vertex)

CMS is a complex detector





- Convolutional networks propose for jet images* and shown to work for some problems
- In general the CMS detector is more complex, e.g. not translational invariant

CMS not “image” like, 2D CNN less easy to use

What is a charged particle in the detector?

Charged particles flow candidates

- Particle flow candidates combine the information of all subdetector
- p_T, η, ϕ, and and particle ID
- Estimated of probability to be from the primary vertex
- Provides links to rawer objects like tracks
- Via particle tracks access to "BTV" features and others
- ***Maybe a DeepParticle candidate would be interesting***

feature	offset	lower bound	upper bound	comment
trackEtaRel	-	-5	15	BTV
trackPtRel	-	-	4	BTV
trackPPar	-	-10^5	10^5	BTV
trackDeltaR	-	-5	5	BTV
trackPParRatio	-10	100	-	BTV
trackSip2dVal	-	-	70	BTV
trackSip2dSig	-	-	$4 \cdot 10^4$	BTV
trackSip3dVal	-	-	10^5	BTV
trackSip3dSig	-	-	$4 \cdot 10^4$	BTV
trackJetDistVal	-	-20	1	BTV
trackJetDistSig	-	-1	10^5	BTV
$p_T(cPF)/p_T(j)$	-1	-1	0	
$\Delta R_m(cPF, SV)$	-5	-5	0	
fromPV	-	-	-	
VTXass	-	-	-	
$w_p(cPF)$	-	-	-	
χ^2	-	-	-	
Npixel hits	-	-	-	

More features of particle jets

Neutral particles candidates

feature	offset	lower bound	upper bound
$p_T(nPF)/p_T(j)$	-1	-1	0
$\Delta R_m(nPF, SV)$	-5	-5	0
isGamma	-	-	-
hadFrac	-	-	-
$\Delta R(nPF)$	-0.6	-0.6	0
$w_p(cPF)$	-	-	-

Secondary vertices

feature	offset	lower bound	upper bound
$p_T(SV)$			
$\Delta R(SV)$	-0.5	-2	0
m_{SV}	-	-	-
$N_{\text{tracks}}(SV)$	-	-	-
$\chi^2(SV)$			
$\chi^2_n(SV)$	0	-1000	1000
$d_{xy}(SV)$	-	-	-
$S_{xy}(SV)$	-	-	800
$d_{3D}(SV)$	-	-	-
$S_{3D}(SV)$	-2	-2	0
$\cos\theta(SV)$	-	-	-
$E_{rel}(SV)$	-	-	-

global features

feature	comment
$p_T(j)$	
$\eta(j)$	
N_{cPF}	
N_{nPF}	
N_{SV}	
N_{PV}	
trackSumJetEtRatio	BTV
trackSumJetDeltaR	BTV
vertexCategory	BTV
trackSip2dValAboveCharm	BTV
trackSip2dSigAboveCharm	BTV
trackSip3dValAboveCharm	BTV
trackSip3dSigAboveCharm	BTV
jetNSelectedTracks	BTV
jetNTracksEtaRel	BTV

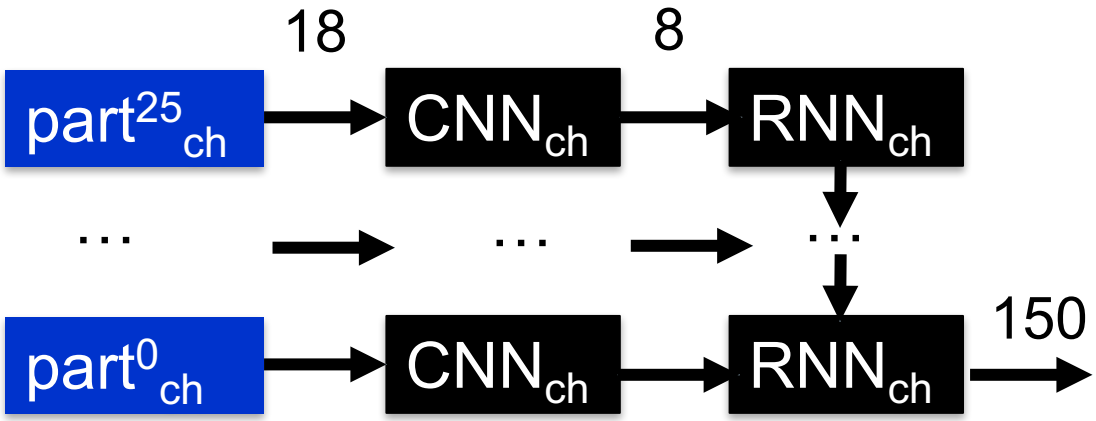
Strategy:

- Add quite extended information of jets
- Build a DNN that can deal with many and potentially low information features

Physics object based NN architecture

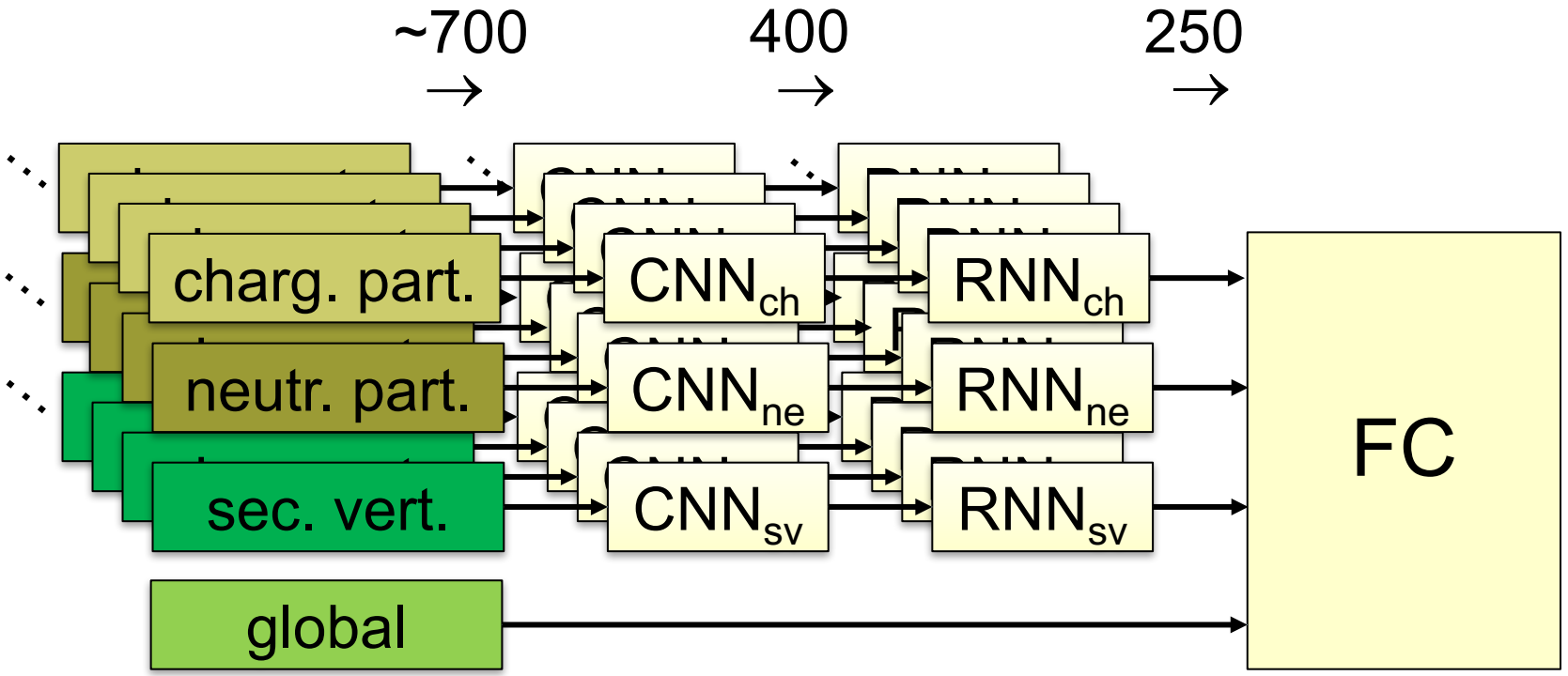
Example: charged particle candidates

- Four 1x1 1D CNN layers reduces 18 to 8 features (feature engineering)



- A recurrent NN (LSTM) represents the sequence of charged particles that is sorted by impact parameter significance
- A constant length vector is than given to the next layers

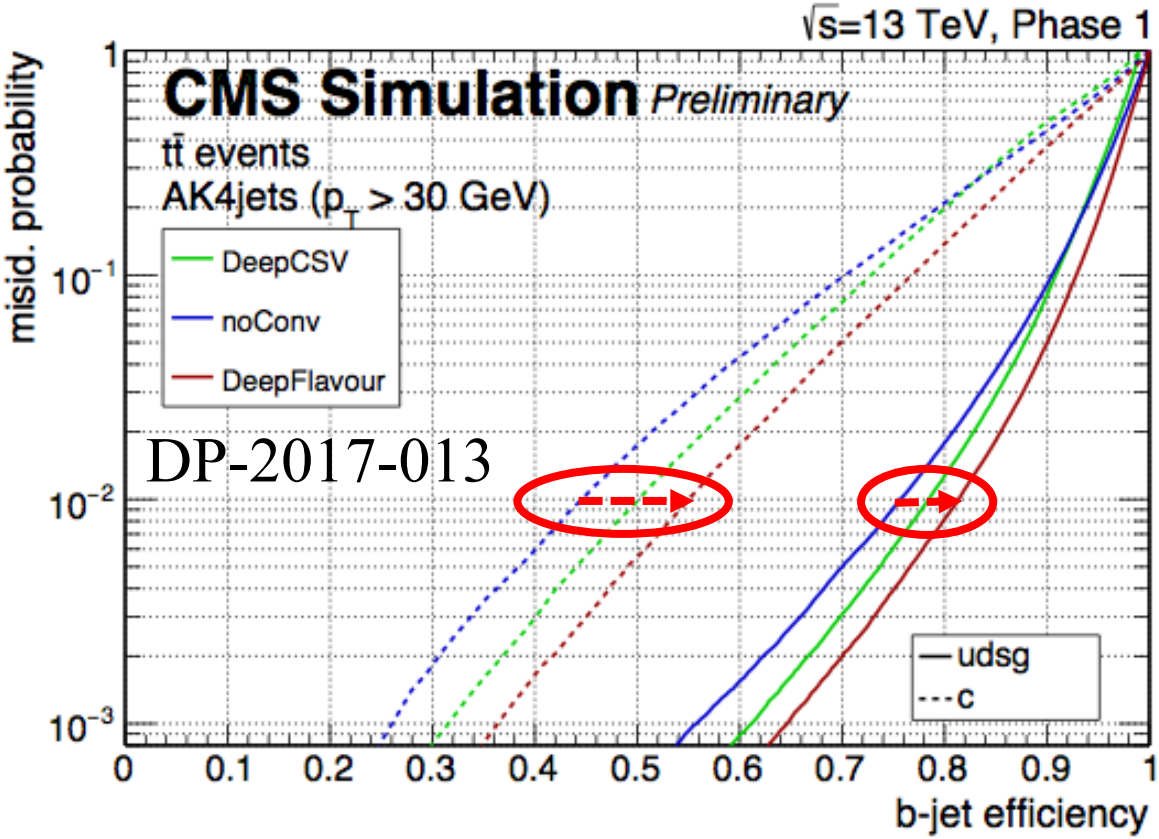
Particle and vertex based DNN: DeepJet



~ 700 inputs and 250.000 model parameters

- Particle and vertex based DNN has factor 10 less free parameters than a generic Dense DNN would have
- 100M jets used for training, overtraining is not an issue

Impact of DNN architecture



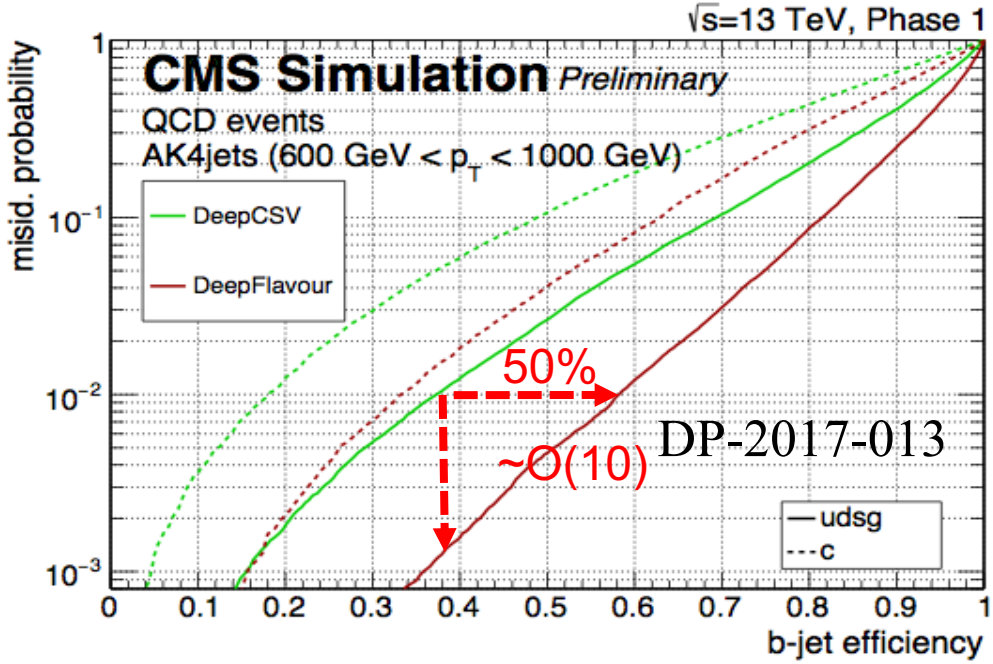
Blue: generic DNN (650 inputs)

Green: CMS tagger (~65 human made inputs)

Red: Physics inspired DNN (650 inputs)

Particle&Vertex based DNN performs best

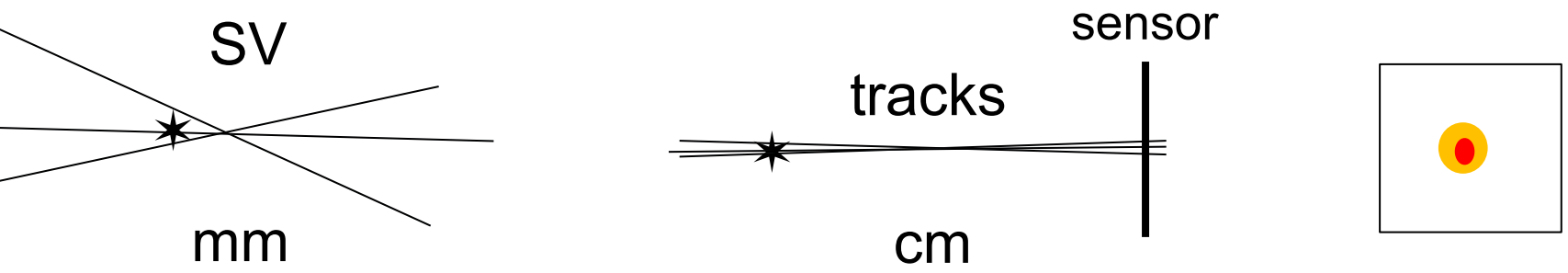
DNN reveals true CMS potential



Very significant gain at high p_T

- With DeeJet network can reproduce DeepCSV if for same inputs
- Increase input step by step:
  - ***Not applying track selection as in BTV*** (lost valuable information)
  - ***More features help***, e.g. number of Pixel hits
- Past human features track selection procedure a bottleneck of performance
- DNN allows more automated evaluation of which information is needed

Simplified p_T evolution of b-tagging

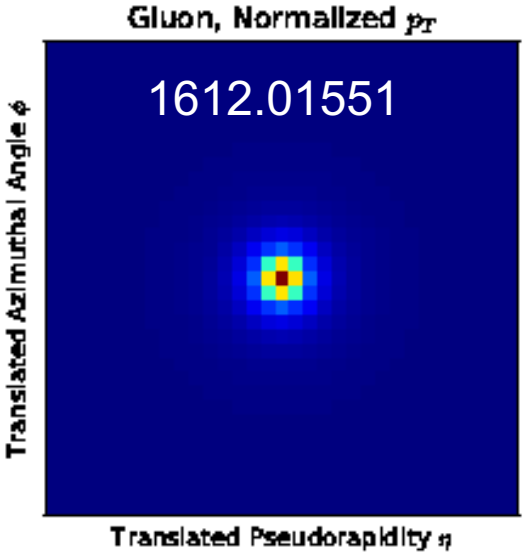


- Vertexing and tracking increasingly difficult at high p_T
- Tracks and e.g. number of pixel hits or even pixel images become more interesting
- Track selection at high p_T was suboptimal in CMS

Quark gluon separation

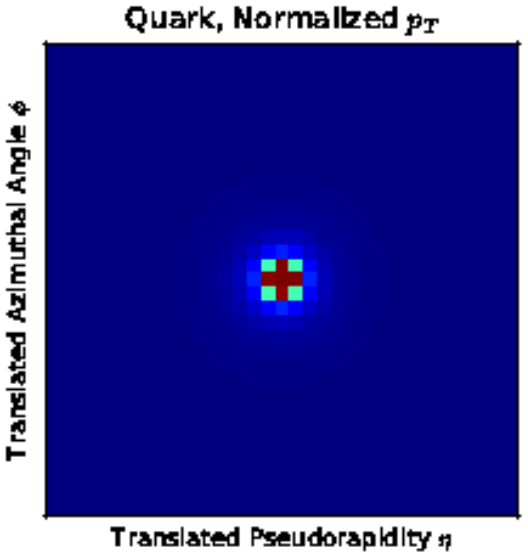
Gluon radiate more:

- Typically wider spread and softer particles
- Thinner and harder particles



Both, quark and gluon have are prompt, i.e. displaced particles and vertices are not relevant

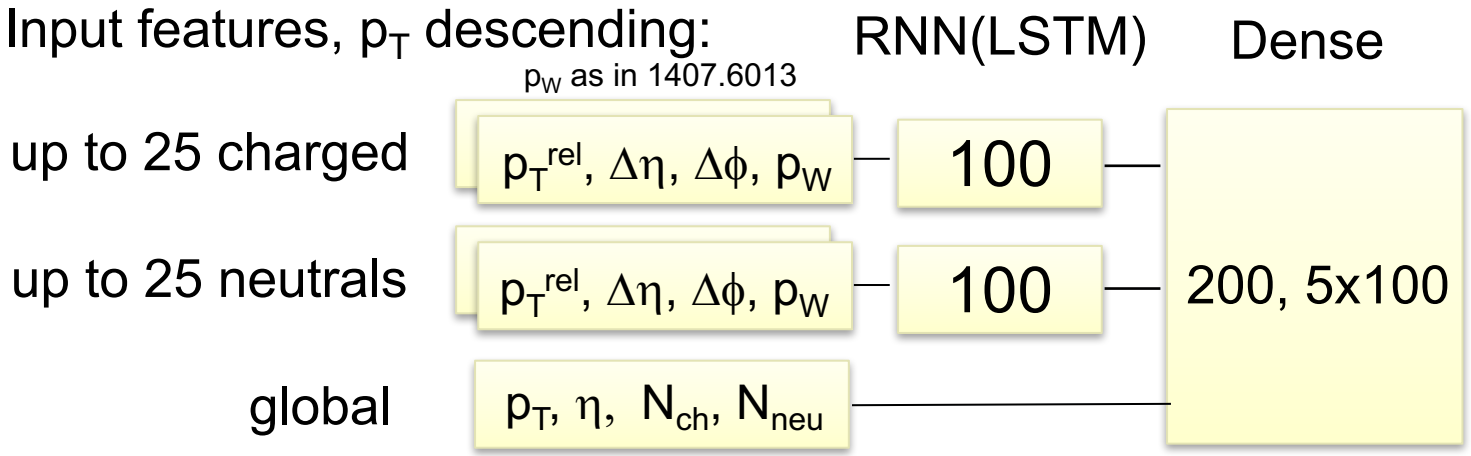
- Image approach proposed in 1612.01551



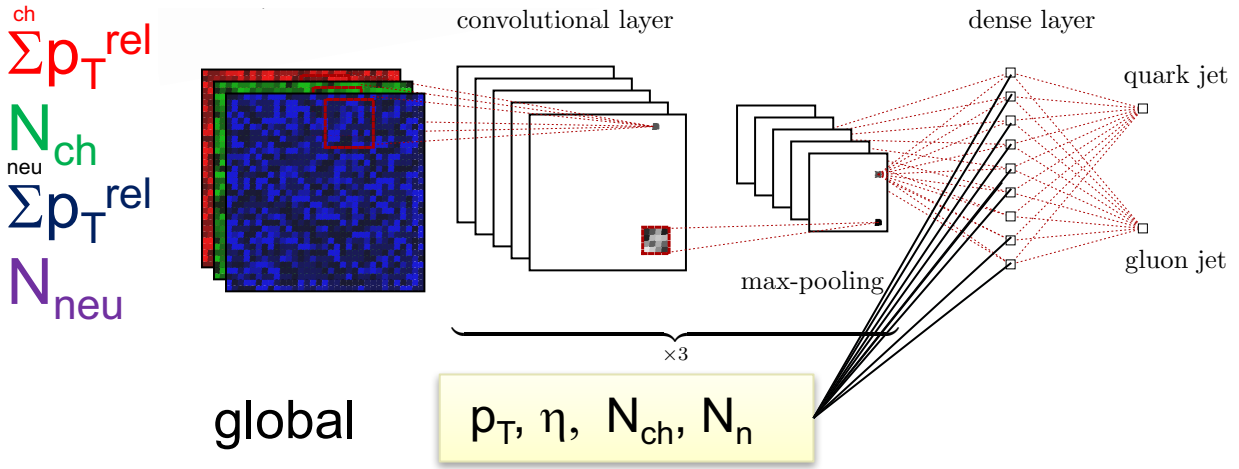
Quark gluon separation

Investigate a few custom DNN q/g tagging:

Recurrent for q/g:

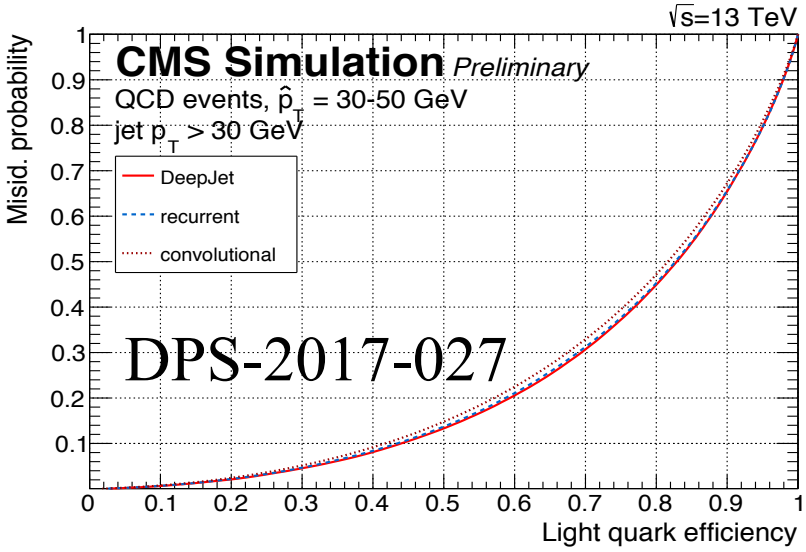


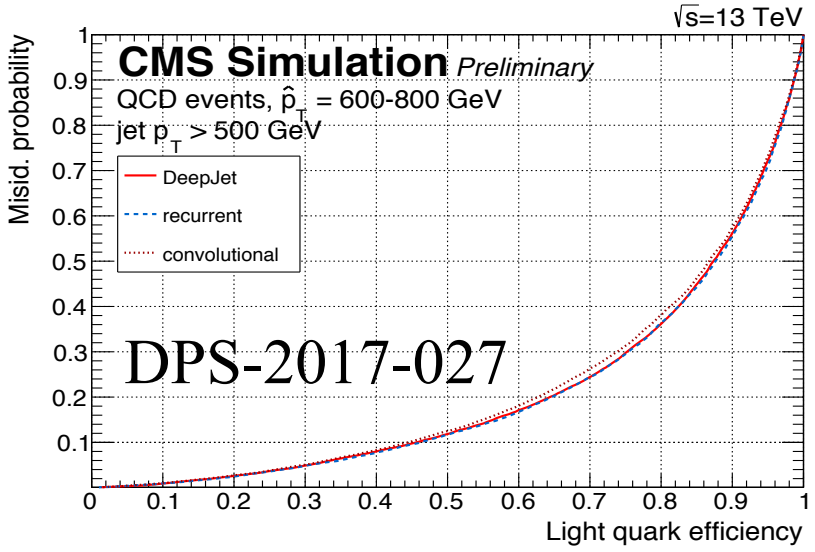
2D **convolutional**, four channels (CNN as in 1612.01551):



Comparisons of DNNs

- We filter on ***generator*** level only light quarks and gluons that did **NOT** split to heavy flavour.





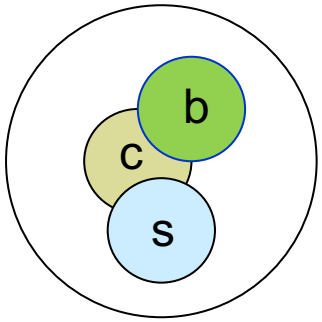
→ Generic DeepJet and custom q/g DNN gave very similar results!

→ **Data is multi-class, without heavy flavour removed DeepJet was clearly best**

(sub) Structure + flavour

Physics object based DNN showed great performance for quark/gluon and flavour tagging.

Should work also for heavy object tagging?



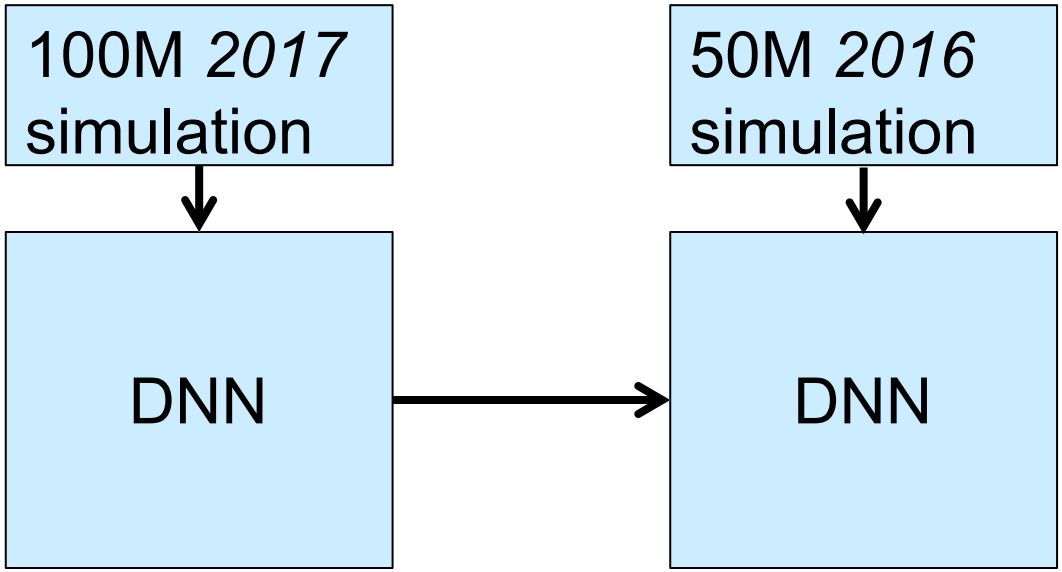
- Particles of subjets can overlap
- No clear that we can factorize into “flavour” and substracture without loss of performance
- Good place to try DeepJet like approaches
- See Huilins talk for NEW CMS results

Could DeepJet work also for regression.

- As densities (as in q vs g) and flavour matter for JEC, DeepJet seems a good approach (work in progress)

Retraining

- New condition (PU or new geometry) require retraining of the network
- Use “similar” training sample with huge statistics to “pre-train”
- Increases effectively your data-sets

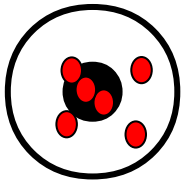


Used 2017 DNN as start or fixed some inner layers for 2016 DeepJet

Methods to deal with simulation

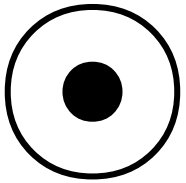
Definition of the target (loss)

Current target



Optimal performance in simulation

Desired target



Optimal and known performance in data

We teach ML to hit the wrong target

Core of the problem: data ≠ MC

a) Data has no labels:
   - fully supervised learning impossible

b) Simulation is not perfect:
   - Data is only similar to simulation, e.g. has worse resolution

Are we on our own or is that a usual problem of data science?

Domain adaptation

Source domain (MC)

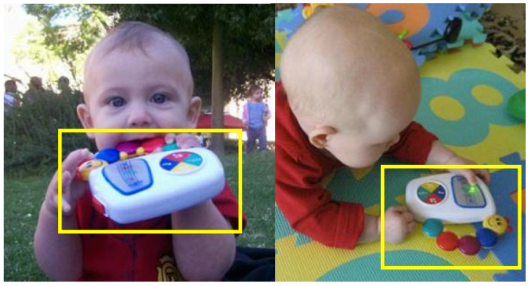
Target domain (real data)

Good samples with **labels** for training a classifier

digital SLR camera

low-cost camera, flash

amazon.com



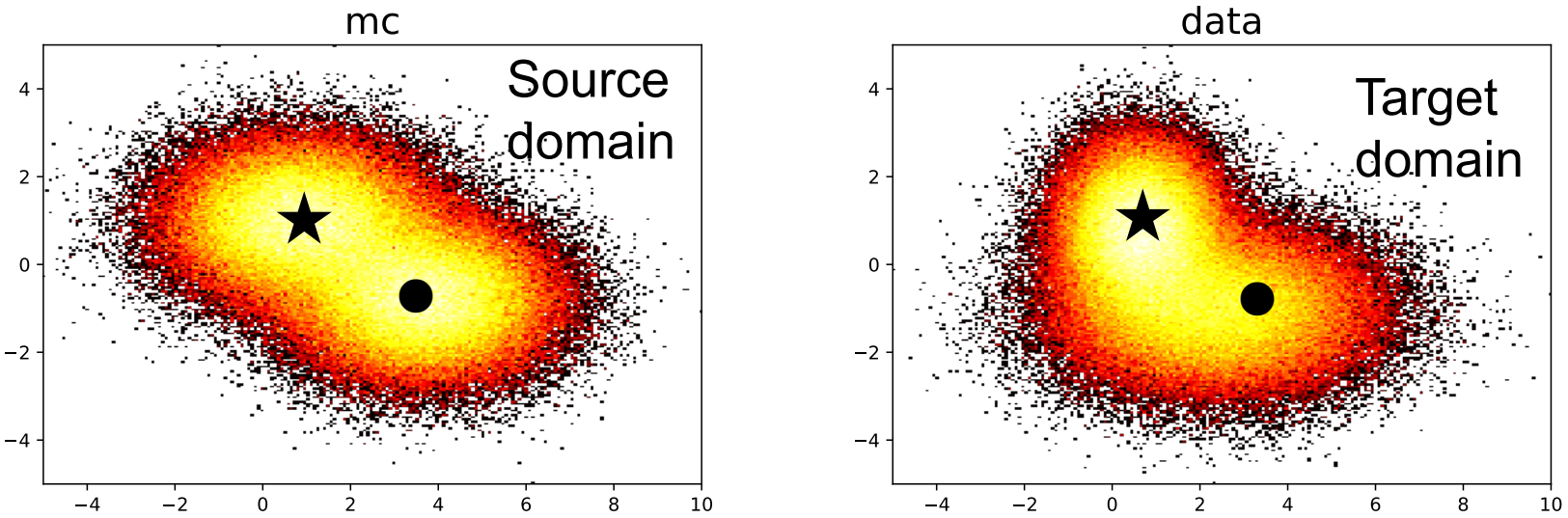
consumer images

User samples to apply the training, **no labels** available

Much literature; mainly aimed to have good performance of classifier in target domain.

Building intuition: Toy

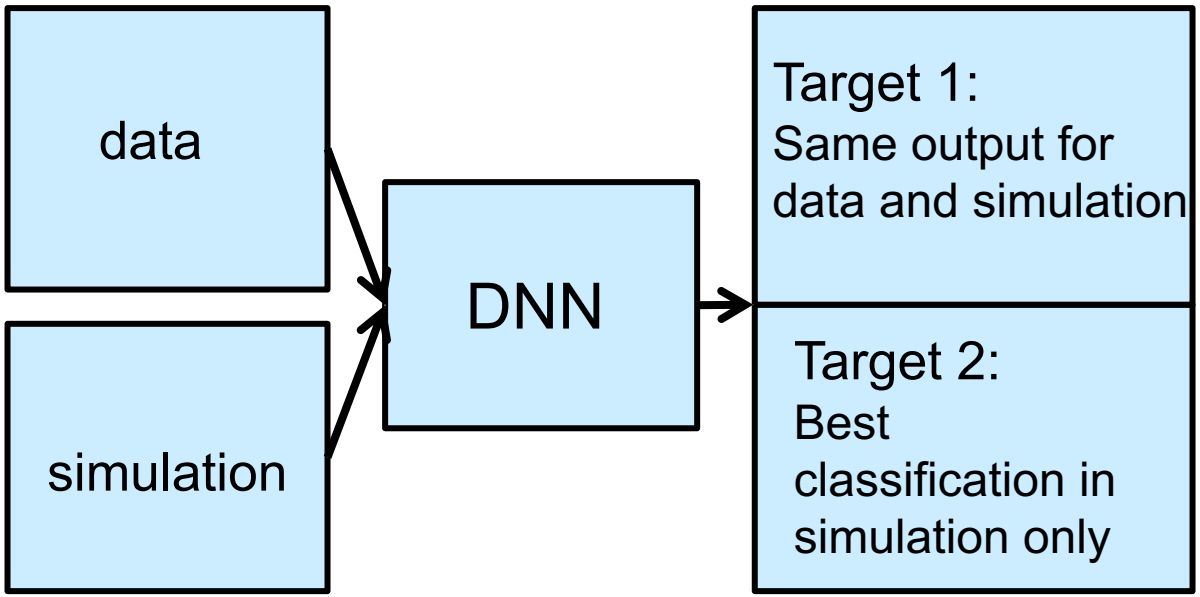
Separating stars and dot samples

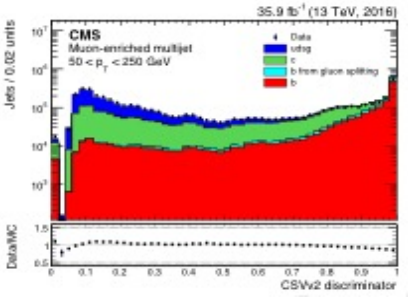


“data” additionally smeared dots

How to get good performance in “data”?

Adding data/MC agreement to target





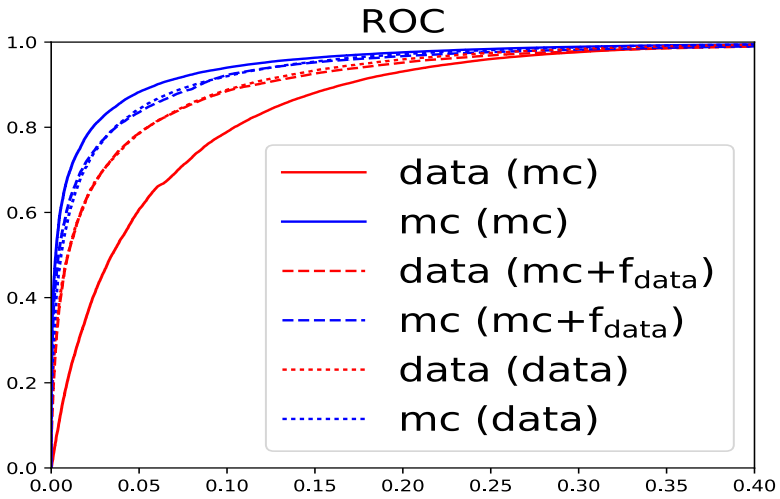
Add RMS of differences of moments in simulation and data

+

MC only x-enthropy

- Enforce that classifier output is independent of source, i.e data or MC
- One solution for DNN: improve data classifier at expense of simulation

Toy Results



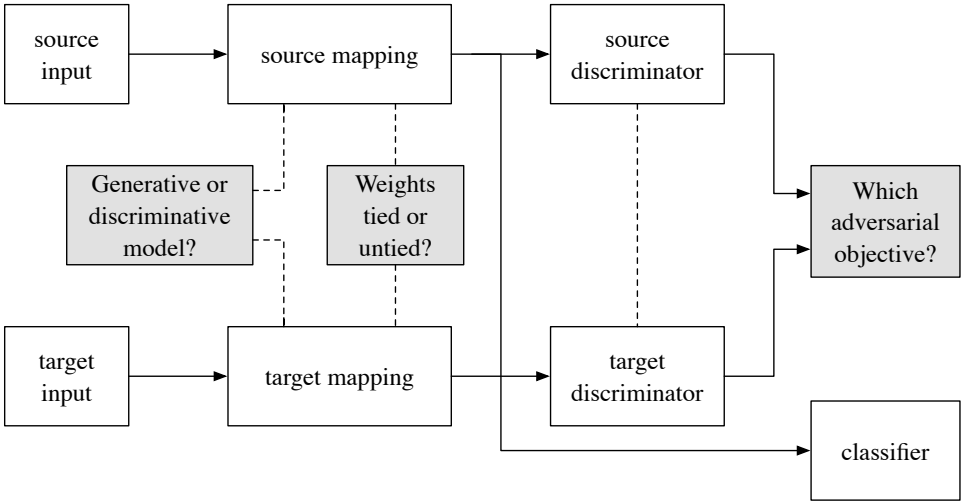
- Trained with mc (mc)
- Retrained with mc adding adversarial loss
- Trained using data ***labels*** (data)

➔ Better data/MC agreement
➔ Smaller systematics uncertainty
➔ Better data performance

Simply toy shows that one can use data to go in the right direction

Domain adaptation methods

arXiv:1702.05464v1



- Many recent papers, active field!
- Special to us, we prefer a classifier that perform similar in source and target domain and want to know the difference in performance and uncertainty

DELPHES
fast simulation

Your
experiment

Needs
you!

Join or complain in your collaboration if wrong incentives inside collaborations stop you from working there on ML.

TensorFlow in CMS software

- (Most) of the training is done on a package on top of KERAS/TensorFlow (github.com/mstoye/DeepJet)
- The inference at production time done using directly the TensorFlow model and C++ TensorFlow API inside CMSSW
- Also for older CMSSW version a recipe (using TenorFlow python is available)
- Significant work to implement TensorFlow into CMSSW, but allow much profit from dynamic TensorFlow developments by huge community.

Conclusions

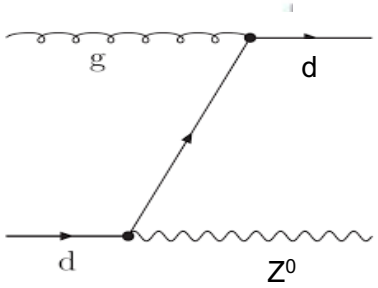
- Deep learning improved CMS reconstruction significantly
- Designed custom DNNs for our problems
- Training on simulation is a common problem with other fields and can partially addresses by ML procedures in the future

Not using simulation at all

Semi-supervised learning works on data only

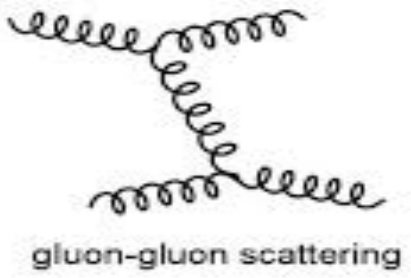
Different sample "bags" with different class compositions, e.g.

Z^0+jets:



many quark jets

Dijet:



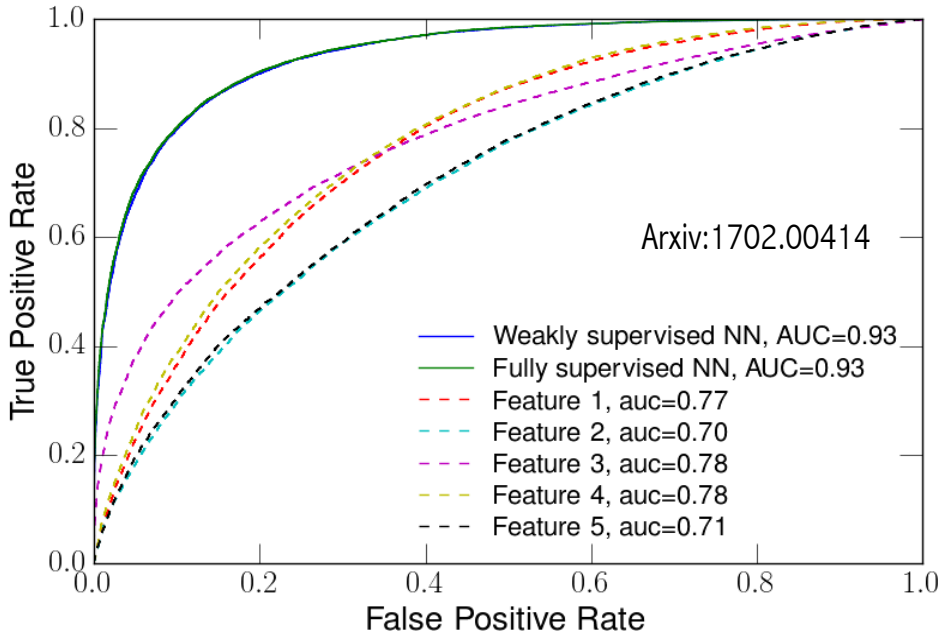
many gluon jets

Assumes that quark gluon is the ONLY difference, e.g. color reconnections are different and many classes present

Quark gluon data only example

Test in simulation with known labels and a simple neural network:

→ Weakly and fully supervised lead to same performance



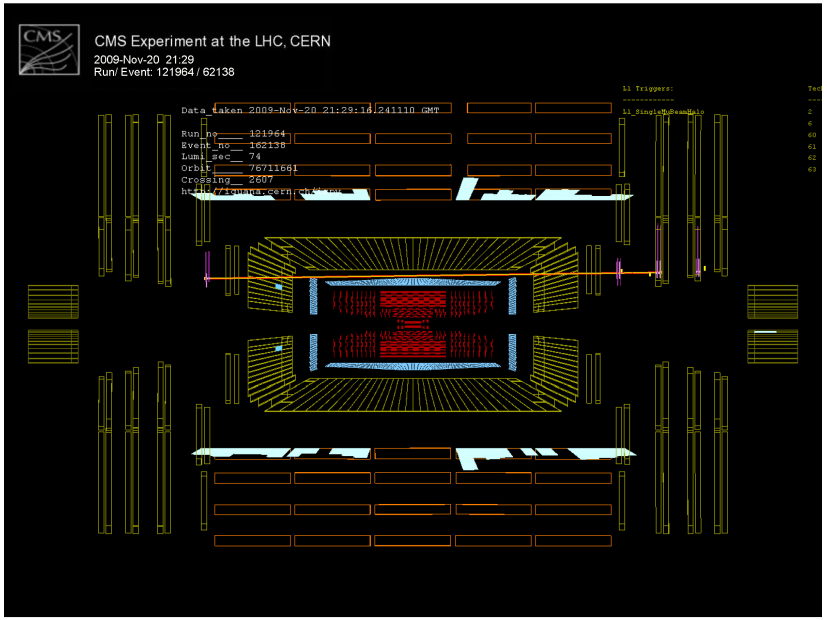
Very interesting approach with a few caveats:

- Limited statistics in data in tails → tricky for deep learning
- Not straight forward to know what "classes" really are finally learned (i.e. anything that separates the sample)

Core of the problem II: data ≠ MC

Simulation misses classes:

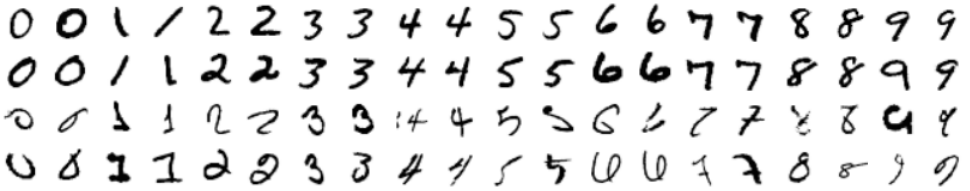
- Data has a few “noise” samples, e.g. beam halo events



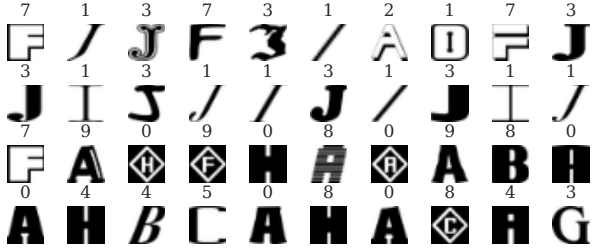
Classifier should not assign this event to a class!

Unknown class problems

Train only on numbers to classify numbers



Problem: In real application text "contaminated" by letters

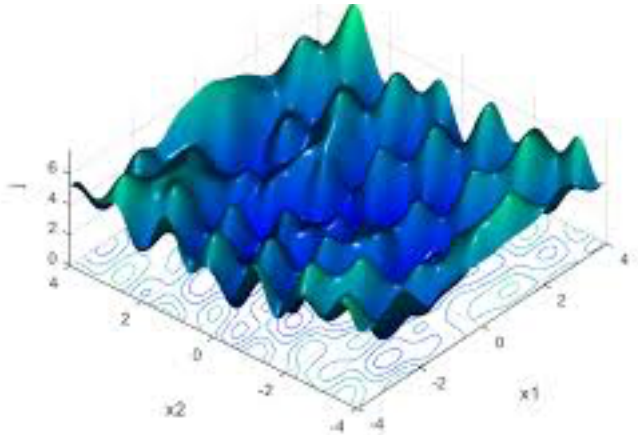


- A neural network might estimate a high probabilities for known class (e.g. $p_1=1$) to an unknown classes
- The high estimated probability of 100% should not be confused with a low uncertainty on the probability

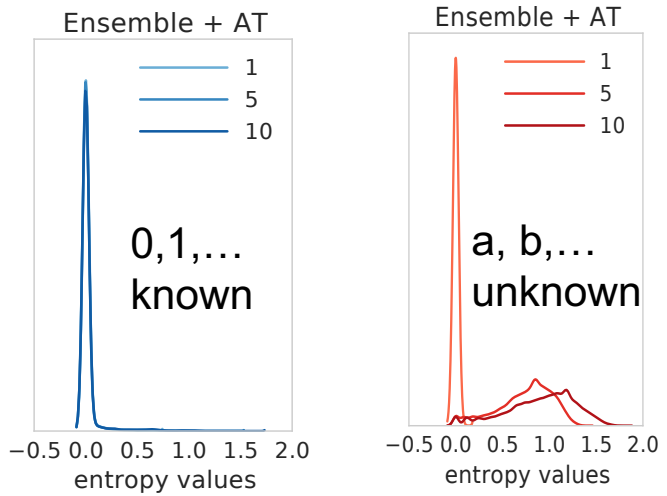
→ Need some sort of "uncertainty"

Intuition: ensemble methods

DNN typically have
many minima (non convex)



- DNNs with different random initial values end in different minima (ensemble), i.e. different models

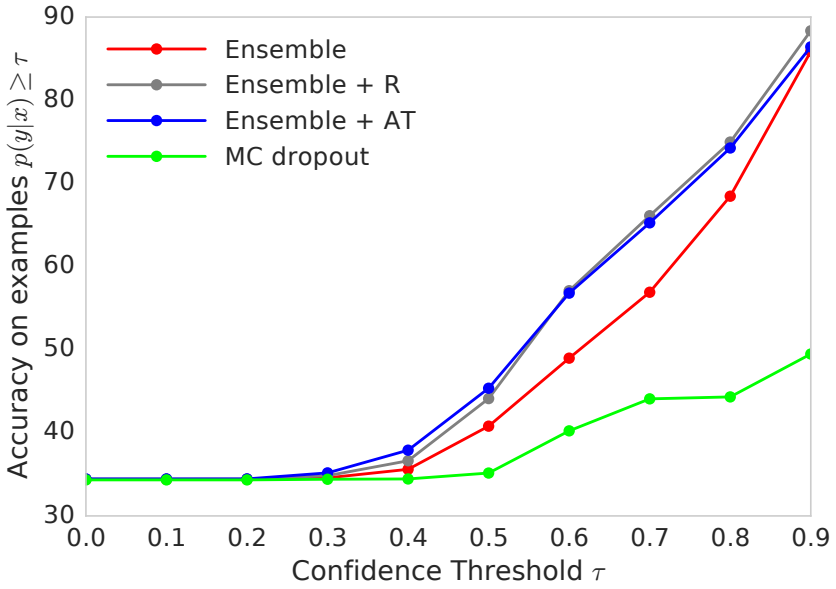


- Known: Ensemble averages with single high probability prediction
- Unknown: Averaged ensemble predictions spread over several classes

Ensemble spread indicates an uncertainty

Predictive uncertainty methods

Mixed known and unknown test sample



- x-axis: prob. threshold
- y-axis" accuracy for sample above threshold

- Ensemble method not generically sufficient, just easy to build intuition
- Many methods and much progress, like MC dropout, ensemble + adversarial training, …
- A typical challenge is to keep it scaleable, e.g. a single DeepJet training already takes **24h**

- Several methods and much recent progress
- Not much used in HEP