Graph Neural Networks for Particle Physics

Part II

曲慧麟 (CERN)

第九届华大QCD讲习班
2021-10-14

Outline

- Part I (yesterday)
  - Motivation: How to represent HEP data for machine learning?
  - Graph neural networks
  - Example applications in HEP
- Part II (today)
  - hands-on tutorial: jet tagging with GNNs
  - practicalities

Jet Tagging

Introduction

- Jet tagging: identifying the hard scattering particle that initiates the jet
  - examples:
    - heavy flavor tagging (bottom/charm)
    - heavy resonance tagging (top/W/Z/Higgs)
    - quark/gluon discrimination
    - exotic jet tagging (displaced, 4-prong, ...)
  - powerful tools for many new physics searches and standard model measurements
- One of the frontiers of ML for HEP
  - playground for novel ML approaches / algorithms
    - rich structure / information in a jet
    - *How far are we from the performance limit?*
  - significant performance improvement in real experiments
  - but also new perspectives and deeper insights into QCD / jet physics

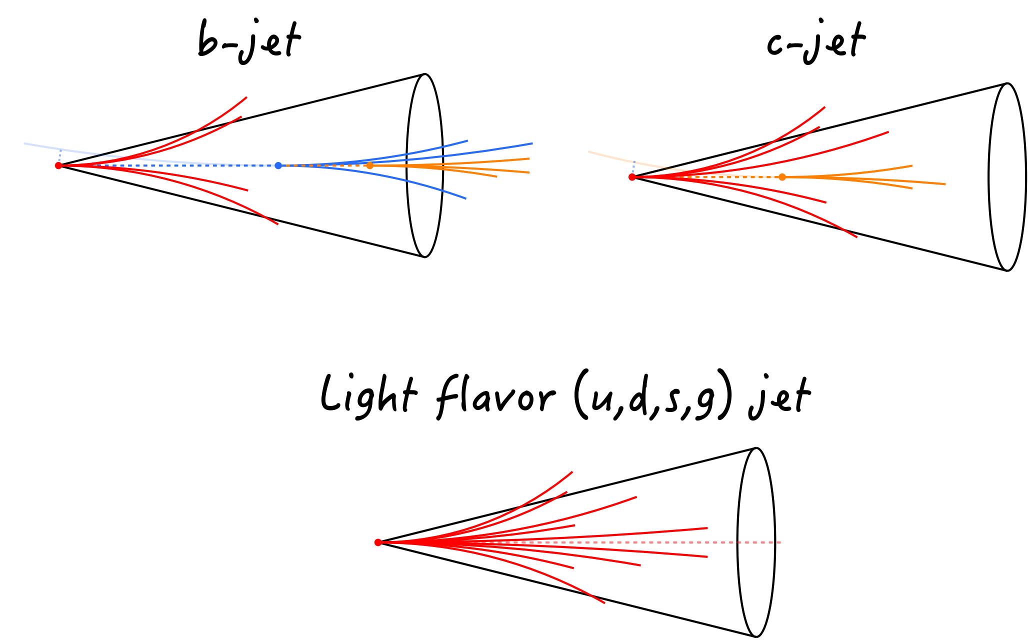


Image credit

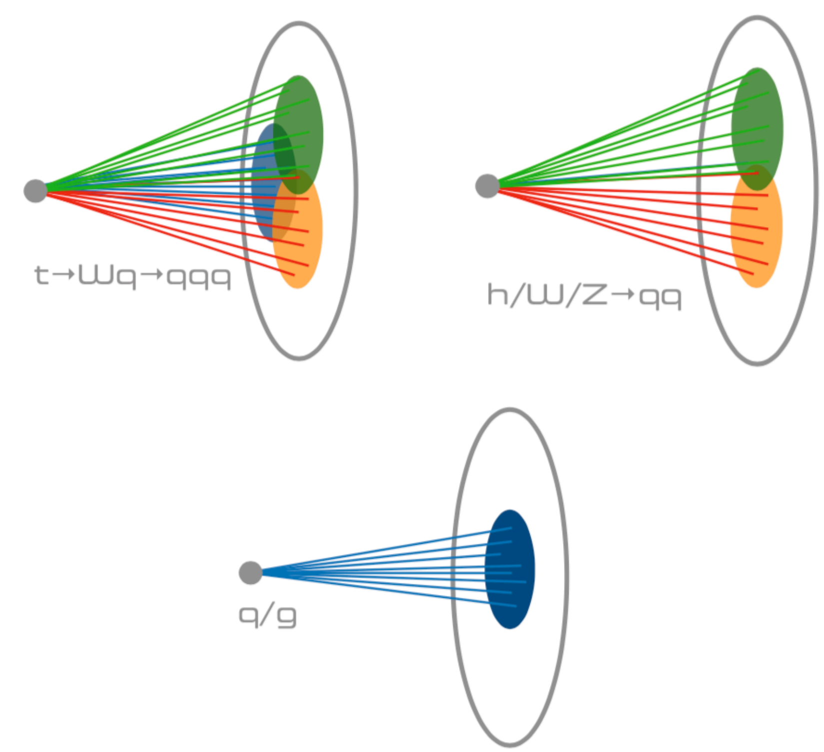
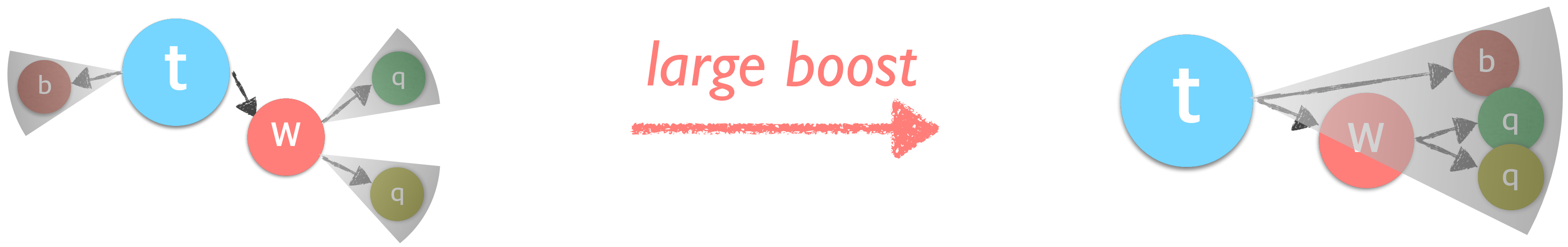


Image credit

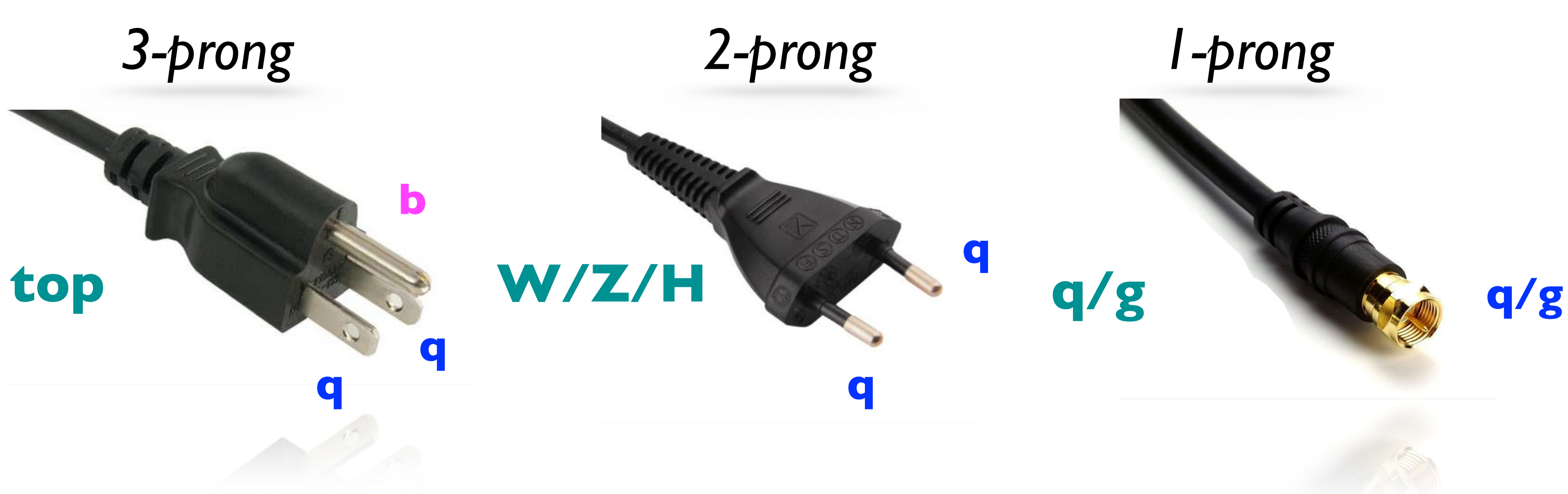
BOOSTED JET TAGGING

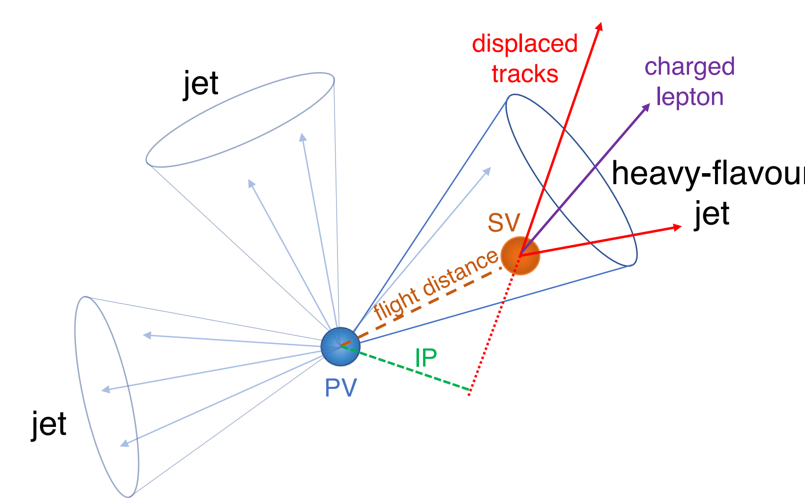
- At high p_T, the decay products from heavy particles (Higgs/W/Z/top) become collimated and can be contained in a single large-R jet



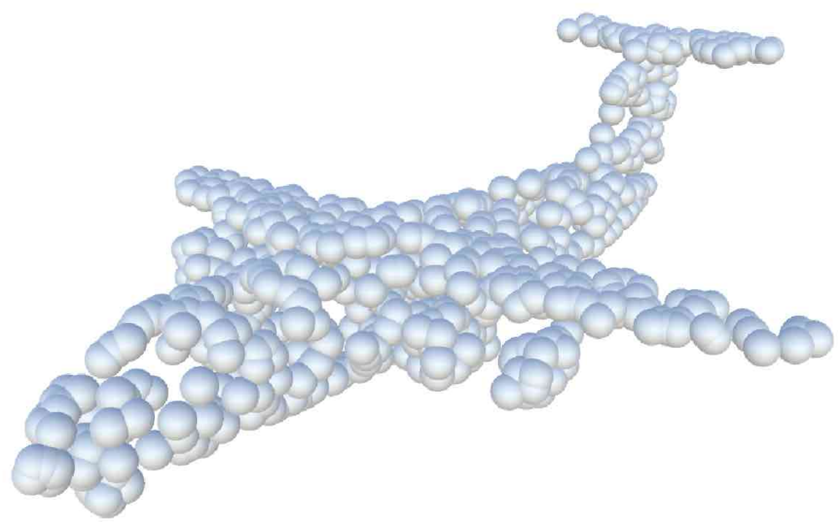
$$\Delta R \sim \frac{2m}{p_T}$$

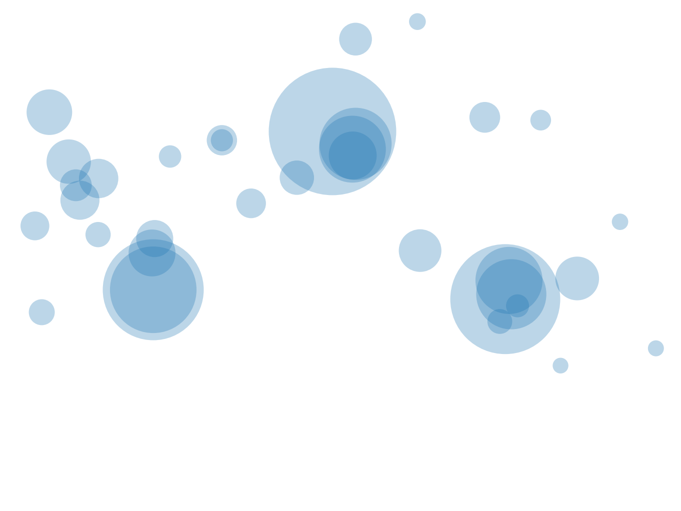
- Large-R jets from resonance (Higgs/W/Z/top) decays exhibit different characteristics that can be used to separate them from jets initiated by QCD radiations
  - different radiation patterns ("**substructure**")
    - 3-prong (top), 2-prong (W/Z/H) vs 1-prong (gluon/light quark jet)
  - different **flavor** content: existence of one or more b-/c-quarks
  - simultaneously exploiting both **substructure** and **flavor** to maximize the performance





JET AS A POINT CLOUD





Point cloud

From Wikipedia, the free encyclopedia

A **point cloud** is a set of data points in space. Point clouds are generally produced by 3D scanners, which measure a large number of points on the external surfaces of objects around them.

Jet (Particle cloud)

From Wikipedia, the free encyclopedia

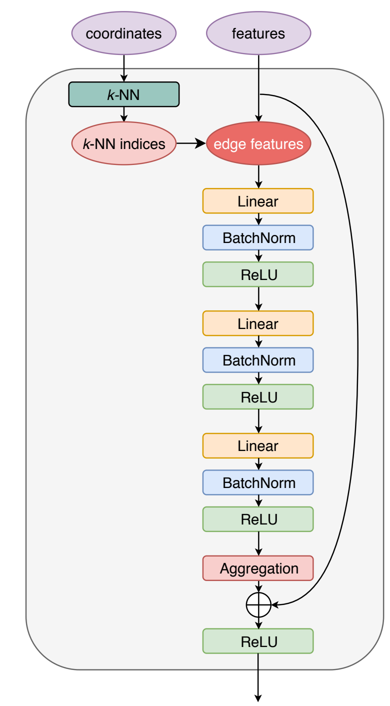
A **jet (particle cloud)** is a set of particles in space. Particle clouds are generally created by clustering a large number of particles measured by particle detectors, e.g., ATLAS EXPERIMENT and CMS.

ARCHITECTURE: *PARTICLENET*

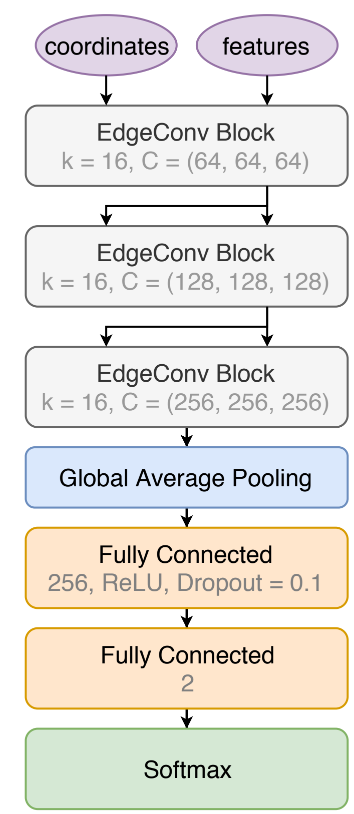
HQ and L. Gouskos
[*Phys.Rev.D 101 (2020) 5, 056019*]

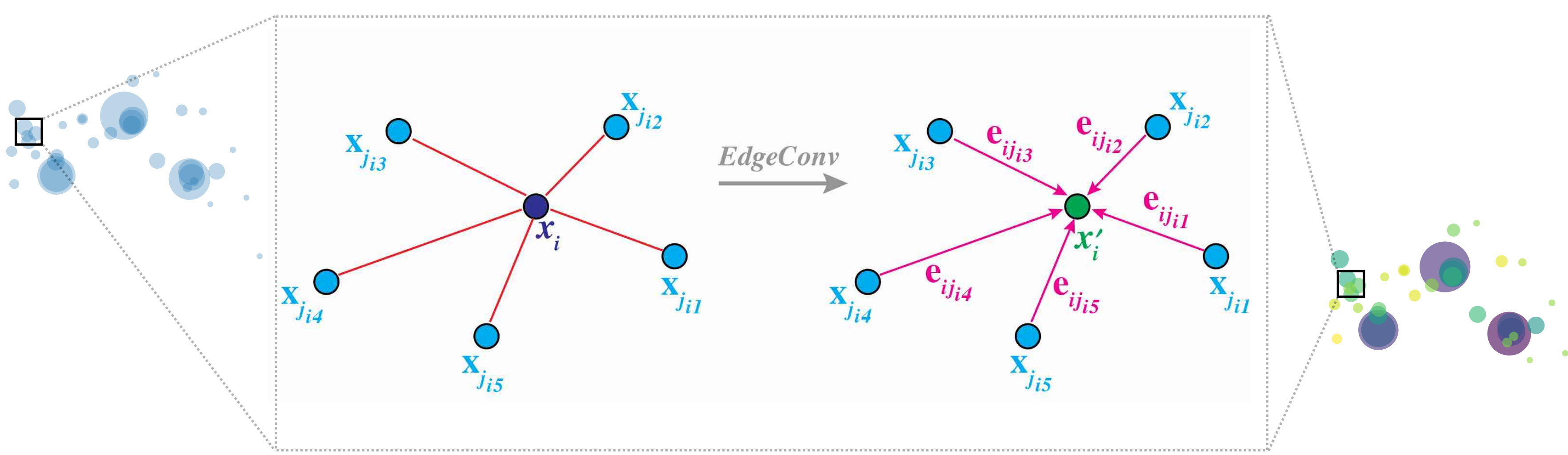
- ParticleNet
  - customized graph neural network architecture for jet tagging with the point cloud approach, based on Dynamic Graph CNN [Y. Wang et al., *arXiv:1801.07829*]
  - explicitly respects the permutation symmetry of the point cloud
- Key building block: *EdgeConv*
  - treating a point cloud as a graph: each point is a vertex
    - for each point, a local patch is defined by finding its k-nearest neighbors
  - designing a permutation-invariant "convolution" function
    - define "edge feature" for each center-neighbor pair: $e_{ij} = h_\Theta(x_i, x_j)$
      - same h_Θ for all neighbor points, and all center points, for symmetry
    - aggregate the edge features in a symmetric way: $x_i' = \text{mean}_j\, e_{ij}$
- EdgeConv can be stacked to form a deep network
  - learning both local and global structures, in a hierarchical way

EdgeConv block



ParticleNet architecture





Performance of ParticleNet

- Performance on the public top tagging benchmark dataset
  - ParticleNet achieves the highest performance among all algorithms

G. Kasieczka et al. [SciPost Phys. 7 (2019) 014]

	AUC	Acc	$1/\epsilon_B$ ($\epsilon_S = 0.3$) single	mean	median	#Param
CNN [16]	0.981	0.930	914±14	995±15	975±18	610k
ResNeXt [30]	0.984	0.936	1122±47	1270±28	1286±31	1.46M
TopoDNN [18]	0.972	0.916	295±5	382± 5	378 ± 8	59k
Multi-body N-subjettiness 6 [24]	0.979	0.922	792±18	798±12	808±13	57k
Multi-body N-subjettiness 8 [24]	0.981	0.929	867±15	918±20	926±18	58k
TreeNiN [43]	0.982	0.933	1025±11	1202±23	1188±24	34k
P-CNN	0.980	0.930	732±24	845±13	834±14	348k
ParticleNet [47] (Preliminary ver.)	0.985	0.938	1298±46	1412±45	1393±41	498k
LBN [19]	0.981	0.931	836±17	859±67	966±20	705k
LoLa [22]	0.980	0.929	722±17	768±11	765±11	127k
Energy Flow Polynomials [21]	0.980	0.932	384			1k
Energy Flow Network [23]	0.979	0.927	633±31	729±13	726±11	82k
Particle Flow Network [23]	0.982	0.932	891±18	1063±21	1052±29	82k
GoaT	0.985	0.939	1368±140		1549±208	35k
ParticleNet-Lite	0.984	0.937	1262±49			26k
ParticleNet	0.986	0.940	1615±93			366k

Architecture used by DeepAK8 (→ P-CNN)

Ensemble of all taggers (→ GoaT)

Hands-on Tutorial

Weaver

- https://github.com/hqucms/weaver
  - a streamlined yet flexible machine learning R&D framework for HEP
  - data loading: both in-memory and out-of-memory (scalable to O(100M) entries/TB level)
    - supports common HEP data formats: ROOT, HDF5, awkward array
  - input preprocessing: transformation/standardization, reweighting/sampling, padding, shuffling, etc.
  - training: built-in training/validation loop for classification and regression
    - monitoring/visualization via TensorBoard
  - deployment: exporting PyTorch model to ONNX
    - optimized inference w/ ONNXRuntime on CPUs/GPUs in Python/C/C++/etc.
- To train a neural network with Weaver:
  - a YAML data configuration file describing how to process the input data.
  - a python model configuration file providing the neural network module and the loss function

Top Tagging Dataset

- https://zenodo.org/record/2603256
  - hadronic tops for signal, qcd dijets background, both generated with Pythia8
    - no MPI/pile-up included
    - Delphes ATLAS detector card
  - clustering of particle-flow entries (produced by Delphes E-flow) into anti-k_T 0.8 jets in the p_T range [550,650] GeV
    - all top jets are matched to a parton-level top within $\Delta R = 0.8$, and to all top decay partons within 0.8
  - the leading 200 jet constituent four-momenta are stored, with zero-padding for jets with fewer than 200
    - constituents are sorted by p_T, with the highest p_T one first
  - 1.2M / 400k / 400k for train / val / testing

HANDS-ON TIME!

- To make it easier to copy the commands:
  - https://gist.github.com/hqucms/3a9d9e9b53bf21253831108e8dbf8889

- Setup Weaver and weaver-benchmark

```
# prerequisite: install the dependent packages
# https://github.com/hqucms/weaver#set-up-your-environmen

git clone https://github.com/hqucms/weaver.git
cd weaver
git pull # update to the latest status
git clone https://github.com/hqucms/weaver-benchmark.git
```

HANDS-ON TIME!

- To make it easier to copy the commands:
  - https://gist.github.com/hqucms/3a9d9e9b53bf21253831108e8dbf8889

- Download and convert the dataset

```
# in the weaver/ directory
mkdir top-dataset
cd top-dataset
# download the top-tagging dataset
curl -O 'https://zenodo.org/record/2603256/files/train.h5'
curl -O 'https://zenodo.org/record/2603256/files/val.h5'
curl -O 'https://zenodo.org/record/2603256/files/test.h5'
cd ..
# back in the weaver/ directory
# convert the h5 files to awkward arrays
python weaver-benchmark/utils/convert_top_datasets.py -i top-dataset/ -o
top-dataset/converted
```

HANDS-ON TIME!

- To make it easier to copy the commands:
  - https://gist.github.com/hqucms/3a9d9e9b53bf21253831108e8dbf8889

- Training the ParticleNet model

```
# in the weaver/ directory
python train.py \
 --data-train top-dataset/converted/train_file_0.awkd \
 --data-val top-dataset/converted/val_file_0.awkd \
 --data-test top-dataset/converted/test_file_0.awkd \
 --data-config weaver-benchmark/data/top/pf_points_features.yaml \
 --network-config weaver-benchmark/networks/top/particlenet_pf.py \
 --model-prefix outputs/{auto}/net \
 --predict-output pred.root \
 --num-workers 1 --fetch-step 1 --data-fraction 1 \
 --gpus 0 --batch-size 128 --num-epochs 20 \
 --start-lr 5e-3 --optimizer ranger \
 --log logs/{auto}.log --tensorboard _particle_net
```

Hands-on Time!

- To make it easier to copy the commands:
  - https://gist.github.com/hqucms/3a9d9e9b53bf21253831108e8dbf8889

- Training the Deep Set / Particle Flow Network

```
# in the weaver/ directory
python train.py \
 --data-train top-dataset/converted/train_file_0.awkd \
 --data-val top-dataset/converted/val_file_0.awkd \
 --data-test top-dataset/converted/test_file_0.awkd \
 --data-config weaver-benchmark/data/top/pf_features_mask.yaml \
 --network-config weaver-benchmark/networks/top/pfn_pf.py \
 --model-prefix outputs/{auto}/net \
 --predict-output pred.root \
 --num-workers 1 --fetch-step 1 --data-fraction 1 \
 --gpus 1 --batch-size 128 --num-epochs 20 \
 --start-lr 5e-3 --optimizer ranger \
 --log logs/{auto}.log --tensorboard _pfn
```

```
--gpus '' # to run on CPU
```

HANDS-ON TIME!

- To make it easier to copy the commands:
  - https://gist.github.com/hqucms/3a9d9e9b53bf21253831108e8dbf8889

- Monitor training progress with TensorBoard

```
# in the weaver/ directory
tensorboard --logdir=runs
# open tensorboard in the web browser
# http://localhost:6006
```

HANDS-ON TIME!

- To make it easier to copy the commands:
  - https://gist.github.com/hqucms/3a9d9e9b53bf21253831108e8dbf8889

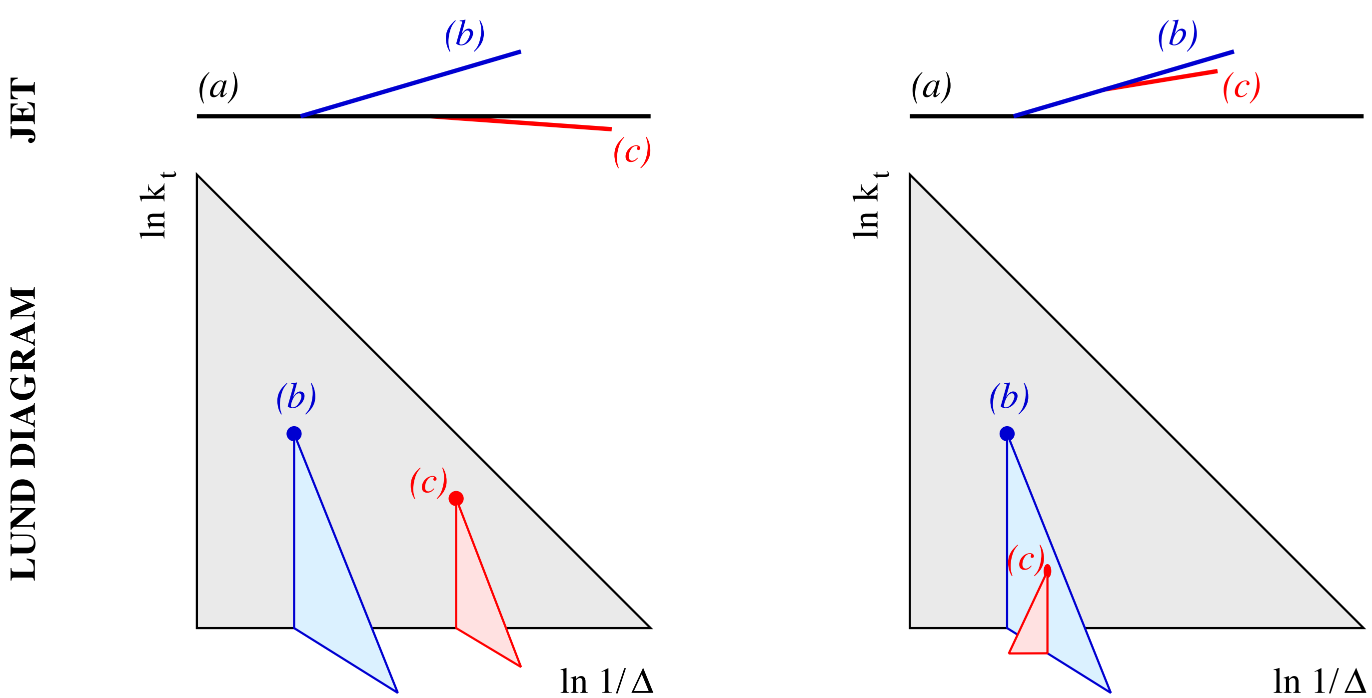
- Evaluate the performance

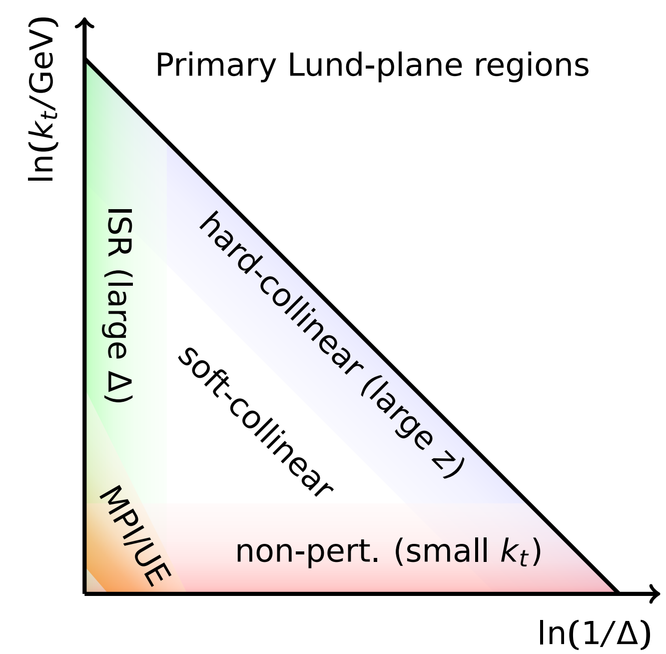
```
# in the weaver/ directory
jupyter notebook
# open jupyter in the web browser
# http://localhost:8888
```

LundNet:
Jet tagging in the Lund plane with graph networks

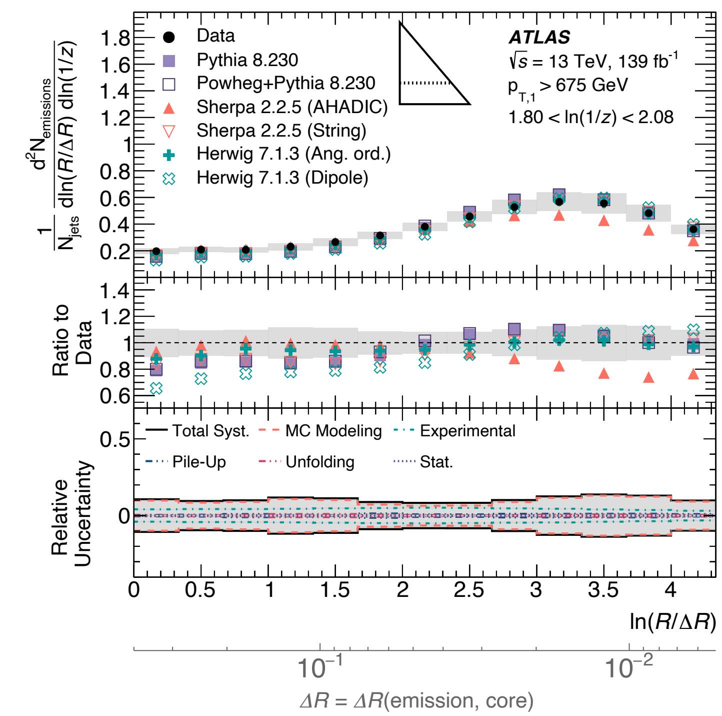
Jets in the Lund Plane

- Jets in the Lund plane
  - each emission (splitting) is mapped to a point in the 2D (angle, transverse momentum) plane
  - further emissions (of the secondary particles) are represented in additional leaf planes





- The Lund plane provides an efficient description of the radiation patterns within a jet
  - different kinematic regimes are clearly separated in the Lund plane
  - often used in the discussion of resummations of large logarithms in perturbation theory / Monte Carlo parton shower generators
  - can also be measured experimentally [ATLAS, PRL 124, 222002 (2020)]

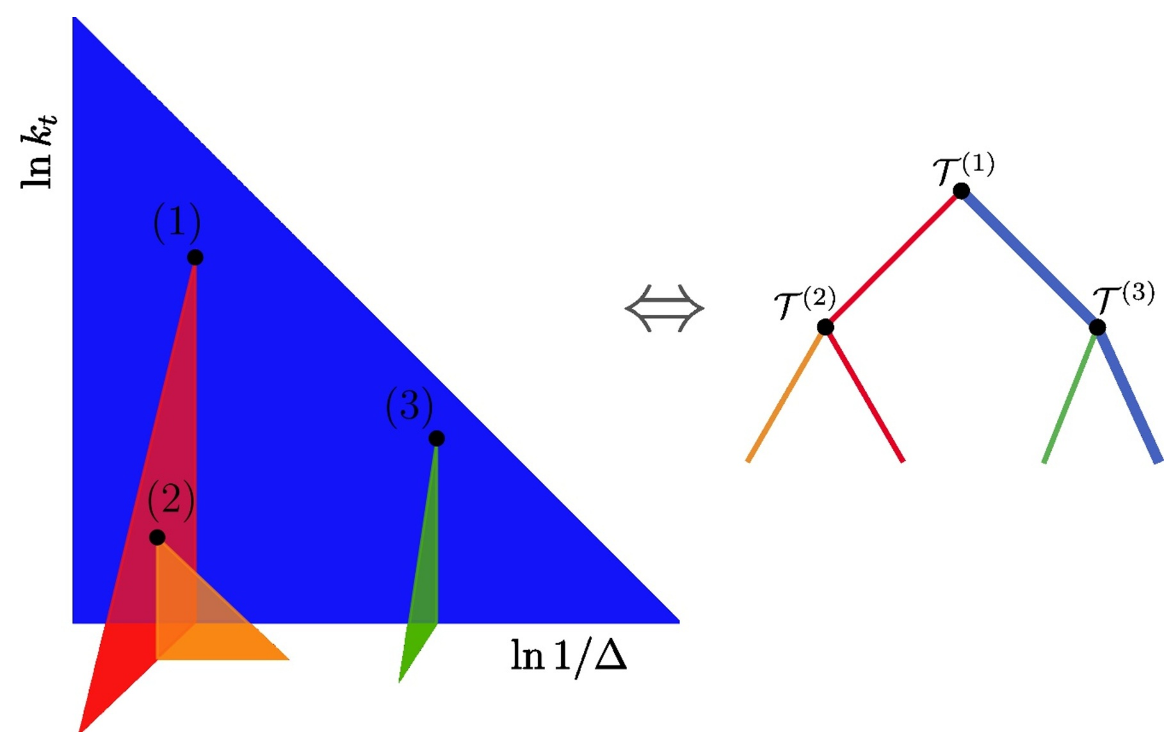


Constructing the Lund Plane

- The Lund plane of a jet can be constructed in the following way:
  - (1) recluster a jet *j* with the Cambridge/Aachen algorithm.
  - (2) undo the last clustering step, defining two subjets j_a, j_b ordered in p_T.
  - (3) a set of kinematic variables (denoted as $\mathscr{T}^{(i)}$) can be defined for the current splitting:

$$\Delta^2 = (y_a - y_b)^2 + (\phi_a - \phi_b)^2, \quad k_t \equiv p_{tb}\Delta_{ab}, \quad m^2 \equiv (p_a + p_b)^2,$$
$$z \equiv \frac{p_{tb}}{p_{ta}+p_{tb}}, \quad \kappa \equiv z\Delta\,, \quad \psi \equiv \tan^{-1}\frac{y_b - y_a}{\phi_b - \phi_a},$$

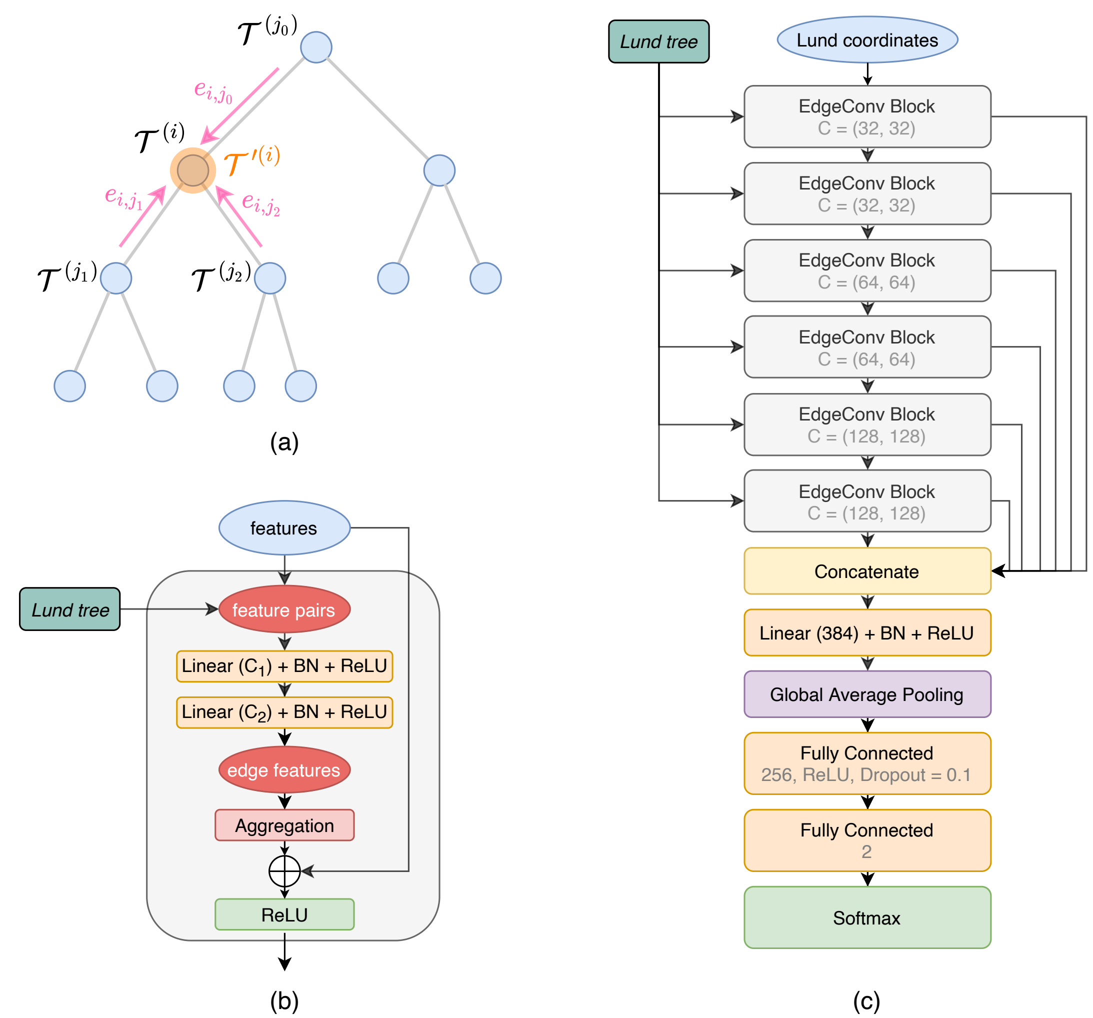
  - (4) repeat (2) and (3) on j_a, j_b until j_a, j_b become single particles.
- Equivalently, the full Lund plane can also be represented as a binary Lund tree, with a tuple of variables $\mathscr{T}^{(i)}$ for each node i



LundNet

F. Dreyer and HQ
[JHEP 03 (2021) 052]

- The Lund plane/Lund tree essentially encodes the full radiation patterns of a jet
  - a natural input for ML algorithms on jets [c.f. F. Dreyer, G. Salam and G. Soyez, JHEP 12 (2018) 064]
- LundNet: a graph neural network on the Lund tree
  - overall architecture similar to ParticleNet
    - each node exchanges information with one parent and two child nodes, using EdgeConv
    - global pooling of all nodes at the end to get feature maps for final classification
  - but unlike ParticleNet:
    - no expensive k-nearest neighbor finding needed
      - graph structure determined by the Lund tree
    - only 3 (instead of 16) neighbors in EdgeConv
      - significantly lower computational cost

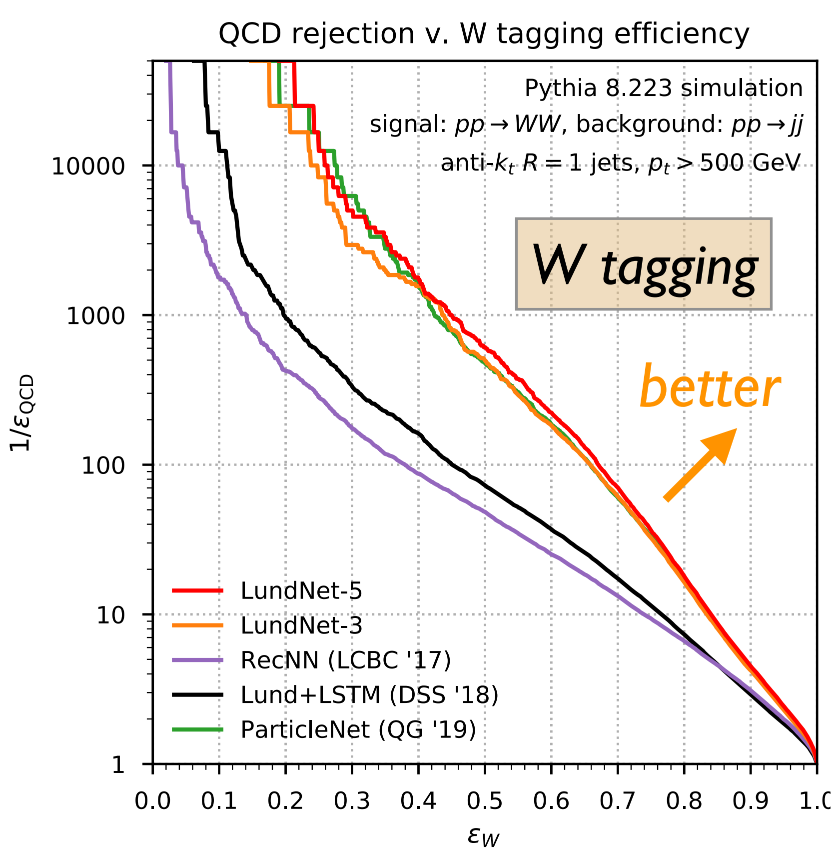


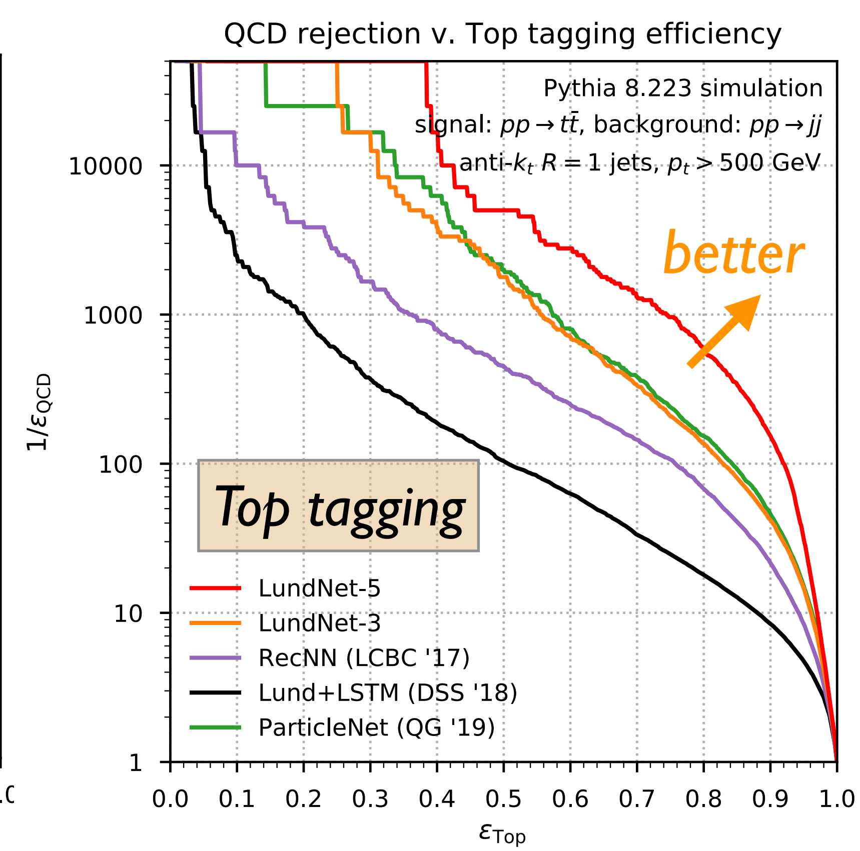
- Two variants of LundNet studied
  - LundNet-5: using all five Lund variables, $(\ln k_t, \ln \Delta, \ln z, \ln m, \psi)$
  - LundNet-3: using only three Lund variables, $(\ln k_t, \ln \Delta, \ln z)$

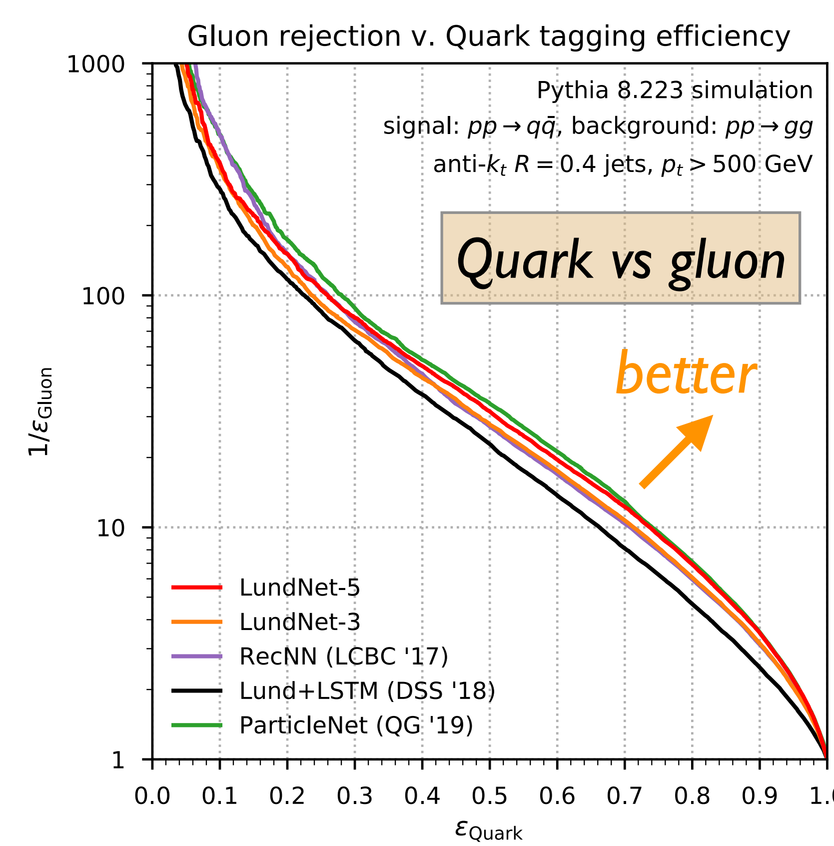
PERFORMANCE OF LUNDNET

F. Dreyer and HQ [JHEP 03 (2021) 052]

- Significantly improved performance for top tagging compared to ParticleNet
  - similar performance for W tagging and q/g discrimination
- Almost an order of magnitude speed-up in training/inference time compared to ParticleNet







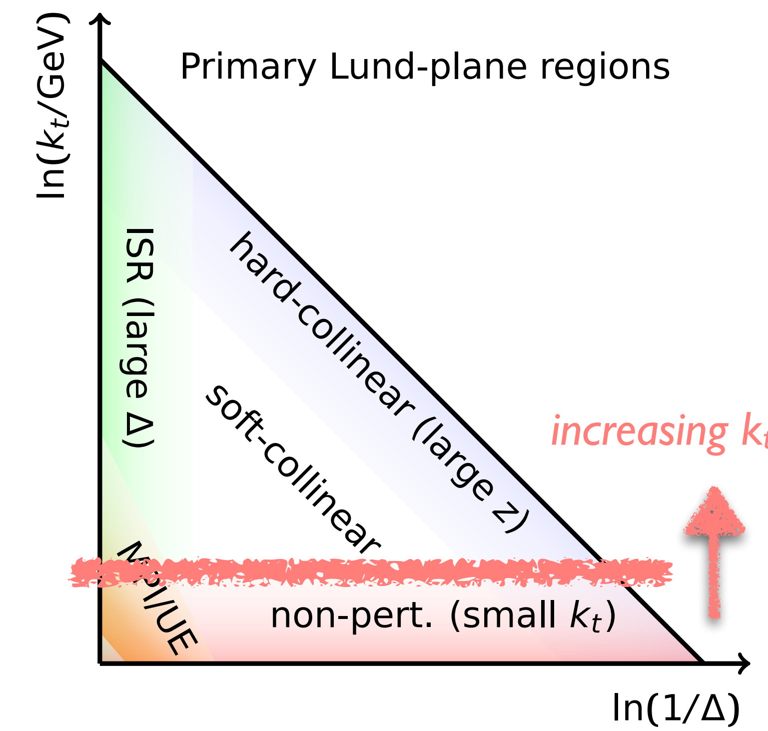
	Number of parameters	Training time [ms/sample/epoch]	Inference time [ms/sample]
LundNet	395k	0.472	0.117
ParticleNet	369k	3.488	1.036
Lund+LSTM	67k	0.424	0.131

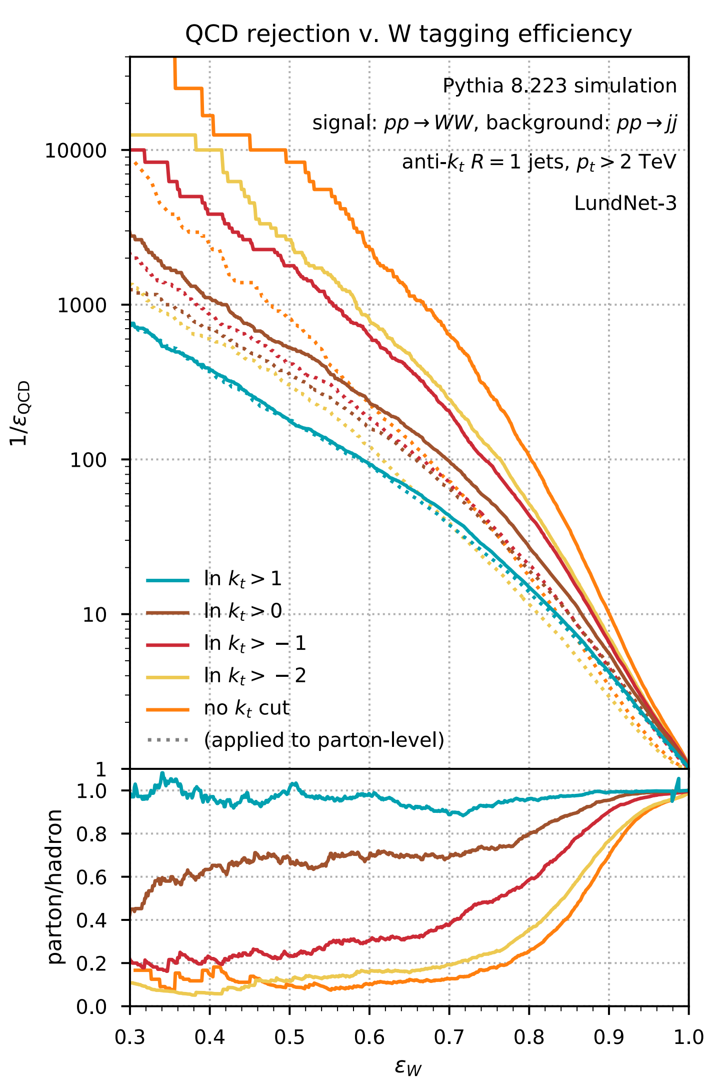
DGL + PyTorch
Nvidia GTX 1080Ti
batch size = 256

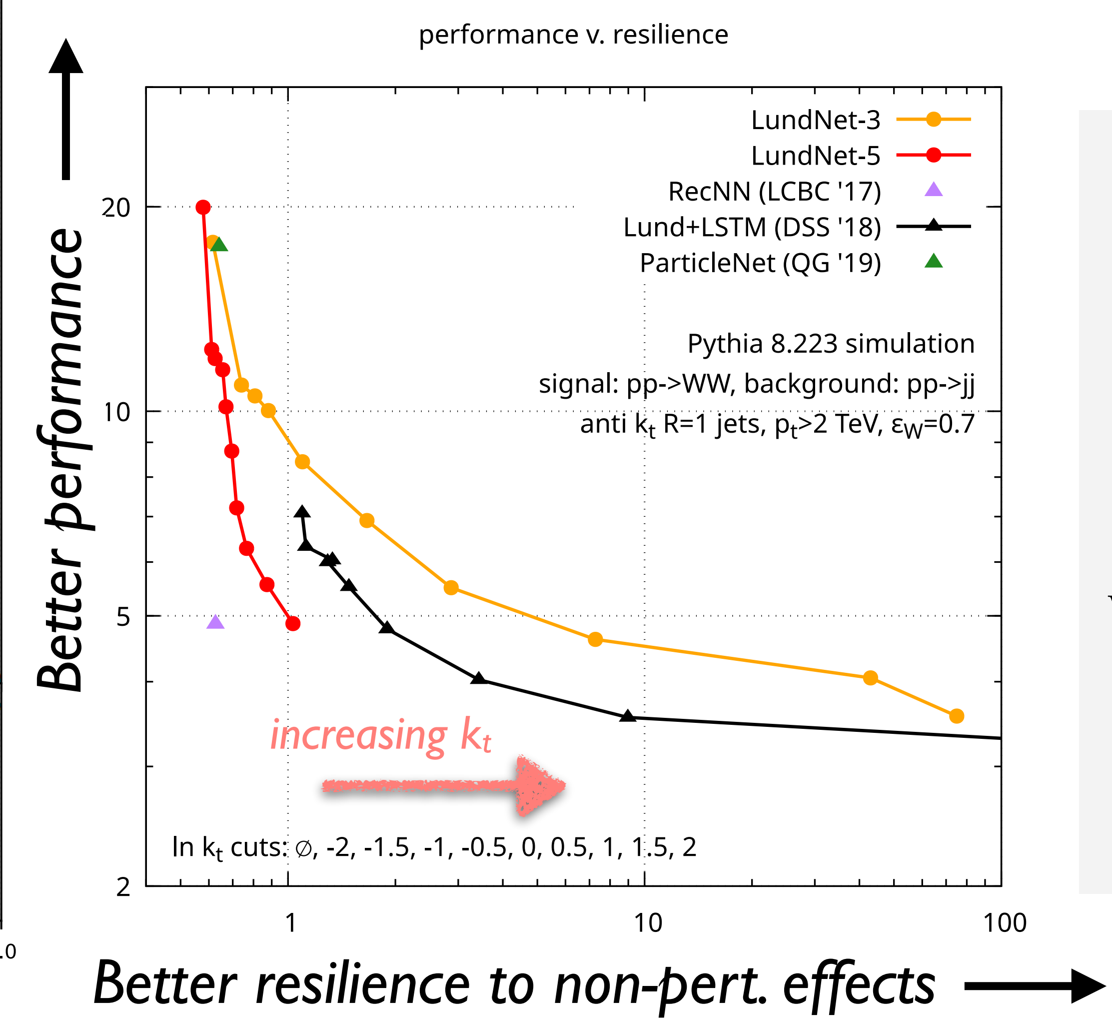
Robustness of LundNet

F. Dreyer and HQ [JHEP 03 (2021) 052]

- Moreover, LundNet provides a systematic way to control the robustness of the tagger
  - robustness assessed by applying the model trained on hadron-level samples to parton-level samples and compare the difference
  - the non-perturbative region can be effectively rejected by applying a k_t cut on the Lund plane, therefore improving the robustness of the tagger against non-perturbative effects
  - LundNet-3 shows much higher resilience than LundNet-5







Performance: $\frac{\epsilon_W}{\sqrt{\epsilon_{\mathrm{QCD}}}}$

Resilience:

$$\zeta_{\mathrm{NP}} = \left(\frac{\Delta\epsilon_W^2}{\langle\epsilon\rangle_W^2} + \frac{\Delta\epsilon_{\mathrm{QCD}}^2}{\langle\epsilon\rangle_{\mathrm{QCD}}^2} \right)^{-1/2}$$

where $\Delta\epsilon = \epsilon - \epsilon'$ and $\langle\epsilon\rangle = 1/2\,(\epsilon + \epsilon')$

ϵ : *hadron-level*

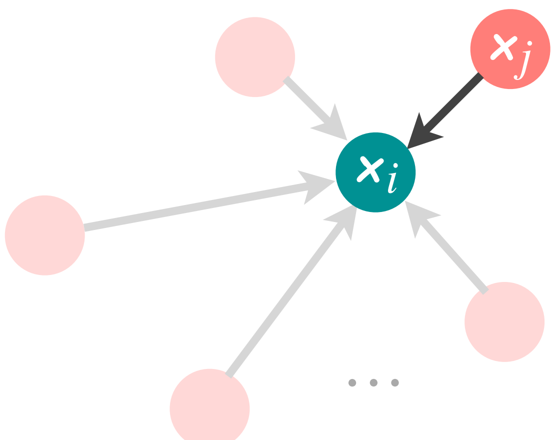
ϵ' : *parton-level*

ParticleNeXt:
Pushing the limit of jet tagging

ParticleNeXt: Pairwise Features

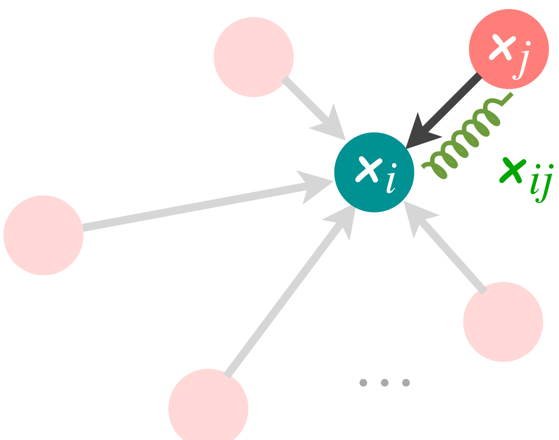
- ParticleNeXt: next-generation of ParticleNet, for better performance
- The first enhancement is the addition of (explicit) pairwise features on the edges

ParticleNet



$$e_{ij} = MLP(x_i, x_j)$$

ParticleNeXt



$$e_{ij} = MLP(x_i, x_j, x_{ij})$$

- Examples of pairwise features:

$$\Delta_{ij}^2 \equiv (y_i - y_j)^2 + (\phi_i - \phi_j)^2, \quad m^2 \equiv (p_i + p_j)^2,$$

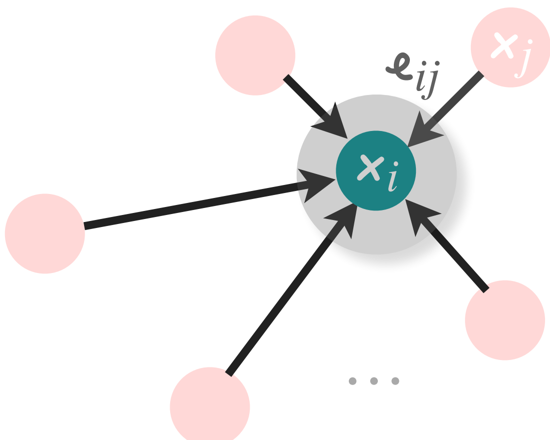
$$k_T \equiv \min(p_{T,i}, p_{T,j})\,\Delta_{ij}, \quad z \equiv \frac{\min(p_{T,i}, p_{T,j})}{p_{T,i} + p_{T,j}}$$

(use the logarithm to improve stability of the training)

PARTICLENEXT: ATTENTIVE POOLING

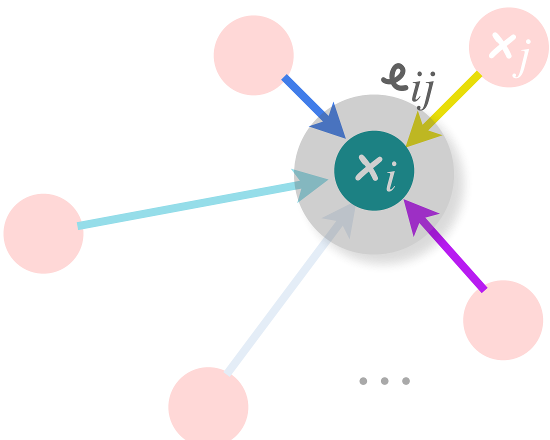
- Use attention-based pooling to increase the expressive power
  - for both the **local neighborhood pooling**, and the final **global pooling**

ParticleNet



$$z_i = \mathrm{mean}_j(e_{ij})$$

ParticleNeXt



$$\mathrm{attn}_{ij} = \mathrm{MLP}(e_{ij})$$

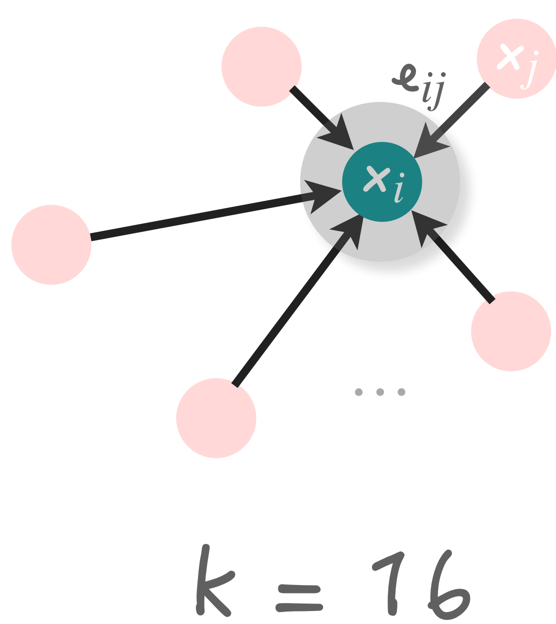
$$w_{ij} = \mathrm{softmax}_j(\mathrm{attn}_{ij})$$

$$z_i = \Sigma_j(w_{ij}\, e_{ij})$$

ParticleNeXt: Multi-Scale Aggregation

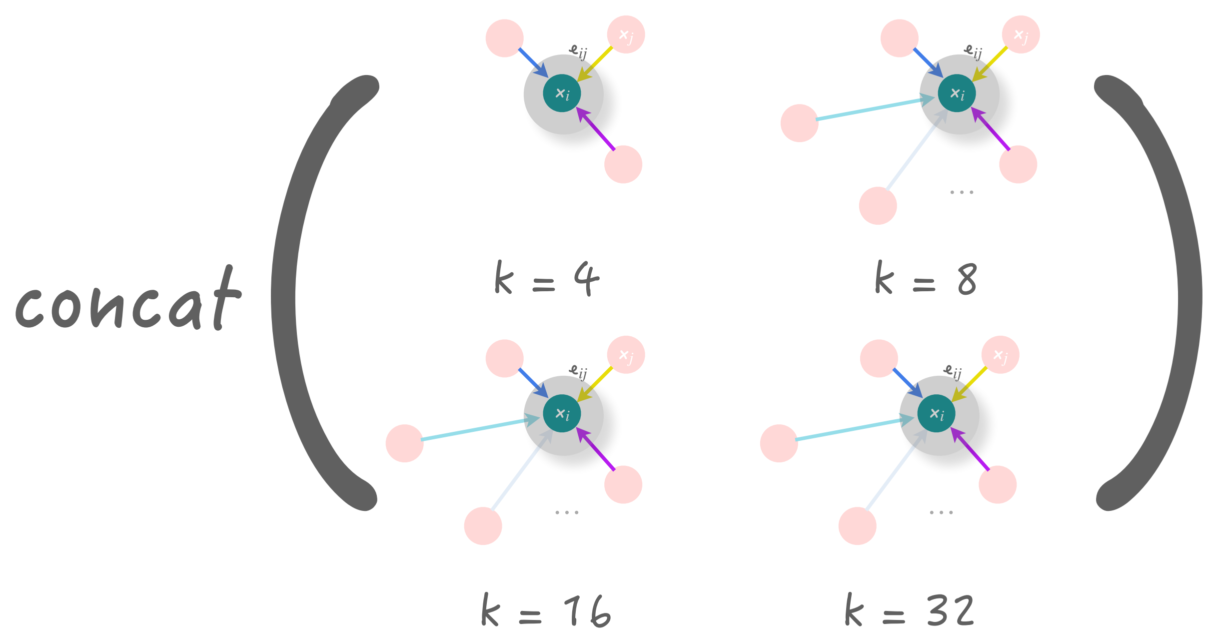
- Introduce multi-scale aggregation to better capture both short- and long-range correlations
  - perform local aggregation for the 4, 8, 16 and 32 nearest neighbors (with different attentive pooling) and combine the 4 aggregated representations with a MLP
  - on the other hand: remove dynamic kNN (based on learned features), i.e., use only kNN in $\eta-\phi$ space, to reduce computational cost
    - in this case the kNN needs to be performed only once, and then the graph connectivity is fixed

ParticleNet



$$x_i' = z_i$$

ParticleNeXt

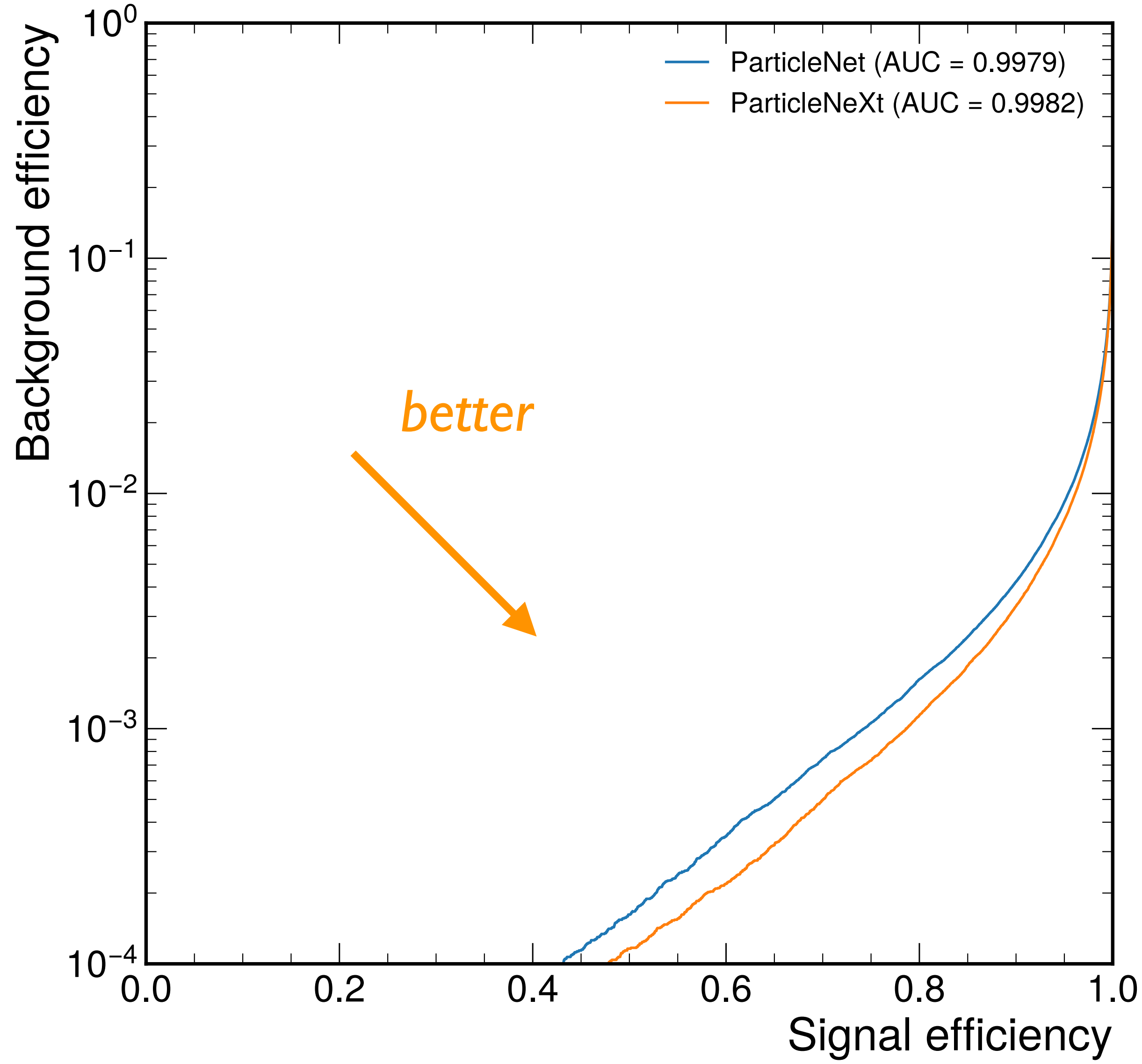


$$x_i' = MLP(z_i^{concat})$$

DATASET

- A new jet tagging dataset was generated for the development of ParticleNeXt
  - all events are generated with MadGraph5_aMC@NLO v3.1.1 at LO and interfaced with Pythia v8.245 for parton shower (w/ the default tune and MPI enabled)
  - fast detector simulation w/ Delphes v3.5.0, using the CMS card
    - tracking resolution parametrization based on the CMS Run1 performance [1405.6569]
  - jets clustered from the Delphes e-flow objects using the anti-kt algorithm w/ R=0.8
    - only consider jets w/ 500 < p_T 1000 GeV, and $|\eta| < 2$
    - input features for each jet constituent particle: 4-momenta, PID, **impact parameters and errors**
  - top-tagging benchmark:
    - Top quark jets: $pp \to t\bar{t}\ (t \to bW, W \to qq')$
      - truth matching criteria: ΔR(jet, q) < 0.8 for all three quarks from hadronic top decay
    - QCD jets: $pp \to Z(\to \nu\bar{\nu}) + j\,(j = u, d, s, c, b, g)$
  - Higgs-tagging benchmark:
    - Higgs boson jets: $pp \to hh\ (h \to b\bar{b})$
      - truth matching criteria: ΔR(jet, b) < 0.8 for both quarks from the Higgs decay
    - QCD jets: $pp \to Z(\to \nu\bar{\nu}) + j\,(j = u, d, s, c, b, g)$

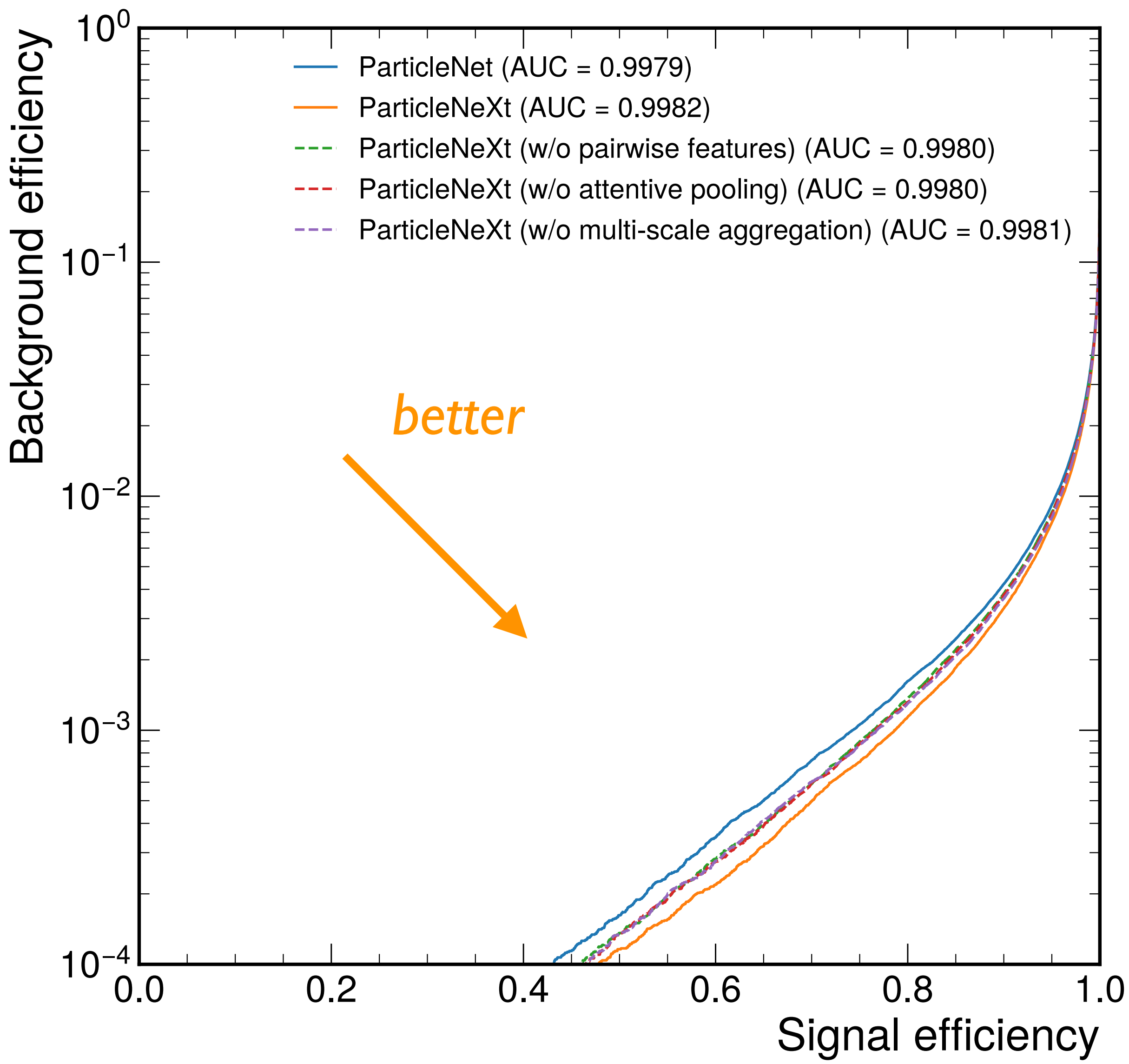
PERFORMANCE: TOP TAGGING



- Training/validation/test splitting:
  - 1.6M / 0.4M / 2M
- Training repeated for 3 times starting from randomly initialized weights
  - the median-accuracy training is reported, and the standard deviation of the 3 trainings is quoted as the uncertainty
- Significant improvement in background rejection w/ ParticleNeXt
  - ~50% higher BKG rejection (@ϵ_S = 70%)
  - computational cost still under control

	Accuracy	AUC	$1/\varepsilon_b$ at $\varepsilon_s = 70\%$	$1/\varepsilon_b$ at $\varepsilon_s = 50\%$	Parameters	Inference time (CPU)	Inference time (GPU)	Training time (GPU)
ParticleNet	0.980	0.9979	1342 ± 4	6173 ± 425	366k	23 ms	0.30 ms	1.0 ms
ParticleNeXt	**0.981**	**0.9982**	$\mathbf{2008 \pm 75}$	$\mathbf{8621 \pm 309}$	560k	30 ms	0.54 ms	1.7 ms

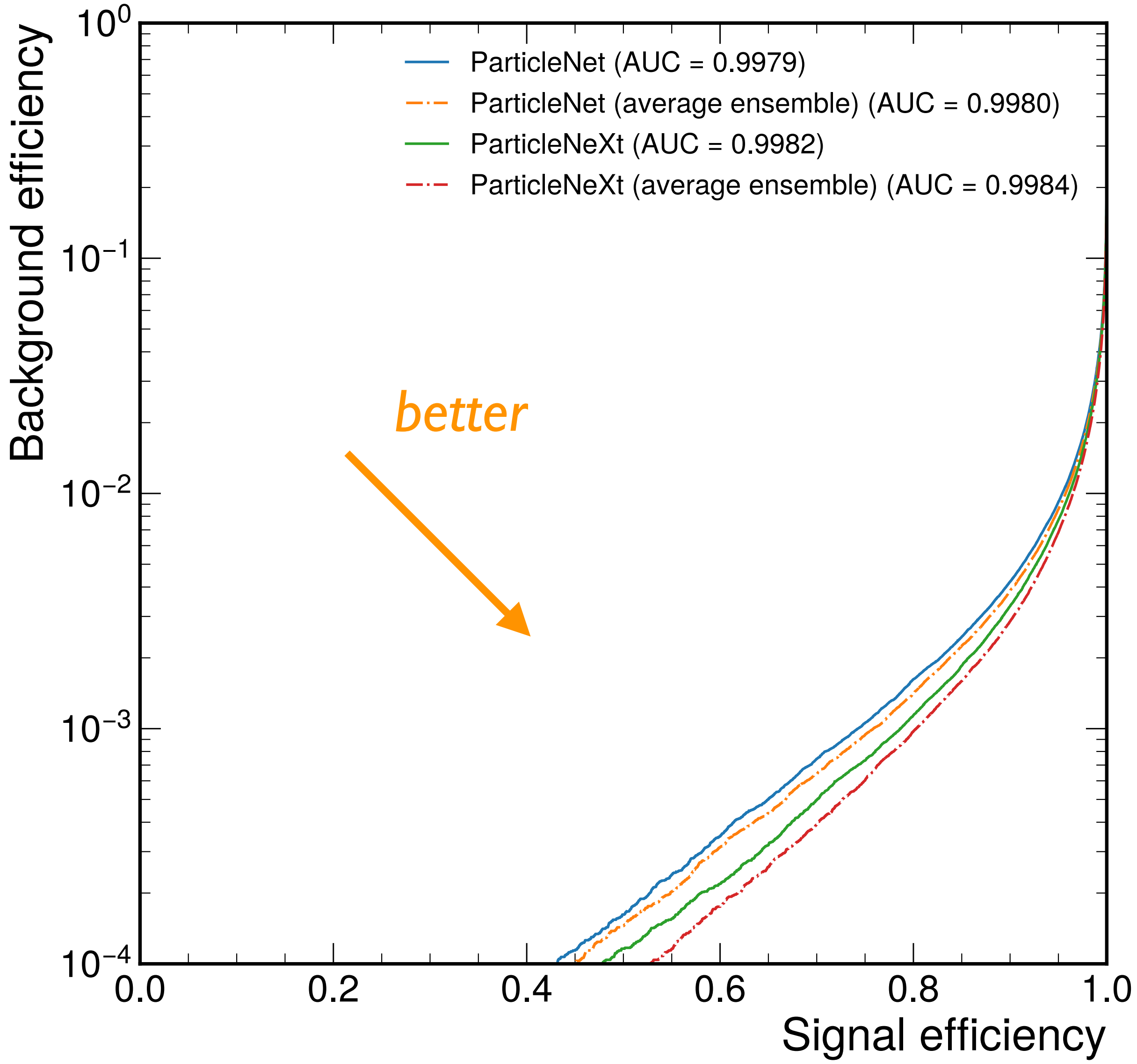
Ablation Study



- Investigated the effects of the new features of ParticleNeXt by removing each of them and repeat the training
  - all the new features contribute
  - ~20% loss in BKG rejection if any of the three is removed

	Accuracy	AUC	$1/\varepsilon_b$ at $\varepsilon_s = 70\%$	$1/\varepsilon_b$ at $\varepsilon_s = 50\%$
ParticleNet	0.980	0.9979	1342 ± 4	6173 ± 425
ParticleNeXt	**0.981**	**0.9982**	$\mathbf{2008 \pm 75}$	$\mathbf{8621 \pm 309}$
ParticleNeXt (w/o pairwise features)	0.980	0.9980	1695 ± 70	7353 ± 193
ParticleNeXt (w/o attentive pooling)	0.980	0.9981	1689 ± 72	7463 ± 696
ParticleNeXt (w/o multi-scale aggregation)	0.981	0.9980	1664 ± 57	7407 ± 193

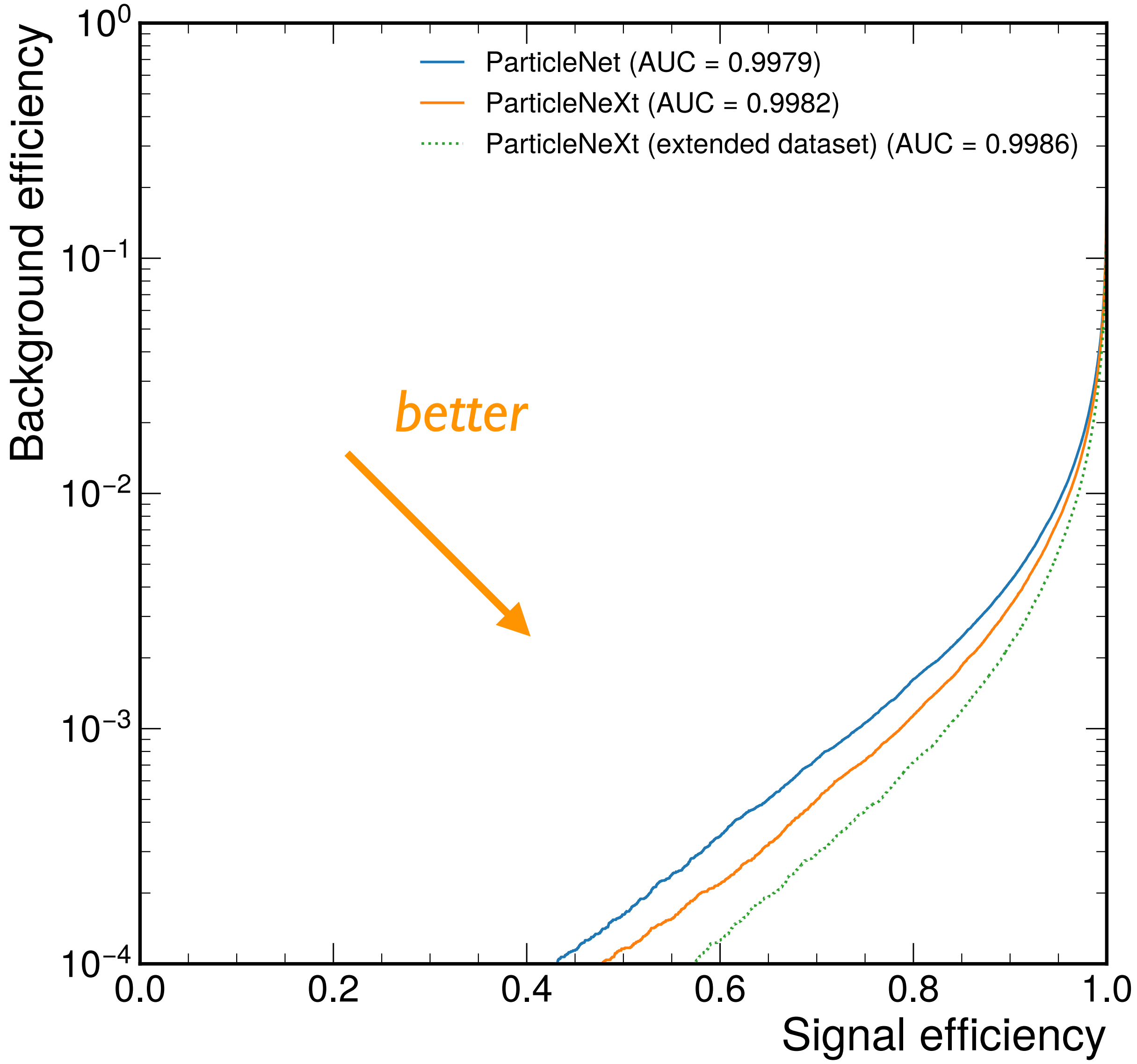
Model Ensembling



- Model ensembling still helps, even for the new ParticleNeXt
  - ensembling method: average the DNN outputs from the 3 independent trainings
  - ~30% improvement for ParticleNeXt with the 3-model ensemble
    - ~15% for ParticleNet

	Accuracy	AUC	$1/\varepsilon_b$ at $\varepsilon_s = 70\%$	$1/\varepsilon_b$ at $\varepsilon_s = 50\%$
ParticleNet	0.980	0.9979	1342 ± 4	6173 ± 425
ParticleNeXt	0.981	0.9982	2008 ± 75	8621 ± 309
ParticleNet (average ensemble)	0.980	0.9980	1558	6897
ParticleNeXt (average ensemble)	**0.982**	**0.9984**	**2558**	**11494**

EXTENDED TRAINING DATASET



- Training on a larger dataset
  - training/validation/test splitting:
    - 10M / 1M / 2M
  - i.e., 5x more jets for training compared to the baseline dataset
- Substantial gain in performance
  - ~70% higher BKG rejection (@ϵ_S = 70%)
- *Question:* Can we encode more physics into the network to make the training more data-efficient?

	Accuracy	AUC	$1/\varepsilon_b$ at $\varepsilon_s = 70\%$	$1/\varepsilon_b$ at $\varepsilon_s = 50\%$
ParticleNet	0.980	0.9979	1342 ± 4	6173 ± 425
ParticleNeXt	0.981	0.9982	2008 ± 75	8621 ± 309
ParticleNeXt (extended dataset)	**0.983**	**0.9986**	**3378**	**15873**

Performance on Public Benchmarks

Top tagging landscape

	AUC	Acc	$1/\epsilon_B$ ($\epsilon_S = 0.3$) single	mean	median	#Param
CNN [16]	0.981	0.930	914±14	995±15	975±18	610k
ResNeXt [30]	0.984	0.936	1122±47	1270±28	1286±31	1.46M
TopoDNN [18]	0.972	0.916	295±5	382± 5	378 ± 8	59k
Multi-body N-subjettiness 6 [24]	0.979	0.922	792±18	798±12	808±13	57k
Multi-body N-subjettiness 8 [24]	0.981	0.929	867±15	918±20	926±18	58k
TreeNiN [43]	0.982	0.933	1025±11	1202±23	1188±24	34k
P-CNN	0.980	0.930	732±24	845±13	834±14	348k
ParticleNet [47]	0.985	0.938	1298±46	1412±45	1393±41	498k
LBN [19]	0.981	0.931	836±17	859±67	966±20	705k
LoLa [22]	0.980	0.929	722±17	768±11	765±11	127k
Energy Flow Polynomials [21]	0.980	0.932	384			1k
Energy Flow Network [23]	0.979	0.927	633±31	729±13	726±11	82k
Particle Flow Network [23]	0.982	0.932	891±18	1063±21	1052±29	82k
GoaT	0.985	0.939	1368±140		1549±208	35k
ParticleNet-Lite	0.984	0.937	1262±49			26k
ParticleNet	0.986	0.940	1615±93			366k
ParticleNeXt	0.987	0.942	1923±48			560k

G. Kasieczka et al. [*1902.09914*]

Quark/gluon tagging

	Acc	AUC	$1/\epsilon_B$ ($\epsilon_S = 0.5$)	$1/\epsilon_B$ ($\epsilon_S = 0.3$)
ResNeXt-50 [16]	0.821	0.9060	30.9	80.8
P-CNN [16]	0.827	0.9002	34.7	91.0
PFN [32]	-	0.9005	34.7±0.4	-
ParticleNet-Lite [16]	0.835	0.9079	37.1	94.5
ParticleNet [16]	0.840	0.9116	39.8±0.2	98.6±1.3
ABCNet [17]	0.840	0.9126	42.6±0.4	**118.4±1.5**
SPCT	0.824	0.899	34.4±0.4	100.3±1.5
PCT	**0.841**	**0.9140**	**43.3±0.7**	117.5±1.4
ParticleNeXt	0.841	0.9129	41±0.1	105±1.0

V. Mikuni, F. Canelli [*2102.05073*]

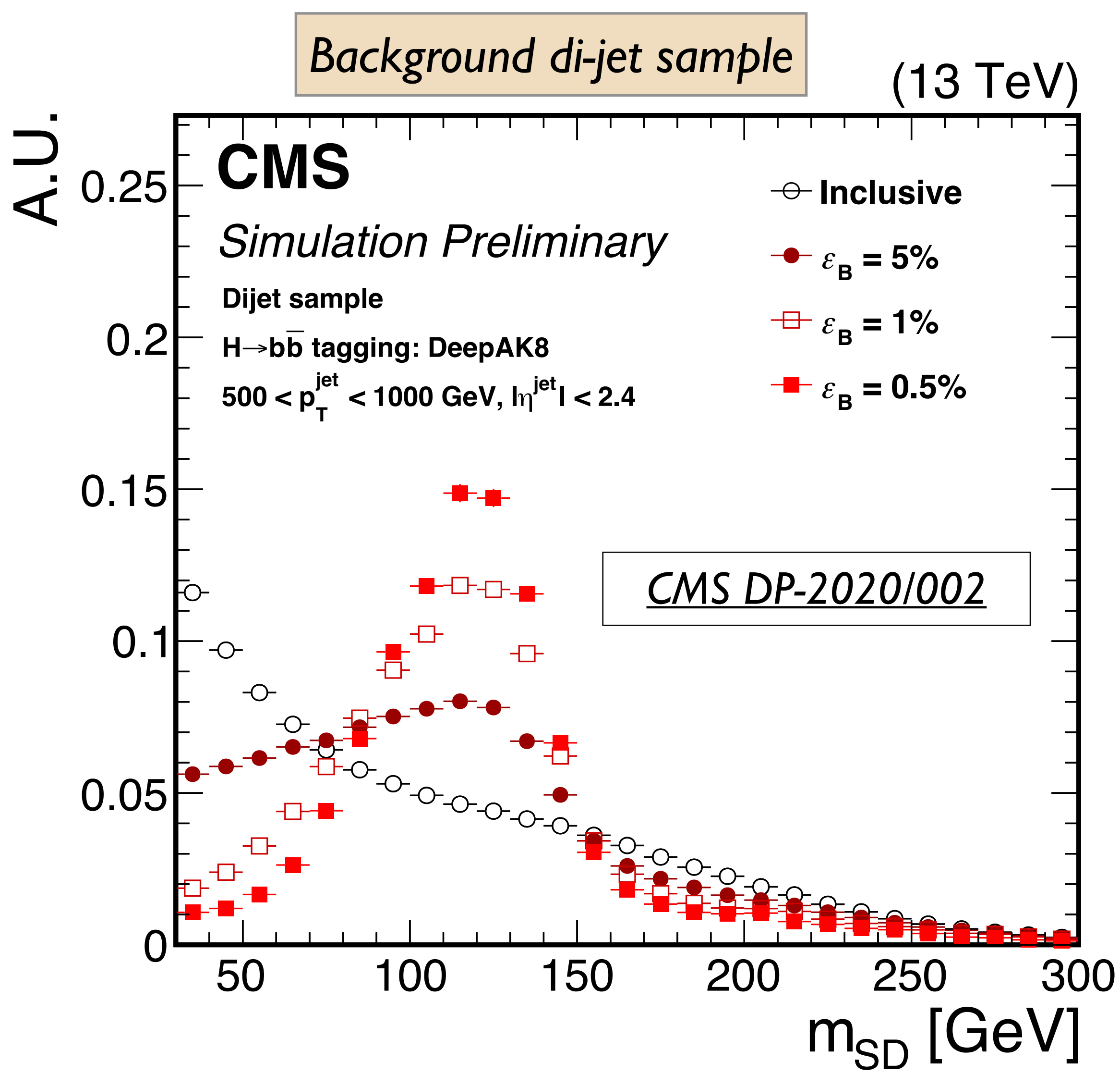
QUESTIONS?

Backups

Mass Decorrelation

Mass (De)Correlation

- One feature of these taggers is the correlation with the jet mass
  - jet mass shape of the background becomes similar to that of the signal after selection with the tagger: "**Mass sculpting**"
  - not necessarily a problem, but a mass-independent tagger is often more desirable:
    - if using the mass variable to separate signal and background
    - tagging signal jets with an unknown mass

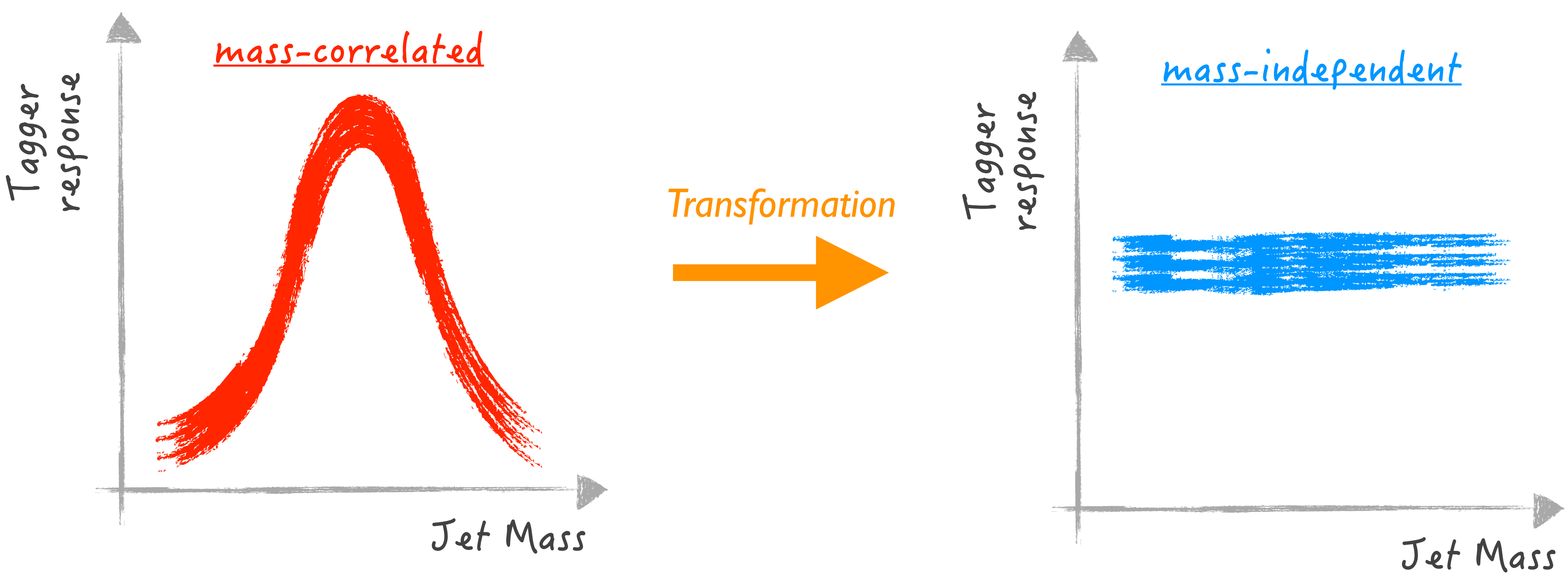


- *How to reduce the tagger's correlation with jet mass?*
- *More broadly: How to develop a classifier that is decorrelated with one or more auxiliary variables?*

METHOD 1: TRANSFORM TAGGER RESPONSE

- A tagger is mass-correlated because its response changes with the jet mass
  - a mass-independent tagger has a uniform response w.r.t the jet mass
- Mass decorrelation method 1: *the "brute-force" way*

Transforming the tagger response such that it no longer changes with the jet mass



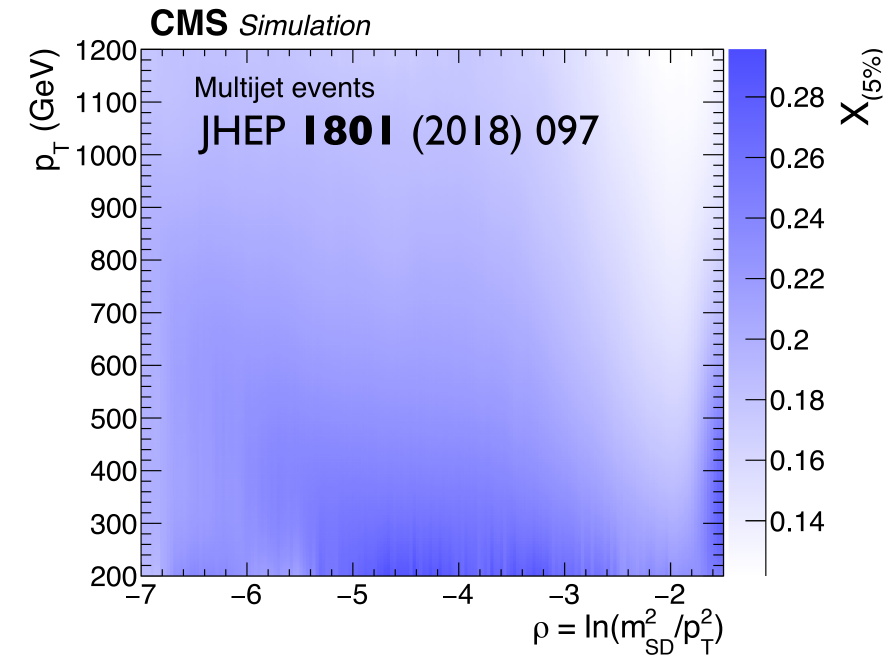
Designing Decorrelated Tagger (DDT)

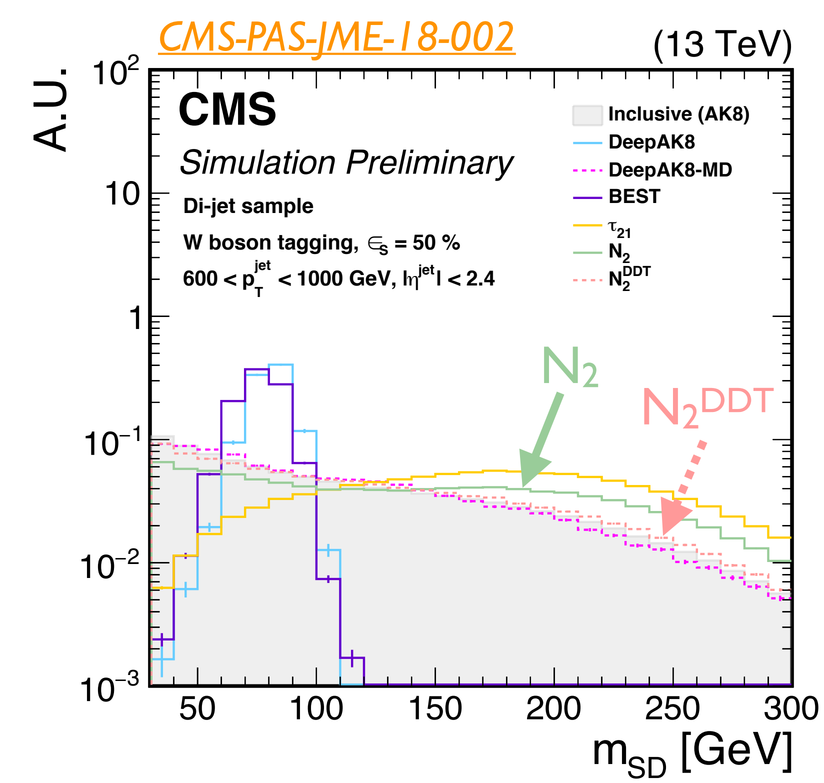
- Designing Decorrelated Tagger (DDT) [JHEP **1605** (2016) 156]
  - transforms the tagger response as a function of the jet p_T and $\rho = \ln(m_{SD}^2/p_T^2)$:

$$\mathrm{Tagger}^{DDT}(\rho,\, p_T) = \mathrm{Tagger}(\rho,\, p_T) - \mathrm{Tagger}^{(x\%)}(\rho,\, p_T)$$

    where $\mathrm{Tagger}^{(x\%)}(\rho,\, p_T)$ is the threshold for a background efficiency of x%, derived from simulated background (QCD) events
  - after the transformation, the selection $\mathrm{Tagger}^{DDT}>0$ (or <0) yields a constant background efficiency of x% across the m_{SD} and the p_T range
- N_2^{DDT}
  - N_2: generalized energy correlation functions [JHEP **1612** (2016 153] for 2-prong (W/Z/H) tagging
  - N_2^{DDT}: mass-decorrelated version of N_2 using the DDT method

DDT map: $N_2^{(5\%)}$





METHOD 2: MODIFY TRAINING PROCEDURE

- Mass decorrelation method 2: *the "active" way*

Modifying the training procedure/target
to prevent mass correlation

$L = L_{CE}$

Cross-Entropy loss

Modification →

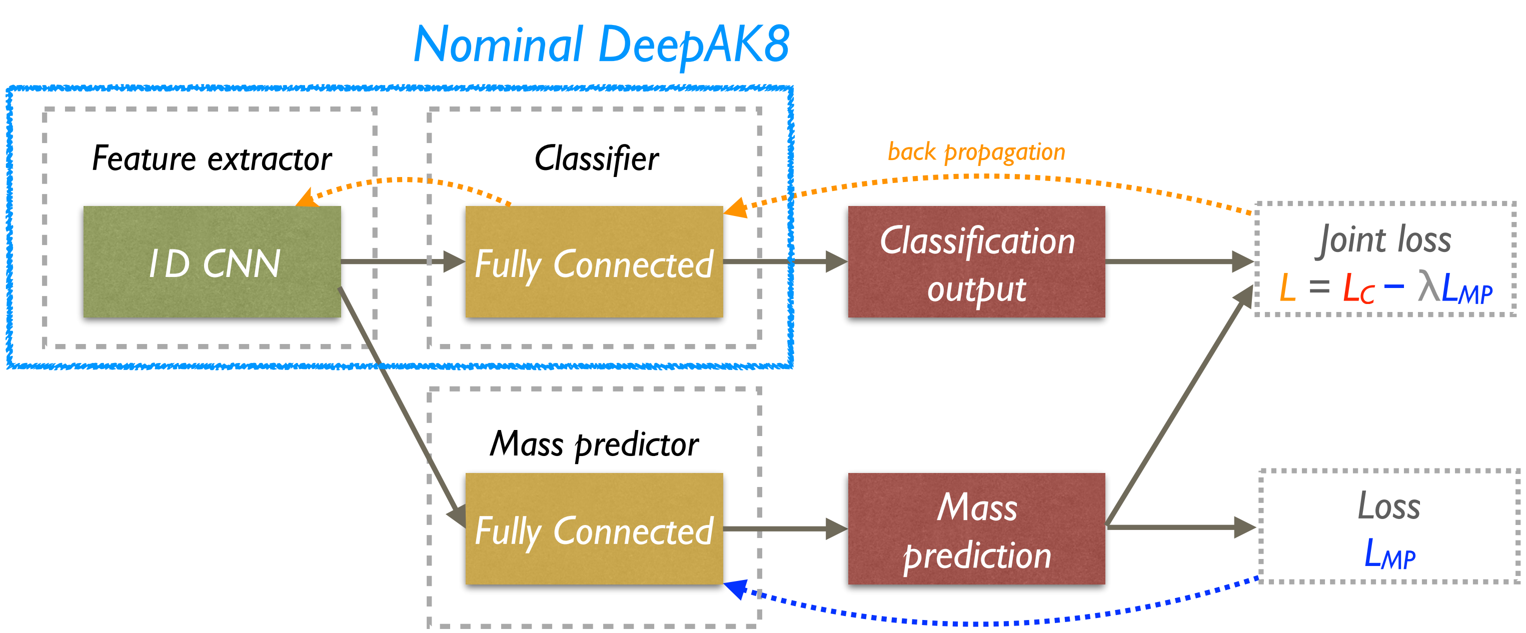
$L = L_{CE} + L_{MD}$

Cross-Entropy loss + Mass-decorrelation loss

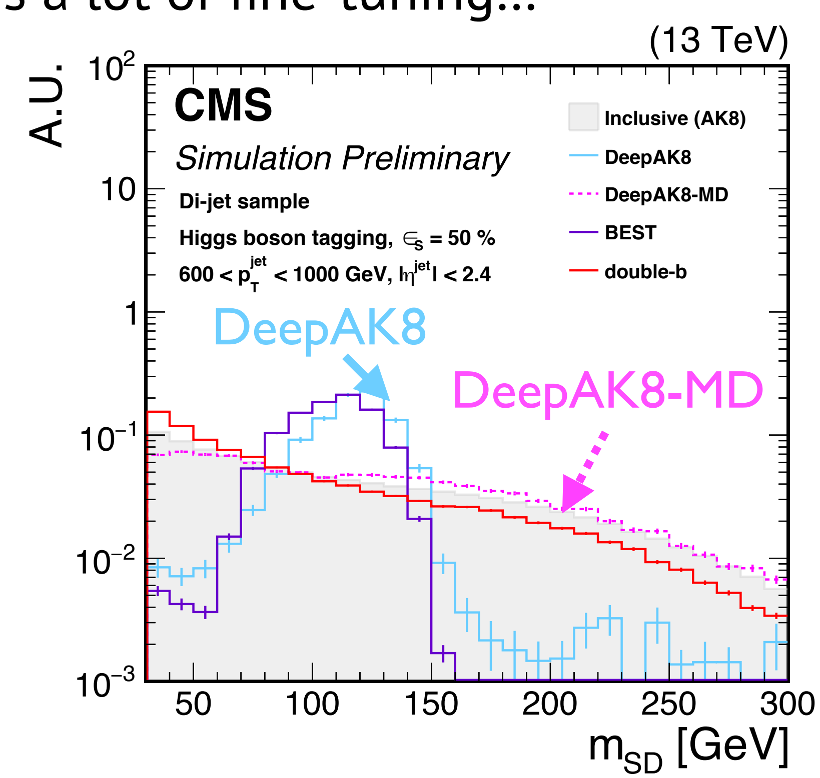
- Broadly speaking, this method involves choosing a differentiable metric to quantify the level of mass correlation and then minimize both the classification loss and this mass correlation metric
  - mass correlation can be measured with a number of metrics
    - KL divergence of the pass / fail mass shapes (e.g., CMS DeepDoubleB/C [CMS-DP-2018-046])
    - mutual information
    - a neural network – the GAN approach (e.g., CMS DeepAK8-MD)
    - distance correlation [Phys. Rev. Lett. 125, 122001]
    - …

DeepAK8-MD

- DeepAK8-MD: mass-decorrelation using adversarial training [1611.01046]
  - added a mass prediction network to predict the jet mass from the learned features
    - higher mass prediction accuracy -> stronger correlation w/ the jet mass
  - accuracy of the mass prediction included in the loss function as a penalty
    - minimizing the joint loss -> improving classification accuracy while preventing mass correlation
  - in addition: signal/background samples reweighted to a ~flat (p_T, m_{SD}) distribution to aid the training
- The adversarial training approach works reasonably well
  - significantly reduced mass sculpting while still strong performance
  - however the training process is quite challenging and requires a lot of fine-tuning…



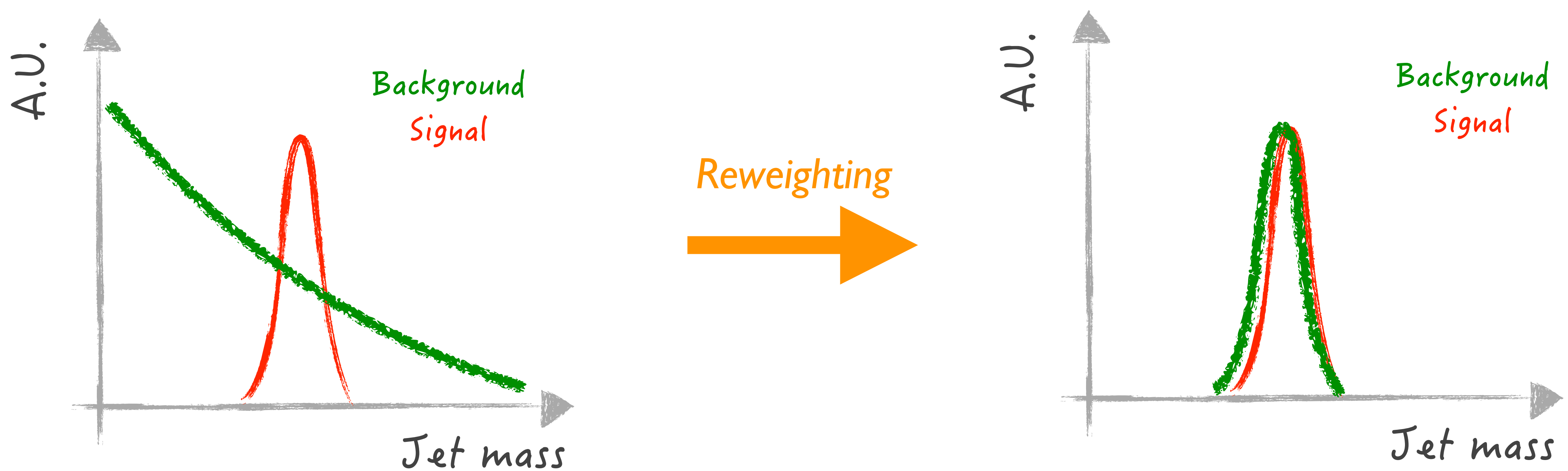
CMS-PAS-JME-18-002



METHOD 3: REWEIGHT TRAINING SAMPLES

- For ML taggers, mass correlation arises because signal (t/W/Z/H) and background (QCD) jets have very different mass distributions
  - maximizing signal/background separation inevitably causes the tagger responses to depend on the jet mass
  - if signal and background jets have similar mass distributions, then mass sculpting simply cannot happen
- Mass decorrelation method 3: *the "passive" way*

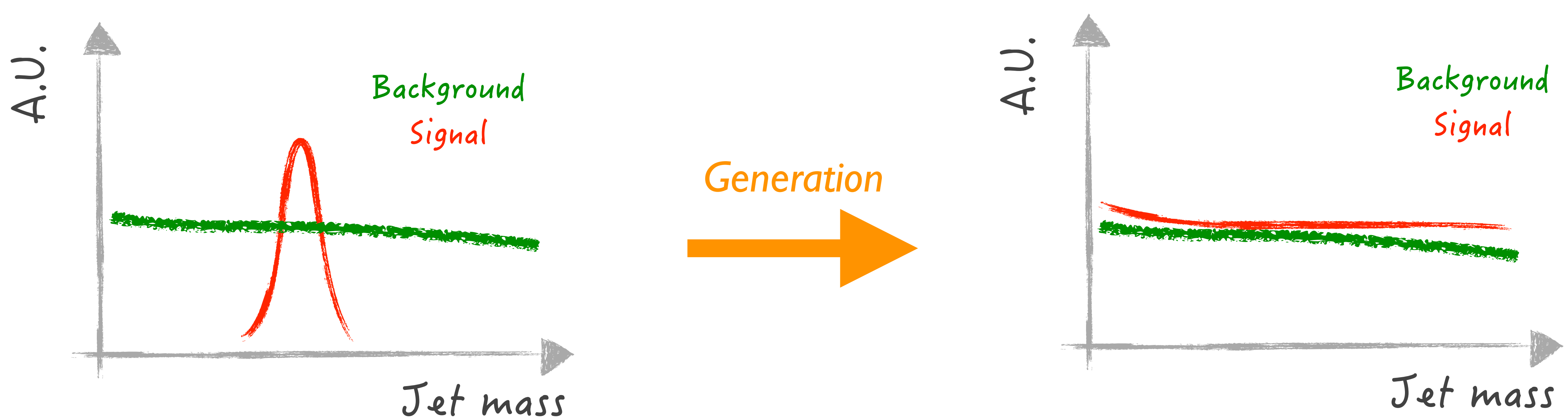
Reweighting the training samples such that signal and background jets have the same mass distributions



Method 4: Generate Special Training Samples

- The reweighting method works well for binary classification, but not sufficient for multi-class taggers
  - multiple signals, so cannot reweight the background mass shape to the signals
  - can possibly reweight everything to a flat / background-like mass distribution
    - but very low stats for signal away from the mass peak -> poor performance
- Instead of reweighting, can generate dedicated samples to populate the full mass range
- Mass decorrelation method 4: *the "actively-passive" way*

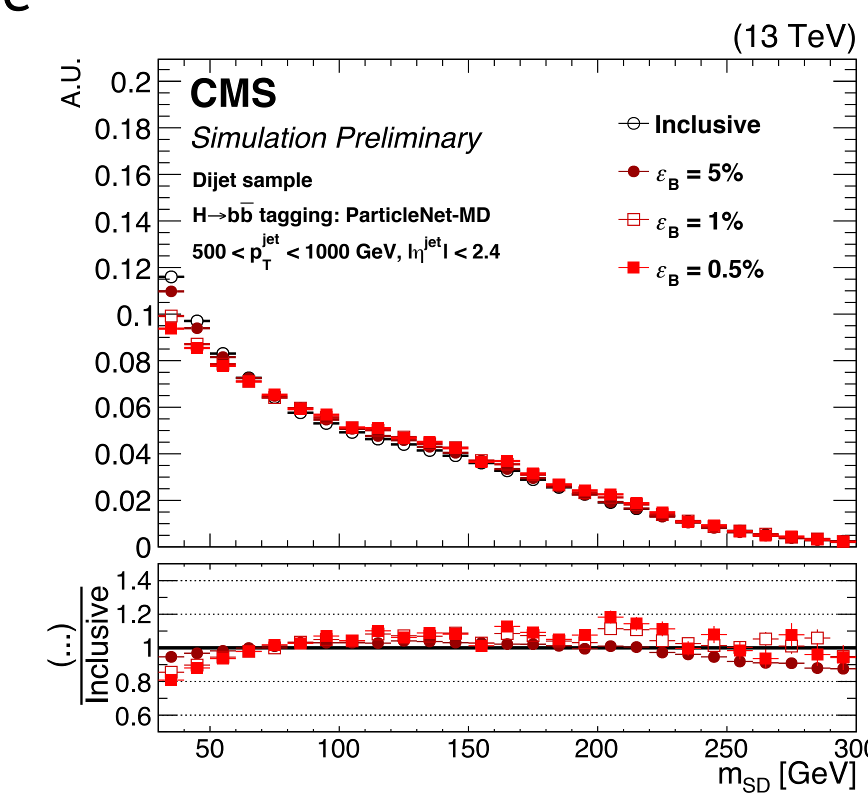
Generating a special training sample in which the signal particle has a flat mass distribution



ParticleNet-MD

- ParticleNet-MD [CMS DP-2020/002]
  - a generic mass-decorrelated 2-prong (W / Z / H / …) tagger
    - w/ also flavour information: i.e., X->bb, X->cc and X->qq
  - trained using a dedicated signal sample
    - hadronic decays of a spin-0 particle X: $X \to b\bar{b}, X \to c\bar{c}, X \to q\bar{q}$
    - flat mass spectrum: $m_X \in [15, 250]$ GeV
  - signal and background further reweighted to a flat [p_T, m_{SD}] distribution
  - using the ParticleNet graph neural network architecture

- Very good mass decorrelation with this approach
  - also very straightforward to train
    - no need to modify training procedure / loss



Tagger Calibration in Data

DeepAK8 in CMS

- Advanced deep learning-based algorithm for boosted object tagging, using AK8 (anti-k_T R=0.8) jets
  - multi-class classifier for top quark and W, Z, Higgs boson tagging
    - sub-classes based on decay modes (e.g., H→bb, H→cc, H→VV*→4q)
    - output scores can be aggregated/transformed for different tasks -> highly versatile tagger
  - directly uses jet constituents (particle-flow candidates / secondary vertices)
  - 1D convolutional neural network (CNN) based on the ResNet [arXiv: 1512.03385] architecture
  - significant performance improvement

Inputs

Substructure — Particles

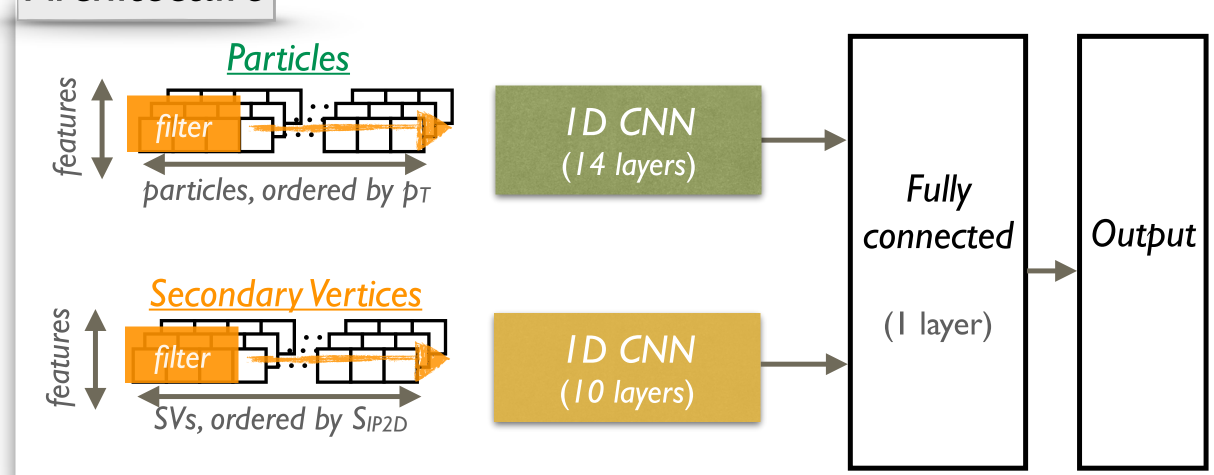
- Up to 100 PF candidates(*)
- Sorted in descending p_T order
- Uses basic kinematic variables, Puppi weights, and track properties (quality, covariance, displacement, etc.)

Flavour — Secondary vertices

- Up to 7 SVs(*) (inside jet cone)
- Sorted in descending S_{IP2D} order
- Uses SV kinematics and properties (quality, displacement, etc.)

(*) Number chosen to include all candidates for ≥ 90% of the events

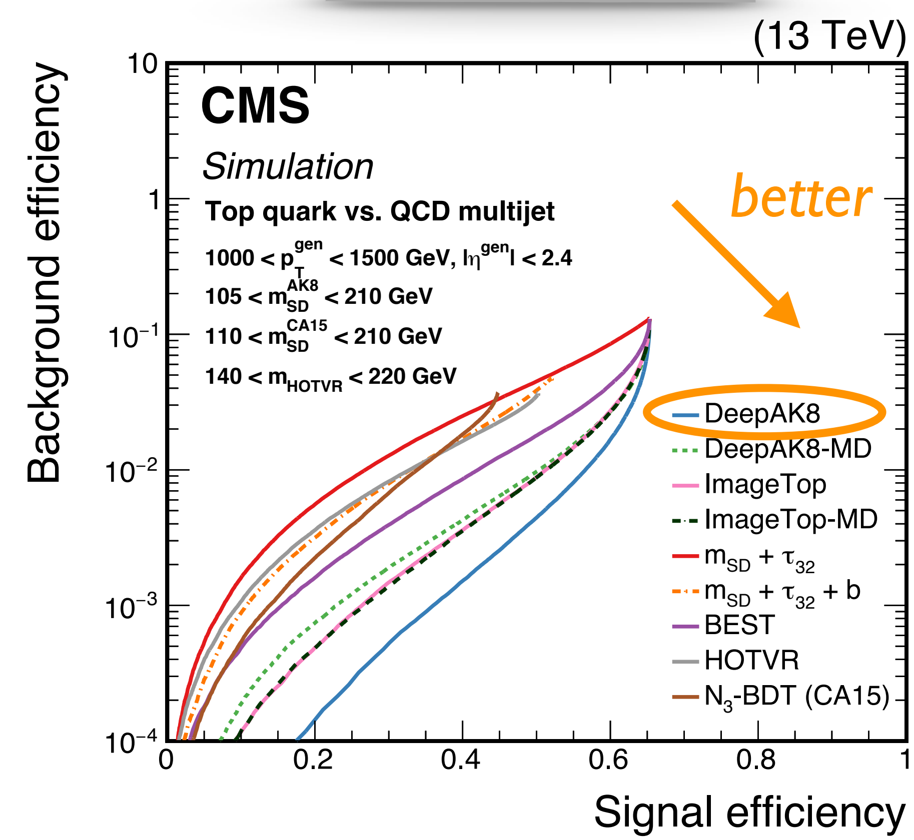
Architecture



Output

Category	Label
Higgs	H (bb)
	H (cc)
	H (VV*→qqqq)
Top	top (bcq)
	top (bqq)
	top (bc)
	top (bq)
W	W (cq)
	W (qq)
Z	Z (bb)
	Z (cc)
	Z (qq)
QCD	QCD (bb)
	QCD (cc)
	QCD (b)
	QCD (c)
	QCD (others)

Top quark tagging

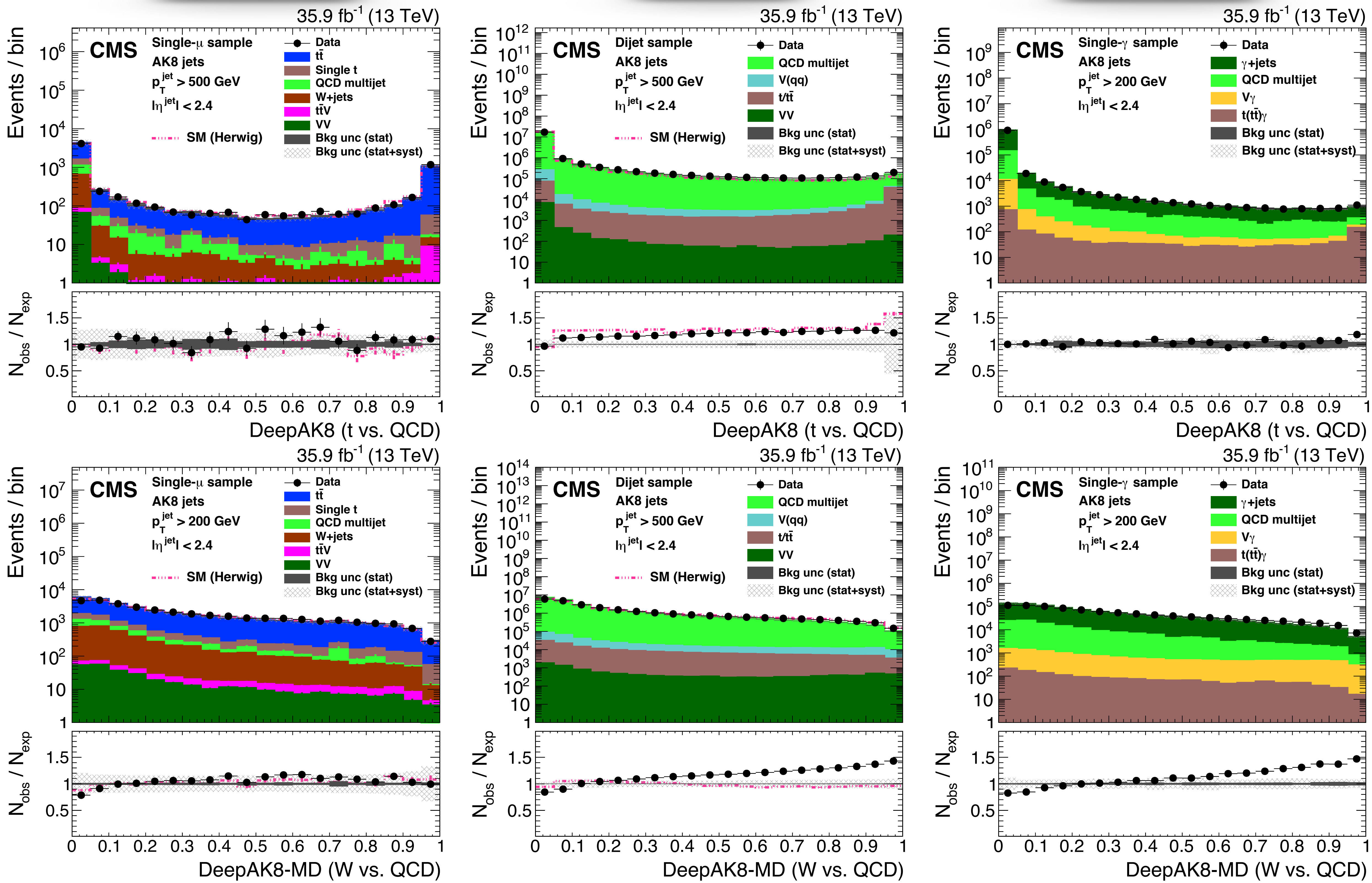


JINST 15 (2020) P06005

Performance in Data

CMS [*JINST 15 (2020) P06005*]

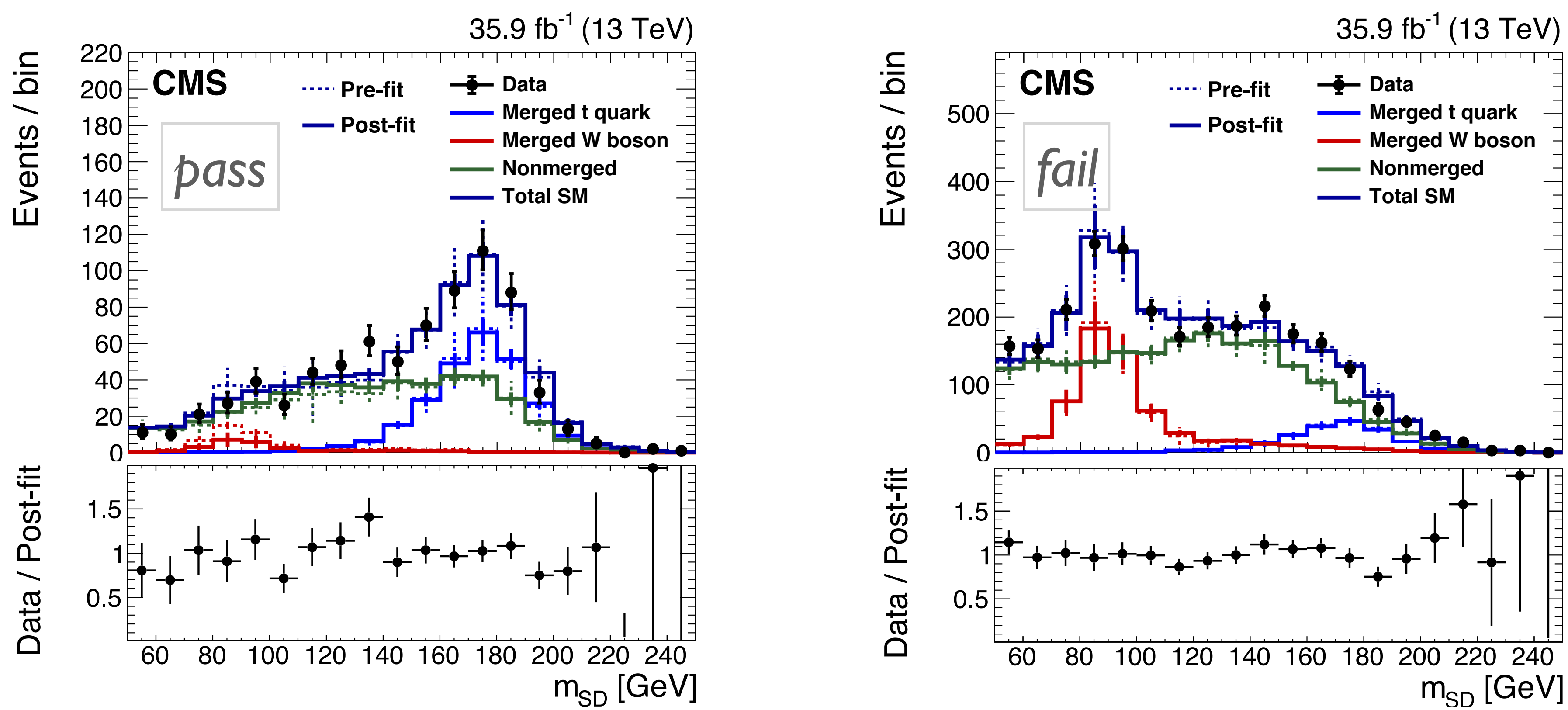
Single-μ sample | *Dijet sample* | *Single-γ sample*



Tagger Calibration in Data

- Crucial to calibrate these taggers in real data for them to be used in analyses
- Top/W tagging efficiency

JINST 15 (2020) P06005

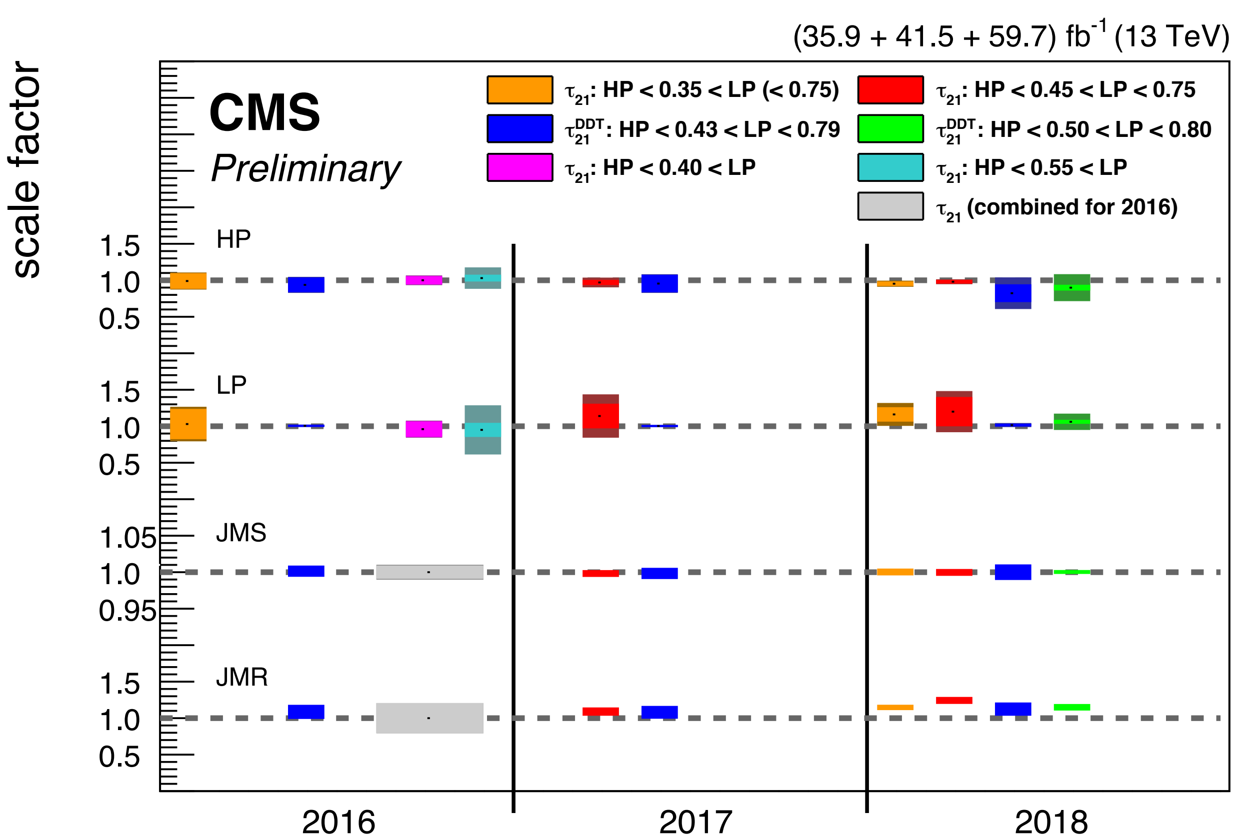


  - measured using the single-μ sample enriched in semi-leptonic ttbar events
  - fit jet mass templates in the "pass" and "fail" categories simultaneously to extract efficiency in data
    - simulation-to-data scale factors SF := eff(data) / eff(MC) derived to correct the simulation
  - jet mass scale and resolution scale factors can also be extracted
- H->bb/H->cc tagging efficiency: measured via proxy jets, gluon->bb/cc, using a di-jet sample
- Mistag rates of background jet typically derived directly from analysis-specific control regions

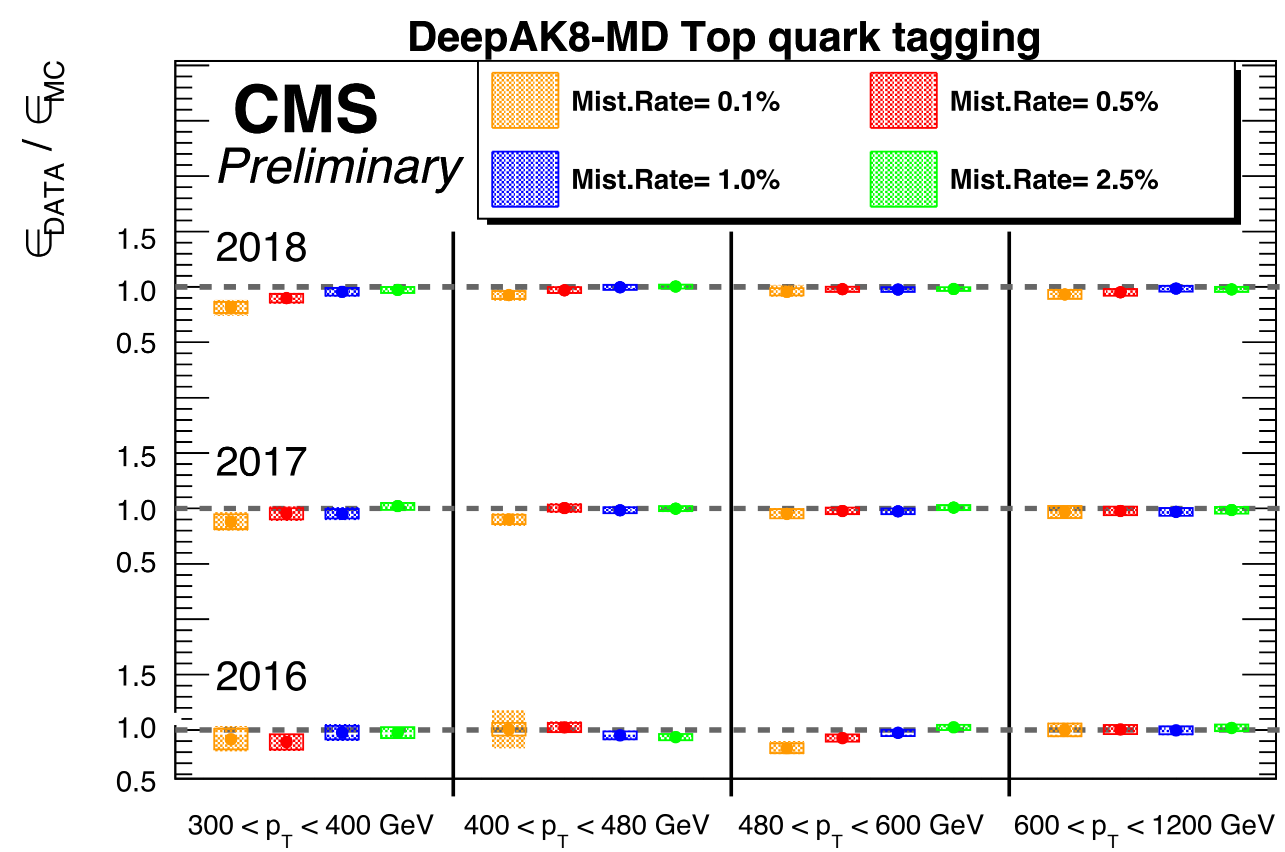
Tagger Calibration in Data (II)

CMS DP-2020/025

W tagging: τ_{21} / τ_{21}^{DDT}



Top tagging: DeepAK8-MD



- Simulation-to-data scale factors typically consistent with 1.0 within 10-20%