A Simple Tutorial on ParticleNet

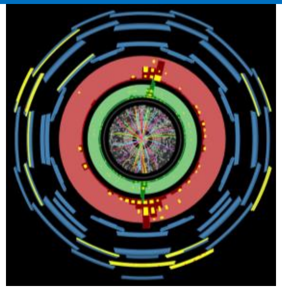
Shudong WANG

Experimental Physics Division

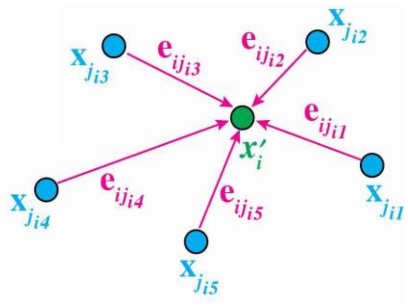
Institute of High Energy Physics, CAS

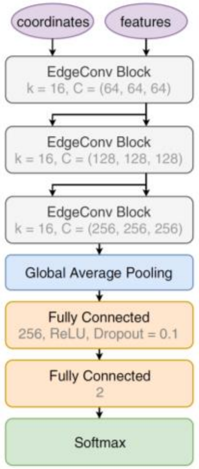
Outline

- **A little background knowledge-Jets**

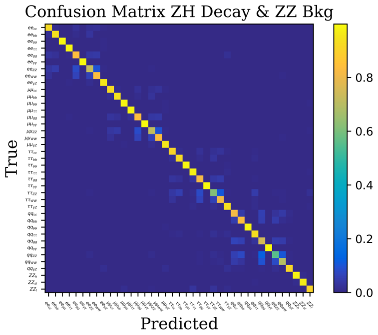


- **A Brief Introduction on ParticleNet**
- Jet/Event as a point cloud
- Point clouds VS Particle clouds
- The architecture of ParticleNet
- Performance of ParticleNet



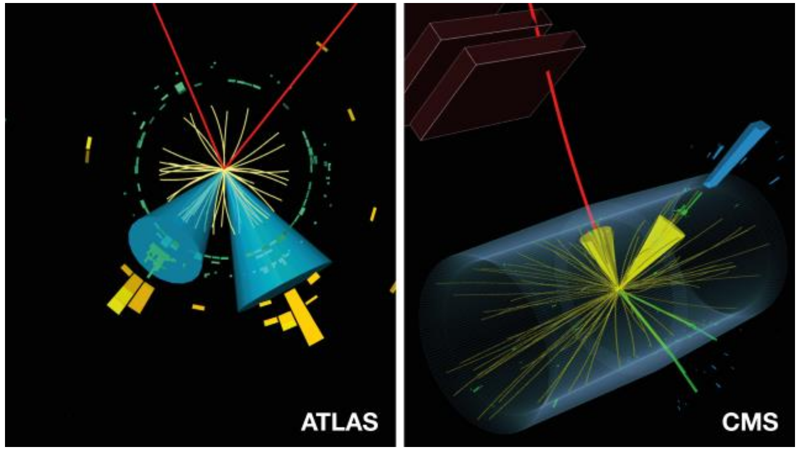


- **How to run ParticleNet**
- Try it yourself!

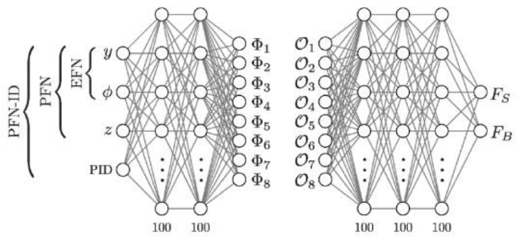


A little background knowledge-Jets

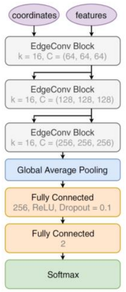
- Jets are ubiquitous at colliders, especially for hadron colliders.
- Jets are collimated sprays of particles initiated by quarks or gluons.



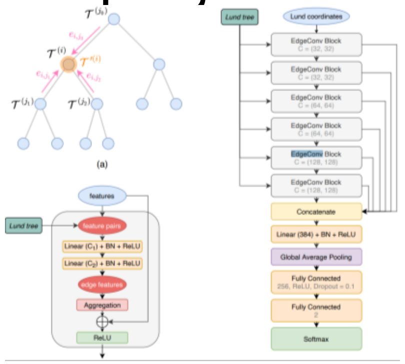
- Jet tagging: identifying the hard scattering particle that initiates the jet.
- The rise of machine learning (ML) has brought lots of new progresses to jet tagging.



JHEP01(2019)121



Phys.Rev.D 101 (2020) 5, 056019



JHEP03(2021)052

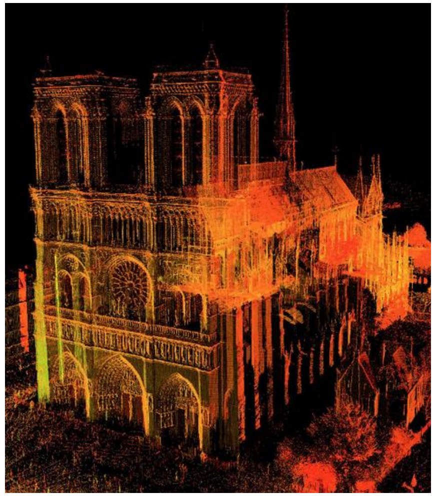
A Brief Introduction on ParticleNet

Based on Huilin Qu(the author of ParticleNet)'s report:

New approaches for jet tagging with machine learning (June 18, 2021) · Indico of IHEP (Indico)

A Brief Introduction on ParticleNet

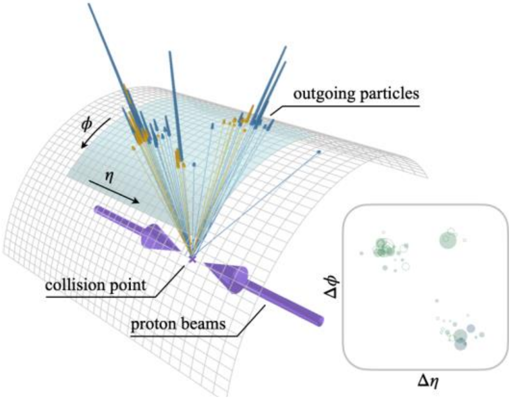
- **Jet/Event as a point cloud**



Point cloud

From Wikipedia, the free encyclopedia

A **point cloud** is a set of data points in space. The points may represent a 3D shape or object. Each point position has its set of Cartesian coordinates (X, Y, Z).[1] Point clouds are generally produced by 3D scanners or by photogrammetry software, which measure many points on the external surfaces of objects around them. As the output of 3D



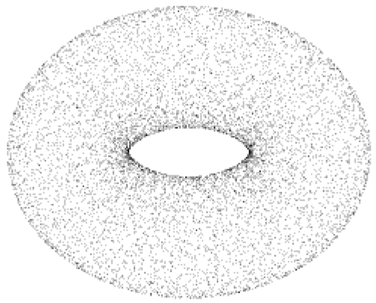
Jet (particle physics)

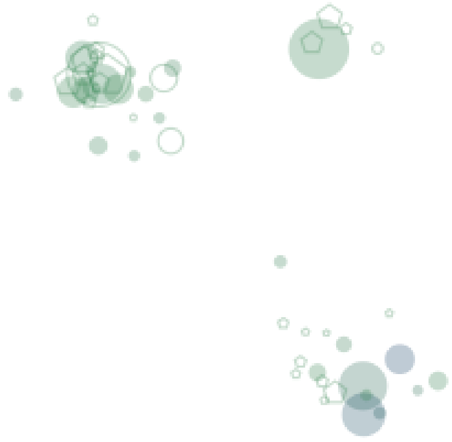
From Wikipedia, the free encyclopedia

A **jet** is a narrow cone of hadrons and other particles produced by the hadronization of a quark or gluon in a particle physics or heavy ion experiment. Particles

A Brief Introduction on ParticleNet

- **Jet/Event as a point cloud**





Point cloud

From Wikipedia, the free encyclopedia

A **point cloud** is a set of data points in space. The points may represent a 3D shape or object. Each point position has its set of Cartesian coordinates (X, Y, Z).[1] Point clouds are generally produced by 3D scanners or by photogrammetry software, which measure many points on the external surfaces of objects around them. As the output of 3D

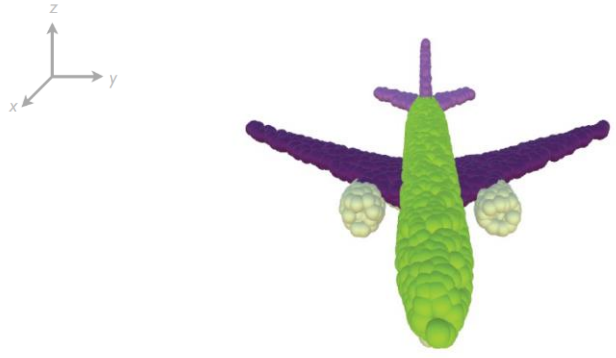
Jet (Particle cloud)

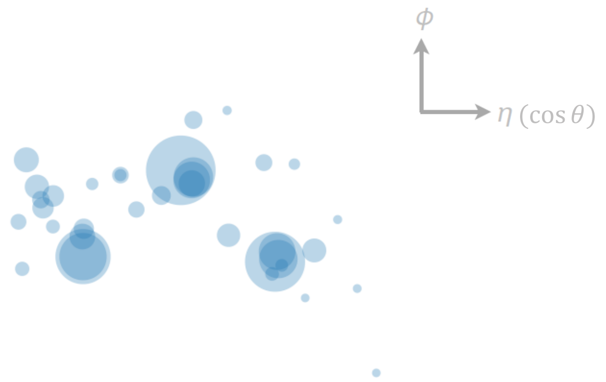
From Wikipedia, the free encyclopedia

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 and CMS. which measure

A Brief Introduction on ParticleNet

- **Point clouds VS Particle clouds**





Point cloud

- points are intrinsically ***unordered***
- points are distributed in space
  - spatial coordinates (3D xyz) encode geometric structure information

Particle cloud

- particles are intrinsically ***unordered***
- particles are distributed in space
  - spatial distribution (2D coordinates in the η(cos θ)-φ space) reflects radiation patterns

But particles have more features:
- energy/momenta/displacement/particle ID/etc.
- more interesting than a plain point cloud!

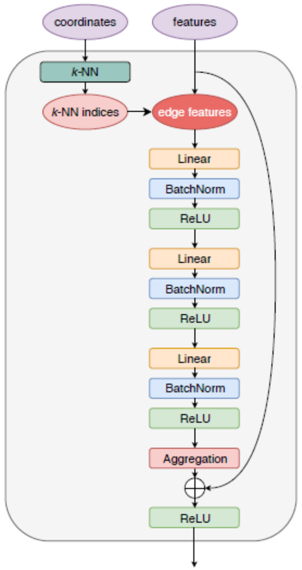
A Brief Introduction on ParticleNet

- **The architecture of ParticleNet** H. Qu and L. Gouskos [*Phys.Rev.D 101 (2020) 5, 056019*]

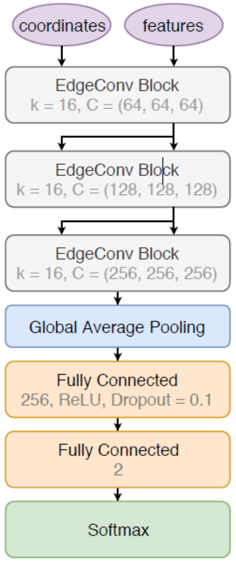
ParticleNet

- customized graph neural network architecture for jet tagging with the point cloud approach, based on Dynamic Graph CNN (DGCNN)
  [Y. Wang et al., arXiv:1801.07829]
- explicitly respects the permutation symmetry of the point cloud

EdgeConv block



ParticleNet architecture



Key building block: EdgeConv

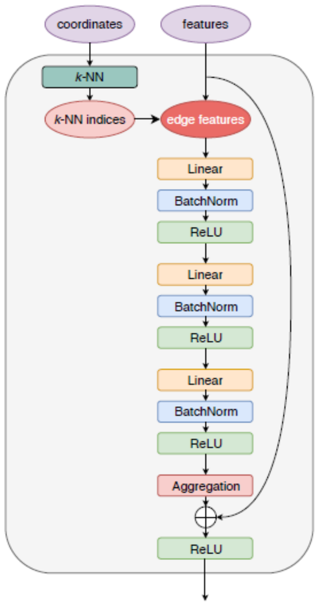
- treating a point cloud as a graph: each point is a vertex
  - for each point, a local patch is defined by finding its k-nearest neighbors
- designing a permutation-invariant "convolution" function
  - define "edge feature" for each center-neighbor pair: $e_{ij} = h_\Theta\left(x_i,\ x_{i_j}\right) = \bar{h}_\Theta\left(x_i,\ x_{i_j} - x_i\right)$
    - same h_Θ for all neighbor points, and all center points, for symmetry
  - aggregate the edge features in a symmetric way: $x_i' = \square_{j=1}^{k} h_\Theta\left(x_i,\ x_{i_j}\right) = \frac{1}{k}\sum h_\Theta\left(x_i,\ x_{i_j}\right)$

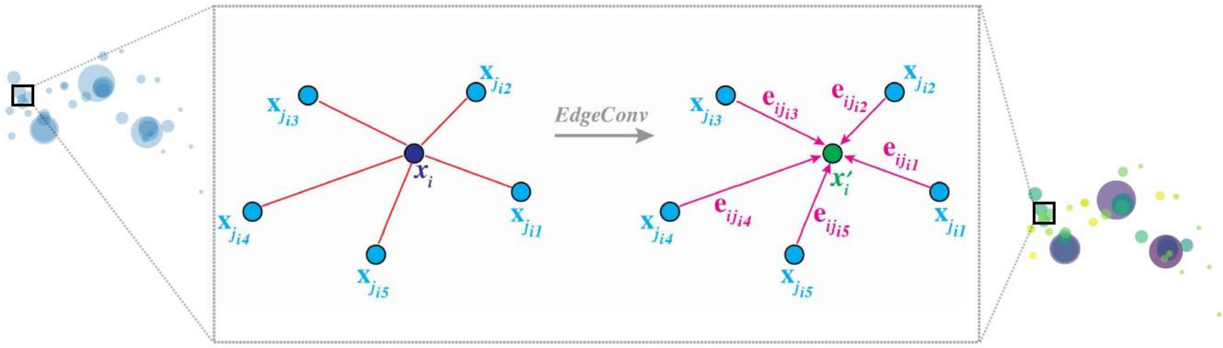
A Brief Introduction on ParticleNet

- **The architecture of ParticleNet** H. Qu and L. Gouskos [*Phys.Rev.D 101 (2020) 5, 056019*]

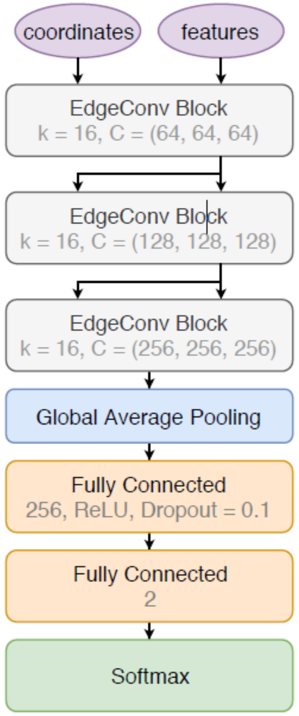
- EdgeConv can be stacked to form a deep network
  - learning both local and global structures, in a hierarchical way

EdgeConv block





ParticleNet architecture



A Brief Introduction on ParticleNet

• Performance of ParticleNet

- Performance comparison on the top tagging benchmark dataset.

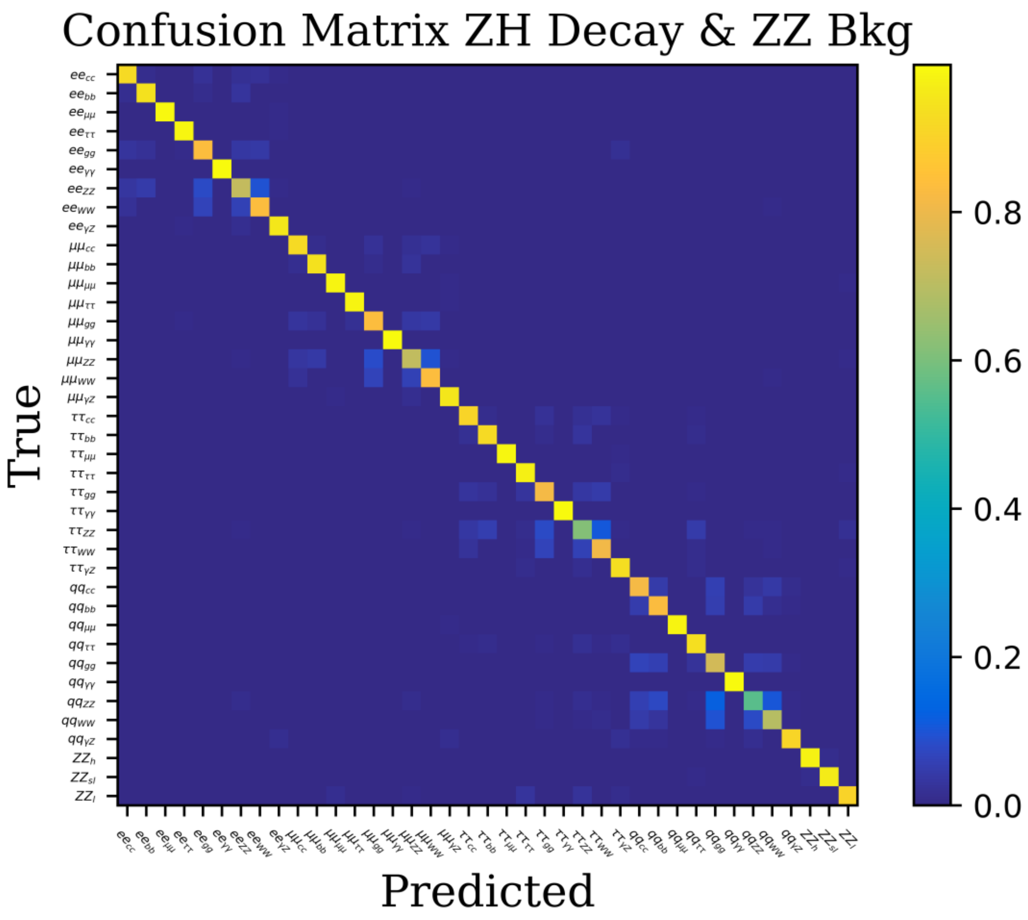
	Accuracy	AUC	$1/\varepsilon_b$ at $\varepsilon_s = 50\%$	$1/\varepsilon_b$ at $\varepsilon_s = 30\%$
ResNeXt-50	0.936	0.9837	302 ± 5	1147 ± 58
P-CNN	0.930	0.9803	201 ± 4	759 ± 24
PFN	...	0.9819	247 ± 3	888 ± 17
ParticleNet-Lite	0.937	0.9844	325 ± 5	1262 ± 49
ParticleNet	**0.940**	**0.9858**	$\mathbf{397 \pm 7}$	$\mathbf{1615 \pm 93}$

- Performance comparison on the quark-gluon tagging benchmark dataset.

	Accuracy	AUC	$1/\varepsilon_b$ at $\varepsilon_s = 50\%$	$1/\varepsilon_b$ at $\varepsilon_s = 30\%$
ResNeXt-50	0.821	0.8960	30.9	80.8
P-CNN	0.818	0.8915	31.0	82.3
PFN	...	0.8911	30.8 ± 0.4	...
ParticleNet-Lite	0.826	0.8993	32.8	84.6
ParticleNet	0.828	0.9014	33.7	85.4
P-CNN (w/ PID)	0.827	0.9002	34.7	91.0
PFN-Ex (w/ PID)	...	0.9005	34.7 ± 0.4	...
ParticleNet-Lite (w/ PID)	0.835	0.9079	37.1	94.5
ParticleNet (w/ PID)	**0.840**	**0.9116**	$\mathbf{39.8 \pm 0.2}$	$\mathbf{98.6 \pm 1.3}$

A Brief Introduction on ParticleNet

- **Performance of ParticleNet**
  - Classify Higgs decays on CEPC (Event Level).



How to run ParticleNet

- First thing to do:
  Login to IHEP cluster and do:

```
cp -r /scratchfs/bes/wangshudong/ParNet_tuto/ /PATH/TO/YOUR/SPACE/ParNet_tuto/
```

How to run ParticleNet

- **Weaver**
    - *Weaver* aims at providing a streamlined yet flexible machine learning R&D framework for high energy physics (HEP) applications. Github-Repo (old ver.)
- **Set up your environment (you can use mine)**
    - Install Miniconda (if you don't already have it)
    - Set up a conda environment and install the required packages
    - On IHEP cluster, simply type commands below to use my conda env (you don't even need to do this, see page 15):

```
#this conda env only support training using CPU, since most of you
#don’t have access to GPU cluster
source "/cefs/higgs/wangshudong/miniconda3/etc/profile.d/conda.sh"
conda activate weaver
```

- **Prepare your configuration files**

  To train a neural network using *Weaver*, you need to prepare:
    - 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.
    - ***Let's move to codes now***

How to run ParticleNet

- **Start running! (general case)**
  - The train.py script is the top-level script to run for training a neural net, getting prediction from trained models, and exporting trained models to ONNX for production.
  - To check all the command-line options for *train.py*, run *python train.py -h*. Examples for training, inference and model exportation are shown below:

- **Training**

```
python train.py --data-train '/path/to/train_files/*/*/*/*/output_*.root’ \
--data-test '/path/to/train_files/*/*/*/*/output_*.root' \
--data-config data/ak15_points_pf_sv.yaml \
--network-config networks/particle_net_pf_sv.py \
--model-prefix /path/to/models/prefix \
--gpus '0,1,2,3' --batch-size 512 --start-lr 5e-3 --num-epochs 20 --optimizer ranger \
--log logs/train.log
```

How to run ParticleNet

- **Prediction/Inference**

```
python train.py --predict --data-test '/path/to/test_files/*/*/*/*/output_*.root' \
 --data-config data/ak15_points_pf_sv.yaml \
 --network-config networks/particle_net_pf_sv.py \
 --model-prefix /path/to/models/prefix_best_epoch_state.pt \
 --gpus '0,1,2,3' --batch-size 512 \
 --predict-output /path/to/output.root
```

How to run ParticleNet

- Start running! (for this tutorial only)

```
cd /PATH/TO/YOUR/SPACE/ParNet_tuto/weaver
source subjob.sh #submit to cluster
OR
source weaver_e2e2h_9cls_cpu.sh #run on login node
```

Then just wait!

How to run ParticleNet

- **Dataset**

  - The dataset prepared for today's tutorial:

    The production mode $\mu\mu H$ for the Higgs boson at 240 GeV are studied here, 9 decay modes are used, which are $H \to c\bar{c}, H \to b\bar{b}, H \to \mu^+\mu^-, H \to \tau^+\tau^-$, $H \to gg, H \to \gamma\gamma, H \to ZZ^*, H \to WW^*$ and $H \to \gamma Z$, respectively.

    For each process, 100 events are generated with WHIZARD 1.9.5 and fed to PYTHIA6 for hadronization, where decays of most intermediate particles, such as W, Z, τ, etc., are also simulated by PYTHIA6 according to its default configuration.

    All the generated samples are simulated in a simplified way to model detector responses. In detail, all particles are simulated according to the performance of the baseline detector in the CEPC CDR

How to run ParticleNet

- **Dataset**
  - The dataset prepared for today's tutorial:

    All the generated samples are simulated in a simplified way to model detector responses. In detail, all particles are simulated according to the performance of the baseline detector in the CEPC CDR:

    The momentum resolution of charged tracks is: $\frac{\sigma(p_t)}{p_t} = 2\times10^{-5} \oplus \frac{0.001}{p\sin^{3/2}\theta}[\mathrm{GeV}^{-1}]$

    The energy resolution of photons is: $\frac{\sigma(E)}{E} = 0.03 \oplus \frac{0.50}{\sqrt{E/(\mathrm{GeV})}}$

    The energy resolution of neural hadrons is: $\frac{\sigma(E)}{E} = 0.01 \oplus \frac{0.20}{\sqrt{E/(\mathrm{GeV})}}$

    All the reconstruction efficiencies are assumed to be 100% in the simulation.
    For impact parameters and particle identification (PID, only e, μ, γ), they are taken directly from the truth of generation.

How to run ParticleNet

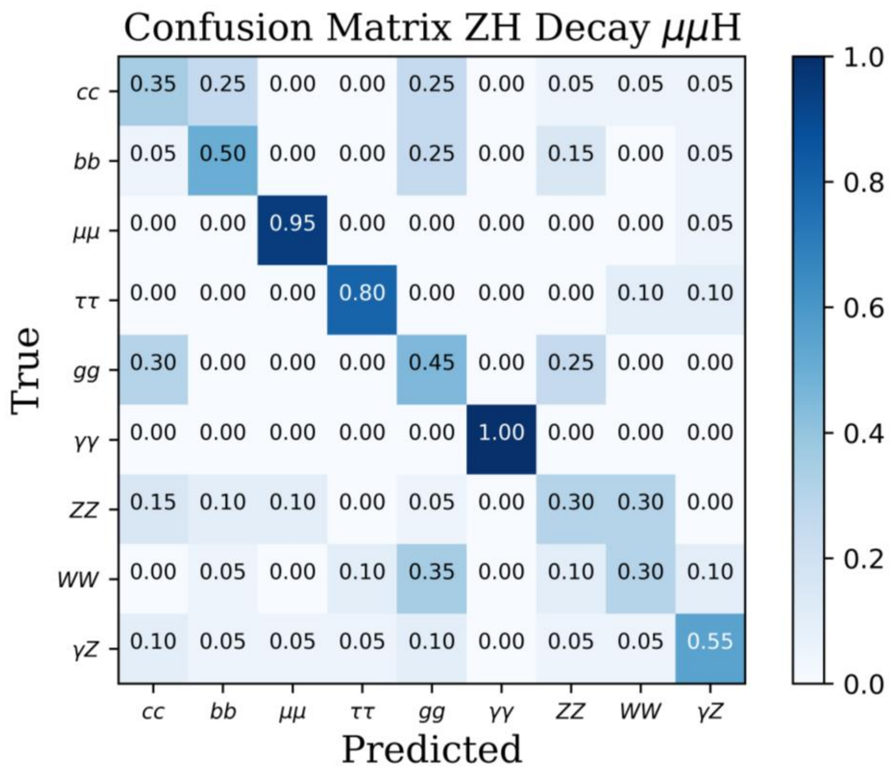
- **How to plot a confusion matrix**

  - confusion matrix:

```
#in weaver conda environment
cd /PATH/TO/YOUR/SPACE/ParNet_tuto/weaver/pltCM
#copy confusion matrix from
#test_e2e2h_9cls_xxxxxxxx_xxxxxx.log and paste it
#in pltCM.py and add some commas
python plotCM.py
```

```
CM = np.array( [[0.35, 0.25, 0.  , 0.  , 0.25, 0.  , 0.05, 0.05, 0.05],
                [0.05, 0.5 , 0.  , 0.  , 0.25, 0.  , 0.15, 0.  , 0.05],
                [0.  , 0.  , 0.95, 0.  , 0.  , 0.  , 0.  , 0.  , 0.05],
                [0.  , 0.  , 0.  , 0.8 , 0.  , 0.  , 0.  , 0.1 , 0.1 ],
                [0.3 , 0.  , 0.  , 0.  , 0.45, 0.  , 0.25, 0.  , 0.  ],
                [0.  , 0.  , 0.  , 0.  , 0.  , 1.  , 0.  , 0.  , 0.  ],
                [0.15, 0.1 , 0.1 , 0.  , 0.05, 0.  , 0.3 , 0.3 , 0.  ],
                [0.  , 0.05, 0.  , 0.1 , 0.35, 0.  , 0.1 , 0.3 , 0.1 ],
                [0.1 , 0.05, 0.05, 0.05, 0.1 , 0.  , 0.05, 0.05, 0.55]])
```

  - Result:



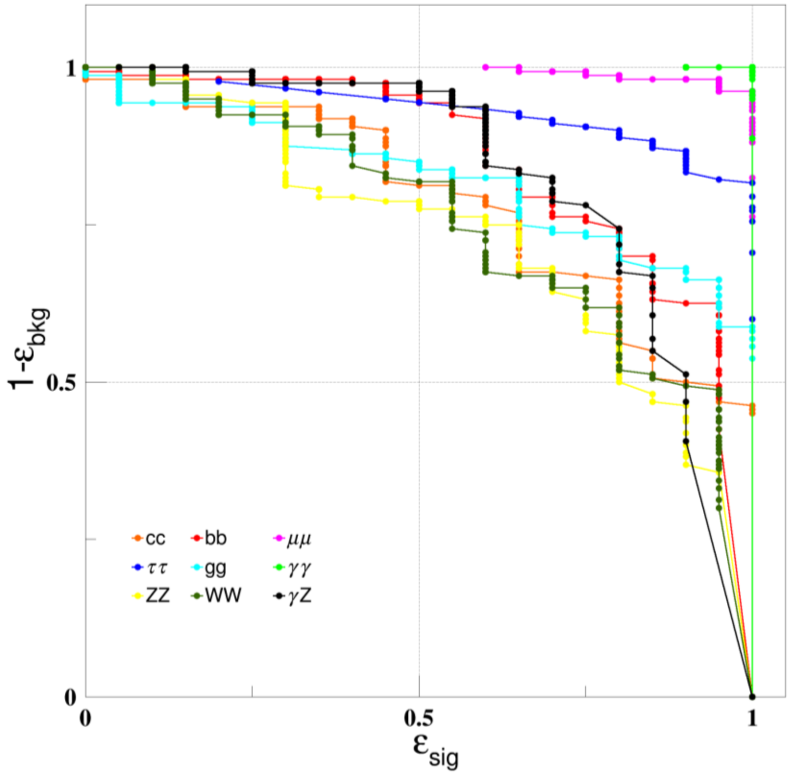
How to run ParticleNet

- **How to plot ROC curves**

  - ROC curves:

```
#deactivate conda environment
cd /PATH/TO/YOUR/SPACE/ParNet_tuto/weaver/pltROC
vim plotROC.C #change the path in line 11 to your
#output root file
#use default root environment
root plotROC.C
```

  - Result:



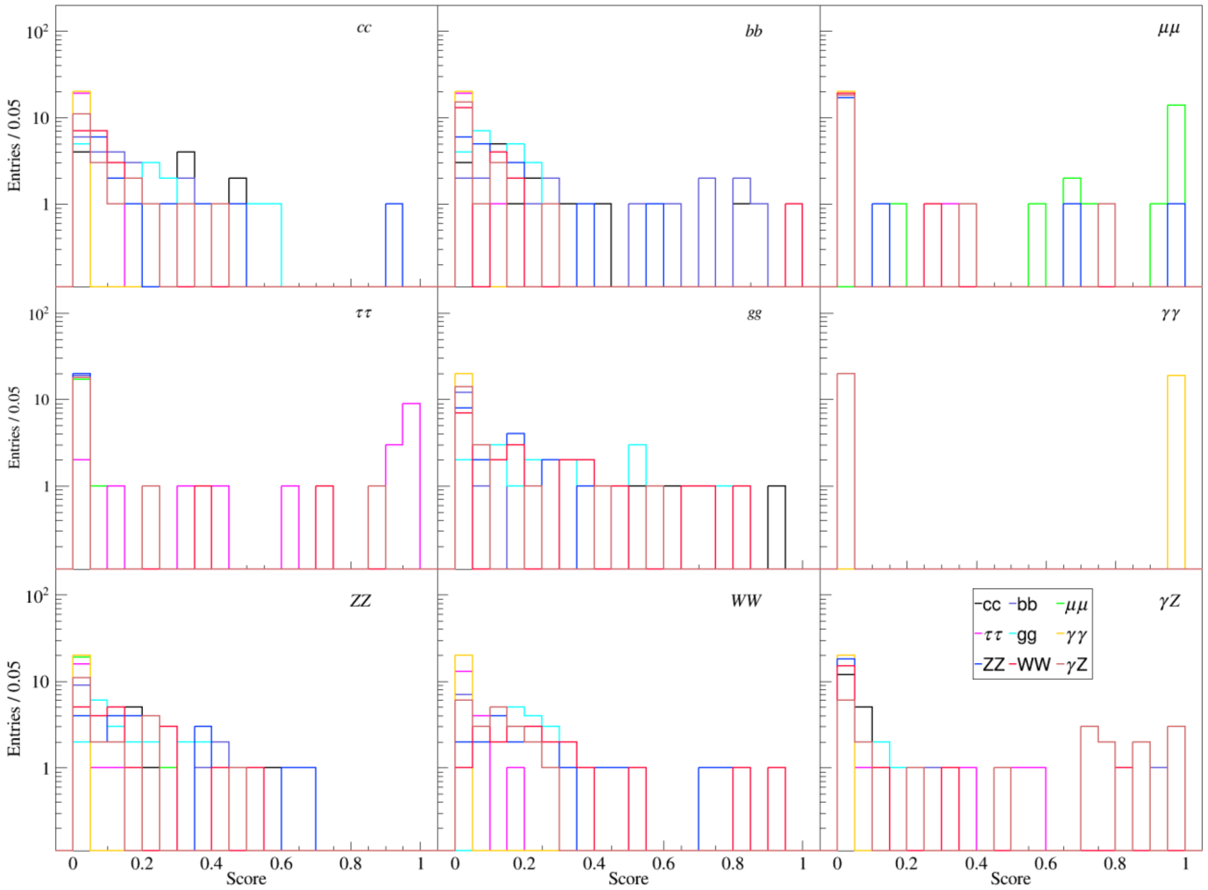
How to run ParticleNet

- **How to plot scores**

  - ROC curves:

```
#deactivate conda environment
cd /PATH/TO/YOUR/SPACE/ParNet_tuto/weaver/pltScore
vim plotScore.C
#change the path in line 125 to your output root
#file
#use default root environment
root plotScore.C
```

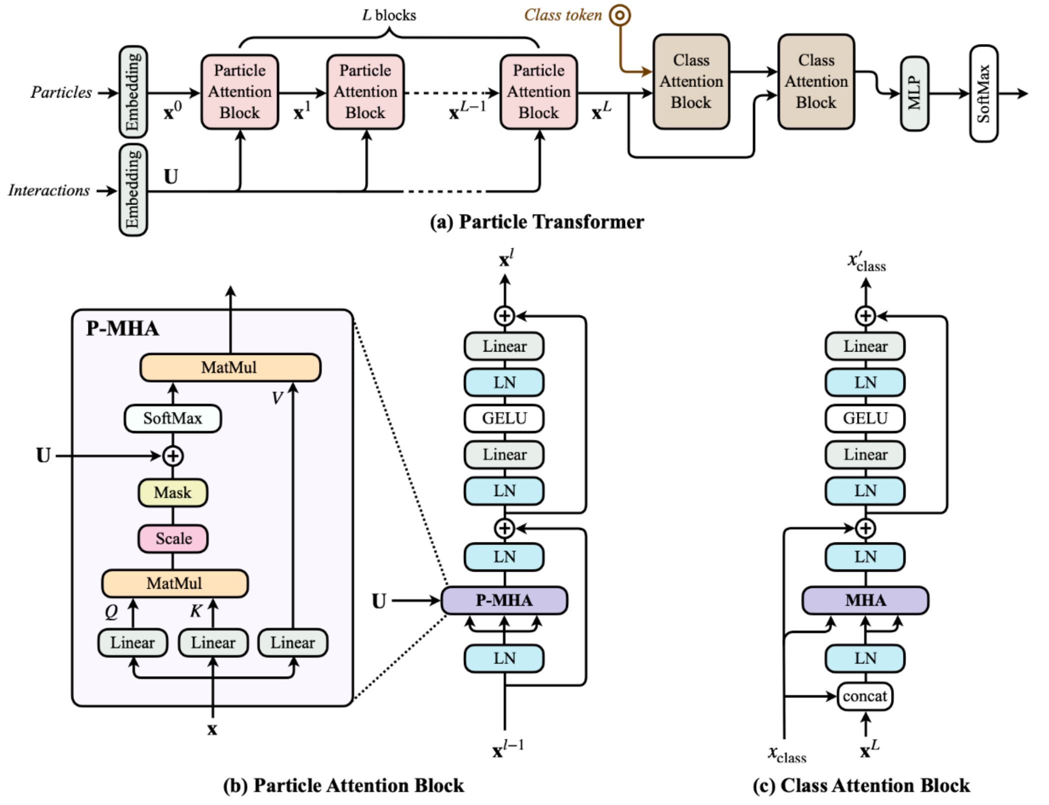
  - Result:



New approach: Particle Transformer

Particle Transformer (ParT)

The **Particle Transformer (ParT)** architecture is described in "Particle Transformer for Jet Tagging", which can serve as a general-purpose backbone for jet tagging and similar tasks in particle physics. It is a Transformer-based architecture, enhanced with pairwise particle interaction features that are incorporated in the multi-head attention as a bias before softmax. The ParT architecture outperforms the previous state-of-the-art, ParticleNet, by a large margin on various jet tagging benchmarks.



(a) Particle Transformer
(b) Particle Attention Block
(c) Class Attention Block

Github Link:
[Official implementation of "Particle Transformer for Jet Tagging". (github.com)]

Indico Link (EPD seminar):
[Jet tagging algorithm respecting Lorentz group symmetry (September 2, 2022) · Indico of IHEP (Indico)]

ParticleNet, PFN, P-CNN are also implemented in this package using ***weaver*** (Github: hqucms/weaver-core) framework

Try new toys!