Machine-Learned Particle Flow

Performance update & exploration of explainable AI techniques

IRIS-HEP

2021/9/20

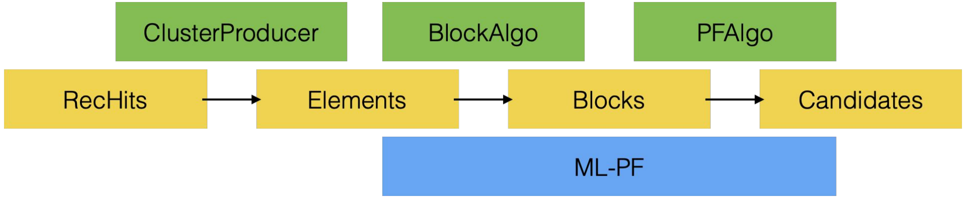
Farouk Mokhtar, UCSD, Duarte Lab

Mentor: Javier Duarte

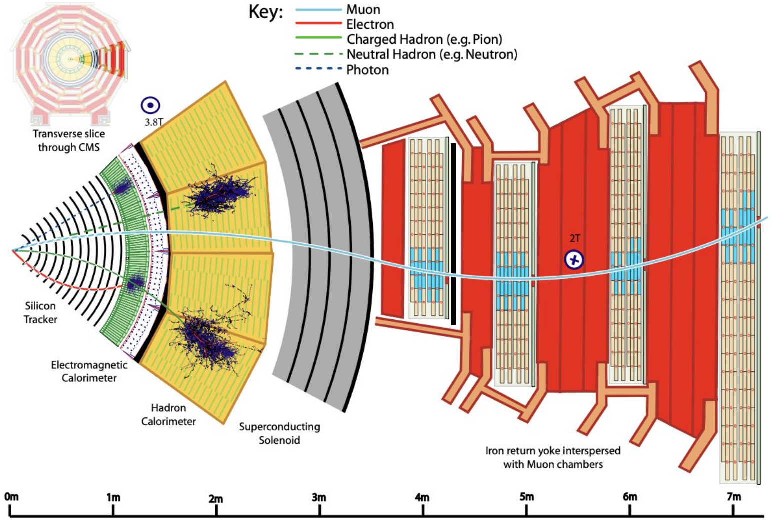
Additional mentors: Joosep Pata, Jean-Roch Vlimant, Raghav Kansal

Overview

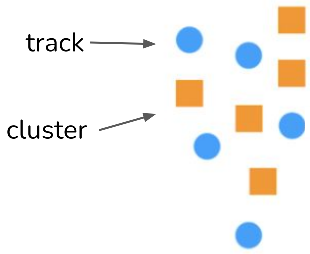
- **PF**: Global event reconstruction; combining information from calorimeter clusters and tracks to reconstruct stable particles
- **MLPF**: an evolution of the rule-based PF algorithm for heterogeneous computing platforms such as GPUs using **supervised machine learning with graph neural networks**
- **Our input is PF-Elements**: calorimeter clusters and tracks



Aim to speed up the parts of PF reconstruction that have not already been ported to GPU!



Event as input
$X = \{x_i\}$



ML model on the Delphes benchmark dataset

Input:

- Each **event** is represented by a **graph** (~5k nodes for ttbar+PU50)
- Each **node** in the graph is a **detector element** (12d-feature vector):

 # cluster: [**type==1**, Et [GeV], η, φ, E [GeV], Eem [GeV], Ehad [GeV], 0, 0, 0, 0]

 # track: [**type==2**, pt [GeV], η, φ, P [GeV], η_{outer}, φ_{outer}, charge, is_gen_muon, is_gen_electron]

Output:

- Multi output: **pid** (6d-one hot encoding) & **p4** (6d-vector):

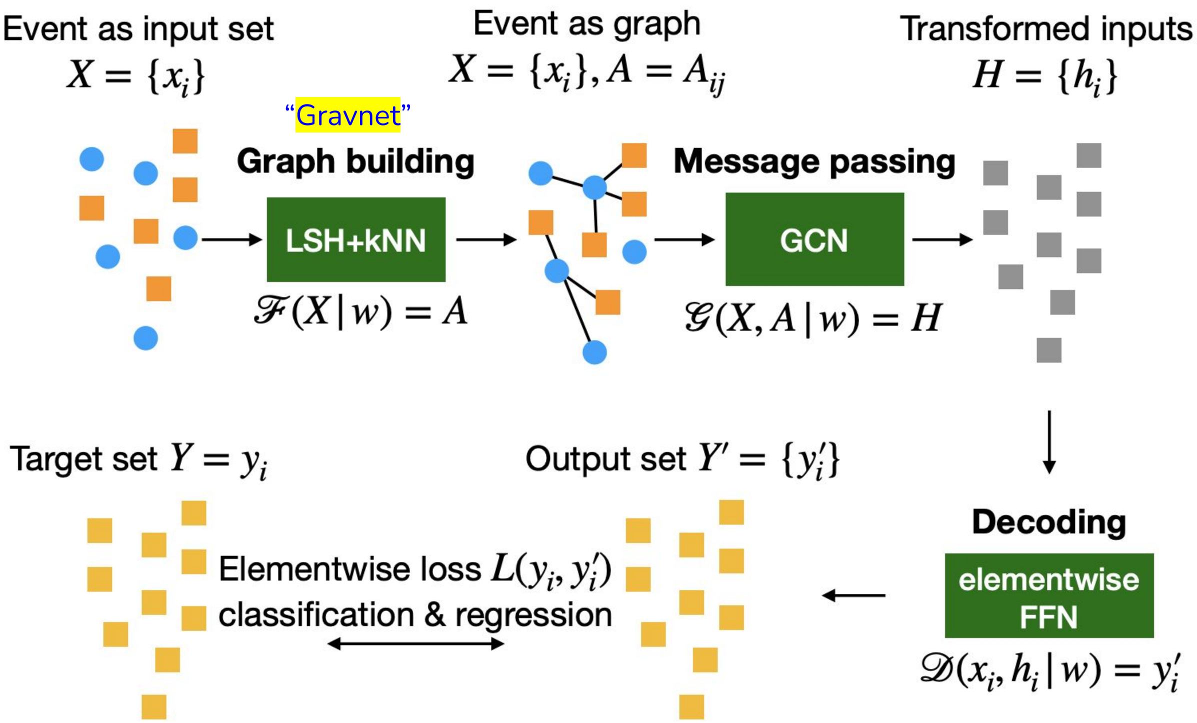
 # pid: [pid]

 # p4: [charge, pt [GeV], eta, sin phi, cos phi, E [GeV]]

ID	PF candidate type	Fraction per event
0	no reconstructed PFCandidate	3%
1	charged hadrons	52%
2	neutral hadrons	18%
3	photons	26%
4	electrons	0.2%
5	muons	0.07%

Eur. Phys. J. C 81, 381 (2021)

Visualizing the architecture:



Link to paper: https://arxiv.org/abs/2101.08578

Link to dataset: https://zenodo.org/record/4452283#.YA_SsGQzY-R

Link to code: https://github.com/jpata/particleflow

Project goals

- **Develop a different training setup for MLPF using pytorch**
- **Implement an explainable AI technique called Layerwise Relevance Propagation (LRP) on MLPF**

Other recent developments and research directions:

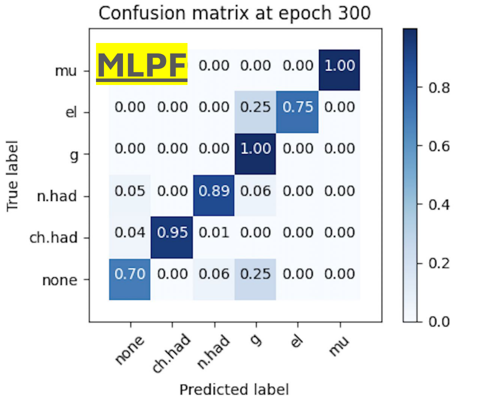
- CMSSW integration
- Transfer learning on a particle gun sample
- Quantization
- Hyperparameter optimization

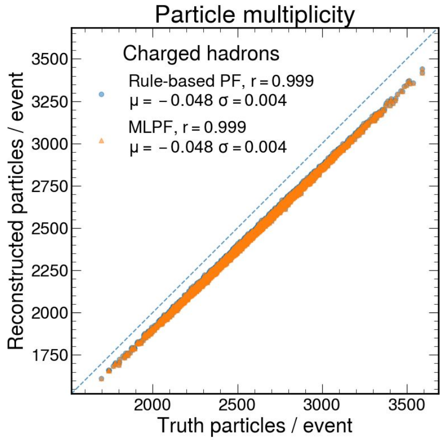
Performance

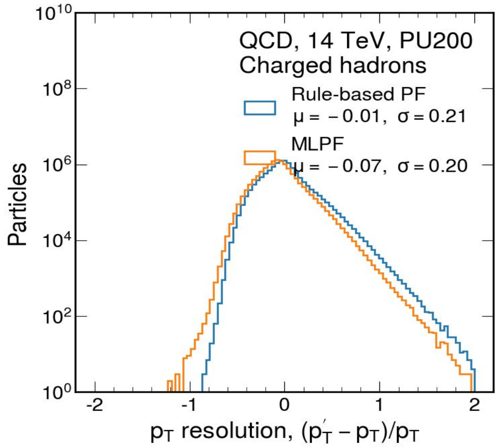
Results for the full training

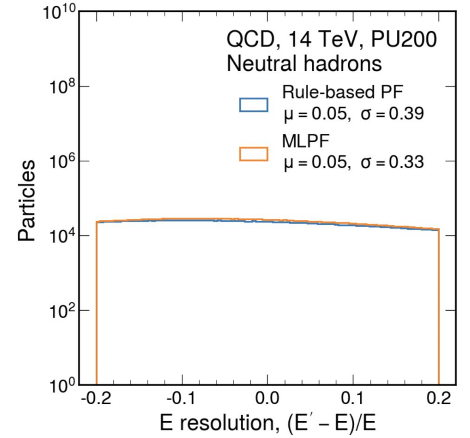
(reproduces the results of the MLPF paper using a different training setup)

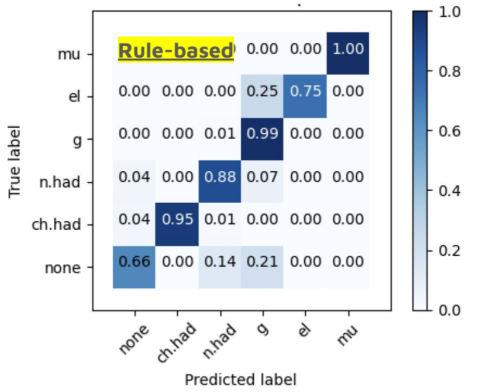
Conclusion: comparable performance to the rule-based algorithm

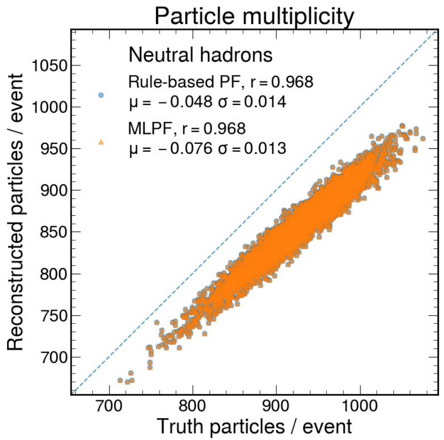


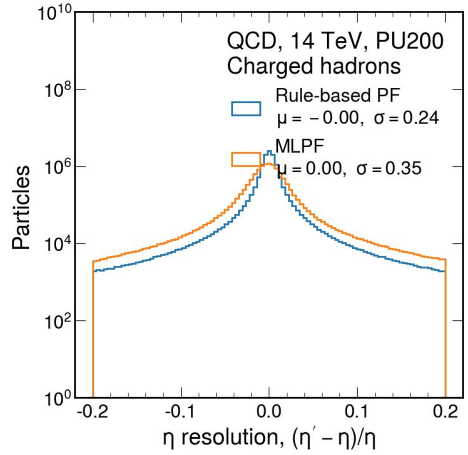


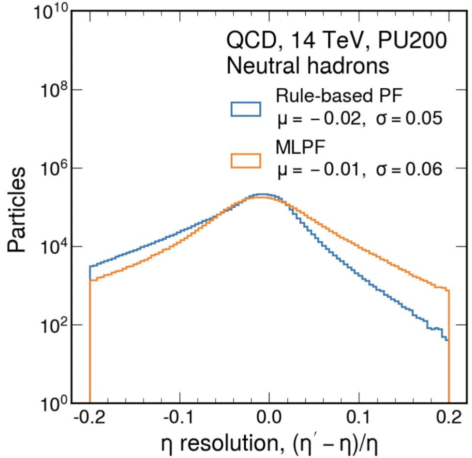










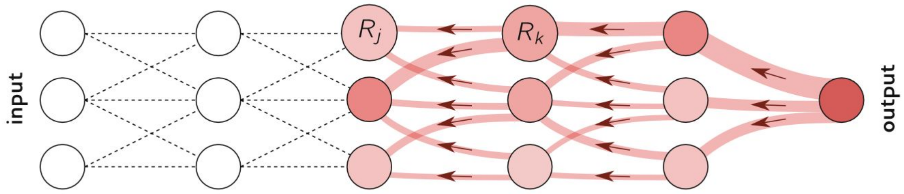


Interpretability

Interpretability

- **L**ayerwise **R**elevance **P**ropagation (LRP) [1]
- **LRP**: provides a systematic way of computing **relevance score** for each neuron
- Allows us to answer: “which detector elements were the most relevant when making inference?”

⇒ we draw **relevance score matrices (relevancy-map):** heatmaps of relevance scores for each classified PID (and each regressed variable) that shows the relevant parts of the graph for this prediction

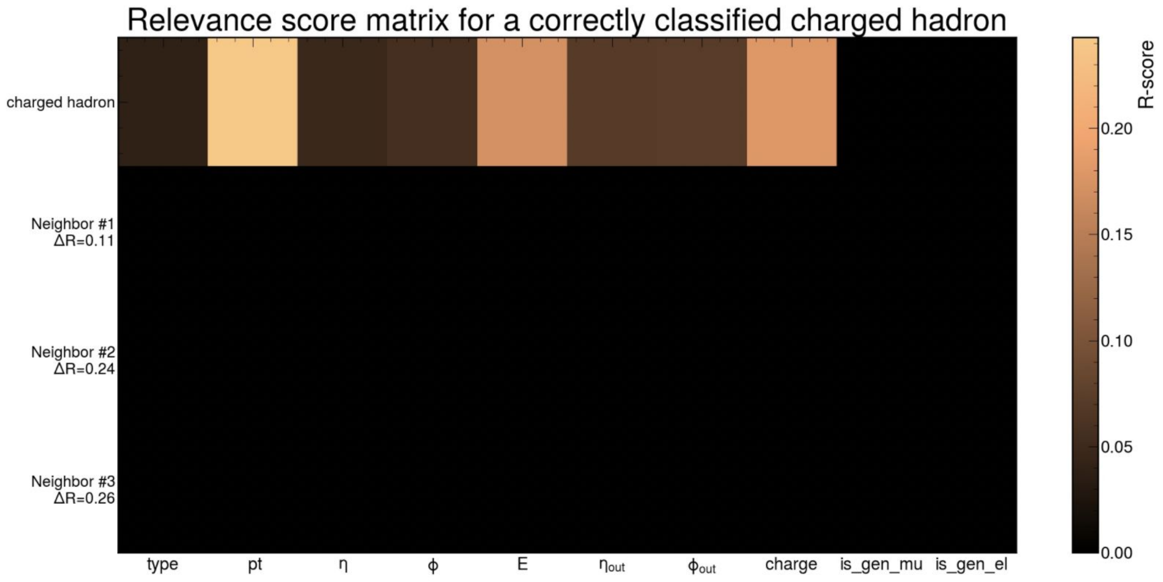


$$R_i^{(l)} = \sum_j \frac{z_{ij}}{\sum_{i'} z_{i'j}} R_j^{(l+1)} \quad \text{with} \quad \mathrm{z_{ij}} = \mathrm{x_i^{(l)}\, w_{ij}^{(l,l+1)}}$$

[1] https://doi.org/10.1007/978-3-030-28954-6_10

Relevancy-map: sample 1

- This is a heatmap plotted for **one PF-candidate** (in this case a **charged hadron** prediction)
- The **rows** correspond to the (relevant) **neighbours** of the charged hadron ordered by distance
- The **columns** correspond to the **12-d feature vectors**
- **Z-scale** (color) is the **relevance score** and brighter means more relevant

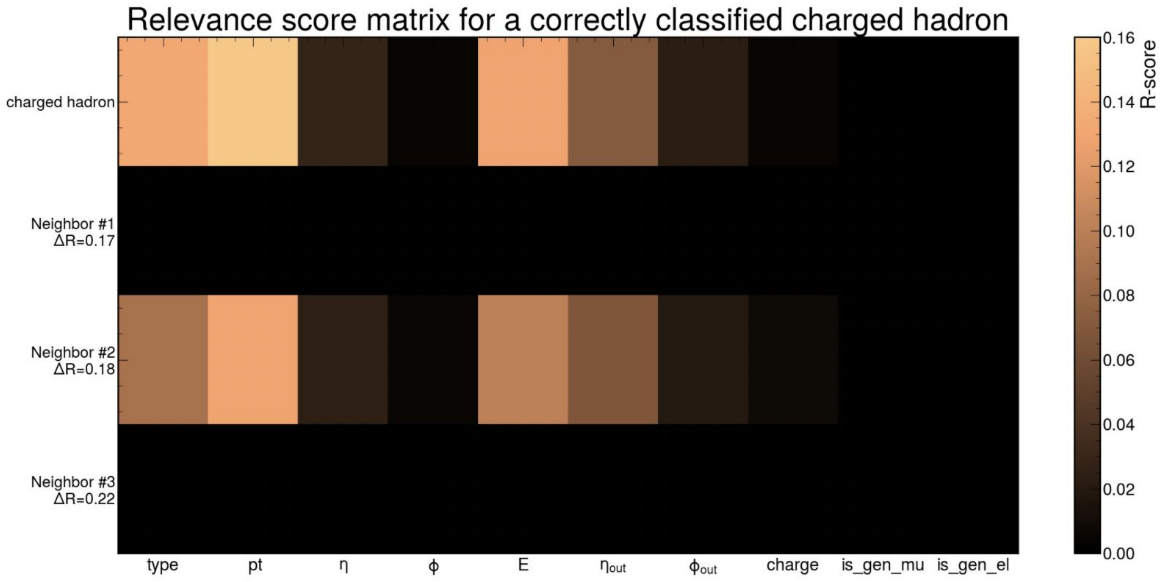


Takeaway from sample 1:

- The **"pt"** feature is the most relevant feature
- Neighbors were not really relevant for prediction

Relevancy-map: sample 2

- This is a heatmap plotted for **one PF-candidate** (in this case a **charged hadron** prediction)
- The **rows** correspond to the (relevant) **neighbours** of the charged hadron ordered by distance
- The **columns** correspond to the **12-d feature vectors**
- **Z-scale** (color) is the **relevance score** and brighter means more relevant



Takeaway from sample 2:

- Neighbor # 2 was relevant for prediction

Processing relevancy-maps

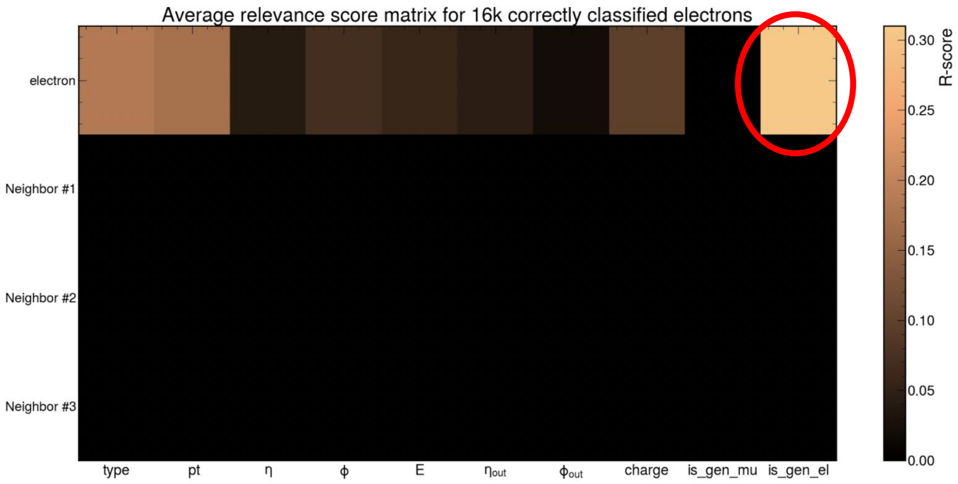
- **Recall**: we have one relevancy map per output neuron
- 12 output neurons * 5k nodes/per event * 5k QCD events → **300k relevancy-maps**
- To process this huge amount of relevancy-maps we:

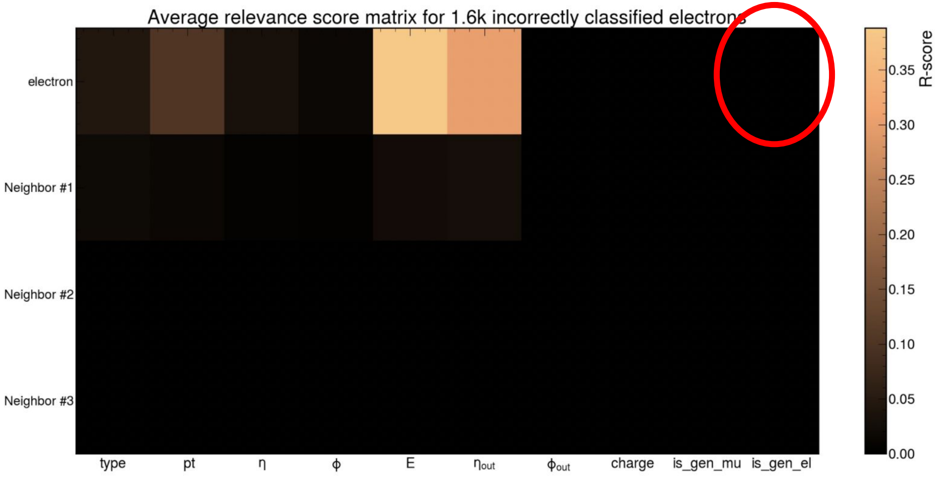
(1) **Average over relevancy-maps** after ordering neighbours by relevance

(2) **Make histogram plots**

Averaging over relevancy-maps

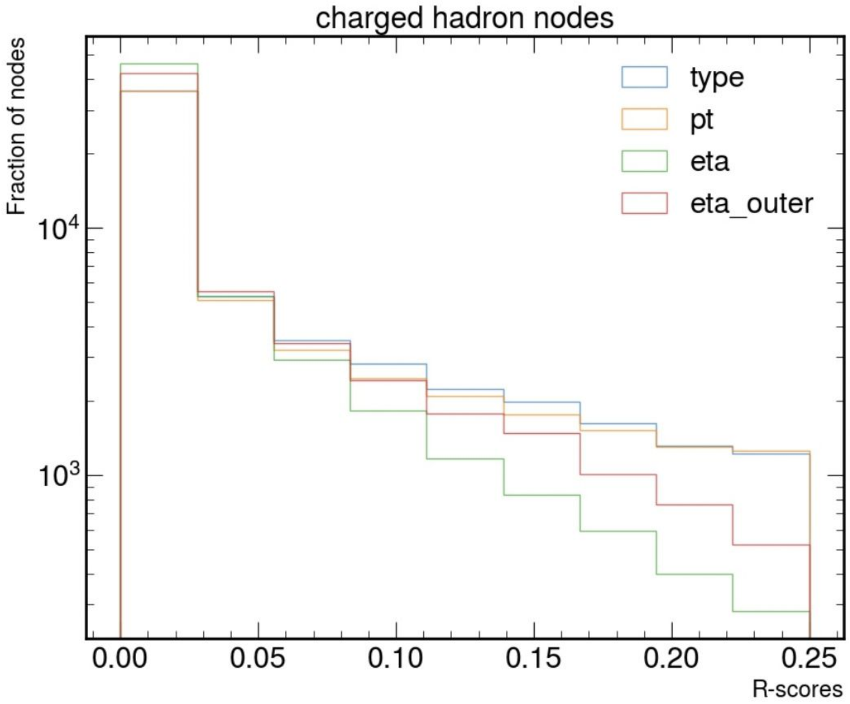
- Correctly classified electrons have high relevance score on ***is_gen_el*** feature
- This is one way to verify the LRP methodology

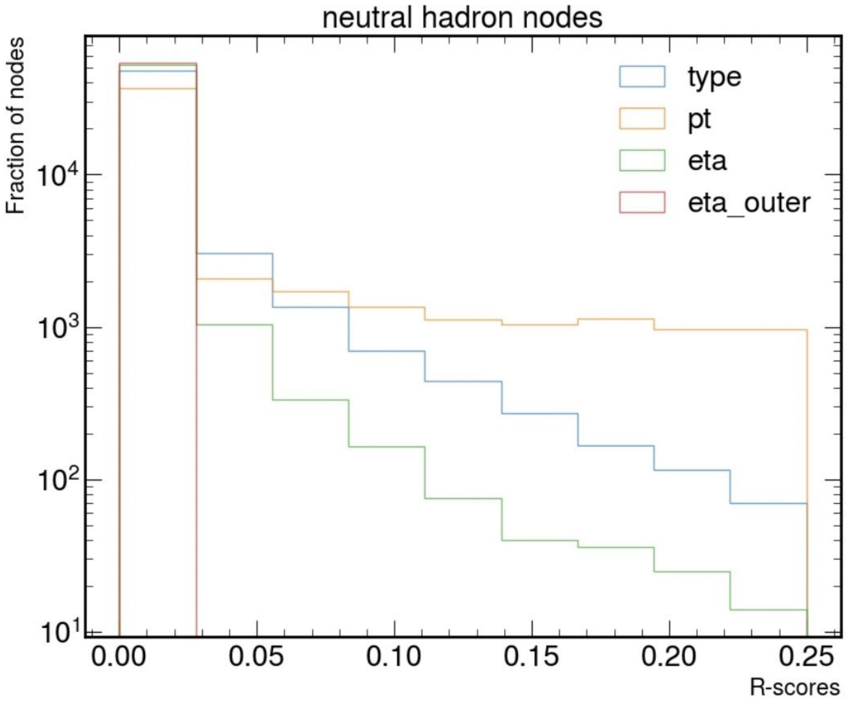




Histogram plots

- ***eta_outer*** is irrelevant when it comes to classifying **nhadrons**





Summary and Outlook

- We developed an alternative baseline MLPF model using pytorch
- We explored the application of an explainable AI technique on MLPF

Further steps:

- Model optimization:
  - Time complexity (explore different graph building techniques)
  - Bfloat or 8-bit quantization
- Use LRP to understand the model’s decision making
- Deployment on new hardware platforms (FPGAs/GPUs/Voyager)
- Application to real CMS/other collider data

Backup

Current pytorch architecture:

This skip connection in the last DNN (which feeds the input features again) is crucial for the regression part to give good results

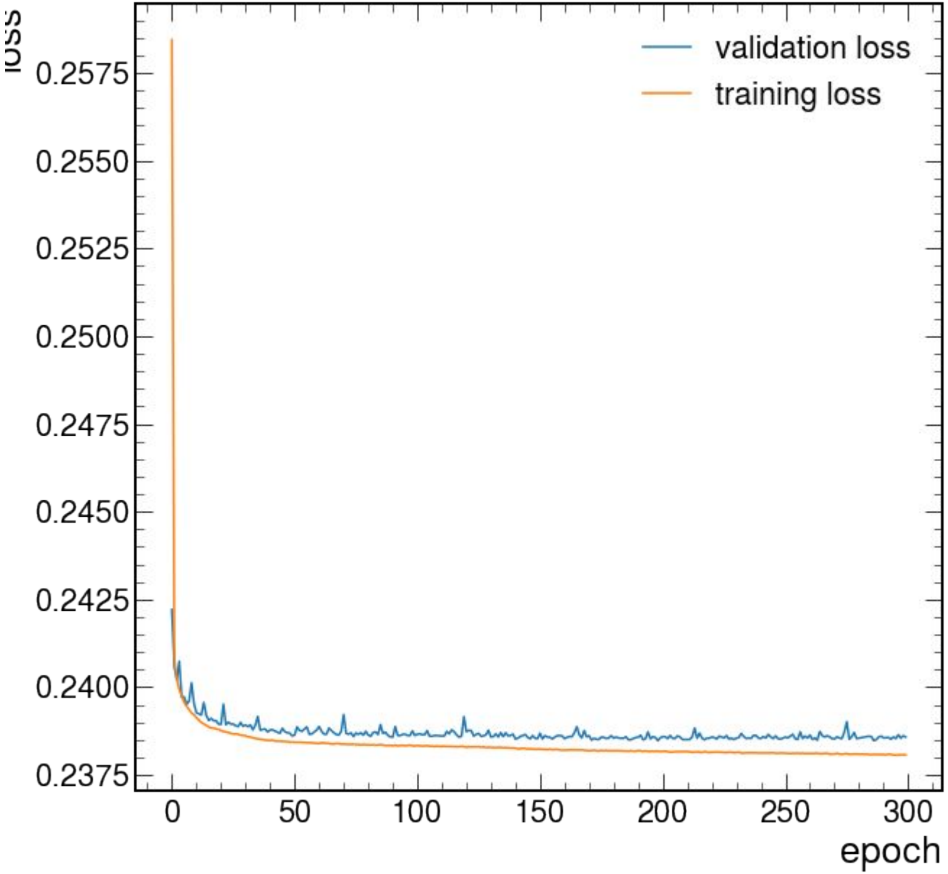
```
# (1) DNN: encoding/decoding of all tracks and clusters
self.nn1 = nn.Sequential(
    nn.Linear(input_dim=12, 64),
    self.elu(),
    nn.Linear(64, 64),
    self.elu(),
    nn.Linear(64, 12),
)

# (2) CNN: Gravnet layer
self.conv1 = GravNetConv(12, 64, space_dim=4, propagate_dimensions=22, k_nearest=16)

# (3) DNN layer: classifying PID
self.nn2 = nn.Sequential(
    nn.Linear(64, 256),
    self.elu(),
    nn.Linear(256, 256),
    self.elu(),
    nn.Linear(256, 256),
    self.elu(),
    nn.Linear(256, output_dim_id=6),
)

# (4) DNN layer: regressing p4
self.nn3 = nn.Sequential(
    nn.Linear(output_dim_id + input_dim + 64, 256),
    self.elu(),
    nn.Linear(256, 256),
    self.elu(),
    nn.Linear(256, 256),
    self.elu(),
    nn.Linear(256, output_dim_p4=6),
)
```

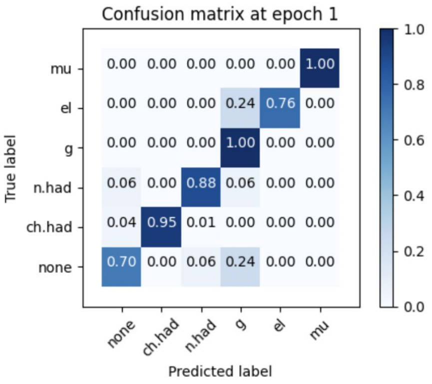
Results for the full dataset training:

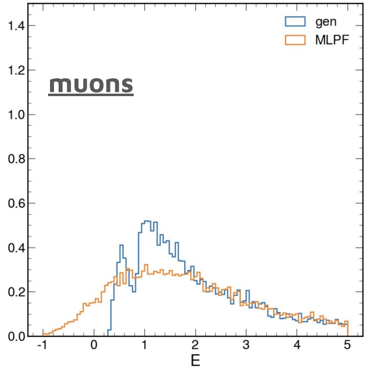


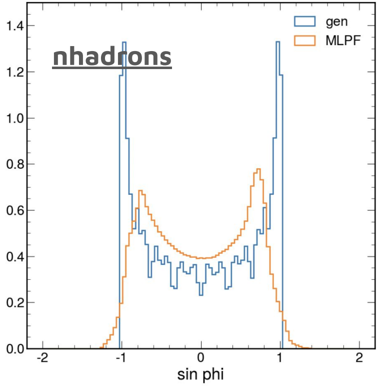
Results for the full dataset training:

- **Interesting to note:** classification converges quickly (more so for the common classes), the regression of the least represented classes gets better with more training

After 1 epoch







After 30 epochs