Explaining machine-learned particle-flow reconstruction

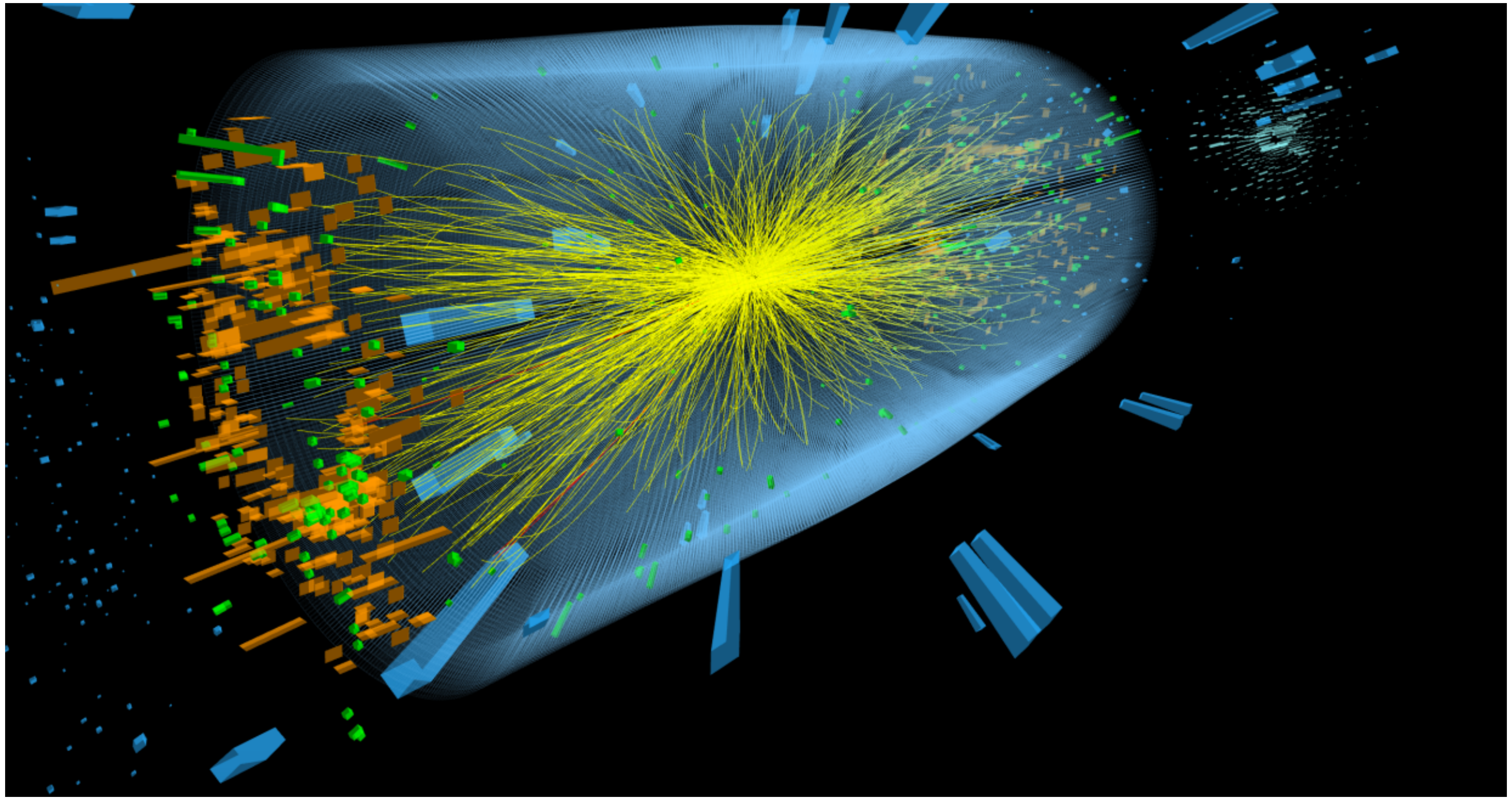
Farouk Mokhtar, UCSD, Duarte Lab
Mentor: Javier Duarte
Additional mentors: Joosep Pata, Daniel Diaz, Jean-Roch Vlimant

LHC at CERN

- We collide ~2800 proton bunches every 25ns (with 115 billion protons per bunch) to generate ~10 Petabytes/sec of data before any filtering or suppression

- At high enough energies unstable particles are created

- We can use machine learning (ML) to reconstruct events by looking at decay products of the particle collisions



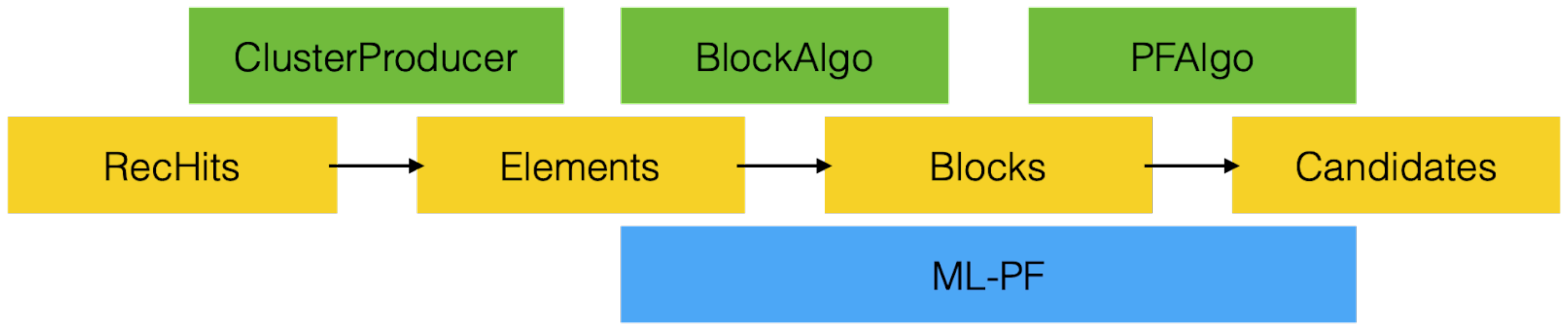
Overview

- **Particle-flow (PF)**: a global event reconstruction that combines information from calorimeter clusters and tracks to reconstruct stable particles

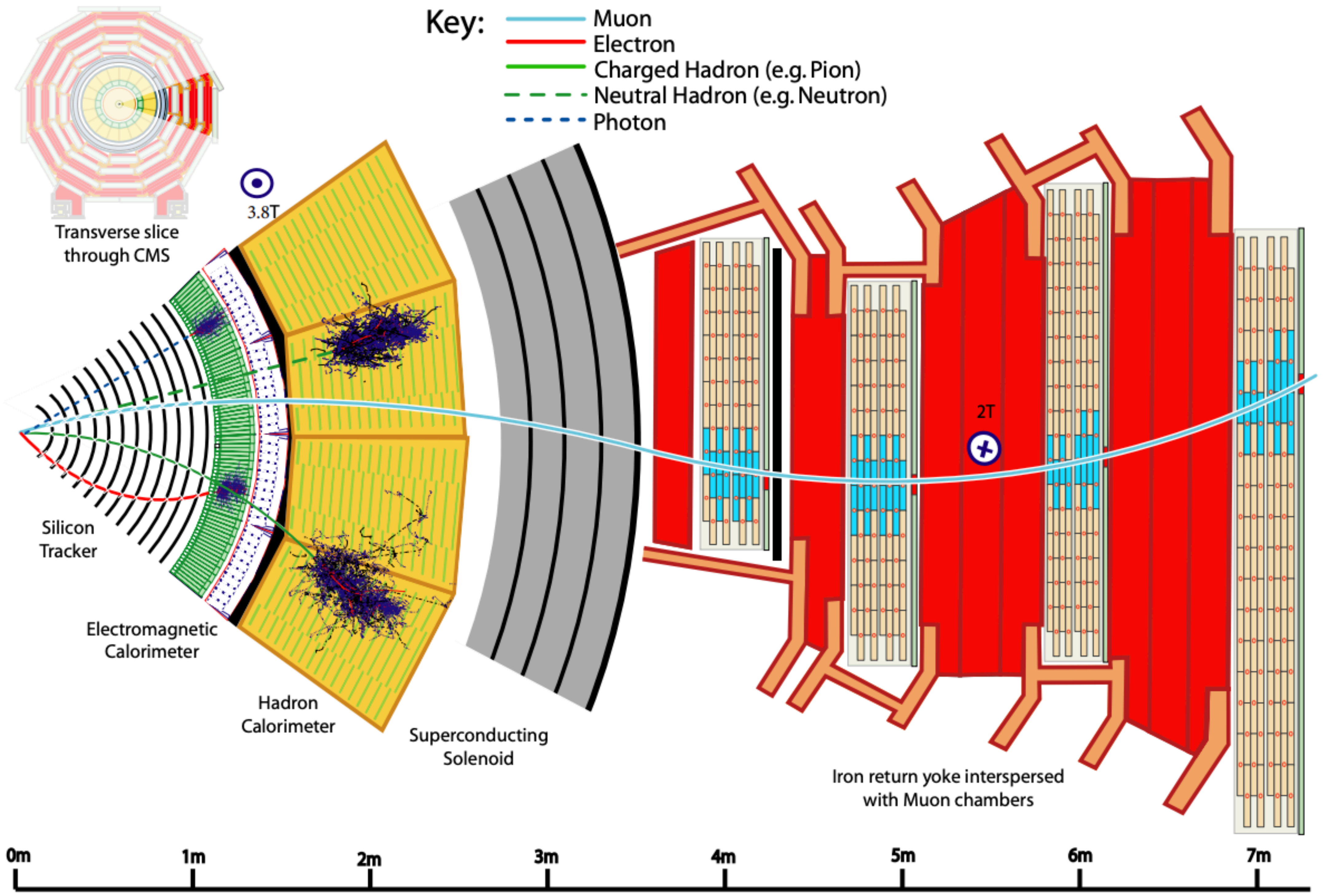
- **Machine-learned particle-flow (MLPF)**: an evolution of the rule-based PF algorithm for heterogeneous computing platforms such as GPUs using supervised machine learning with **graph neural networks**

- Inputs are **PF elements**: calorimeter energy clusters and tracks

- Outputs are **PF candidates**: particles' IDs and kinematics



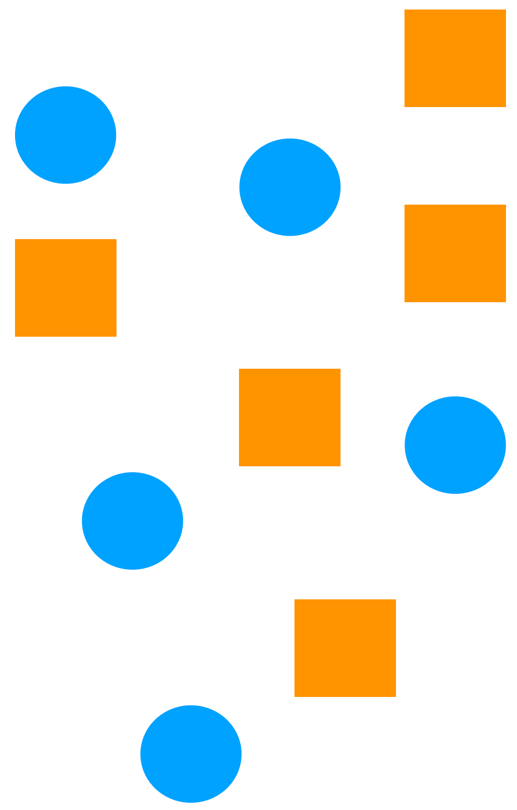
Aim to speed up the parts of PF reconstruction
that have not already been ported to GPU!



MLPF

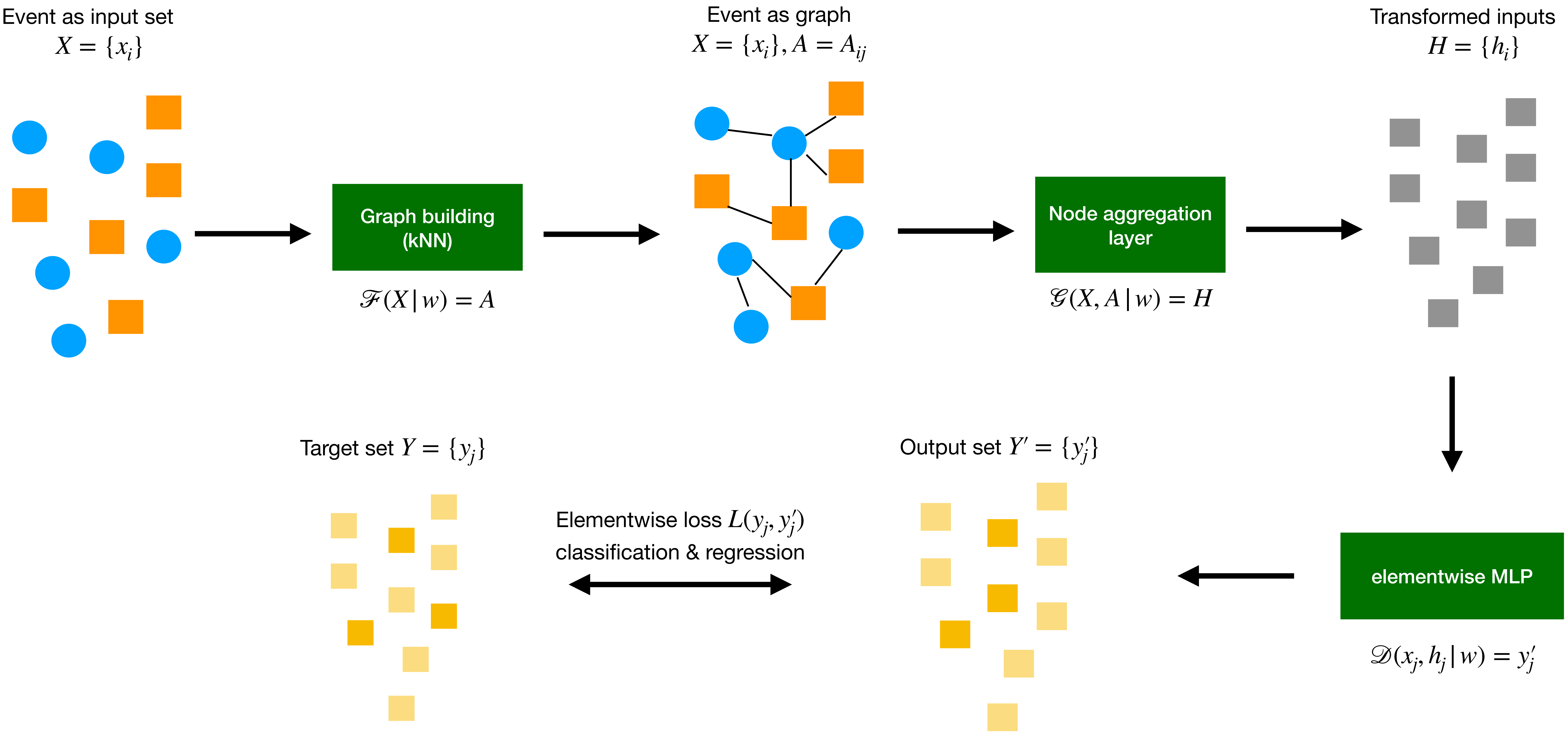
- **Dataset:** detector-agnostic DELPHES dataset[1]
  - 50k top quark-antiquark (tt) events for training
  - 5k QCD events for testing
- **Tasks:**
  - Classification of particles **PID** (classification)
  - Prediction of kinematics **p** (regression)
- **Input:**
  - Event is represented by a graph (~5k nodes)
  - Node in the graph is a detector element (12D feature vector):
    1. Cluster: [type==1, E_T, η, ϕ, E, E_{ECAL}, E_{HCAL}, 0, 0, 0, 0]
    2. Track: [type==2, p_T, η, ϕ, P, η_{outer}, ϕ_{outer}, q, is_gen_muon, is_gen_electron]
- **Multi-output:**
  - PID: [pid]
  - p: [q, pt [GeV], eta, sin phi, cos phi, E [GeV]]

Event



[1] https://zenodo.org/record/4452283#.YA_SsGQzY-R

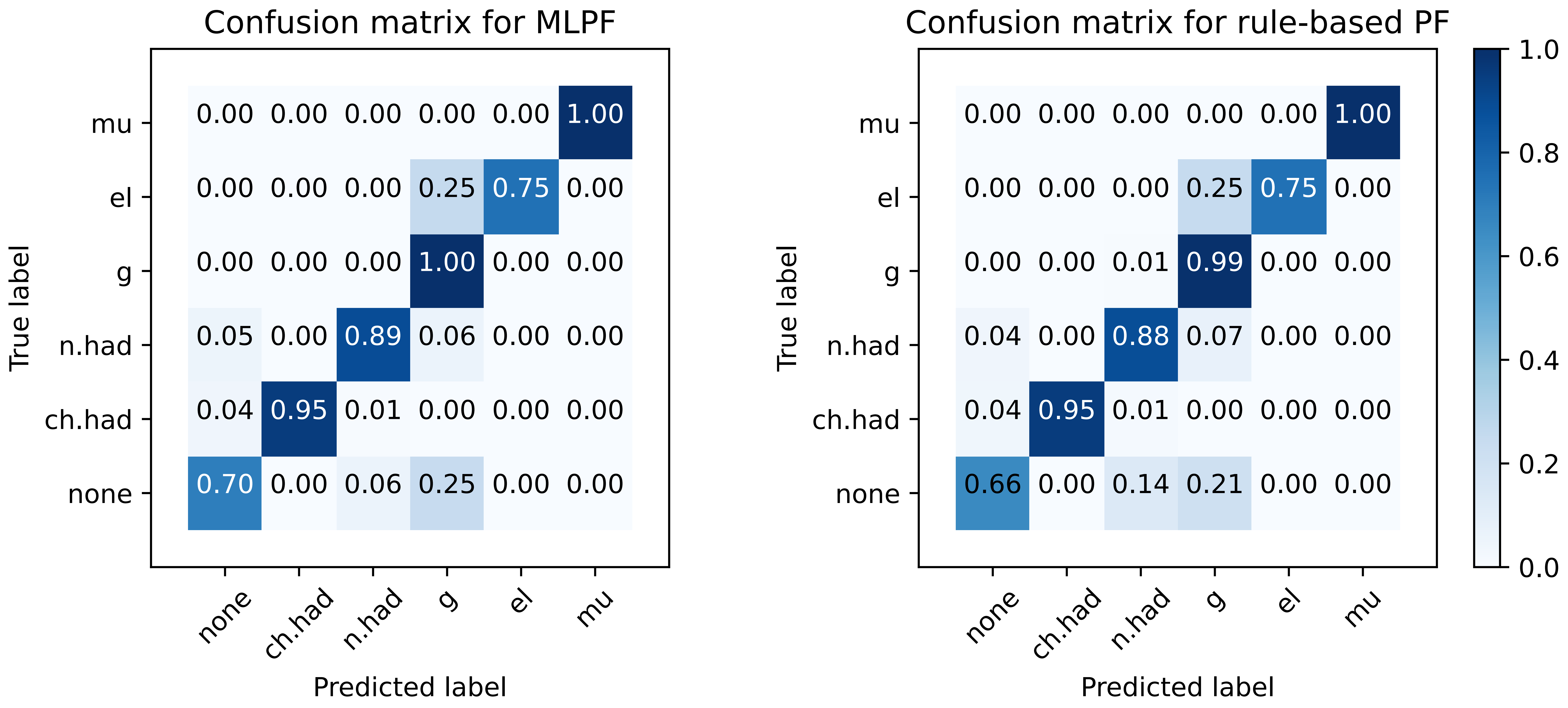
Visualizing the architecture



Link to paper: https://arxiv.org/abs/2101.08578
Link to code: https://github.com/jpata/particleflow

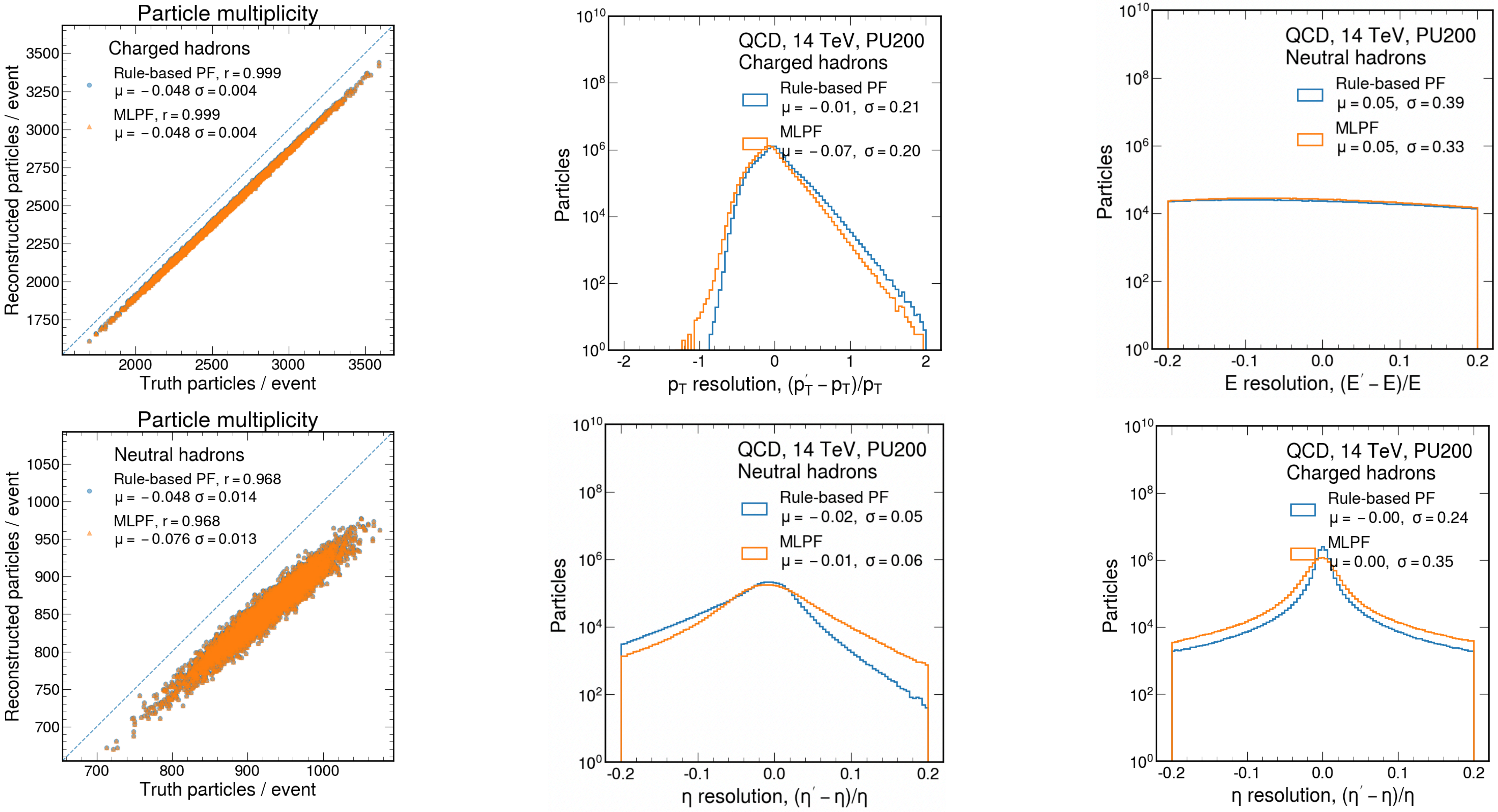
MLPF performance

Results for the full training (300 epochs)



Conclusion: MLPF shows comparable classification performance to the rule-based algorithm

Results for the full training (300 epochs)



Conclusion: MLPF shows comparable regression performance to the rule-based algorithm

Interpretability of MLPF

Interpretability

- ML models often suffer from a lack of interpretability ~ "black boxes"

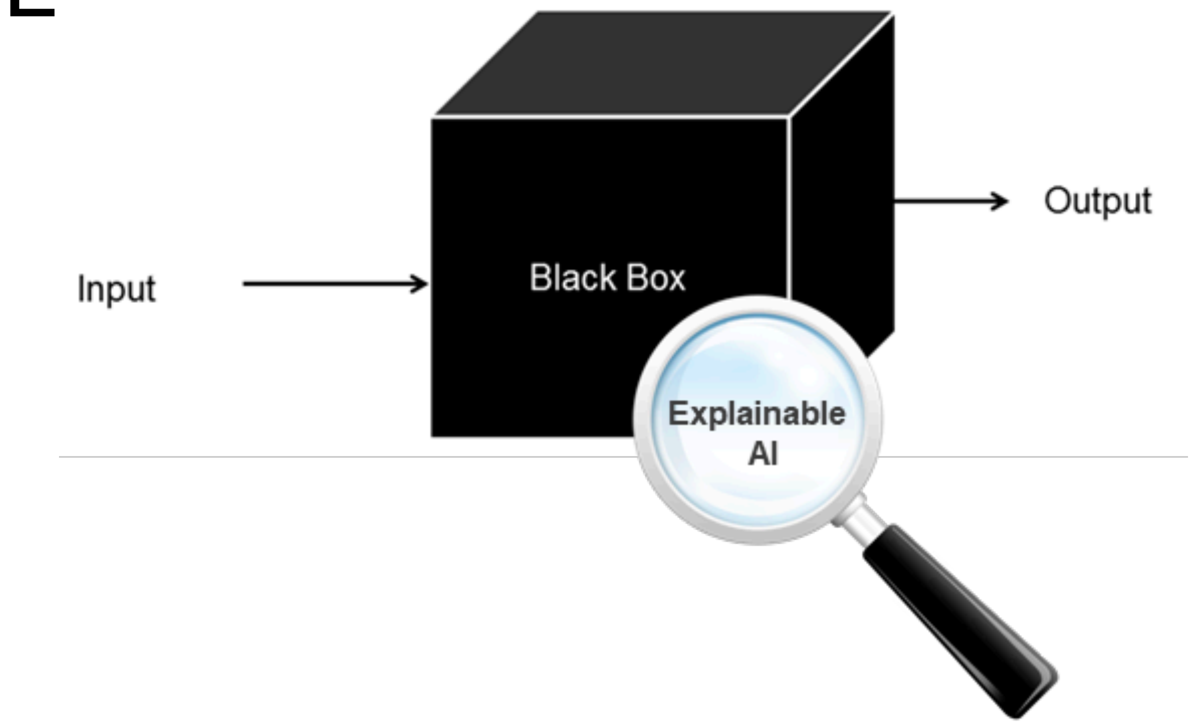
- **Explainable AI**[1] refers to the set of techniques employed to provide explanations for ML model predictions

- For us, understanding the decision-making behind MLPF is valuable as:

  - To increase confidence

  - To ensure robustness under changing conditions

  - To glean new insights about the detector performance or reconstruction not currently utilized by the rule-based PF algorithm



[1] https://link.springer.com/book/10.1007/978-3-030-28954-6

LRP

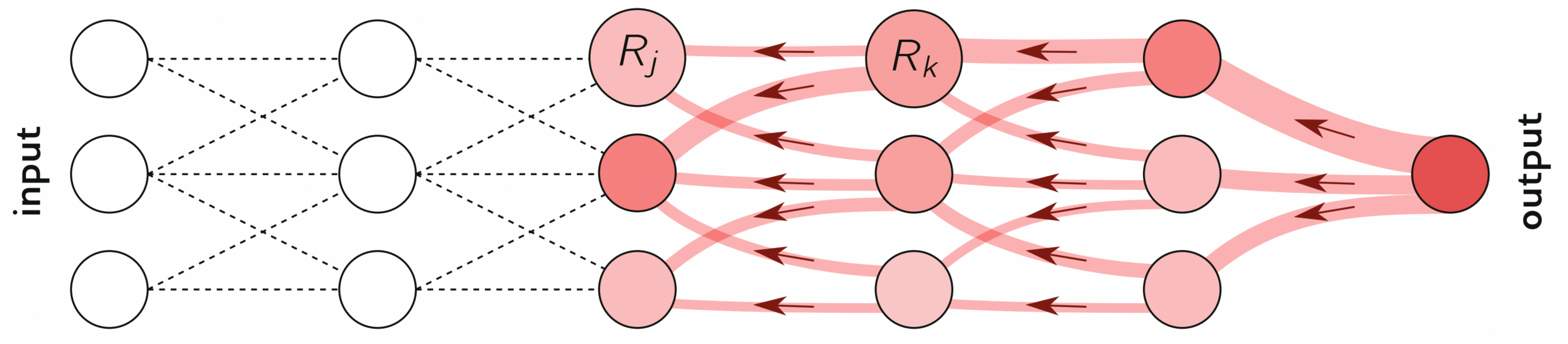
- **Goal**: to learn which detector elements were the most relevant for each given output class prediction

- **Layerwise relevance propagation[1] (LRP)**: provides a systematic way of computing relevance scores (**R-scores**) for each neuron

$$R_j^{(l)} = \sum_k \frac{z_{jk}}{\sum_m z_{mk}} R_k^{(l+1)} \quad \text{with} \quad z_{jk} = x_j^{(l)} w_{jk}^{(l+1)}$$

- A neuron's R-score is a measure of its contribution to the model's inference

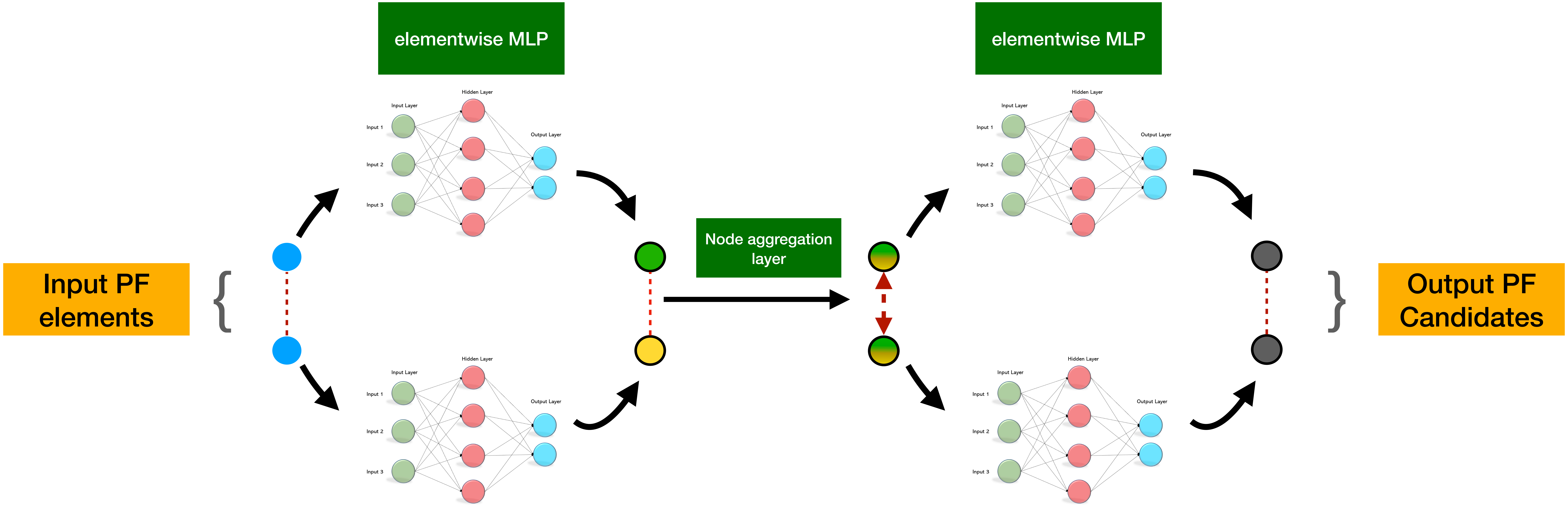
- The following figure illustrates the flow of R-scores for a multilayer perceptron (MLP)



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

LRP on graphs

- A key feature of GNNs is the **Node aggregation layer** which will require special treatment when applying LRP
- It is useful to note that besides this layer, everything else about the MLPF is a sequence of MLP layers that operate on the node-level which standard LRP rules apply
- For example, consider the following MLPF visualization

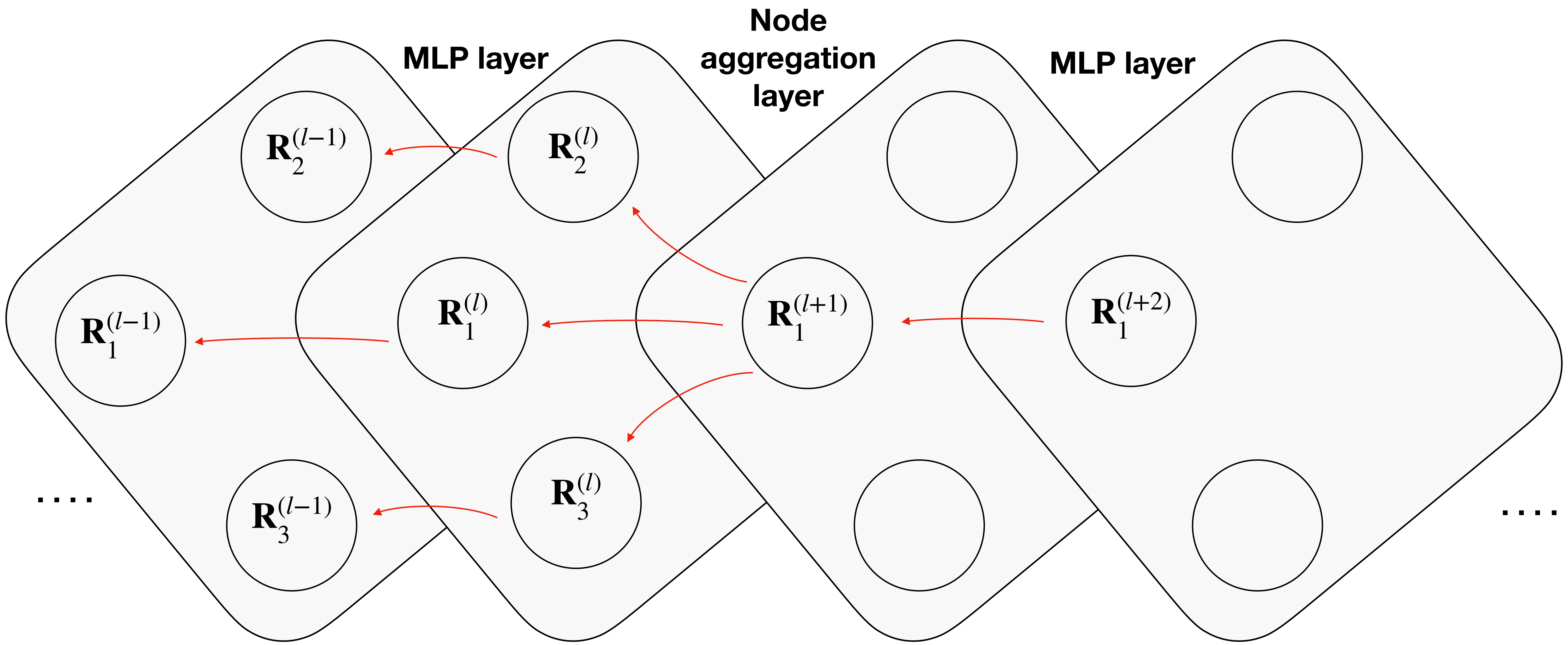


LRP on graphs

- A key feature of GNNs is the **Node aggregation layer**
- This layer can be casted as an MLP layer with the following forward propagation rule $z_k^{(l+1)} = x_j^{(l)} A_{jk}$ where A_{jk} is the adjacency matrix
- This allows us to use the LRP standard rule in a straightforward fashion as

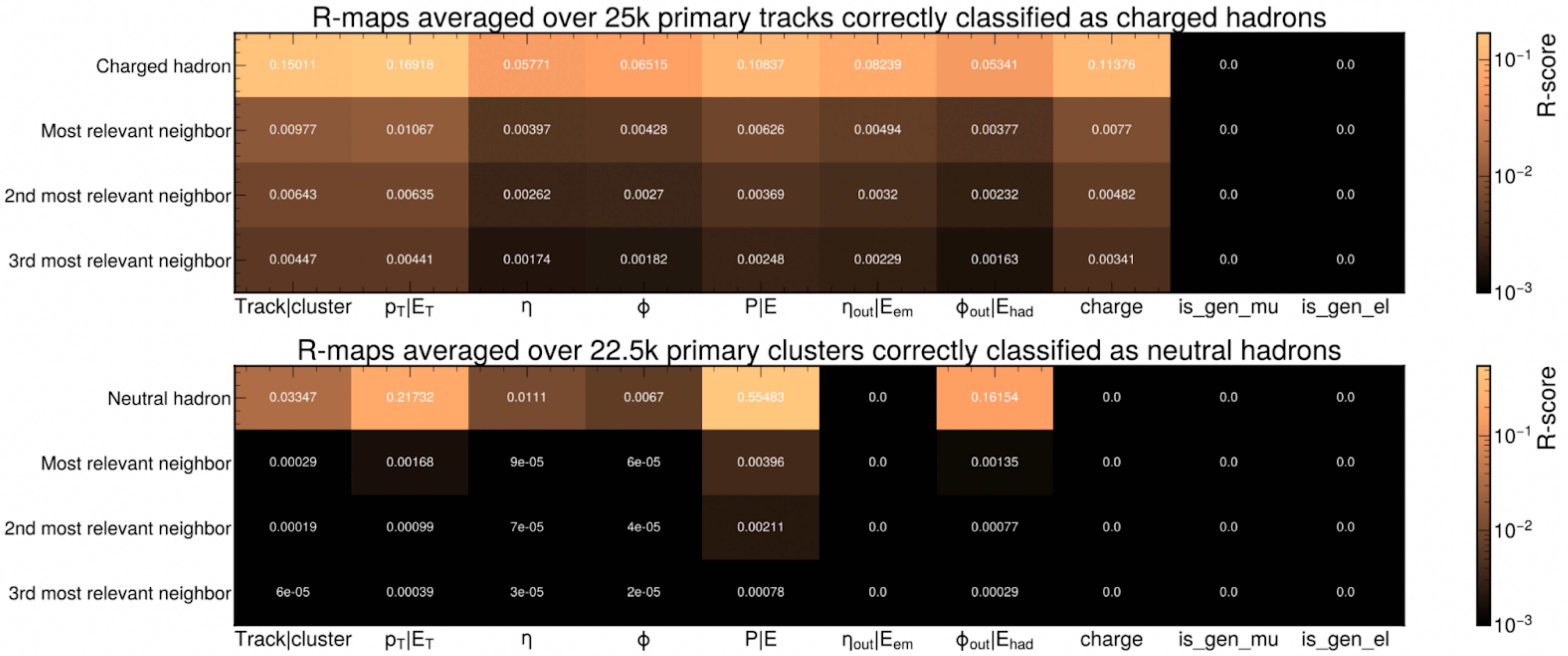
$$\mathbf{R}_j^{(l)} = \sum_k \frac{x_j A_{jk}}{\sum_m x_m A_{mk}} \mathbf{R}_k^{(l+1)}$$

where now the R-scores, originally $R_j^{(l)}$, are promoted to vectors $\mathbf{R}_j^{(l)}$ that run over all nodes in the graph



R-maps

- We construct relevance score matrices (**R-maps**)[1] to visualize the R-scores of all neurons in the model



Reading the R-maps:

- The rows correspond to the (relevant) neighbors of the classified particle
- The columns correspond to the 12D input features
- Z-scale (color) is the R-score (normalized per R-map)

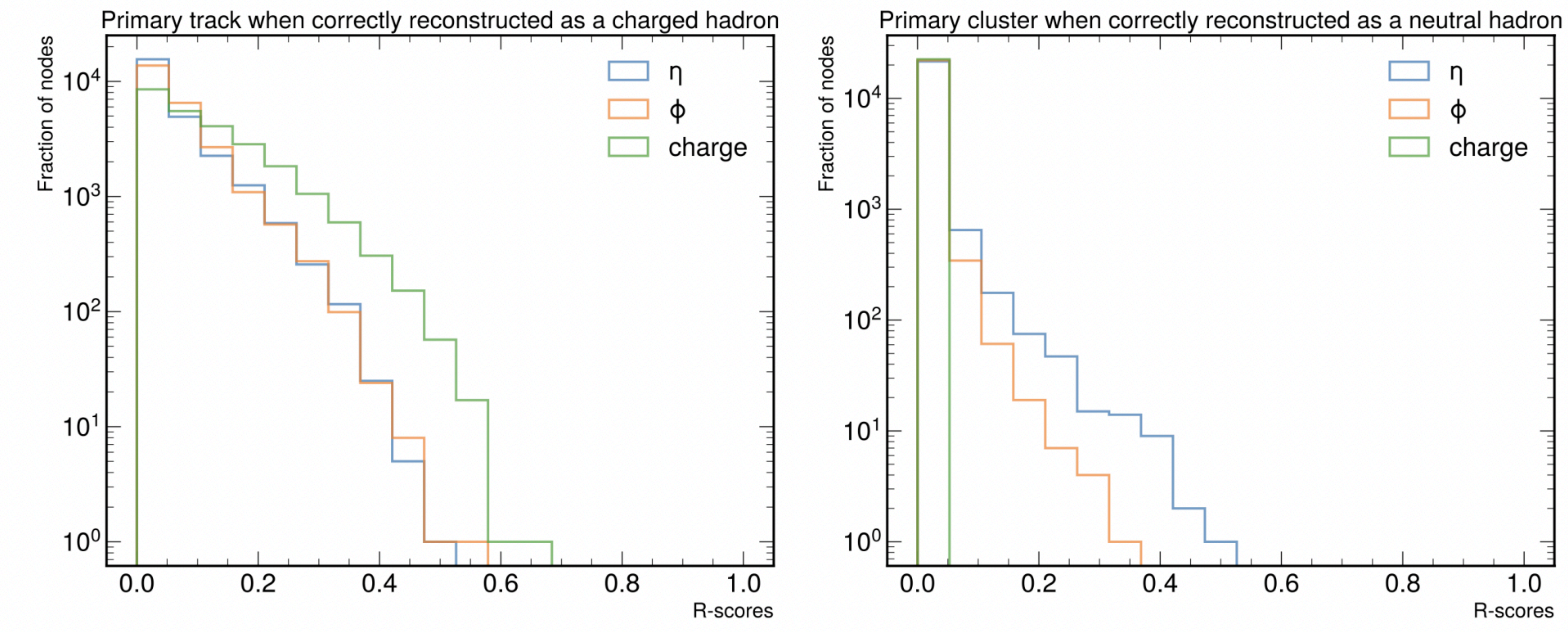
Takeaway:

- Different features are more active when classifying different classes
- Charged hadrons make use of neighbor information more than neutral hadrons (perhaps because they may give rise to multiple tracks)

[1] https://doi.org/10.48550/arXiv.2111.12840

Histograms

- Another way of visualizing the R-scores



- As expected, we find that the charge feature is significantly more relevant for identifying charged hadrons compared to neutral hadrons
- This is a sanity check but paves the way for exploring possibly other connections that can be inferred using LRP

Summary

- Explored the application of an explainable AI technique on MLPF
- Explored how the number of relevant nodes change when classifying different particles
- Explored which features were more relevant when classifying different particles

Outlook

- This effort attempts to examine how an ML model makes its decisions for a challenging particle physics task
- Allows us to better trust the ML model (and even potentially learn from it)
- LRP can be also used as a **pruning method** to limit the resources required to perform PF-reconstruction

Future steps

- Apply LRP to an MLPF model trained on "gen-level" targets in CMS

FAIR4HEP: cookie-cutter project

By: Ishaan Kavoori (detailed presentation is here)

- **Goal:** to create a standard project template to publish "FAIR" AI models
- FAIR → **Findable**, **Accessible, Interoperable, Reusable**
- Code is open source, people can collaborate on adding features!
- Command line utility that will create a project directory from a given template directory (documentation here)

```
cookiecutter <online repository link>
```

What is included?

- Auto-download data from source (e.g. Zenodo) and specify data DOI in README – **Interoperable**
- Facilitates publishing the results of a trained model on a platform (e.g. DLHub) – **Reusable**, **Interoperable**
- A Dockerfile skeleton to publish the docker image on DockerHub – **Reusable**, **Interoperable**
- Resulting repo can also be published on Zenodo to generate repo DOI – **Accessible**

Thanks!