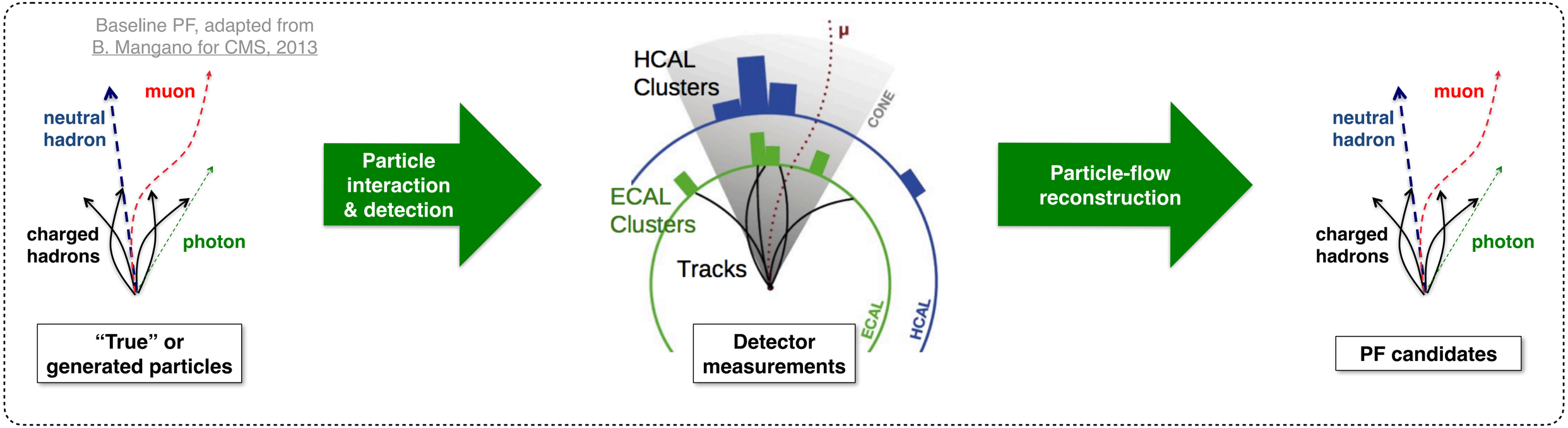
Machine Learning for Particle Flow reconstruction at CMS

Joosep Pata (NICPB, Estonia)
on behalf of the CMS Collaboration

contact: cms-conveners-particleflow@cern.ch
December 2, 2021

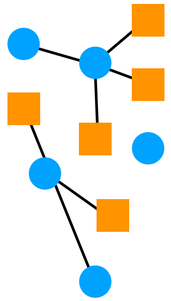
ACAT 2021
Daejeon, South Korea

Particle Flow in CMS



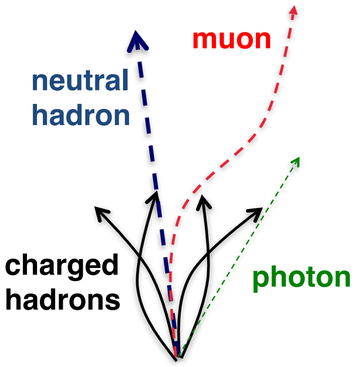
The CMS particle-flow algorithm aims to identify and reconstruct individually all of the particles produced in a collision, through an optimal combination of the information from the entire detector.

Machine learned particle flow (MLPF)



Graph neural network

Machine-learned particle-flow reconstruction

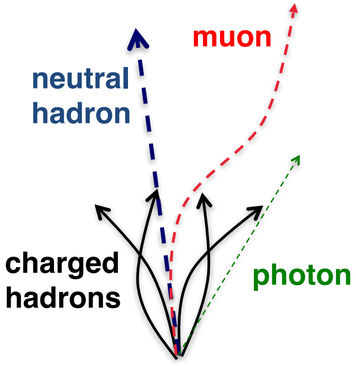


MLPF candidates

Extract features

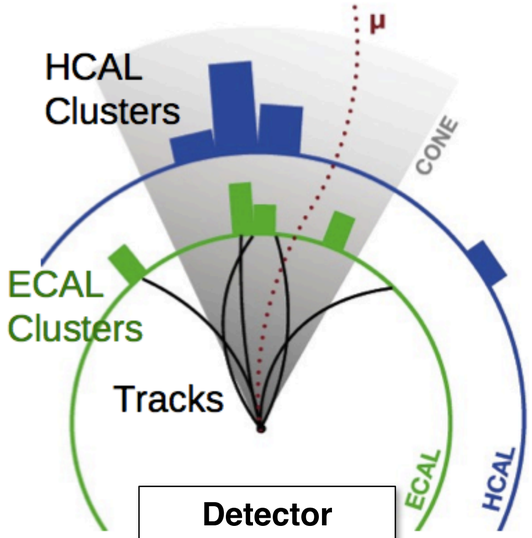
Compare via loss function

Baseline PF, adapted from B. Mangano for CMS, 2013



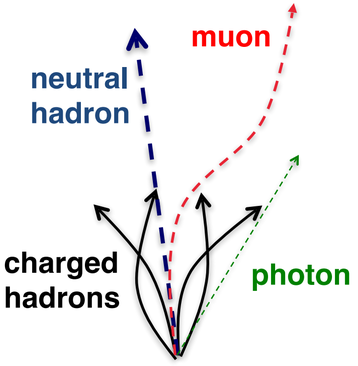
"True" or generated particles

Particle interaction & detection



Detector measurements

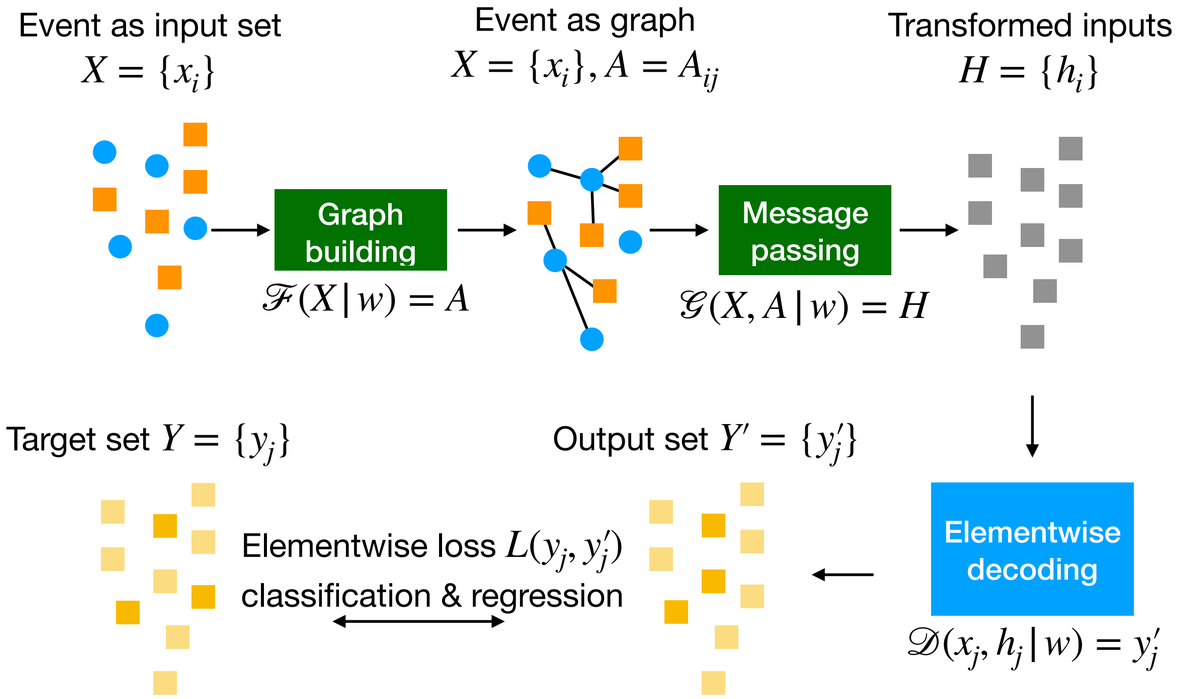
Particle-flow reconstruction



PF candidates

MLPF overview

- **Full event processing**: list of tracks and clusters in the event → particle candidates
- A **per-particle loss** function based on multi-classification and regression
  - Current **training targets are based on the baseline particle flow** candidates
  - A **generator-level training may allow to improve the physics response** with respect to the baseline PF
- A dynamic graph neural network model generates **multiple internal graph representations**
  - **Graph structure optimized on the fly**
- Based on Eur. Phys. J. C (2021) 81: 381



$$x_i = [\text{elem . type}, p_\text{T}, E_\text{ECAL}, E_\text{HCAL}, \eta, \phi, \eta_\text{outer}, \phi_\text{outer}, q, \ldots]$$

$$y_j = [\text{PID}, p_\text{T}, E, \eta, \phi, q], \quad \text{PID} \in \{\text{none, charged hadron, neutral hadron}, \gamma, e^\pm, \mu^\pm, \ldots\}$$

$$h_i \in \mathbb{R}^{N_\text{hidden}}$$

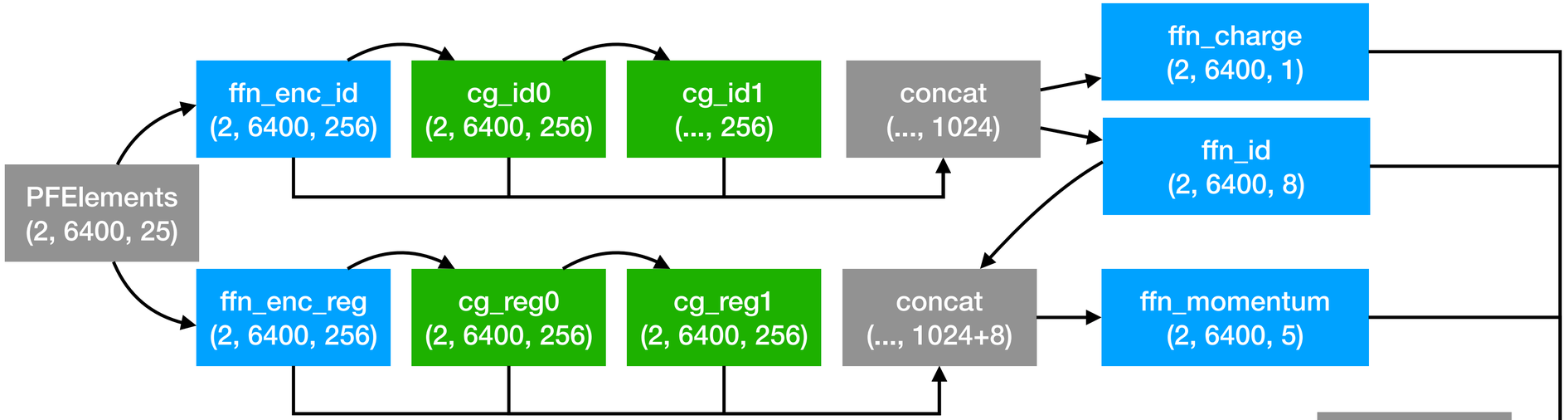
Trainable neural networks: $\mathcal{F}, \mathcal{G}, \mathcal{D}$

● - track, ■ - calorimeter cluster, ■ - encoded element

■ - target (predicted) particle, ■ - no target (predicted) particle

Tensorflow implementation

As an example (batches, elements, features) = (2, 6400, 25)



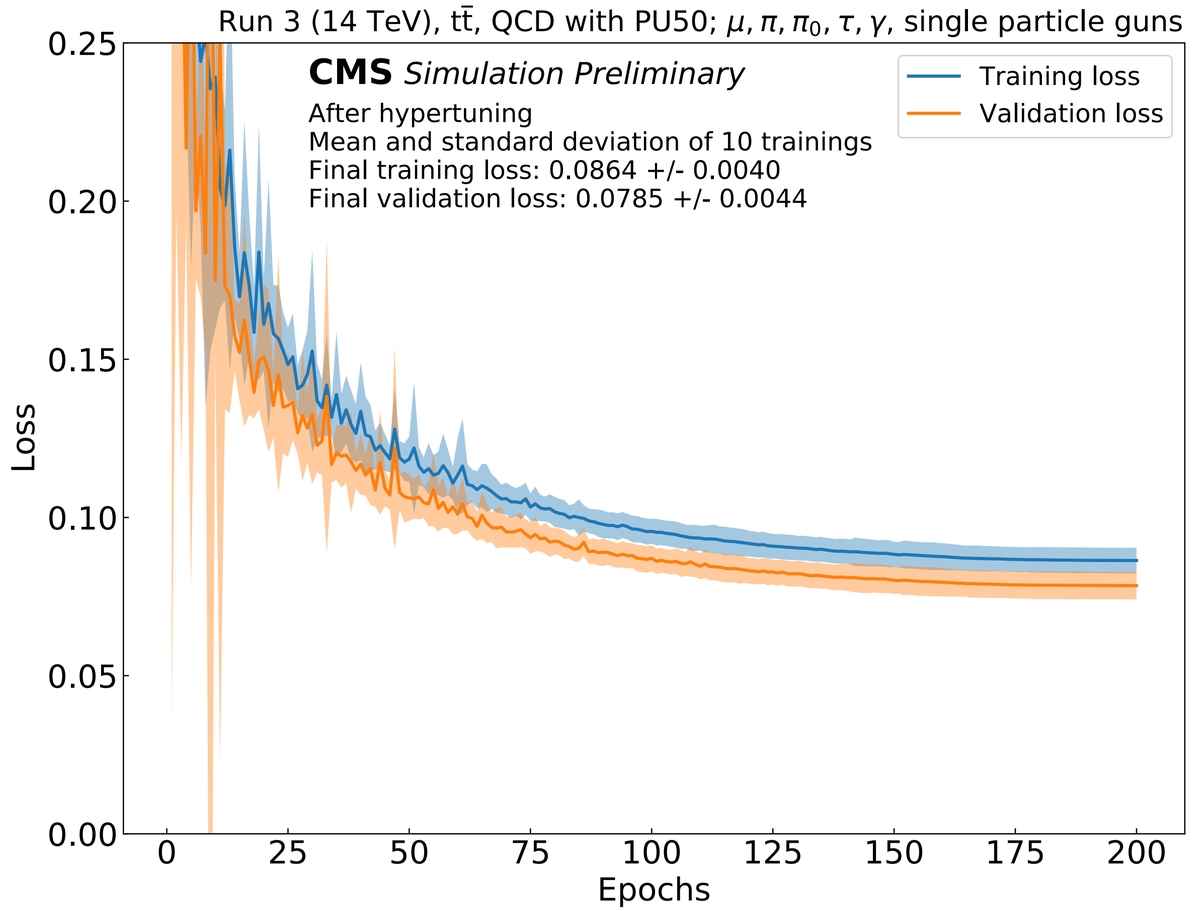
Stacked CombinedGraph (cg) layers, each builds a a new graph in a learnable way and propagates information using graph convolutions.

Training

- The MLPF model is trained on **approximately 40k simulated ttbar and tau** events with PU
  - We augment by using single-particle gun samples, significantly improving the performance
- Training takes **approximately 5 days on 5 consumer-grade GPUs**

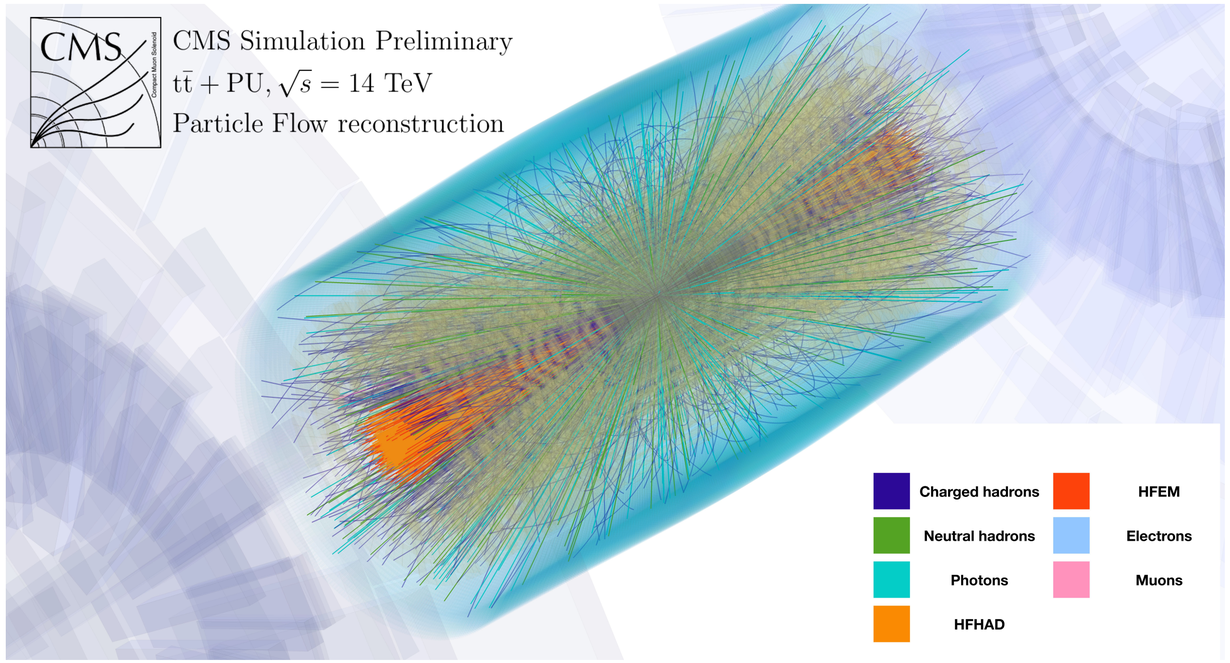
sample fragment	PU configuration	MC events
top-antitop pairs	flat 55-75	20k
$Z \to \tau\tau$ all-hadronic	flat 55-75	20k
single electron flat $p_T \in [1, 100]$ GeV	no PU	400k
single muon flat $p_T \in [0.7, 10]$ GeV	no PU	400k
single π^0 flat $p_T \in [0, 10]$ GeV	no PU	400k
single π flat $p_T \in [0.7, 10]$ GeV	no PU	400k
single τ flat $p_T \in [2, 150]$ GeV	no PU	400k
single γ flat $p_T \in [10, 100]$ GeV	no PU	400k

Table 1: MC simulation samples used for optimizing the MLPF model.

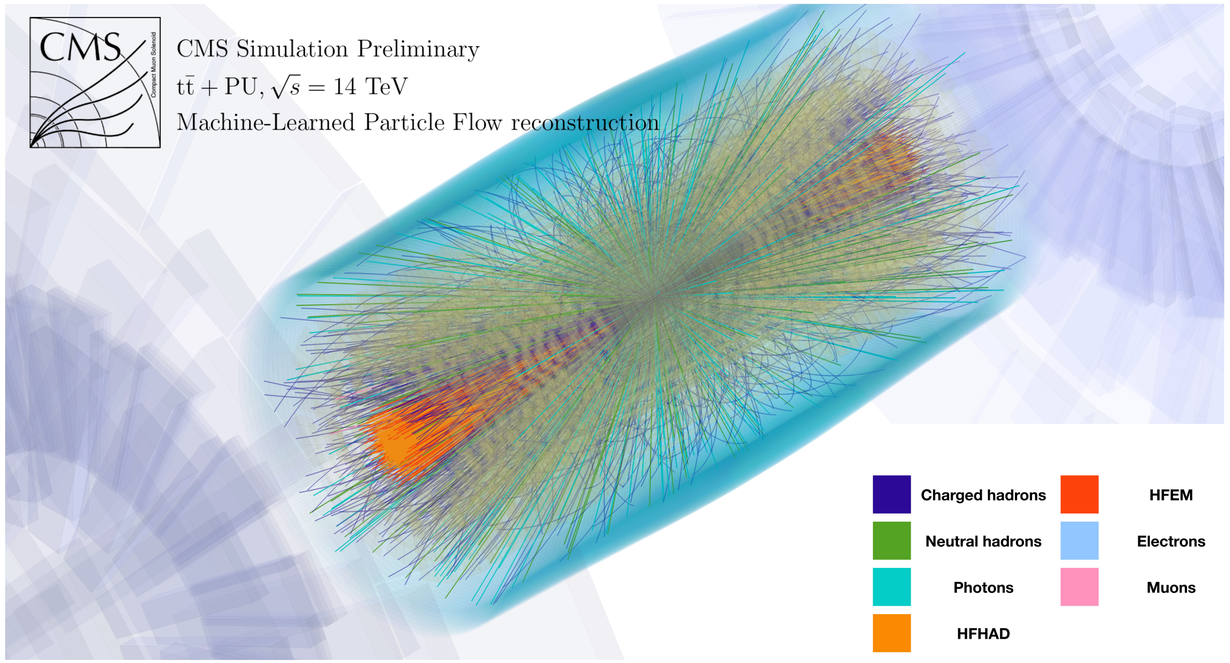


Event displays

One simulated ttbar event with pileup under Run3 conditions



Standard particle flow candidates

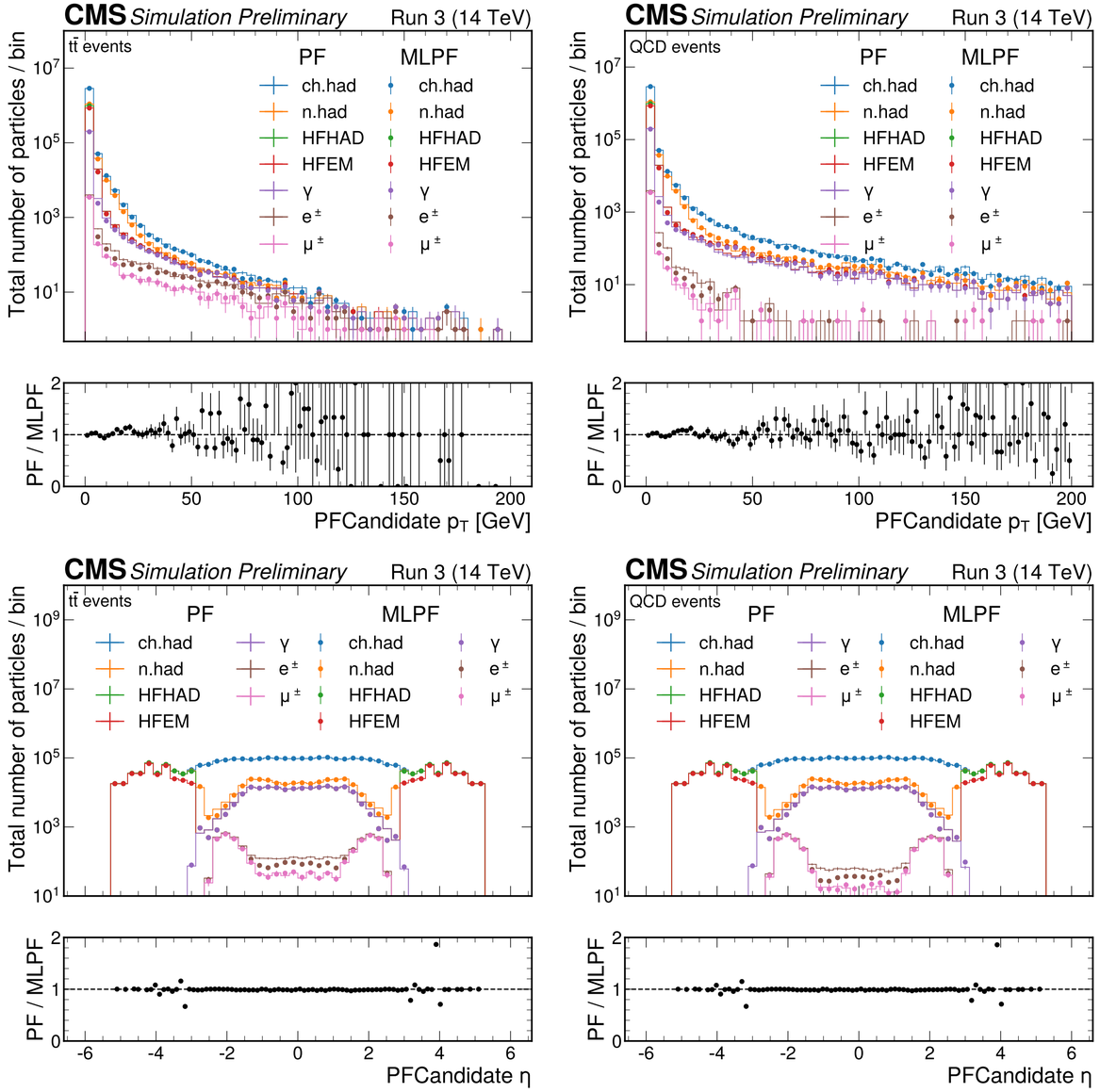


Machine-learned particle flow candidates

Particle candidates

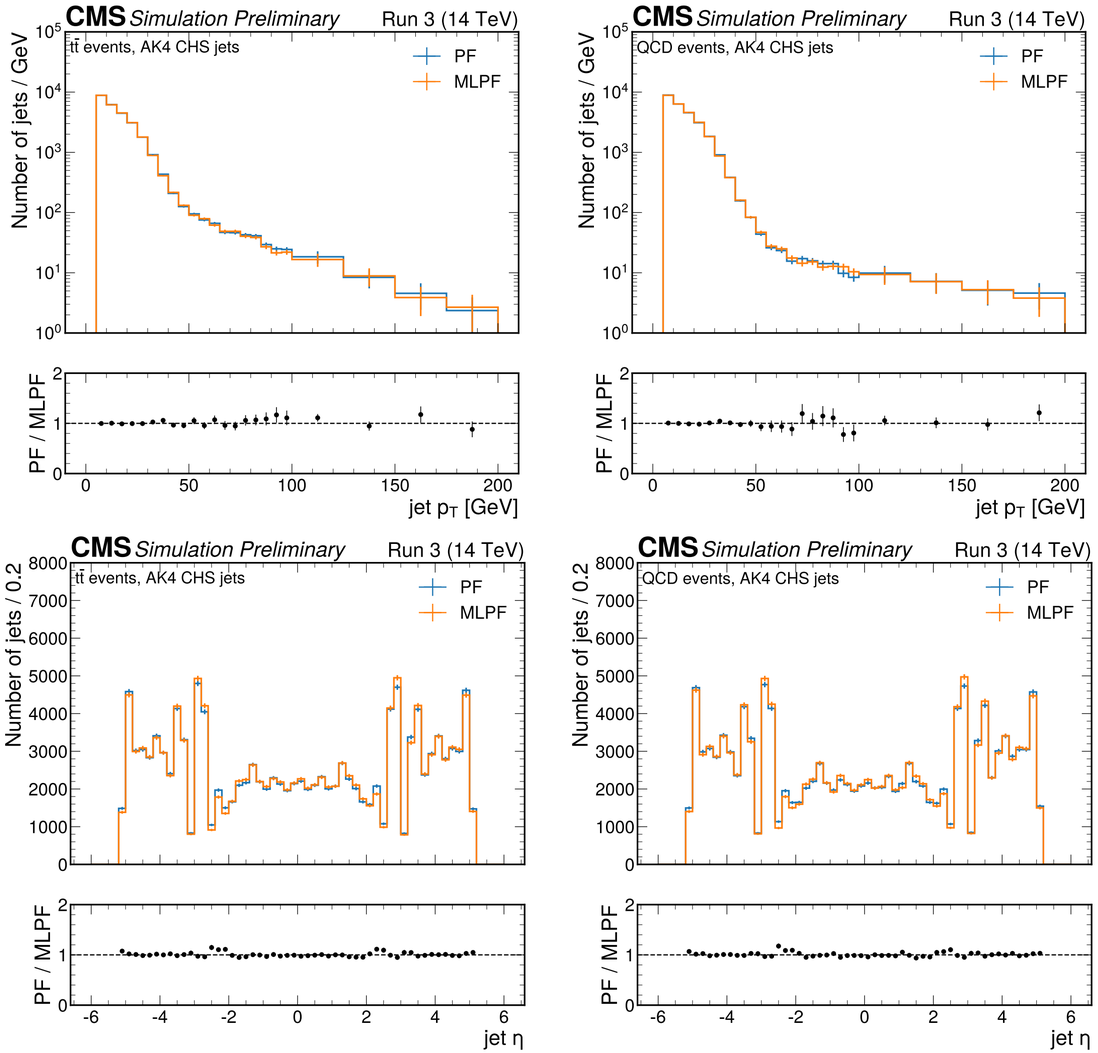
- An integration with full CMSSW offline reconstruction is possible and is undergoing tests
  - 1000 ttbar, QCD events with pileup, reconstruct with PF and MLPF
  - QCD was never used in training, **tests the generalization properties** of the model
- In general, **a good particle-level correspondence** is observed between PF and MLPF particle candidates for both samples
  - A notable but important exception is **electrons: challenging EM showers, low training statistics**

ttbar | **QCD**



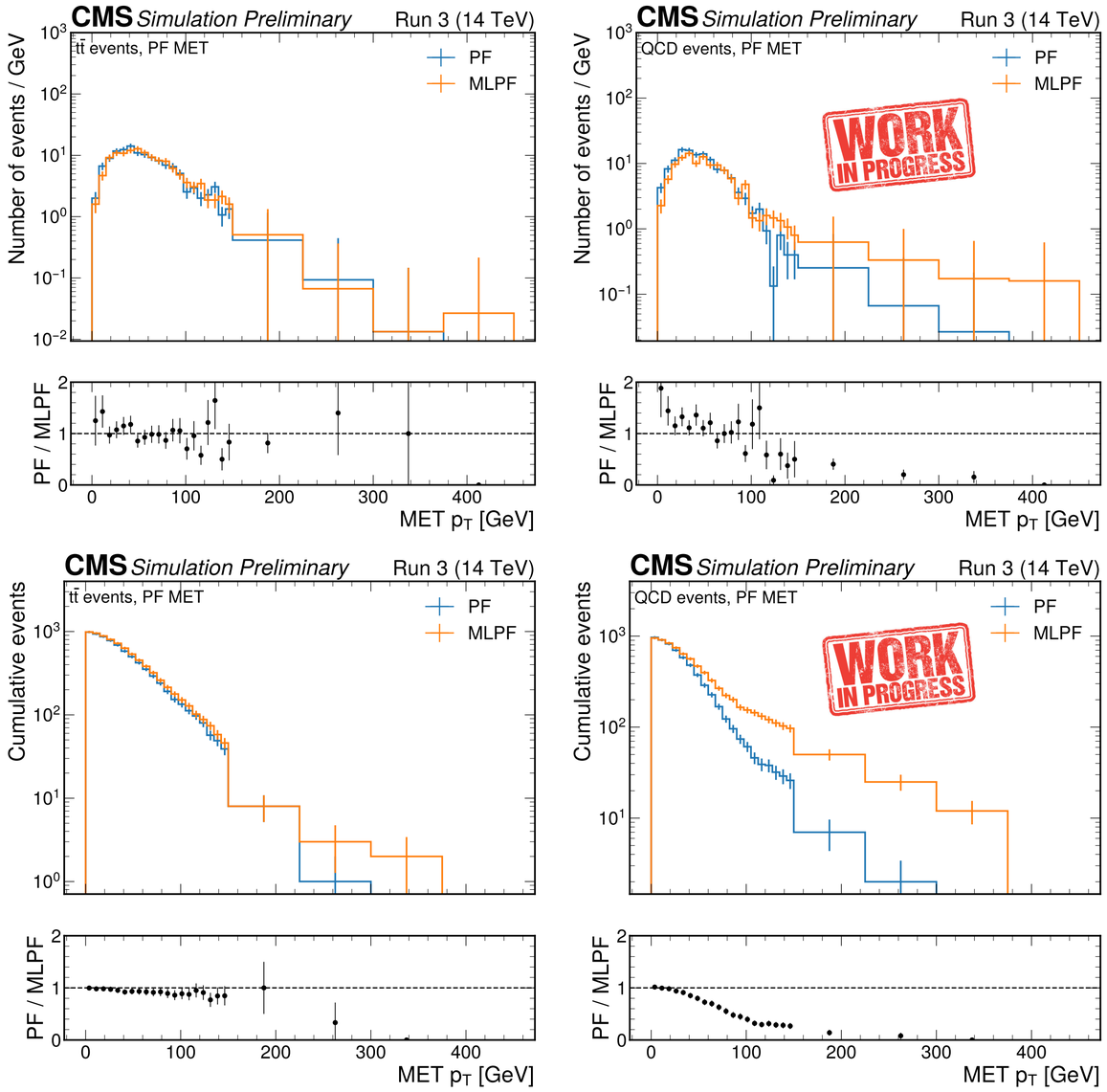
Jets

- Jets reconstructed based on the particle candidates, **charged pileup removed with charged hadron subtraction (CHS)**
- Generally, a good correspondence is observed between the jet kinematic distributions
- **Further checks are required** on jet resolution and response with respect to generator-level jets



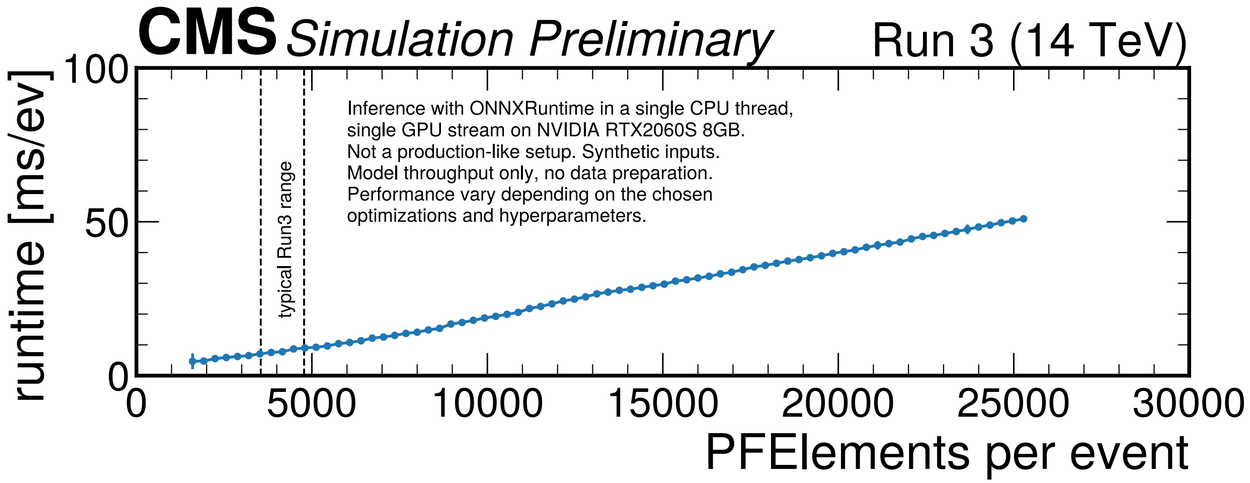
MET

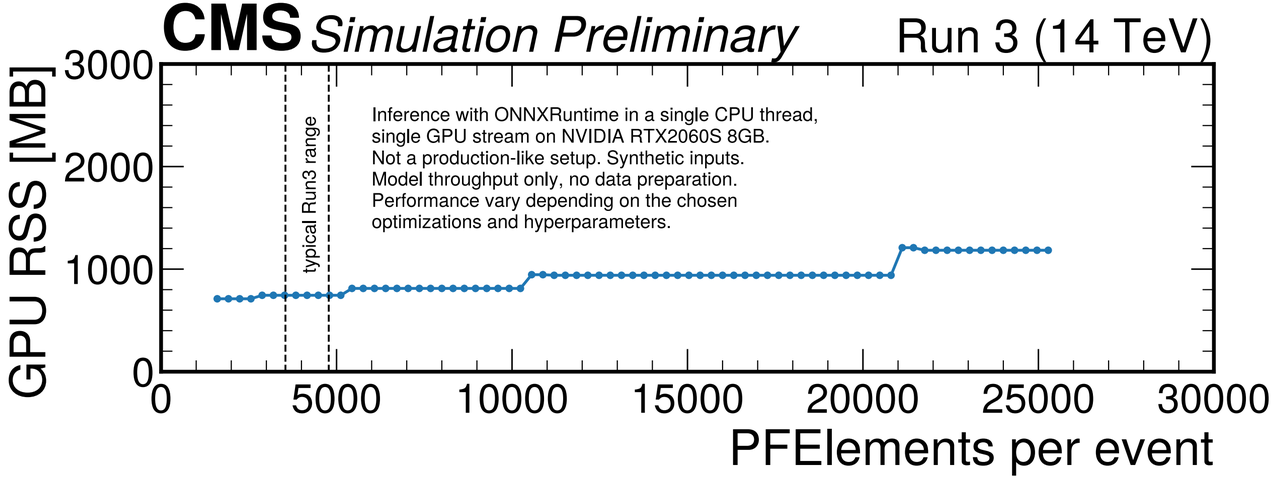
- Missing transverse energy reconstructed based on the particle candidates
- **In the ttbar sample, a good correspondence** is observed between PF and MLPF
- **In QCD, we observe high MET tails** misreconstructed by MLPF
  - Could possibly be mitigated by **a more extensive training** on a wider range of physics samples
- **Additional checks on resolution with respect to generator-level** truth are required



Computing performance

- This version of **MLPF requires approximately ~10 ms/ev, 800MB GPU memory** on a consumer-grade GPU (not a fully loaded production setup!)
  - For context, in a single CPU thread PF: ~90 ms/ev, MLPF: 320 ms/ev
- **Approximately linear scaling** in per-event runtime and memory usage ensures scalability towards Phase2 conditions



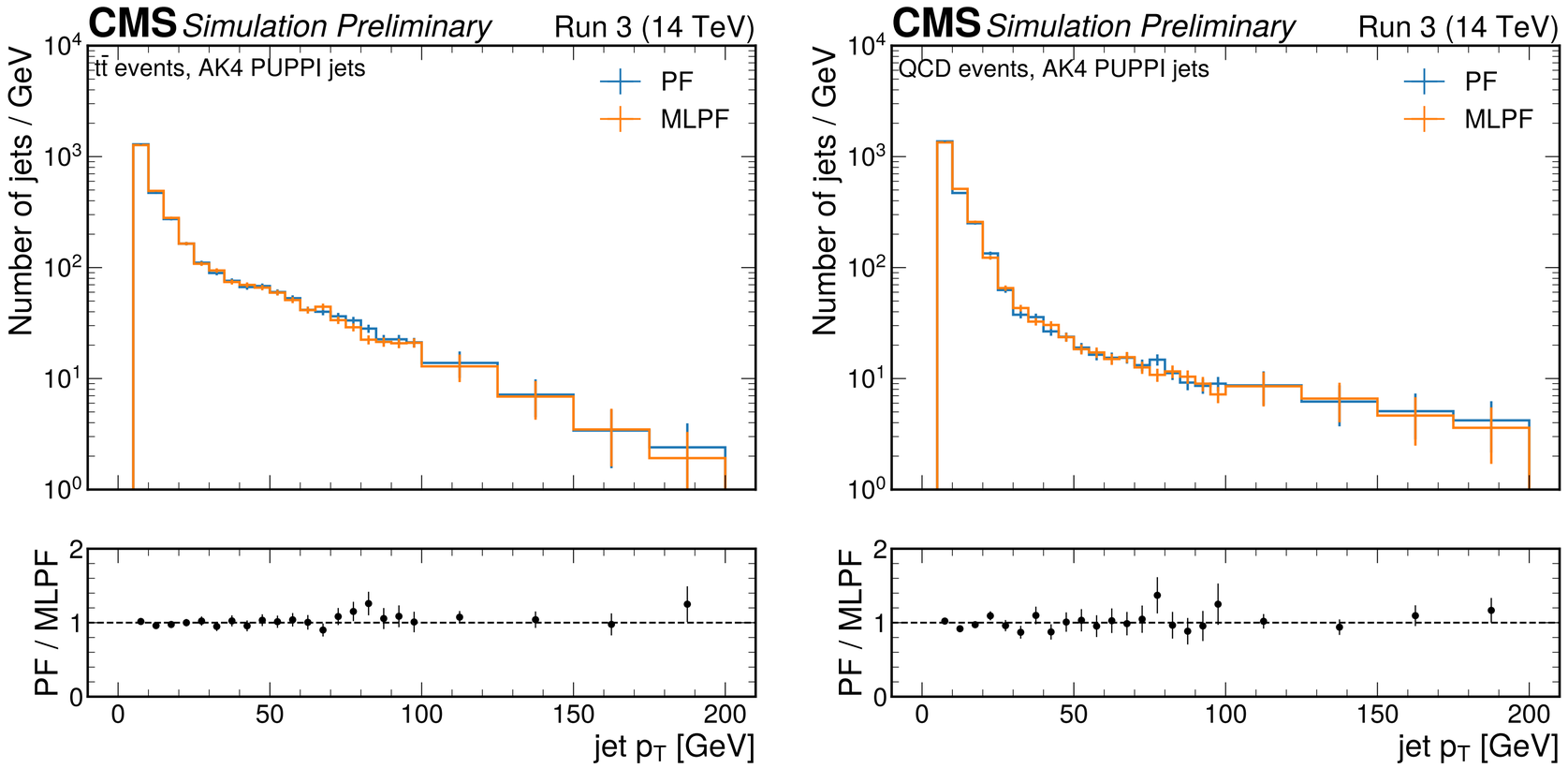


Summary

- We **have developed an ML-based method for full event particle flow reconstruction**
  - Builds **on top of the concurrent developments in CMS**, such as porting calorimeter clustering to GPUs, see e.g. [GPU accelerated algorithms for CMS PF]
- **Comparable physics performance with the baseline particle flow** is observed for particle candidates and jets, but **additional work is needed to improve generalization** (e.g. MET tails)
- The algorithm is **GPU-native and has approximately linear complexity** with the number of detector hits
- **Integration with offline reconstruction is underway** to carry out further validations during Run 3, towards Phase 2
  - Currently not enabled by default, but a **promising R&D for scalable, flexible full-event reconstruction**

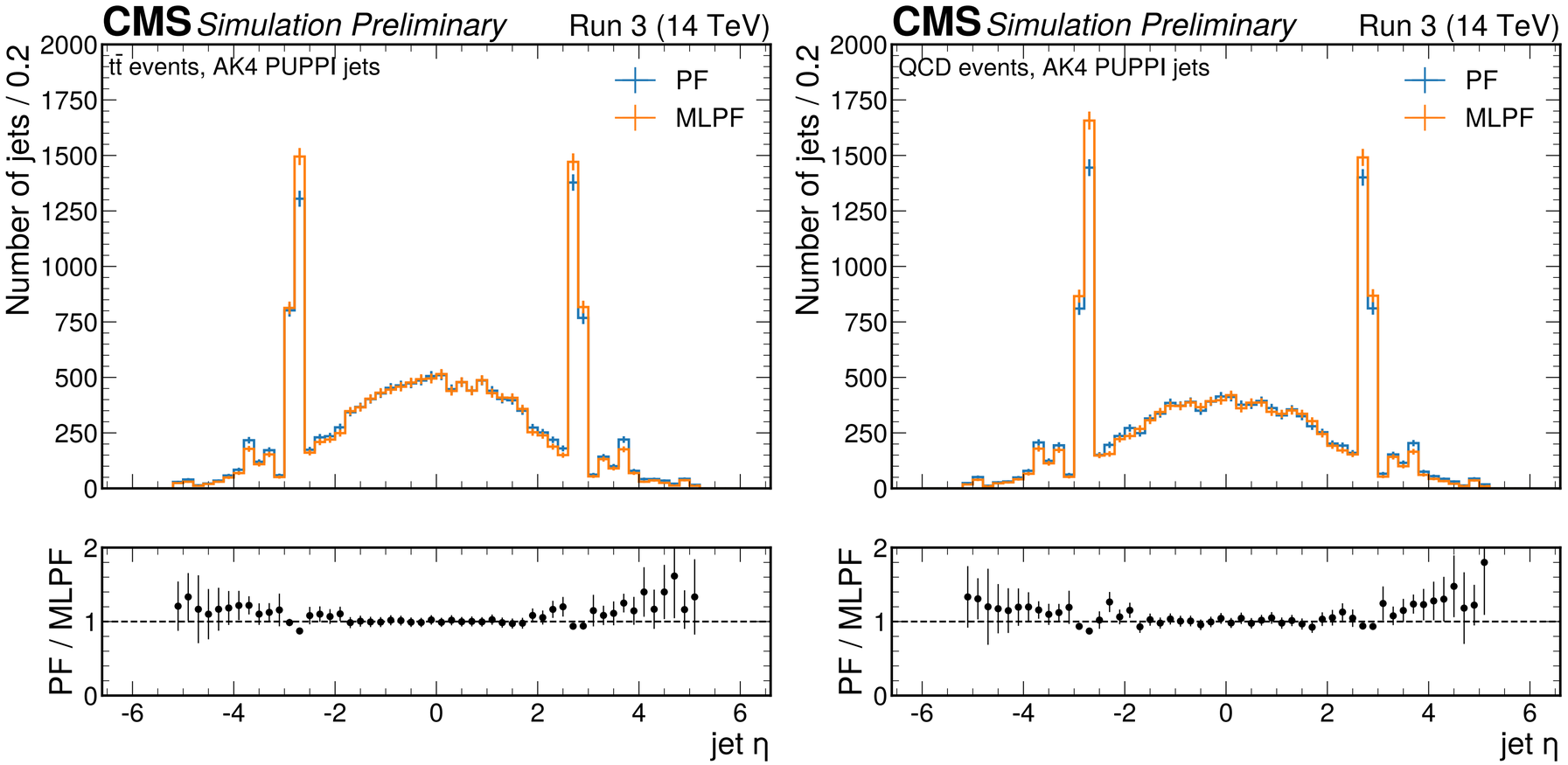
Backup

PUPPI jets from PF and MLPF



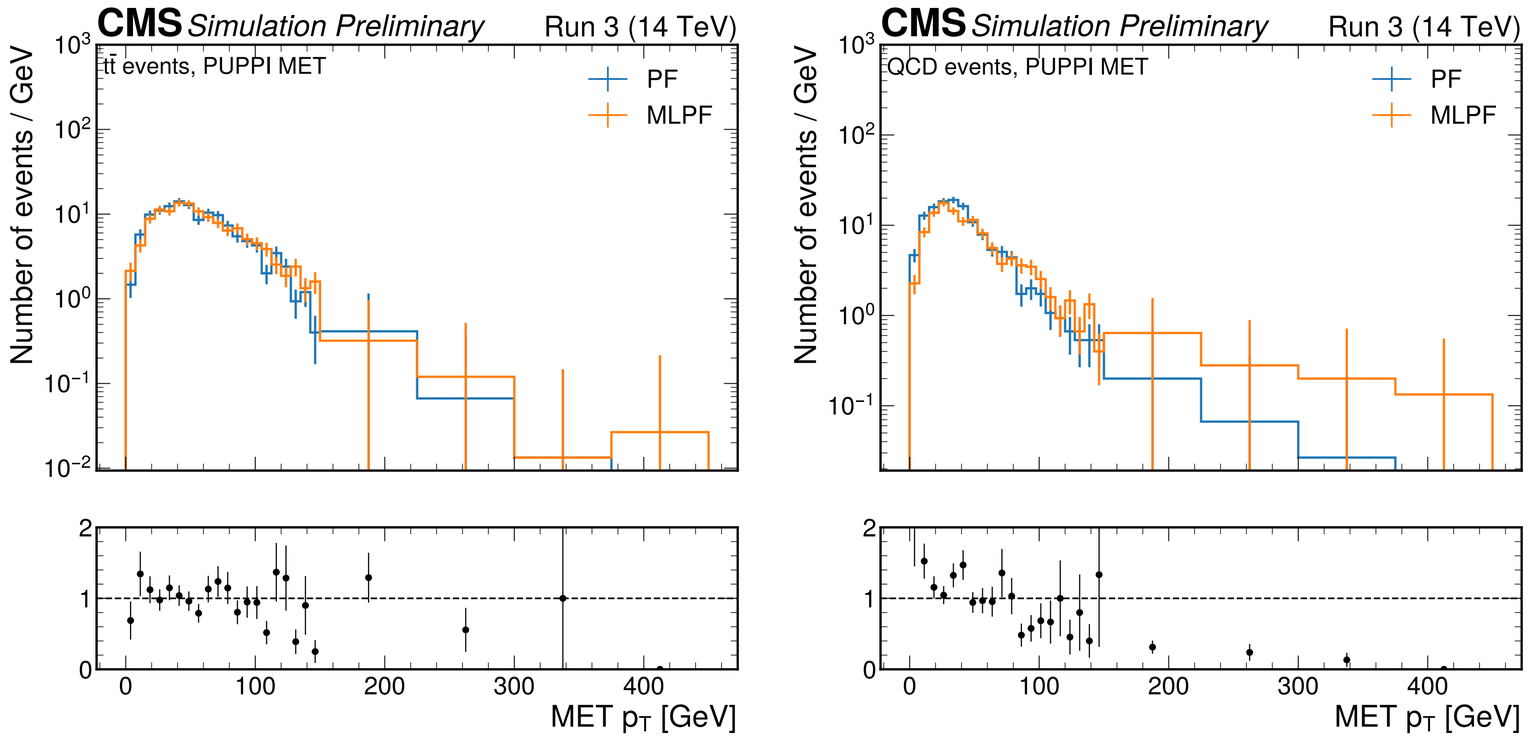
Jet kinematic distributions from CMSSW reconstruction, shown for 1000 Run 3 ttbar events (left) and QCD events (right). We compare the distributions as reconstructed by standard PF to the ones reconstructed by machine-learned particle flow. Exactly the same events were used for both algorithms, thus the statistical errors are correlated. The PUPPI algorithm was used to remove pileup contributions from the jets in both cases.

PUPPI jets from PF and MLPF



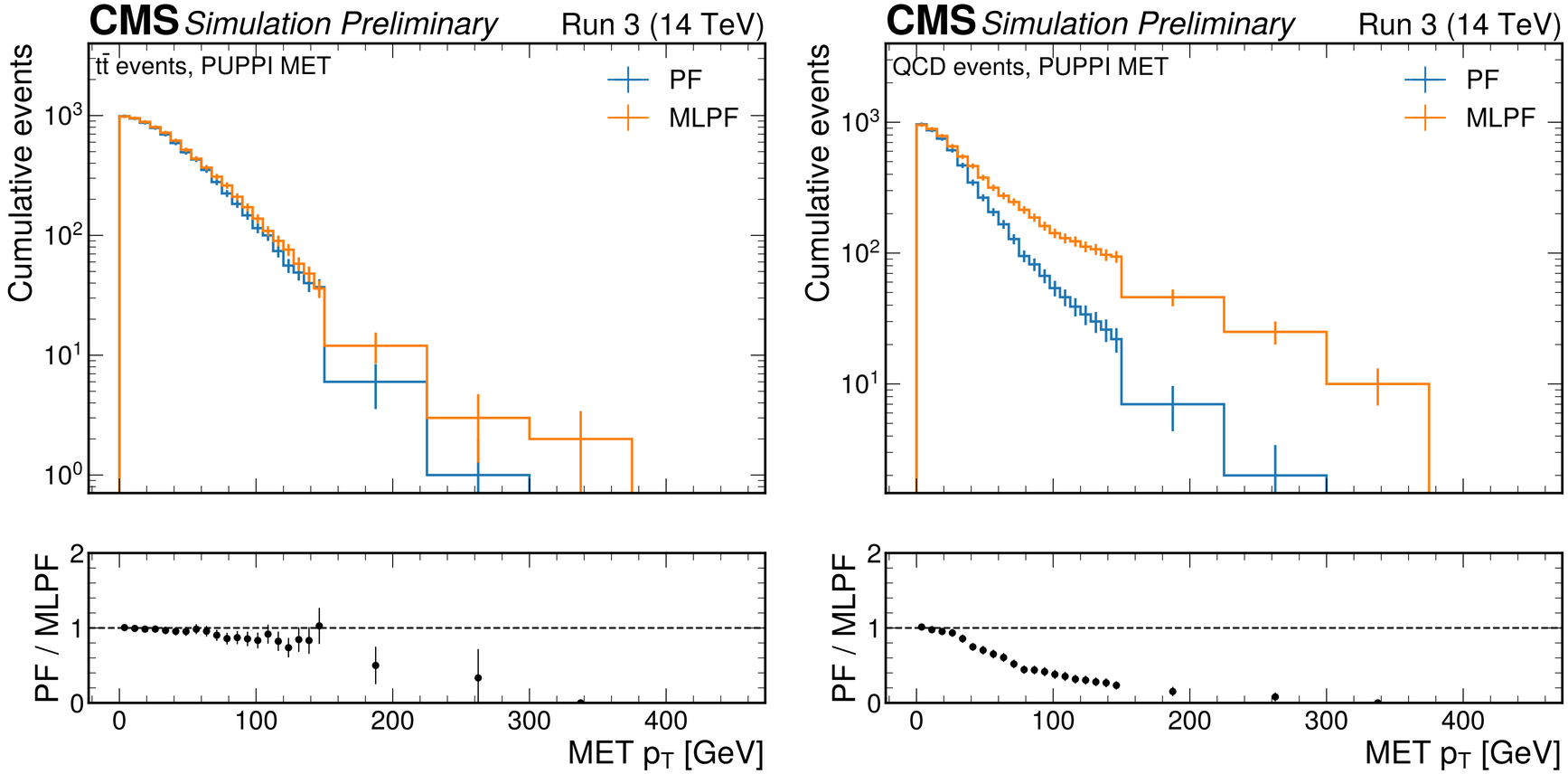
Jet kinematic distributions from CMSSW reconstruction, shown for 1000 Run 3 ttbar events (left) and QCD events (right). We compare the distributions as reconstructed by standard PF to the ones reconstructed by machine-learned particle flow. Exactly the same events were used for both algorithms, thus the statistical errors are correlated. The PUPPI algorithm was used to remove pileup contributions from the jets in both cases.

Puppi MET from PF and MLPF



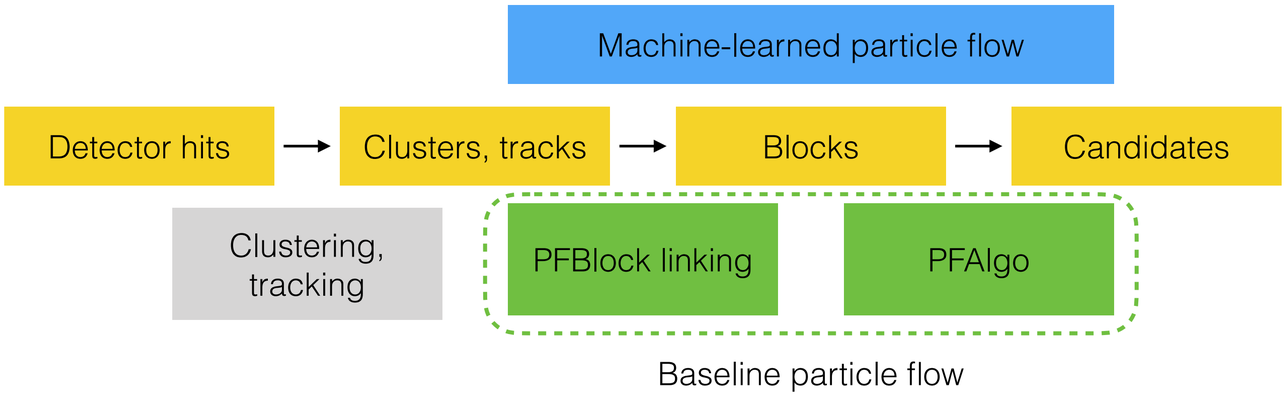
The missing transverse energy (MET) kinematic distributions from CMSSW reconstruction, shown for 1000 Run 3 ttbar events (left) and QCD events (right). We compare the distributions as reconstructed by standard PF to the ones reconstructed by machine-learned particle flow. Exactly the same events were used for both algorithms, thus the statistical errors are correlated. The MET was reconstructed based on PUPPI inputs. We observe a misreconstructed high-MET tail in the QCD sample that was not used in training. This may possibly be mitigated with a per-event loss component, additional training samples and is left for a future study.

Cumulative PUPPI MET

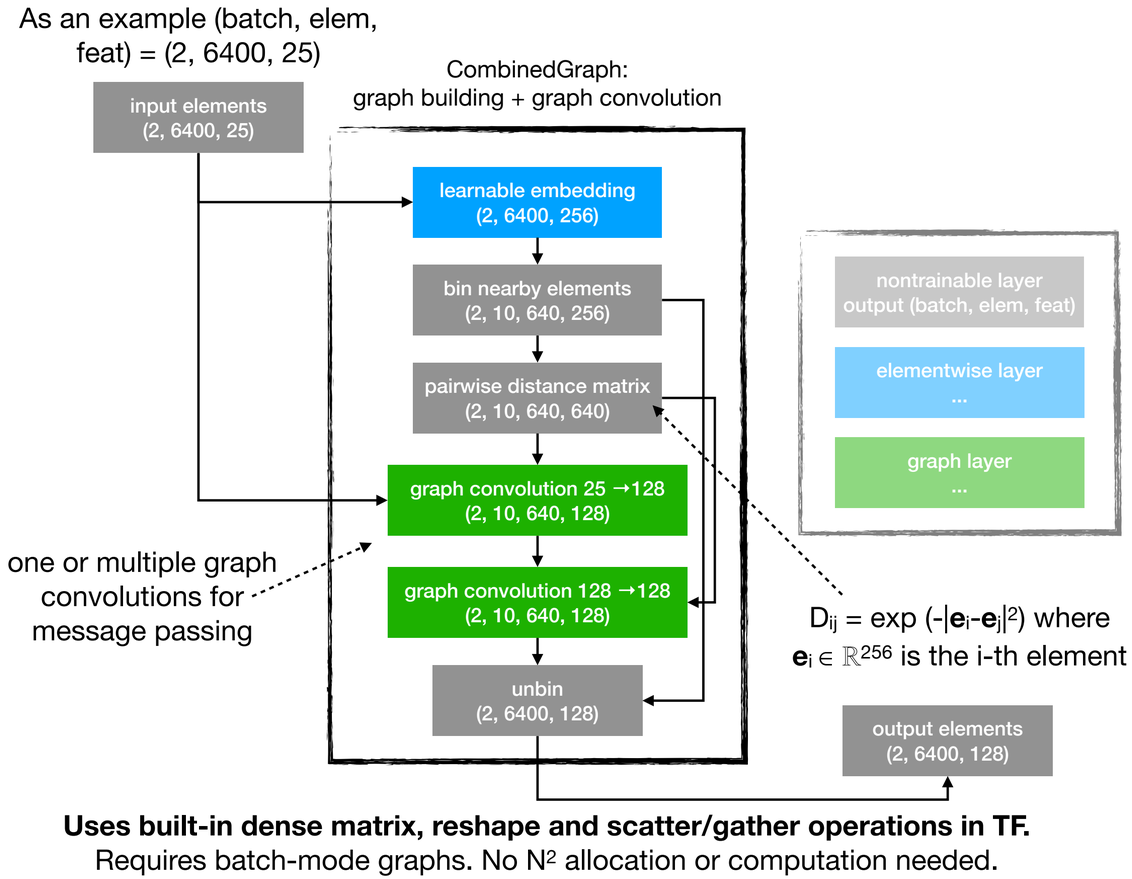


The cumulative missing transverse energy (MET) kinematic distributions from CMSSW reconstruction, shown for 1000 Run 3 ttbar events (left) and QCD events (right). We compare the distributions as reconstructed by standard PF to the ones reconstructed by machine-learned particle flow. Exactly the same events were used for both algorithms, thus the statistical errors are correlated. The MET was reconstructed based on PUPPI inputs. We observe a misreconstructed high-MET tail in the QCD sample that was not used in training. This may possibly be mitigated with a per-event loss component, additional training samples and is left for a future study.

MLPF integration with PF



Scalable CombinedGraph layer



One scalable combined graph layer. The input elements are projected into a learnable embedding space. Nearby elements in the embedding space are binned to fixed-size bins. A fully-connected graph is built in each bin, which is used for one or multiple graph convolutions that are used to transform the input elements. Finally, the transformed elements are unbinned.

Model inputs and outputs

The input features of the `PFElements` are as follows:

- ECAL, HCAL, HF calorimeter clusters: cluster energy, corrected energy, η, ϕ, x, y, z position; number of hits; layer; depth; cluster flags
- ECAL supercluster: cluster energy, η, ϕ, x, y, z position; number of hits
- KF tracks: p_T, η, ϕ, p_x, p_y, p_z, $|p|$ at the vertex; η, ϕ extrapolated to the ECAL shower max and HCAL entrance; number of hits; track charge; muon ref type, number of DT, CSC hits
- GSF tracks: p_T, η, ϕ, p_x, p_y, p_z, E at the inner point; η, ϕ at the outer point; track charge; number of hits; a flag to denote if the electron seed is ECAL or tracker driven.
- BREM points: index of the trajectory point, ΔP, $\sigma(\Delta P)$

The input features for all input elements in an event are concatenated to a per-event feature matrix X.

The target particles y_i are described by a feature vector

$$y_i = [\mathrm{ID}, p_\mathrm{T}, \eta, \phi, E, q]$$

Loss function

The total loss function for one event is then:

$$L(Y, Y') = \sum_k \text{Focal}(y_k, y'_k) + \text{Huber}(y_k, y'_k),\ k = 1 \ldots |X| \tag{1}$$

where we use the focal loss [38] and Huber loss [39] for classification and momentum regression respectively. The focal loss is a modification of regular cross entropy loss, originally proposed for the application of object detection in Ref. [38], and is defined according to

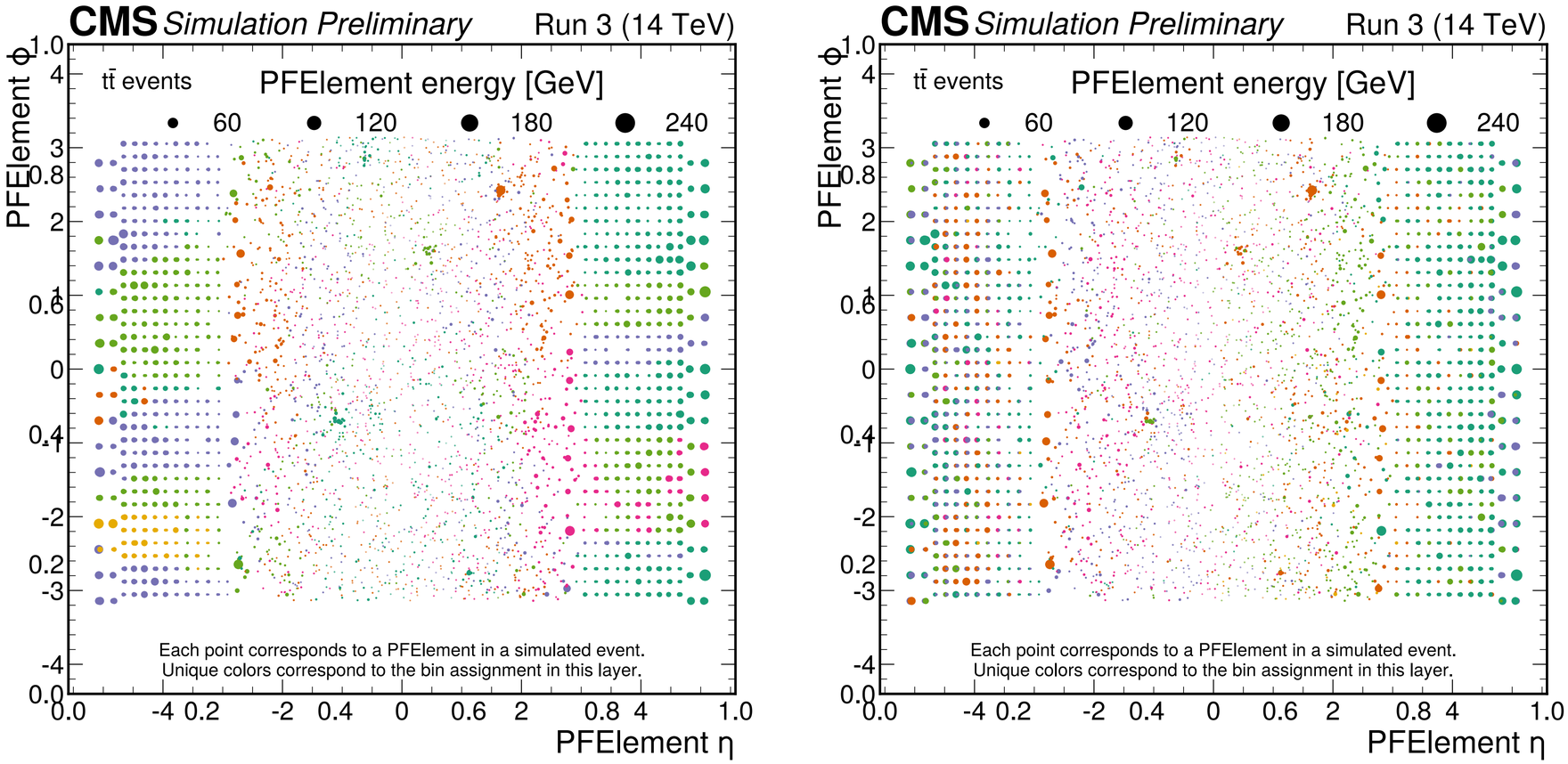
$$\text{Focal}(y_k, y'_k) = \begin{cases} -\alpha(1-p)^\gamma \log(p) & \text{if } y'_k = 1 \\ -(1-\alpha)p^\gamma \log(1-p) & \text{otherwise} \end{cases} \tag{2}$$

where the loss is computed on the one-hot encoded part of y'_k and the models corresponding prediction, $p \in [0, 1]$ is the model's estimated probability for the correct class and α is a weighting factor. The Huber loss, designed for regression tasks, is defined as

$$\text{Huber}(y_k, y'_k) = \begin{cases} \frac{1}{2}(y'_k - y_k)^2 & \text{for } (y'_k - y_k) \leq \delta \\ \delta(|(y'_k - y_k)| - \frac{1}{2}\delta) & \text{otherwise.} \end{cases} \tag{3}$$

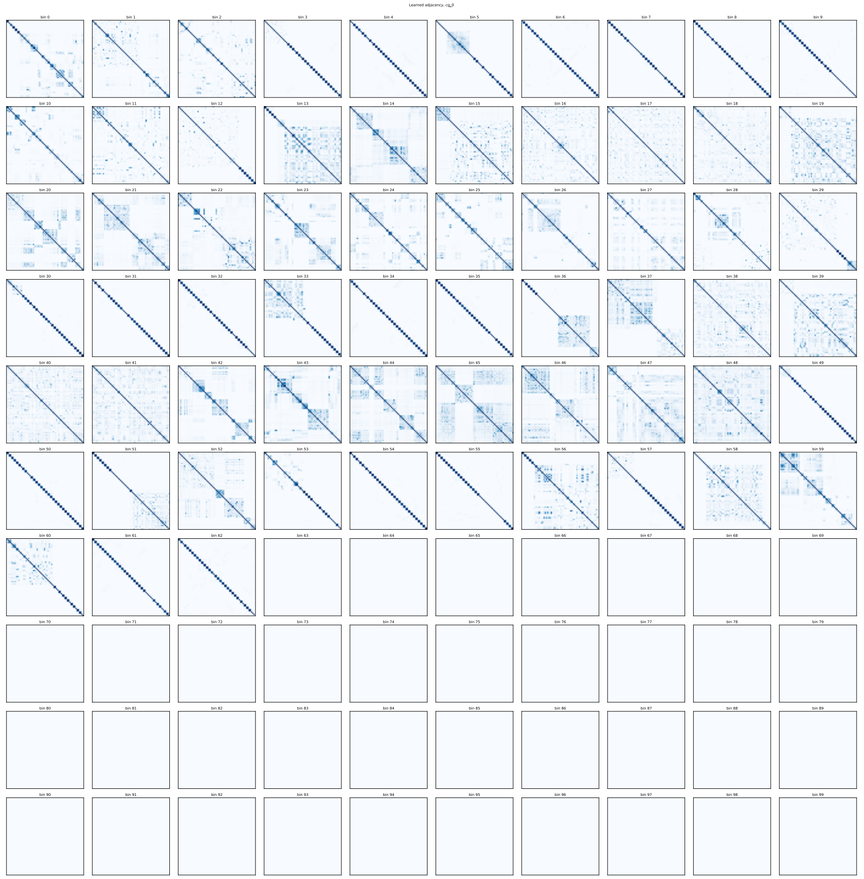
The per-event loss function is computed as a sum over all particles in the event. Classification and regression are used to predict particle candidates along with their momenta. Focal loss is used for classification, whereas Huber loss is used for regression.

Learned binning



The learned binning structure in the first two layers of the model. We show one simulated ttbar event, with each point corresponding to a PFElement in the event. The colors correspond to the assignment of the PFElements into the bins in each layer.

Learned graph structure



The learned graph structure in the first layer of the model. Each plot corresponds to a bin where an all-to-all graph adjacency matrix is built between the PFElements in the bin. Emtpy bins are present since for efficiency reasons in GPU training and inference, the model operates on a fixed-size zero-padded input.