Semi-supervised GraphNN for Pileup Noise Removal

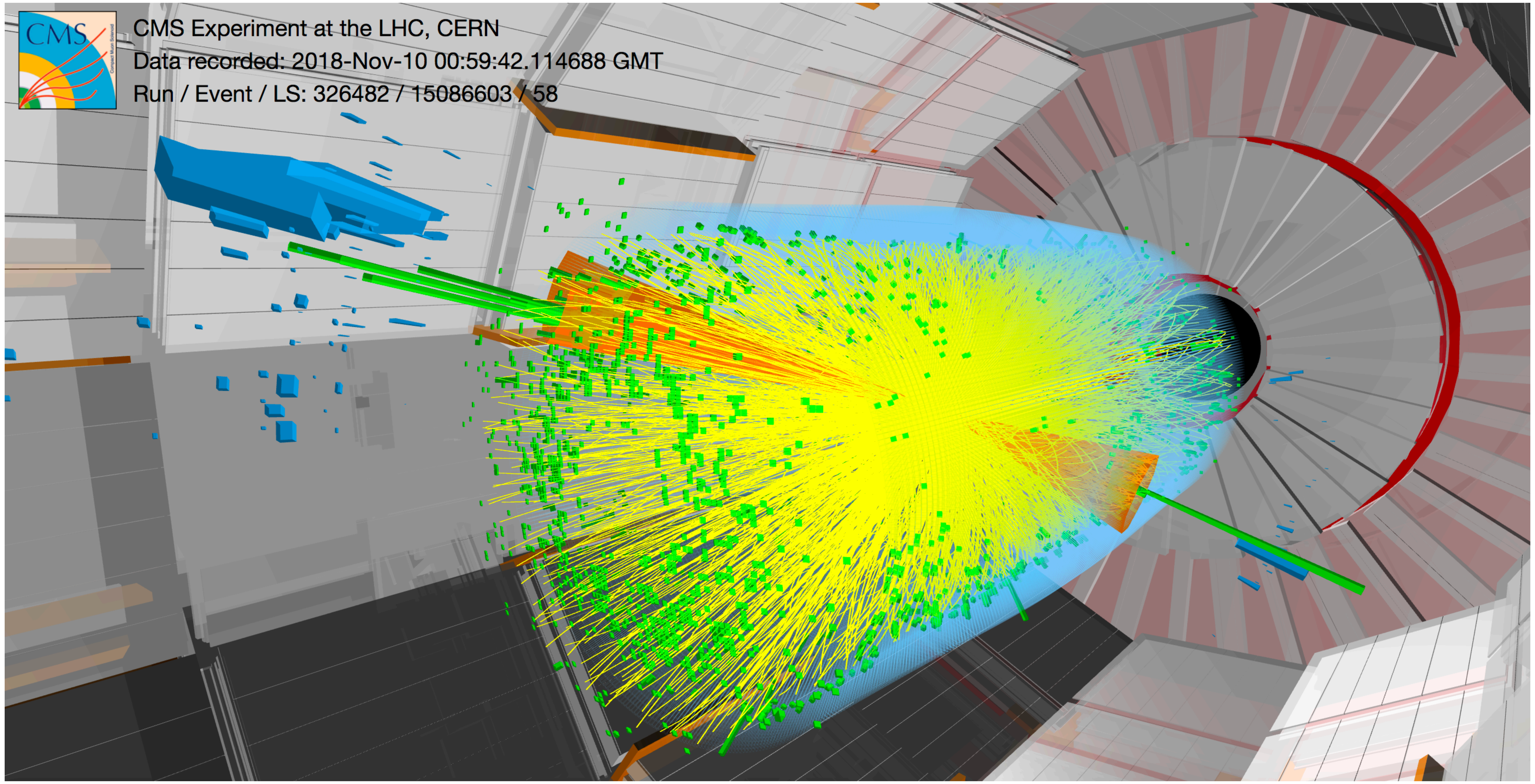
Shikun Liu, Tianchun Li, Pan Li, Miaoyuan Liu, Lisa Paspalaki (Purdue)

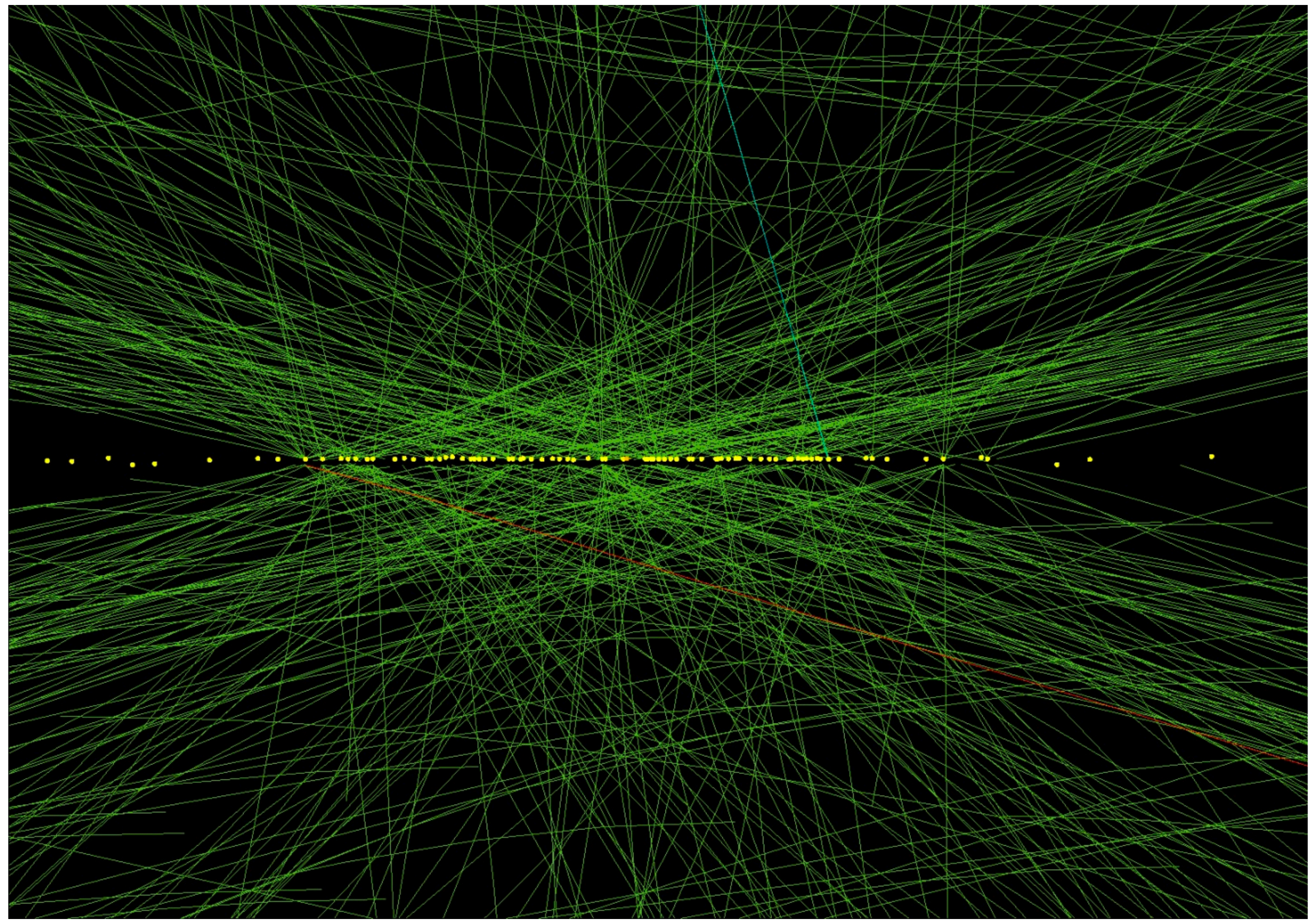
Yongbin Feng, Nhan Tran (Fermilab)

EPE Machine Learning

May 3rd, 2022

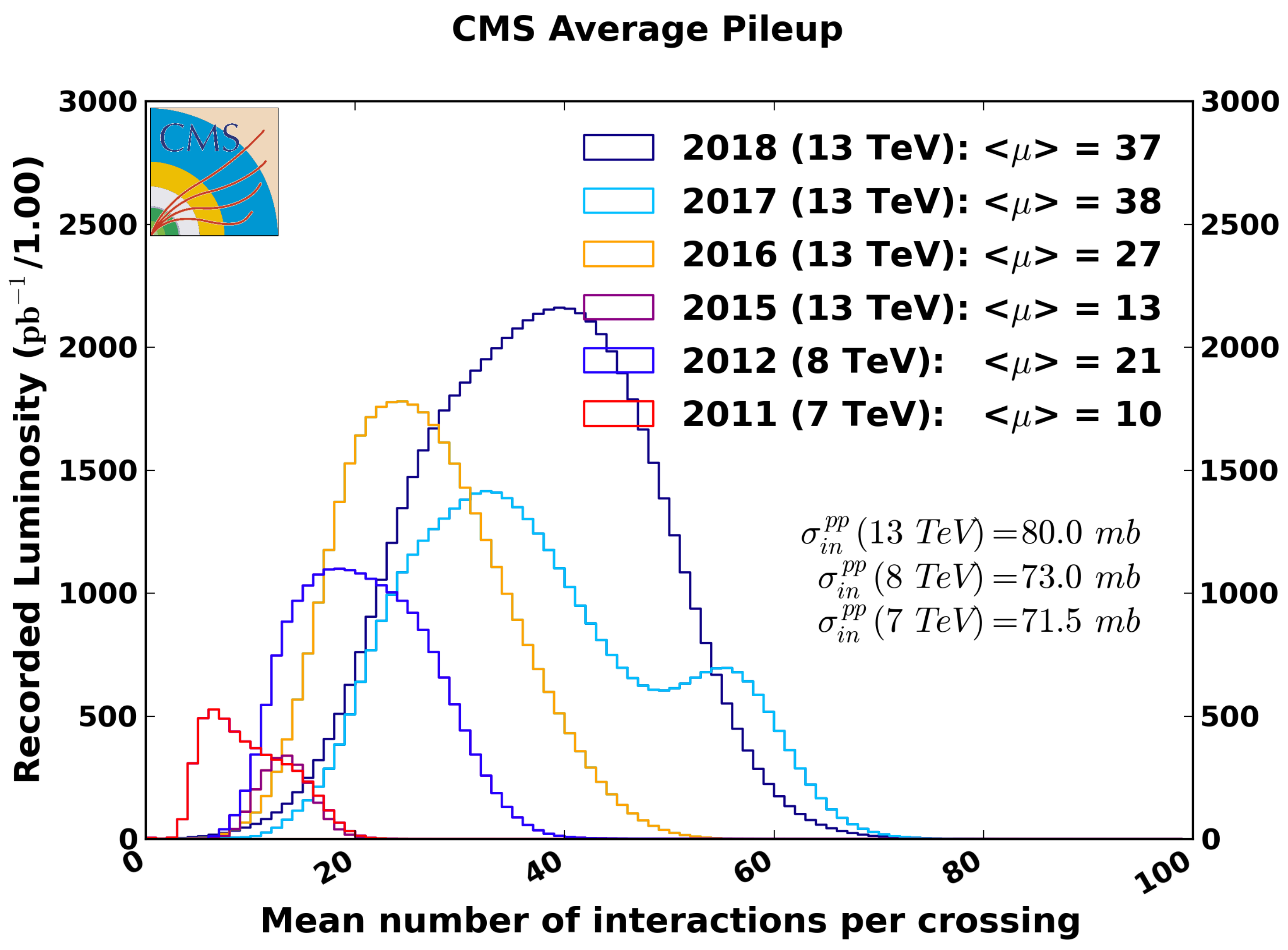
What is PileUp

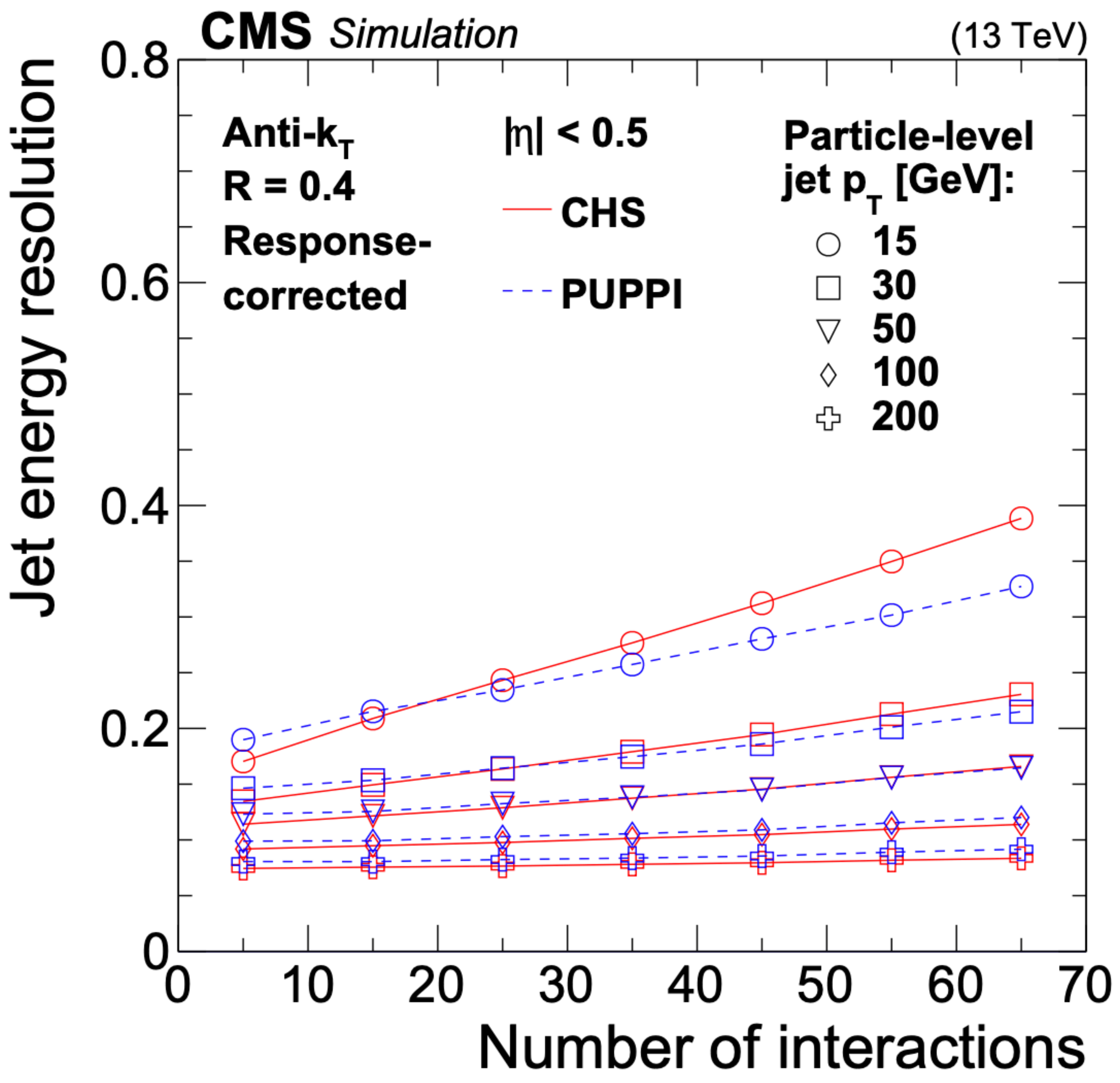


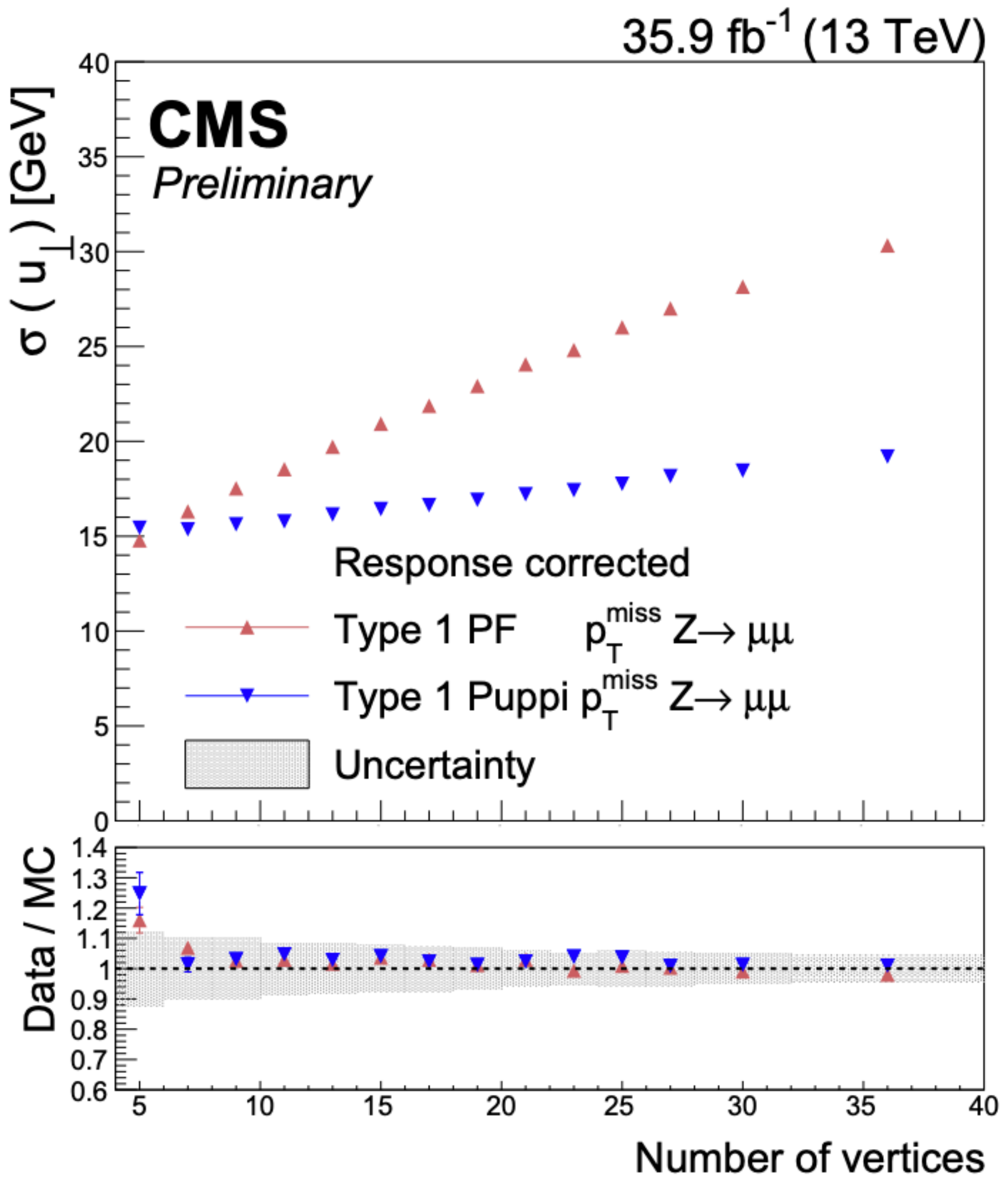


- Pileup (PU): additional proton-proton interactions in the same or nearby bunch crossings

Why PileUp Mitigation



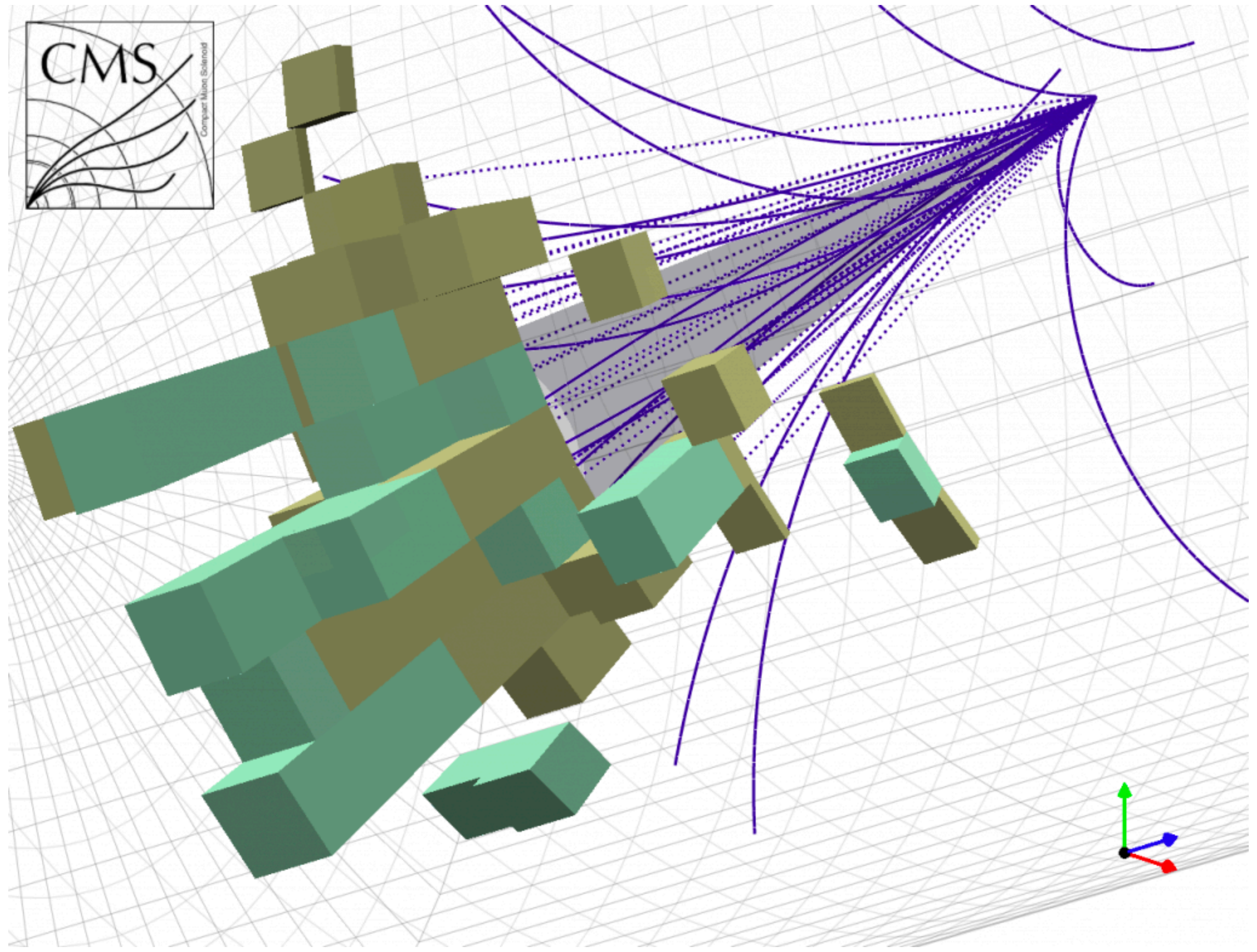


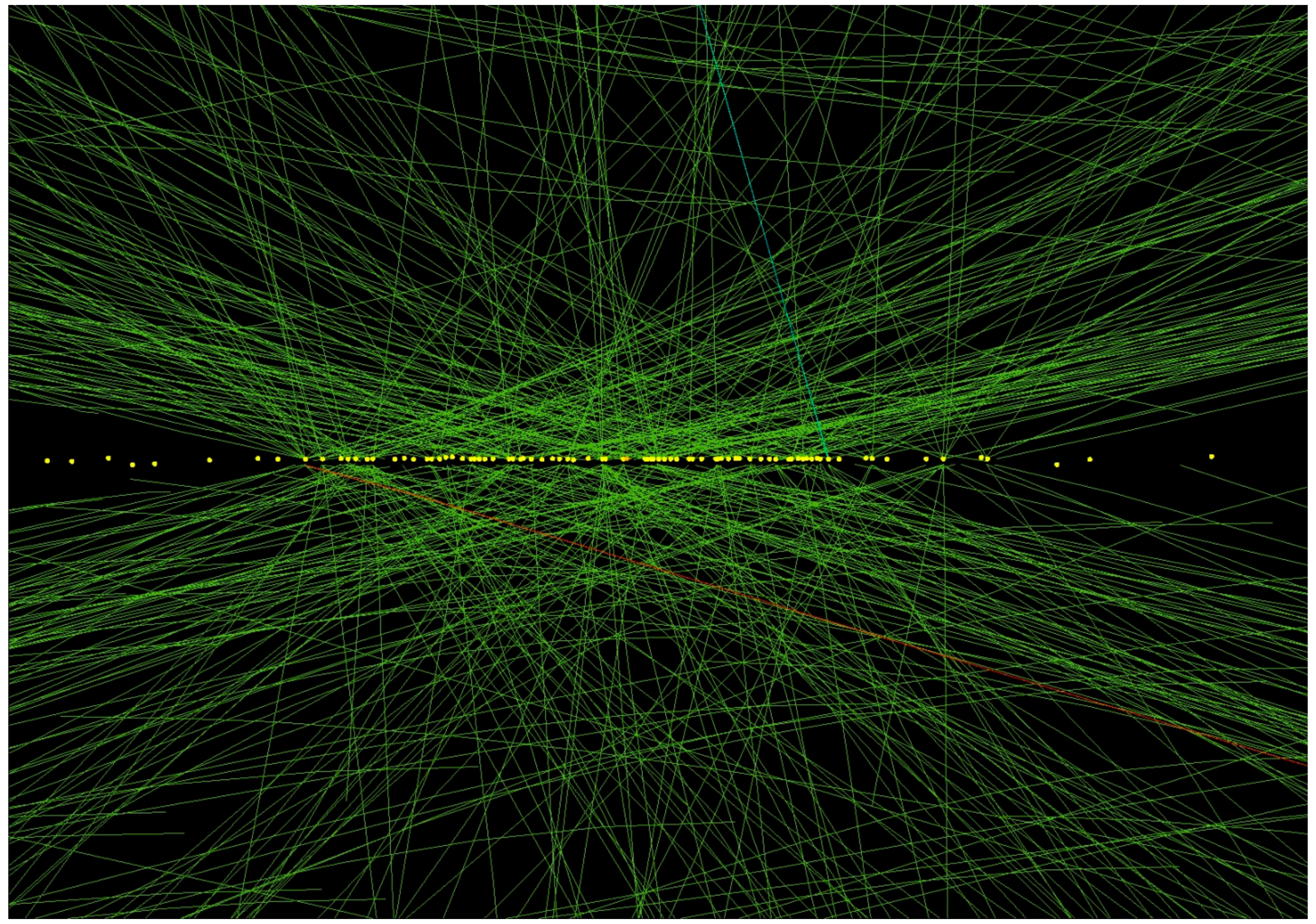


- PU at Run-II: ~30-40; expected to increase to 140-150 at HL-LHC
- PU can significantly affect the reconstruction and performance of many physics observables, such as jet mass, jet pT, and pTmiss
- PU mitigation is needed

PileUp Mitigation: How?

- Charged particles easy to deal with - Leading Vertex (LV) or Pileup (PU) charged particles can be easily identified because of excellent tracking and vertexing efficiency and

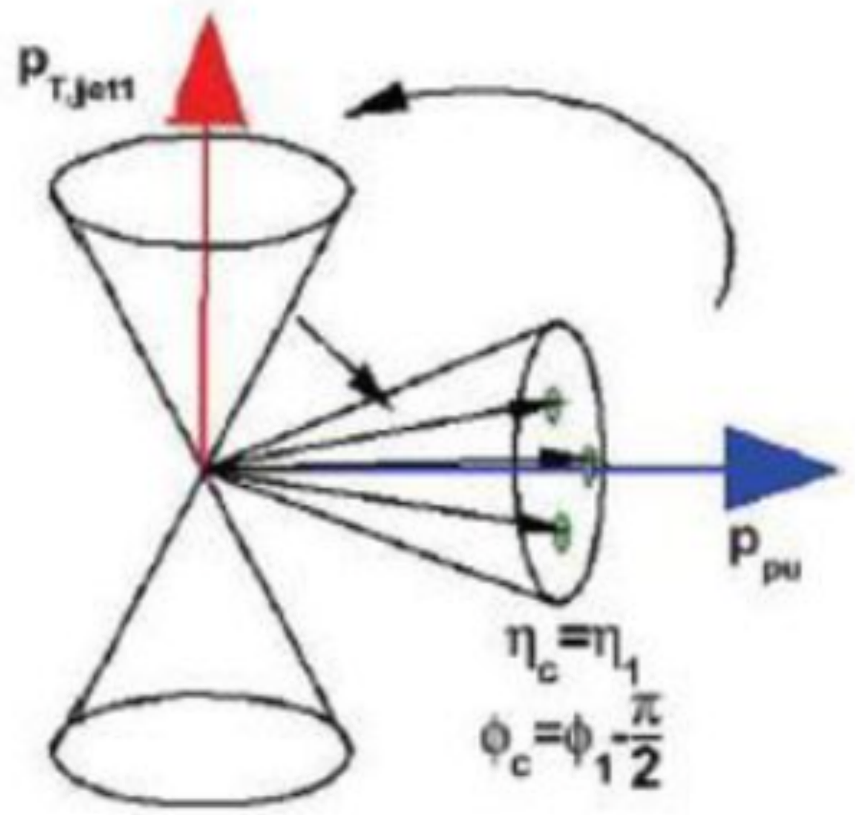




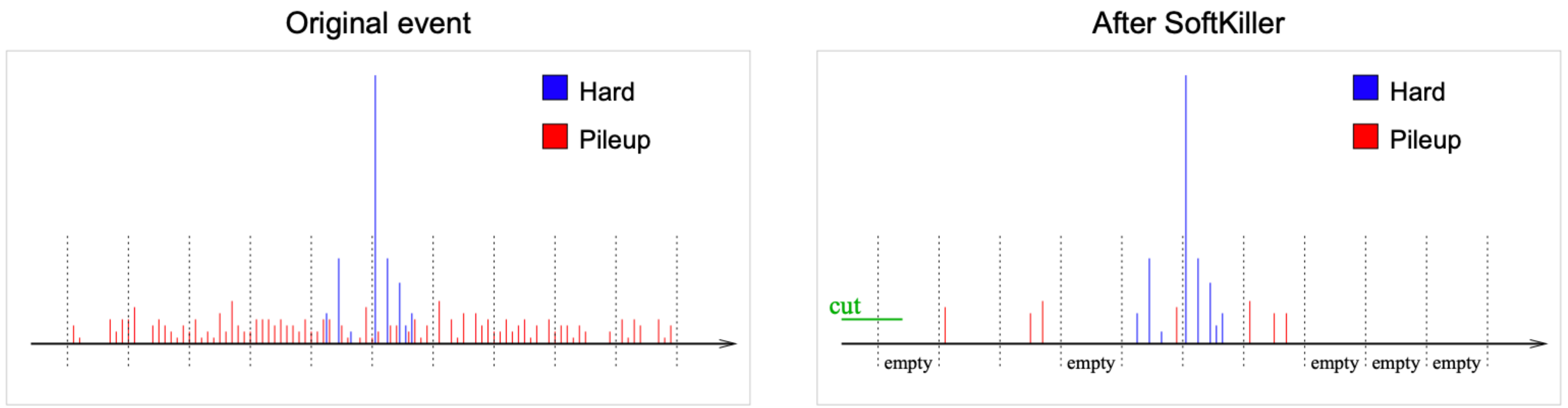
- Problem is how to identify pileup neutral particles and remove these

Classical PileUp Mitigation Techniques

- Run-I: Area-based pileup subtraction:
  - e.g., calculate the pileup energy density outside the jet cone; and use the average to correct jet energy



- Later on: Soft-Killer [Arxiv.1407.0408]
  - Pileup particles have lower pT; kill the pileup by removing "soft" particles
  - Calculate the median pT: $p_T^{\mathrm{cut}} = \mathrm{median}_{i \in \mathrm{patches}} p_{T,i}^{\max}$; cut on the median pT to remove pileup
  - pT is a particle's "self feature"; no strong connection with the other particles in the same event



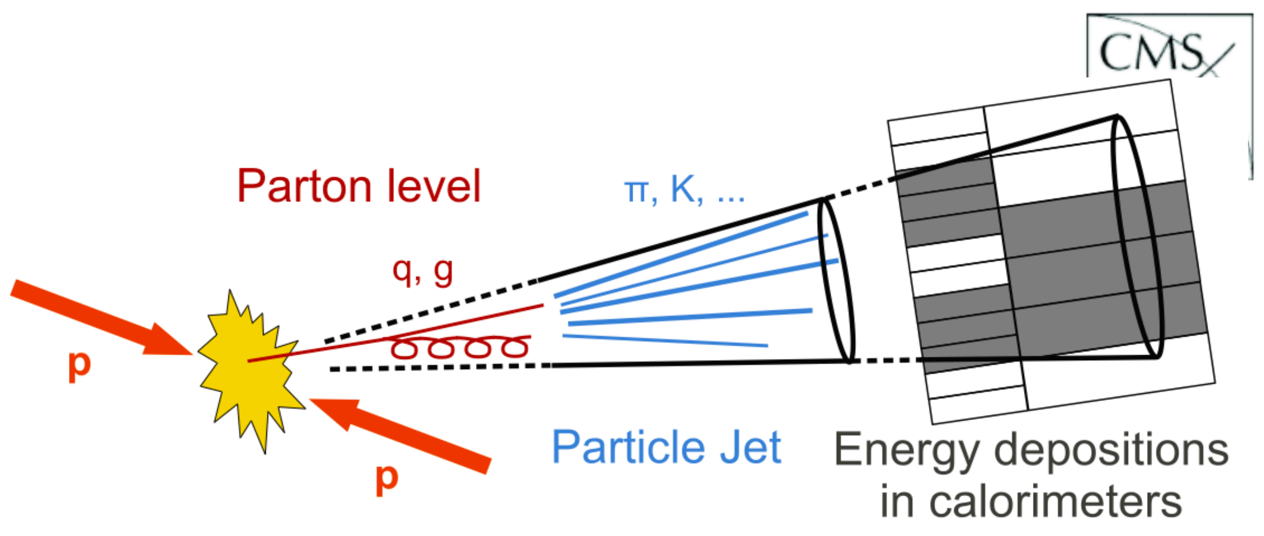
Classical PileUp Mitigation Techniques

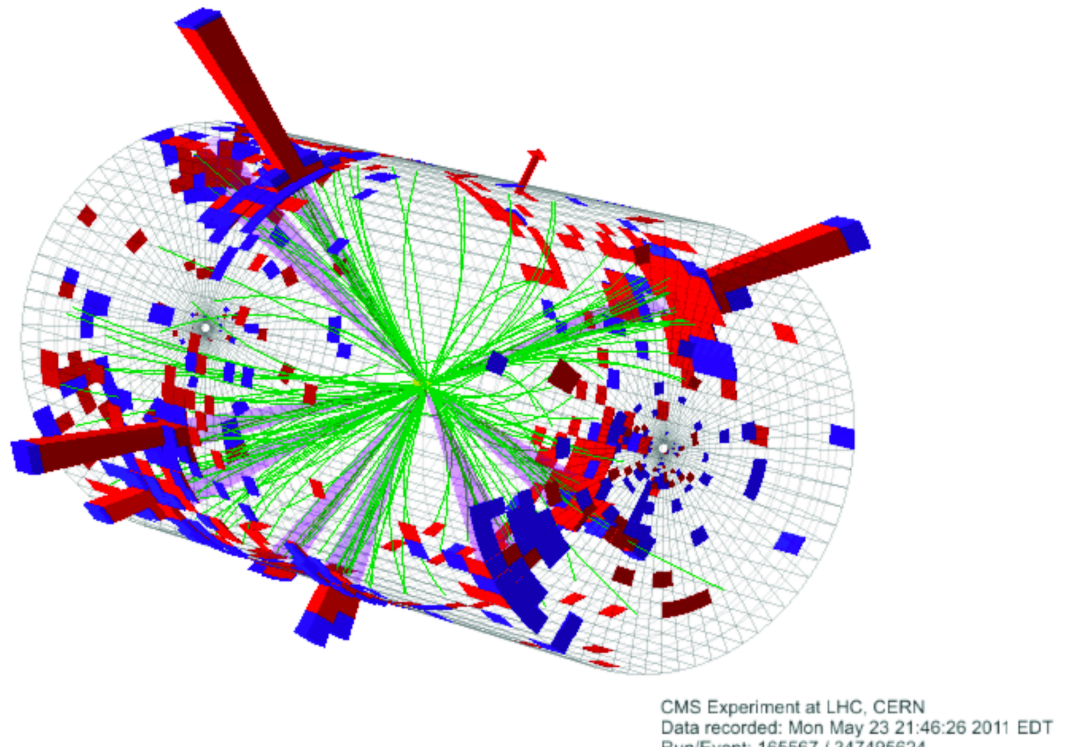
- PUPPI: [Arxiv:1407.6013]
  - Makes use of the neighboring particle features: LV particles are usually surrounded by LV particles; PU particles are more isotropic
  - Calculate a local shape variable alpha:

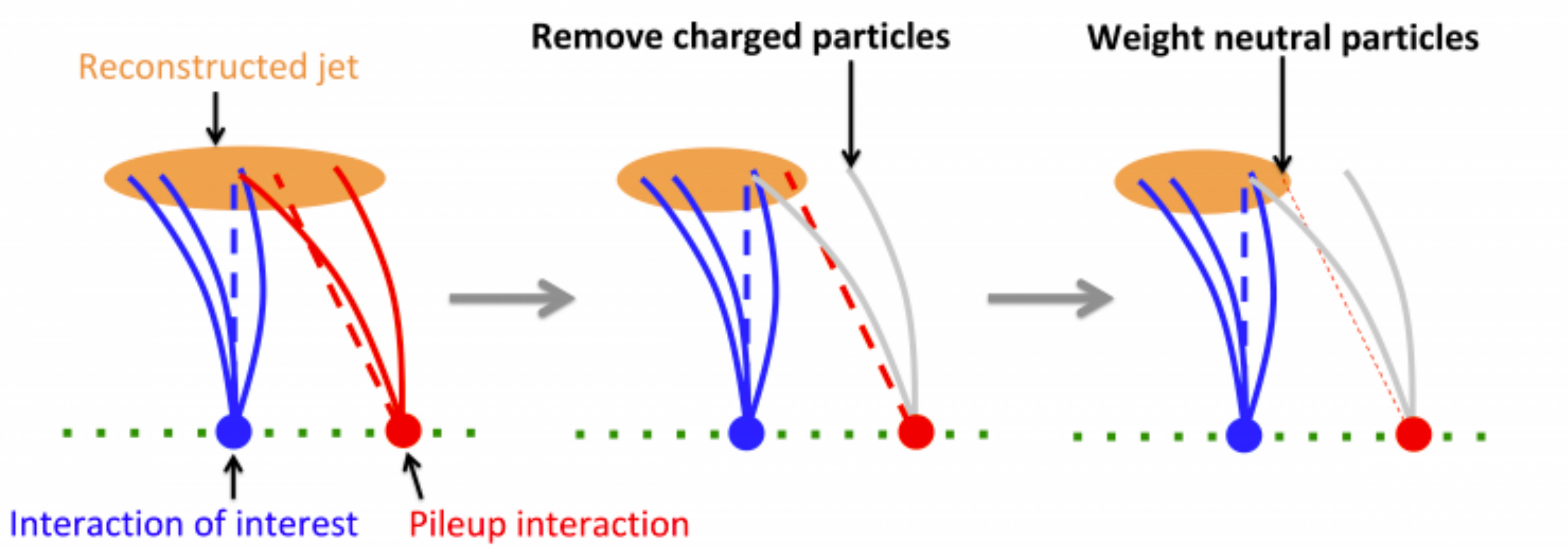
$$\alpha_i = \log \sum_{j \in \text{event}} \xi_{ij} \times \Theta(R_{\min} \leq \Delta R_{ij} \leq R_0),$$

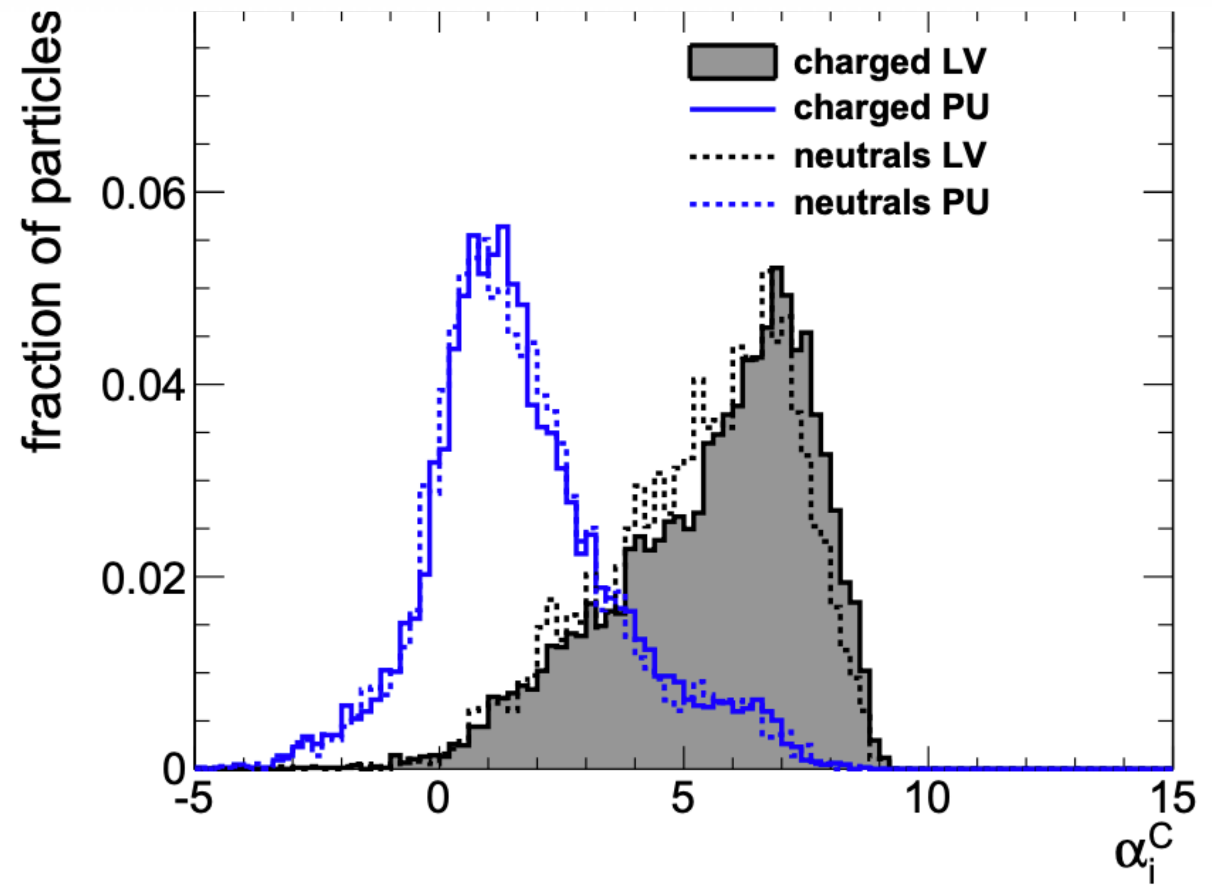
$$\text{where } \xi_{ij} = \frac{p_{Tj}}{\Delta R_{ij}}.$$

  - Alpha is aggregating information from the neighboring particles. e.g., aggregating ξ_{ij} only from the neighboring charged LV particles
  - Per-particle weight (PUPPI weight, in the range of 0-1) is calculated based on alpha; particle 4-momenta are rescaled based on the PUPPI weight









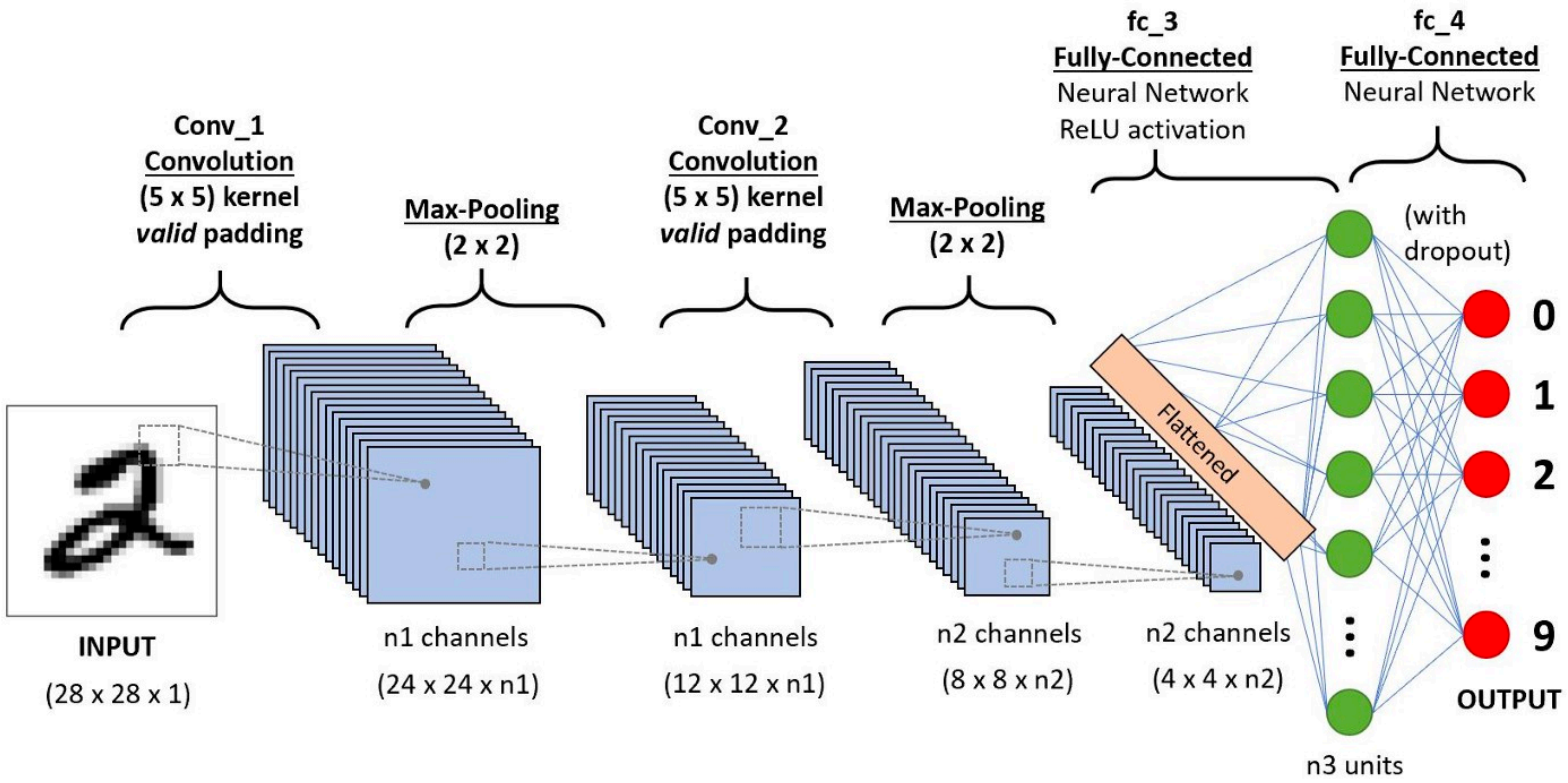
Learned From Classical Techniques

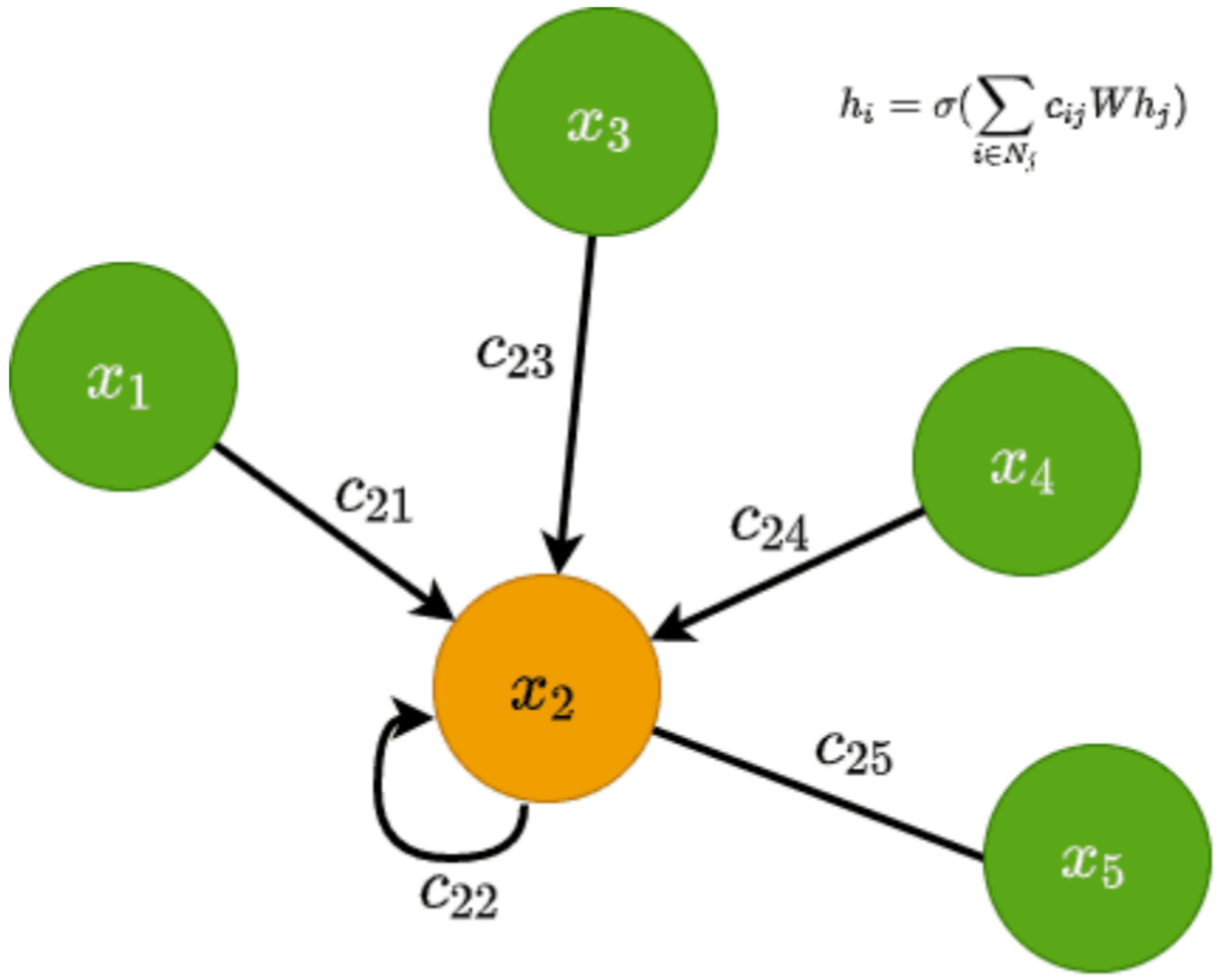
- Information we can use for pileup mitigation
  - Per-particle individual features: PU particles low pT; LV particles high pT; PU particles more in the forward region; LV particles more in the central region
  - Particle neighboring features: PU particle neighbors are more likely to be PU; LV particle neighbors are more likely to be LV

- To put together make use all such information together:
  - Combining particle individual features and neighboring features;
  - Avoid preselections, cut tunings, matrix selectetc

- GraphNN is an efficient and effective way to do this.

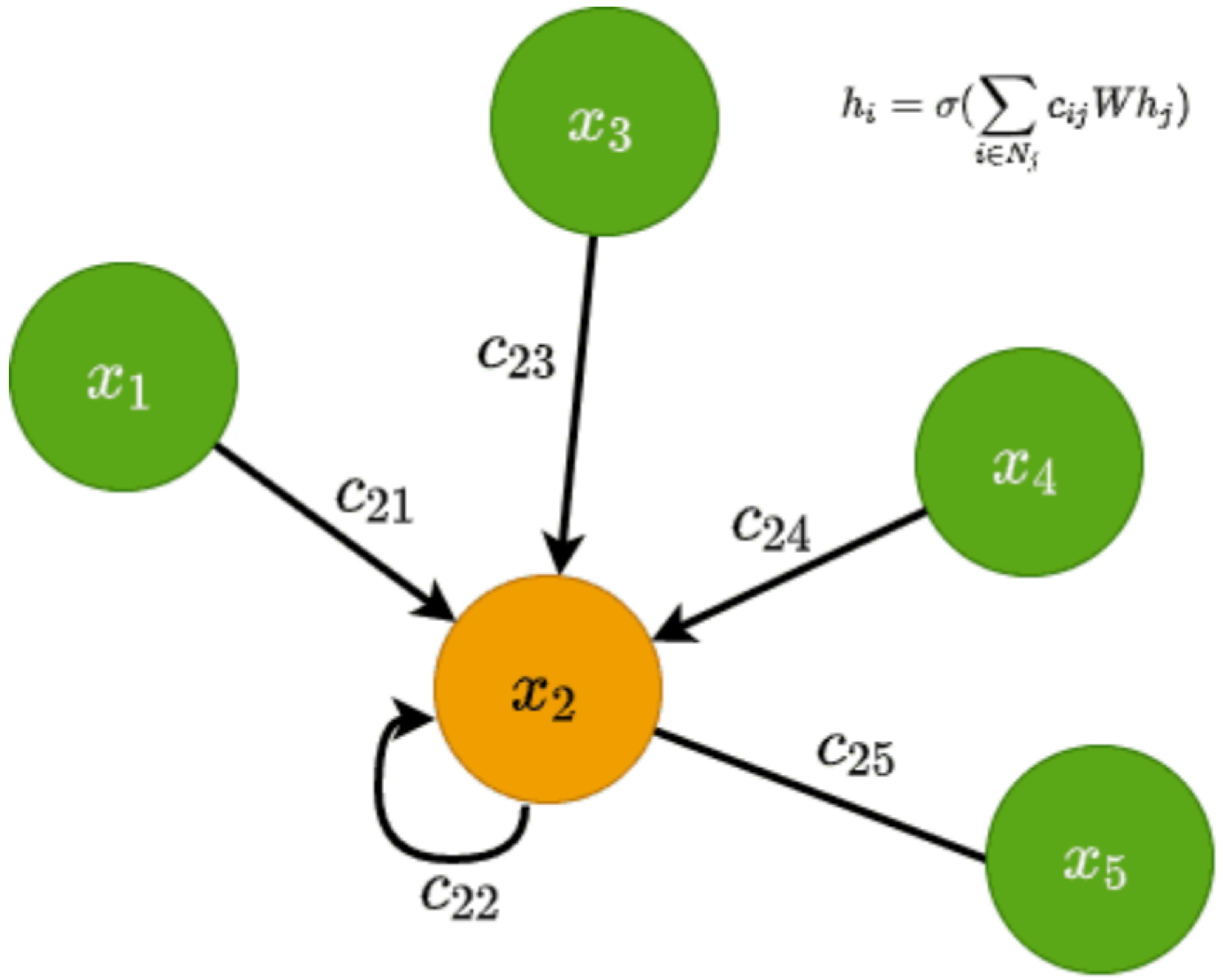
CNN -> GNN





- Convolutional Neural Networks work on Euclidean space and can aggregate information from the "real" neighbors adjacent to each target.
- Moving to Non-Euclidean space; do the similar type of "convolutions" to extract and aggregate information from neighboring particles -> Graph Neural Network (More general and more powerful)

Graph Neural Networks



- One Graph (G) has nodes (V) and edges (E): G = (V,E)
- For one node x_i, represented with h_i. the available information includes:
  - Target node itself: h_i
  - Neighboring nodes: $\{h_j\}$
  - Neighboring edges: $\{e_{ij}\}$
- "message passing". The node representation upgrade in the k-th iteration, is a function of $(h_i, \{h_j\}, \{e_{ij}\})$
  - $h_i^{k+1} = f(h_i^k, h_j, e_{ij})$
- Can aggregate information from both target node, neighboring node, and the edges
- Information upgrades have a lot of degrees of freedom;
  - Can include different types of symmetries in the expression; works very well on point-cloud data (HEP data is mostly point-cloud)

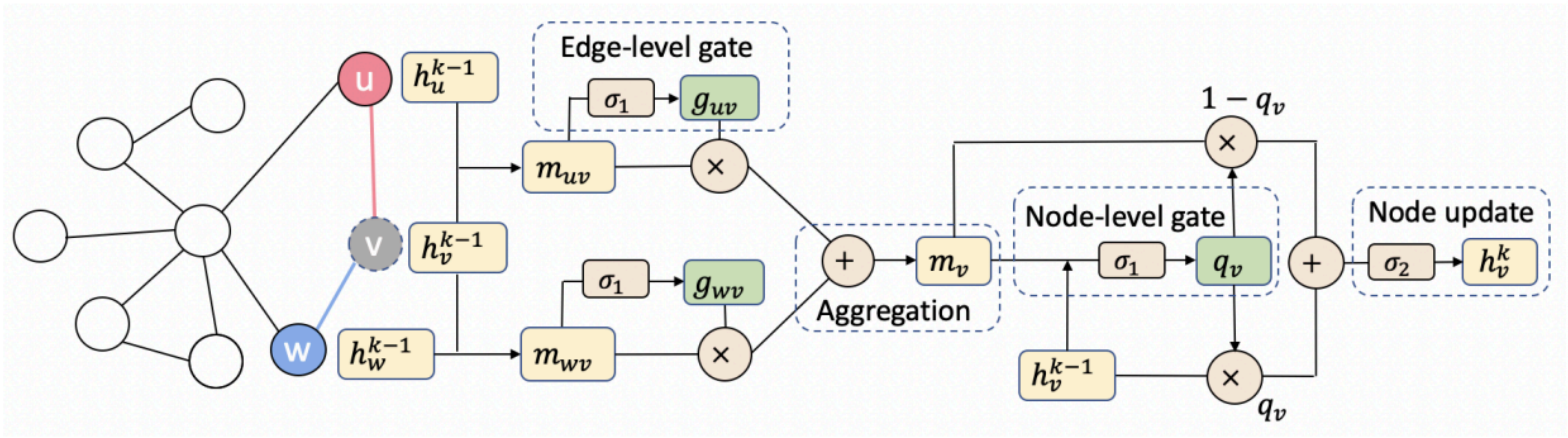
Typical Graph Neural Networks

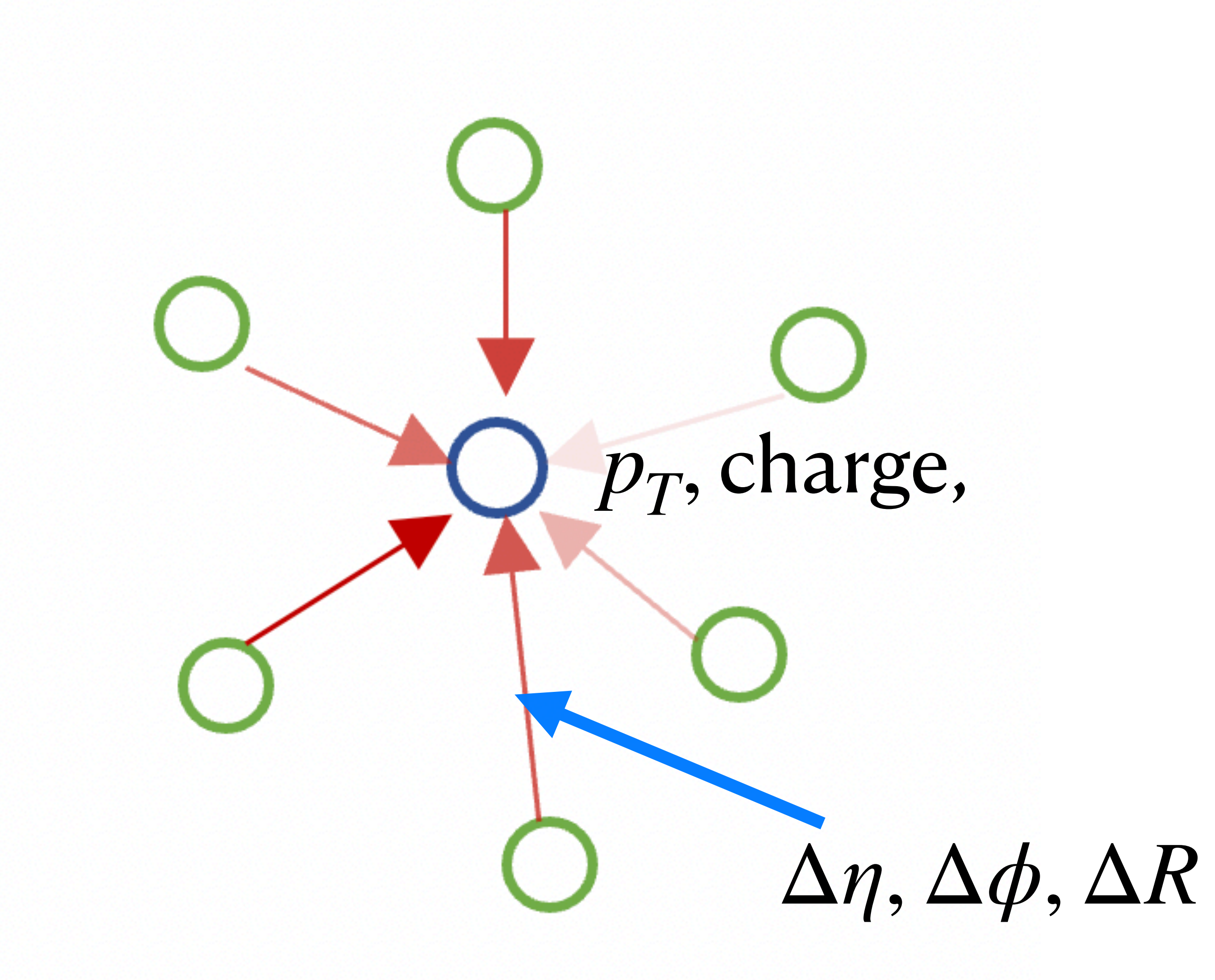
- GraphSage:
  - Poor the neighboring information together, combine with the target node information
  - $h_u^{k+1} = f(h_u^k, \{h_{v_j}^k\}, \{e_{u,v_j}^k\}) = \sigma(h_u^k w_1^k + \sum_{v_j} h_{v_j}^k w_2^k)$

- Gate models:
  - Add one gate G_u^k to control the message passing ("importance"):
  - $\bar{h}_u^{k+1} = f(h_u^k, \{h_{v_j}^k\}, \{e_{u,v_j}^k\})$
  - $h_u^{k+1} = G_u^k \bar{h}_u^k + (1 - G_u^k) h_{v_j}^k$, where $G_u^k = Sigmoid(\bar{h}_u^k, h_u^k)$ is in the range of 0-1

- Attention models:
  - "Attention" to different nodes and edges
  - $\sum_{v_i} h_{v_i}^k \rightarrow \sum_{v_i} \mathrm{Att}_{u,v_j}^k h_{v_j}^k$, where $\sum_{v_i \in N} \mathrm{Att}_{uv_j}^k = 1$ for normalization

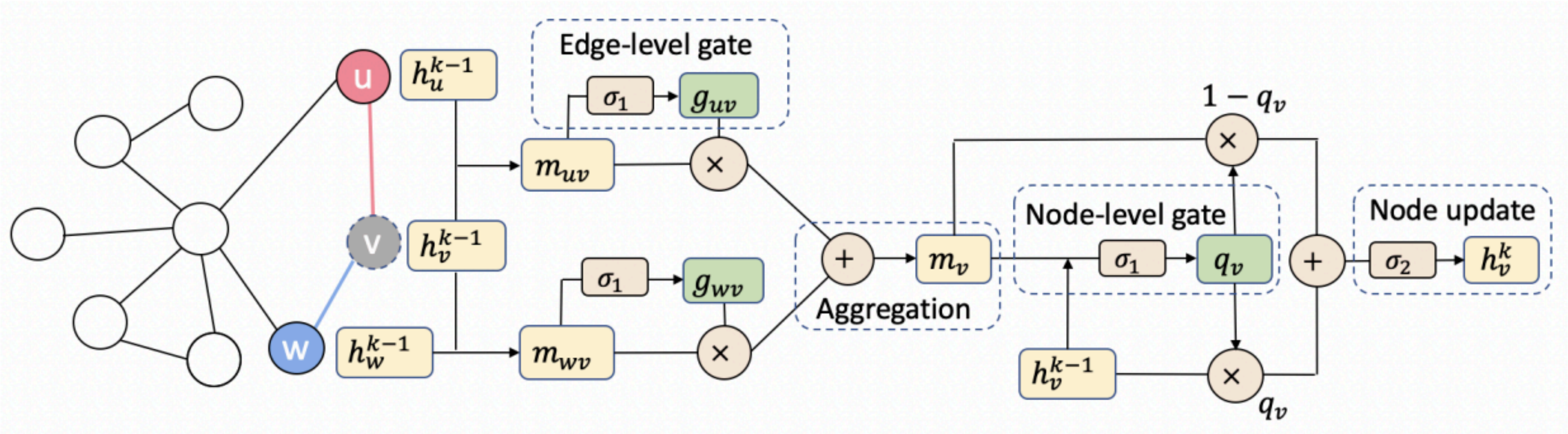
Our Model Architecture

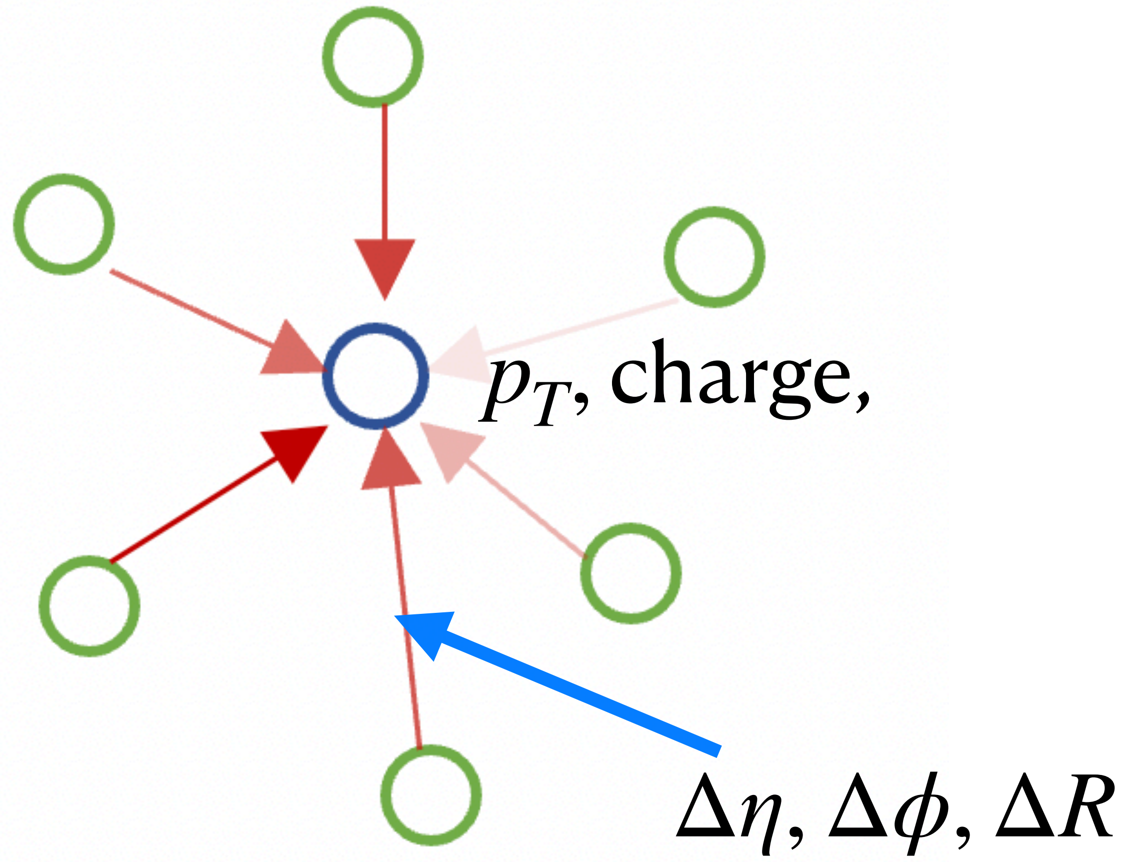




- Build graph in $\eta - \phi$ space. Connect the particles in the $\Delta R = 0.4/0.8$ cone.Input features:
  - Node features: p_T , charge
  - Edge features: $\Delta\eta$, $\Delta\phi$, and ΔR between particles
- Outputs are a weight between 0 and 1, representing the probability that the particle is produced from the LV
- Model architecture: gated model

Our Model Architecture





Message formulation: $m_{uv} = \left[h_u^{k-1}, h_v^{k-1}, \Delta\eta_{uv}, \Delta\phi_{uv}, \Delta R_{uv}, h_g^{k-1}\right]$,

Aggregation: $m_v = \sum_{u \in N(v)} g_{uv} m_{uv}$, where $g_{uv} = \mathrm{Sigmoid}(W_1 m_{uv} + b_1)$

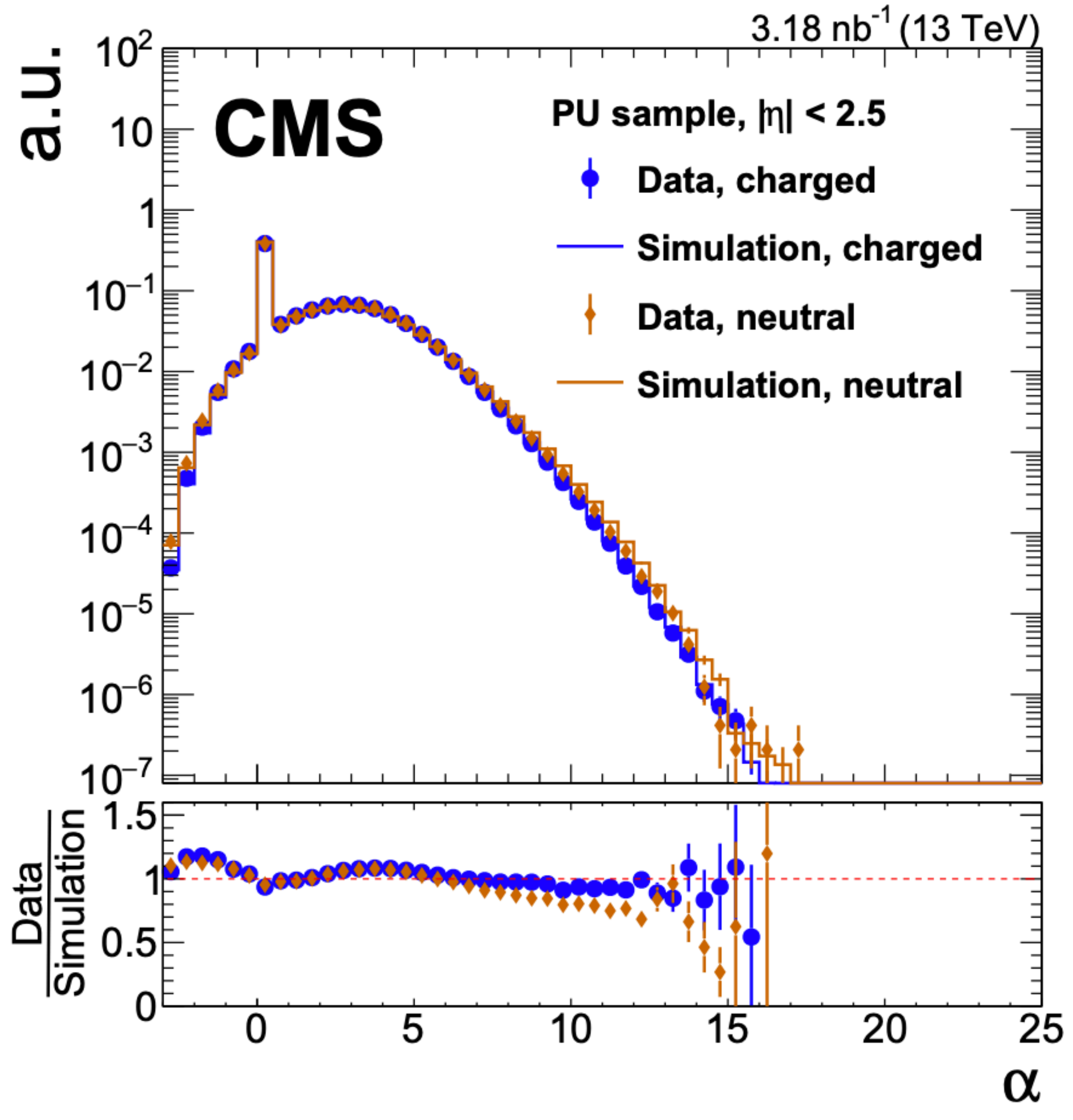
Node-level gate: $q_v = \mathrm{Sigmoid}(W_2[h_v^{k-1}, m_v] + b_2)$

Node update: $h_v^k = \mathrm{ReLU}(q_v(W_3 h_v^{k-1} + b_3)) + (1 - q_v)(W_4 m_v + b_4))$,

Semi-Supervised Learning

- To operate at particle level, the current ML models would require the prior knowledge of whether the particle is produced from PU or LV, as the ground truth information
  - For charged particles, it is easy to retrieve, even in the real data
  - For neutral particles, currently very hard to recover truth information, sometimes mixed LV/PU; no truth information in the real data

- How about we train the model using the charged particles, and then do inference on neutral particles?
  - This semi-supervised ML method would allow us to train directly on real data/ full simulation, without worrying about the labels for the ground truth information
  - This semi-supervised training strategy would work on different ML models and architectures



Masking & Training Details

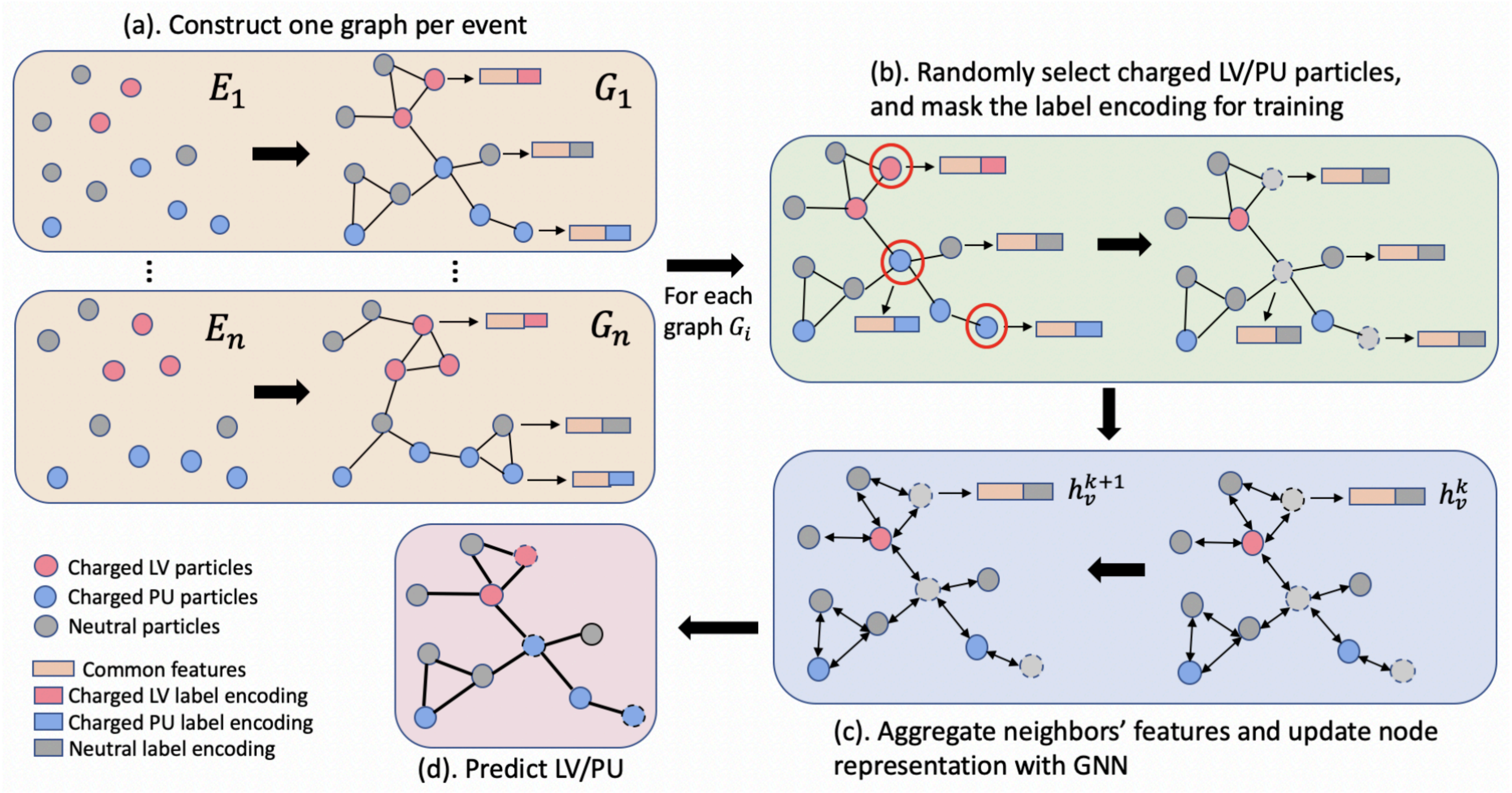


Figure 1: A diagram illustrating the SSL model training flow

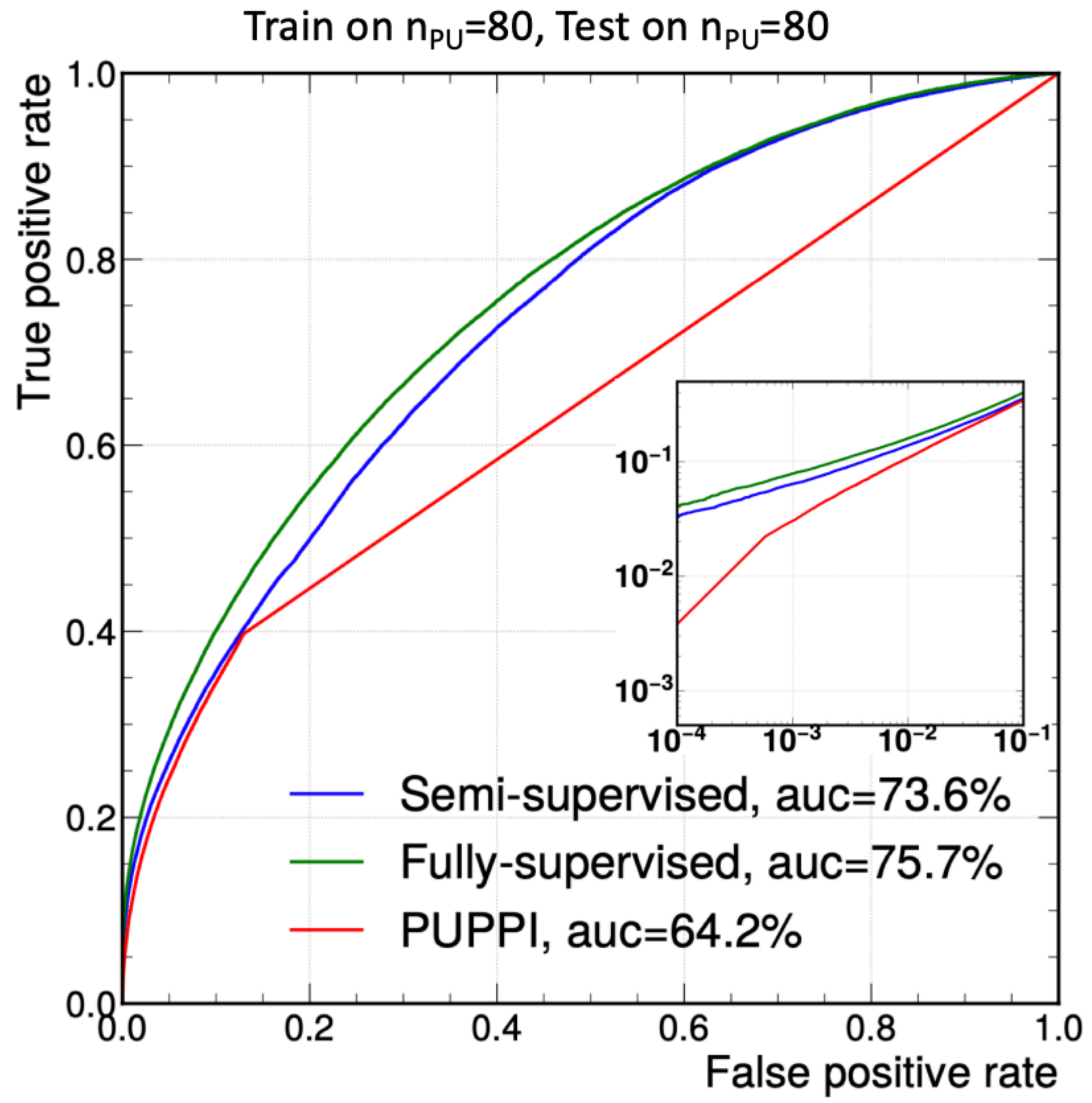
- Build graph in $\eta - \phi$ space
- Randomly select and mask a subset of charged particles
- Train on these charged particles
- Move to the next event and repeat

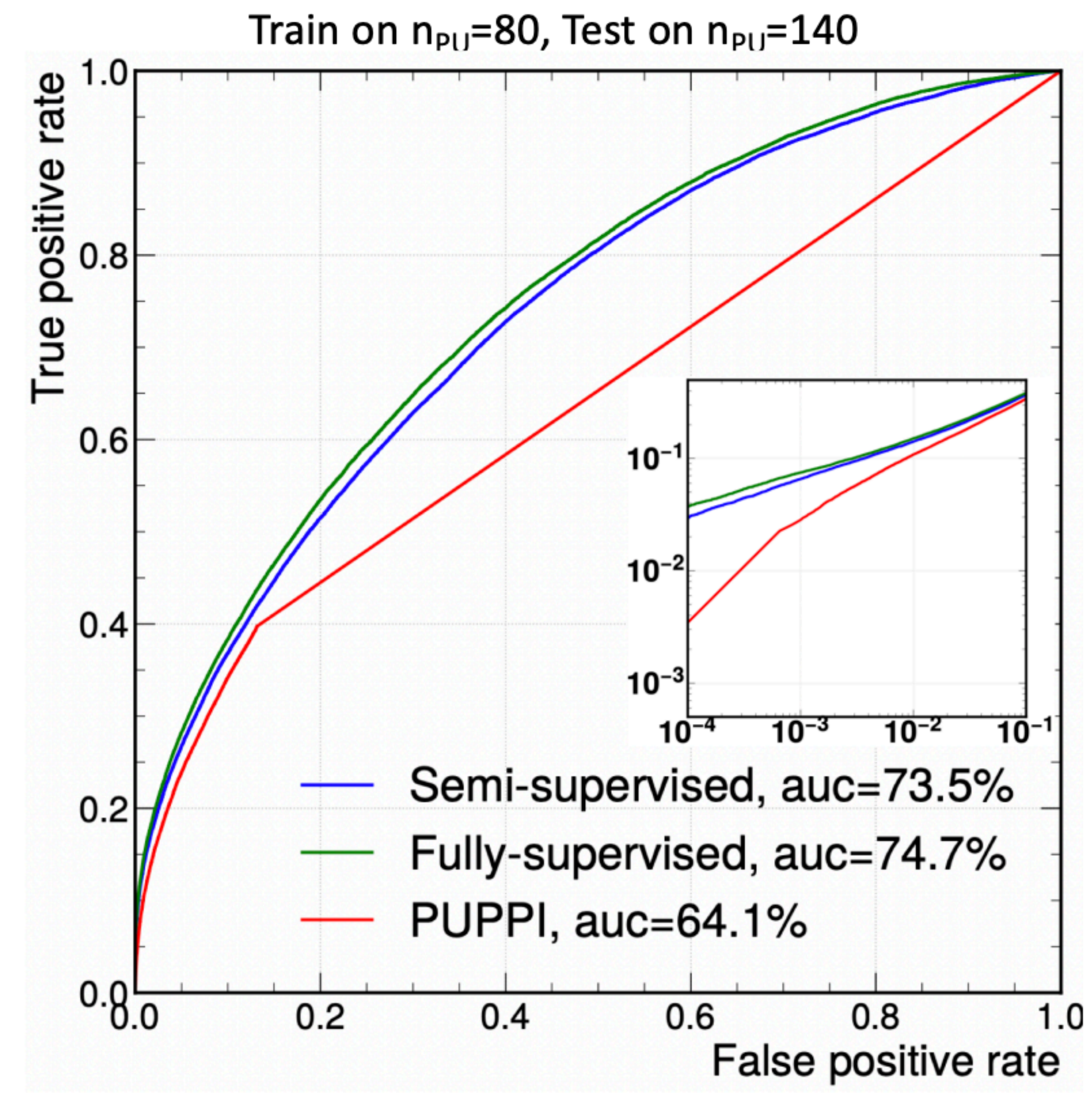
Training Datasets

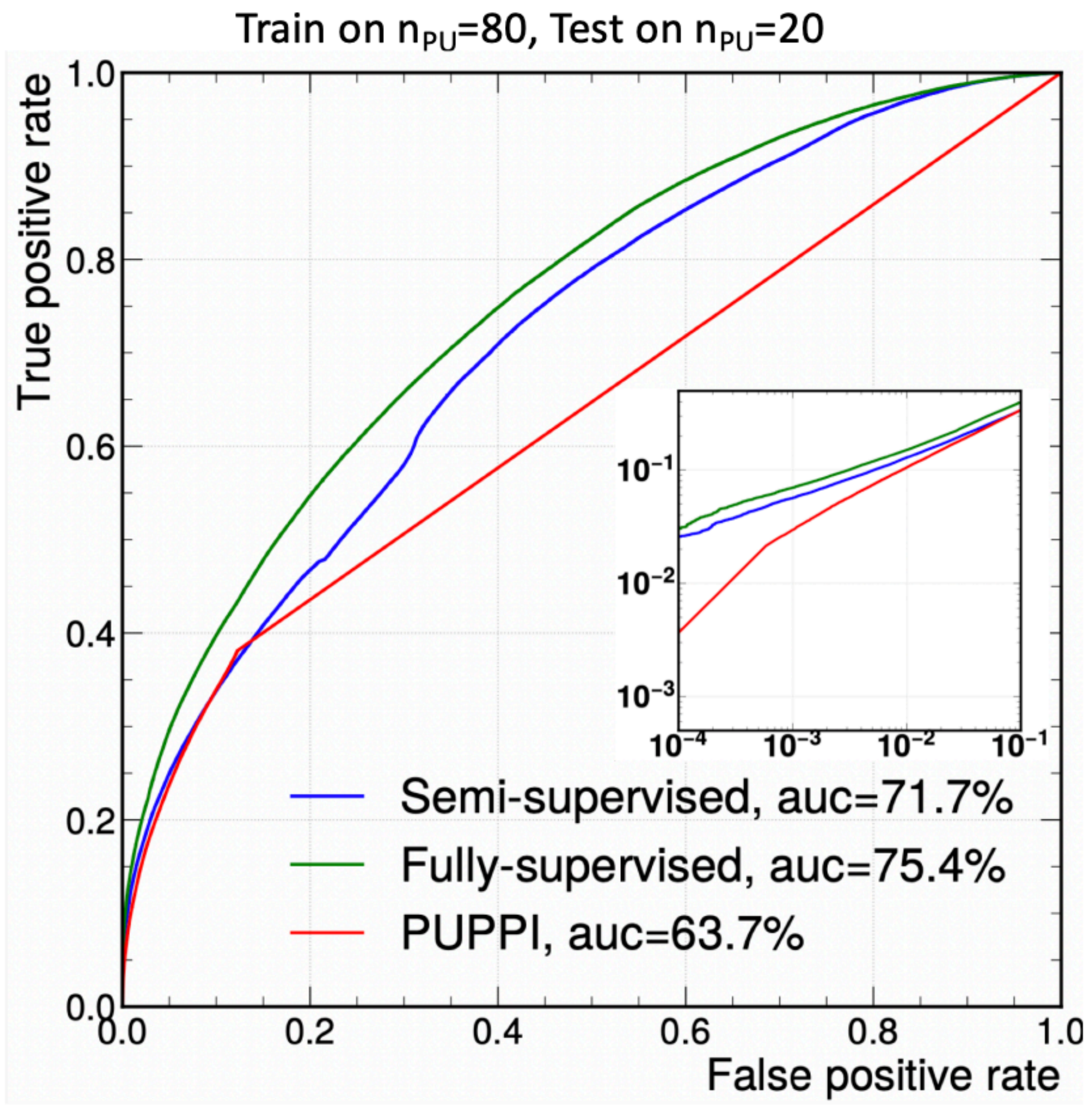
- Using similar setup as the PUPPIML
  - Pythia 8.223 + Delphes 3.3.2 for simulation
  - $Z(\nu\nu)$+jets signal processes
  - Pythia-generated QCD events as pileup; Poisson distribution sampled with the average pileup of 80 and 140
  - Charged particle flag for the LV and PU is set to be perfect
- Number of different particles per event at PU=80 (with pT>0.5GeV cut)

# Particles	Charged	Neutral
LV	85	50
PileUp	1600	800

Per-Particle Performances

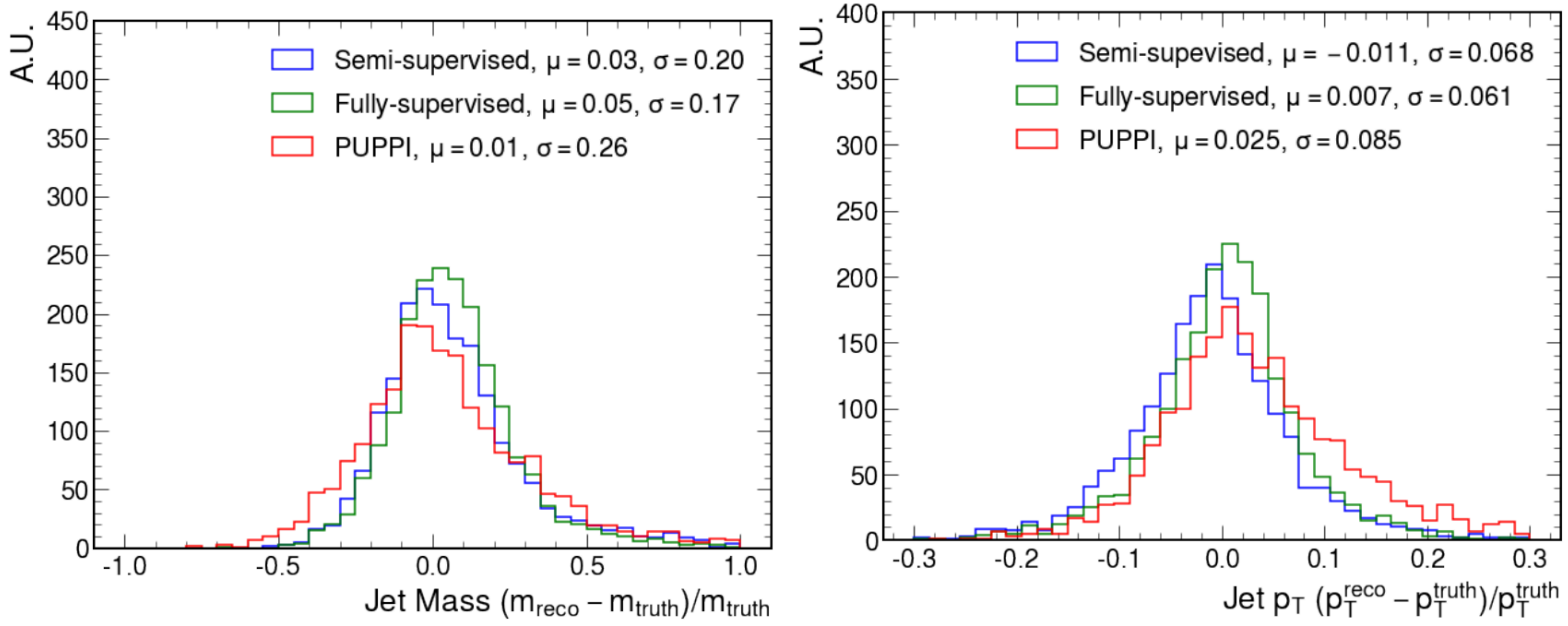






- Train on nPU=80; test on nPU=80; both supervised and semi-supervised outperforms PUPPI; supervised and semi-supervised results are close
- Train on nPU=80, still performs well on nPU=149 and nPU=20

Performance on Jet Mass, pT (PU80)



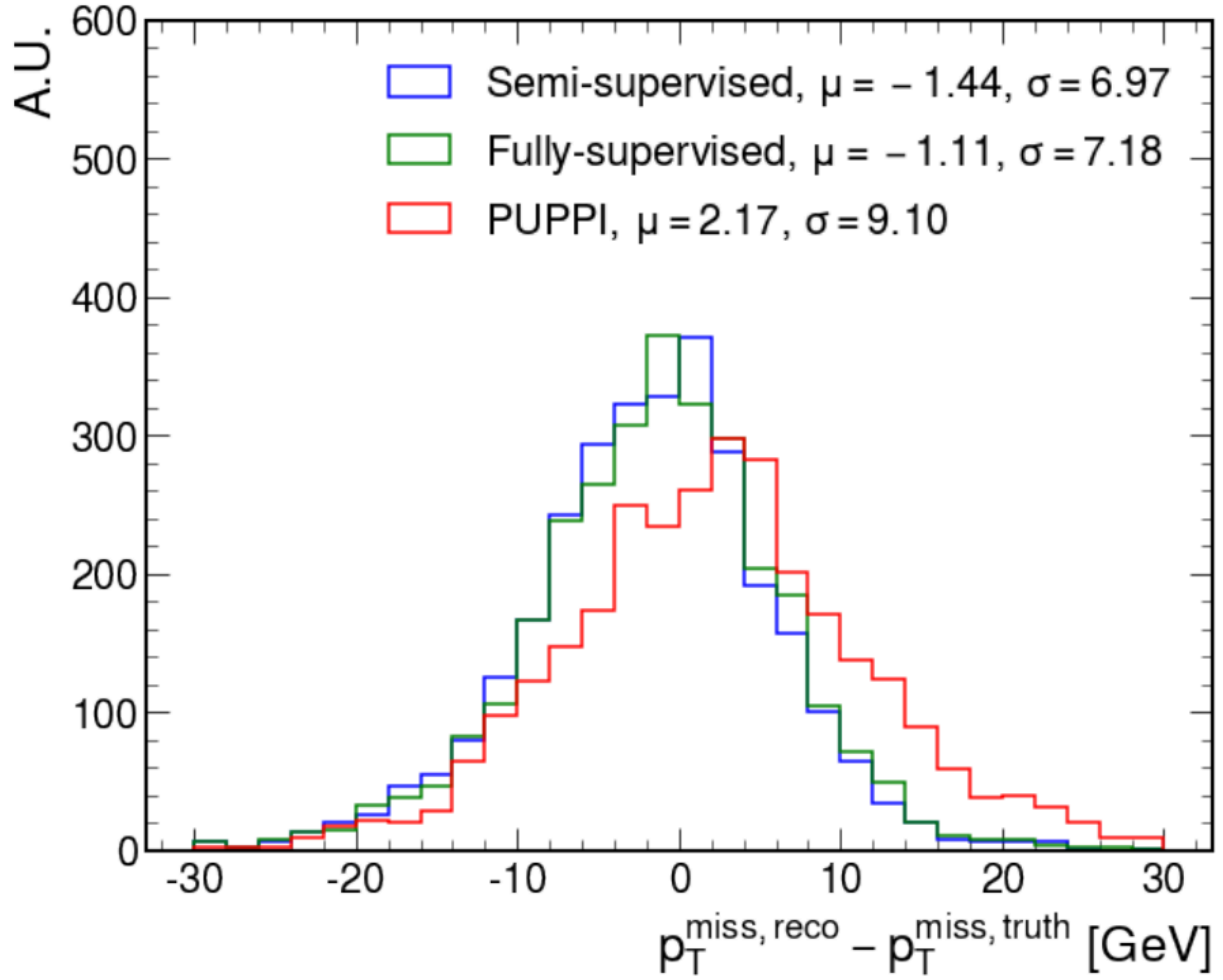
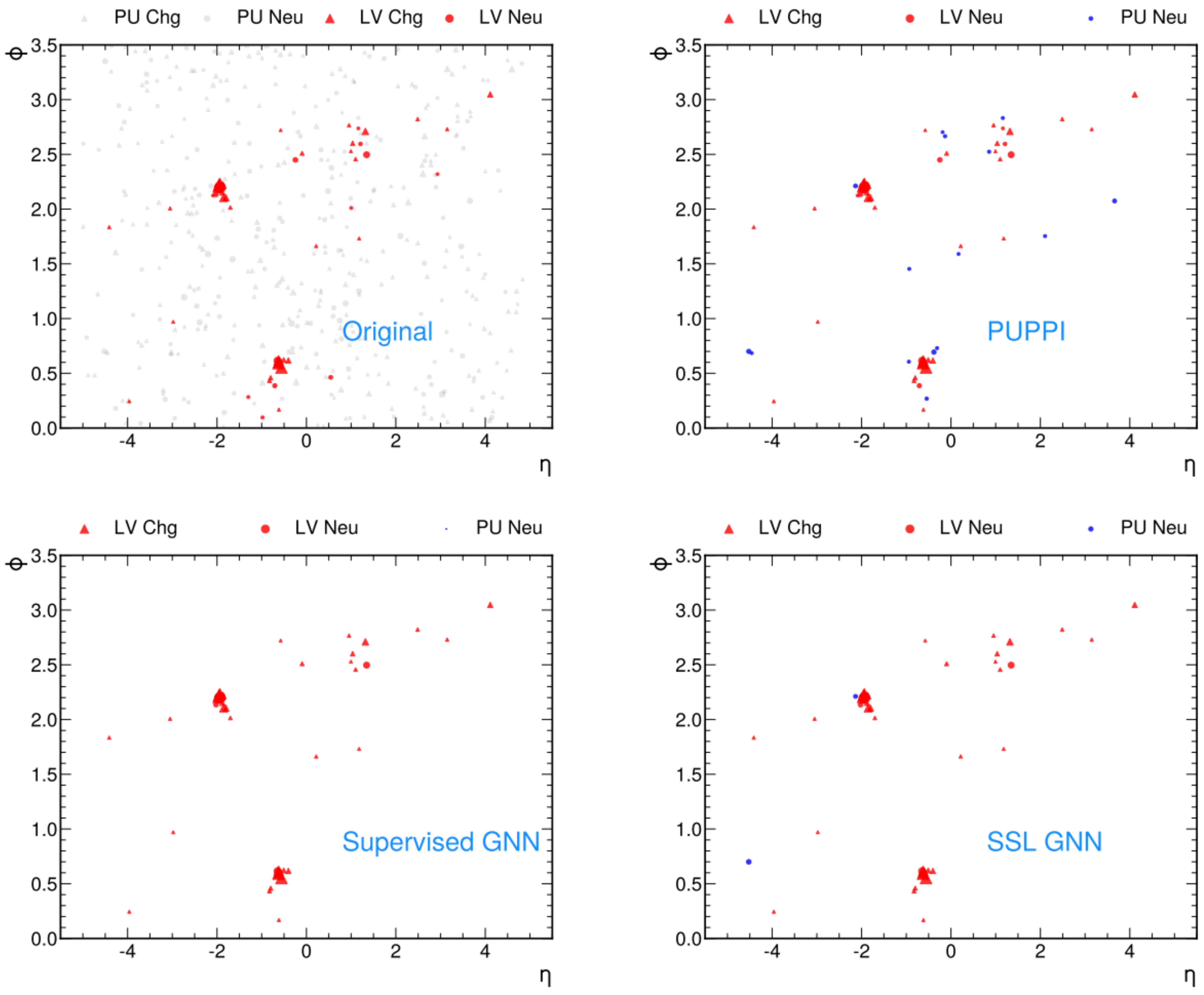


Figure 4: Performance on jet mass and jet p_T with different pileup mitigation techniques for $n_{PU} = 80$.

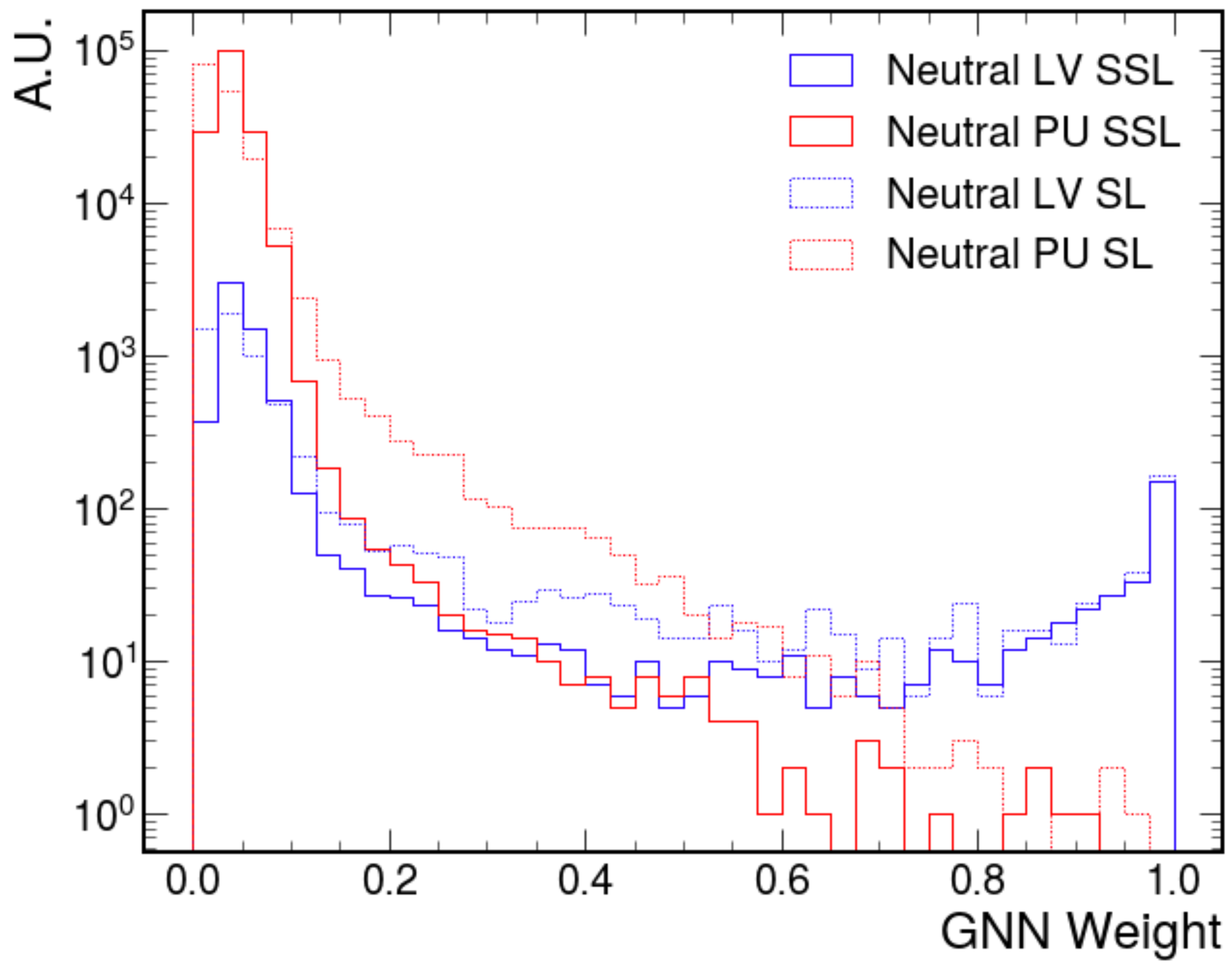
- Similar performances on jets and MET for supervised and semi-supervised; both are better than PUPPI

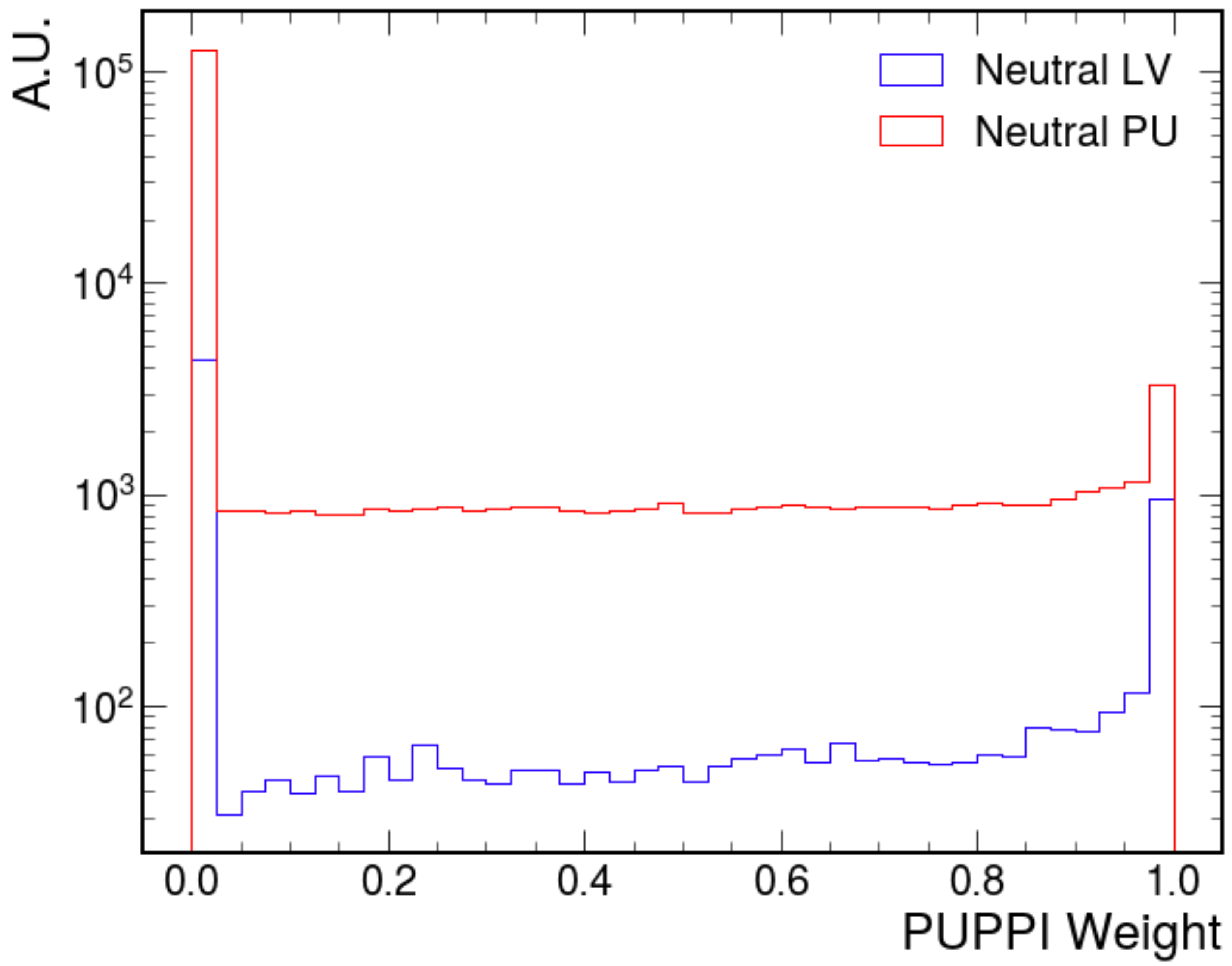
Event Display



- One event display example
- Supervised and Semi-supervised clean the pileup more effective than PUPPI
- Supervised and semi-supervised are similar

GNN Weights on Neutral Particles





- GNN Weights of neutral particles from the LV (blue) and pileup (red) on the left; right plot is the PUPPI weight distribution as a reference
- Much smaller fraction of particles get a weight around 1.
- Compared with Supervised training, the semi-supervised training seems to tend to have fewer particles in the middle weight range

Summary

- Presented the study of applying semi-supervised training for pileup mitigation with GraphNN, where the training is done on charged particles, and the inference is on neutral particles

- Results look very promising
  - Better ROC curve and resolutions on jet mass and MET for both supervised training and semi-supervised training
  - No significant performance drop going from supervised to semi-supervised

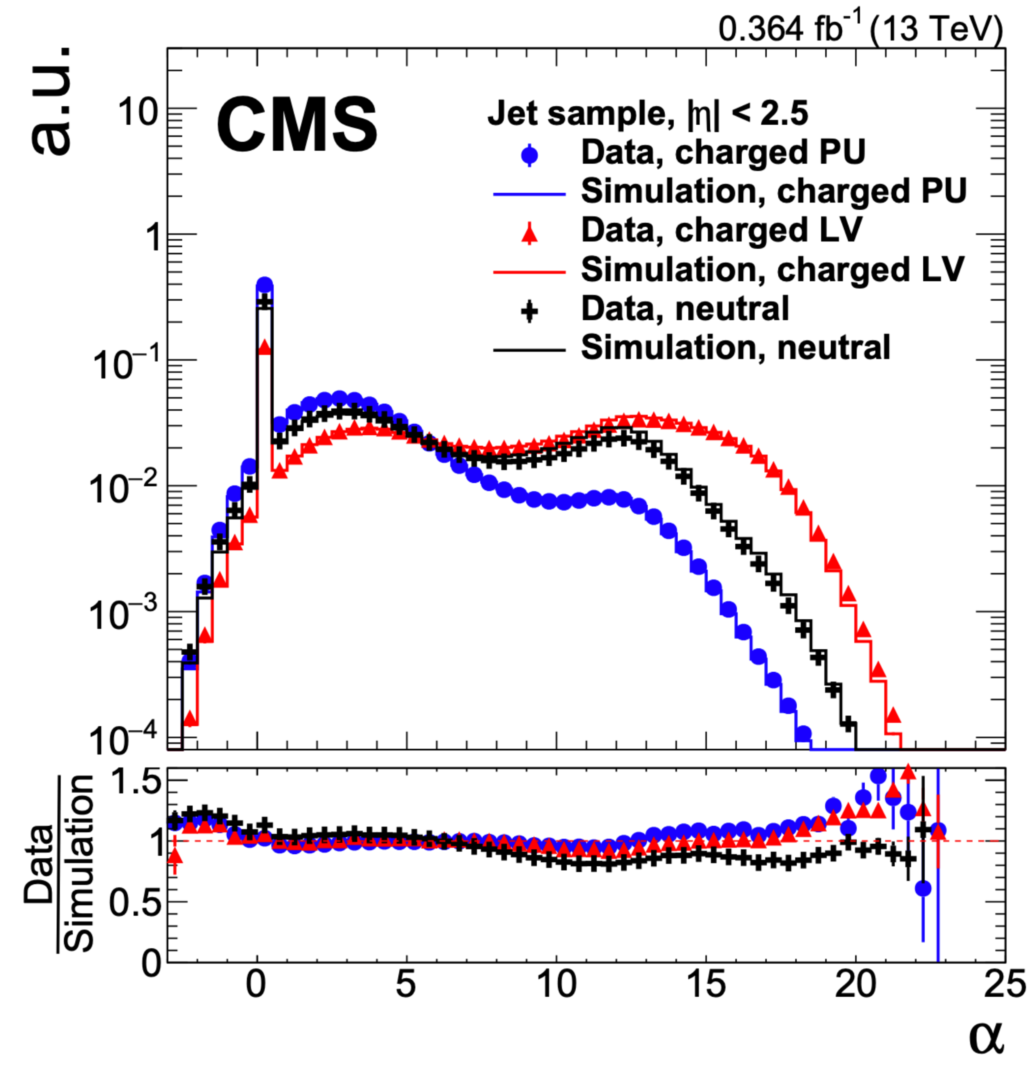
- Working on the evaluations on the CMS full-simulations; more features to explore; more realistic conditions and challenges to handle.

Back Up

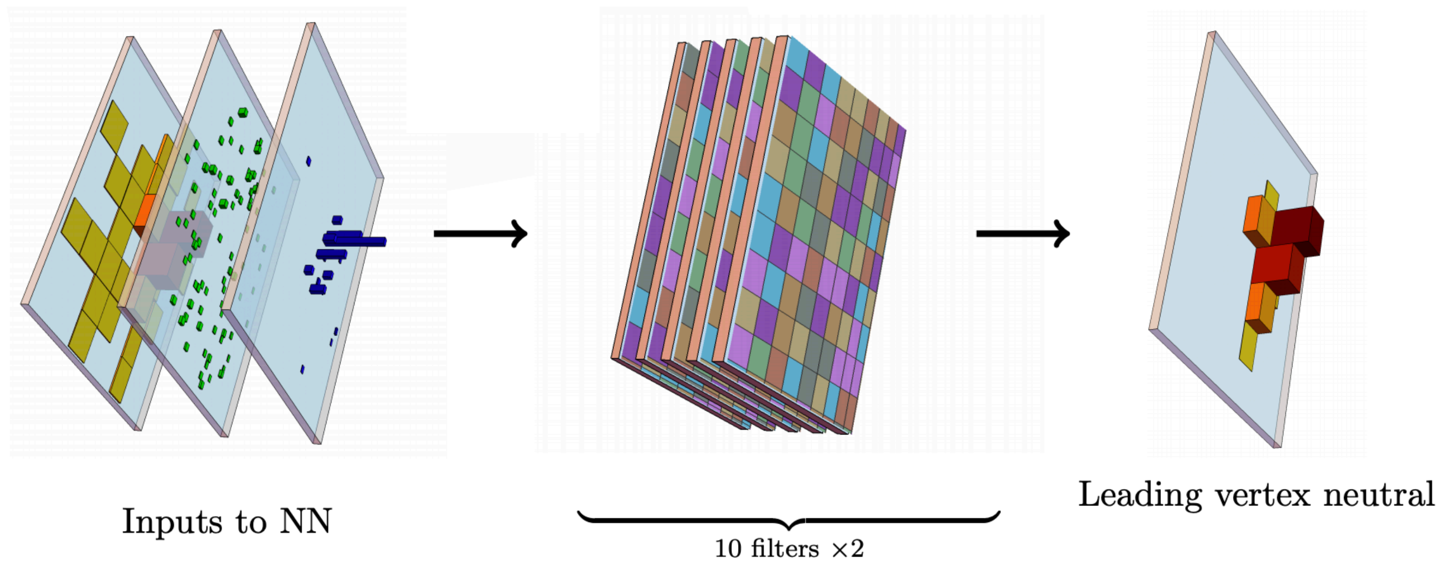
Introduction: Pileup Mitigation Studies

- Most charged leading vertex (LV) and pileup (PU) particles can be identified (Charged hadron subtraction, CHS)
- Particle self features: PU particles have lower-pT -> SoftKiller
- Particle neighboring features: PUPPI. A local shape variable α is defined and PUPPI weights are calculated based on α

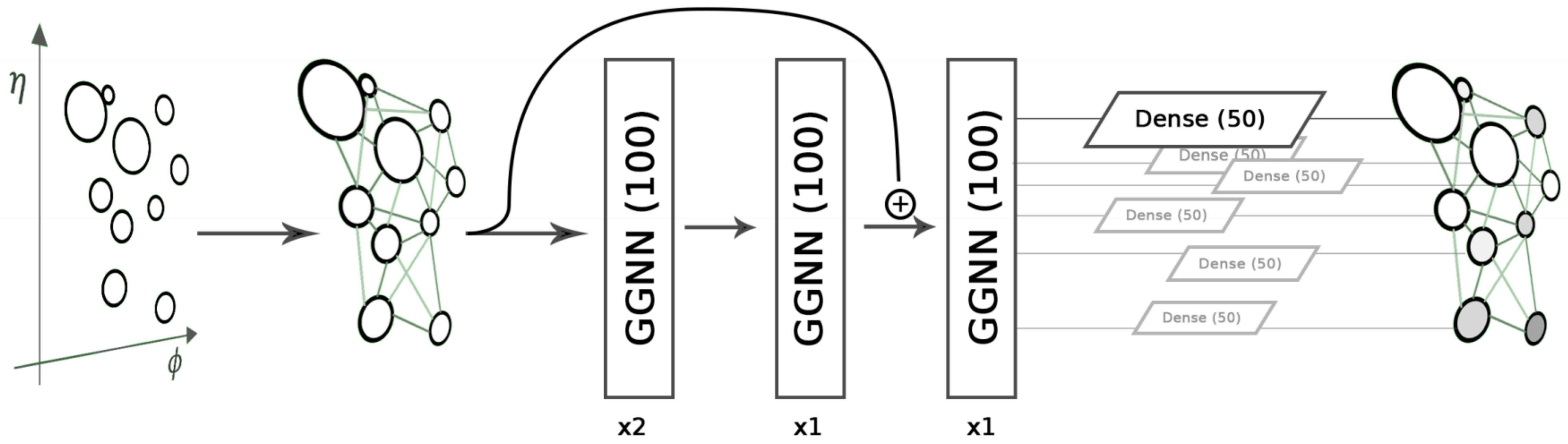
$$\alpha_i = \log \sum_{j \in \text{event}} \xi_{ij} \times \Theta(R_{\min} \leq \Delta R_{ij} \leq R_0), \quad \text{where } \xi_{ij} = \frac{p_{Tj}}{\Delta R_{ij}}.$$



- Pileup Mitigation with ML (PUMML, arxiv 1707.08600): Convolutional neural network on jet image



- Gated Graph Neural Network for PUPPI (PUPPIML, arxiv.1810.07988): GGNN on particle graph
- Other graph/attention models: (ABCNet, arxiv.2001.05311), (PUMA: with transformer, arxiv.2107.02779)



More Event Display

More Event Display