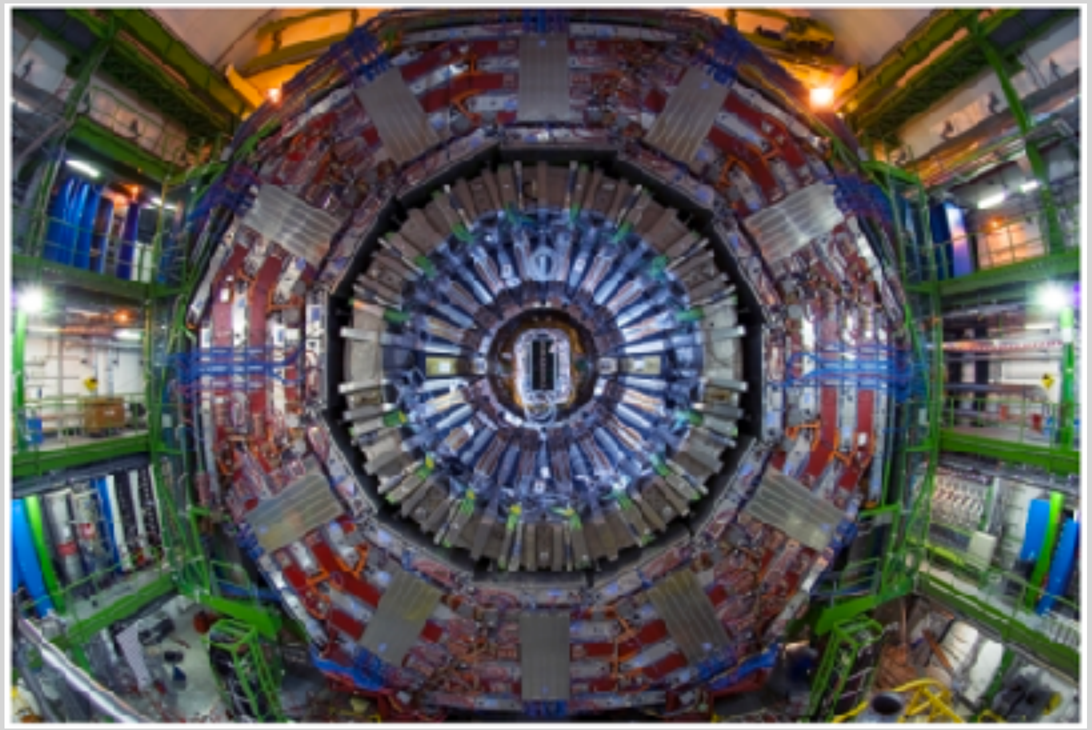
MLPF:
Machine Learning for Particle Flow

Joosep Pata, Eric Wulff, ***Farouk Mokhtar***, David Southwick,
Michael Zhang, Maria Girone, Javier Duarte

Fast Machine Learning for Science 2023

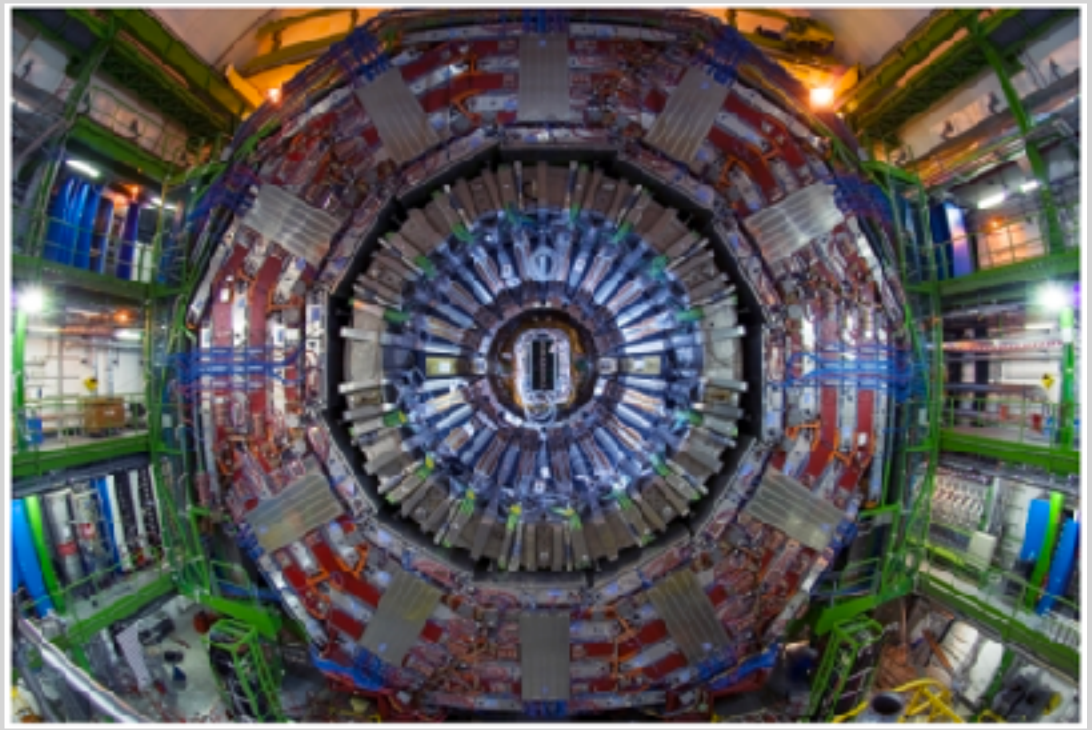
LHC Workflow

Data taking

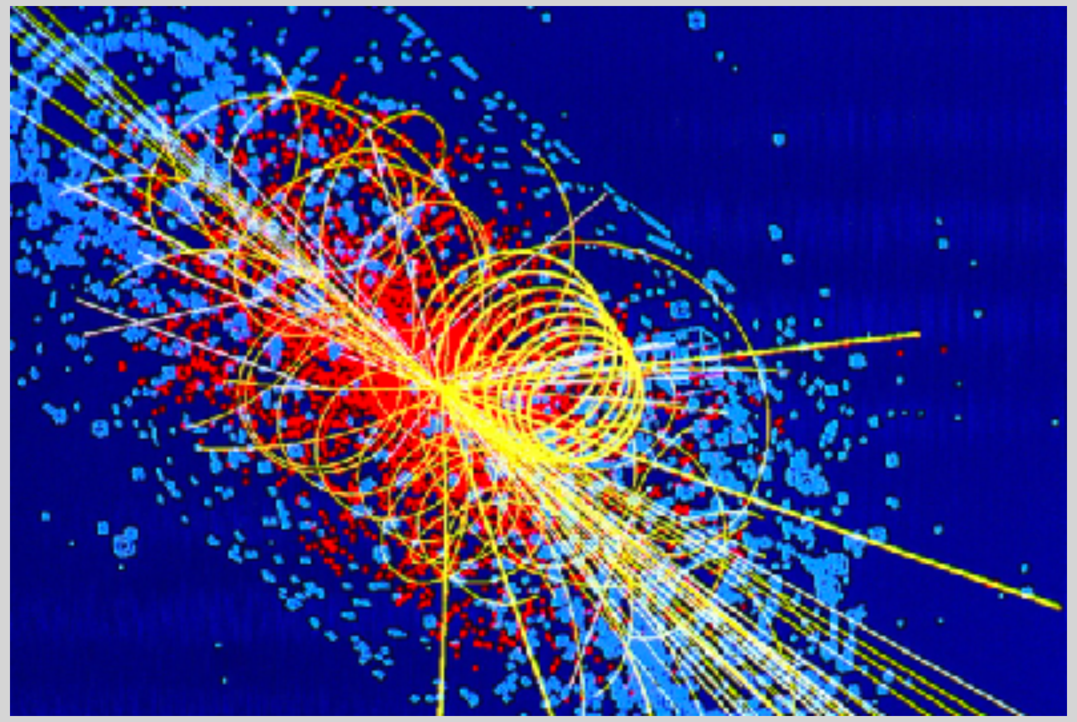


LHC Workflow

Data taking

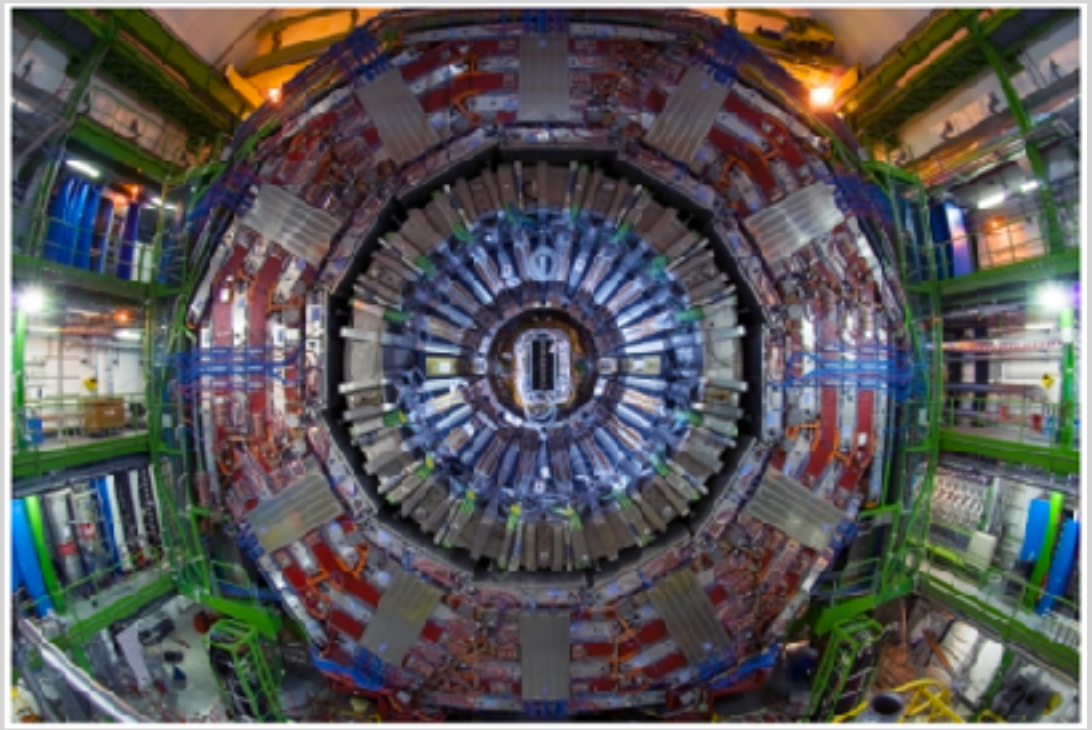


Simulation

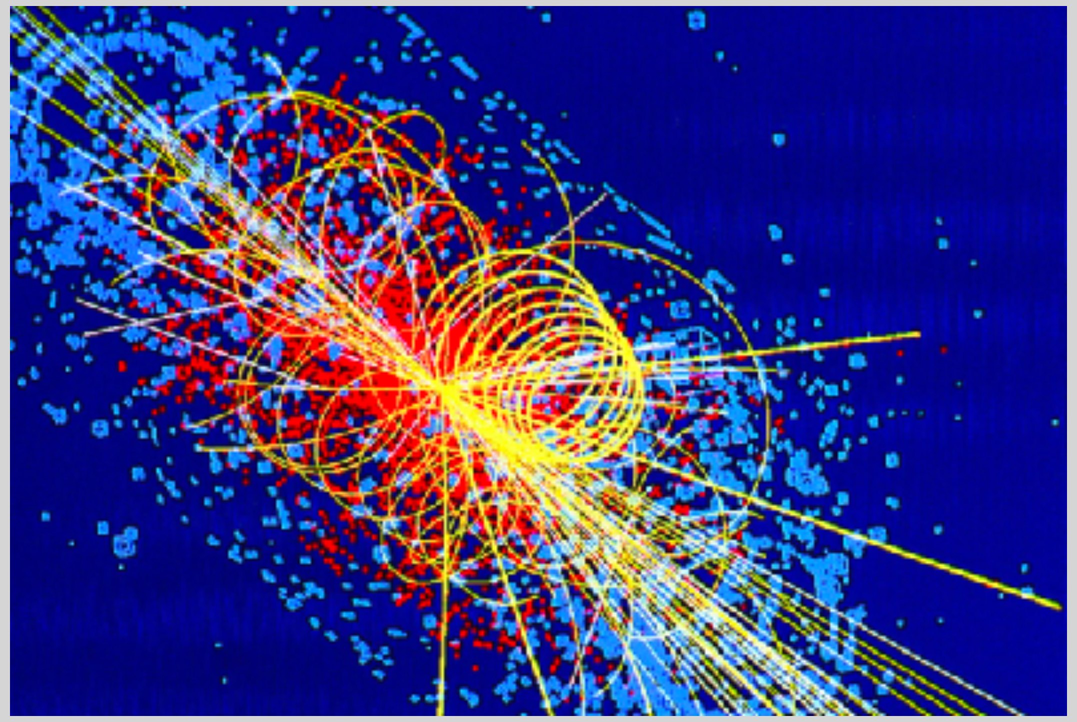


LHC Workflow

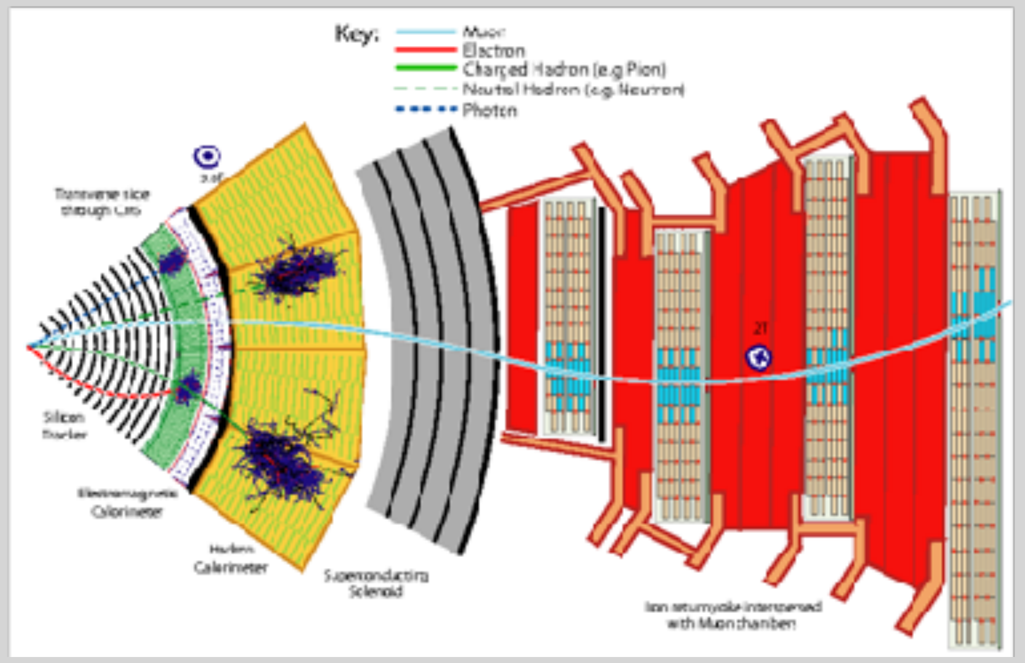
Data taking



Simulation



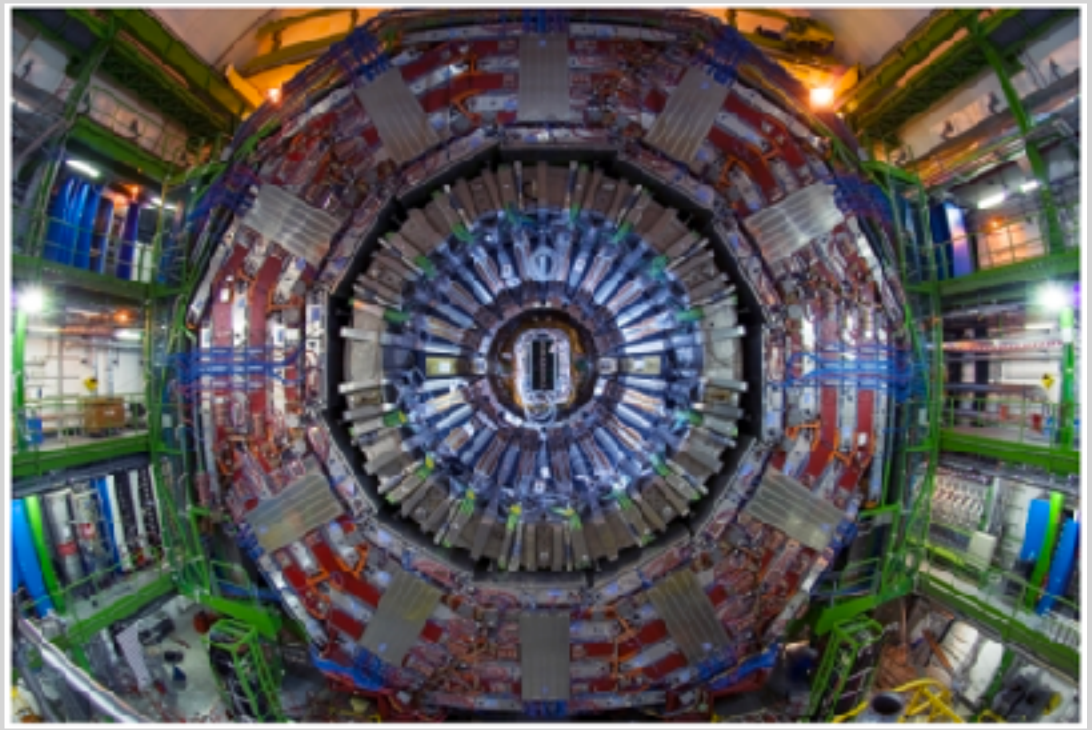
Reconstruction



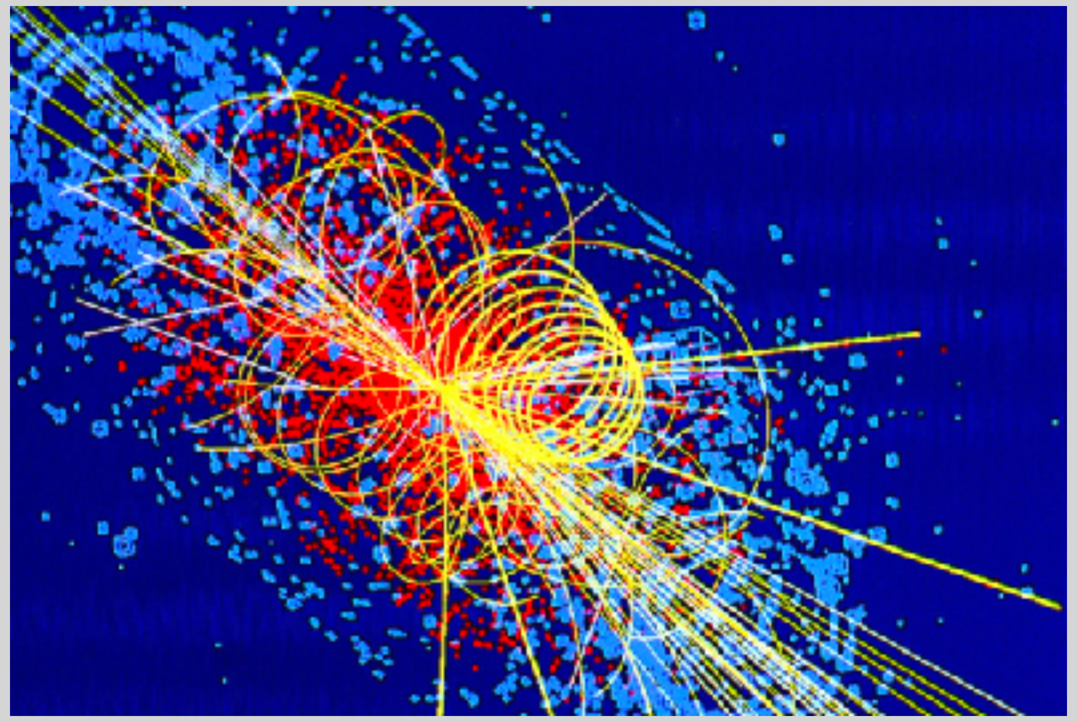
Key:
Muon
Electron
Charged Hadron (e.g Pion)
Neutral Hadron (e.g. Neutron)
Photon
Transverse slice through CMS
Silicon Tracker
Electromagnetic Calorimeter
Hadron Calorimeter
Superconducting Solenoid
Iron return yoke interspersed with Muon chambers

LHC Workflow

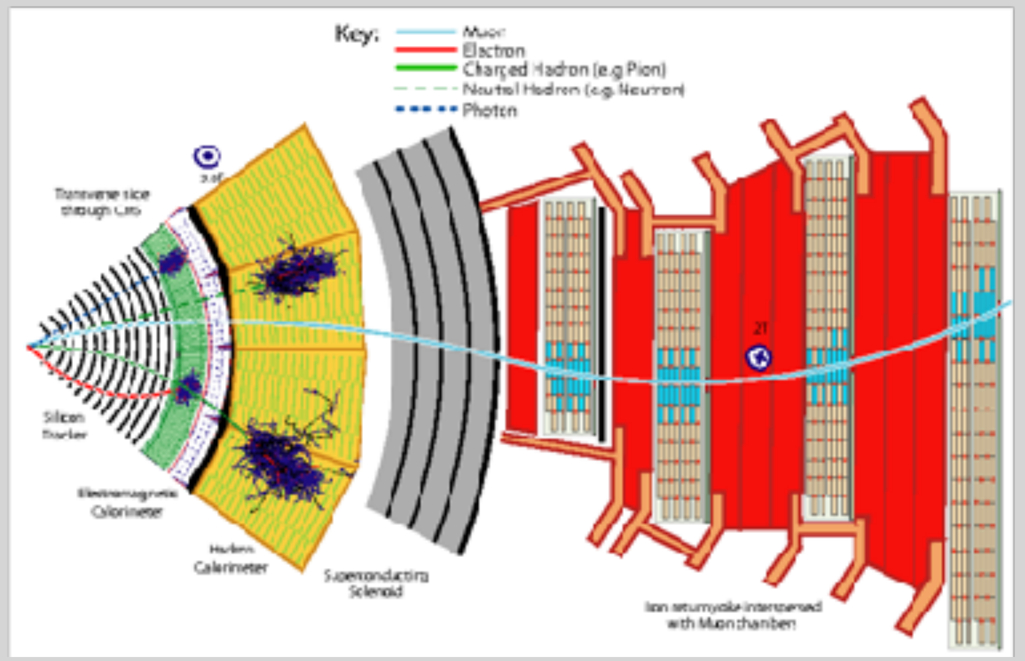
Data taking



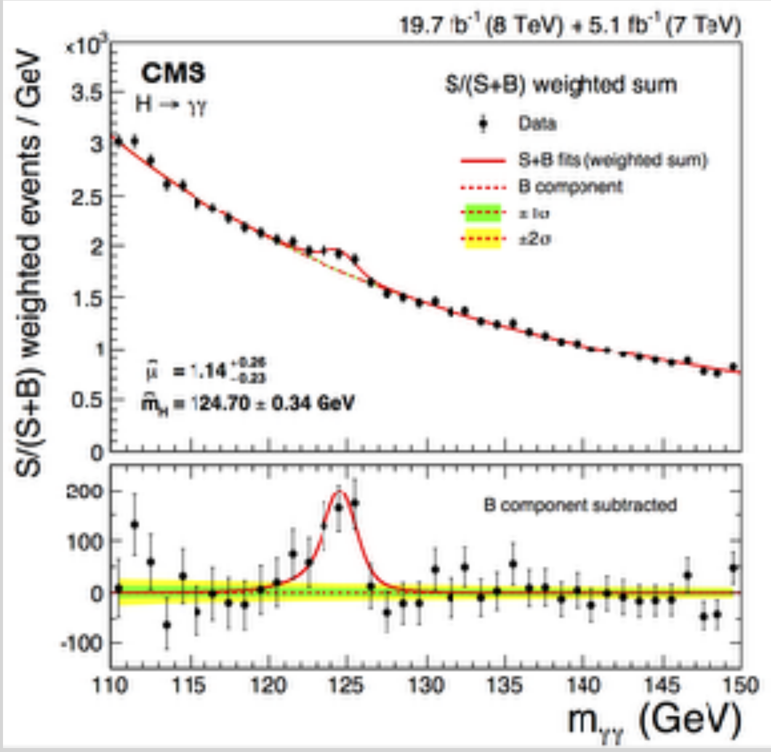
Simulation



Reconstruction

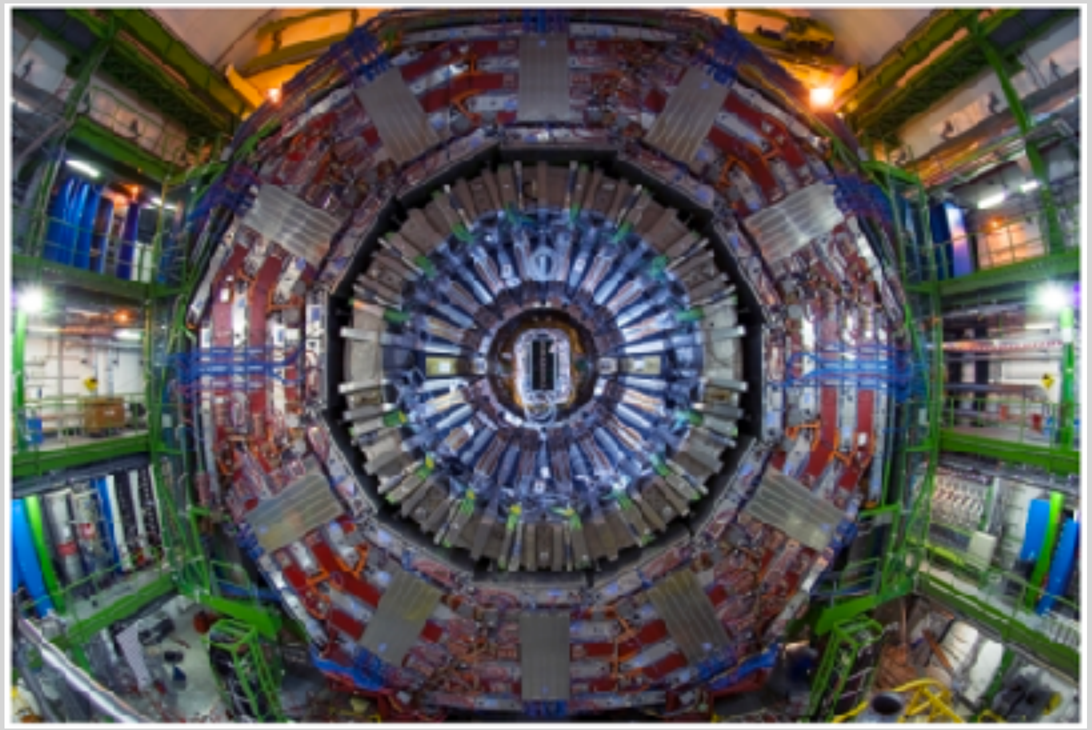

Key:
Muon
Electron
Charged Hadron (e.g Pion)
Neutral Hadron (e.g. Neutron)
Photon
Transverse slice through CMS
Silicon Tracker
Electromagnetic Calorimeter
Hadron Calorimeter
Superconducting Solenoid
Iron return yoke interspersed with Muon chambers

Analysis

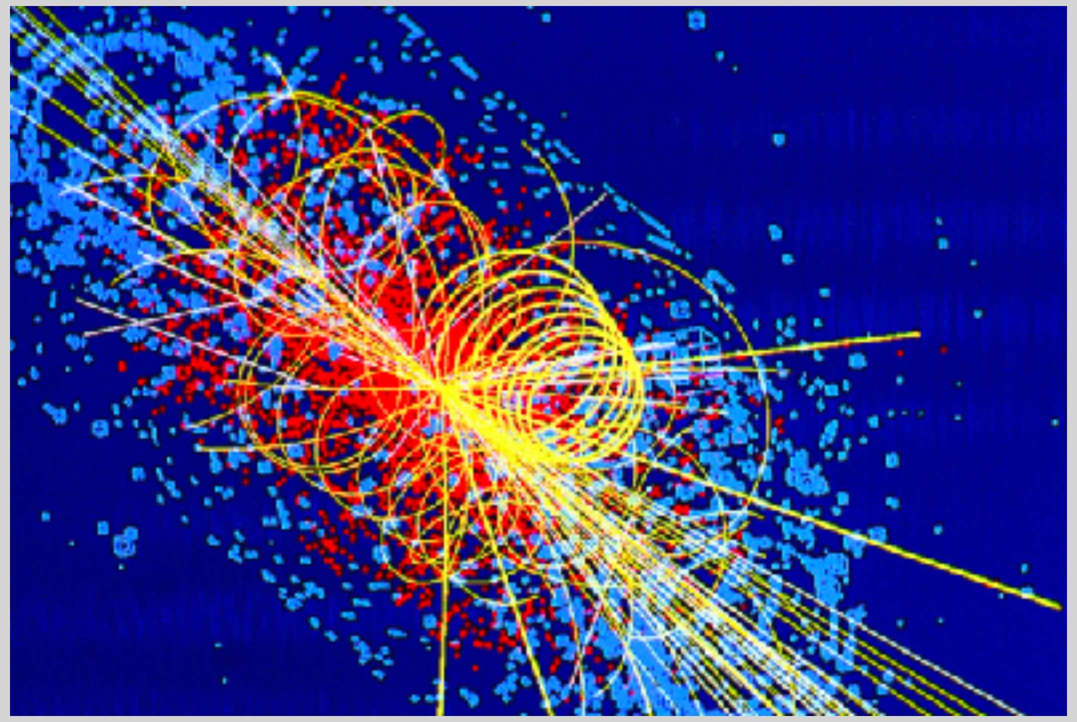


LHC Workflow

Data taking

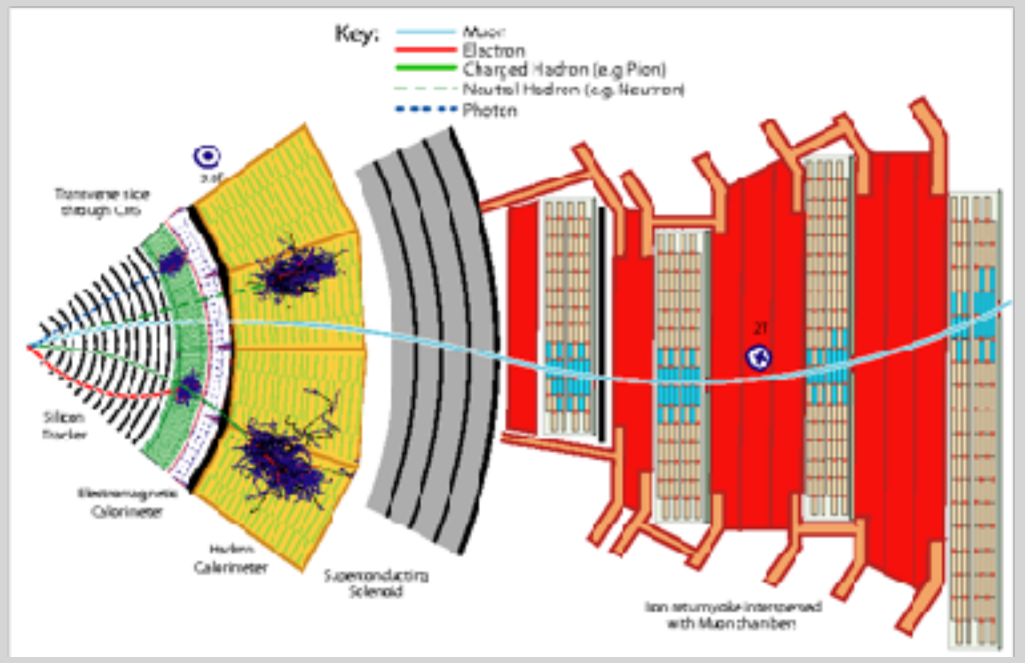


Simulation

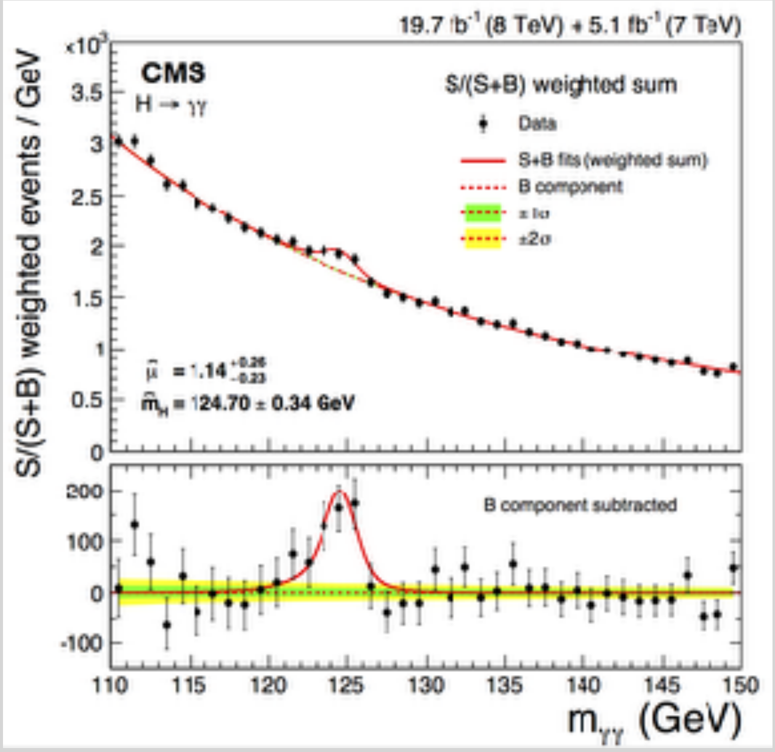


This talk

Reconstruction

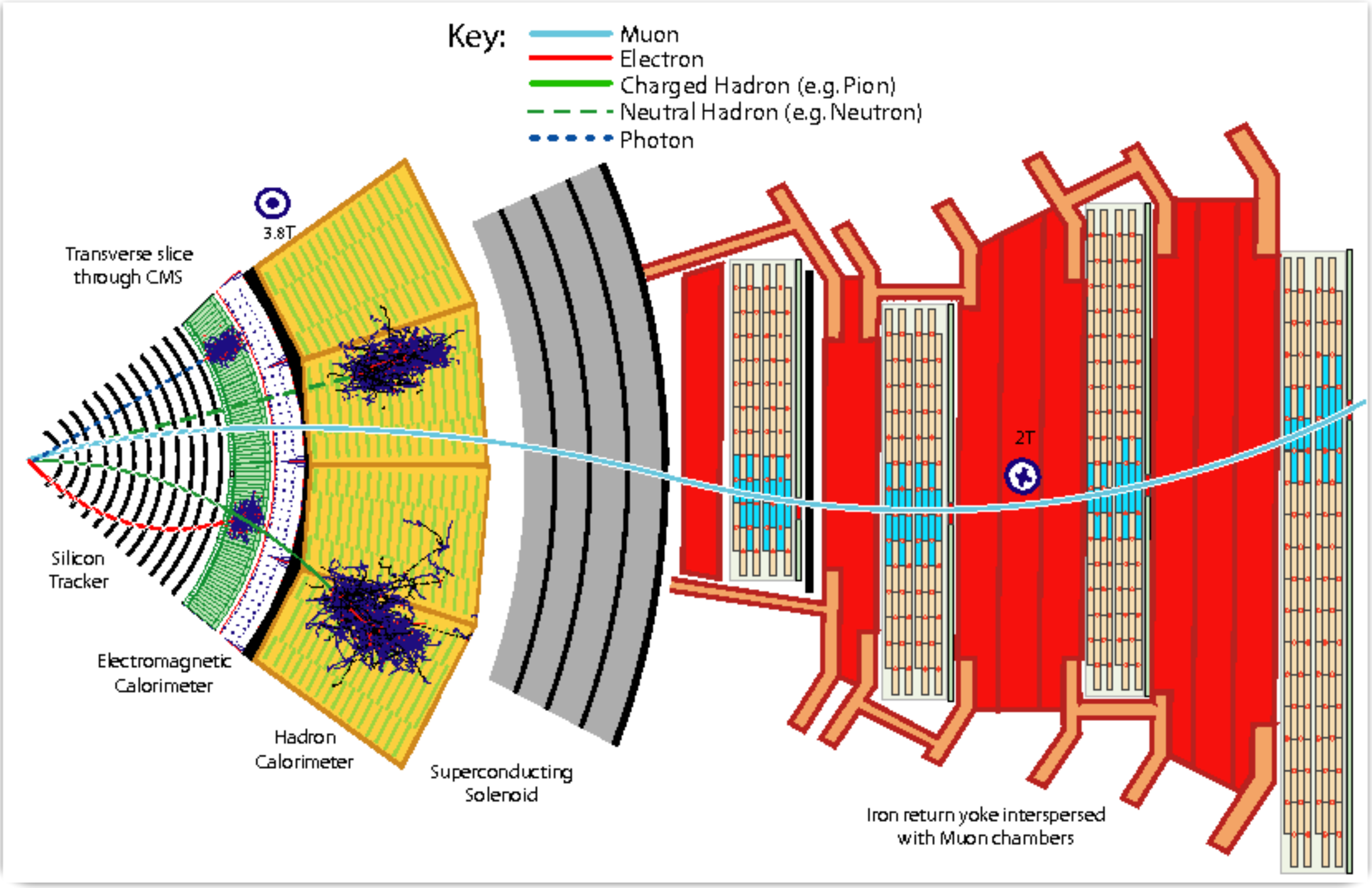

Key:
Muon
Electron
Charged Hadron (e.g Pion)
Neutral Hadron (e.g. Neutron)
Photon
Transverse slice through CMS
Silicon Tracker
Electromagnetic Calorimeter
Hadron Calorimeter
Superconducting Solenoid
Iron return yoke interspersed with Muon chambers

Analysis



Event reconstruction

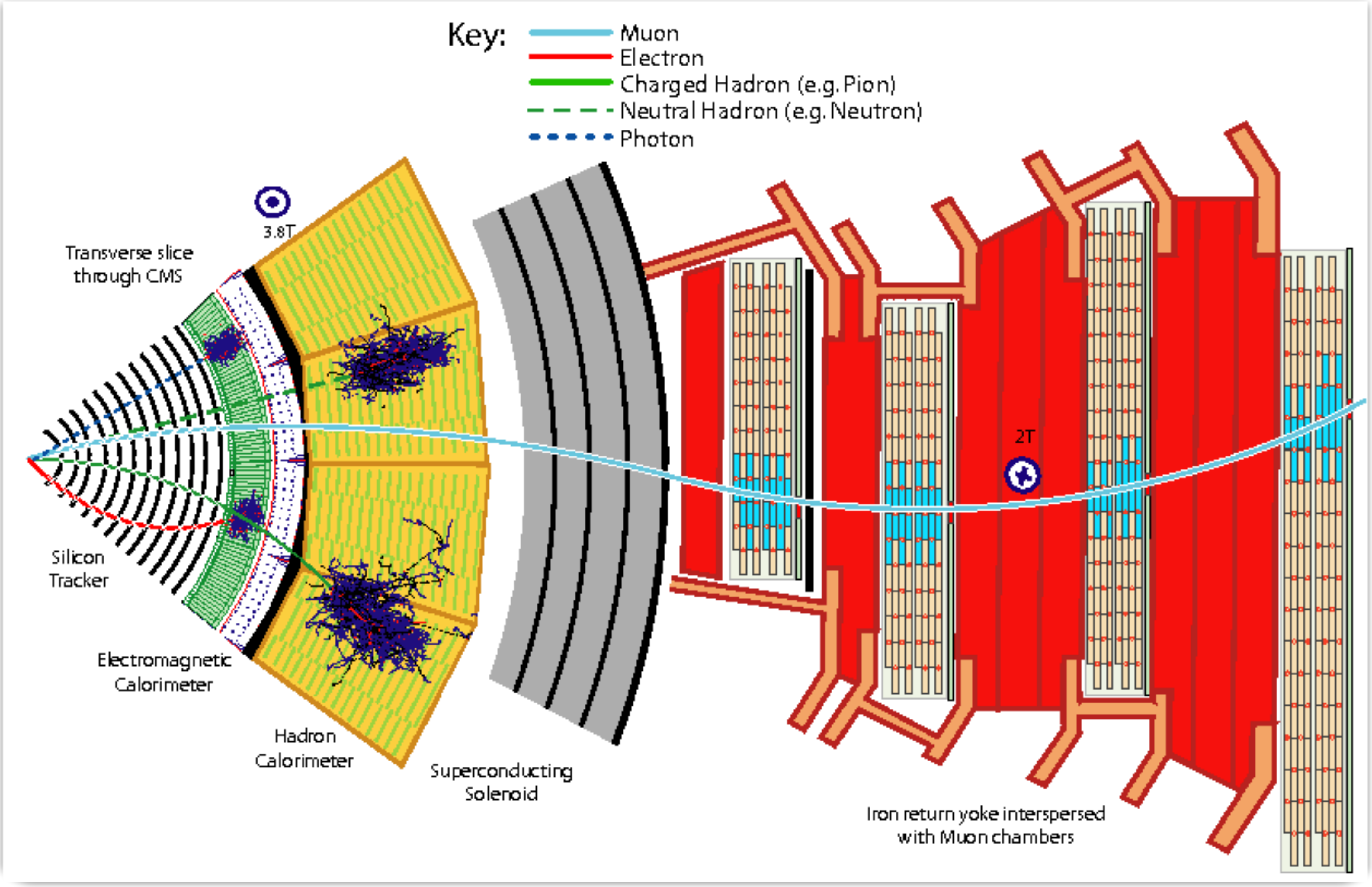
Different particles interact differently with the detector subsystems


Key:
Muon
Electron
Charged Hadron (e.g. Pion)
Neutral Hadron (e.g. Neutron)
Photon
3.8T
Transverse slice through CMS
Silicon Tracker
Electromagnetic Calorimeter
Hadron Calorimeter
Superconducting Solenoid
2T
Iron return yoke interspersed with Muon chambers

Event reconstruction

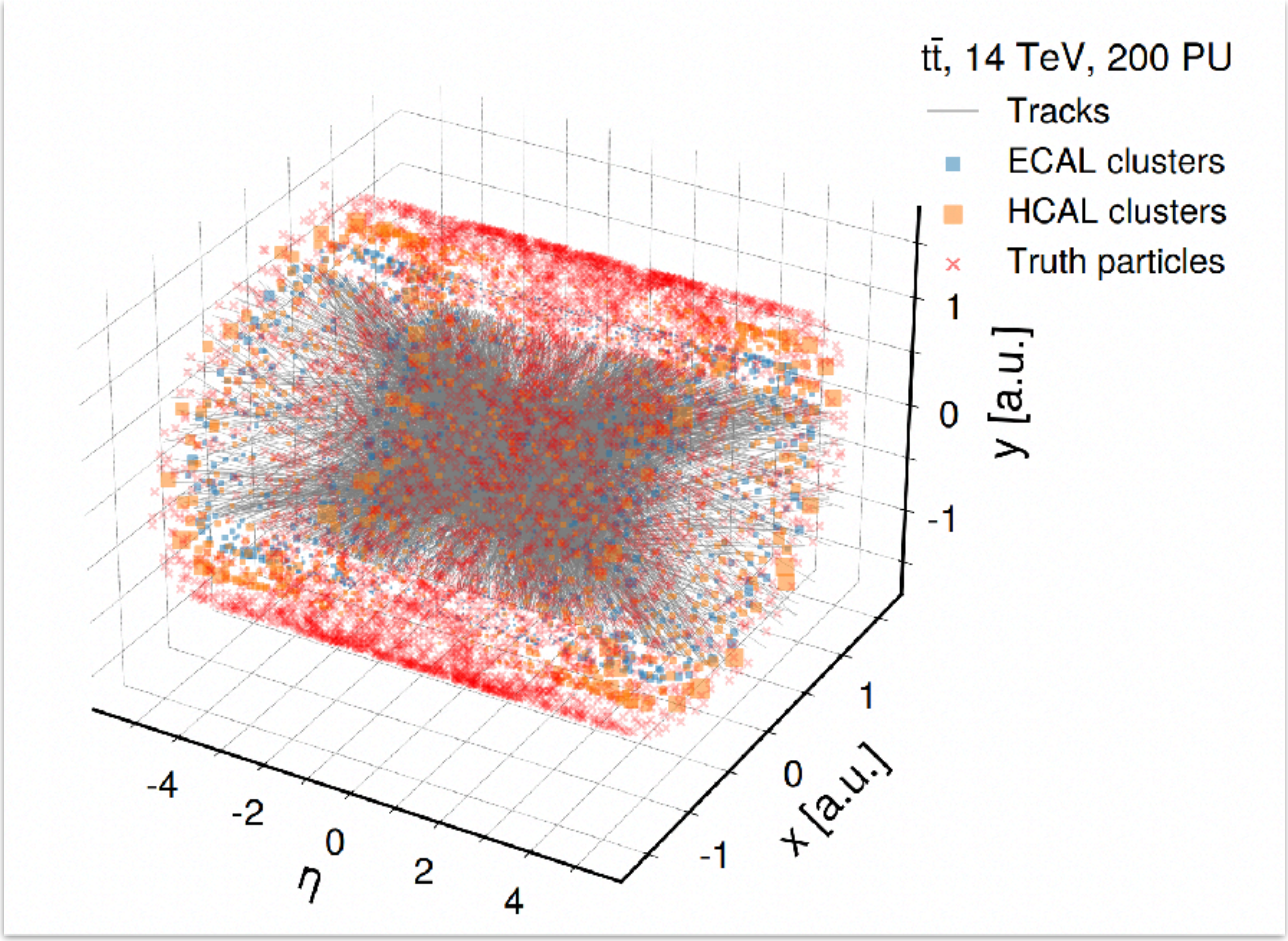
- The Particle Flow algorithm reconstructs particles from tracks and calorimeter energy clusters

Different particles interact differently with the detector subsystems

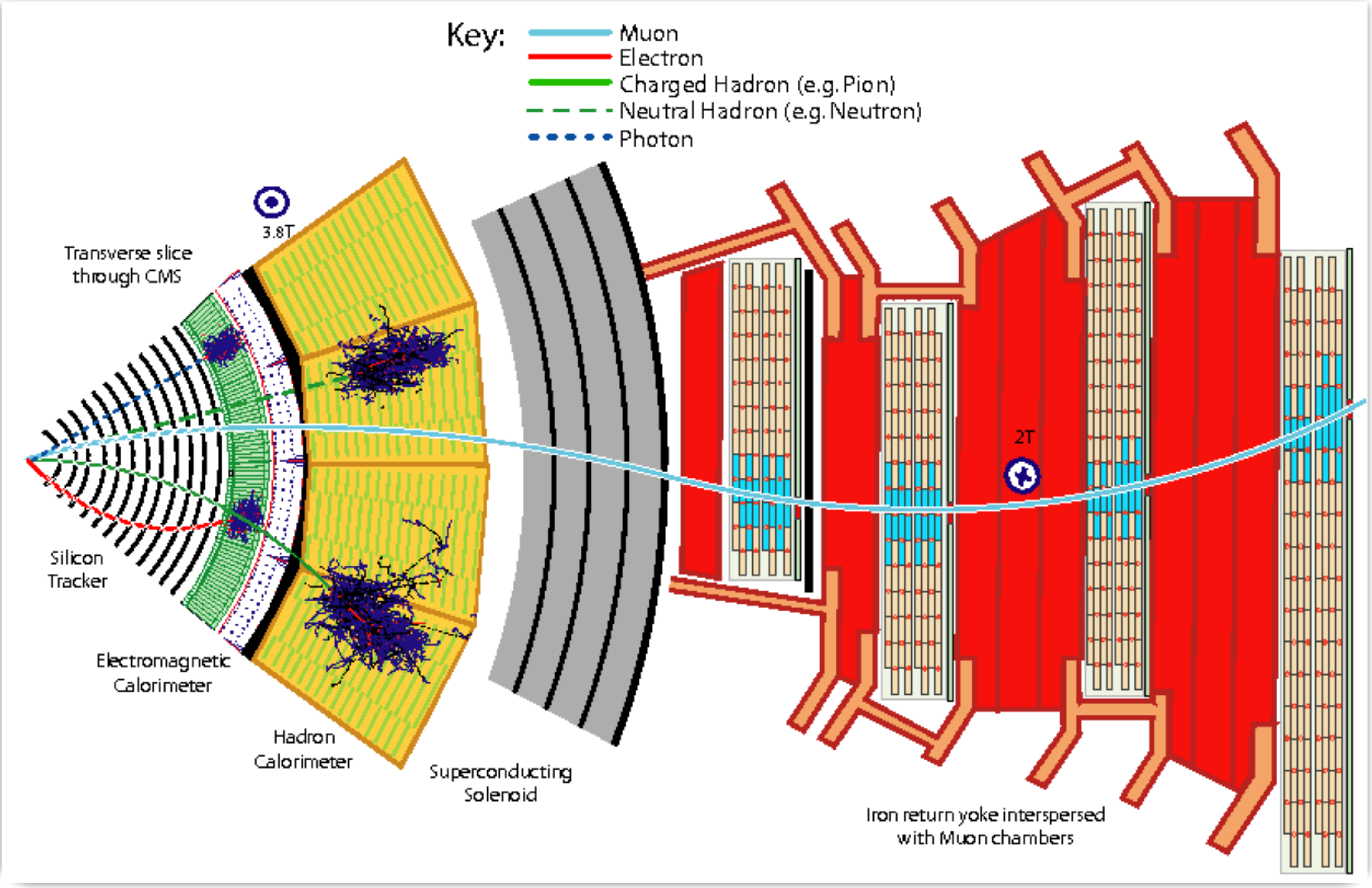

Key:
Muon
Electron
Charged Hadron (e.g. Pion)
Neutral Hadron (e.g. Neutron)
Photon
3.8T
Transverse slice through CMS
Silicon Tracker
Electromagnetic Calorimeter
Hadron Calorimeter
Superconducting Solenoid
2T
Iron return yoke interspersed with Muon chambers

Event reconstruction

- The Particle Flow algorithm reconstructs particles from tracks and calorimeter energy clusters



Different particles interact differently with the detector subsystems

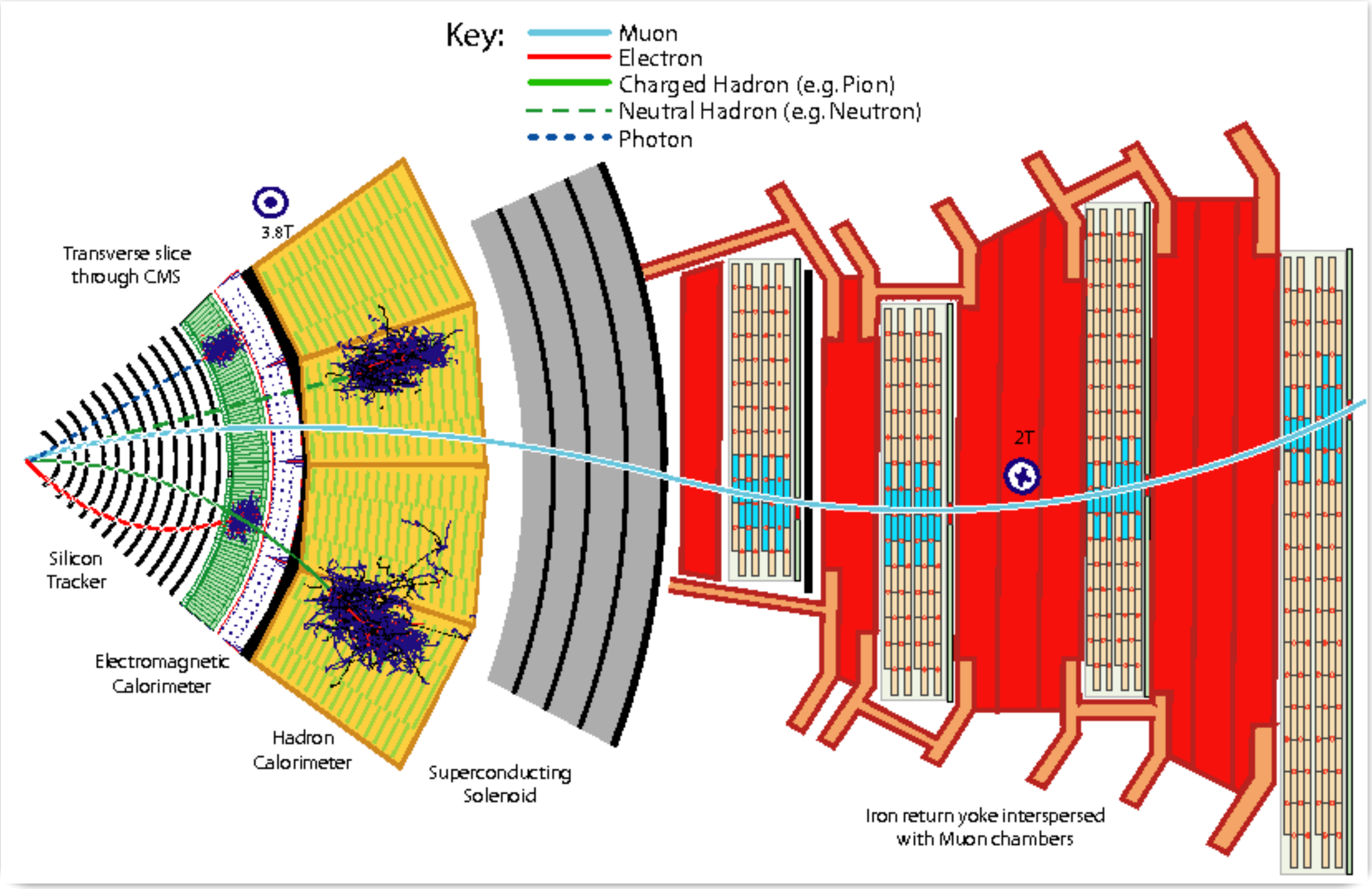


Key:
Muon
Electron
Charged Hadron (e.g. Pion)
Neutral Hadron (e.g. Neutron)
Photon
3.8T
2T
Transverse slice through CMS
Silicon Tracker
Electromagnetic Calorimeter
Hadron Calorimeter
Superconducting Solenoid
Iron return yoke interspersed with Muon chambers

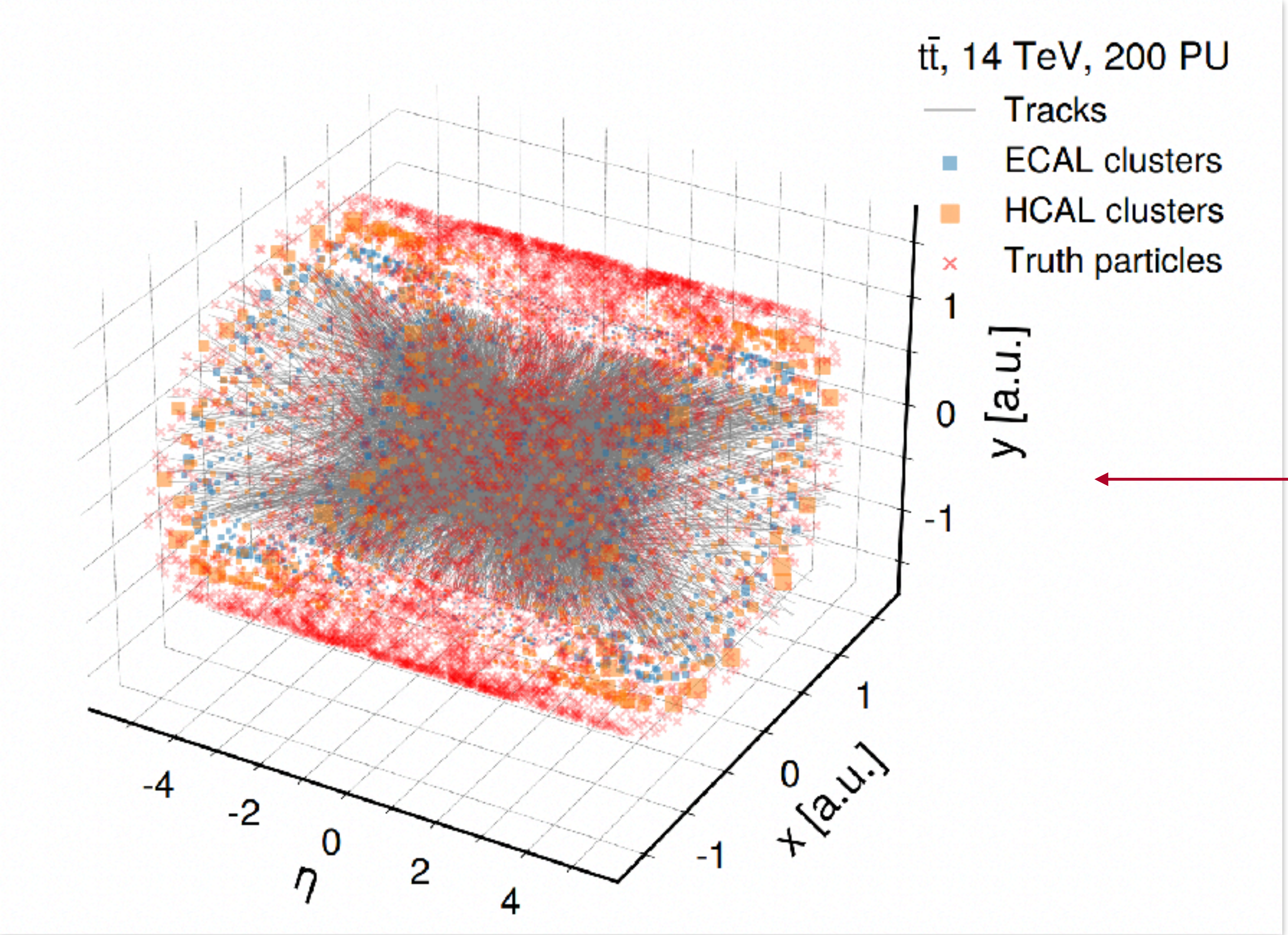
Event reconstruction

- The Particle Flow algorithm reconstructs particles from tracks and calorimeter energy clusters

Different particles interact differently with the detector subsystems



Key:
Muon
Electron
Charged Hadron (e.g. Pion)
Neutral Hadron (e.g. Neutron)
Photon
3.8T
2T
Transverse slice through CMS
Silicon Tracker
Electromagnetic Calorimeter
Hadron Calorimeter
Superconducting Solenoid
Iron return yoke interspersed with Muon chambers

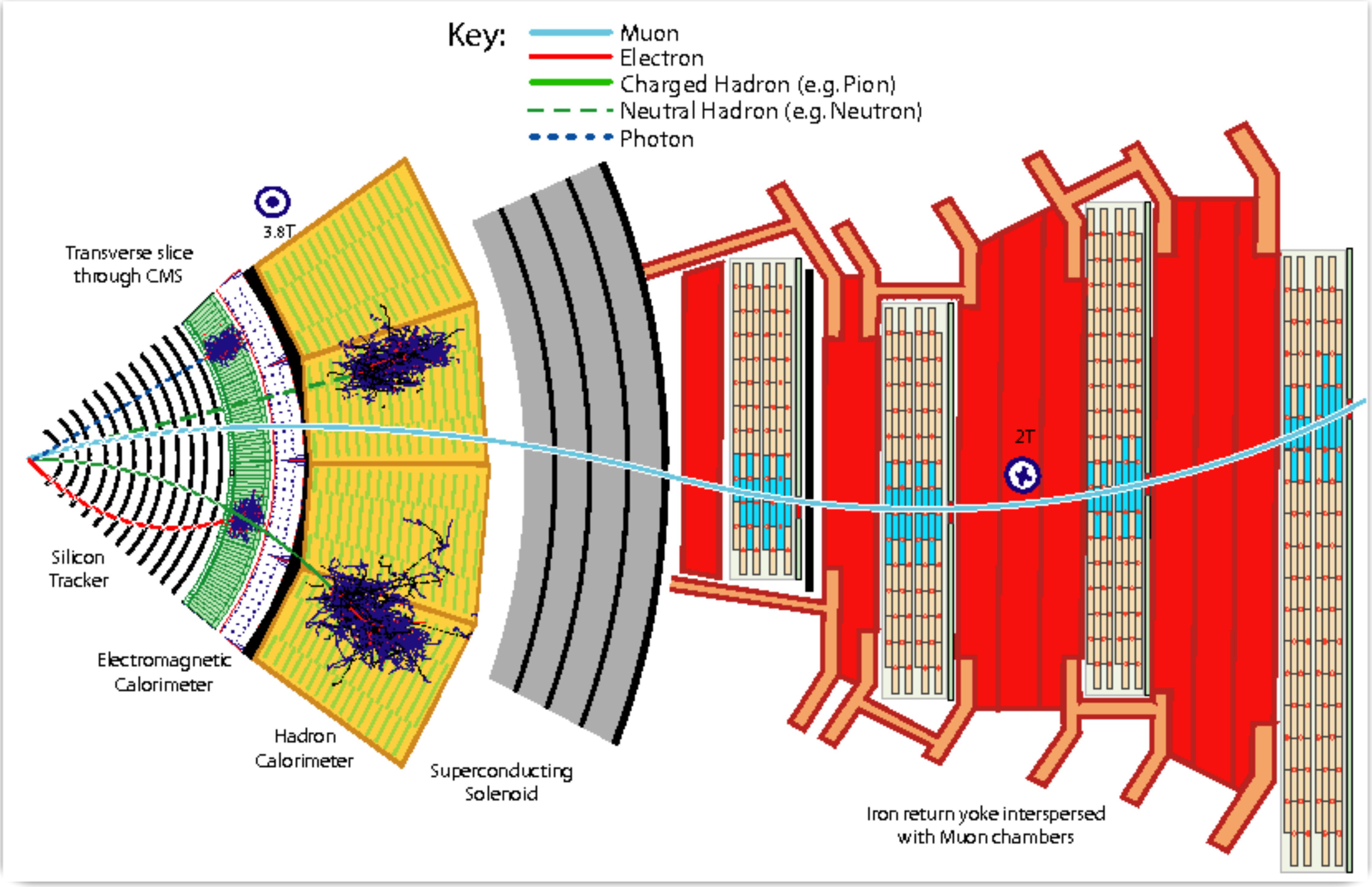


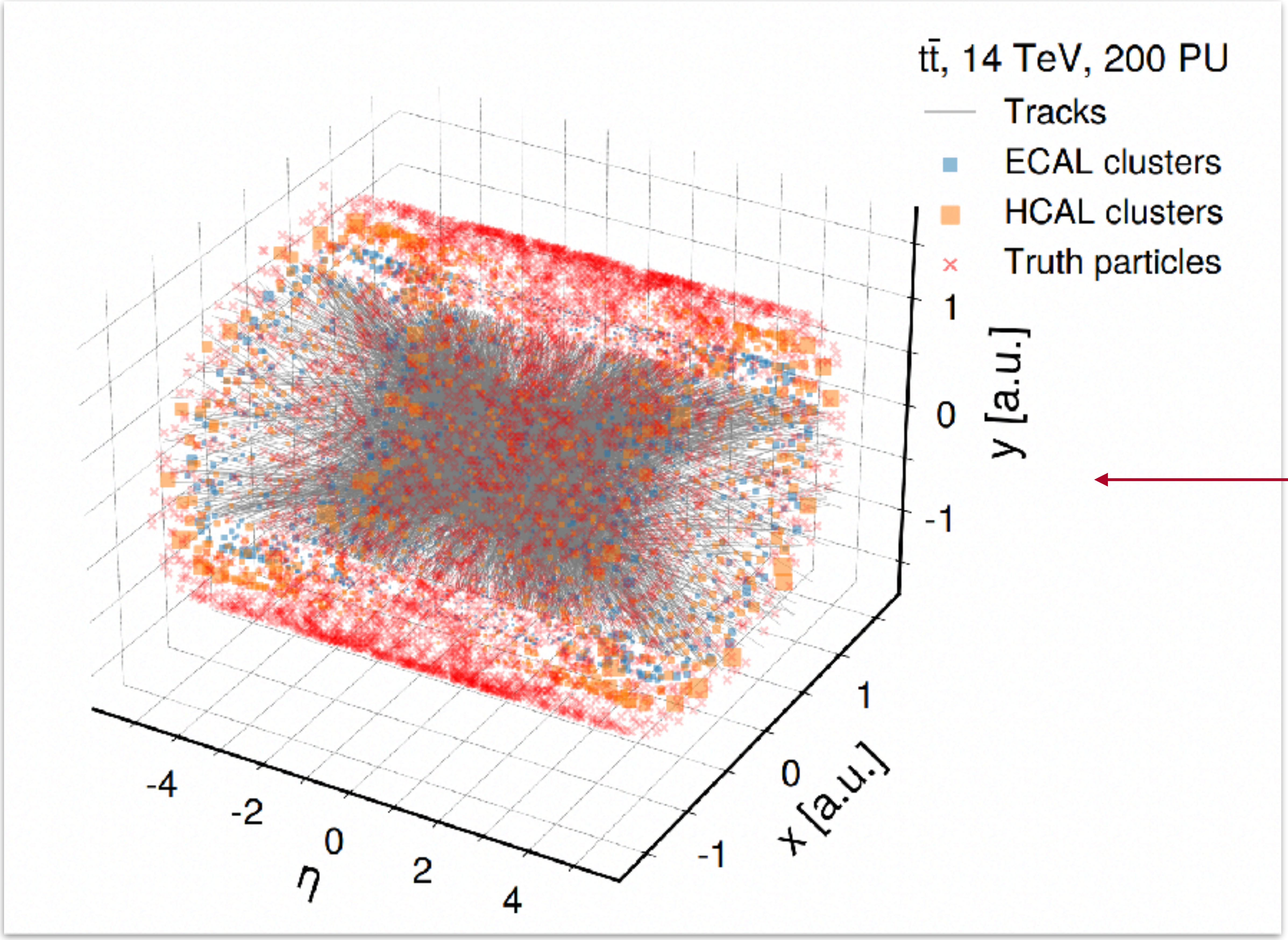
Events are getting busier with higher luminosity and possibly in need for scalable AI-driven algorithms

Event reconstruction

- The Particle Flow algorithm reconstructs particles from tracks and calorimeter energy clusters

Different particles interact differently with the detector subsystems


Key:
Muon
Electron
Charged Hadron (e.g. Pion)
Neutral Hadron (e.g. Neutron)
Photon
3.8T
2T
Transverse slice through CMS
Silicon Tracker
Electromagnetic Calorimeter
Hadron Calorimeter
Superconducting Solenoid
Iron return yoke interspersed with Muon chambers



Events are getting busier with higher luminosity and possibly in need for scalable AI-driven algorithms

- We develop ML Particle Flow (MLPF): a graph neural network that is highly scalable and efficient at reconstruction

- Belayneh, D., Carminati, F., Farbin, A. *et al.* Calorimetry with deep learning: particle simulation and reconstruction for collider physics. *Eur. Phys. J. C* **80,** 688 (2020). https://doi.org/10.1140/epjc/s10052-020-8251-9

- Jan Kieseler, Object condensation: one-stage grid-free multi-object reconstruction in physics detectors, graph, and image data. Eur. Phys. J. C 80, 886 (2020). https://doi.org/10.1140/epjc/s10052-020-08461-2

- Saptaparna Bhattacharya, Nadezda Chernyavskaya, Saranya Ghosh, Lindsey Gray, Jan Kieseler et al. GNN-based end-to-end reconstruction in the CMS Phase 2 High-Granularity Calorimeter. ACAT 2021. https://doi.org/10.48550/arXiv.2203.01189

- Shah Rukh Qasim, Nadezda Chernyavskaya, Jan Kieseler, Kenneth Long, Oleksandr Viazlo et al. End-to-end multi-particle reconstruction in high occupancy imaging calorimeters with graph neural networks. https://doi.org/10.48550/arXiv.2204.01681

- Di Bello, F.A., Ganguly, S., Gross, E. et al. Towards a computer vision particle flow. Eur. Phys. J. C 81, 107 (2021). https://doi.org/10.1140/epjc/s10052-021-08897-0

- **Joosep Pata, Javier Duarte, Jean-Roch Vlimant, Maurizio Pierini & Maria Spiropulu. MLPF: efficient machine-learned particle-flow reconstruction using graph neural networks. Eur. Phys. J. C 81, 381 (2021). https://doi.org/10.1140/epjc/s10052-021-09158-w**

- **Joosep Pata, Javier Duarte, FM, Eric Wulff, Jieun Yoo, Jean-Roch Vlimant, Maurizio Pierini, Maria Girone. Machine Learning for Particle Flow Reconstruction at CMS. ACAT 2021. https://doi.org/10.48550/arXiv.2203.00330, http://cds.cern.ch/record/2792320**

- **FM, Raghav Kansal, Daniel Diaz, JD, Joosep Pata, Maurizio Pierini, Jean-Roch Vlimant. Explaining machine-learned particle-flow reconstruction. Machine Learning for Physical Sciences NeurIPS 2021 workshop. https://arxiv.org/abs/2111.12840**

- Francesco Armando Di Bello, Etienne Dreyer, Sanmay Ganguly, Eilam Gross, Lukas Heinrich, Anna Ivina, Marumi Kado, Nilotpal Kakati, Lorenzo Santi, Jonathan Shlomi, Matteo Tusoni, Reconstructing particles in jets using set transformer and hypergraph prediction networks. https://arxiv.org/abs/2212.01328

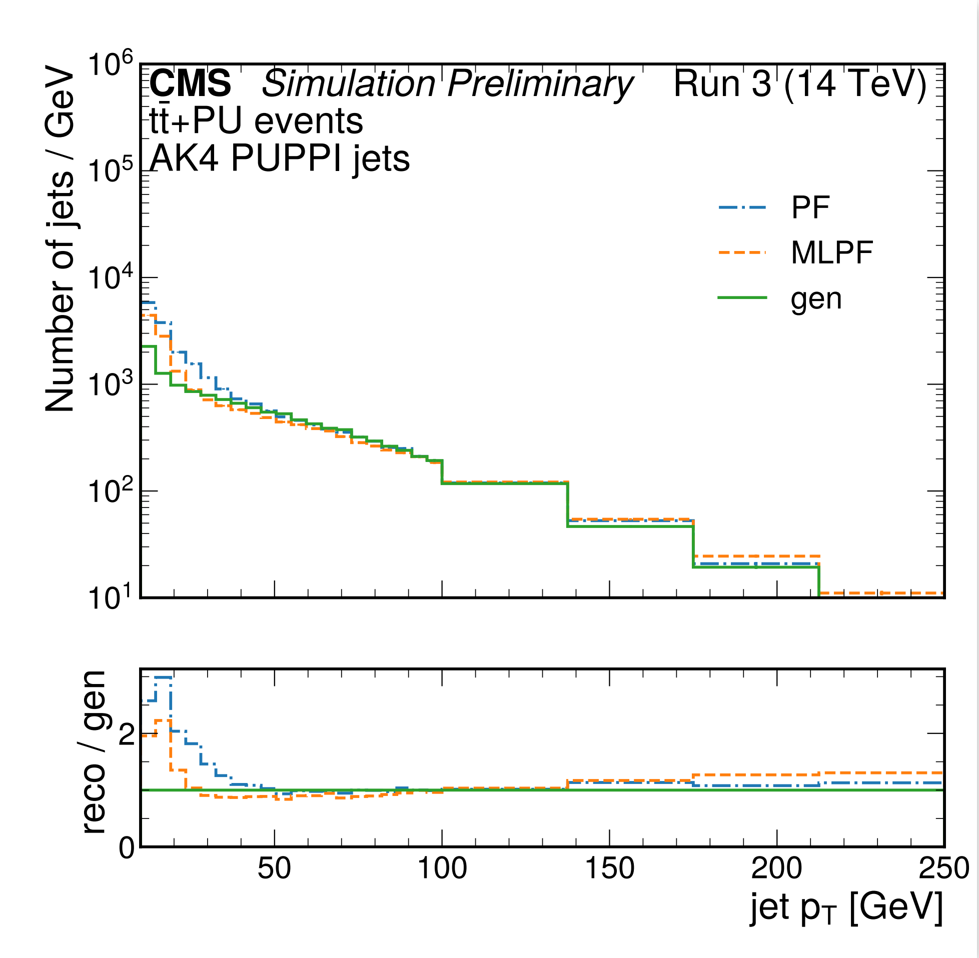
- **FM, Joosep Pata, Javier Duarte, Eric Wulff, Dylan Rankin, Maurizio Pierini, Jean-Roch Vlimant. Progress towards an improved particle flow algorithm at CMS with machine learning. ACAT 2022 and ML4Jets 2022. https://arxiv.org/abs/2303.17657, http://cds.cern.ch/record/2842375**

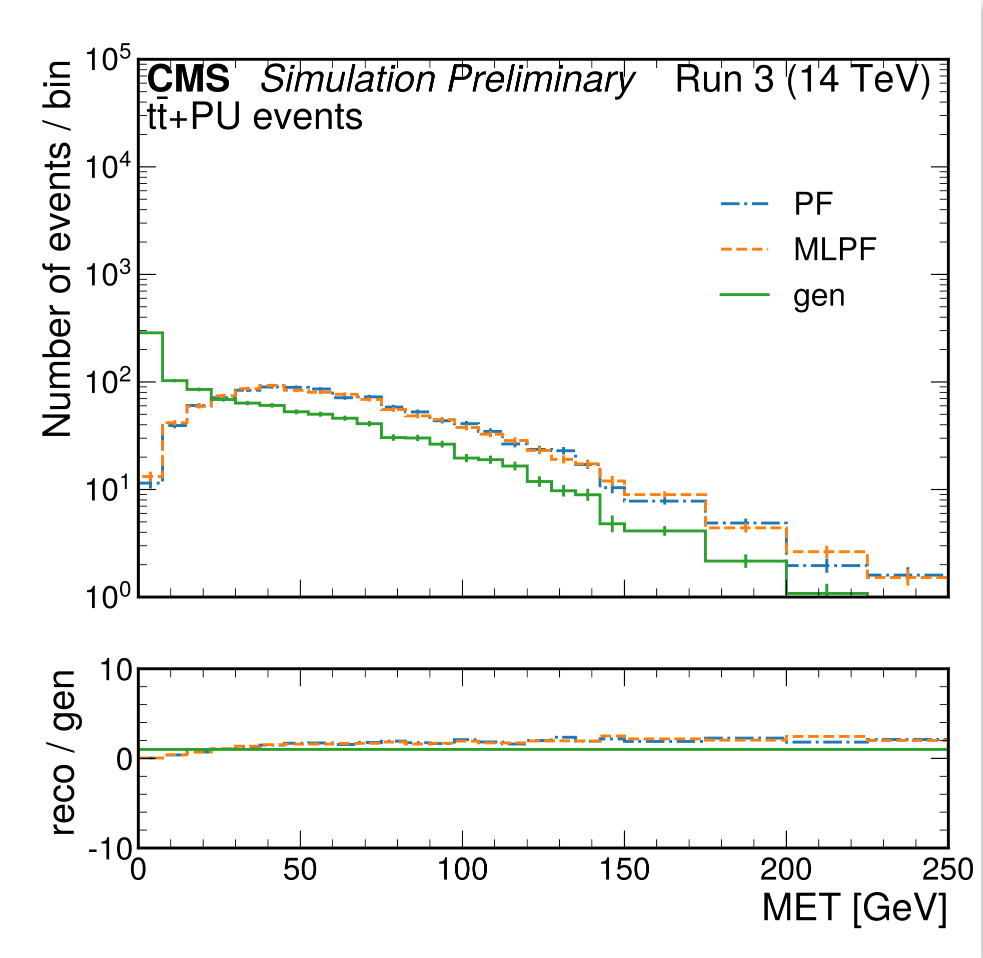
- **Joosep Pata, Eric Wulff, FM, David Southwick, Mengke Zhang, Maria Girone, JD. Scalable neural network models and terascale datasets for particle-flow reconstruction. ML4Jets 2023. https://arxiv.org/abs/2309.06782**

ML-based reconstruction is an active area of research. In the interest of time, I will focus on the recent MLPF developments that I'm more familiar with.

MLPF in CMS

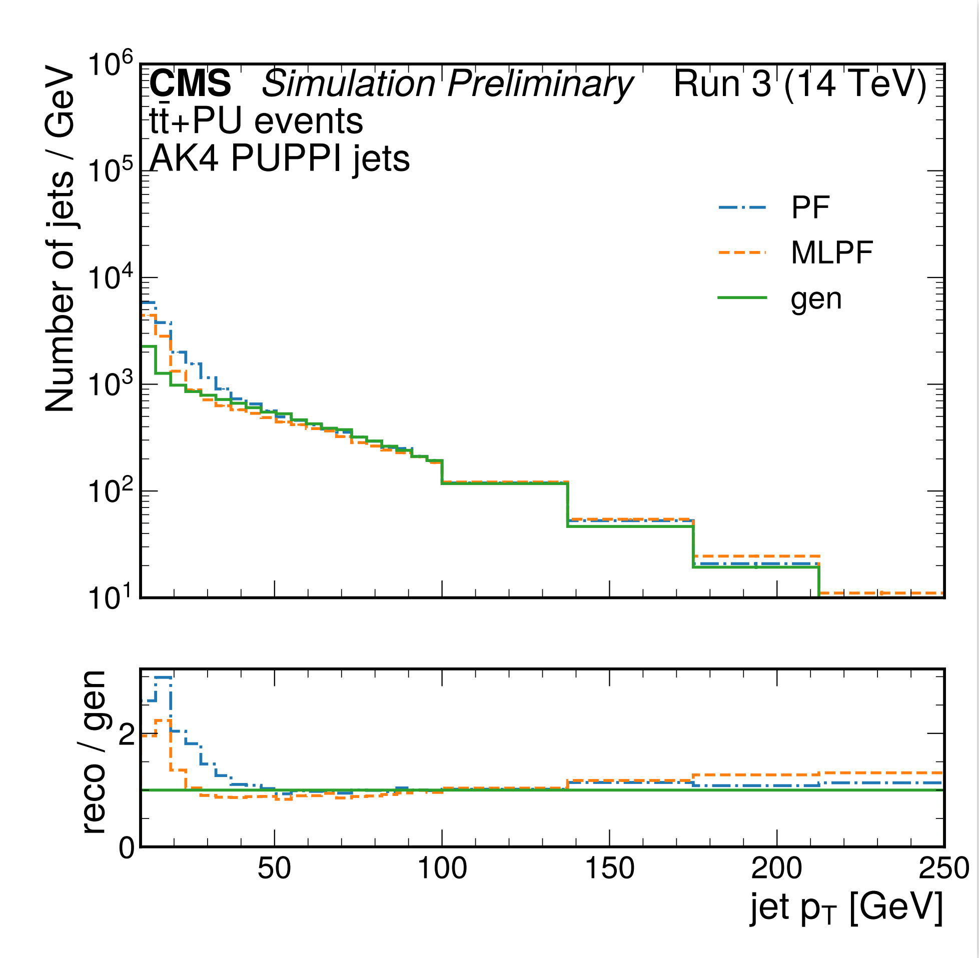
The last CMS results were published and presented in ACAT 2022[1]

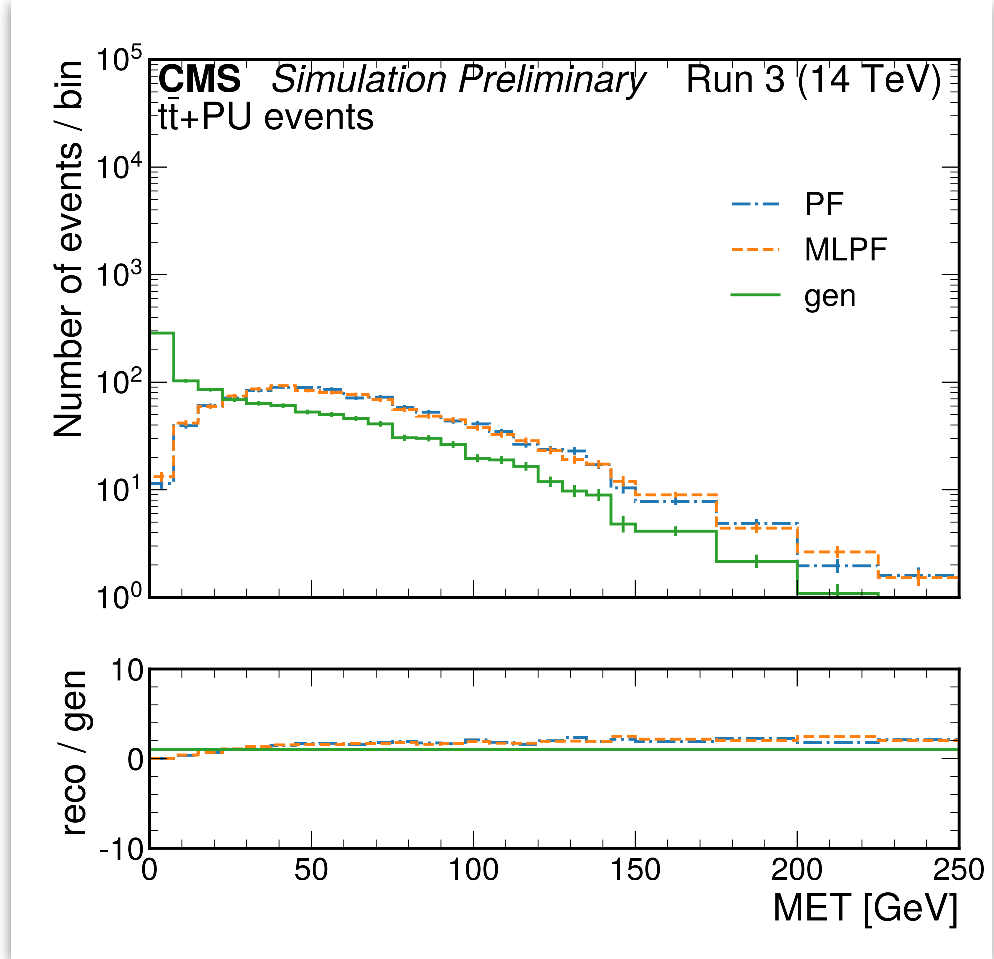




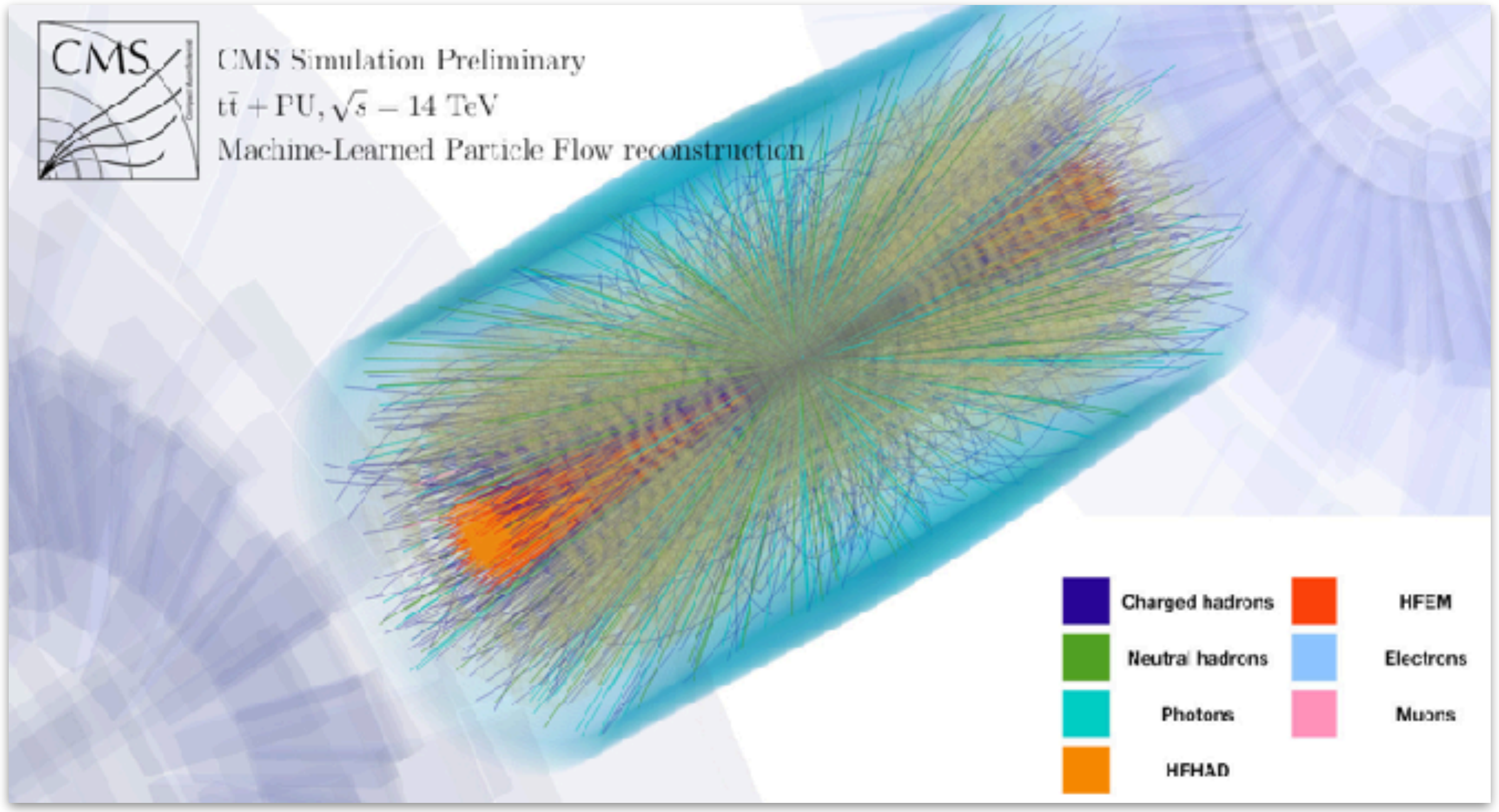
[1] https://arxiv.org/abs/2303.17657 [2] https://github.com/cms-sw/cmssw/pull/36841

MLPF in CMS

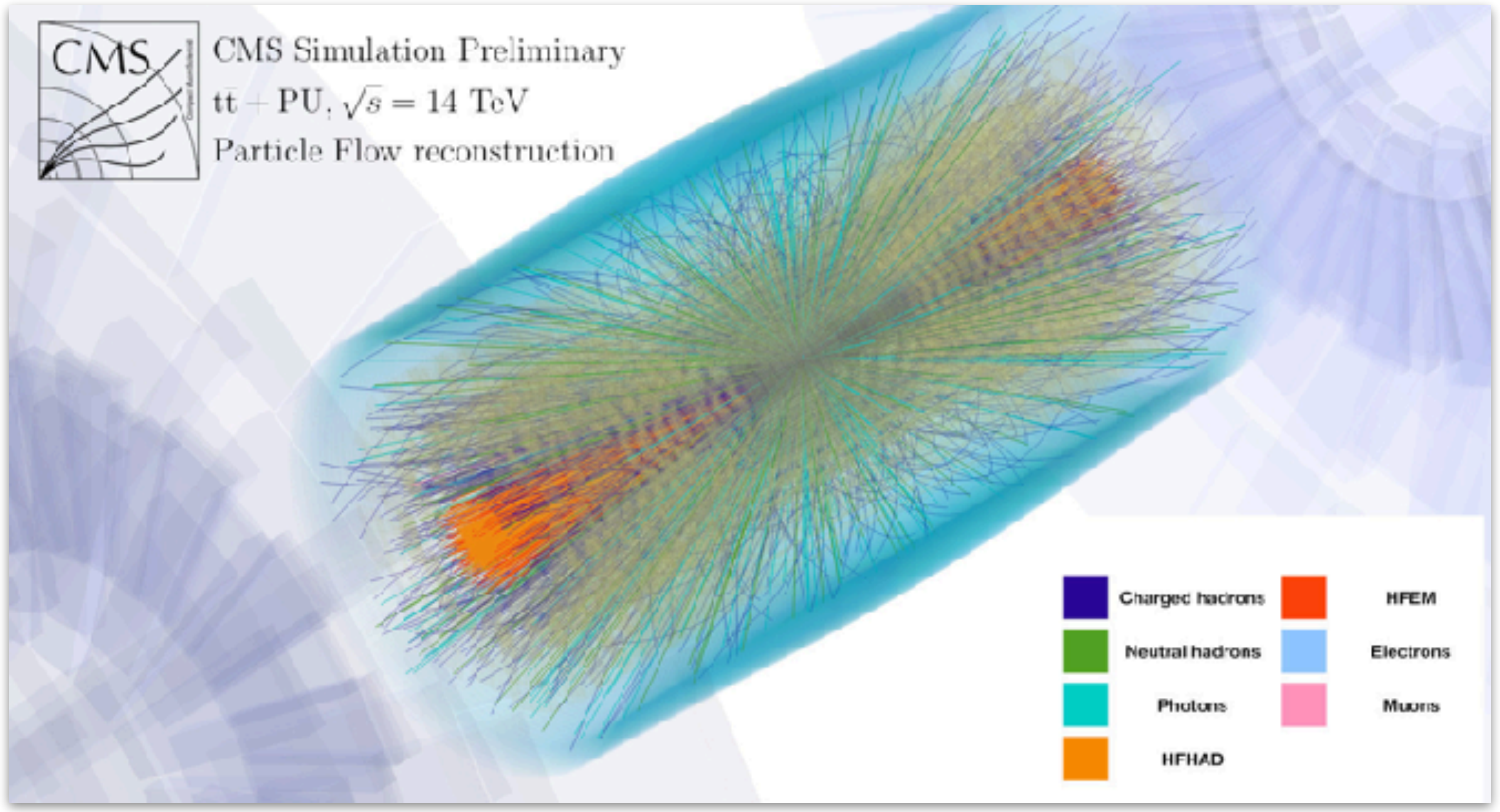




The last CMS results were published and presented in ACAT 2022[1]

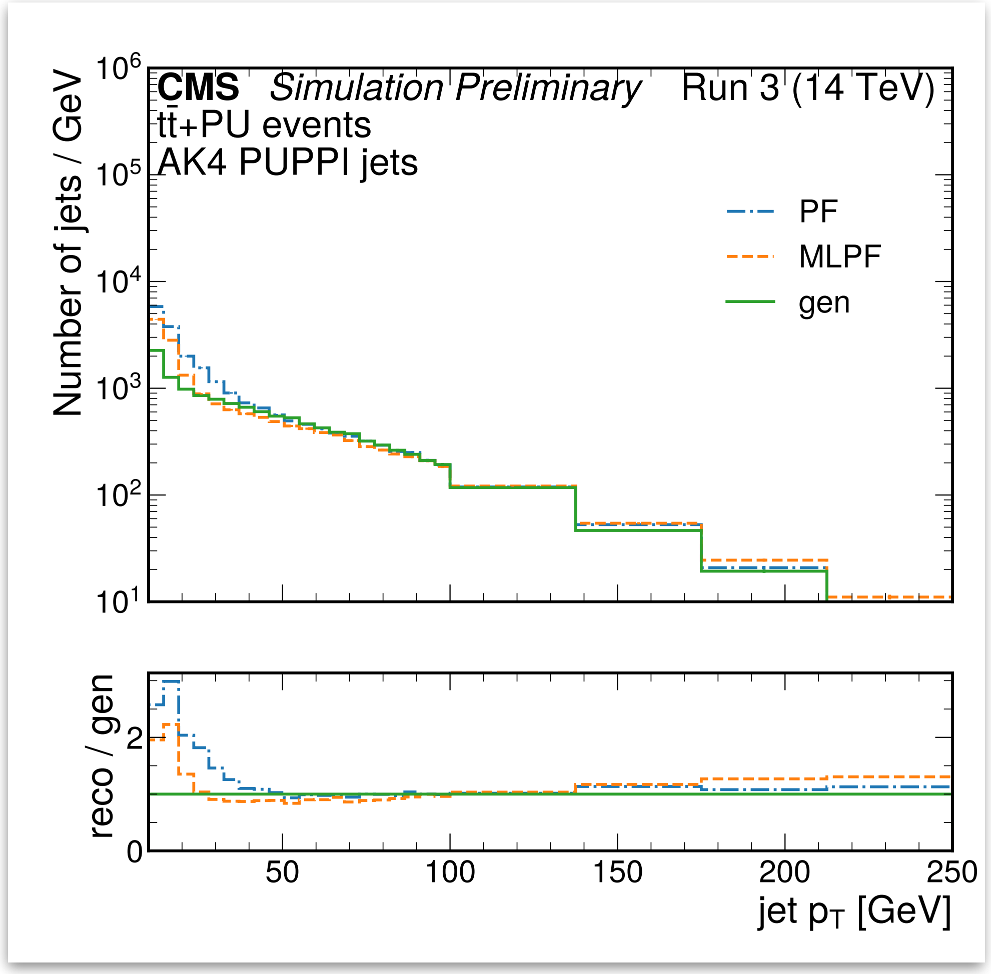

CMS
CMS Simulation Preliminary
$t\bar{t}$ + PU, $\sqrt{s}$ = 14 TeV
Machine-Learned Particle Flow reconstruction
Charged hadrons
HFEM
Neutral hadrons
Electrons
Photons
Muons
HFHAD

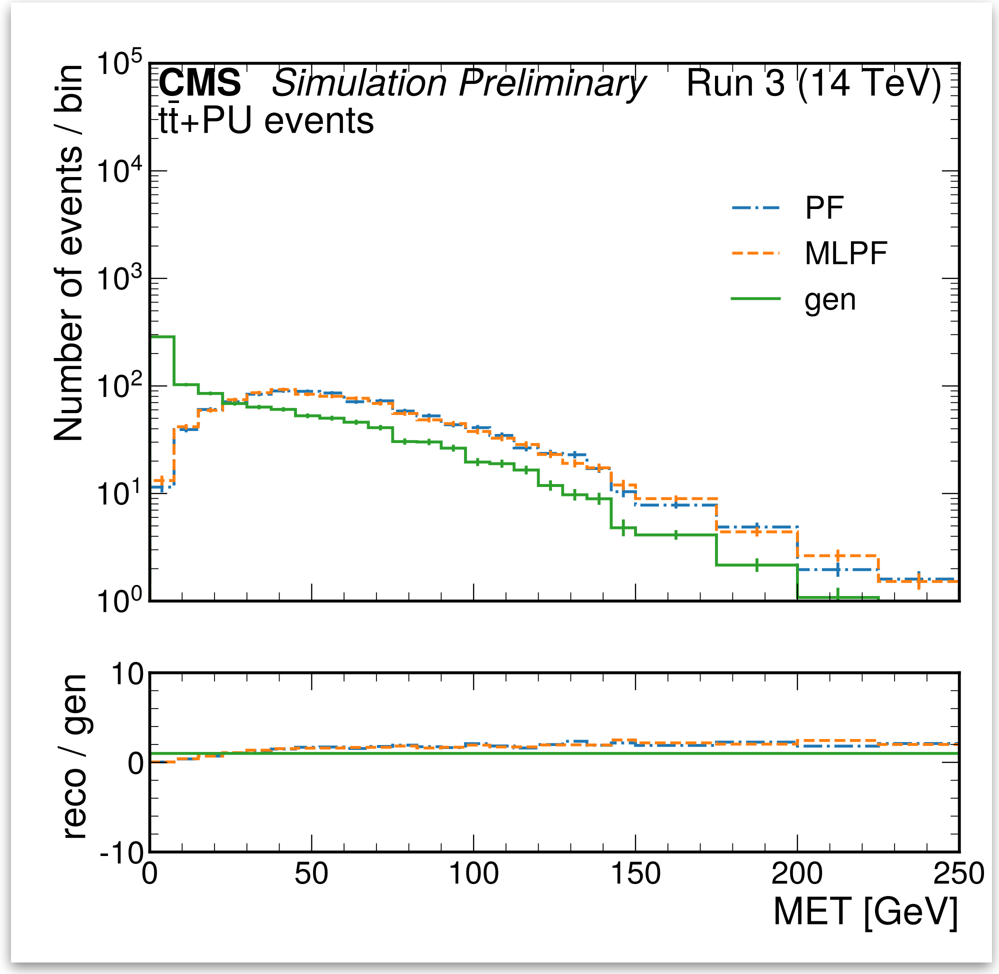
MLPF shows comparable event-level reconstruction to native PF


CMS
CMS Simulation Preliminary
$t\bar{t}$ + PU, $\sqrt{s}$ = 14 TeV
Particle Flow reconstruction
Charged hadrons
HFEM
Neutral hadrons
Electrons
Photons
Muons
HFHAD

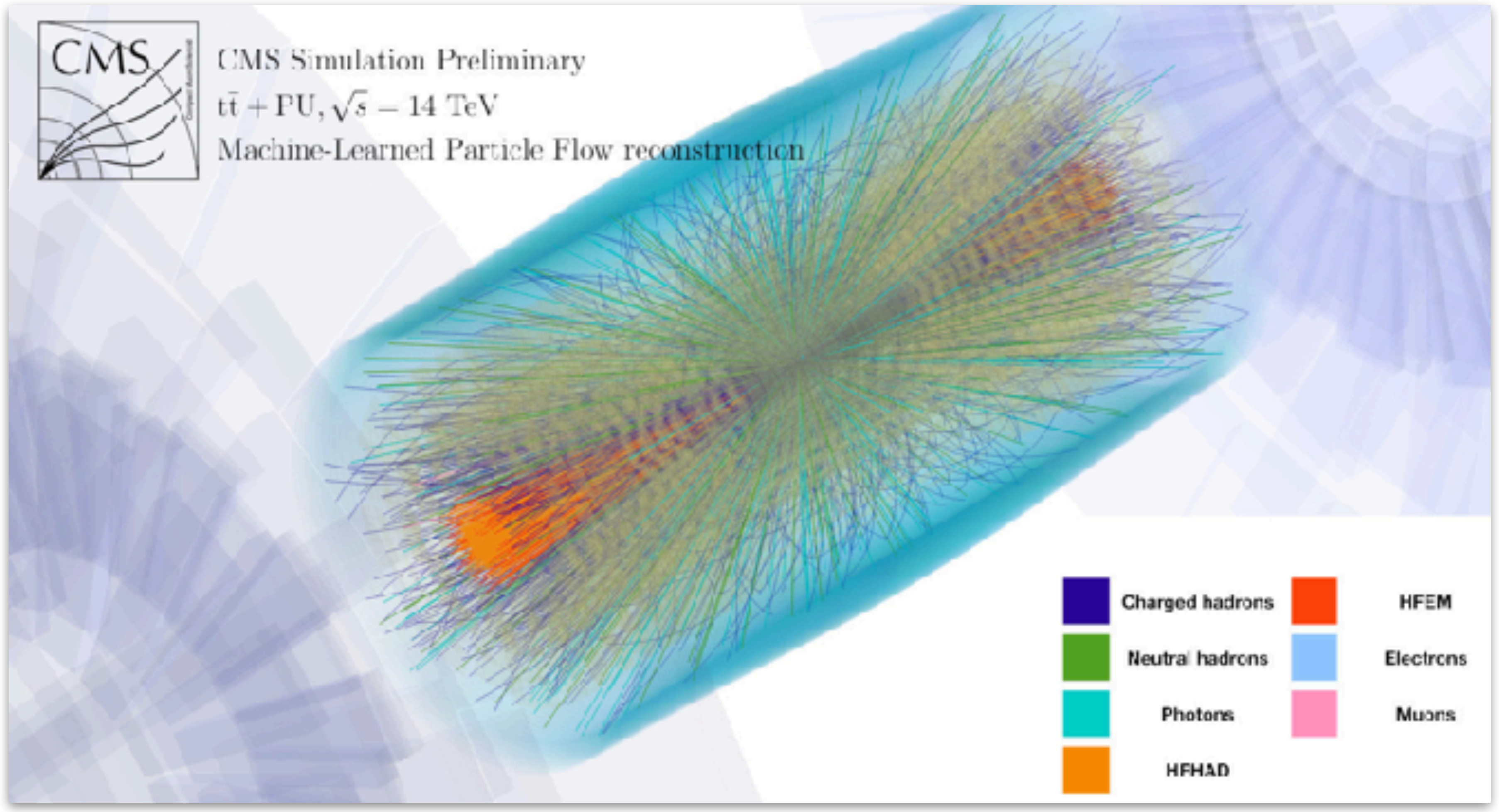
[1] https://arxiv.org/abs/2303.17657 [2] https://github.com/cms-sw/cmssw/pull/36841

MLPF in CMS

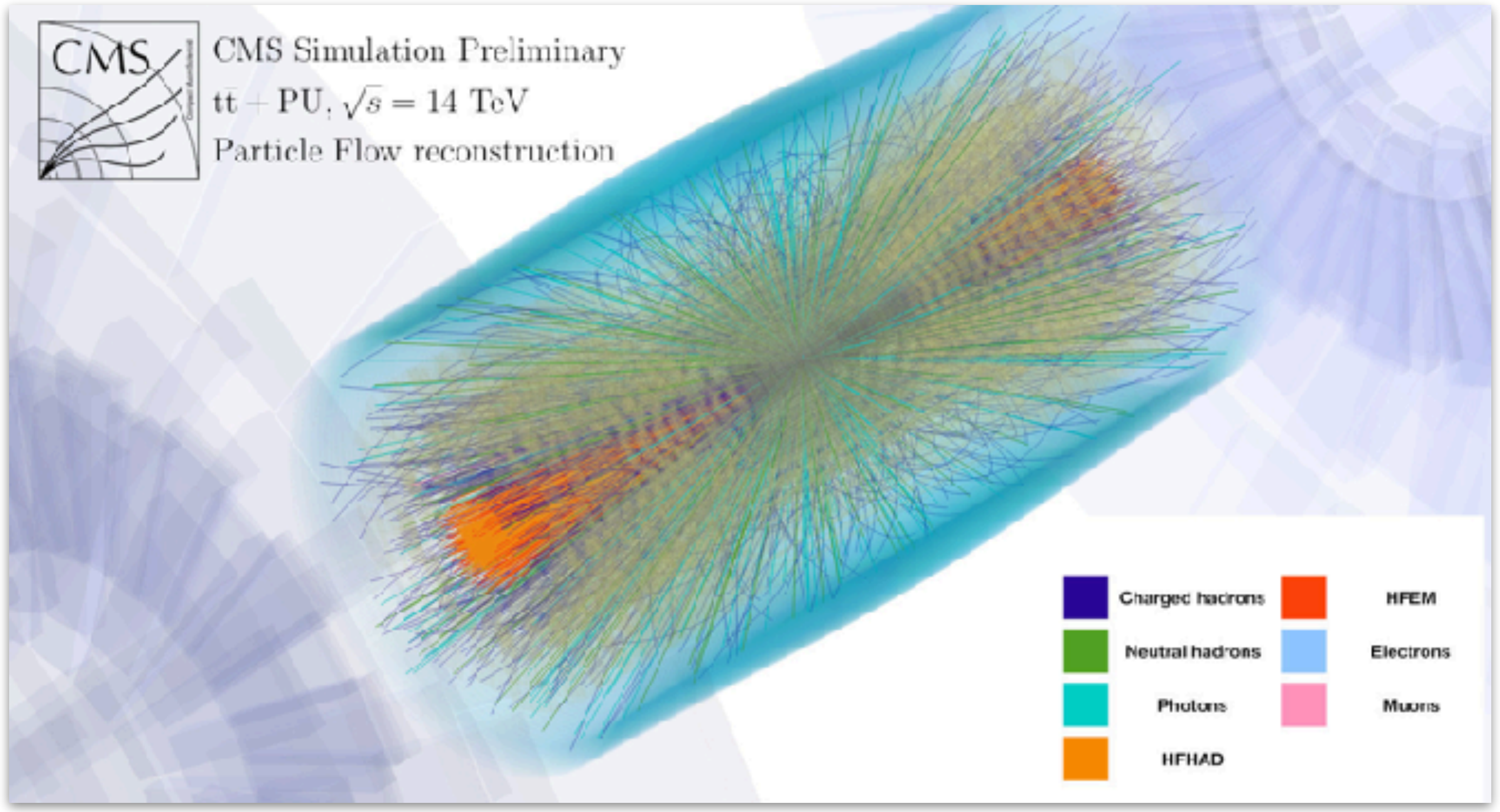




The last CMS results were published and presented in ACAT 2022[1]


CMS
CMS Simulation Preliminary
$t\bar{t}$ + PU, $\sqrt{s}$ = 14 TeV
Machine-Learned Particle Flow reconstruction
Charged hadrons
Neutral hadrons
Photons
HFHAD
HFEM
Electrons
Muons

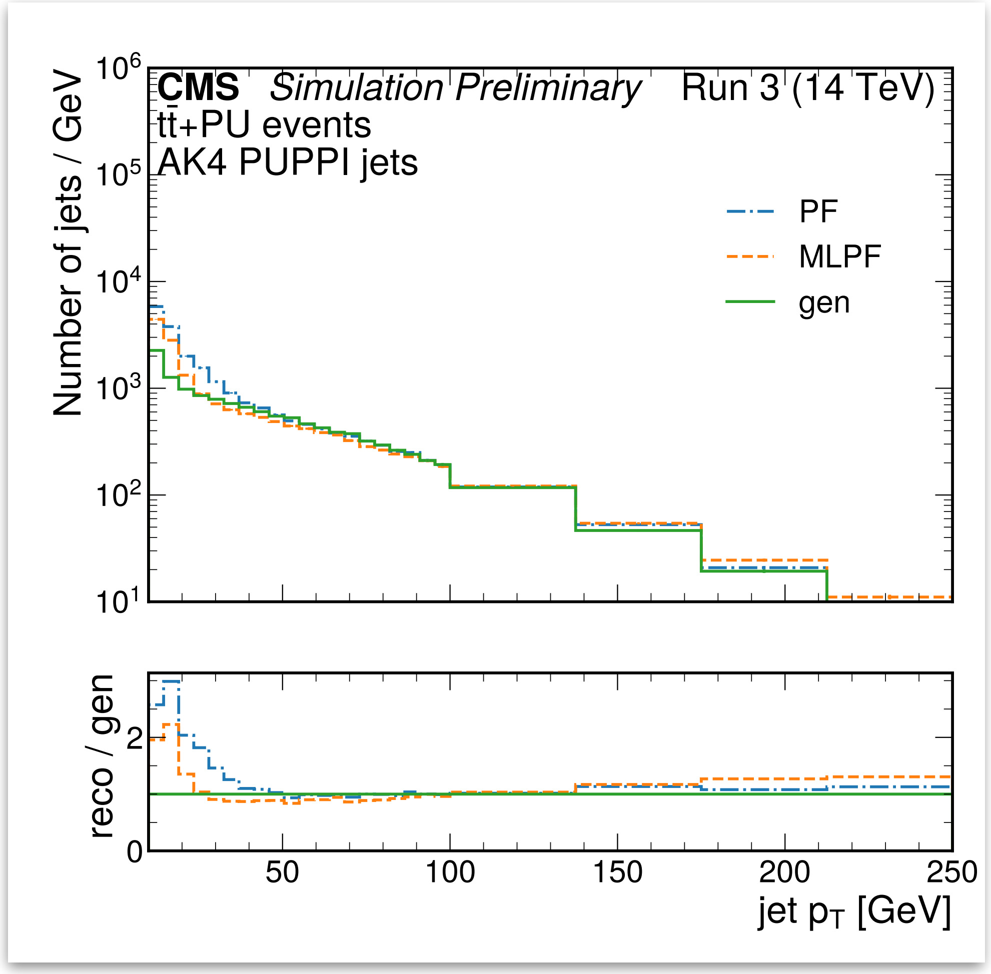
MLPF shows comparable event-level reconstruction to native PF

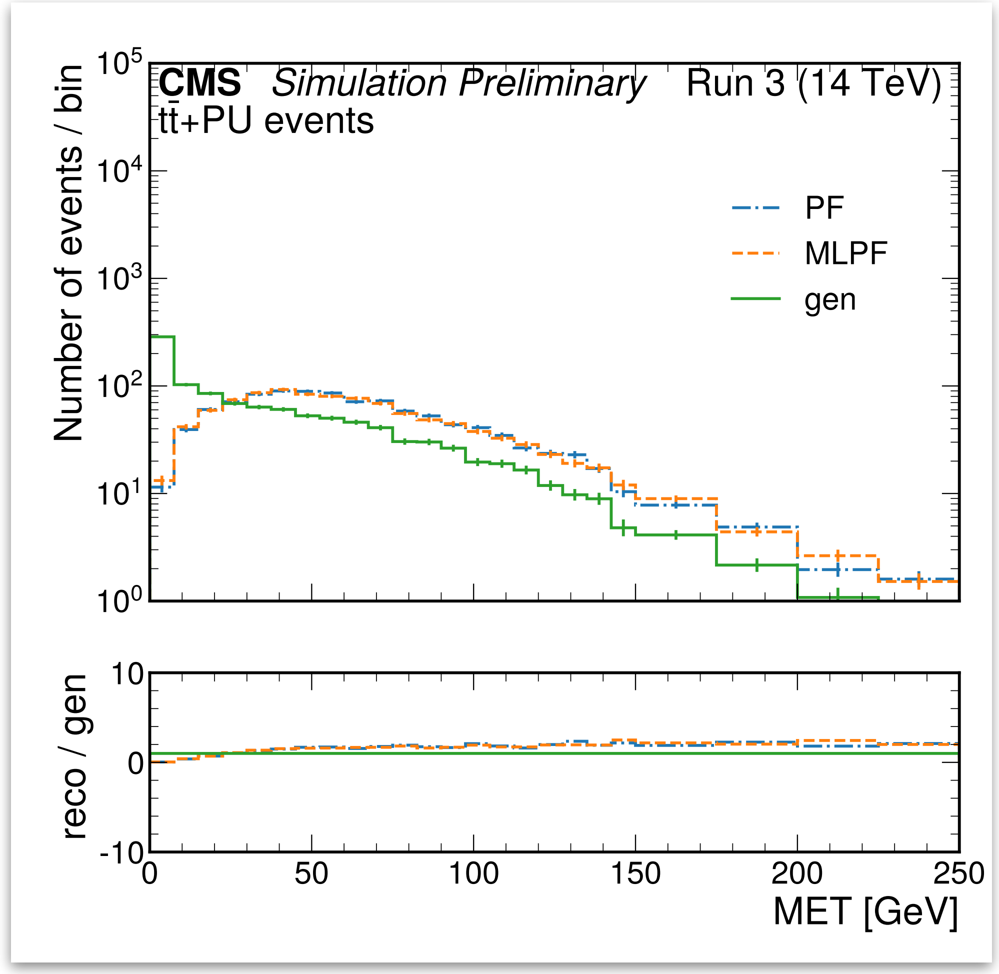

CMS
CMS Simulation Preliminary
$t\bar{t}$ + PU, $\sqrt{s}$ = 14 TeV
Particle Flow reconstruction
Charged hadrons
Neutral hadrons
Photons
HFHAD
HFEM
Electrons
Muons

- MLPF is currently integrated in CMSSW and can be enabled with a flag to switch out standard PF as a testbed[2]

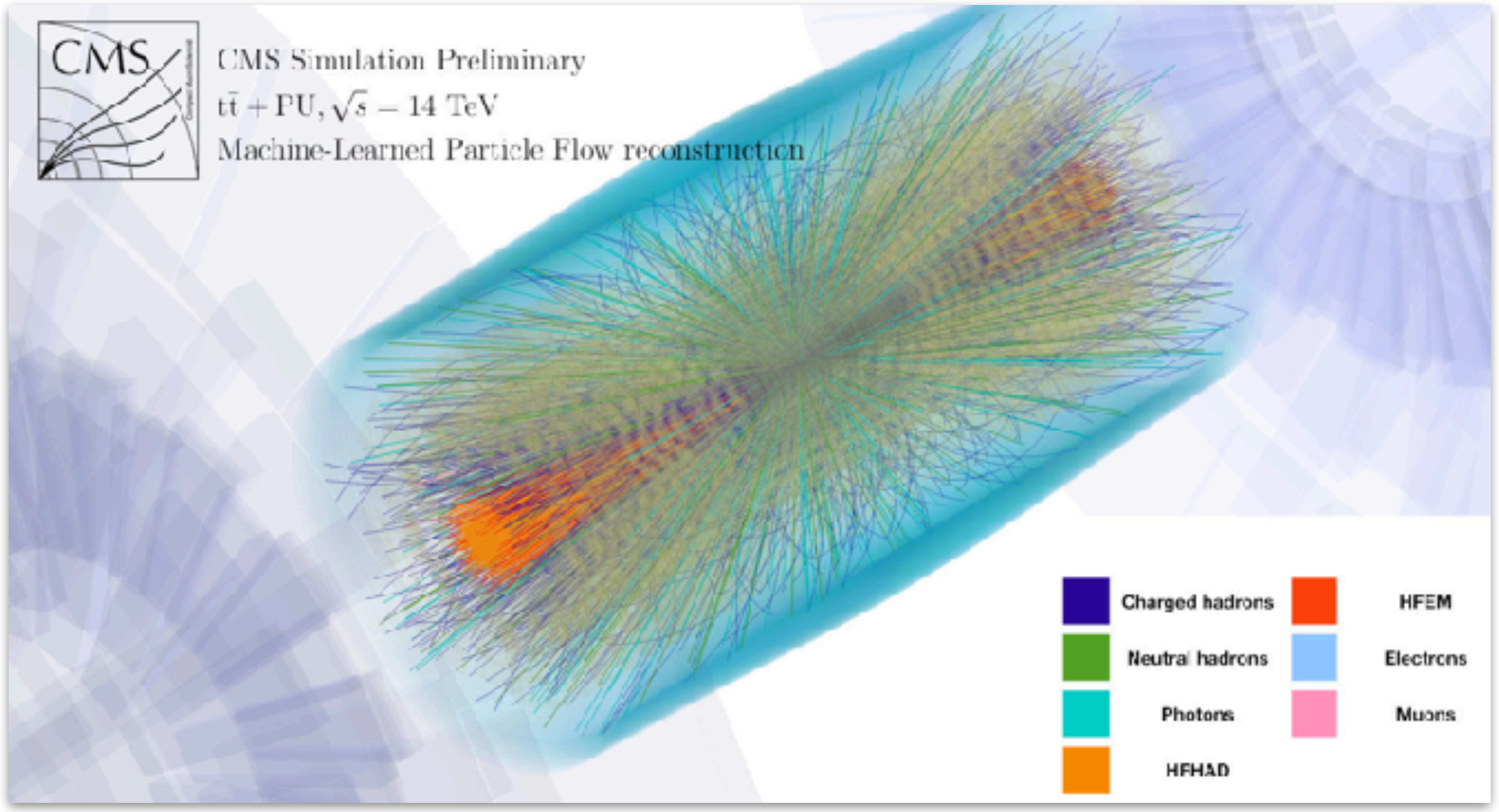
[1] https://arxiv.org/abs/2303.17657 [2] https://github.com/cms-sw/cmssw/pull/36841

MLPF in CMS

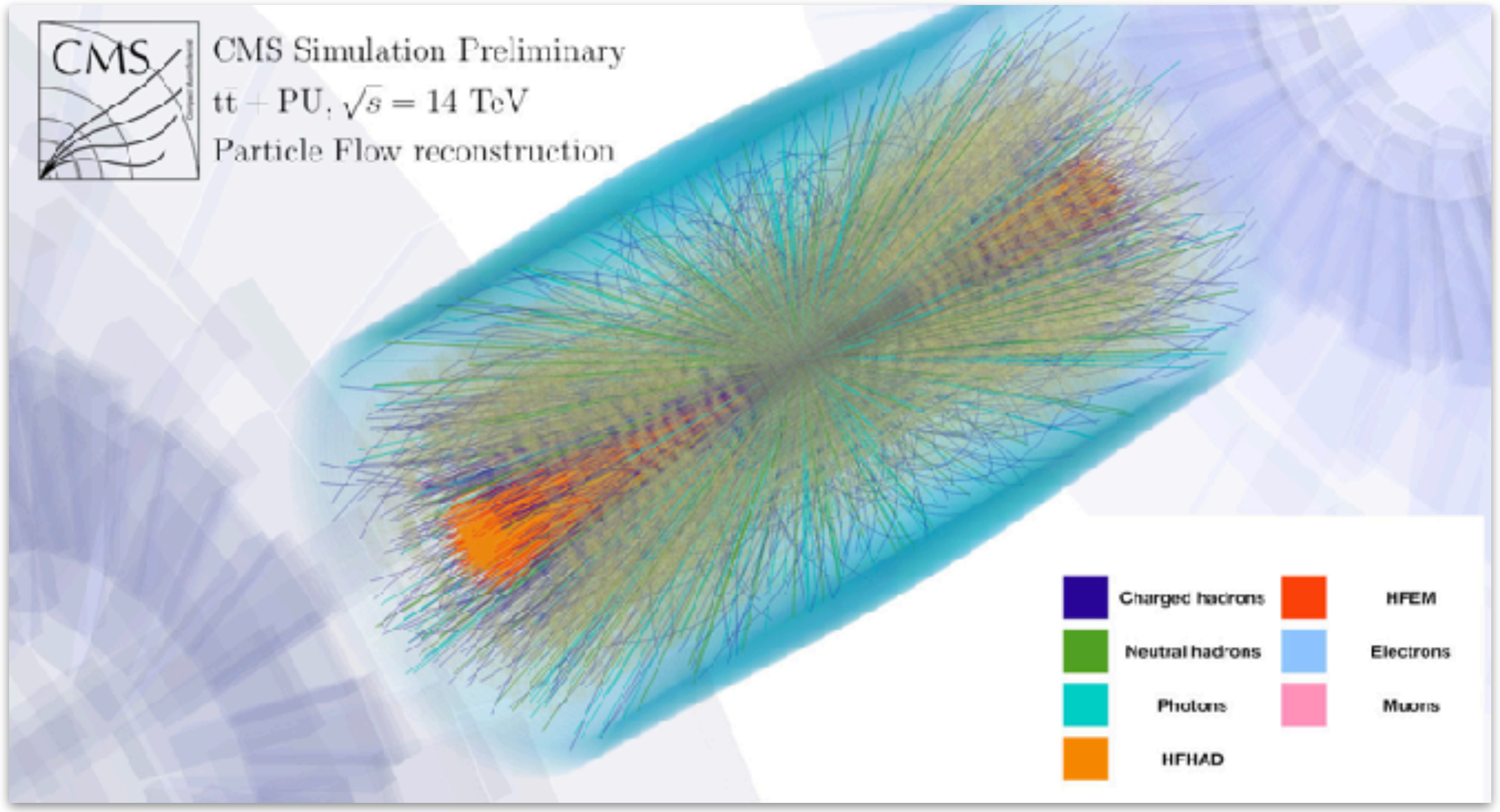




The last CMS results were published and presented in ACAT 2022[1]



MLPF shows comparable event-level reconstruction to native PF



- MLPF is currently integrated in CMSSW and can be enabled with a flag to switch out standard PF as a testbed[2]

- But why MLPF? Scalability and portability!

[1] https://arxiv.org/abs/2303.17657 [2] https://github.com/cms-sw/cmssw/pull/36841

Latest developments

- Performed studies on a newly developed multi-terabyte dataset of high-energy e^+e^- collision simulated with GEANT4 and stored in EDM4HEP format[1]

[1] https://zenodo.org/record/8260741

Latest developments

- Performed studies on a newly developed multi-terabyte dataset of high-energy e^+e^- collision simulated with GEANT4 and stored in EDM4HEP format[1]

- Developed state-of-the-art ML models that exceed the current physics performance of the baseline PF

[1] https://zenodo.org/record/8260741

Latest developments

- Performed studies on a newly developed multi-terabyte dataset of high-energy e^+e^- collision simulated with GEANT4 and stored in EDM4HEP format[1]
- Developed state-of-the-art ML models that exceed the current physics performance of the baseline PF
- Studied the models extensively on several supercomputers with different accelerator chip architectures

[1] https://zenodo.org/record/8260741

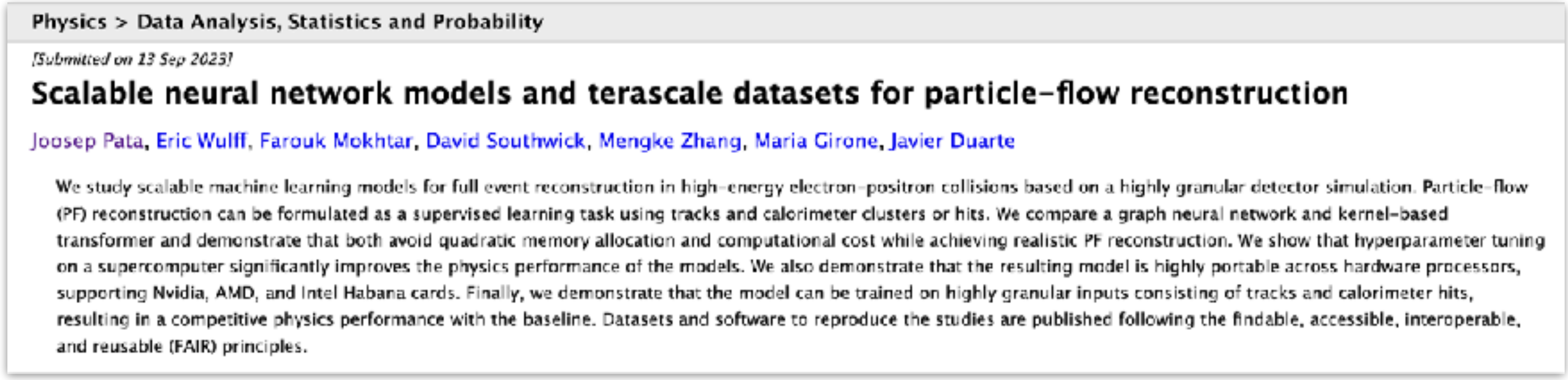
Latest developments

- Performed studies on a newly developed multi-terabyte dataset of high-energy e^+e^- collision simulated with GEANT4 and stored in EDM4HEP format[1]
- Developed state-of-the-art ML models that exceed the current physics performance of the baseline PF
- Studied the models extensively on several supercomputers with different accelerator chip architectures
- Studied the models also on a highly granular dataset composed directly of tracks and calorimeter hits

[1] https://zenodo.org/record/8260741

Latest developments

- Performed studies on a newly developed multi-terabyte dataset of high-energy e^+e^- collision simulated with GEANT4 and stored in EDM4HEP format[1]
- Developed state-of-the-art ML models that exceed the current physics performance of the baseline PF
- Studied the models extensively on several supercomputers with different accelerator chip architectures
- Studied the models also on a highly granular dataset composed directly of tracks and calorimeter hits



Physics > Data Analysis, Statistics and Probability

[Submitted on 13 Sep 2023]

Scalable neural network models and terascale datasets for particle-flow reconstruction

Joosep Pata, Eric Wulff, Farouk Mokhtar, David Southwick, Mengke Zhang, Maria Girone, Javier Duarte

We study scalable machine learning models for full event reconstruction in high-energy electron-positron collisions based on a highly granular detector simulation. Particle-flow (PF) reconstruction can be formulated as a supervised learning task using tracks and calorimeter clusters or hits. We compare a graph neural network and kernel-based transformer and demonstrate that both avoid quadratic memory allocation and computational cost while achieving realistic PF reconstruction. We show that hyperparameter tuning on a supercomputer significantly improves the physics performance of the models. We also demonstrate that the resulting model is highly portable across hardware processors, supporting Nvidia, AMD, and Intel Habana cards. Finally, we demonstrate that the model can be trained on highly granular inputs consisting of tracks and calorimeter hits, resulting in a competitive physics performance with the baseline. Datasets and software to reproduce the studies are published following the findable, accessible, interoperable, and reusable (FAIR) principles.

→ Now on arXiv:2309.06782

[1] https://zenodo.org/record/8260741

Compact Linear Collider (CLIC)

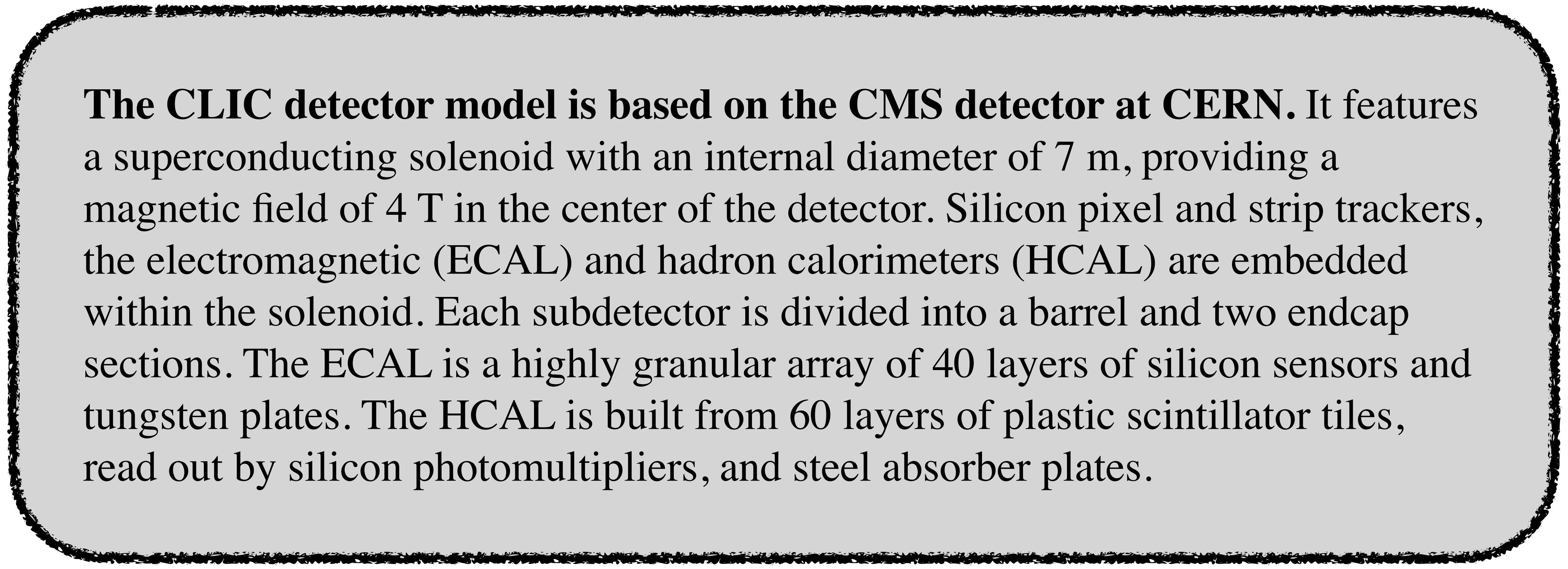
- CLIC dataset is an open dataset of $\mathrm{e}^{+}\mathrm{e}^{-}$collision events at a center of mass energy $\sqrt{s} = 380\mathrm{GeV}$ with full GEANT4 simulation available in the EDM4HEP format

The CLIC detector model is based on the CMS detector at CERN. It features a superconducting solenoid with an internal diameter of 7 m, providing a magnetic field of 4 T in the center of the detector. Silicon pixel and strip trackers, the electromagnetic (ECAL) and hadron calorimeters (HCAL) are embedded within the solenoid. Each subdetector is divided into a barrel and two endcap sections. The ECAL is a highly granular array of 40 layers of silicon sensors and tungsten plates. The HCAL is built from 60 layers of plastic scintillator tiles, read out by silicon photomultipliers, and steel absorber plates.

Compact Linear Collider (CLIC)

- CLIC dataset is an open dataset of e^+e^-collision events at a center of mass energy $\sqrt{s} = 380\mathrm{GeV}$ with full GEANT4 simulation available in the EDM4HEP format

- Why CLIC? Publicly available, well documented, realistic, and smaller input multiplicity compared to CMS



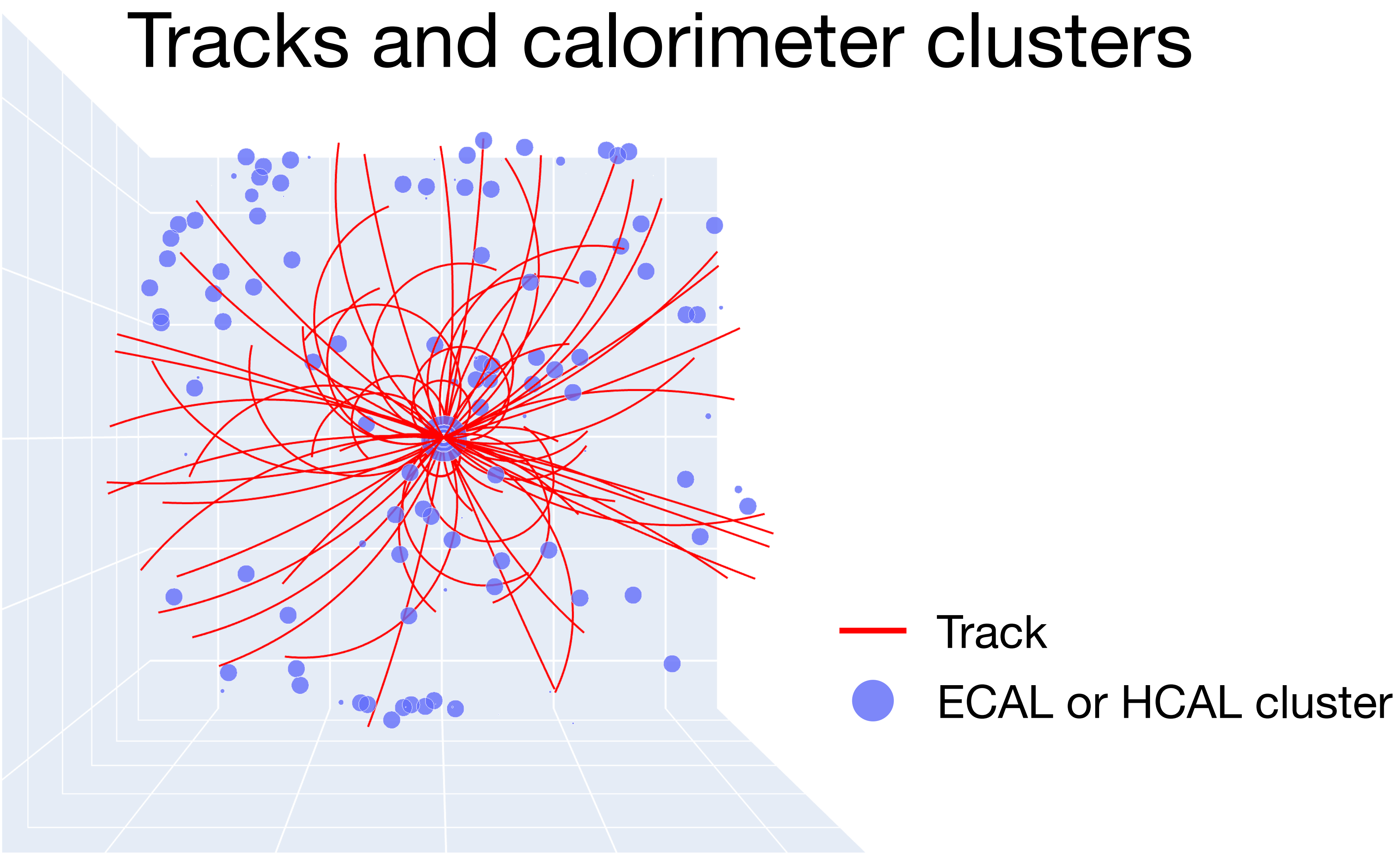
The CLIC detector model is based on the CMS detector at CERN. It features a superconducting solenoid with an internal diameter of 7 m, providing a magnetic field of 4 T in the center of the detector. Silicon pixel and strip trackers, the electromagnetic (ECAL) and hadron calorimeters (HCAL) are embedded within the solenoid. Each subdetector is divided into a barrel and two endcap sections. The ECAL is a highly granular array of 40 layers of silicon sensors and tungsten plates. The HCAL is built from 60 layers of plastic scintillator tiles, read out by silicon photomultipliers, and steel absorber plates.

Compact Linear Collider (CLIC)

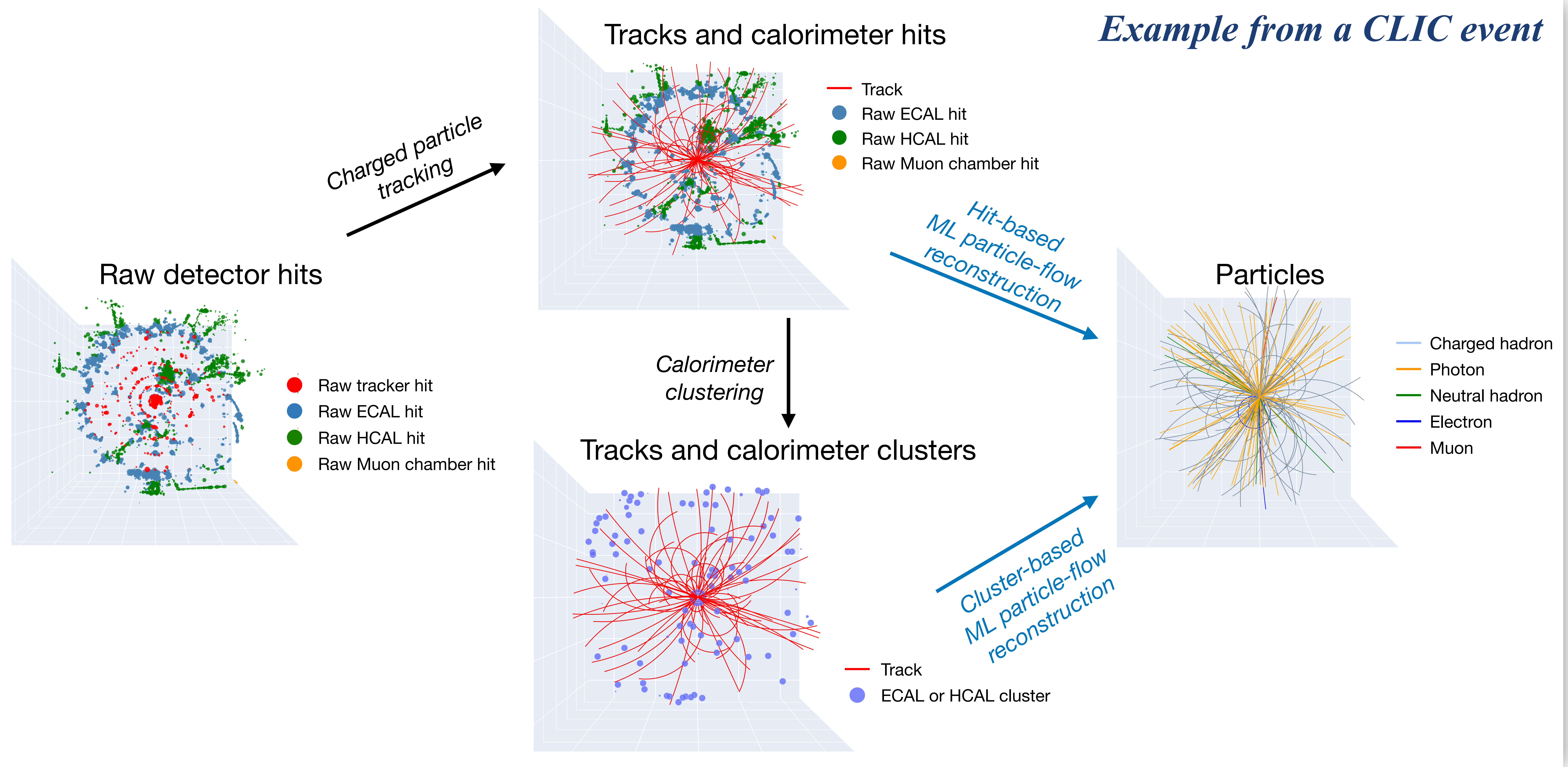
- CLIC dataset is an open dataset of e^+e^-collision events at a center of mass energy $\sqrt{s} = 380\mathrm{GeV}$ with full GEANT4 simulation available in the EDM4HEP format

- Why CLIC? Publicly available, well documented, realistic, and smaller input multiplicity compared to CMS

The CLIC detector model is based on the CMS detector at CERN. It features a superconducting solenoid with an internal diameter of 7 m, providing a magnetic field of 4 T in the center of the detector. Silicon pixel and strip trackers, the electromagnetic (ECAL) and hadron calorimeters (HCAL) are embedded within the solenoid. Each subdetector is divided into a barrel and two endcap sections. The ECAL is a highly granular array of 40 layers of silicon sensors and tungsten plates. The HCAL is built from 60 layers of plastic scintillator tiles, read out by silicon photomultipliers, and steel absorber plates.


Tracks and calorimeter clusters

Reconstruction pipeline



Example from a CLIC event

Raw detector hits

- ● Raw tracker hit
- ● Raw ECAL hit
- ● Raw HCAL hit
- ● Raw Muon chamber hit

Charged particle tracking

Tracks and calorimeter hits

- — Track
- ● Raw ECAL hit
- ● Raw HCAL hit
- ● Raw Muon chamber hit

Calorimeter clustering

Tracks and calorimeter clusters

- — Track
- ● ECAL or HCAL cluster

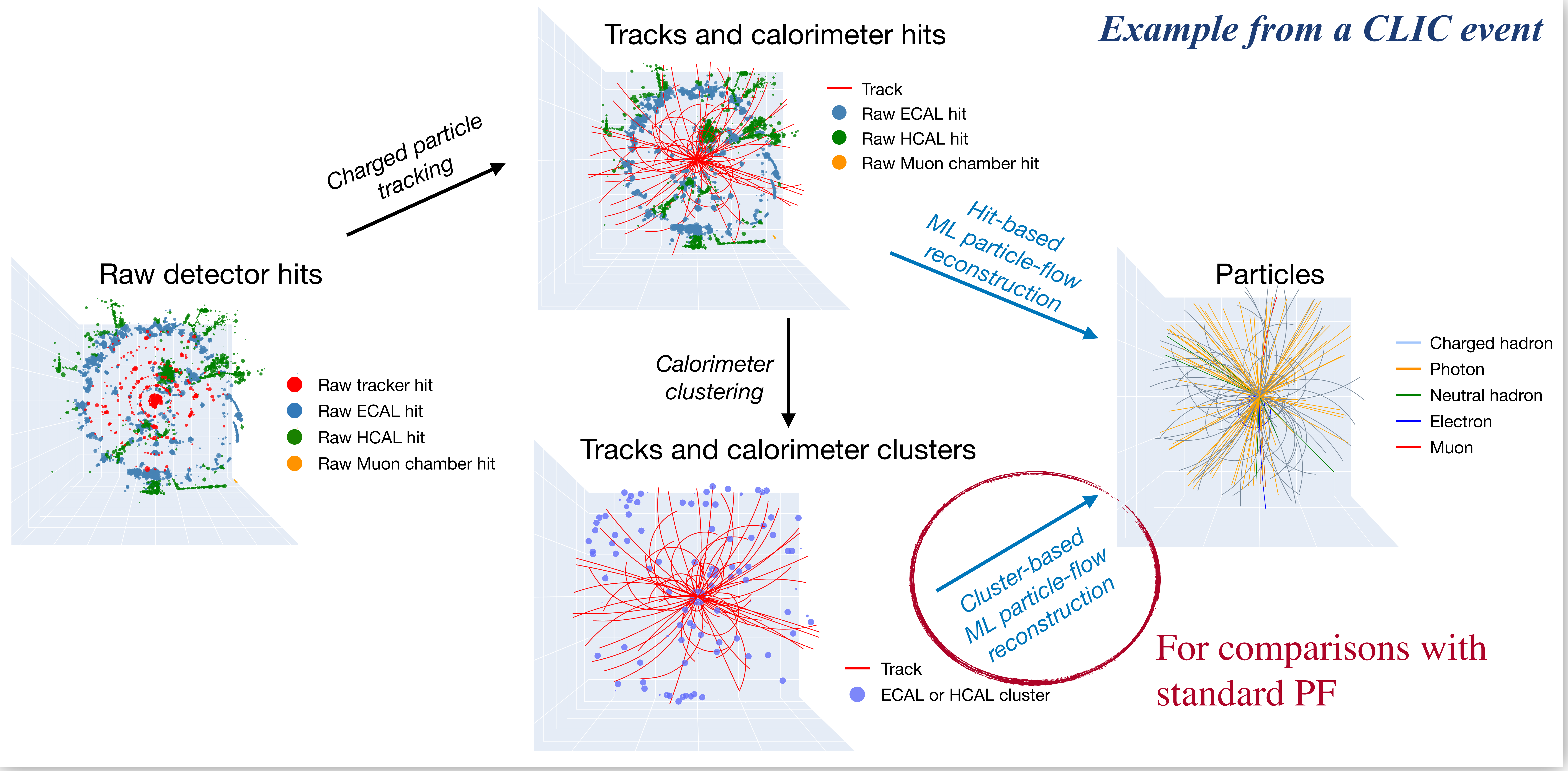
Hit-based ML particle-flow reconstruction

Cluster-based ML particle-flow reconstruction

Particles

- — Charged hadron
- — Photon
- — Neutral hadron
- — Electron
- — Muon

Reconstruction pipeline



Example from a CLIC event

Raw detector hits

Charged particle tracking

Tracks and calorimeter hits

Calorimeter clustering

Tracks and calorimeter clusters

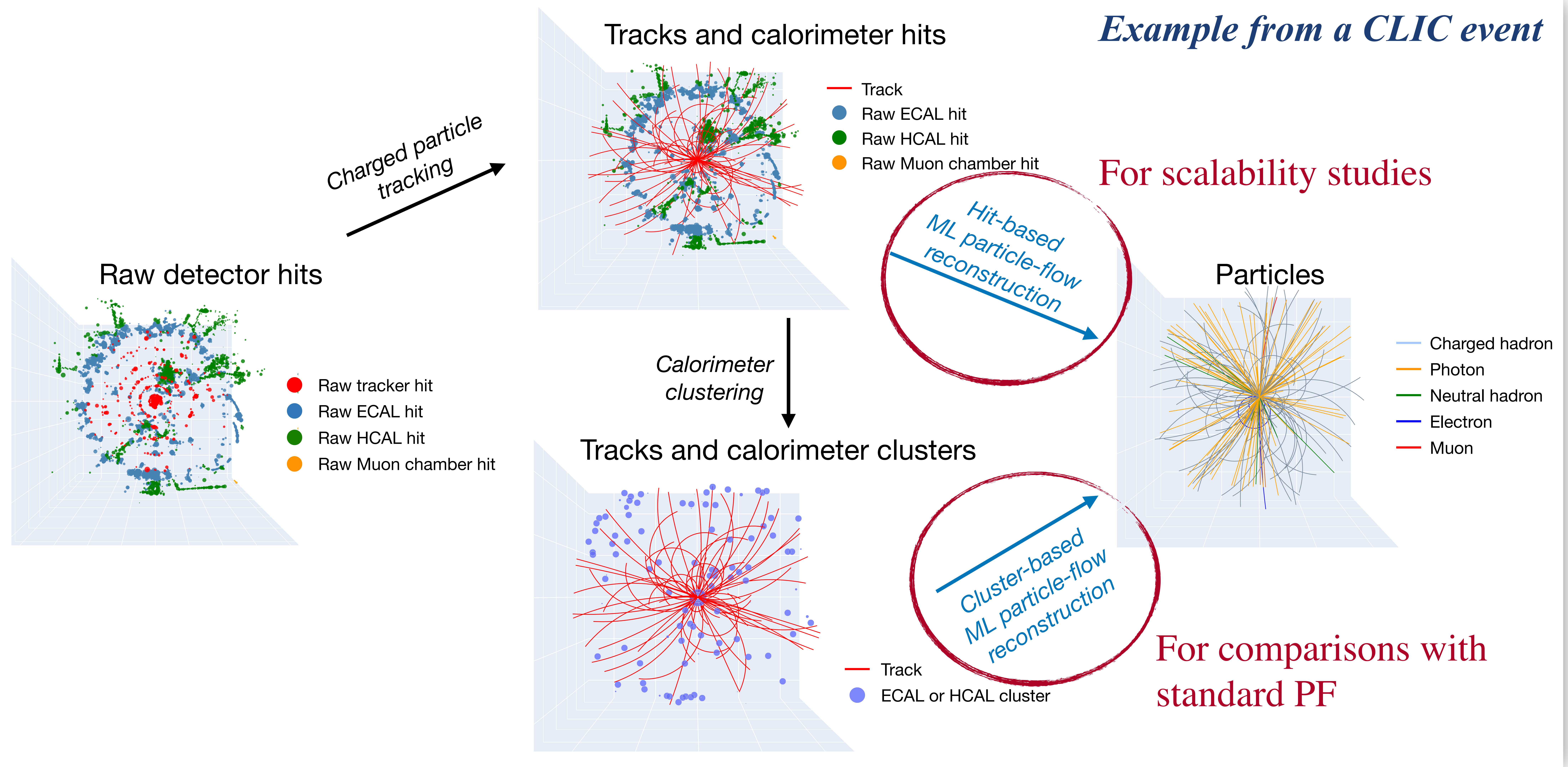
Hit-based ML particle-flow reconstruction

Cluster-based ML particle-flow reconstruction

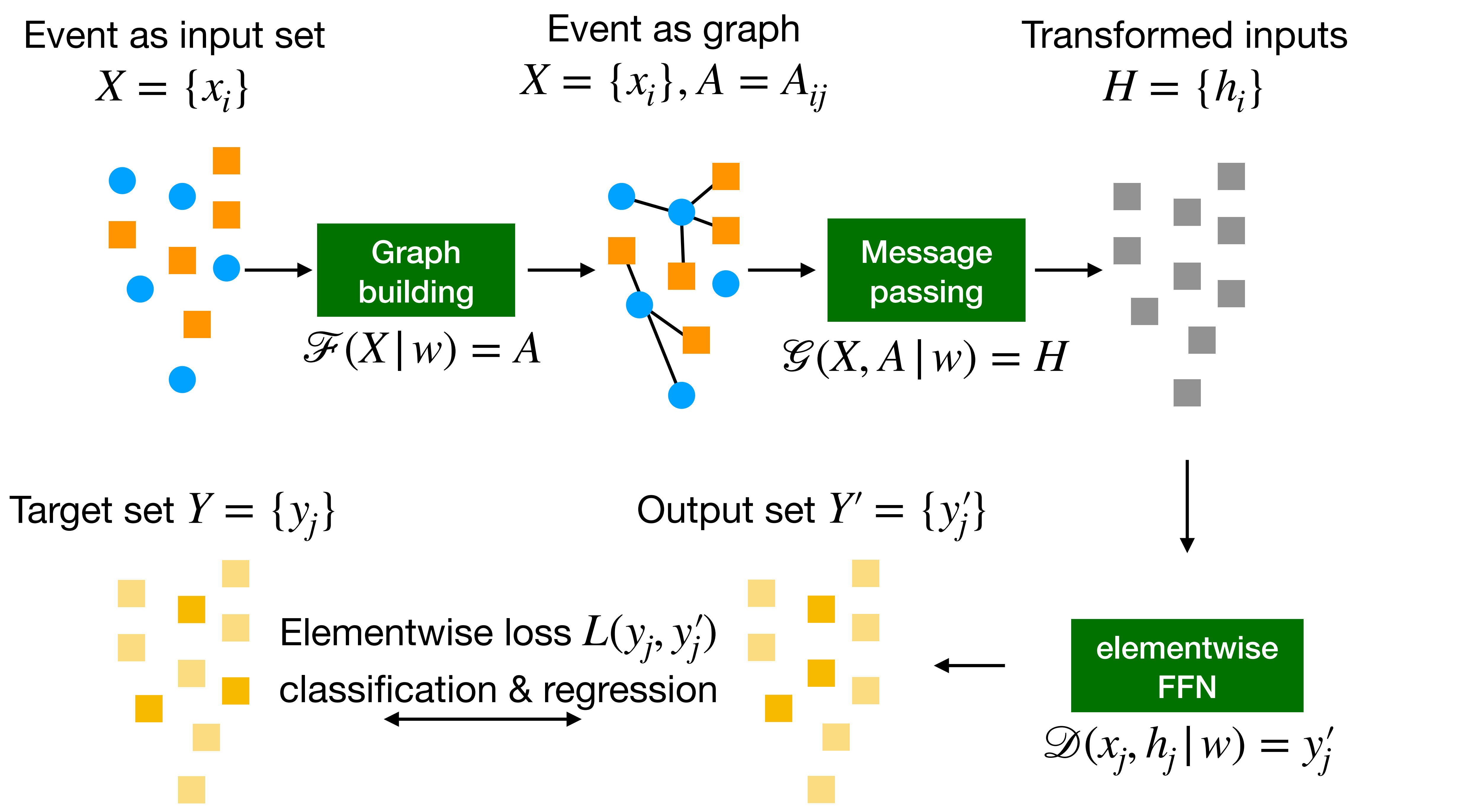
Particles

For comparisons with standard PF

Reconstruction pipeline



MLPF approach



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

Graph building

$\mathcal{F}(X \mid w) = A$

Event as graph $X = \{x_i\}, A = A_{ij}$

Message passing

$\mathcal{G}(X, A \mid w) = H$

Transformed inputs $H = \{h_i\}$

elementwise FFN

$\mathcal{D}(x_j, h_j \mid w) = y'_j$

Output set $Y' = \{y'_j\}$

Elementwise loss $L(y_j, y'_j)$ classification & regression

Target set $Y = \{y_j\}$

$$x_i = [\mathrm{type}, p_\mathrm{T}, E_\mathrm{ECAL}, E_\mathrm{HCAL}, \eta, \phi, \eta_\mathrm{outer}, \phi_\mathrm{outer}, q, \ldots], \quad \mathrm{type} \in \{\mathrm{track}, \mathrm{cluster}\}$$

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

$$h_i \in \mathbb{R}^{256}$$

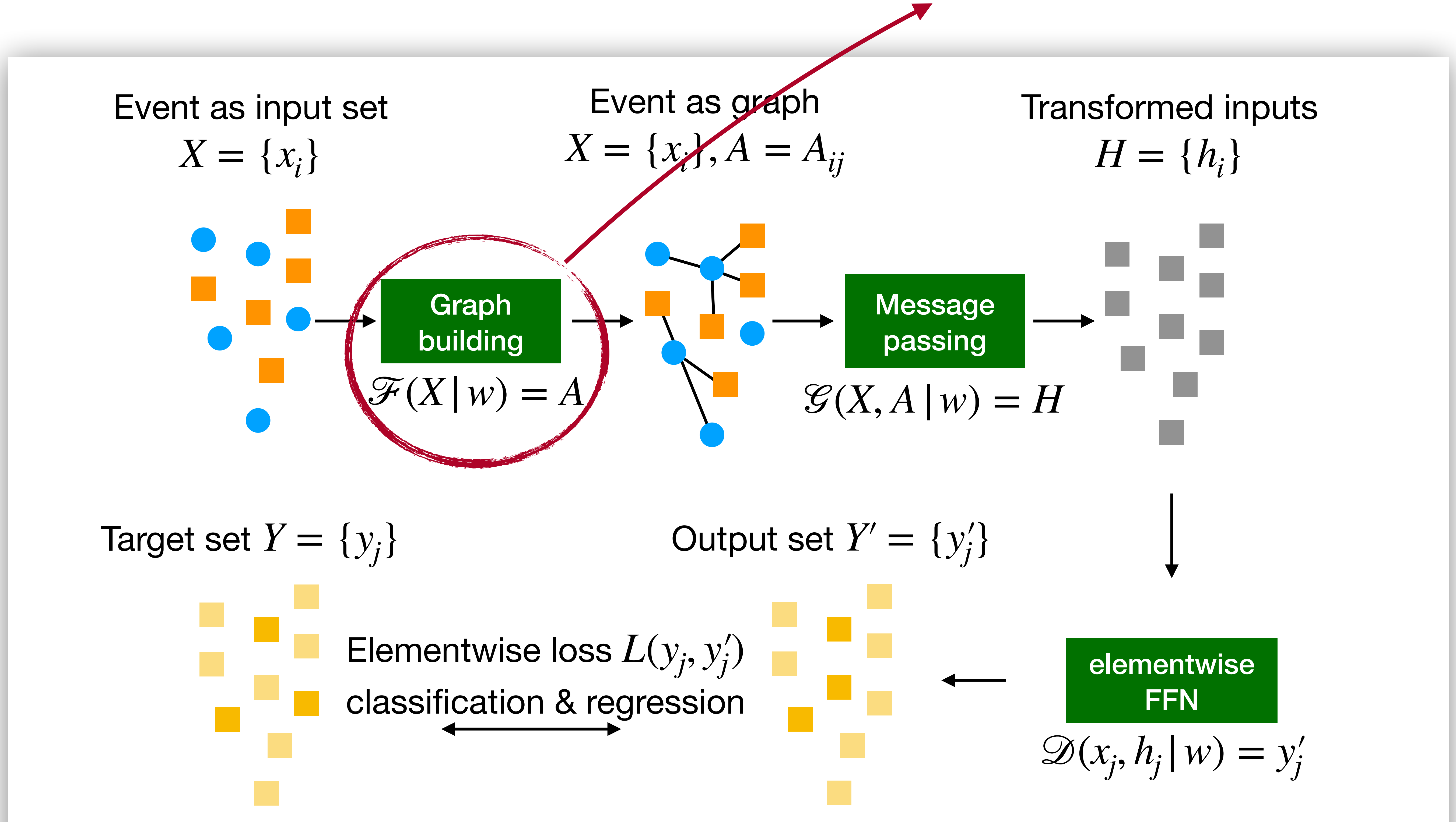
Trainable neural networks: $\mathcal{F}, \mathcal{G}, \mathcal{D}$

● - track, 🟧 - calorimeter cluster, ◼ - encoded element

🟨 - target (predicted) particle, ▢ - no target (predicted) particle

MLPF approach

1. Must avoid quadratic scaling!


Event as input set
$X = \{x_i\}$
Graph building
$\mathscr{F}(X \,|\, w) = A$
Event as graph
$X = \{x_i\}, A = A_{ij}$
Message passing
$\mathscr{G}(X, A \,|\, w) = H$
Transformed inputs
$H = \{h_i\}$
Target set $Y = \{y_j\}$
Elementwise loss $L(y_j, y_j')$
classification & regression
Output set $Y' = \{y_j'\}$
elementwise FFN
$\mathscr{D}(x_j, h_j \,|\, w) = y_j'$

$x_i = [\text{type}, p_\mathrm{T}, E_\mathrm{ECAL}, E_\mathrm{HCAL}, \eta, \phi, \eta_\mathrm{outer}, \phi_\mathrm{outer}, q, \ldots], \quad \text{type} \in \{\text{track}, \text{cluster}\}$

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

$h_i \in \mathbb{R}^{256}$

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

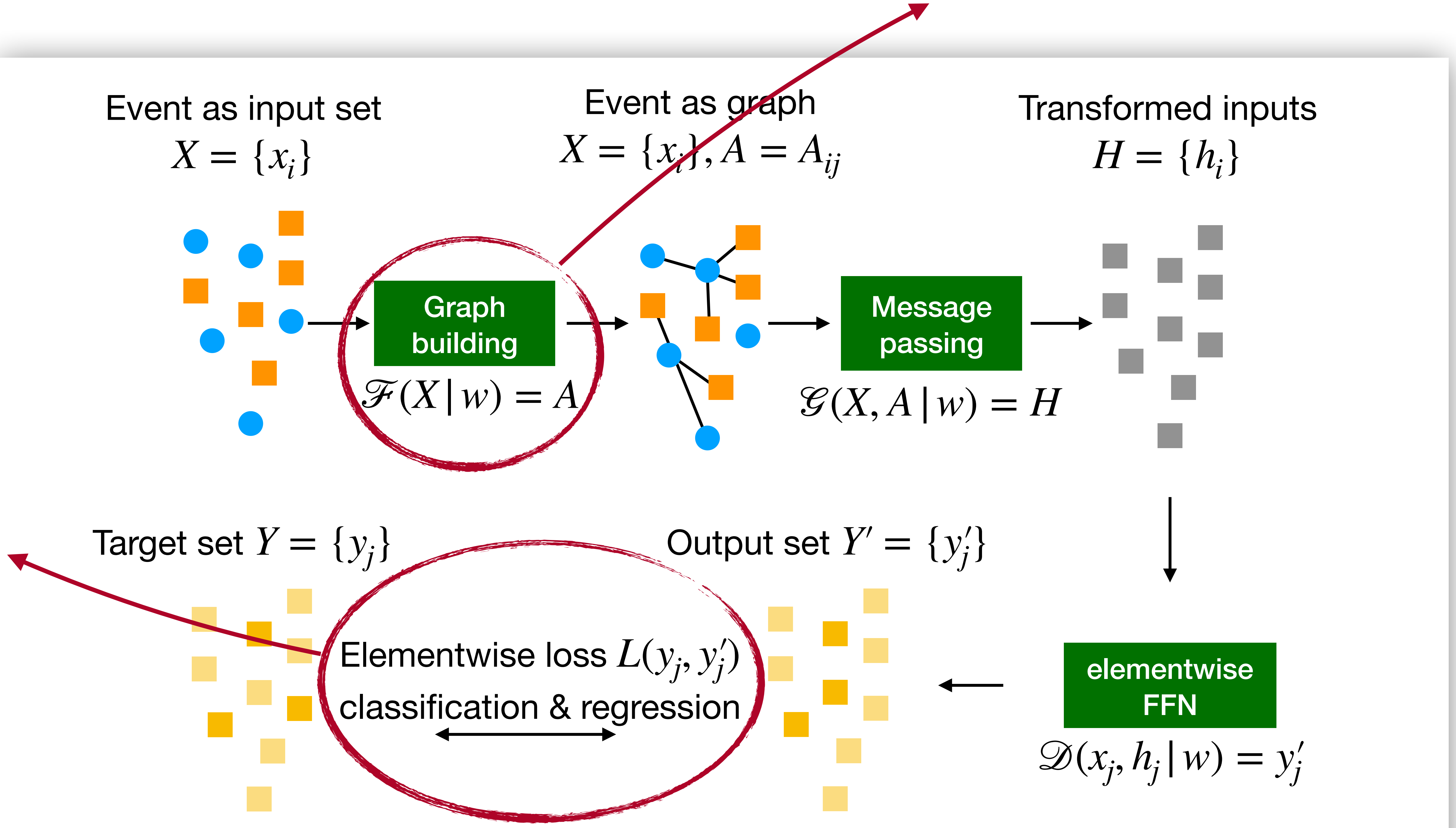
● - track, 🟧 - calorimeter cluster, ■ - encoded element

🟨 - target (predicted) particle, ▨ - no target (predicted) particle

MLPF approach

1. Must avoid quadratic scaling!

2. Must reconstruct physics objects (e.g. Jets)!



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

Graph building $\mathscr{F}(X\,|\,w) = A$

Event as graph $X = \{x_i\}, A = A_{ij}$

Message passing $\mathscr{G}(X, A\,|\,w) = H$

Transformed inputs $H = \{h_i\}$

elementwise FFN $\mathscr{D}(x_j, h_j\,|\,w) = y_j'$

Output set $Y' = \{y_j'\}$

Elementwise loss $L(y_j, y_j')$ classification & regression

Target set $Y = \{y_j\}$

$$x_i = [\text{type}, p_{\mathrm{T}}, E_{\mathrm{ECAL}}, E_{\mathrm{HCAL}}, \eta, \phi, \eta_{\mathrm{outer}}, \phi_{\mathrm{outer}}, q, \ldots], \quad \text{type} \in \{\text{track}, \text{cluster}\}$$

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

$$h_i \in \mathbb{R}^{256}$$

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

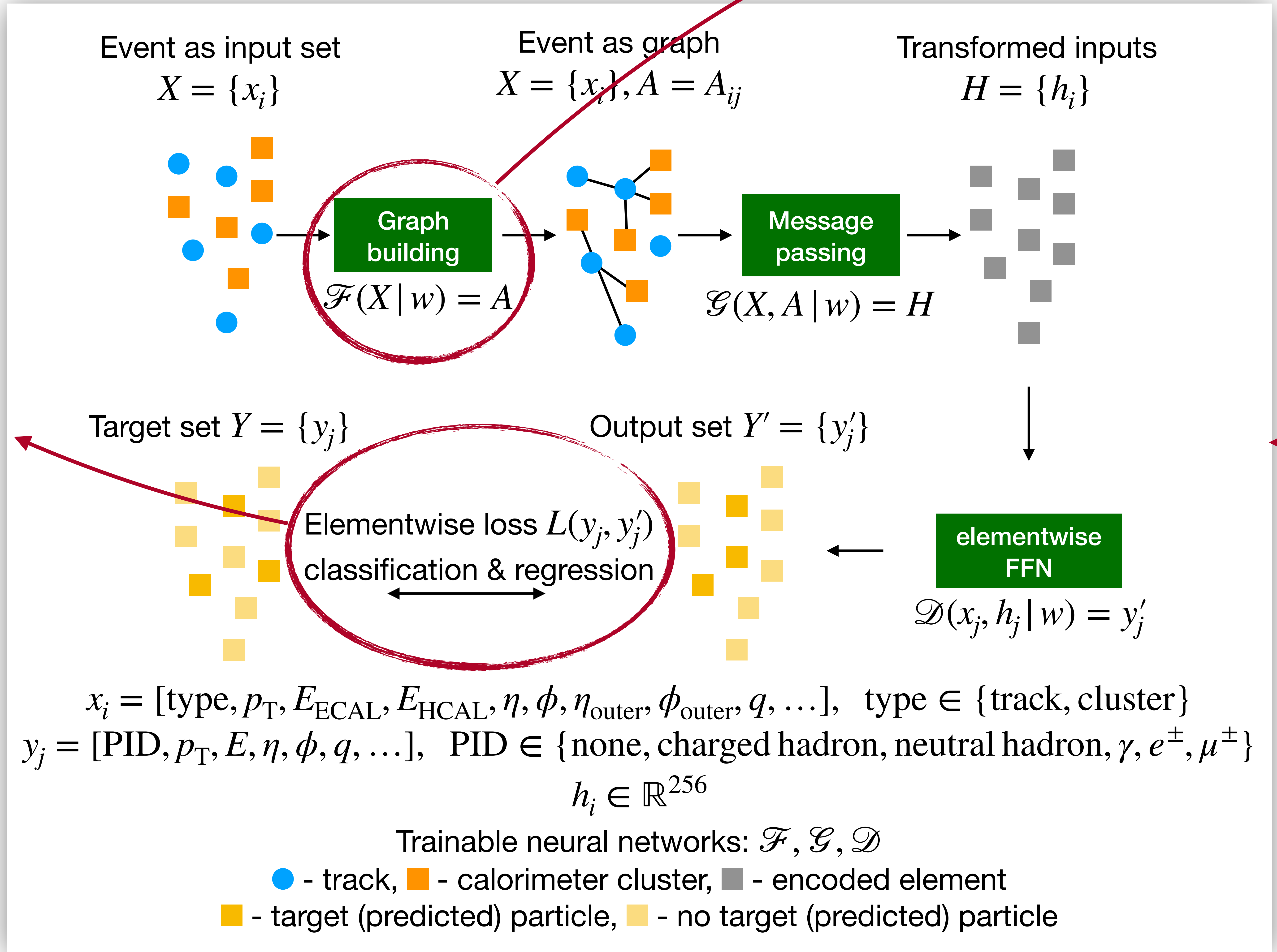
● - track, 🟧 - calorimeter cluster, ■ - encoded element

🟨 - target (predicted) particle, □ - no target (predicted) particle

MLPF approach

1. Must avoid quadratic scaling!

2. Must reconstruct physics objects (e.g. Jets)!



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

Graph building $\mathscr{F}(X\,|\,w) = A$

Event as graph $X = \{x_i\}, A = A_{ij}$

Message passing $\mathscr{G}(X, A\,|\,w) = H$

Transformed inputs $H = \{h_i\}$

elementwise FFN $\mathscr{D}(x_j, h_j\,|\,w) = y'_j$

Output set $Y' = \{y'_j\}$

Target set $Y = \{y_j\}$

Elementwise loss $L(y_j, y'_j)$

classification & regression

$$x_i = [\text{type}, p_\mathrm{T}, E_\mathrm{ECAL}, E_\mathrm{HCAL}, \eta, \phi, \eta_\mathrm{outer}, \phi_\mathrm{outer}, q, \ldots], \quad \text{type} \in \{\text{track}, \text{cluster}\}$$

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

$$h_i \in \mathbb{R}^{256}$$

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

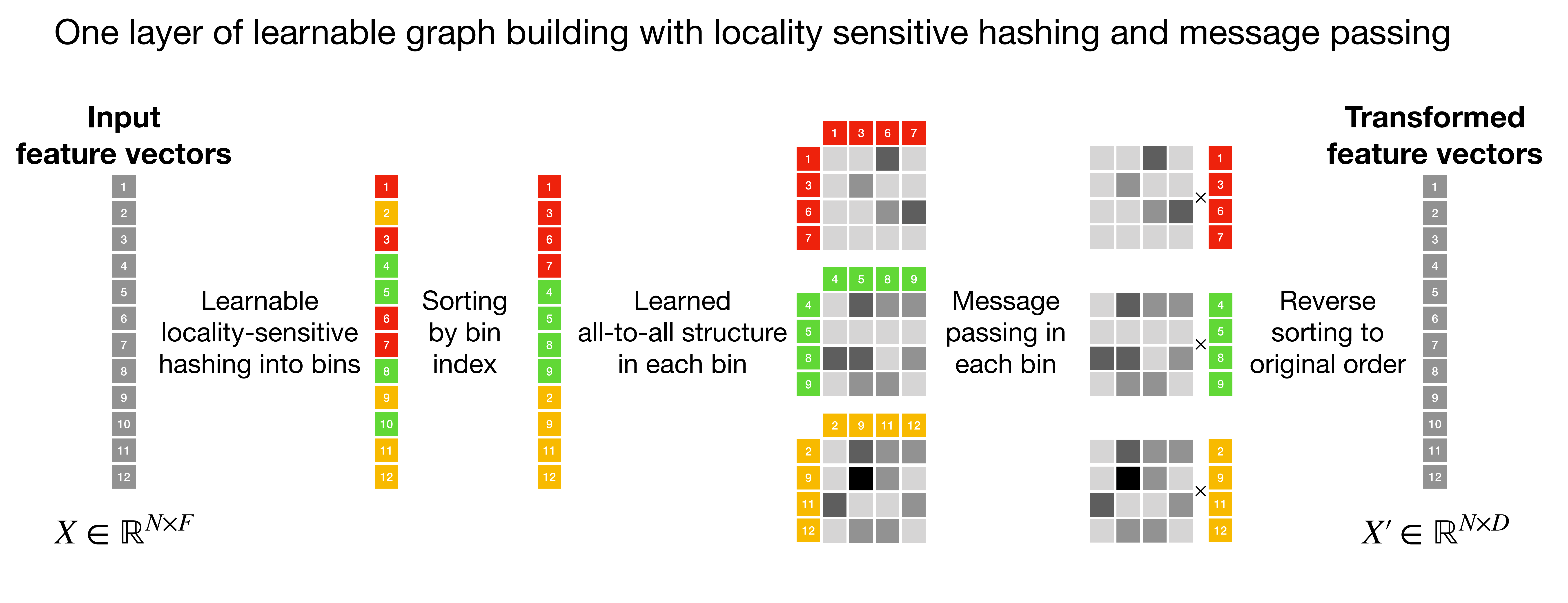
● - track, 🟧 - calorimeter cluster, ■ - encoded element

🟨 - target (predicted) particle, ▨ - no target (predicted) particle

We study two alternative models

MLPF model #1: GNN

- The first model we study uses a **dynamically learned graph structure**[1], but avoids a full quadratic allocation or computation by using a learnable binning[2, 3]



One layer of learnable graph building with locality sensitive hashing and message passing

Input feature vectors

Learnable locality-sensitive hashing into bins

Sorting by bin index

Learned all-to-all structure in each bin

Message passing in each bin

Reverse sorting to original order

Transformed feature vectors

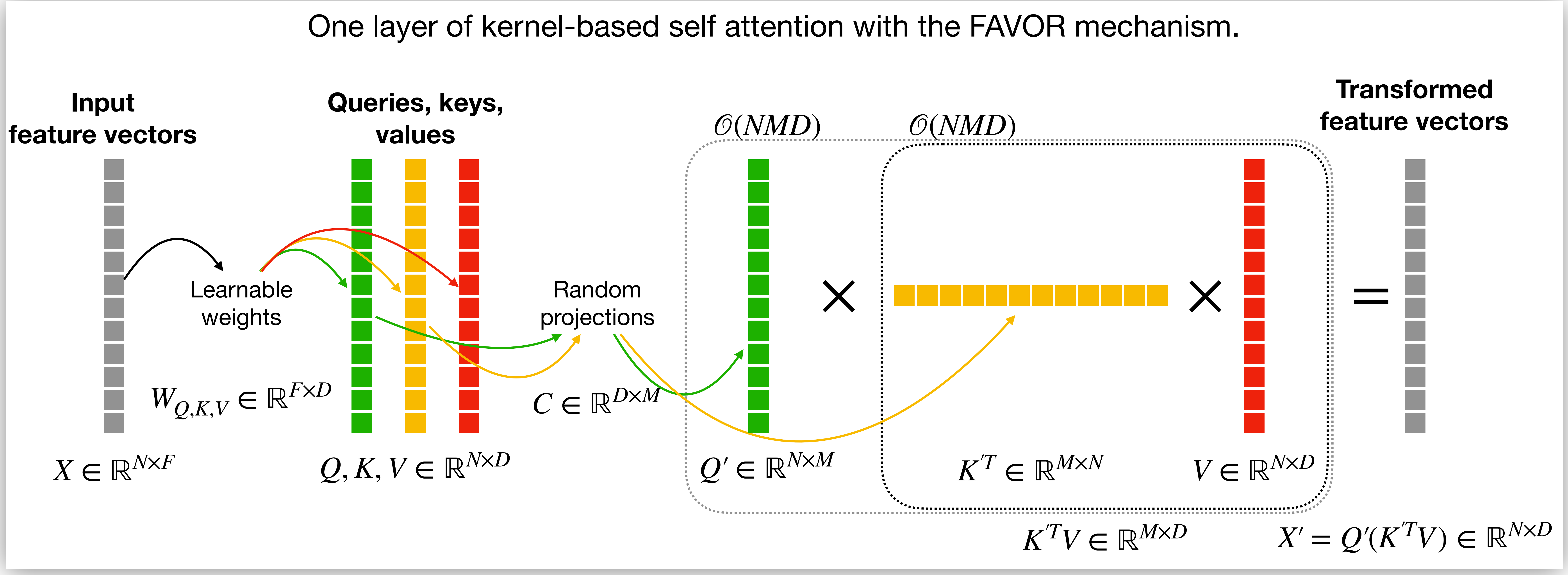
$X \in \mathbb{R}^{N \times F}$

$X' \in \mathbb{R}^{N \times D}$

[1] https://arxiv.org/abs/2101.08578 [2] https://arxiv.org/abs/2203.00330 [3] https://arxiv.org/abs/2303.17657

MLPF model #2: Transformer

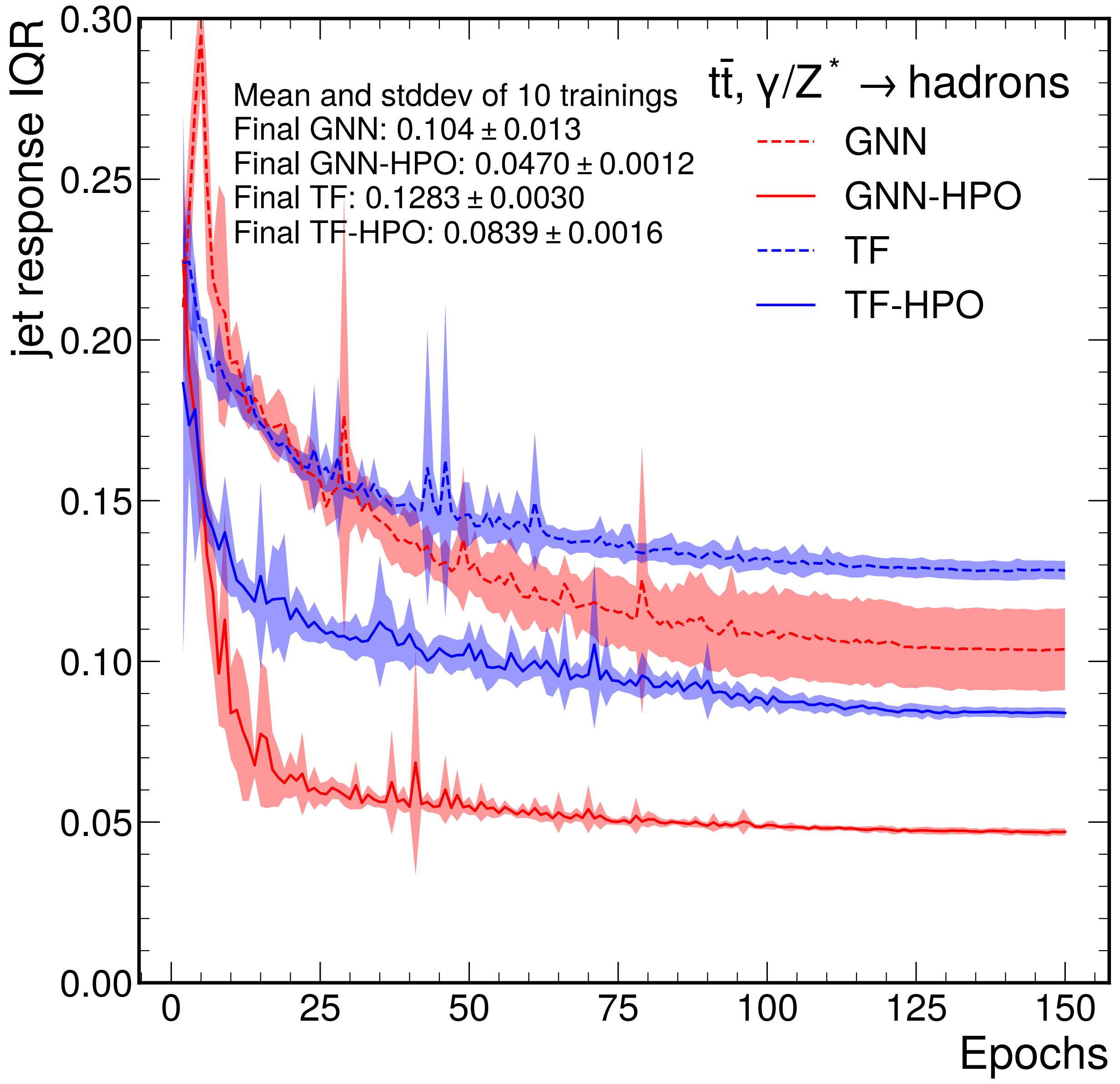
- The second model we study is a **transformer (TF)** in which the softmax self-attention layer is approximated using fast attention via positive orthogonal random features[1]

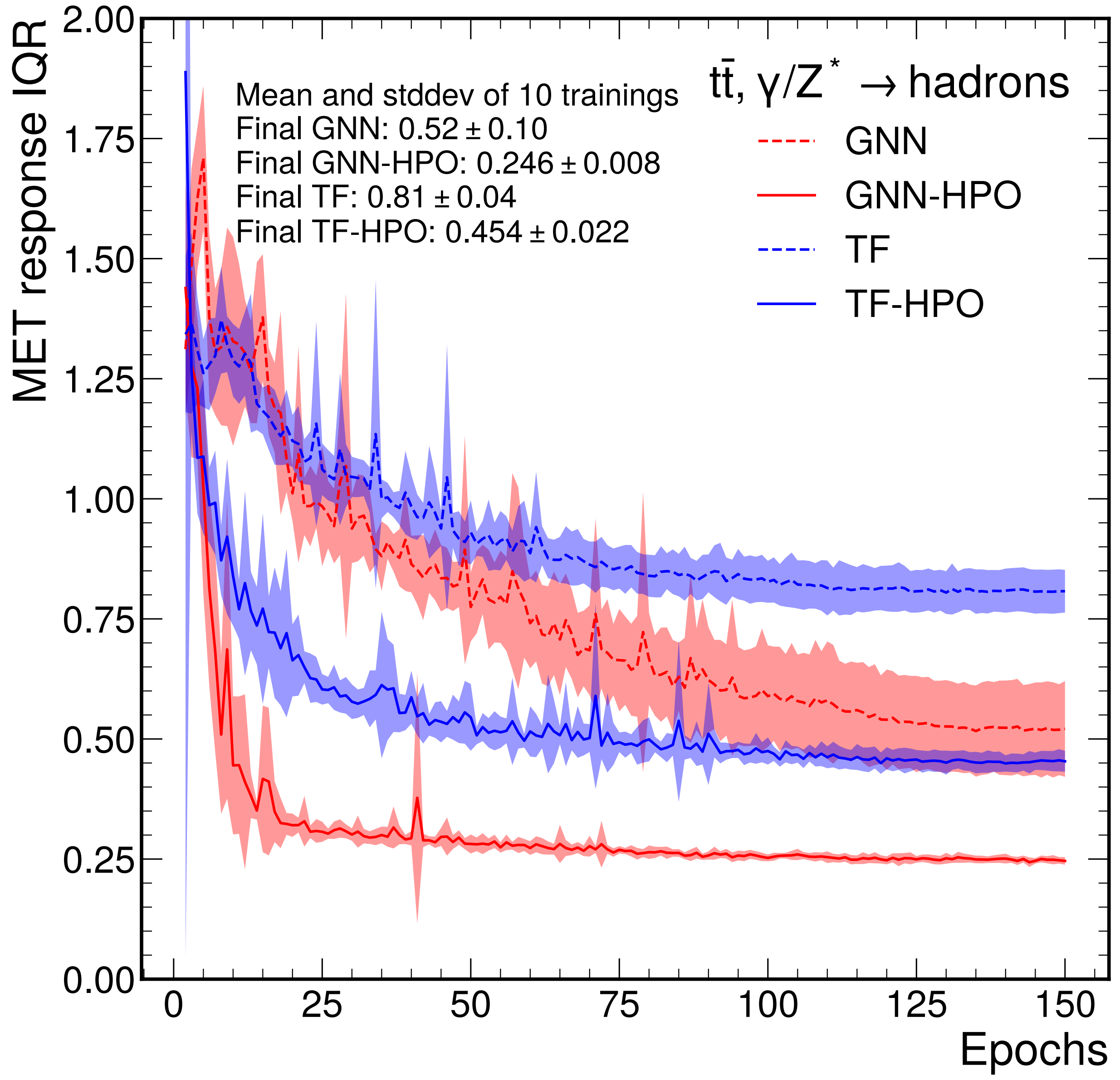

One layer of kernel-based self attention with the FAVOR mechanism.
Input feature vectors
Learnable weights
$W_{Q,K,V} \in \mathbb{R}^{F\times D}$
$X \in \mathbb{R}^{N\times F}$
Queries, keys, values
$Q, K, V \in \mathbb{R}^{N\times D}$
Random projections
$C \in \mathbb{R}^{D\times M}$
$\mathcal{O}(NMD)$
$\mathcal{O}(NMD)$
$Q' \in \mathbb{R}^{N\times M}$
×
$K'^T \in \mathbb{R}^{M\times N}$
×
$V \in \mathbb{R}^{N\times D}$
=
Transformed feature vectors
$K'^T V \in \mathbb{R}^{M\times D}$
$X' = Q'(K'^T V) \in \mathbb{R}^{N\times D}$

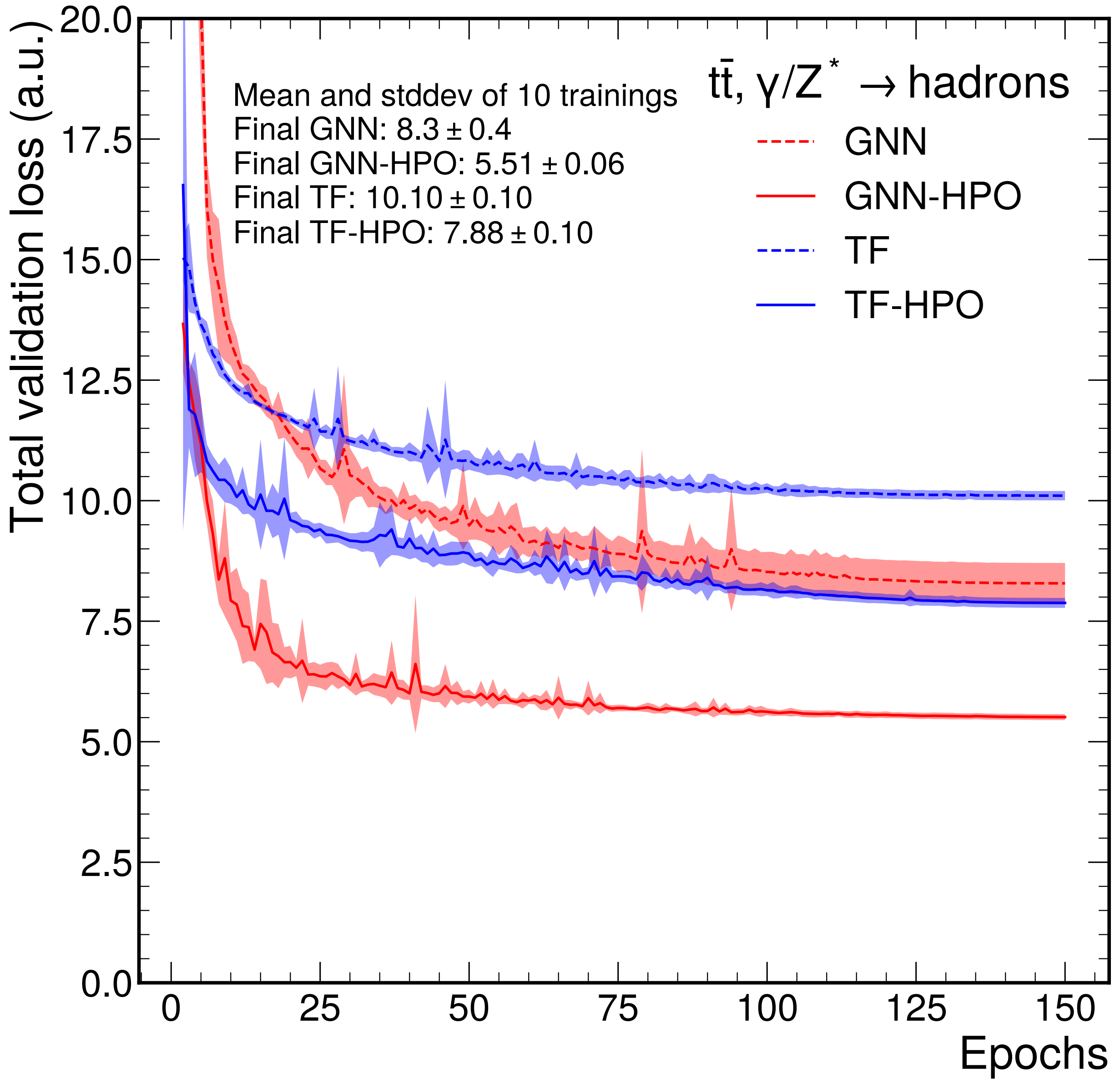
[1] K. Choromanski et al., “Rethinking attention with performers”, 2020

Results

- An extensive hyperparameter optimization (**HPO**) was performed for both models using the JURECA supercomputer[1]



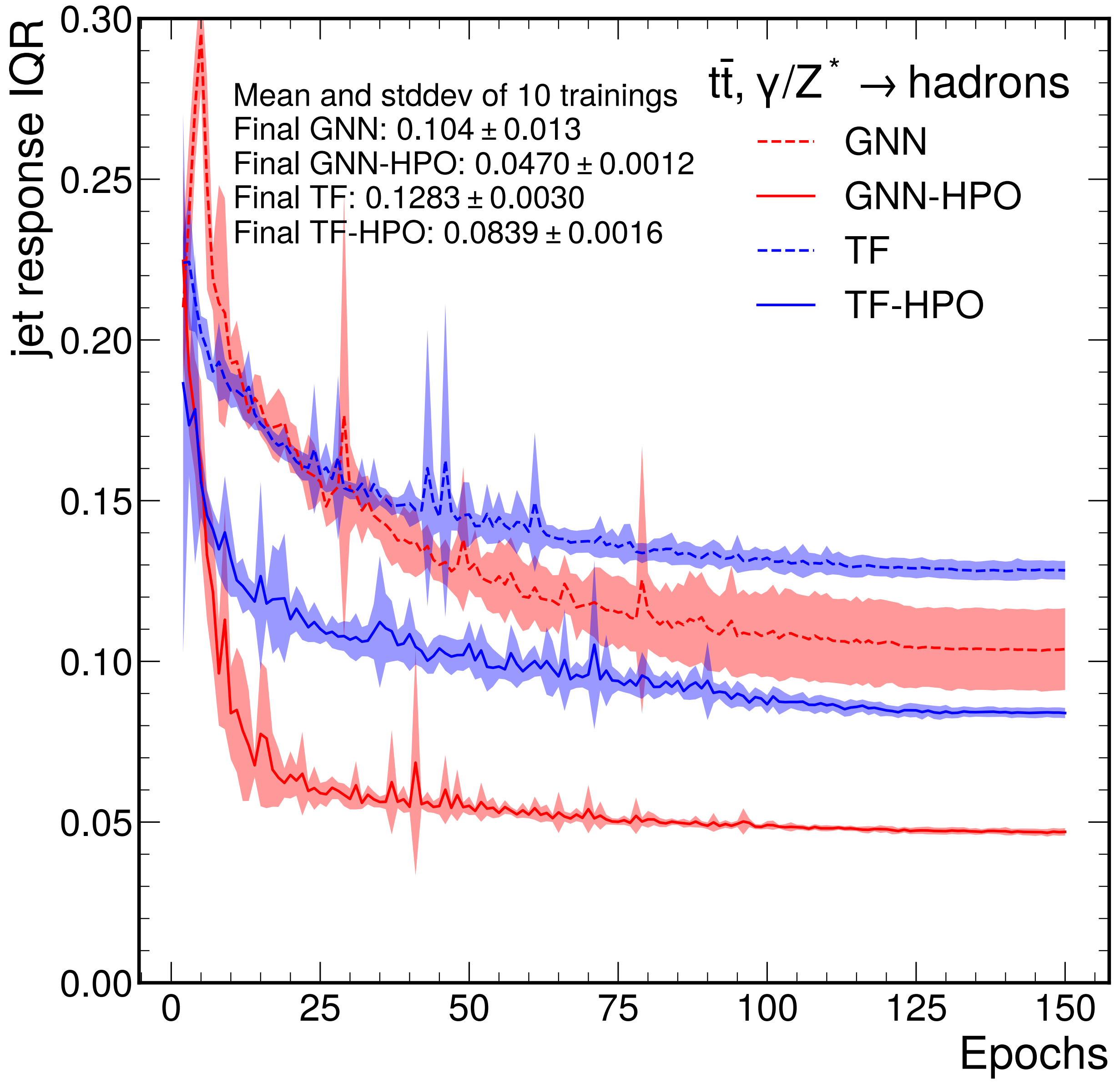


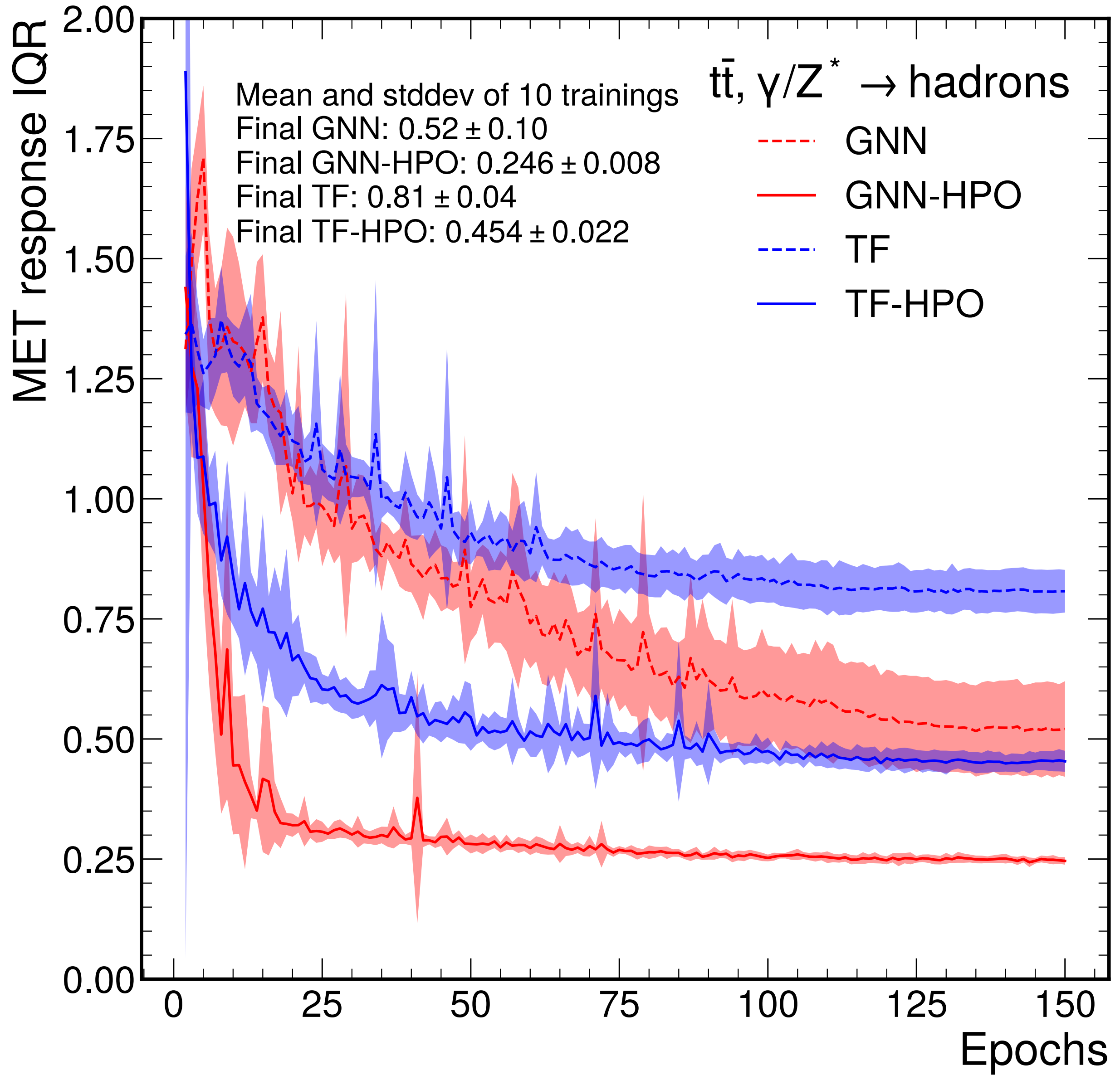


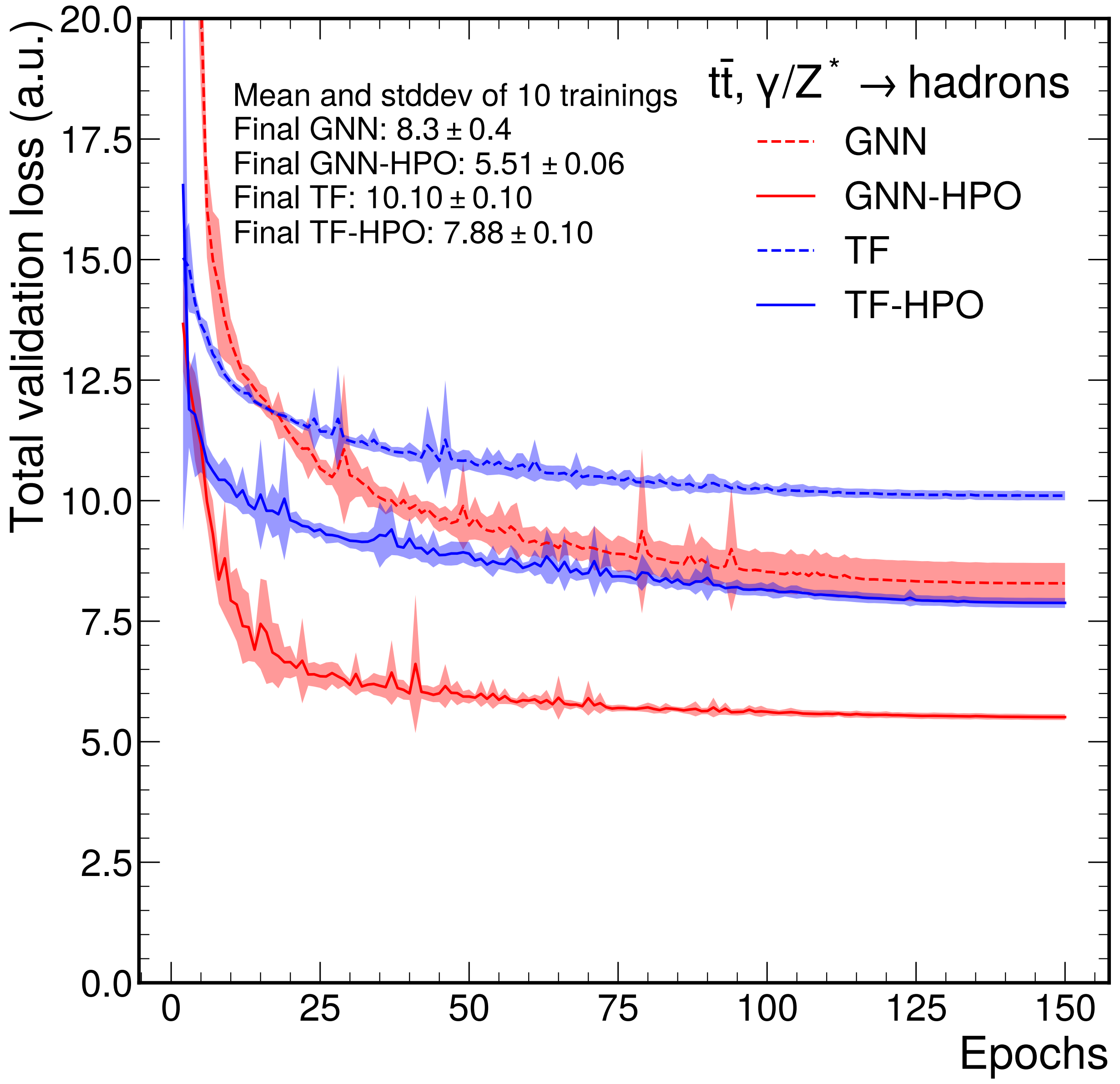
[1] JURECA is a pre-exascale modular supercomputer operated by Jülich Supercomputing Centre at Forschungszentrum Jülich

Results

- An extensive hyperparameter optimization (**HPO**) was performed for both models using the JURECA supercomputer[1]



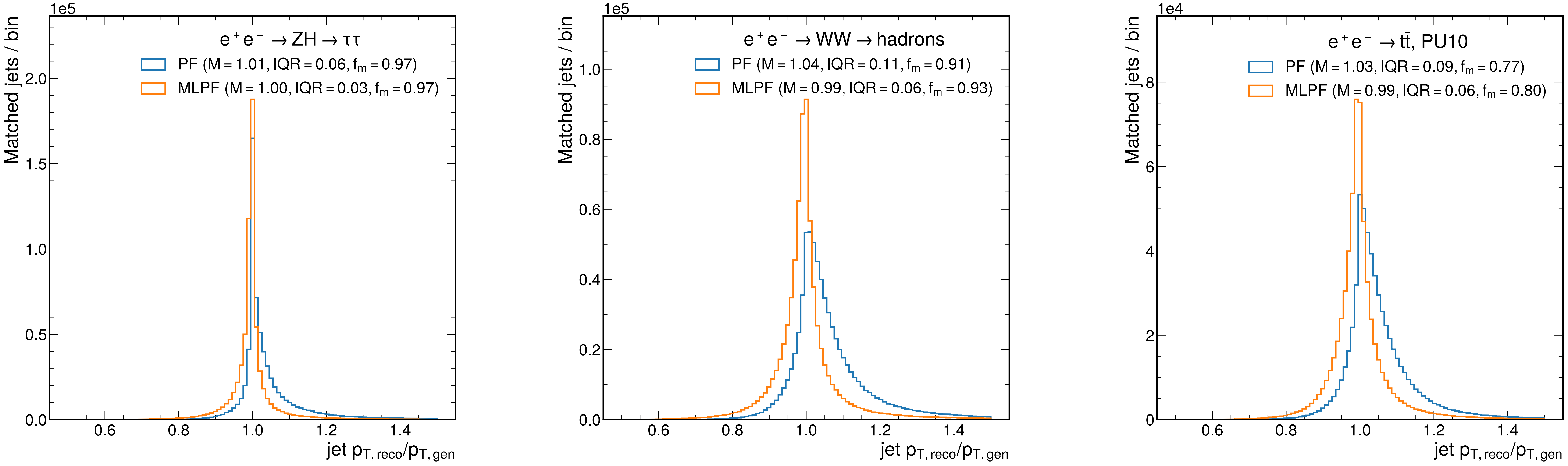




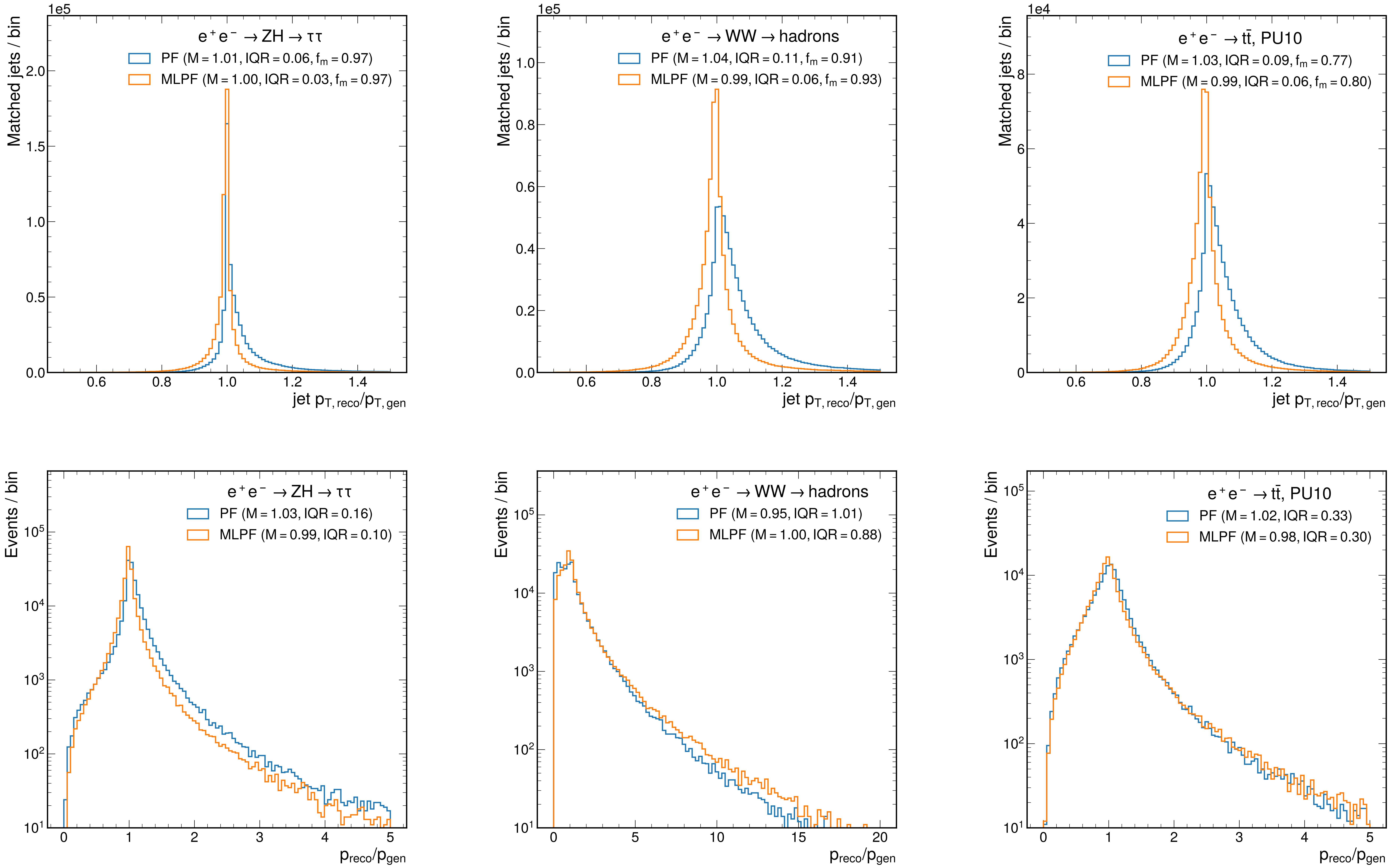
The optimized version of the GNN significantly outperforms the transformer, although both use a similar number of trainable parameters (≈5 × 106)

[1] JURECA is a pre-exascale modular supercomputer operated by Jülich Supercomputing Centre at Forschungszentrum Jülich

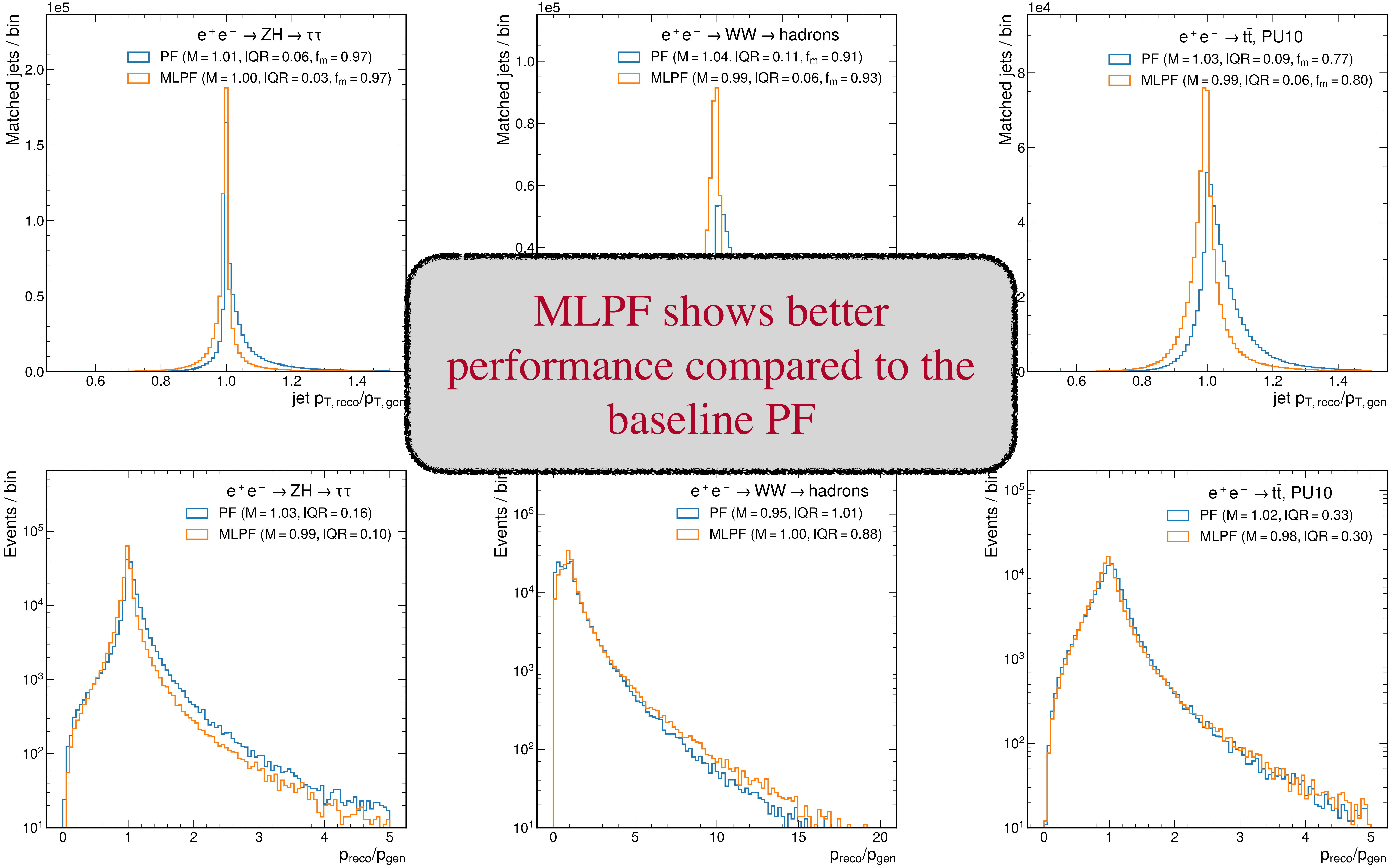
We use ZH, WW and TTbar PU10 events to evaluate out-of-distribution physics performance



We use ZH, WW and TTbar PU10 events to evaluate out-of-distribution physics performance



We use ZH, WW and TTbar PU10 events to evaluate out-of-distribution physics performance



MLPF shows better performance compared to the baseline PF

Hit-based MLPF

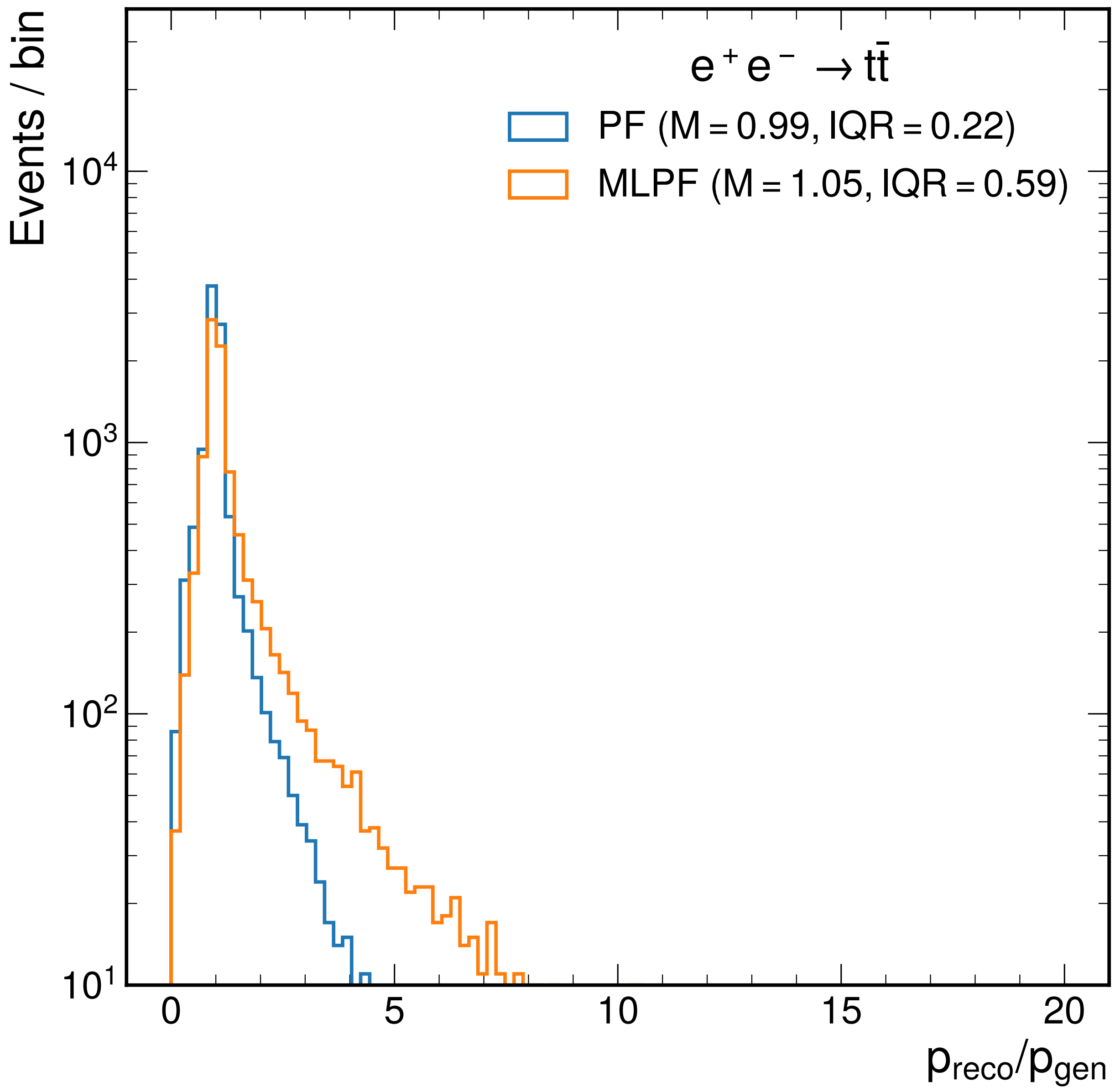
- To test the computational scalability of the approach, and to study the physics performance using lower-level inputs

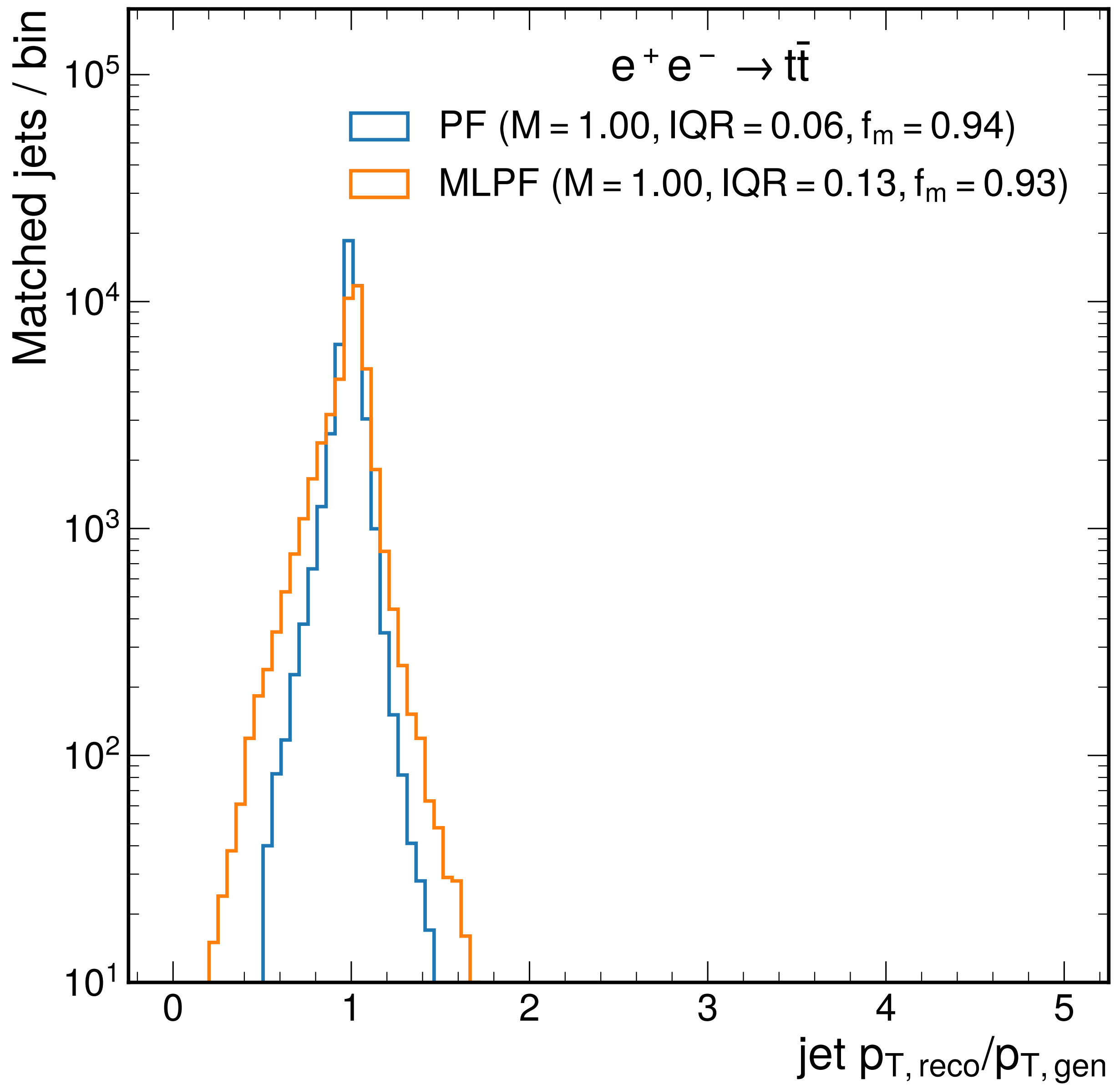
- Input for cluster-based MLPF ~ O(10^2) tracks or calorimeter clusters per event

- Input for hit-based MLPF ~ O(10^4) tracks or calorimeter hits per event

Hit-based MLPF

- To test the computational scalability of the approach, and to study the physics performance using lower-level inputs
  - Input for cluster-based MLPF ~ O(10^2) tracks or calorimeter clusters per event
  - Input for hit-based MLPF ~ O(10^4) tracks or calorimeter hits per event

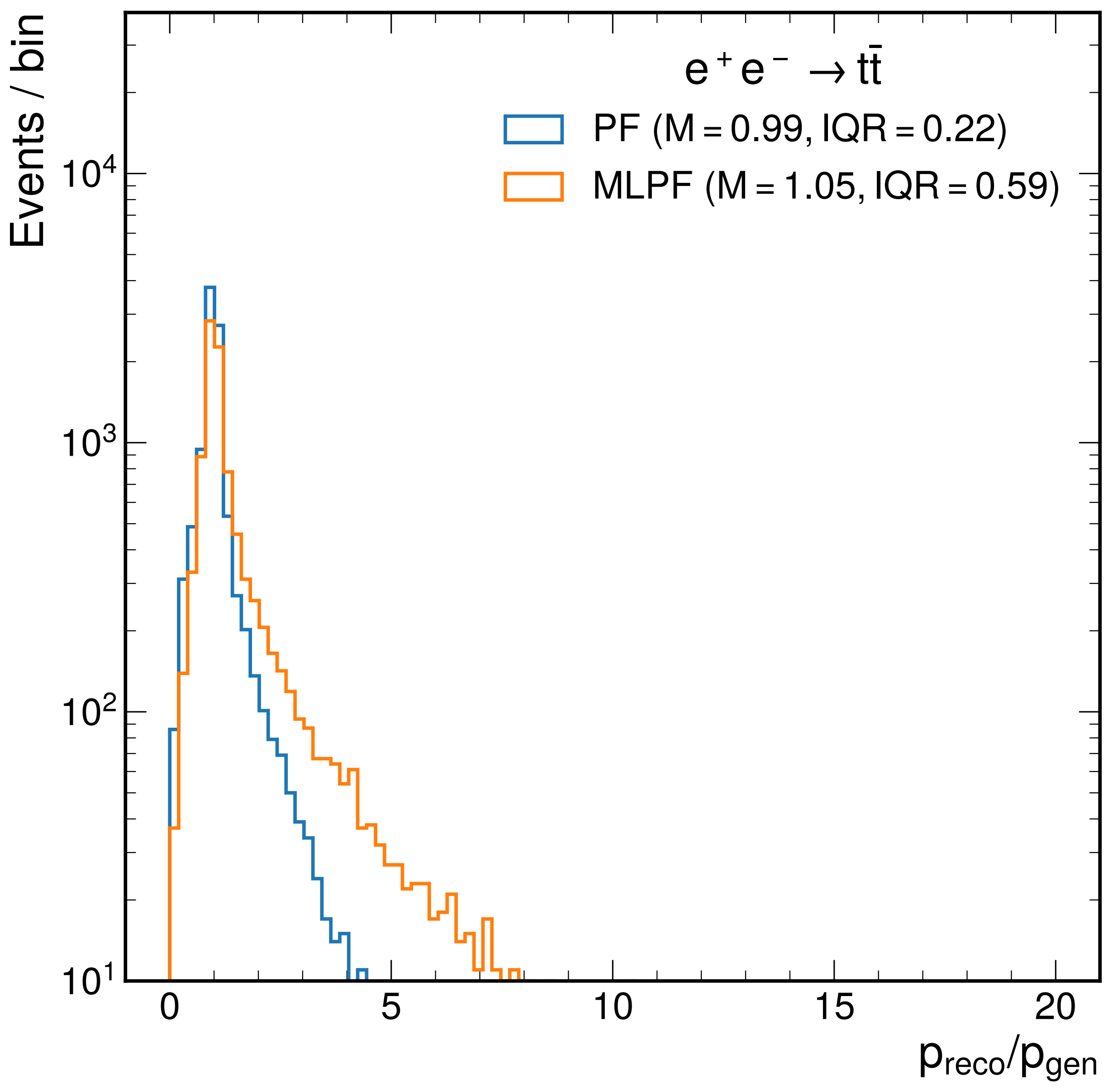


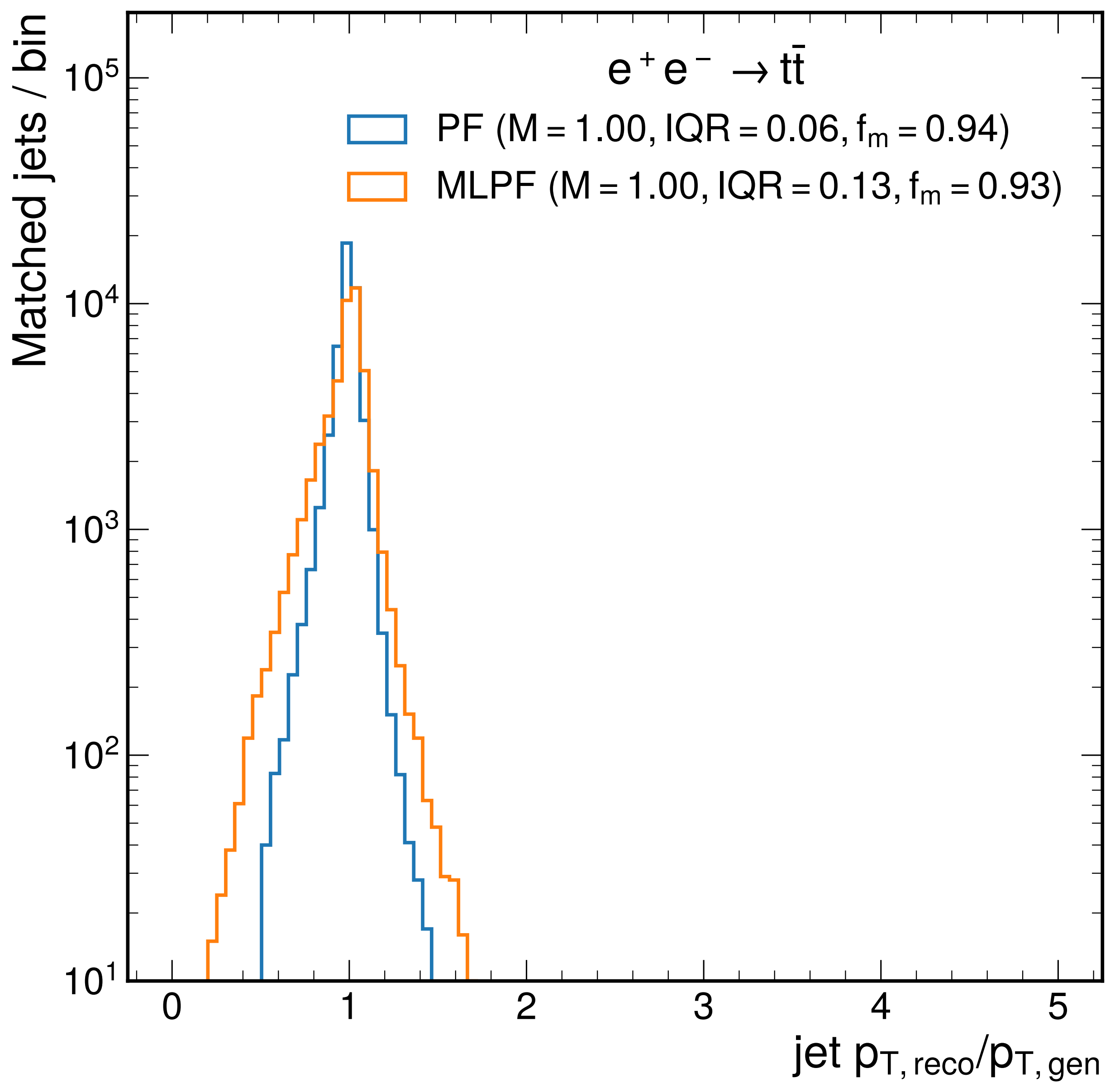


Hit-based MLPF

- To test the computational scalability of the approach, and to study the physics performance using lower-level inputs
  - Input for cluster-based MLPF ~ O(10^2) tracks or calorimeter clusters per event
  - Input for hit-based MLPF ~ O(10^4) tracks or calorimeter hits per event

Comparable performance to baseline PF





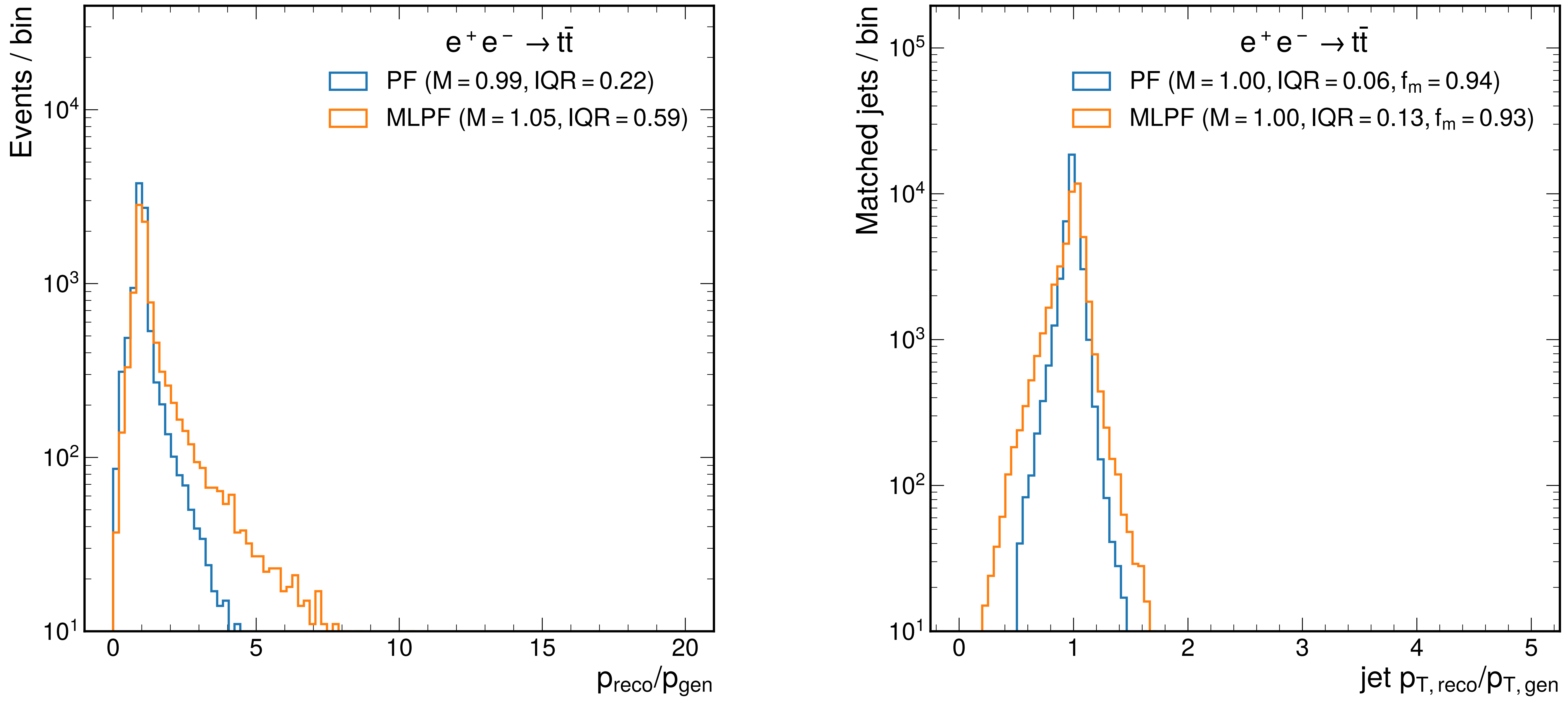
Hit-based MLPF

- To test the computational scalability of the approach, and to study the physics performance using lower-level inputs

- Input for cluster-based MLPF ~ O(10^2) tracks or calorimeter clusters per event

- Input for hit-based MLPF ~ O(10^4) tracks or calorimeter hits per event

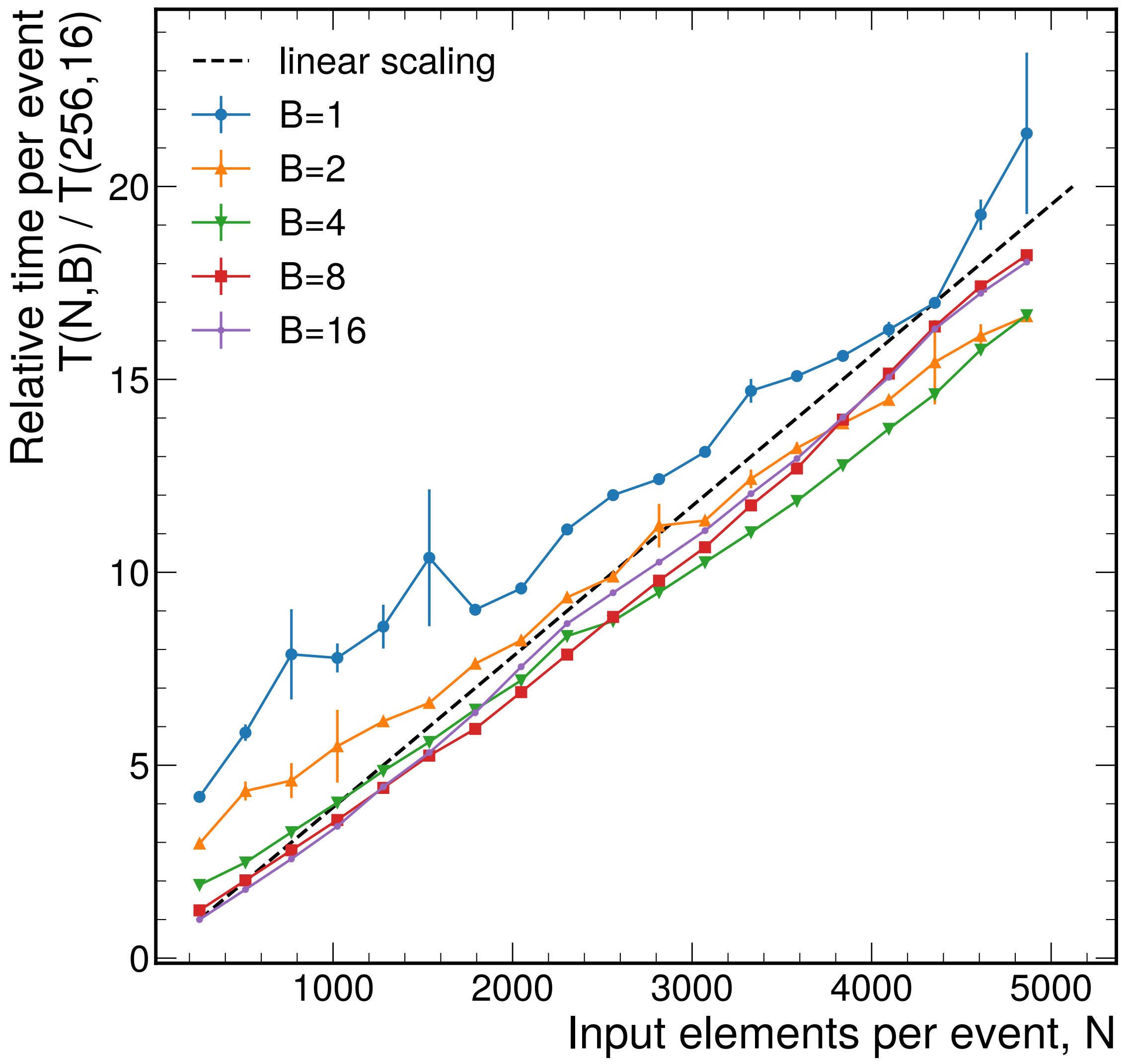
Comparable performance to baseline PF



Note: not using an optimized set of hyperparameters, and not training for a sufficient number of epochs — both limited by computational throughput

Scalability

- The training scalability is tested on three different HPC centers with different accelerator hardware: Nvidia H100 GPUs[1], AMD MI250 GPUs[2], and Intel Habana Gaudi HPUs[3]

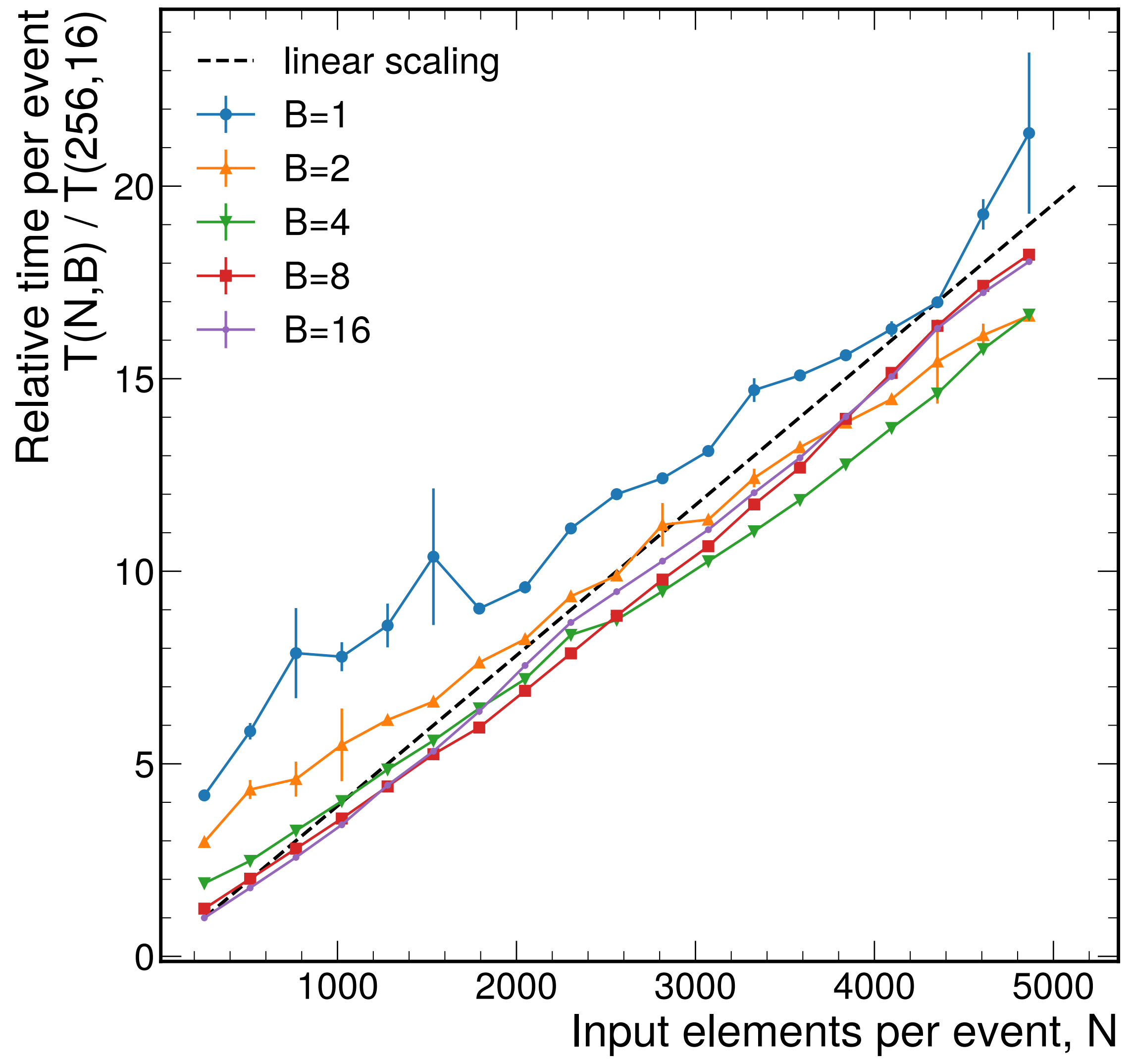


GNN-based model inference time scales approximately linearly with increasing input size

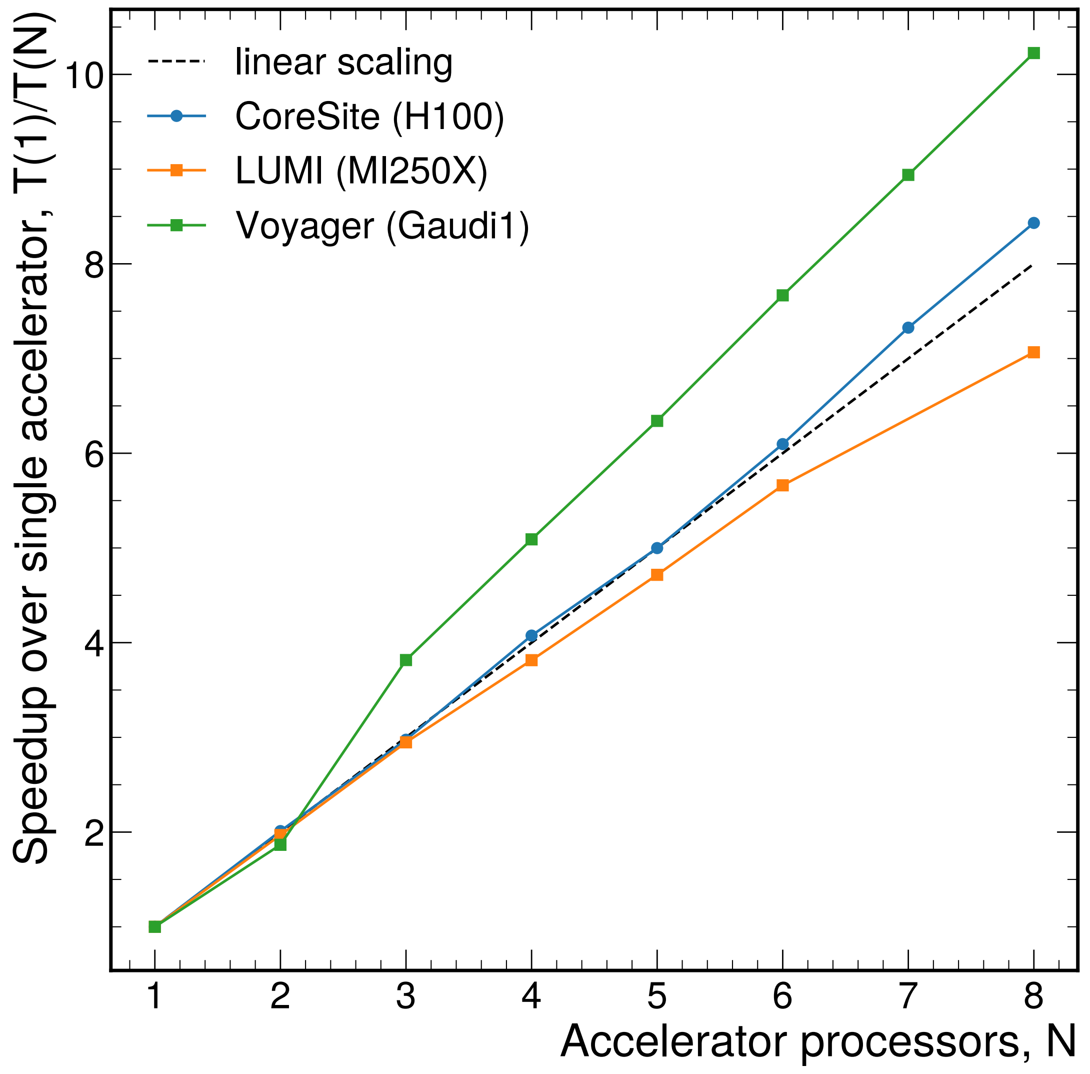
[1] Flatiron Institute, "CoreSite Cluster" [2] LUMI Consortium, "LUMI Supercomputer" [3] Voyager supercomputer: https://dl.acm.org/doi/10.1145/3569951.3597597

Scalability

- The training scalability is tested on three different HPC centers with different accelerator hardware: Nvidia H100 GPUs[1], AMD MI250 GPUs[2], and Intel Habana Gaudi HPUs[3]



GNN-based model inference time scales approximately linearly with increasing input size

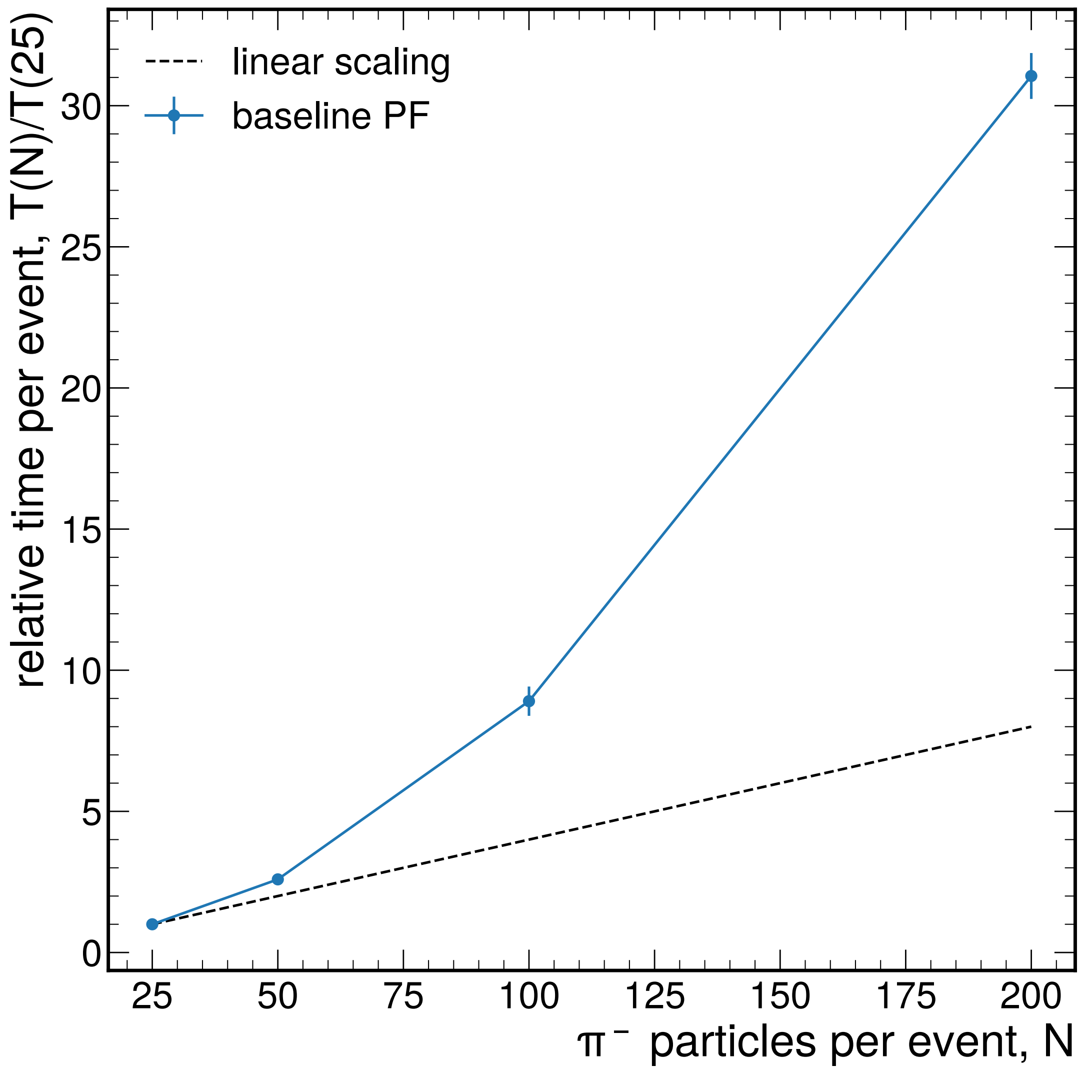


Training speed scales linearly with number of GPUs for all three accelerator types

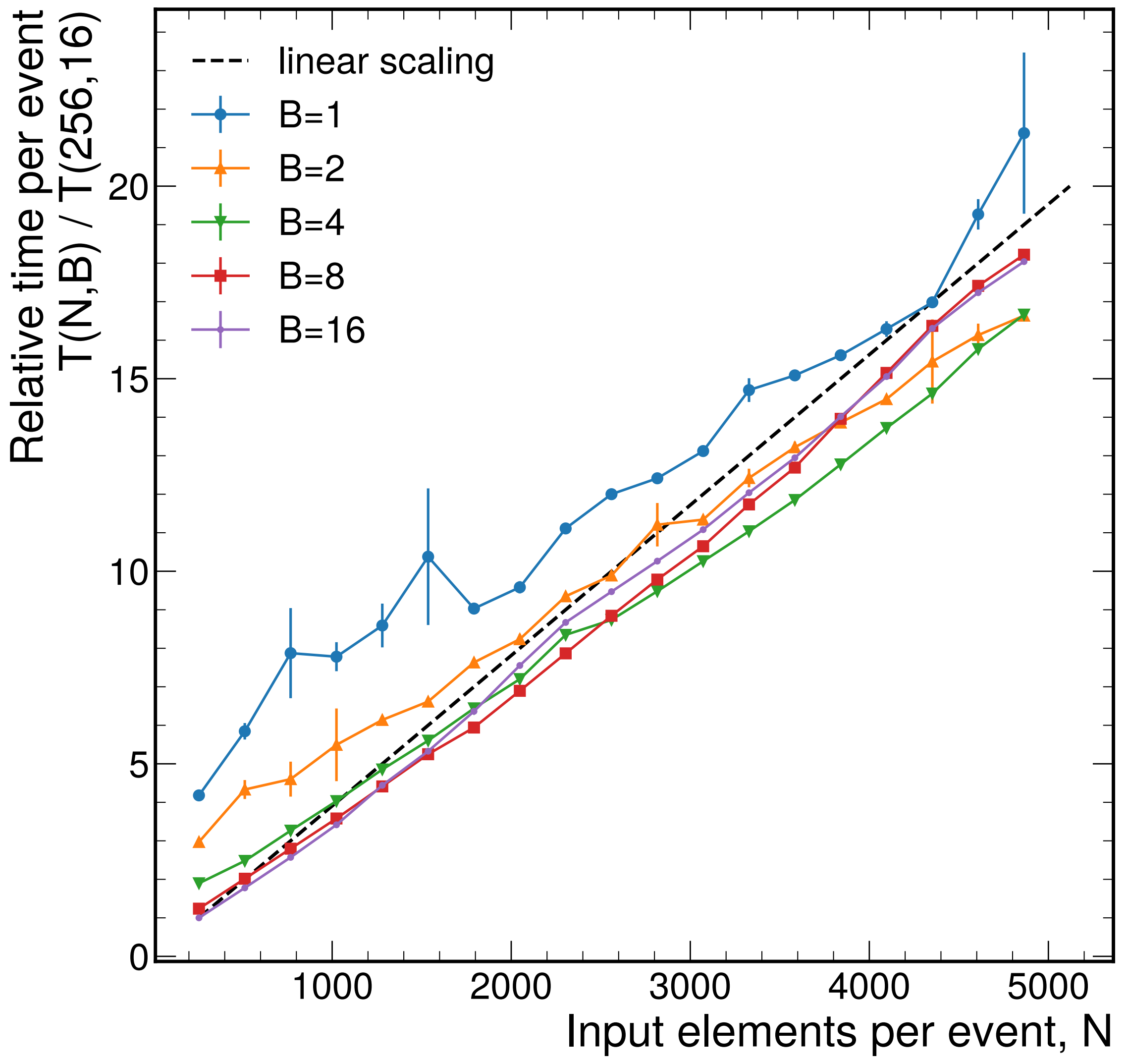
[1] Flatiron Institute, "CoreSite Cluster" [2] LUMI Consortium, "LUMI Supercomputer" [3] Voyager supercomputer: https://dl.acm.org/doi/10.1145/3569951.3597597

Scalability

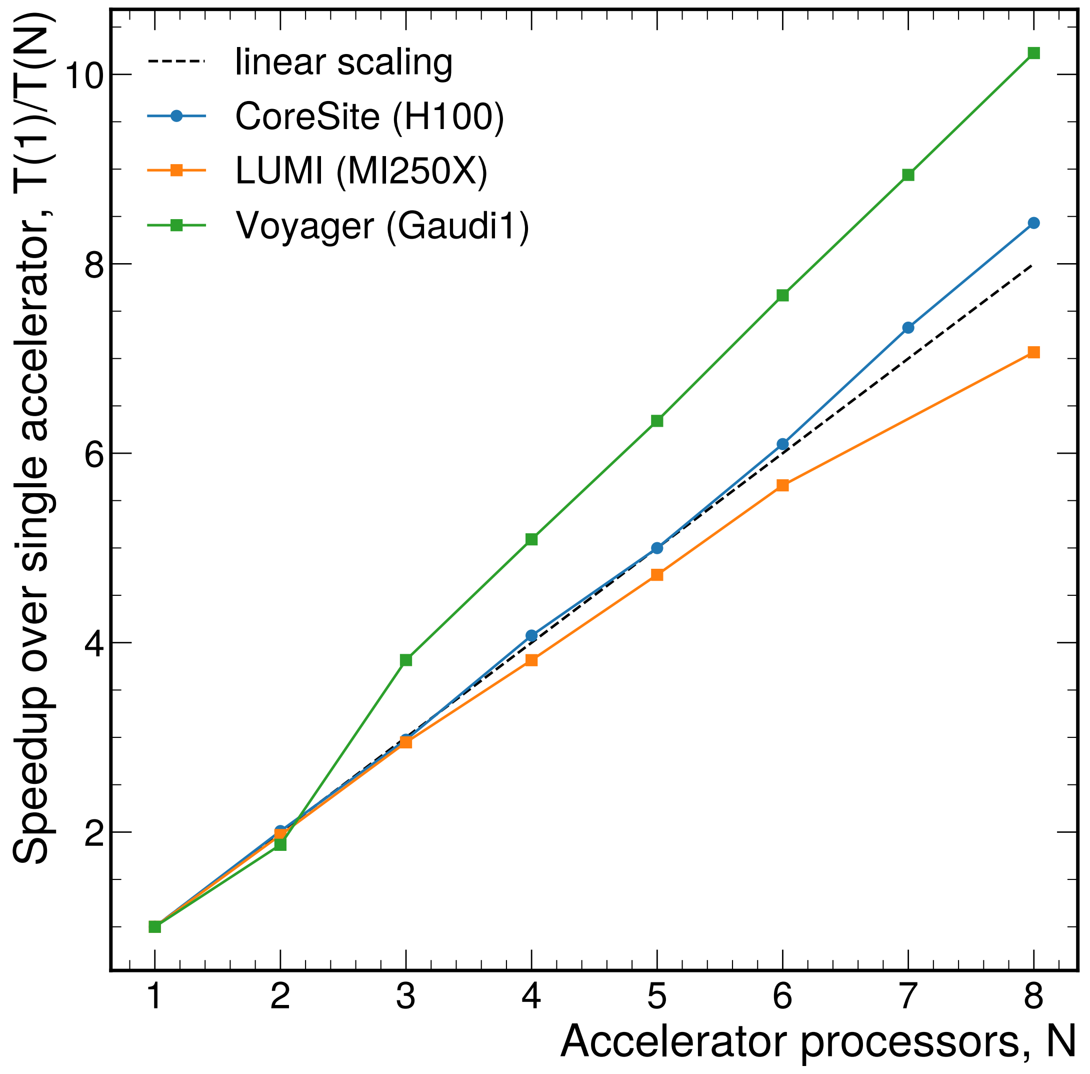
- The training scalability is tested on three different HPC centers with different accelerator hardware: Nvidia H100 GPUs[1], AMD MI250 GPUs[2], and Intel Habana Gaudi HPUs[3]



PF baseline scales non-linearly with increasing input size



GNN-based model inference time scales approximately linearly with increasing input size



Training speed scales linearly with number of GPUs for all three accelerator types

[1] Flatiron Institute, "CoreSite Cluster" [2] LUMI Consortium, "LUMI Supercomputer" [3] Voyager supercomputer: https://dl.acm.org/doi/10.1145/3569951.3597597

Summary

- We develop state-of-the-art, efficient and scalable, AI models to reconstruct events at the LHC

[1] Dataset developed following the findable, accessible, interoperable, and reusable (FAIR) principles: https://zenodo.org/record/8260741

Summary

- We develop state-of-the-art, efficient and scalable, AI models to reconstruct events at the LHC
- We test the models on different collider datasets - lately a **multi-terabyte** dataset of high-energy e^+e^- collisions at **future colliders**[1]

[1] Dataset developed following the findable, accessible, interoperable, and reusable (FAIR) principles: https://zenodo.org/record/8260741

Summary

- We develop state-of-the-art, efficient and scalable, AI models to reconstruct events at the LHC
- We test the models on different collider datasets - lately a **multi-terabyte** dataset of high-energy e^+e^- collisions at **future colliders[1]**
- We optimize the model for scalability for efficient event reconstruction at the high luminosity era

[1] Dataset developed following the findable, accessible, interoperable, and reusable (FAIR) principles: https://zenodo.org/record/8260741

Summary

- We develop state-of-the-art, efficient and scalable, AI models to reconstruct events at the LHC
- We test the models on different collider datasets - lately a **multi-terabyte** dataset of high-energy e^+e^- collisions at **future colliders**[1]
- We optimize the model for scalability for efficient event reconstruction at the high luminosity era
- We demonstrate the model's portability on several different hardware accelerators

[1] Dataset developed following the findable, accessible, interoperable, and reusable (FAIR) principles: https://zenodo.org/record/8260741

Summary

- We develop state-of-the-art, efficient and scalable, AI models to reconstruct events at the LHC
- We test the models on different collider datasets - lately a **multi-terabyte** dataset of high-energy e^+e^- collisions at **future colliders[1]**
- We optimize the model for scalability for efficient event reconstruction at the high luminosity era
- We demonstrate the model's portability on several different hardware accelerators
- **Next steps include**

[1] Dataset developed following the findable, accessible, interoperable, and reusable (FAIR) principles: https://zenodo.org/record/8260741

Summary

- We develop state-of-the-art, efficient and scalable, AI models to reconstruct events at the LHC
- We test the models on different collider datasets - lately a **multi-terabyte** dataset of high-energy e^+e^- collisions at **future colliders**[1]
- We optimize the model for scalability for efficient event reconstruction at the high luminosity era
- We demonstrate the model's portability on several different hardware accelerators
- **Next steps include**
    - Further optimization of the inference on CPU/GPU in CMSSW

[1] Dataset developed following the findable, accessible, interoperable, and reusable (FAIR) principles: https://zenodo.org/record/8260741

Summary

- We develop state-of-the-art, efficient and scalable, AI models to reconstruct events at the LHC
- We test the models on different collider datasets - lately a **multi-terabyte** dataset of high-energy e^+e^- collisions at **future colliders[1]**
- We optimize the model for scalability for efficient event reconstruction at the high luminosity era
- We demonstrate the model's portability on several different hardware accelerators
- **Next steps include**
  - Further optimization of the inference on CPU/GPU in CMSSW
  - Exploring self-supervised learning techniques to leverage the power of the gigantic unlabelled data gathered by the LHC

[1] Dataset developed following the findable, accessible, interoperable, and reusable (FAIR) principles: https://zenodo.org/record/8260741

Thank you!

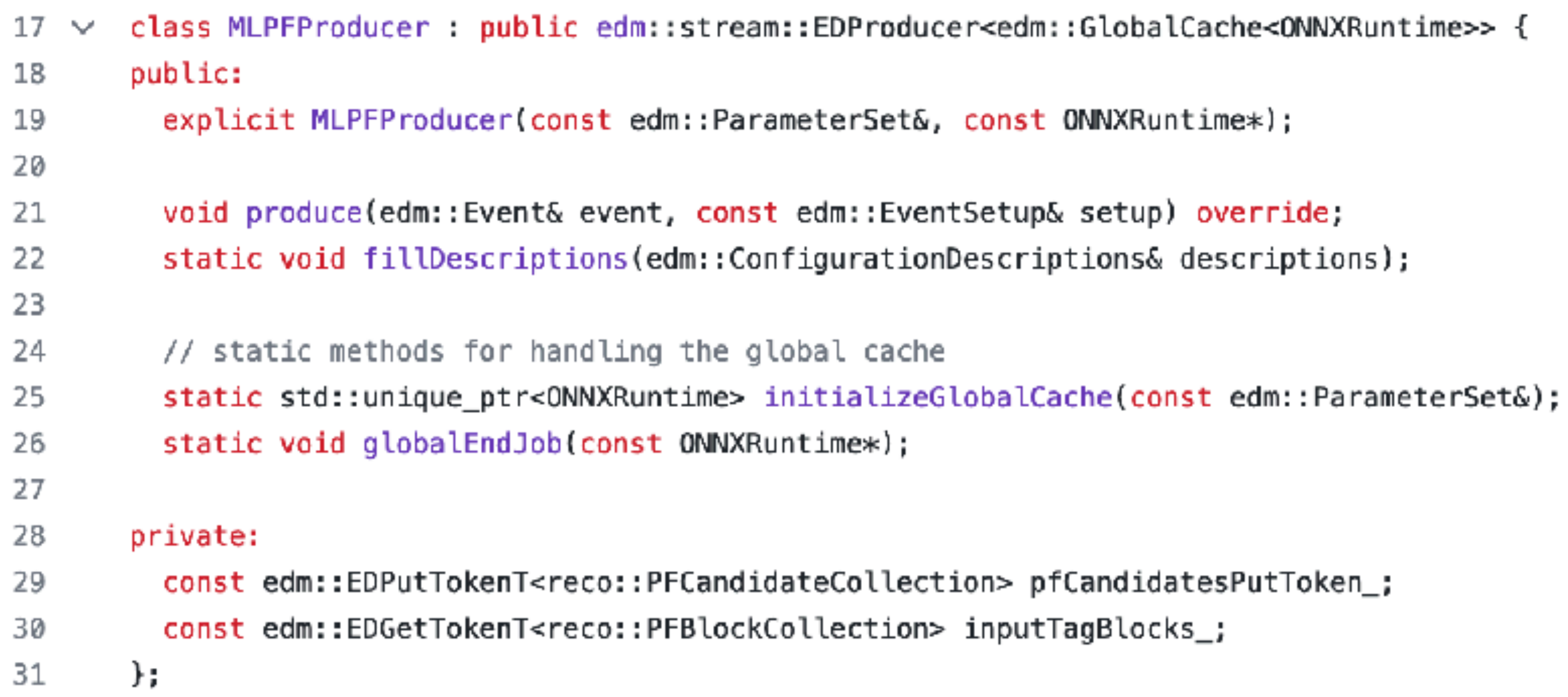
Datasets and software to reproduce the studies are published following the FAIR principles

1. The datasets are available in https://zenodo.org/record/8260741
2. The results in https://zenodo.org/record/8328683
3. The software used for analysis in https://zenodo.org/record/8290119

Backup

CMS status (continued)

- MLPF is integrated with CMSSW since ~2 years and can be enabled with a flag to switch out standard PF as a testbed[1]

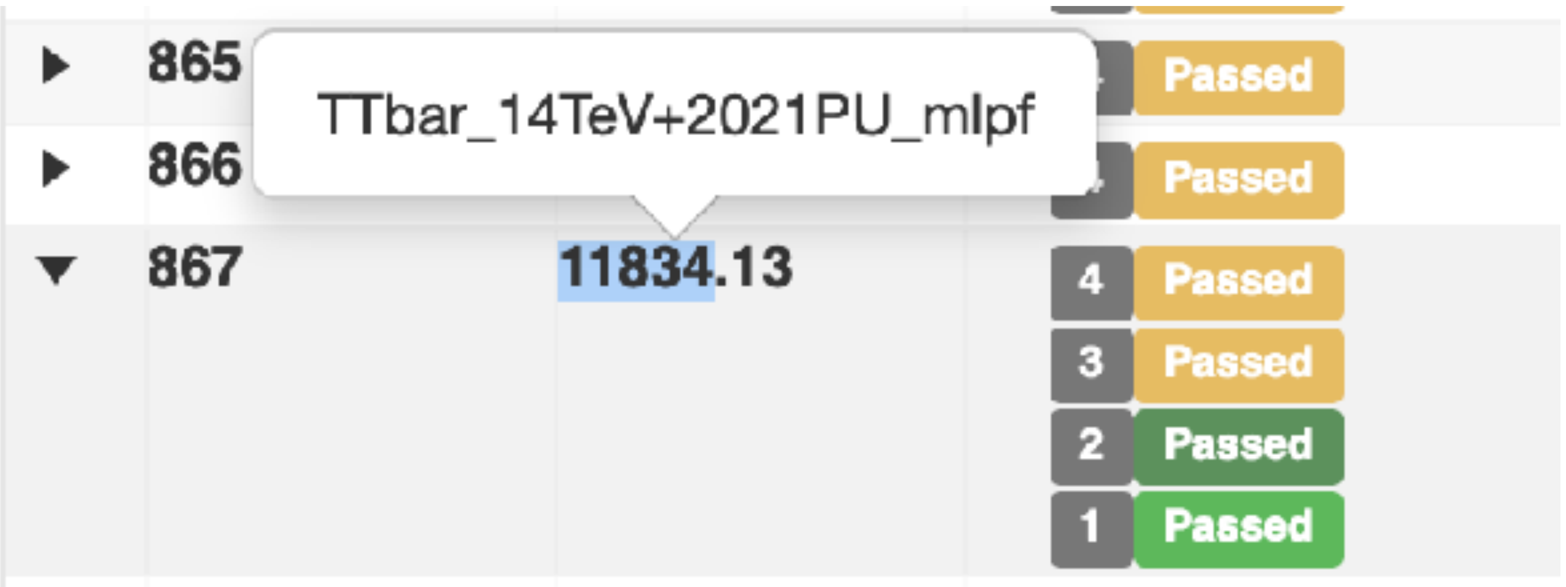


```cpp
class MLPFProducer : public edm::stream::EDProducer<edm::GlobalCache<ONNXRuntime>> {
public:
  explicit MLPFProducer(const edm::ParameterSet&, const ONNXRuntime*);

  void produce(edm::Event& event, const edm::EventSetup& setup) override;
  static void fillDescriptions(edm::ConfigurationDescriptions& descriptions);

  // static methods for handling the global cache
  static std::unique_ptr<ONNXRuntime> initializeGlobalCache(const edm::ParameterSet&);
  static void globalEndJob(const ONNXRuntime*);

private:
  const edm::EDPutTokenT<reco::PFCandidateCollection> pfCandidatesPutToken_;
  const edm::EDGetTokenT<reco::PFBlockCollection> inputTagBlocks_;
};
```



TTbar_14TeV+2021PU_mlpf

▶	865		Passed
▶	866		Passed
▼	867	11834.13	4 Passed 3 Passed 2 Passed 1 Passed

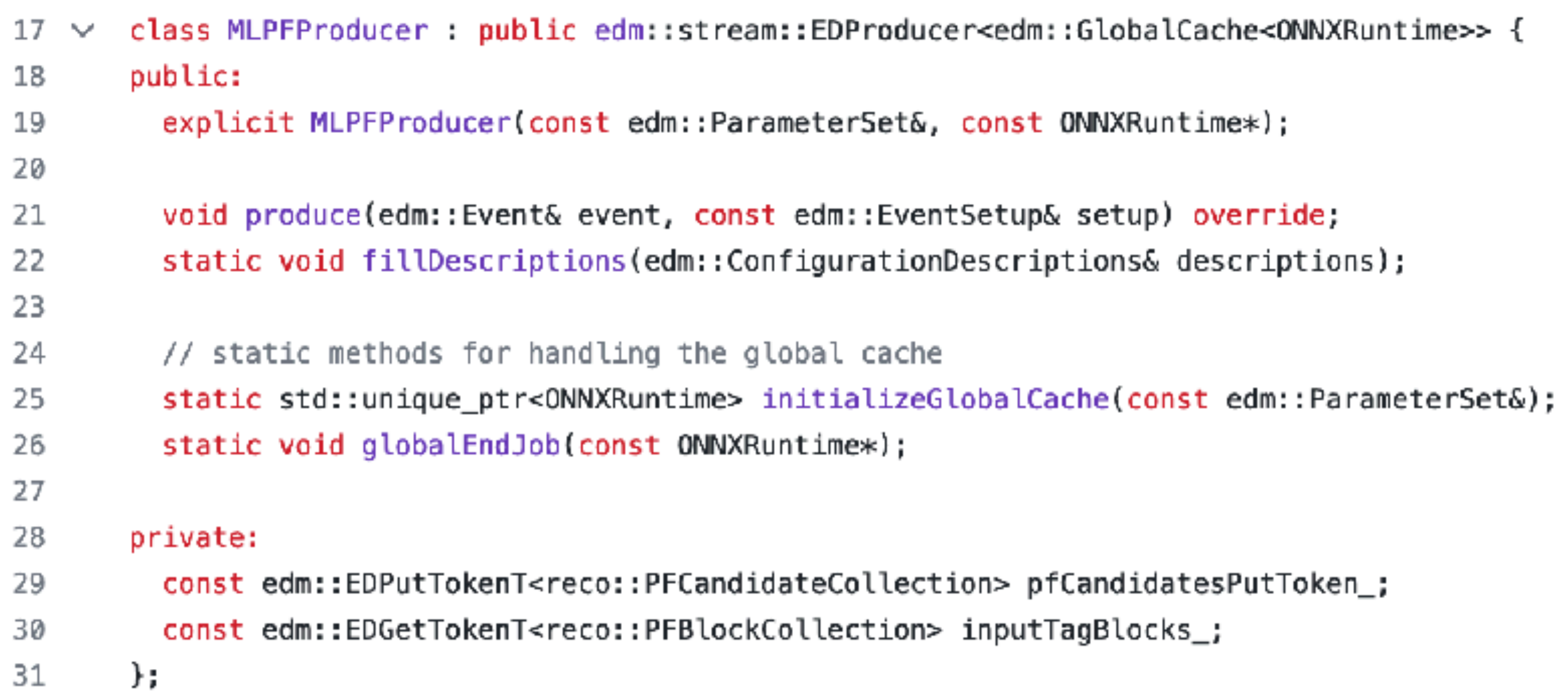
[1] https://github.com/cms-sw/cmssw/pull/36841

[2] cms-data/RecoParticleFlow-PFProducer

CMS status (continued)

- MLPF is integrated with CMSSW since ~2 years and can be enabled with a flag to switch out standard PF as a testbed[1]

- Running in RelVals as 11834.13 / _mlpf

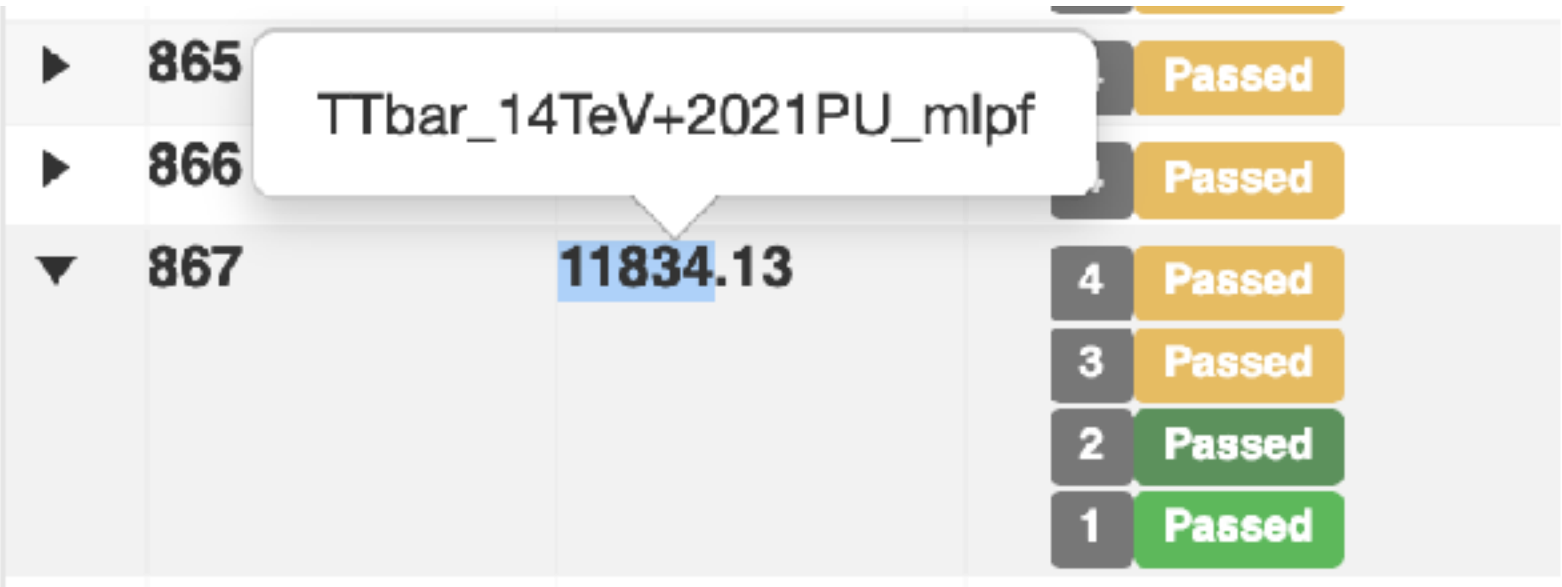


```
class MLPFProducer : public edm::stream::EDProducer<edm::GlobalCache<ONNXRuntime>> {
public:
  explicit MLPFProducer(const edm::ParameterSet&, const ONNXRuntime*);

  void produce(edm::Event& event, const edm::EventSetup& setup) override;
  static void fillDescriptions(edm::ConfigurationDescriptions& descriptions);

  // static methods for handling the global cache
  static std::unique_ptr<ONNXRuntime> initializeGlobalCache(const edm::ParameterSet&);
  static void globalEndJob(const ONNXRuntime*);

private:
  const edm::EDPutTokenT<reco::PFCandidateCollection> pfCandidatesPutToken_;
  const edm::EDGetTokenT<reco::PFBlockCollection> inputTagBlocks_;
};
```



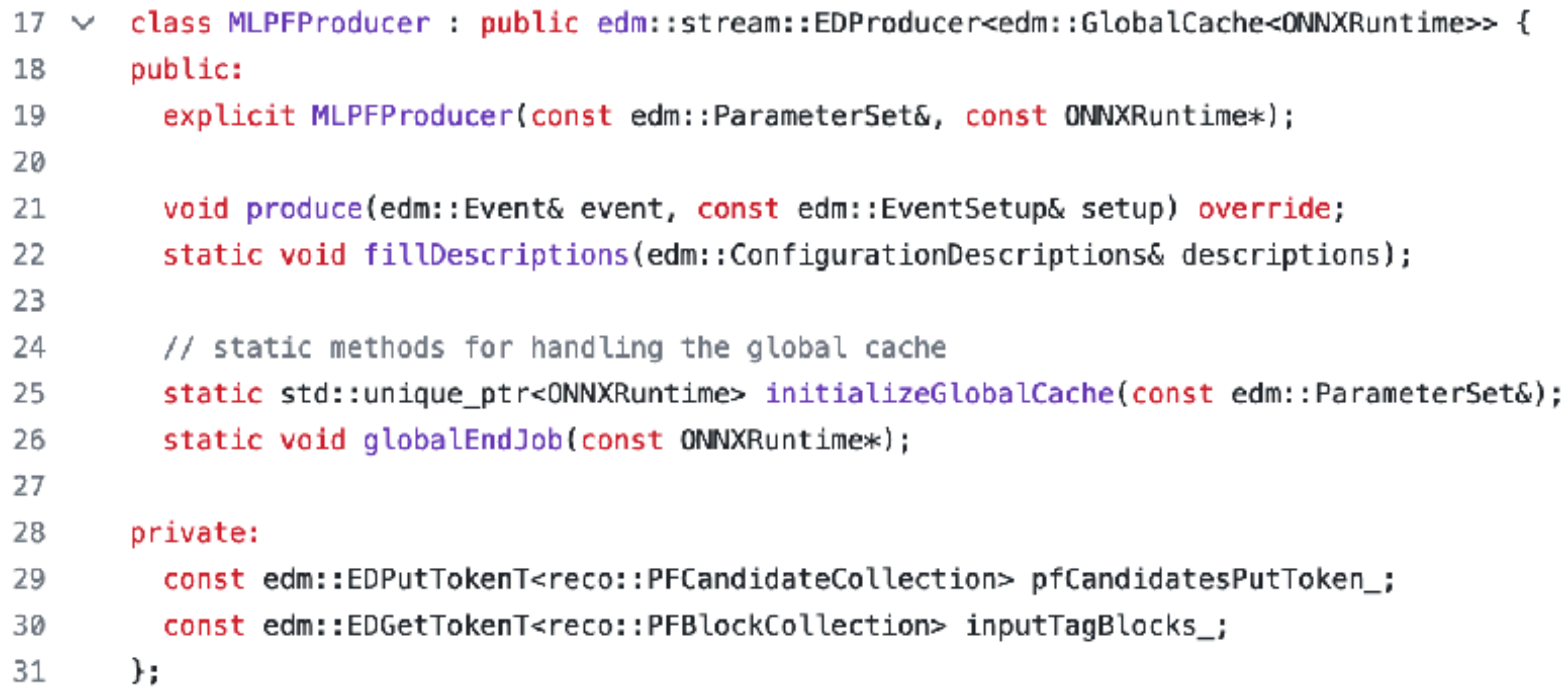
```
▶ 865                                  Passed
▶ 866     TTbar_14TeV+2021PU_mlpf      Passed
▼ 867     11834.13                   4 Passed
                                     3 Passed
                                     2 Passed
                                     1 Passed
```

[1] https://github.com/cms-sw/cmssw/pull/36841

[2] cms-data/RecoParticleFlow-PFProducer

CMS status (continued)

- MLPF is integrated with CMSSW since ~2 years and can be enabled with a flag to switch out standard PF as a testbed[1]
- Running in RelVals as 11834.13 / _mlpf
- We now need to provide an updated weight file[2] that improves physics performance sufficiently

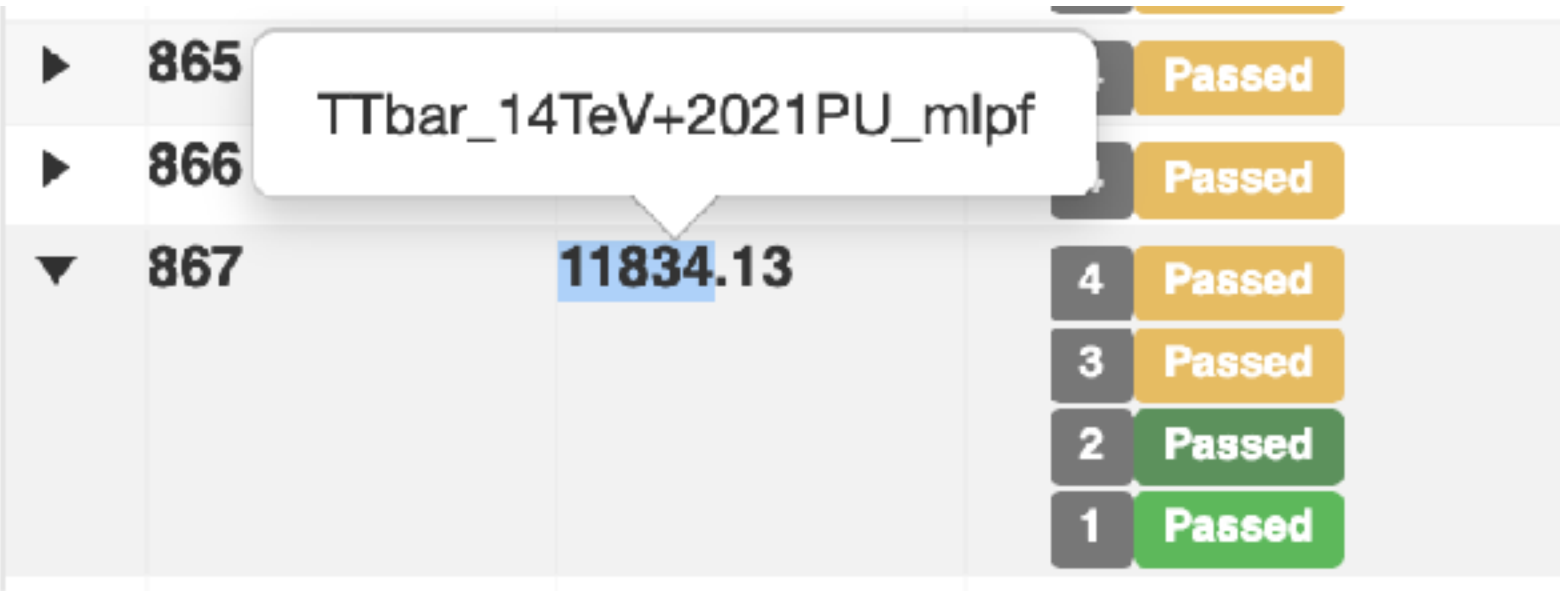


```
class MLPFProducer : public edm::stream::EDProducer<edm::GlobalCache<ONNXRuntime>> {
public:
  explicit MLPFProducer(const edm::ParameterSet&, const ONNXRuntime*);

  void produce(edm::Event& event, const edm::EventSetup& setup) override;
  static void fillDescriptions(edm::ConfigurationDescriptions& descriptions);

  // static methods for handling the global cache
  static std::unique_ptr<ONNXRuntime> initializeGlobalCache(const edm::ParameterSet&);
  static void globalEndJob(const ONNXRuntime*);

private:
  const edm::EDPutTokenT<reco::PFCandidateCollection> pfCandidatesPutToken_;
  const edm::EDGetTokenT<reco::PFBlockCollection> inputTagBlocks_;
};
```



▶ 865 Passed

▶ 866 Passed

TTbar_14TeV+2021PU_mlpf

▼ 867 11834.13

4 Passed

3 Passed

2 Passed

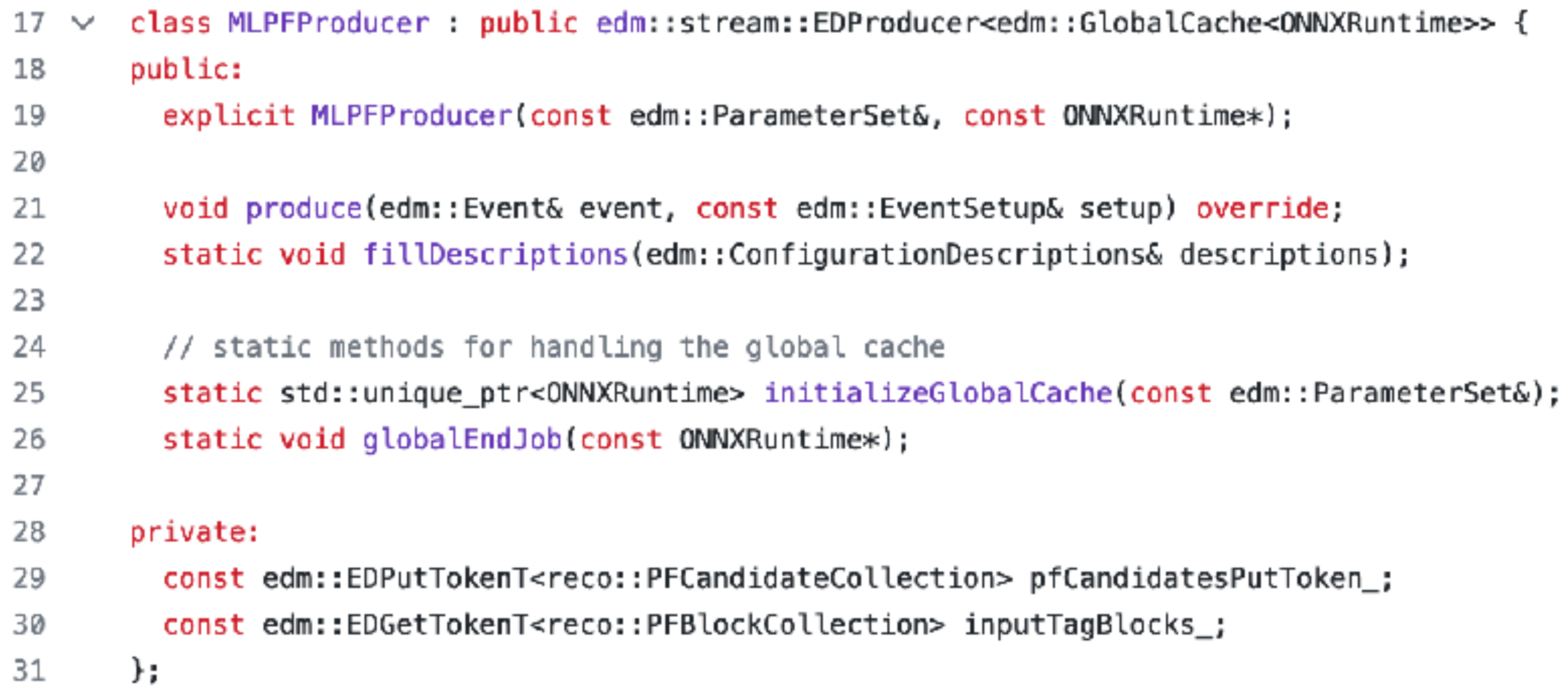
1 Passed

[1] https://github.com/cms-sw/cmssw/pull/36841

[2] cms-data/RecoParticleFlow-PFProducer

CMS status (continued)

- MLPF is integrated with CMSSW since ~2 years and can be enabled with a flag to switch out standard PF as a testbed[1]
- Running in RelVals as 11834.13 / _mlpf
- We now need to provide an updated weight file[2] that improves physics performance sufficiently
- This will be followed by optimizing the inference time on CPU & GPU in CMSSW by quantization and sparsification

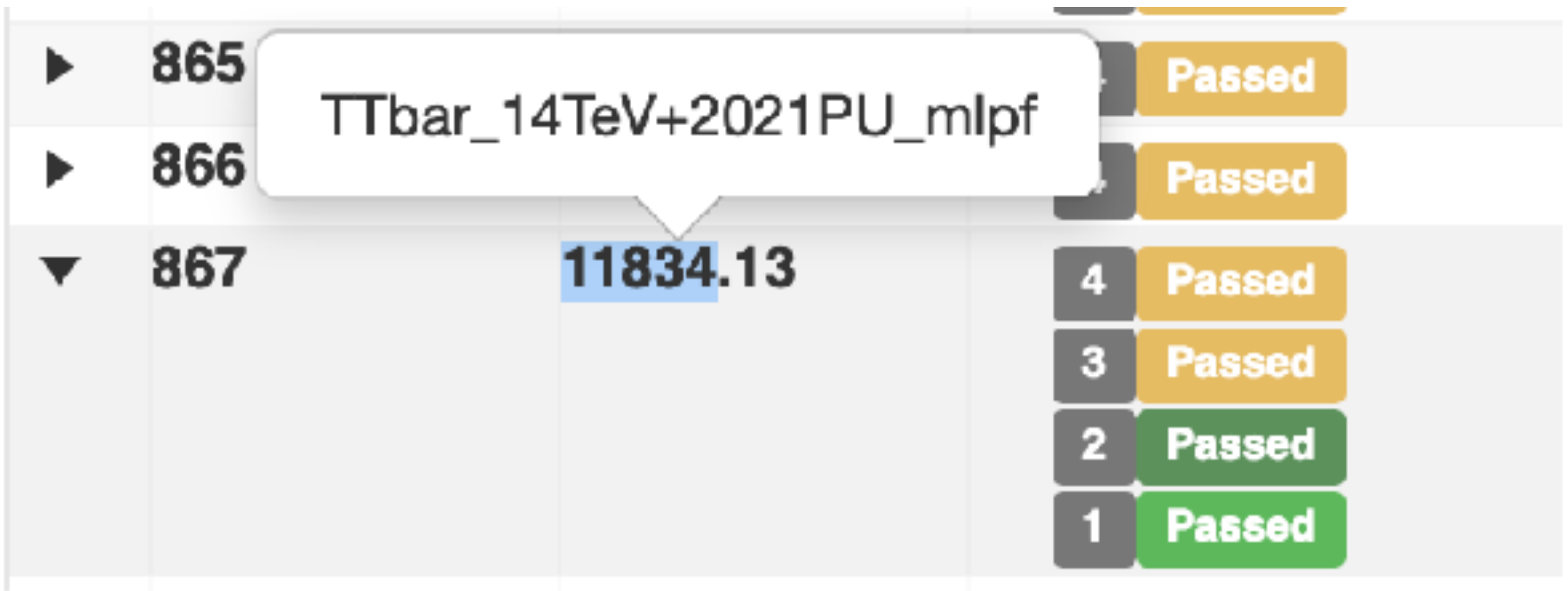


```
class MLPFProducer : public edm::stream::EDProducer<edm::GlobalCache<ONNXRuntime>> {
public:
  explicit MLPFProducer(const edm::ParameterSet&, const ONNXRuntime*);

  void produce(edm::Event& event, const edm::EventSetup& setup) override;
  static void fillDescriptions(edm::ConfigurationDescriptions& descriptions);

  // static methods for handling the global cache
  static std::unique_ptr<ONNXRuntime> initializeGlobalCache(const edm::ParameterSet&);
  static void globalEndJob(const ONNXRuntime*);

private:
  const edm::EDPutTokenT<reco::PFCandidateCollection> pfCandidatesPutToken_;
  const edm::EDGetTokenT<reco::PFBlockCollection> inputTagBlocks_;
};
```



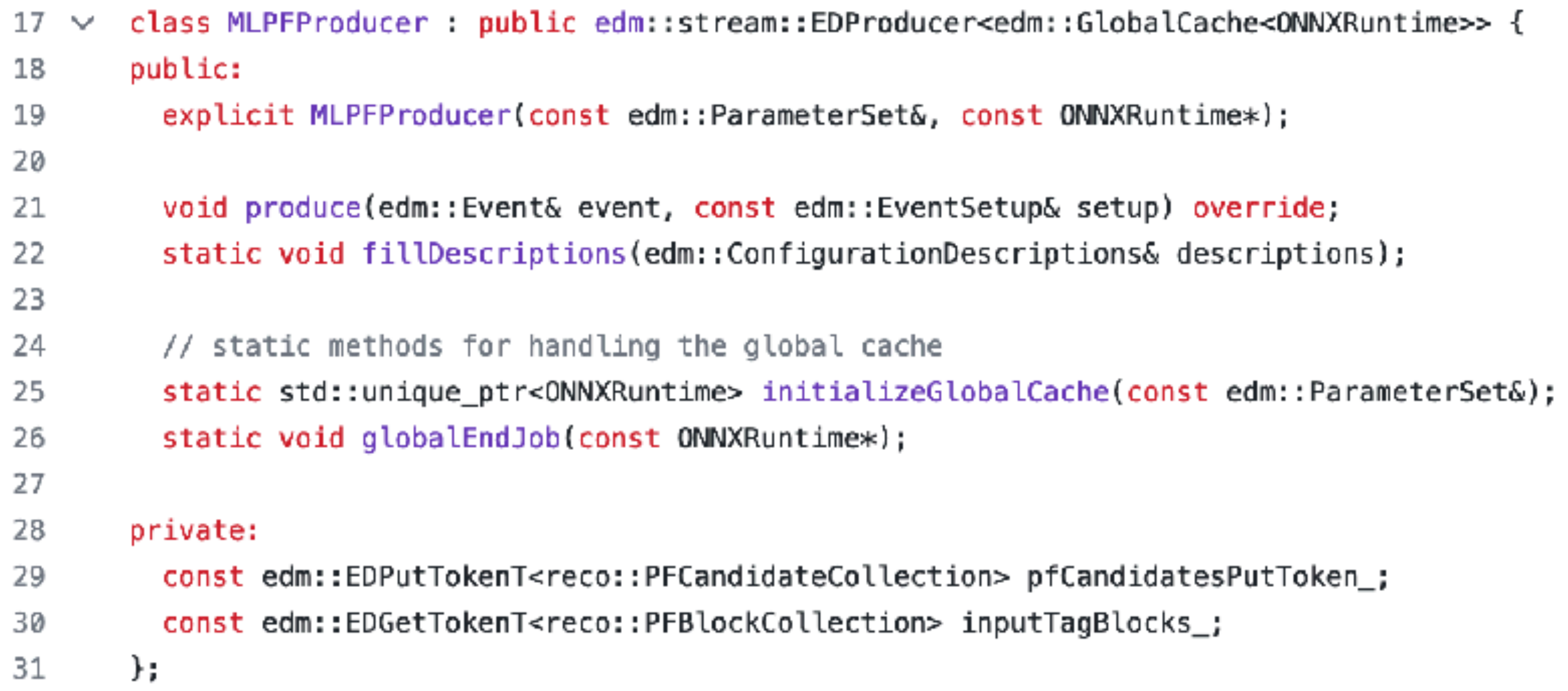
TTbar_14TeV+2021PU_mlpf

▶ 865			Passed
▶ 866			Passed
▼ 867	11834.13	4	Passed
		3	Passed
		2	Passed
		1	Passed

[1] https://github.com/cms-sw/cmssw/pull/36841
[2] cms-data/RecoParticleFlow-PFProducer

CMS status (continued)

- MLPF is integrated with CMSSW since ~2 years and can be enabled with a flag to switch out standard PF as a testbed[1]
- Running in RelVals as 11834.13 / _mlpf
- We now need to provide an updated weight file[2] that improves physics performance sufficiently
- This will be followed by optimizing the inference time on CPU & GPU in CMSSW by quantization and sparsification
- The input features, tests and inference have to be modernized, but can be done later as a refactor

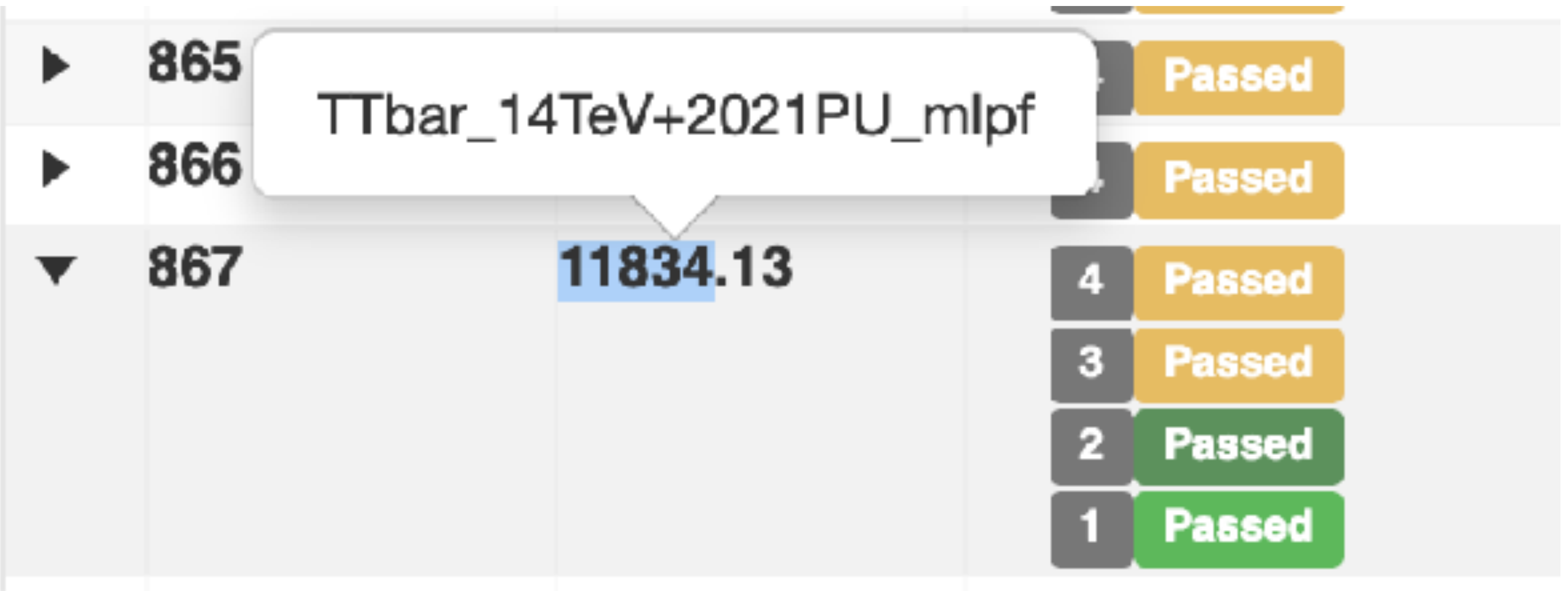


```
class MLPFProducer : public edm::stream::EDProducer<edm::GlobalCache<ONNXRuntime>> {
public:
  explicit MLPFProducer(const edm::ParameterSet&, const ONNXRuntime*);

  void produce(edm::Event& event, const edm::EventSetup& setup) override;
  static void fillDescriptions(edm::ConfigurationDescriptions& descriptions);

  // static methods for handling the global cache
  static std::unique_ptr<ONNXRuntime> initializeGlobalCache(const edm::ParameterSet&);
  static void globalEndJob(const ONNXRuntime*);

private:
  const edm::EDPutTokenT<reco::PFCandidateCollection> pfCandidatesPutToken_;
  const edm::EDGetTokenT<reco::PFBlockCollection> inputTagBlocks_;
};
```



▶ 865 Passed

▶ 866 Passed

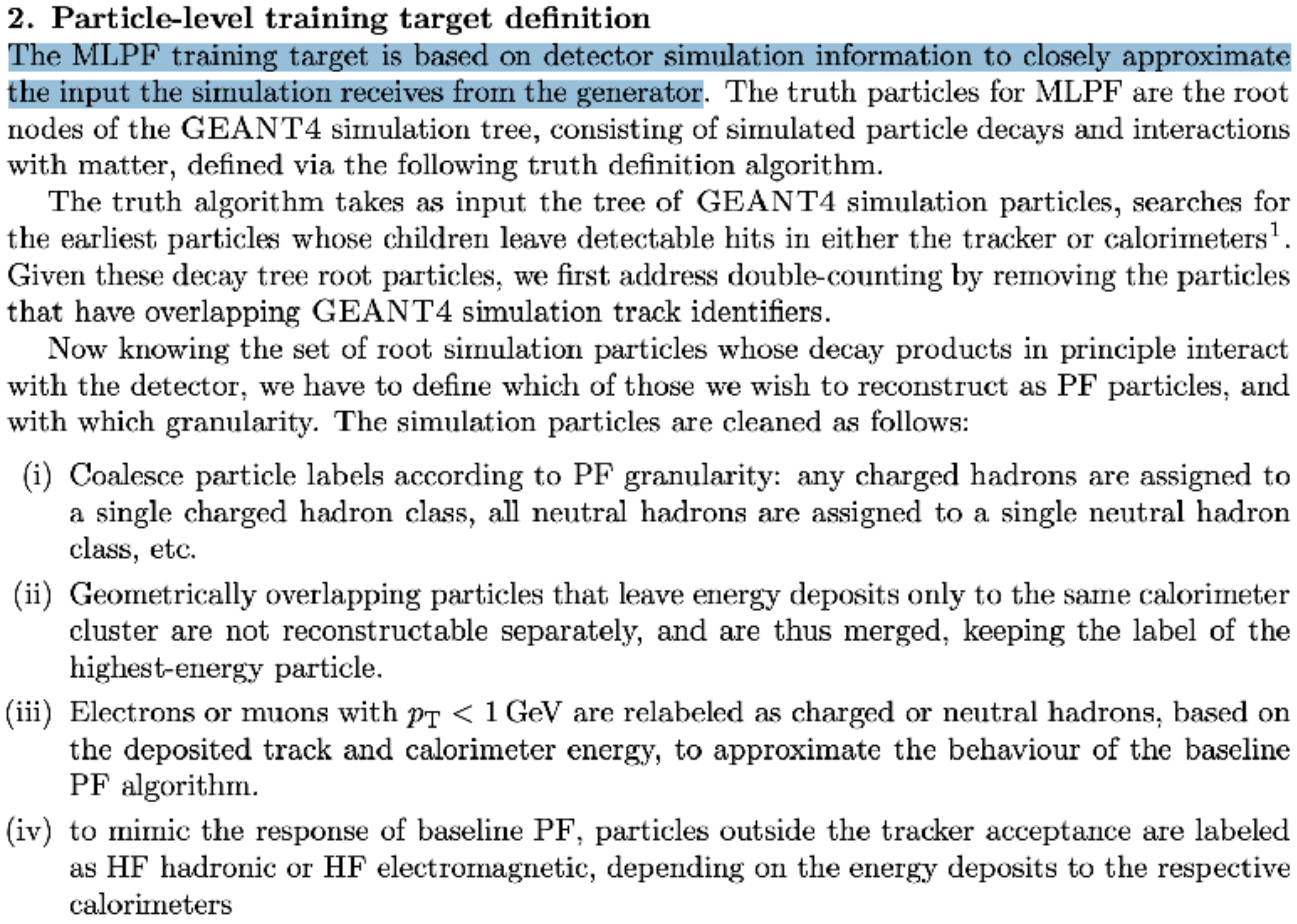
TTbar_14TeV+2021PU_mlpf

▼ 867 11834.13 4 Passed
3 Passed
2 Passed
1 Passed

[1] https://github.com/cms-sw/cmssw/pull/36841

[2] cms-data/RecoParticleFlow-PFProducer

CMS truth definition[1]



2. Particle-level training target definition

The MLPF training target is based on detector simulation information to closely approximate the input the simulation receives from the generator. The truth particles for MLPF are the root nodes of the GEANT4 simulation tree, consisting of simulated particle decays and interactions with matter, defined via the following truth definition algorithm.

The truth algorithm takes as input the tree of GEANT4 simulation particles, searches for the earliest particles whose children leave detectable hits in either the tracker or calorimeters[1]. Given these decay tree root particles, we first address double-counting by removing the particles that have overlapping GEANT4 simulation track identifiers.

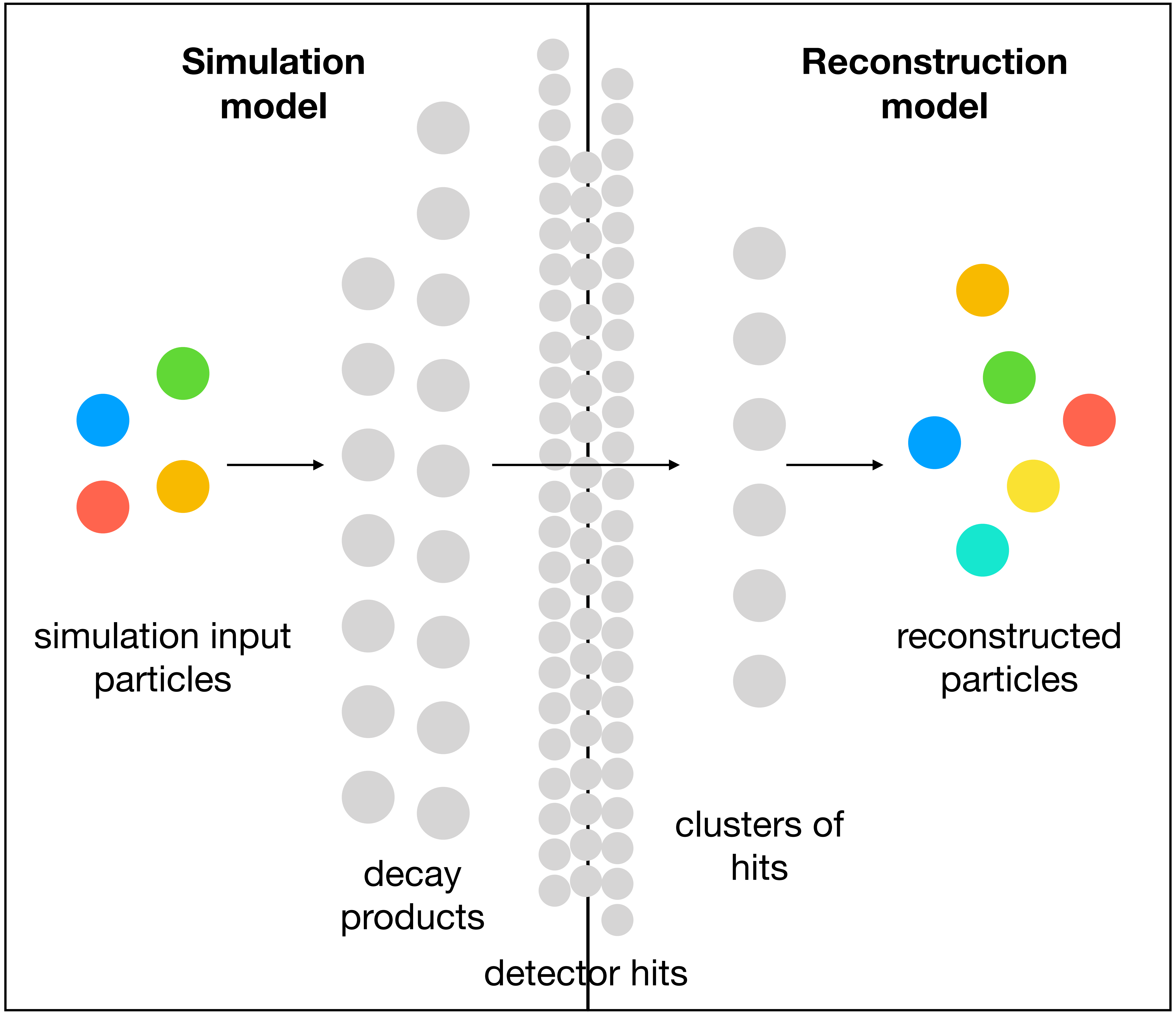
Now knowing the set of root simulation particles whose decay products in principle interact with the detector, we have to define which of those we wish to reconstruct as PF particles, and with which granularity. The simulation particles are cleaned as follows:

(i) Coalesce particle labels according to PF granularity: any charged hadrons are assigned to a single charged hadron class, all neutral hadrons are assigned to a single neutral hadron class, etc.

(ii) Geometrically overlapping particles that leave energy deposits only to the same calorimeter cluster are not reconstructable separately, and are thus merged, keeping the label of the highest-energy particle.

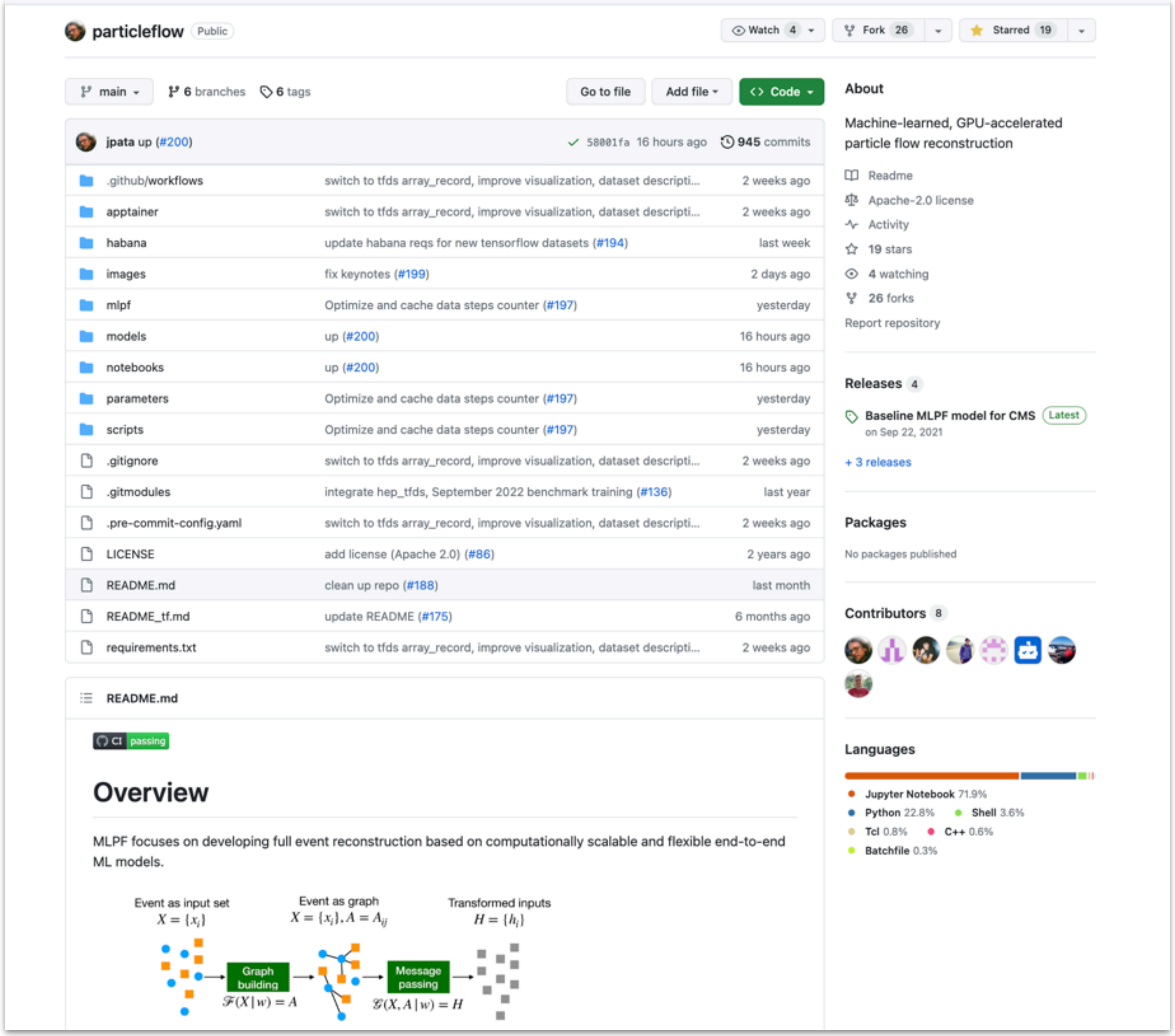
(iii) Electrons or muons with $p_T < 1$ GeV are relabeled as charged or neutral hadrons, based on the deposited track and calorimeter energy, to approximate the behaviour of the baseline PF algorithm.

(iv) to mimic the response of baseline PF, particles outside the tracker acceptance are labeled as HF hadronic or HF electromagnetic, depending on the energy deposits to the respective calorimeters



[1] https://arxiv.org/abs/2303.17657

More information on the code[1]



particleflow Public

Watch 4 · Fork 26 · Starred 19

main · 6 branches · 6 tags · Go to file · Add file · Code

jpata up (#200) ✓ 58001fa 16 hours ago · 945 commits

.github/workflows	switch to tfds array_record, improve visualization, dataset descripti...	2 weeks ago
apptainer	switch to tfds array_record, improve visualization, dataset descripti...	2 weeks ago
habana	update habana reqs for new tensorflow datasets (#194)	last week
images	fix keynotes (#199)	2 days ago
mlpf	Optimize and cache data steps counter (#197)	yesterday
models	up (#200)	16 hours ago
notebooks	up (#200)	16 hours ago
parameters	Optimize and cache data steps counter (#197)	yesterday
scripts	Optimize and cache data steps counter (#197)	yesterday
.gitignore	switch to tfds array_record, improve visualization, dataset descripti...	2 weeks ago
.gitmodules	integrate hep_tfds, September 2022 benchmark training (#136)	last year
.pre-commit-config.yaml	switch to tfds array_record, improve visualization, dataset descripti...	2 weeks ago
LICENSE	add license (Apache 2.0) (#86)	2 years ago
README.md	clean up repo (#188)	last month
README_tf.md	update README (#175)	6 months ago
requirements.txt	switch to tfds array_record, improve visualization, dataset descripti...	2 weeks ago

README.md

CI passing

Overview

MLPF focuses on developing full event reconstruction based on computationally scalable and flexible end-to-end ML models.

About

Machine-learned, GPU-accelerated particle flow reconstruction

- Readme
- Apache-2.0 license
- Activity
- 19 stars
- 4 watching
- 26 forks

Report repository

Releases 4

Baseline MLPF model for CMS Latest
on Sep 22, 2021

+ 3 releases

Packages

No packages published

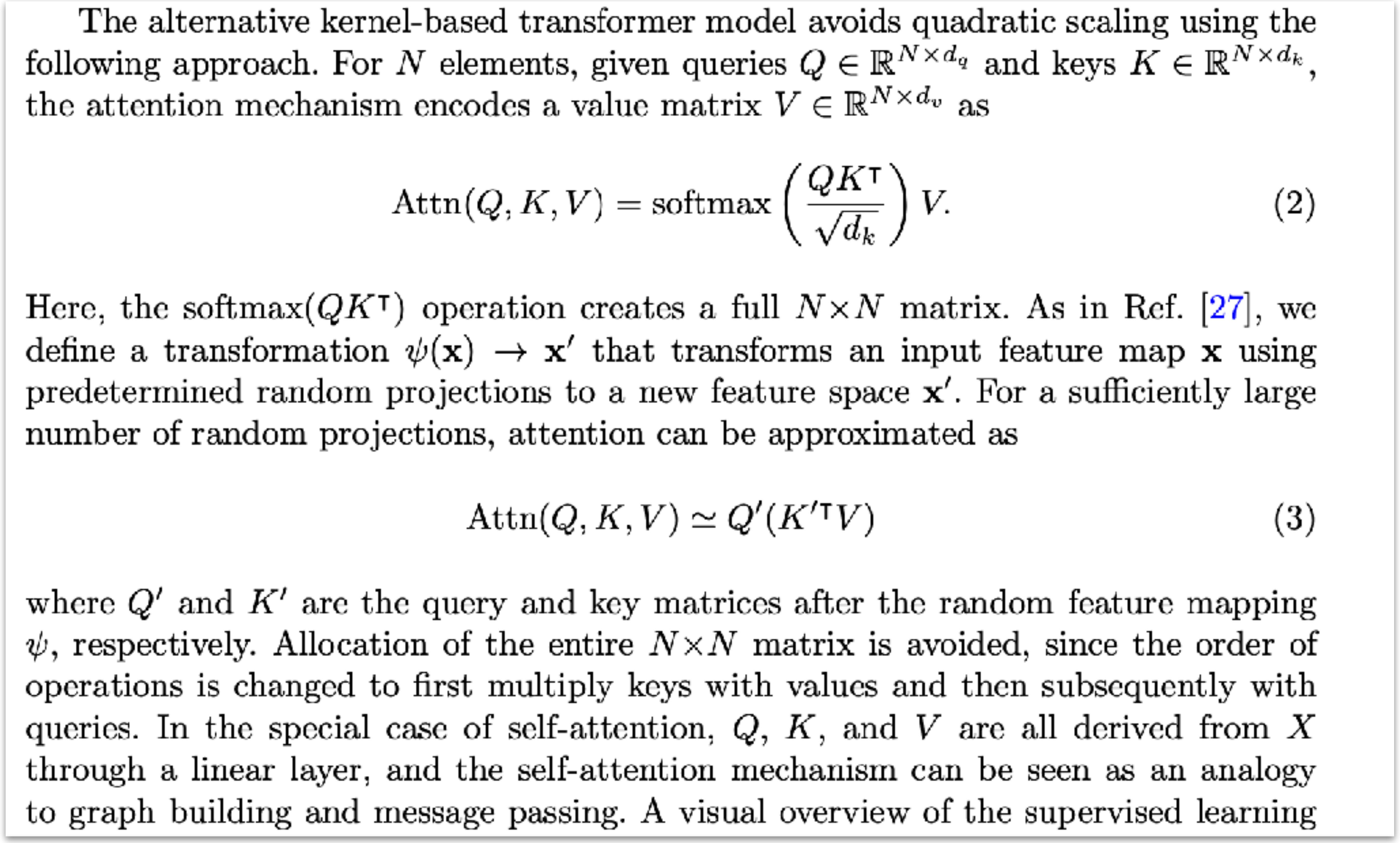
Contributors 8

Languages

- Jupyter Notebook 71.9%
- Python 22.8%
- Shell 3.6%
- Tcl 0.8%
- C++ 0.6%
- Batchfile 0.3%

[1] https://github.com/jpata/particleflow

More information on the Transformer model[1]



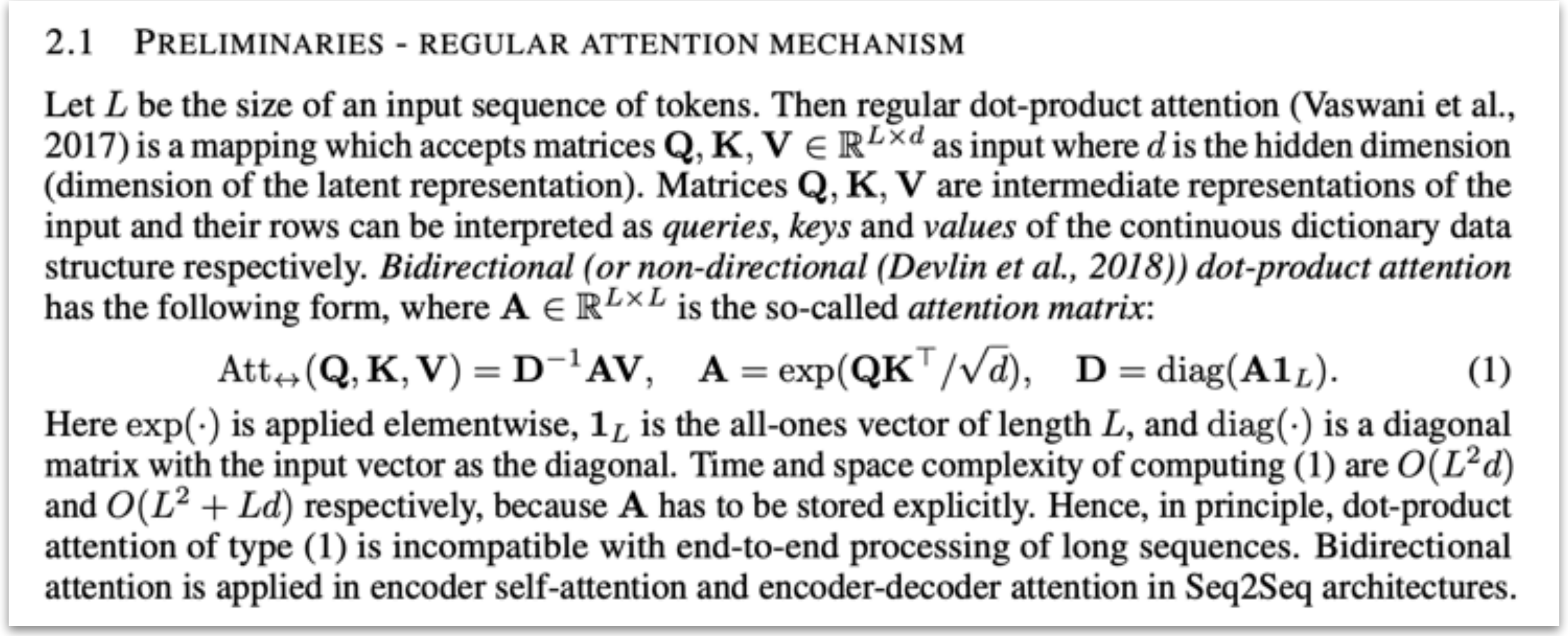
The alternative kernel-based transformer model avoids quadratic scaling using the following approach. For N elements, given queries $Q \in \mathbb{R}^{N\times d_q}$ and keys $K \in \mathbb{R}^{N\times d_k}$, the attention mechanism encodes a value matrix $V \in \mathbb{R}^{N\times d_v}$ as

$$\mathrm{Attn}(Q,K,V) = \mathrm{softmax}\left(\frac{QK^\intercal}{\sqrt{d_k}}\right)V. \tag{2}$$

Here, the softmax$(QK^\intercal)$ operation creates a full $N\times N$ matrix. As in Ref. [27], we define a transformation $\psi(\mathbf{x}) \to \mathbf{x}'$ that transforms an input feature map $\mathbf{x}$ using predetermined random projections to a new feature space $\mathbf{x}'$. For a sufficiently large number of random projections, attention can be approximated as

$$\mathrm{Attn}(Q,K,V) \simeq Q'(K'^\intercal V) \tag{3}$$

where Q' and K' are the query and key matrices after the random feature mapping ψ, respectively. Allocation of the entire $N\times N$ matrix is avoided, since the order of operations is changed to first multiply keys with values and then subsequently with queries. In the special case of self-attention, Q, K, and V are all derived from X through a linear layer, and the self-attention mechanism can be seen as an analogy to graph building and message passing. A visual overview of the supervised learning

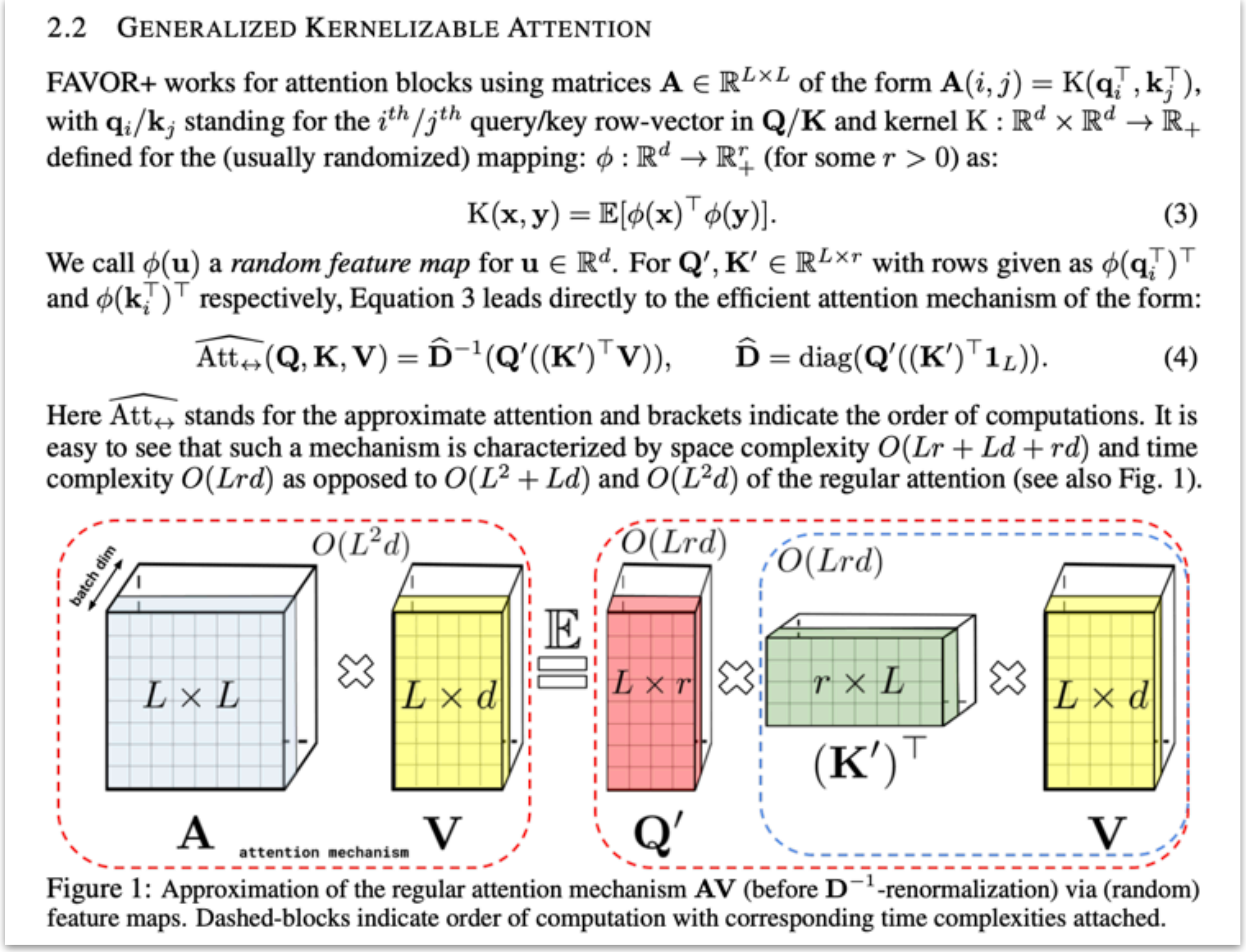


2.1 PRELIMINARIES - REGULAR ATTENTION MECHANISM

Let L be the size of an input sequence of tokens. Then regular dot-product attention (Vaswani et al., 2017) is a mapping which accepts matrices $\mathbf{Q},\mathbf{K},\mathbf{V} \in \mathbb{R}^{L\times d}$ as input where d is the hidden dimension (dimension of the latent representation). Matrices $\mathbf{Q},\mathbf{K},\mathbf{V}$ are intermediate representations of the input and their rows can be interpreted as *queries*, *keys* and *values* of the continuous dictionary data structure respectively. *Bidirectional (or non-directional (Devlin et al., 2018)) dot-product attention* has the following form, where $\mathbf{A} \in \mathbb{R}^{L\times L}$ is the so-called *attention matrix*:

$$\mathrm{Att}_{\leftrightarrow}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathbf{D}^{-1}\mathbf{A}\mathbf{V}, \quad \mathbf{A} = \exp(\mathbf{Q}\mathbf{K}^\top/\sqrt{d}), \quad \mathbf{D} = \mathrm{diag}(\mathbf{A}\mathbf{1}_L). \tag{1}$$

Here $\exp(\cdot)$ is applied elementwise, $\mathbf{1}_L$ is the all-ones vector of length L, and $\mathrm{diag}(\cdot)$ is a diagonal matrix with the input vector as the diagonal. Time and space complexity of computing (1) are $O(L^2d)$ and $O(L^2+Ld)$ respectively, because $\mathbf{A}$ has to be stored explicitly. Hence, in principle, dot-product attention of type (1) is incompatible with end-to-end processing of long sequences. Bidirectional attention is applied in encoder self-attention and encoder-decoder attention in Seq2Seq architectures.



2.2 GENERALIZED KERNELIZABLE ATTENTION

FAVOR+ works for attention blocks using matrices $\mathbf{A} \in \mathbb{R}^{L\times L}$ of the form $\mathbf{A}(i,j) = \mathrm{K}(\mathbf{q}_i^\top, \mathbf{k}_j^\top)$, with $\mathbf{q}_i/\mathbf{k}_j$ standing for the i^{th}/j^{th} query/key row-vector in $\mathbf{Q}/\mathbf{K}$ and kernel $\mathrm{K}: \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$ defined for the (usually randomized) mapping: $\phi: \mathbb{R}^d \to \mathbb{R}_+^r$ (for some $r > 0$) as:

$$\mathrm{K}(\mathbf{x},\mathbf{y}) = \mathbb{E}[\phi(\mathbf{x})^\top\phi(\mathbf{y})]. \tag{3}$$

We call $\phi(\mathbf{u})$ a *random feature map* for $\mathbf{u} \in \mathbb{R}^d$. For $\mathbf{Q}',\mathbf{K}' \in \mathbb{R}^{L\times r}$ with rows given as $\phi(\mathbf{q}_i^\top)^\top$ and $\phi(\mathbf{k}_i^\top)^\top$ respectively, Equation 3 leads directly to the efficient attention mechanism of the form:

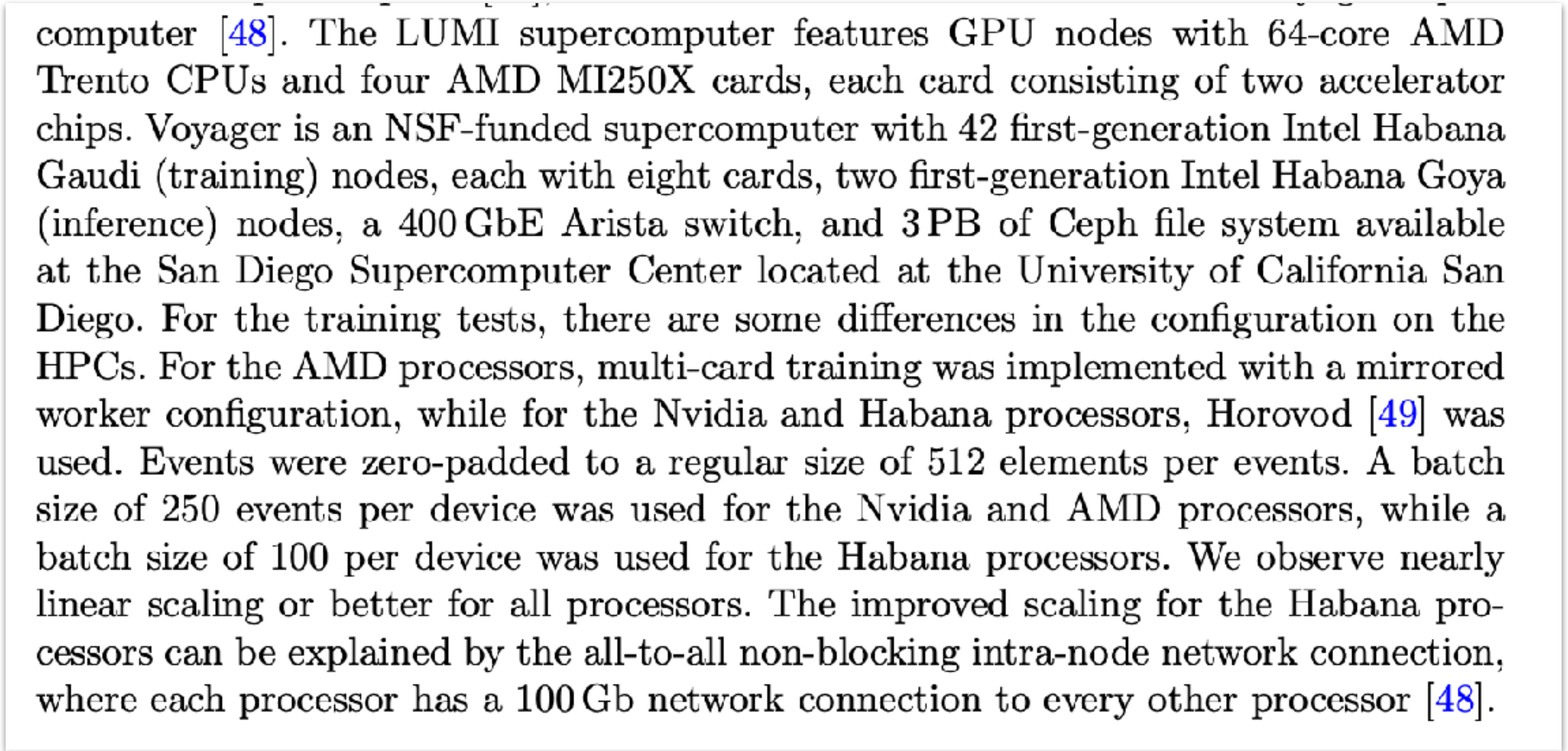
$$\widehat{\mathrm{Att}_{\leftrightarrow}}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \widehat{\mathbf{D}}^{-1}(\mathbf{Q}'((\mathbf{K}')^\top\mathbf{V})), \qquad \widehat{\mathbf{D}} = \mathrm{diag}(\mathbf{Q}'((\mathbf{K}')^\top\mathbf{1}_L)). \tag{4}$$

Here $\widehat{\mathrm{Att}_{\leftrightarrow}}$ stands for the approximate attention and brackets indicate the order of computations. It is easy to see that such a mechanism is characterized by space complexity $O(Lr+Ld+rd)$ and time complexity $O(Lrd)$ as opposed to $O(L^2+Ld)$ and $O(L^2d)$ of the regular attention (see also Fig. 1).

Figure 1: Approximation of the regular attention mechanism $\mathbf{AV}$ (before $\mathbf{D}^{-1}$-renormalization) via (random) feature maps. Dashed-blocks indicate order of computation with corresponding time complexities attached.

[1] K. Choromanski et al., "Rethinking attention with performers", 2020

More information on the supercomputers[1-3]



computer [48]. The LUMI supercomputer features GPU nodes with 64-core AMD Trento CPUs and four AMD MI250X cards, each card consisting of two accelerator chips. Voyager is an NSF-funded supercomputer with 42 first-generation Intel Habana Gaudi (training) nodes, each with eight cards, two first-generation Intel Habana Goya (inference) nodes, a 400 GbE Arista switch, and 3 PB of Ceph file system available at the San Diego Supercomputer Center located at the University of California San Diego. For the training tests, there are some differences in the configuration on the HPCs. For the AMD processors, multi-card training was implemented with a mirrored worker configuration, while for the Nvidia and Habana processors, Horovod [49] was used. Events were zero-padded to a regular size of 512 elements per events. A batch size of 250 events per device was used for the Nvidia and AMD processors, while a batch size of 100 per device was used for the Habana processors. We observe nearly linear scaling or better for all processors. The improved scaling for the Habana processors can be explained by the all-to-all non-blocking intra-node network connection, where each processor has a 100 Gb network connection to every other processor [48].

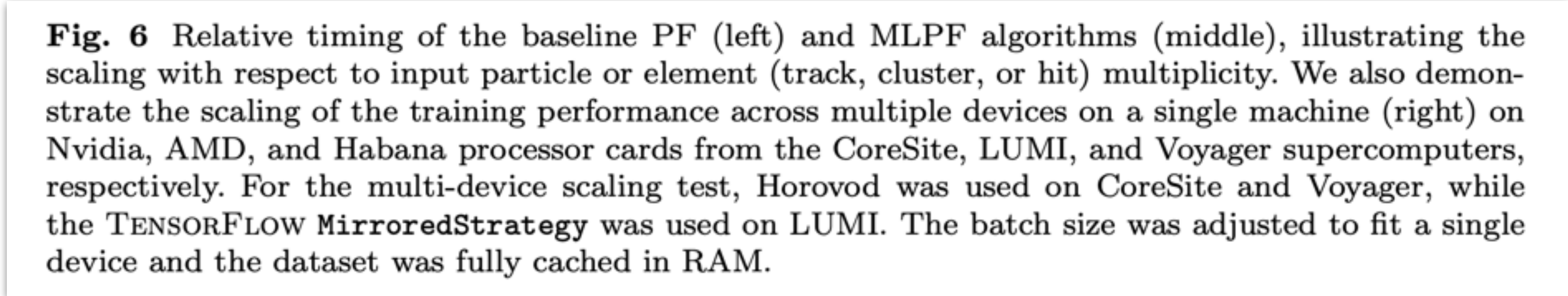


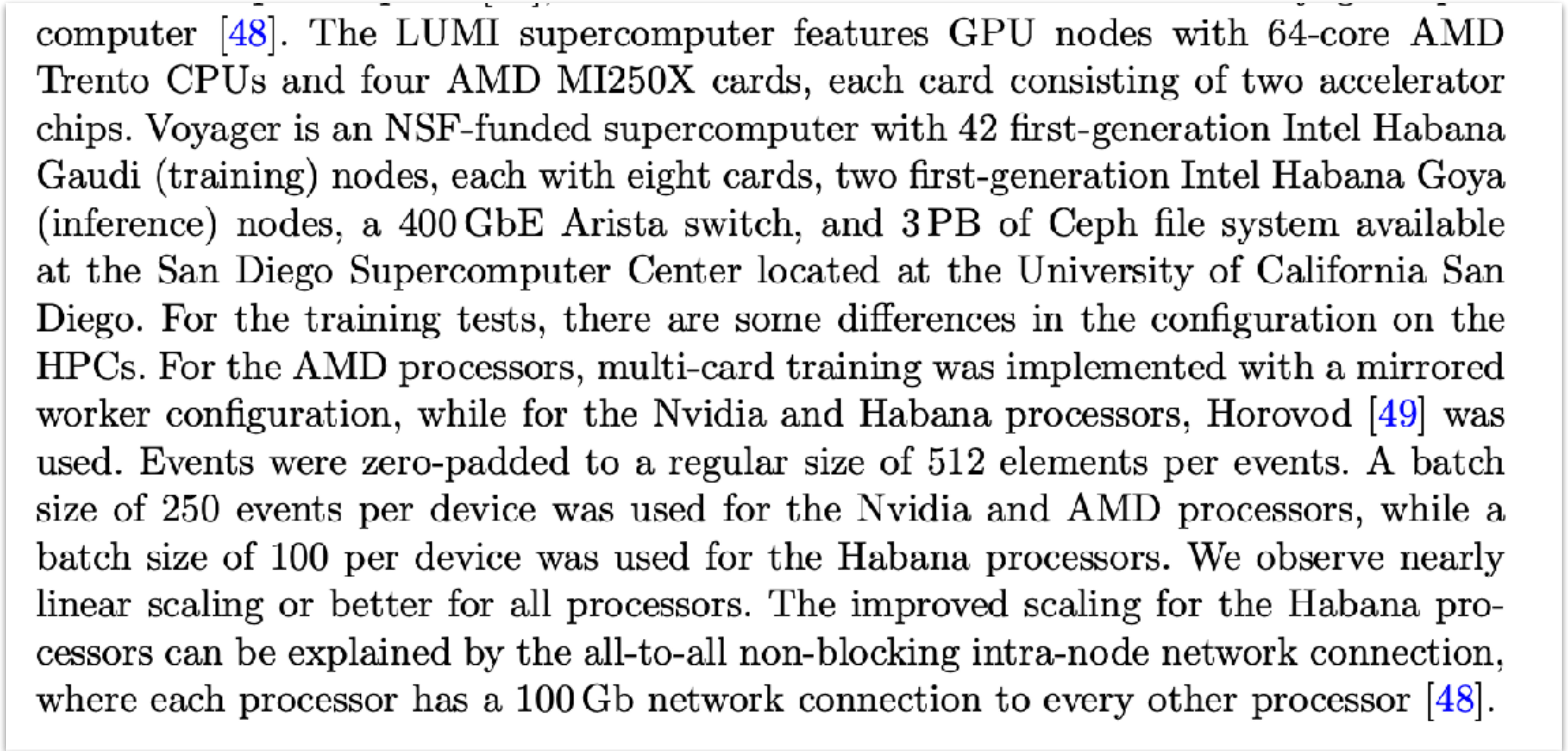
Fig. 6 Relative timing of the baseline PF (left) and MLPF algorithms (middle), illustrating the scaling with respect to input particle or element (track, cluster, or hit) multiplicity. We also demonstrate the scaling of the training performance across multiple devices on a single machine (right) on Nvidia, AMD, and Habana processor cards from the CoreSite, LUMI, and Voyager supercomputers, respectively. For the multi-device scaling test, Horovod was used on CoreSite and Voyager, while the TensorFlow `MirroredStrategy` was used on LUMI. The batch size was adjusted to fit a single device and the dataset was fully cached in RAM.

[1] Flatiron Institute, "CoreSite Cluster", 2023

[2] LUMI Consortium, "LUMI Supercomputer", 2023

[3] Voyager supercomputer: https://dl.acm.org/doi/10.1145/3569951.3597597

More information on the supercomputers[1-3]



computer [48]. The LUMI supercomputer features GPU nodes with 64-core AMD Trento CPUs and four AMD MI250X cards, each card consisting of two accelerator chips. Voyager is an NSF-funded supercomputer with 42 first-generation Intel Habana Gaudi (training) nodes, each with eight cards, two first-generation Intel Habana Goya (inference) nodes, a 400 GbE Arista switch, and 3 PB of Ceph file system available at the San Diego Supercomputer Center located at the University of California San Diego. For the training tests, there are some differences in the configuration on the HPCs. For the AMD processors, multi-card training was implemented with a mirrored worker configuration, while for the Nvidia and Habana processors, Horovod [49] was used. Events were zero-padded to a regular size of 512 elements per events. A batch size of 250 events per device was used for the Nvidia and AMD processors, while a batch size of 100 per device was used for the Habana processors. We observe nearly linear scaling or better for all processors. The improved scaling for the Habana processors can be explained by the all-to-all non-blocking intra-node network connection, where each processor has a 100 Gb network connection to every other processor [48].

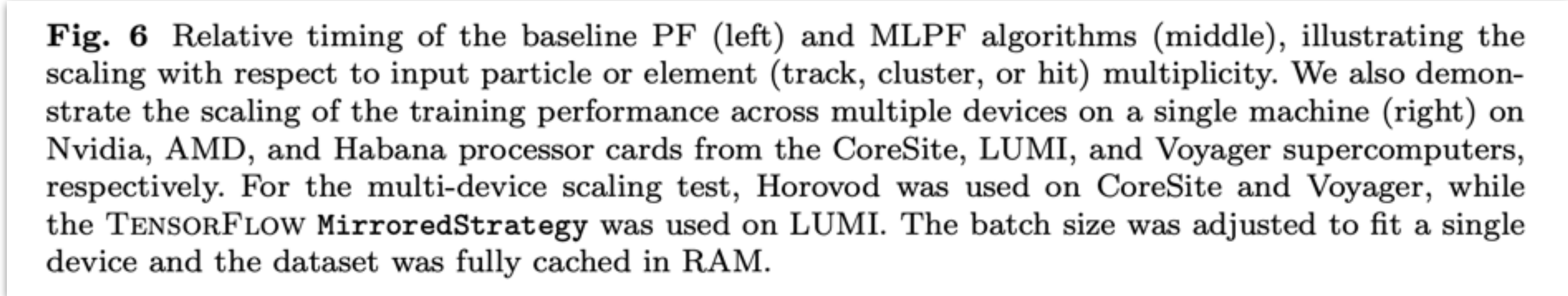


Fig. 6 Relative timing of the baseline PF (left) and MLPF algorithms (middle), illustrating the scaling with respect to input particle or element (track, cluster, or hit) multiplicity. We also demonstrate the scaling of the training performance across multiple devices on a single machine (right) on Nvidia, AMD, and Habana processor cards from the CoreSite, LUMI, and Voyager supercomputers, respectively. For the multi-device scaling test, Horovod was used on CoreSite and Voyager, while the TensorFlow `MirroredStrategy` was used on LUMI. The batch size was adjusted to fit a single device and the dataset was fully cached in RAM.

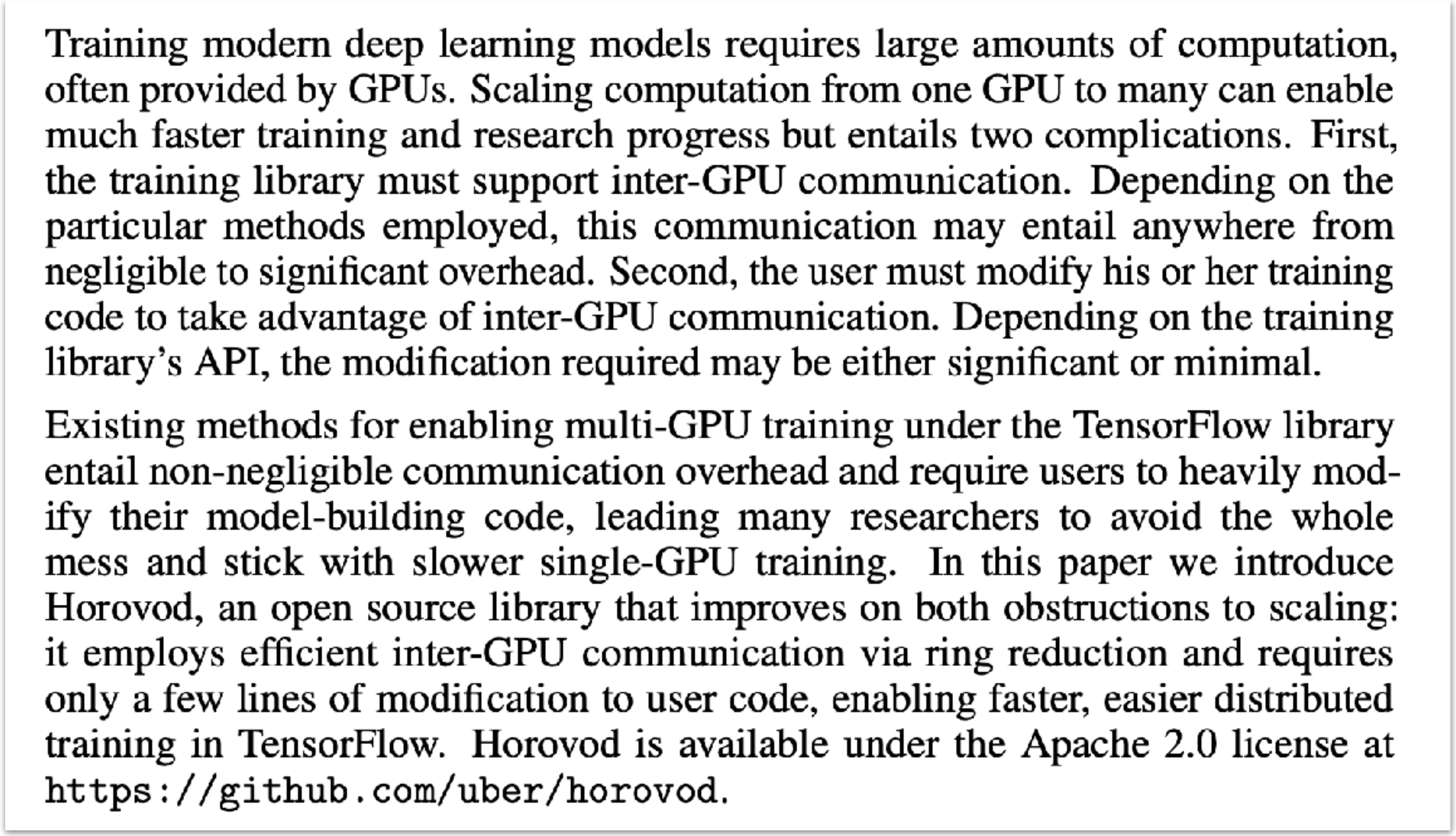
→ Training information

[1] Flatiron Institute, “CoreSite Cluster”, 2023

[2] LUMI Consortium, “LUMI Supercomputer”, 2023

[3] Voyager supercomputer: https://dl.acm.org/doi/10.1145/3569951.3597597

More information on Horovod[1]



Training modern deep learning models requires large amounts of computation, often provided by GPUs. Scaling computation from one GPU to many can enable much faster training and research progress but entails two complications. First, the training library must support inter-GPU communication. Depending on the particular methods employed, this communication may entail anywhere from negligible to significant overhead. Second, the user must modify his or her training code to take advantage of inter-GPU communication. Depending on the training library's API, the modification required may be either significant or minimal.

Existing methods for enabling multi-GPU training under the TensorFlow library entail non-negligible communication overhead and require users to heavily modify their model-building code, leading many researchers to avoid the whole mess and stick with slower single-GPU training. In this paper we introduce Horovod, an open source library that improves on both obstructions to scaling: it employs efficient inter-GPU communication via ring reduction and requires only a few lines of modification to user code, enabling faster, easier distributed training in TensorFlow. Horovod is available under the Apache 2.0 license at `https://github.com/uber/horovod`.

[1] https://arxiv.org/pdf/1802.05799.pdf