JAVIER DUARTE

MIT 8.S50

JANUARY 27, 2021

GRAPH NEURAL NETWORKS FOR PARTICLE PHYSICS

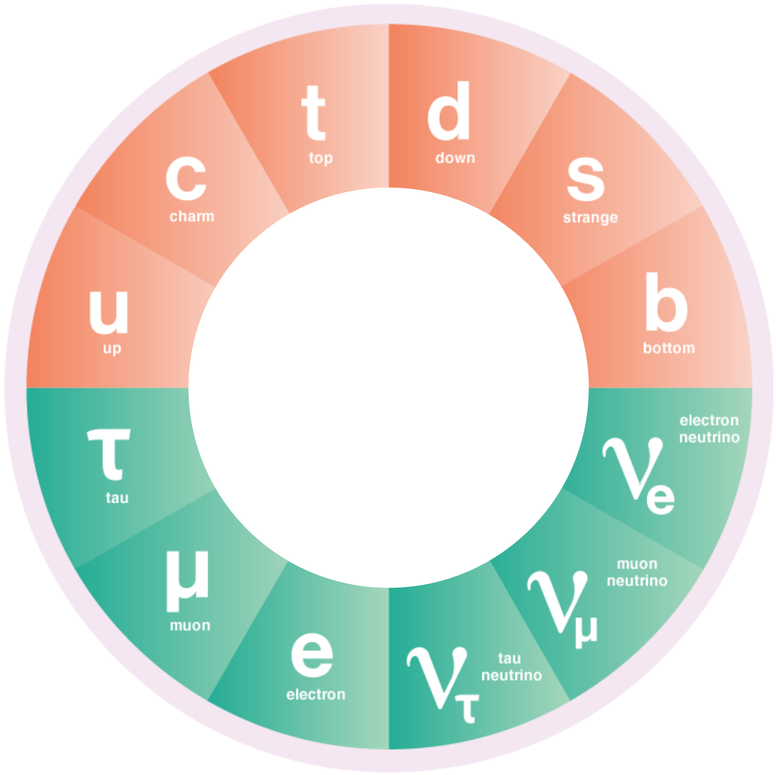
UCSD

MIT CONNECTIONS

- Undergrad at MIT, Courses 8 & 18 (2006–2010)

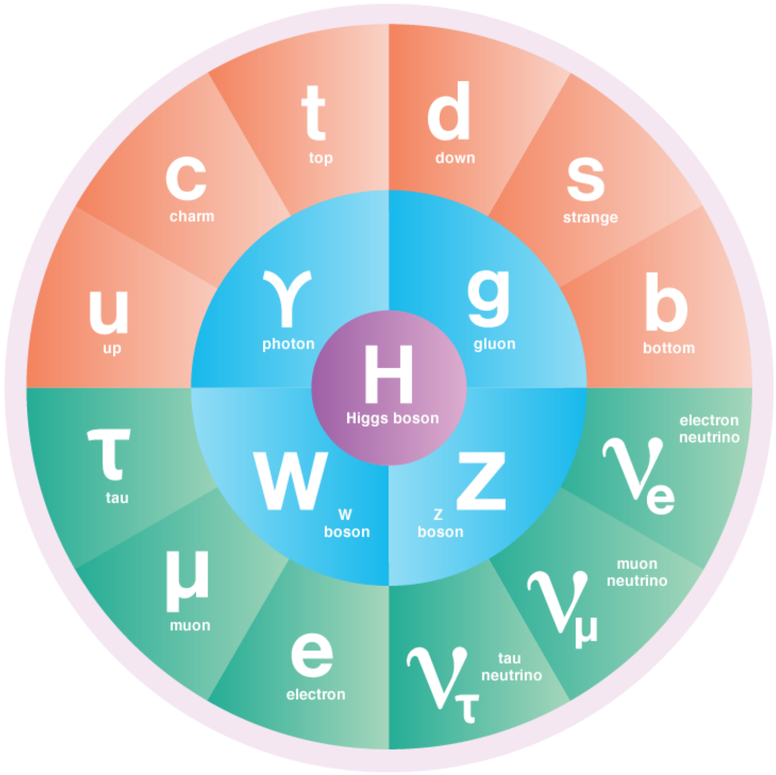
- Technical Instructor for Junior Lab (2010–2011)

THE STANDARD MODEL (OR WHY STUDY PARTICLE PHYSICS?)



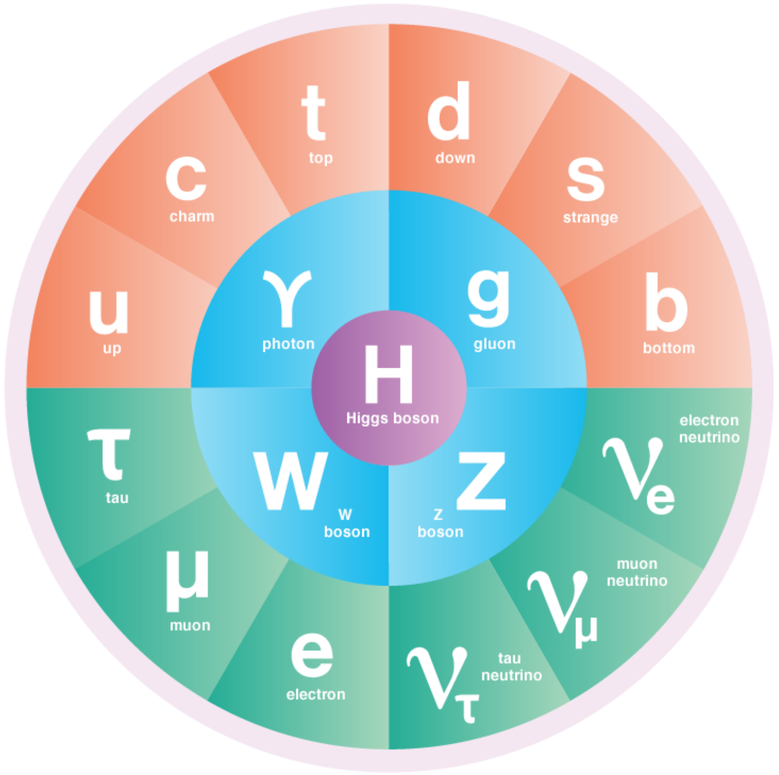
particles

THE STANDARD MODEL (OR WHY STUDY PARTICLE PHYSICS?)



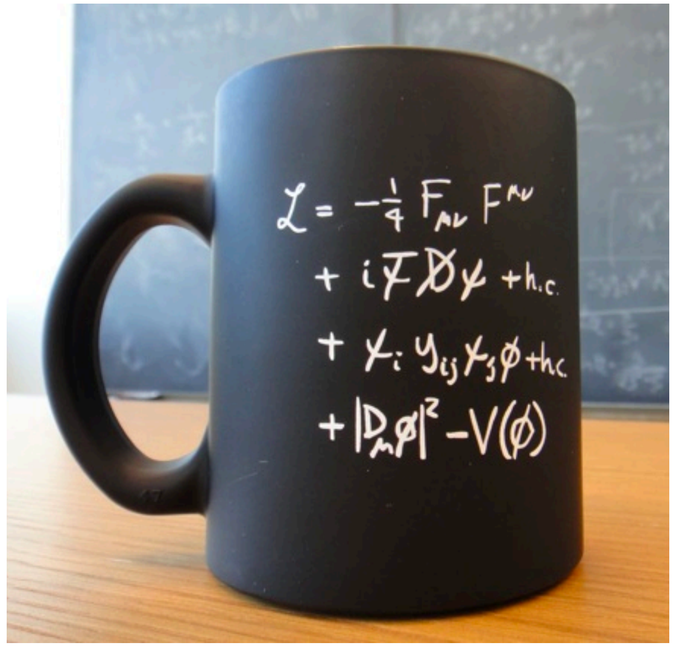
particles

THE STANDARD MODEL (OR WHY STUDY PARTICLE PHYSICS?)



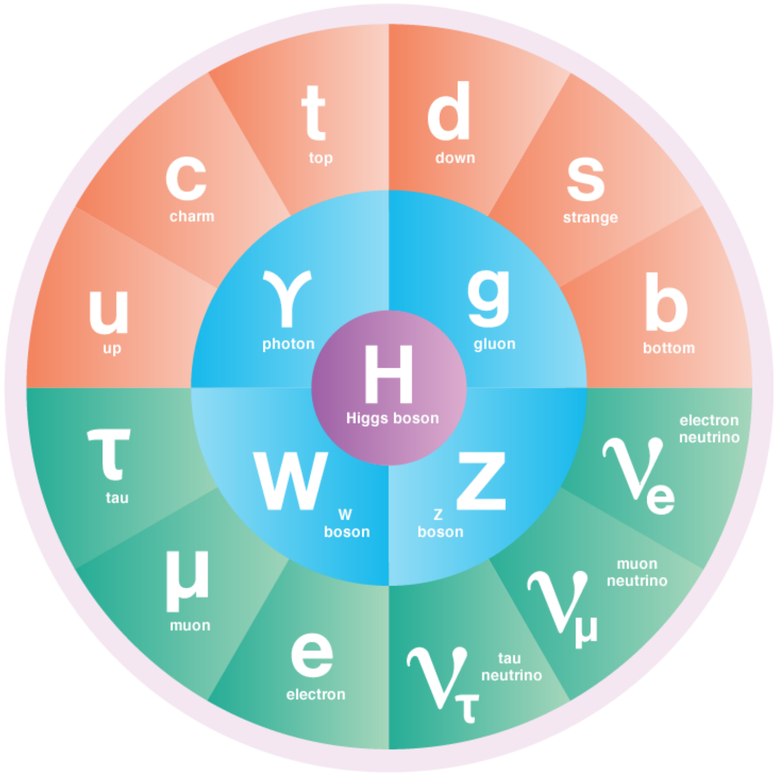
particles

+

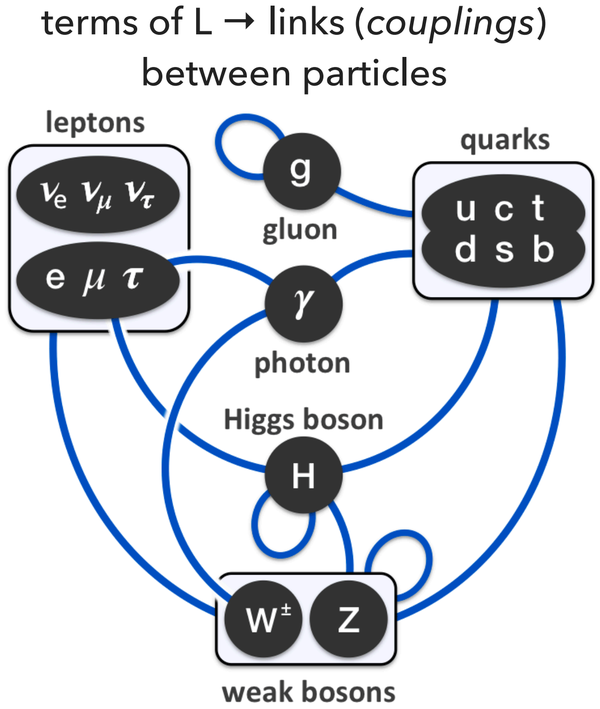


interactions

THE STANDARD MODEL (OR WHY STUDY PARTICLE PHYSICS?)



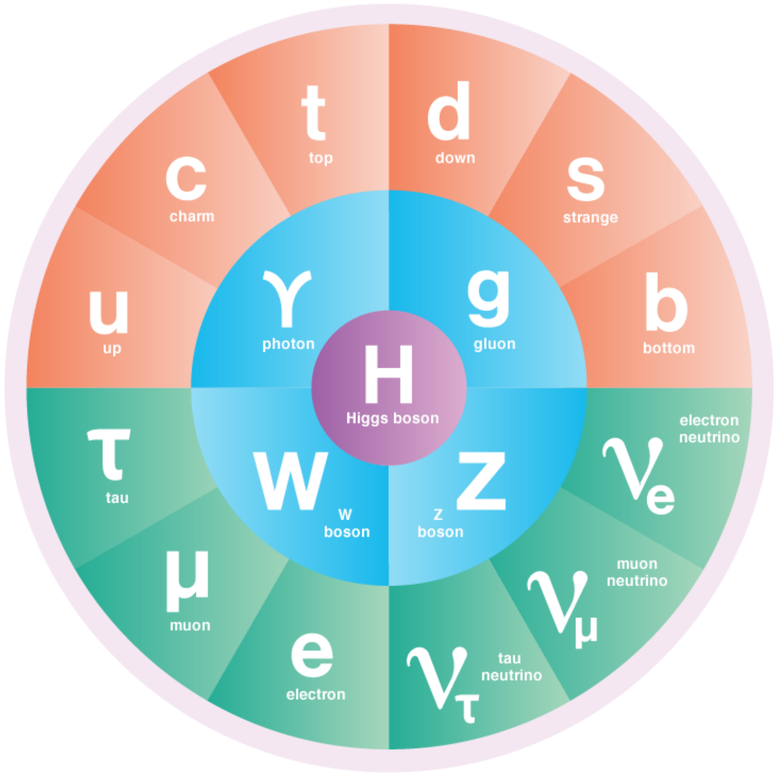
+



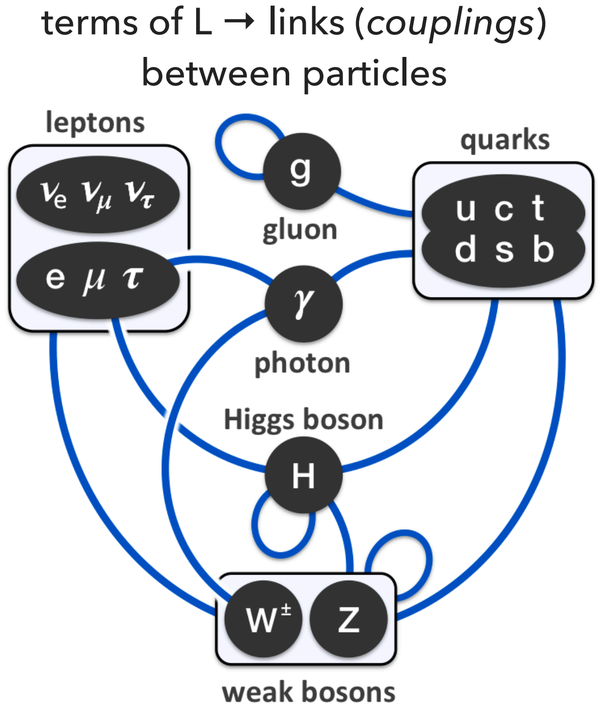
particles

interactions

THE STANDARD MODEL (OR WHY STUDY PARTICLE PHYSICS?)



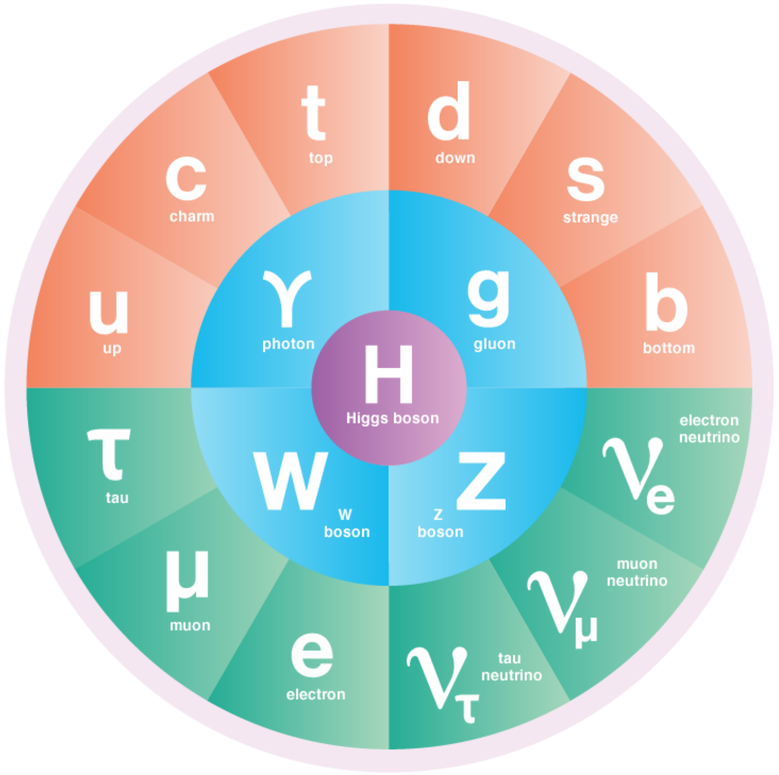
+



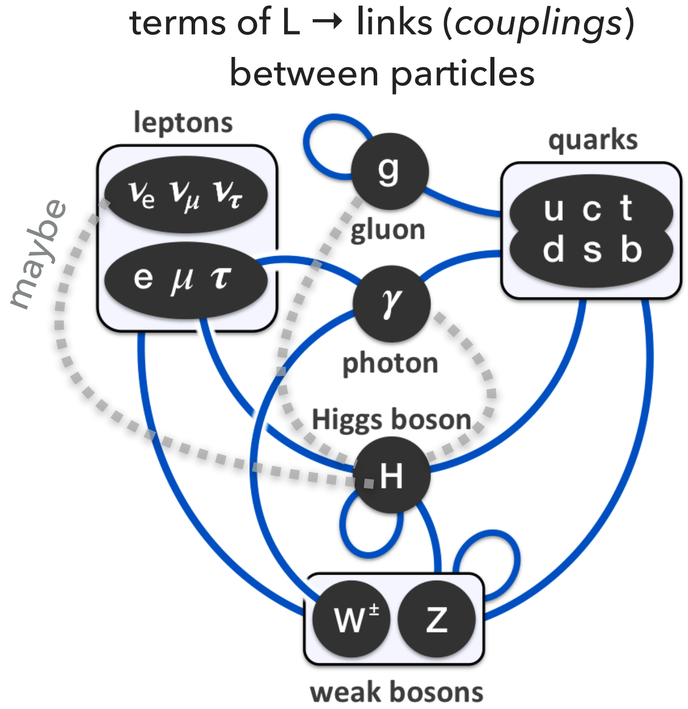
particles *interactions*

- But there has to be more to it! SM does not explain **dark matter**, **neutrino masses**, and the **matter-antimatter asymmetry**…

THE STANDARD MODEL (OR WHY STUDY PARTICLE PHYSICS?)



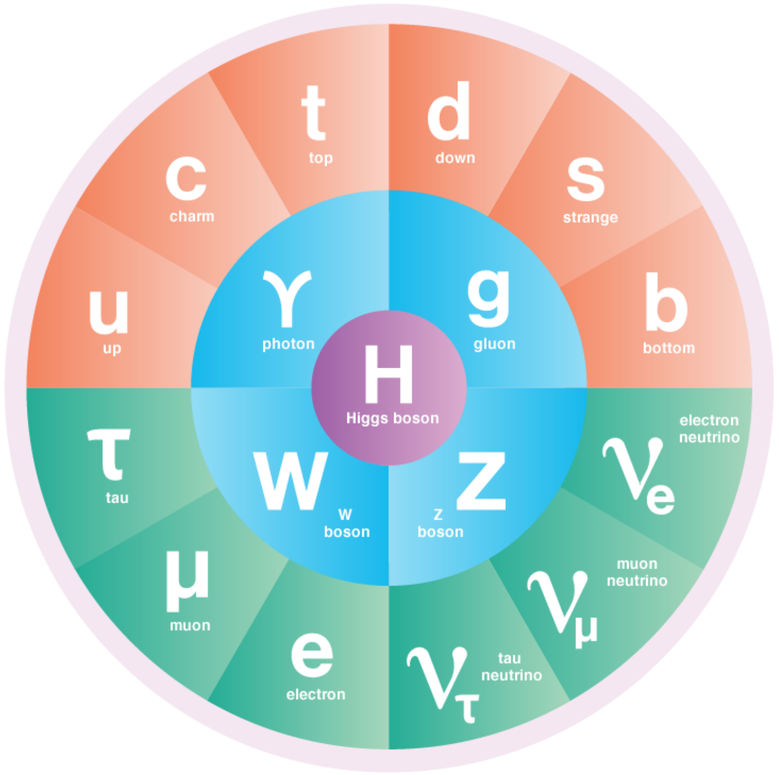
+



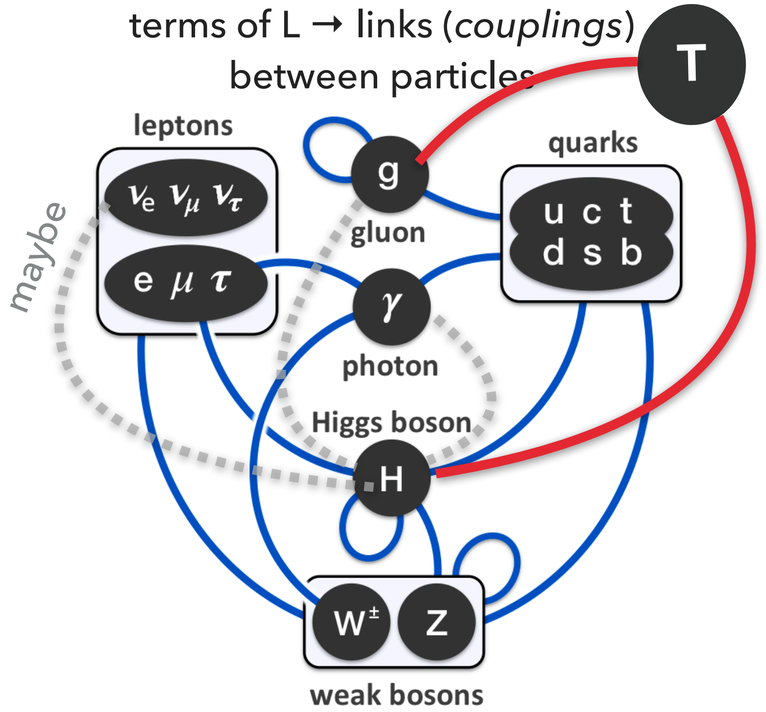
particles *interactions*

- But there has to be more to it! SM does not explain **dark matter**, **neutrino masses**, and the **matter-antimatter asymmetry**…
- Higgs boson is the ***centerpiece:*** all particles interact with it

THE STANDARD MODEL (OR WHY STUDY PARTICLE PHYSICS?)



+



particles *interactions*

- But there has to be more to it! SM does not explain **dark matter**, **neutrino masses**, and the **matter-antimatter asymmetry**…
- Higgs boson is the ***centerpiece:*** all particles interact with it
- May be a link to new particles or interactions

THE LARGE HADRON COLLIDER

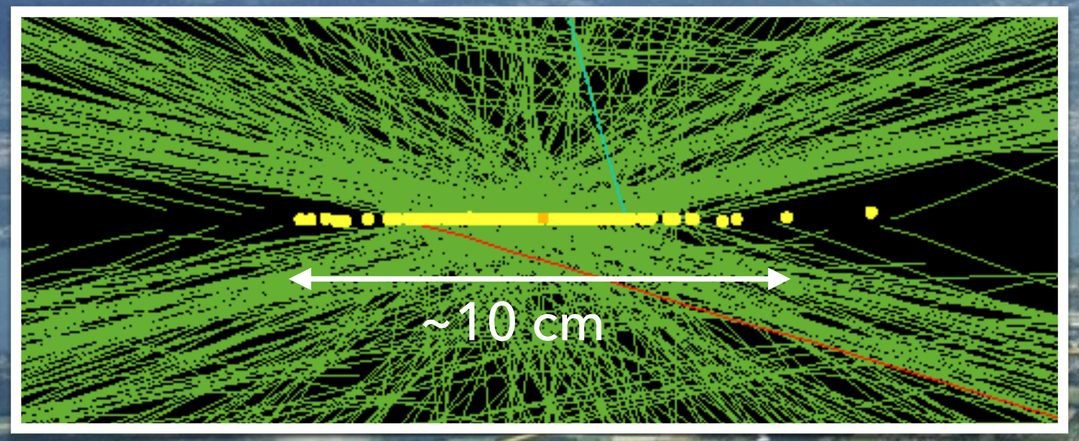
proton-proton collider @ 13 TeV center-of-mass energy

THE LARGE HADRON COLLIDER

proton-proton collider @ 13 TeV center-of-mass energy

4 interaction points

THE LARGE HADRON COLLIDER

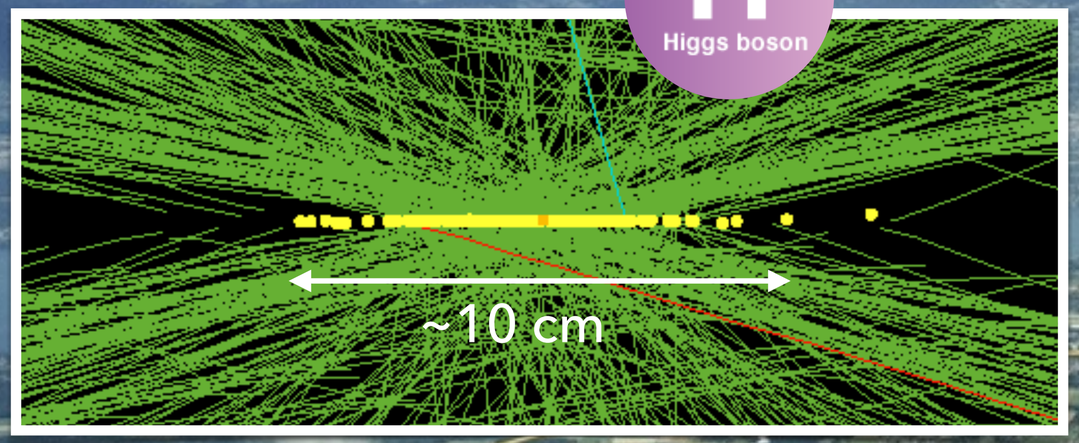


4

LHCb
ATLAS
CERN Meyrin
CERN Prévessin
SPS 7 km
SUISSE
FRANCE
CMS
ALICE
P
LHC 27 km
P

proton-proton collider @ 13 TeV center-of-mass energy
4 interaction points
40 million collisions / second

THE LARGE HADRON COLLIDER



4

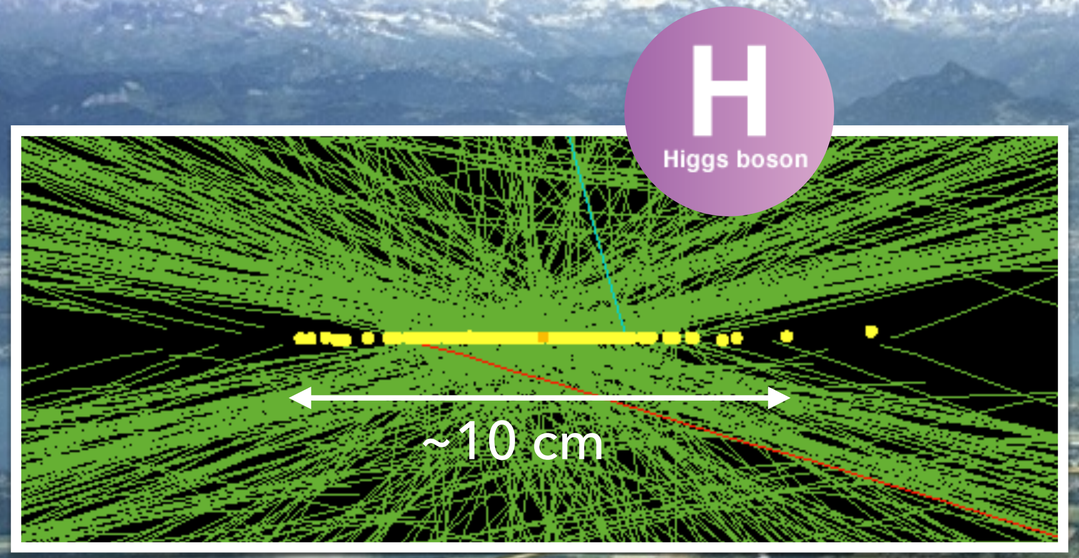
proton-proton collider @ 13 TeV center-of-mass energy

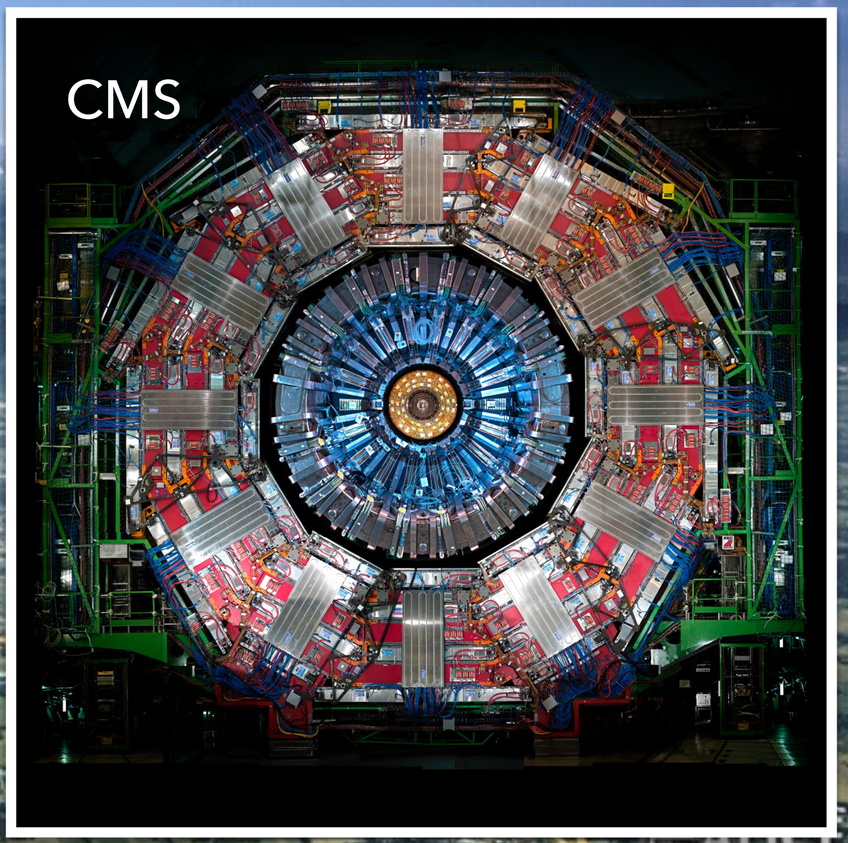
4 interaction points

40 million collisions / second

Higgs boson produced 1/10 billion collisions (every 4 minutes)

THE LARGE HADRON COLLIDER





LHCb

CERN Prévessin

SUISSE

FRANCE

CMS

ALICE

proton-proton collider @ 13 TeV center-of-mass energy

4 interaction points

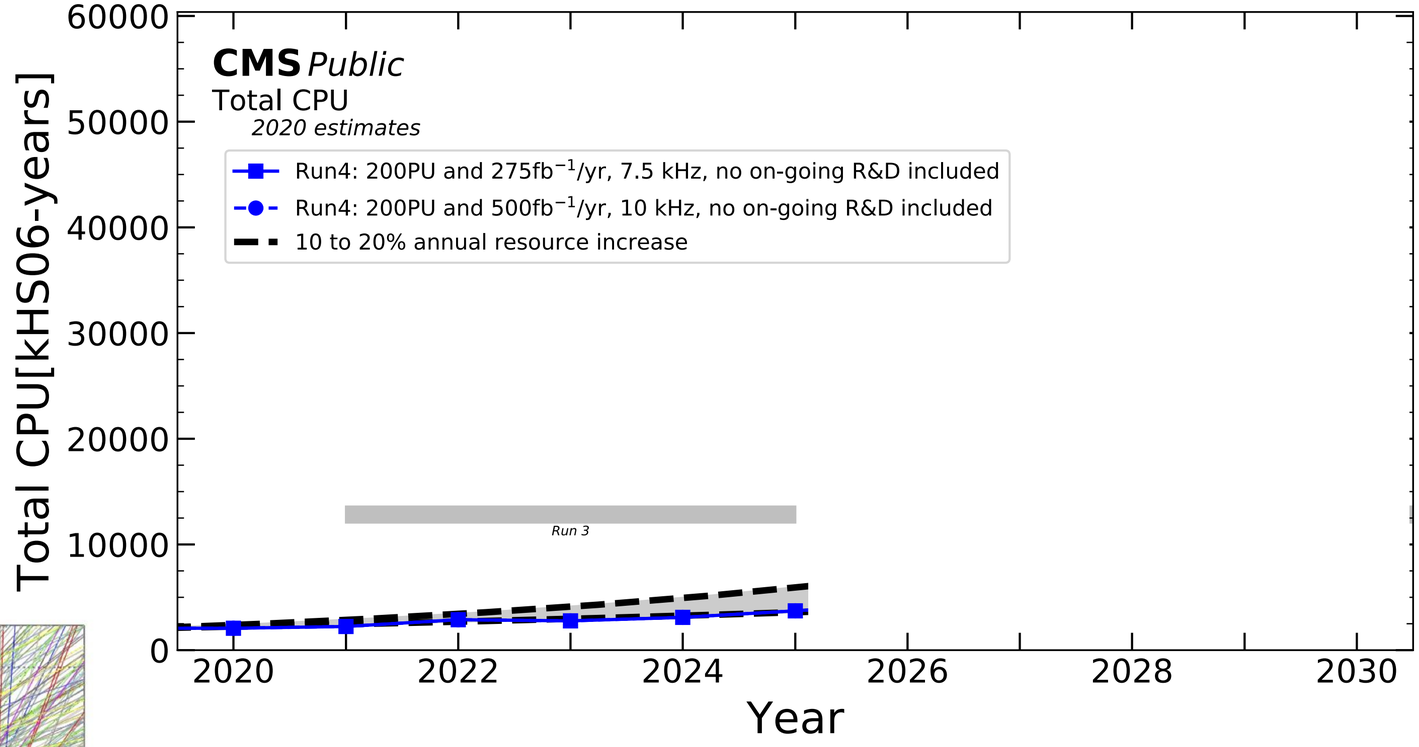
40 million collisions / second

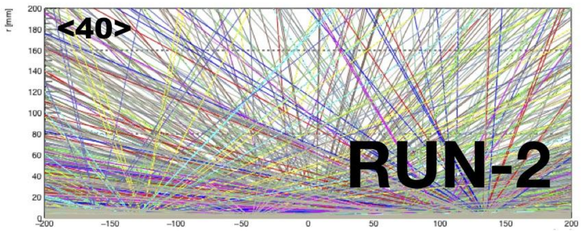
Higgs boson produced 1/10 billion collisions (every 4 minutes)

analyze ~1000 collisions / second

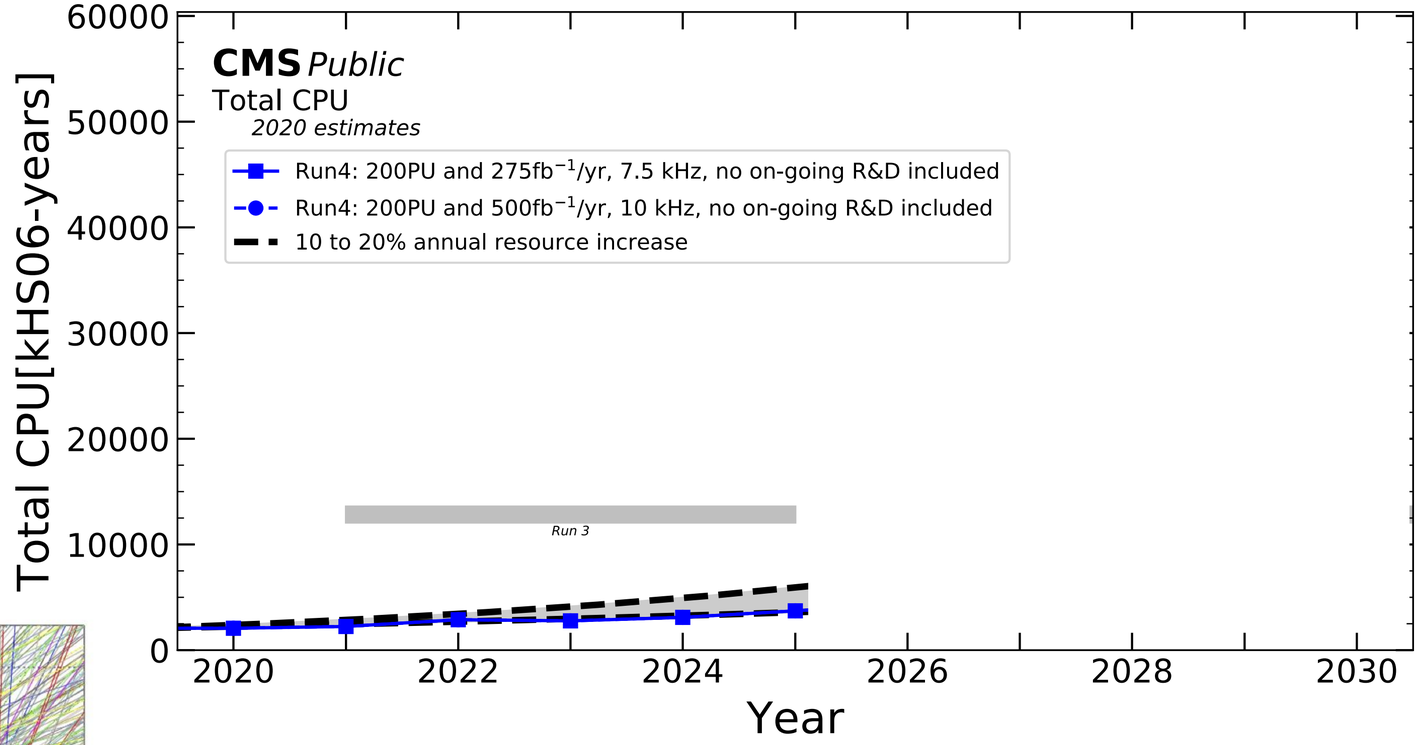
P LHC 27 km P

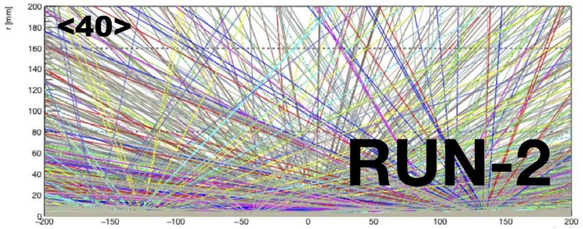
MOTIVATION 1: CPU DEMANDS AT THE UPGRADED LHC



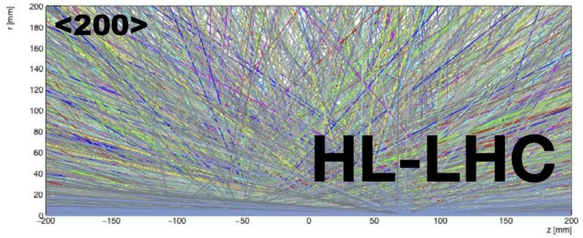


MOTIVATION 1: CPU DEMANDS AT THE UPGRADED LHC



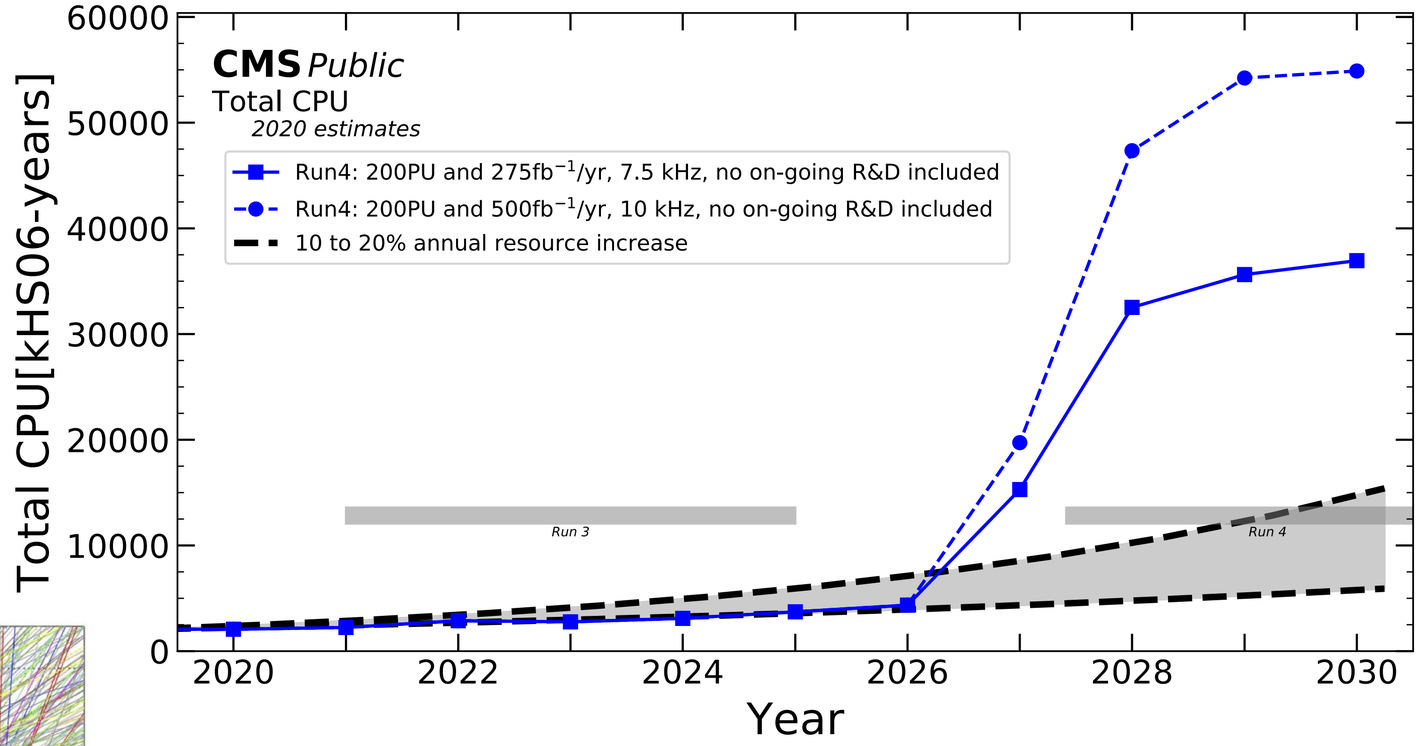


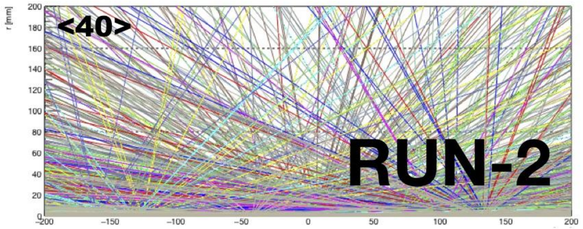
LHC upgrade

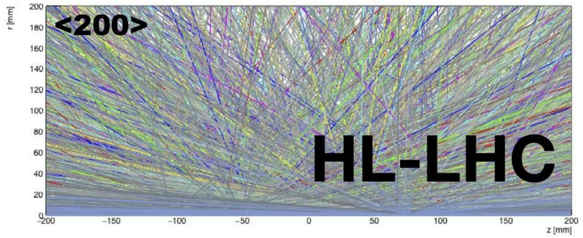


- To analyze more Higgs bosons, need more data, i.e. more **intense beams**

MOTIVATION 1: CPU DEMANDS AT THE UPGRADED LHC

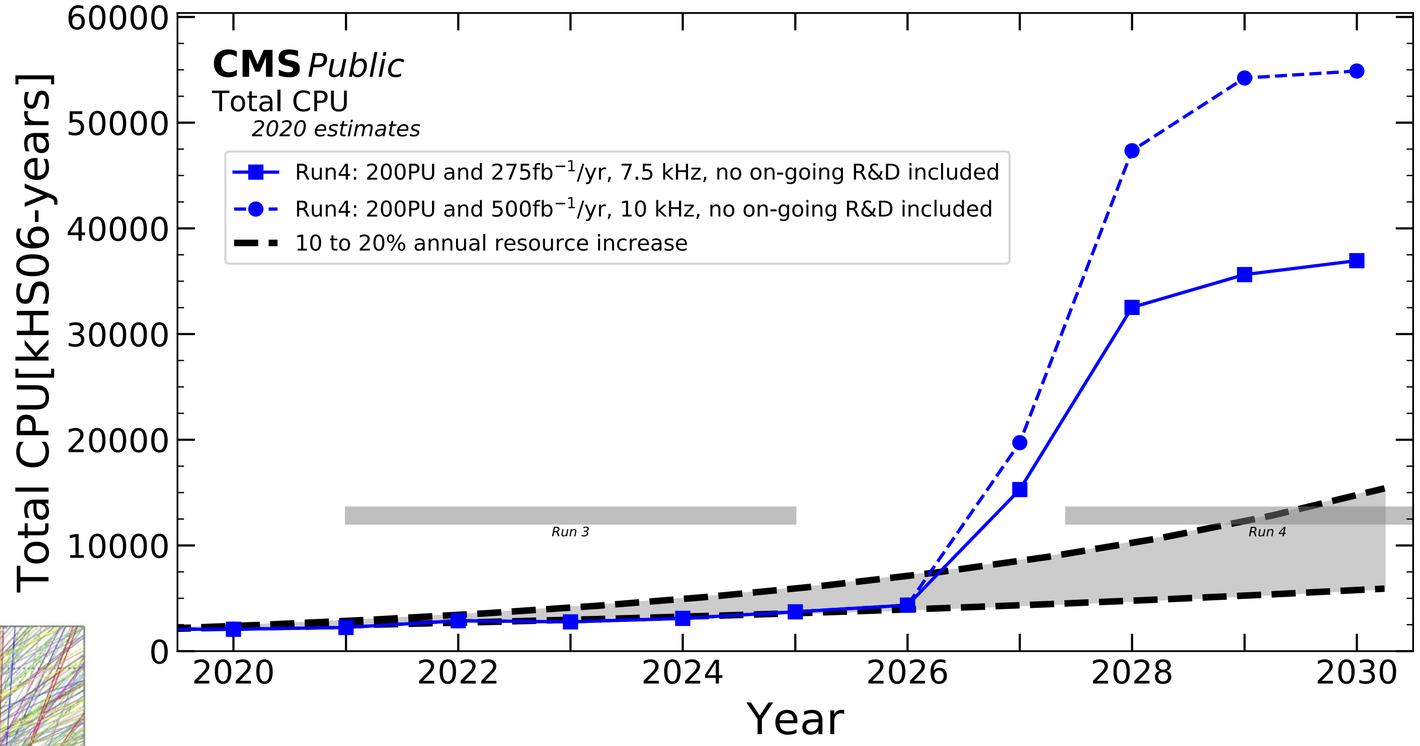


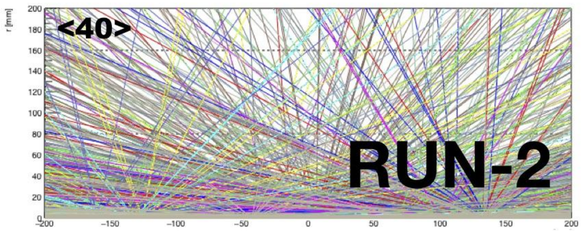


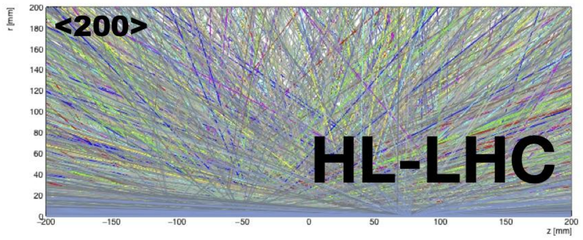


- To analyze more Higgs bosons, need more data, i.e. more **intense beams**
- Computing demands (for event reco.) increase nonlinearly with increasing “pileup”

MOTIVATION 1: CPU DEMANDS AT THE UPGRADED LHC







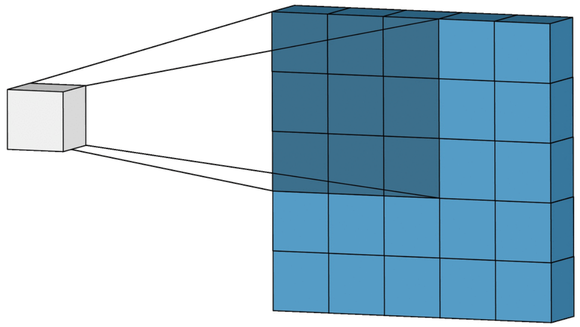
- To analyze more Higgs bosons, need more data, i.e. more **intense beams**
- Computing demands (for event reco.) increase nonlinearly with increasing “pileup”
- Need more processing power (or smarter algorithms) to keep up with data demands

MOTIVATION 2: A NATURAL LANGUAGE

- In deep learning, tailoring algorithms to the structure (and symmetries) of the data has led to groundbreaking performance

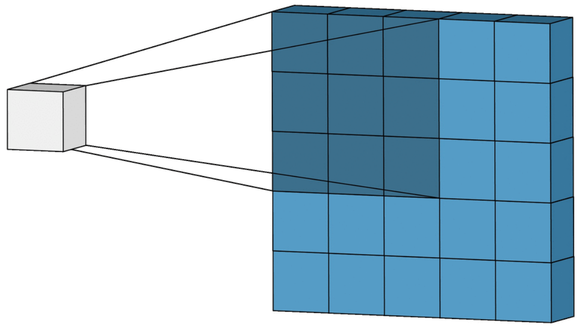
MOTIVATION 2: A NATURAL LANGUAGE

- In deep learning, tailoring algorithms to the structure (and symmetries) of the data has led to groundbreaking performance
  - CNNs for images

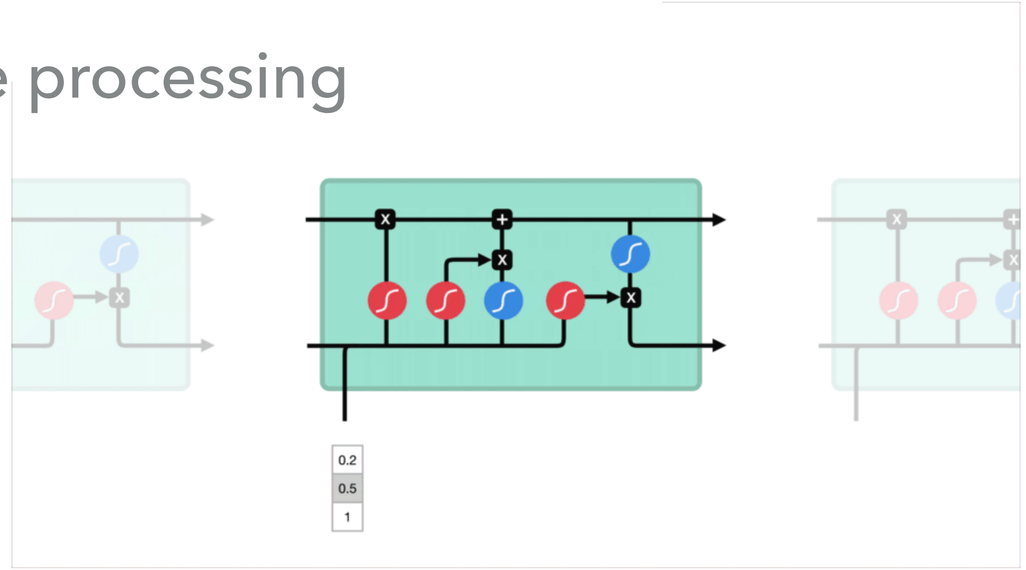


MOTIVATION 2: A NATURAL LANGUAGE

- In deep learning, tailoring algorithms to the structure (and symmetries) of the data has led to groundbreaking performance
  - CNNs for images

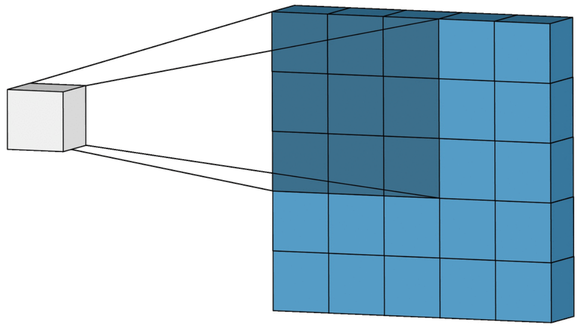


  - RNNs for language processing

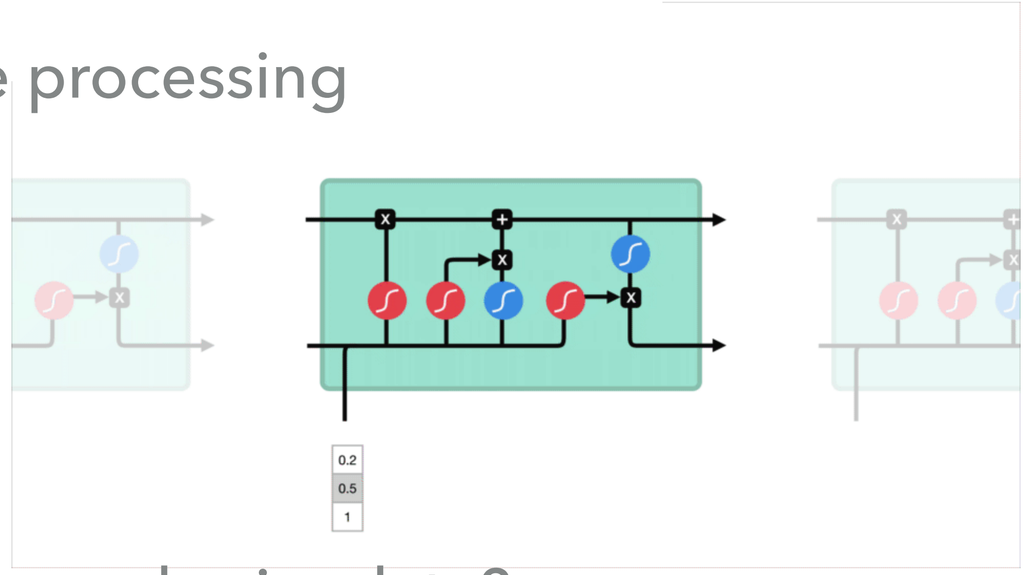


MOTIVATION 2: A NATURAL LANGUAGE

- In deep learning, tailoring algorithms to the structure (and symmetries) of the data has led to groundbreaking performance
  - CNNs for images



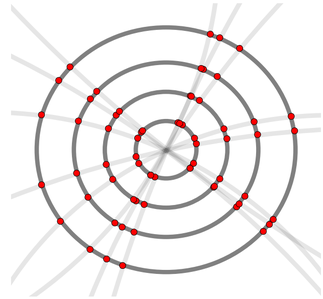
  - RNNs for language processing



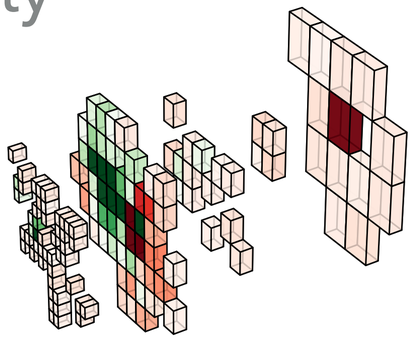
- What about high energy physics data?

HIGH ENERGY PHYSICS DATA

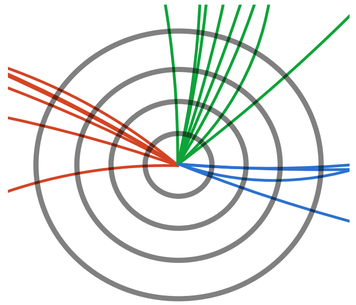
- Properties
  - Measurements distributed in space (and time) irregularly



  - Sparse (most detector channels are empty), but pockets of density



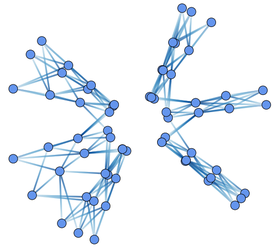
  - Complex interdependencies between measurements
  - Physics “objects” composed of multiple measurements



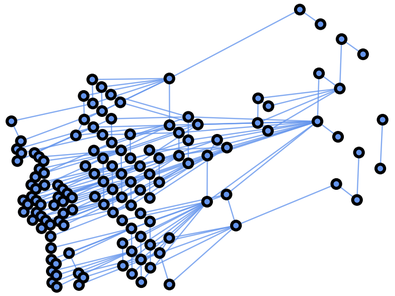
  - Inherent symmetries (Lorentz boosts, rotational)*

HIGH ENERGY PHYSICS DATA

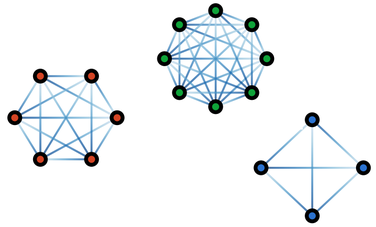
- Properties
  - Measurements distributed in space (and time) irregularly



  - Sparse (most detector channels are empty), but pockets of density



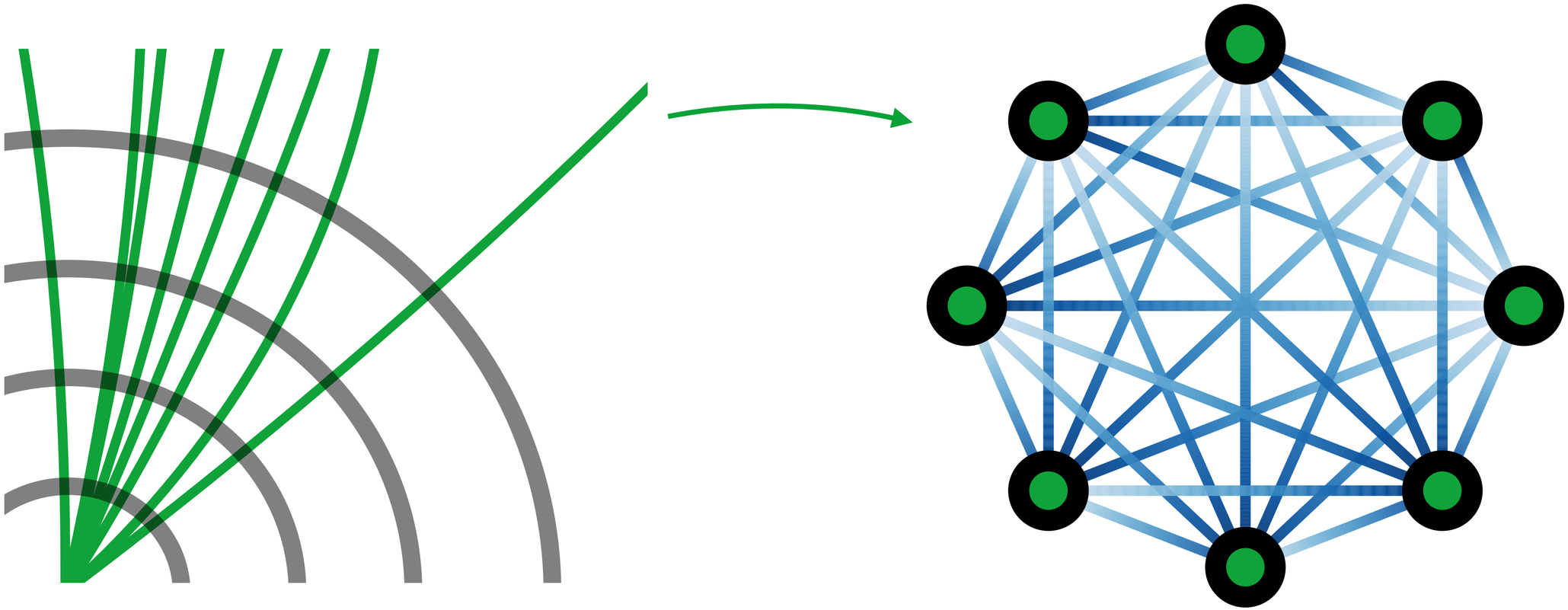
  - Complex interdependencies between measurements
  - Physics “objects” composed of multiple measurements



  - Inherent symmetries (Lorentz boosts, rotational)*

- Graph (or point cloud) embedding of the data can handle these properties

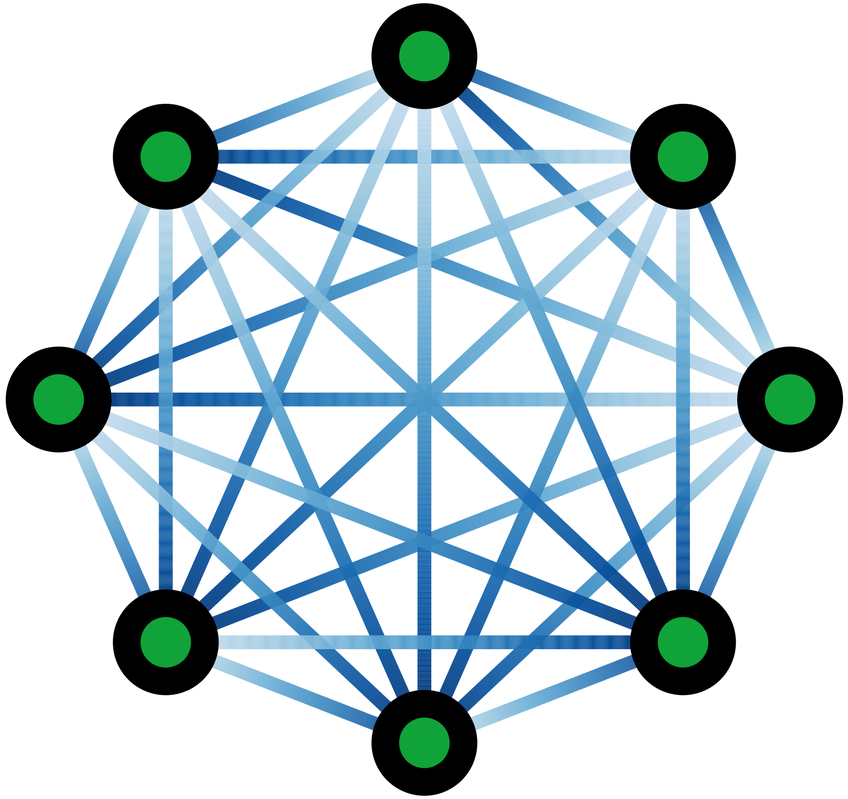
NODE, EDGE, GRAPH FEATURES IN HEP (E.G. JET)



NODE, EDGE, GRAPH FEATURES IN HEP (E.G. JET)

- Node features: particle 4-momentum

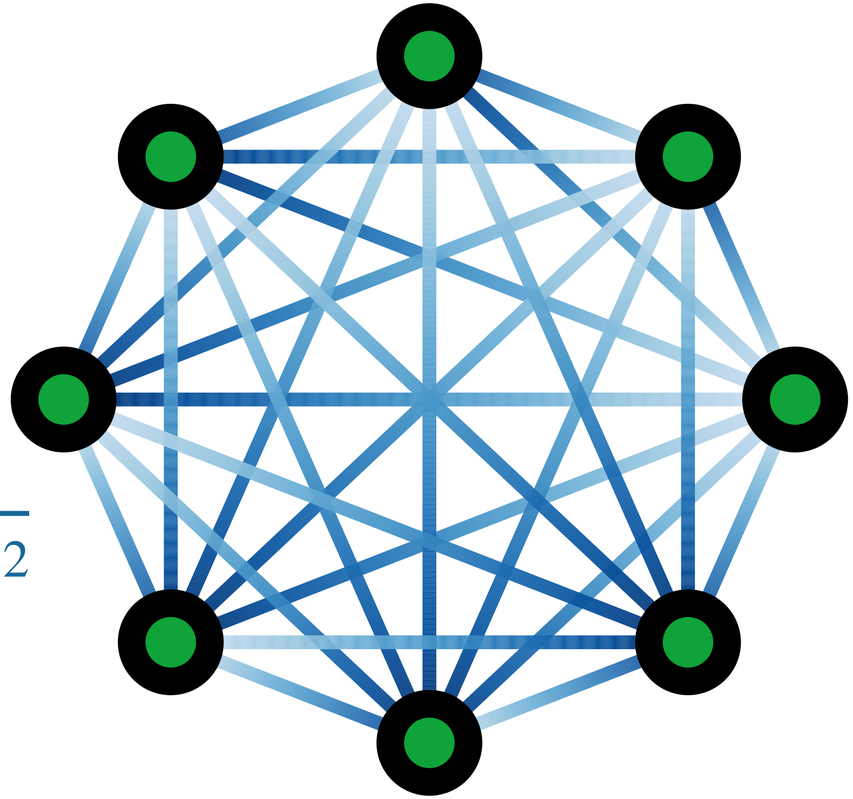
$$p = [E, p_x, p_y, p_z] \equiv [p_\mathrm{T}, \eta, \phi, m]$$



NODE, EDGE, GRAPH FEATURES IN HEP (E.G. JET)

- Node features: particle 4-momentum

$$p = [E, p_x, p_y, p_z] \equiv [p_\mathrm{T}, \eta, \phi, m]$$



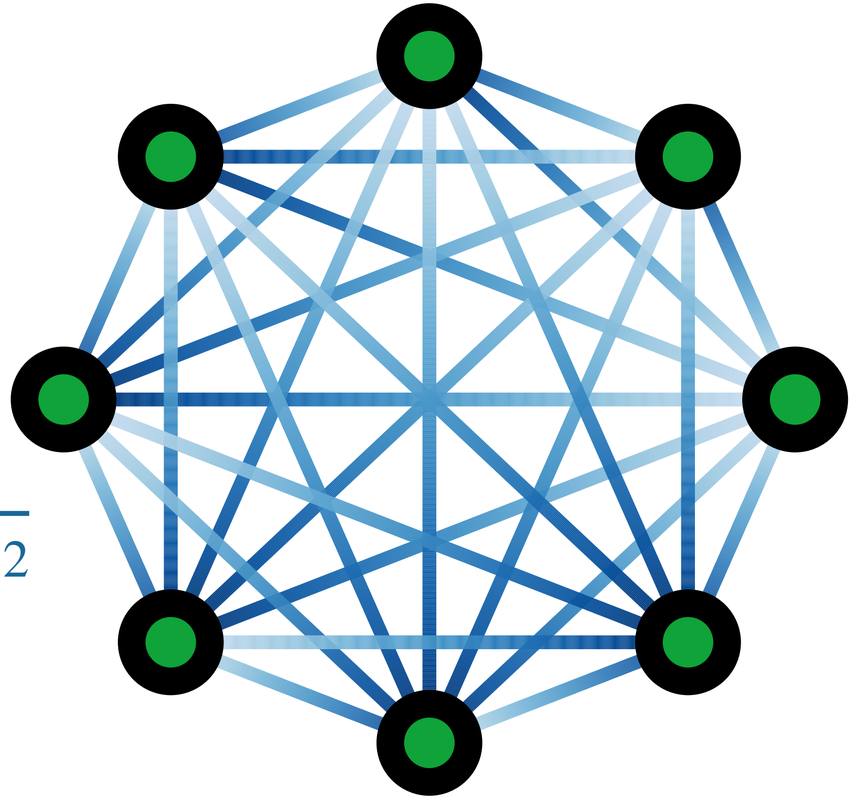
- Edge features: pseudoangular distance between particles

$$\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2}$$

NODE, EDGE, GRAPH FEATURES IN HEP (E.G. JET)

- Node features: particle 4-momentum

$$p = [E, p_x, p_y, p_z] \equiv [p_\mathrm{T}, \eta, \phi, m]$$



- Edge features: pseudoangular distance between particles

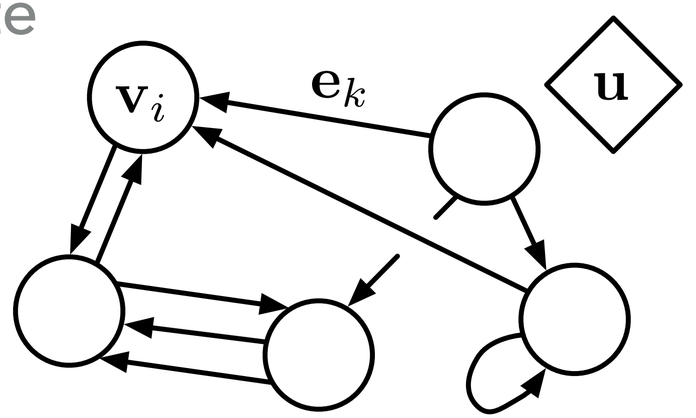
$$\Delta R = \sqrt{\Delta\eta^2 + \Delta\phi^2}$$

- Graph (global) features: jet mass

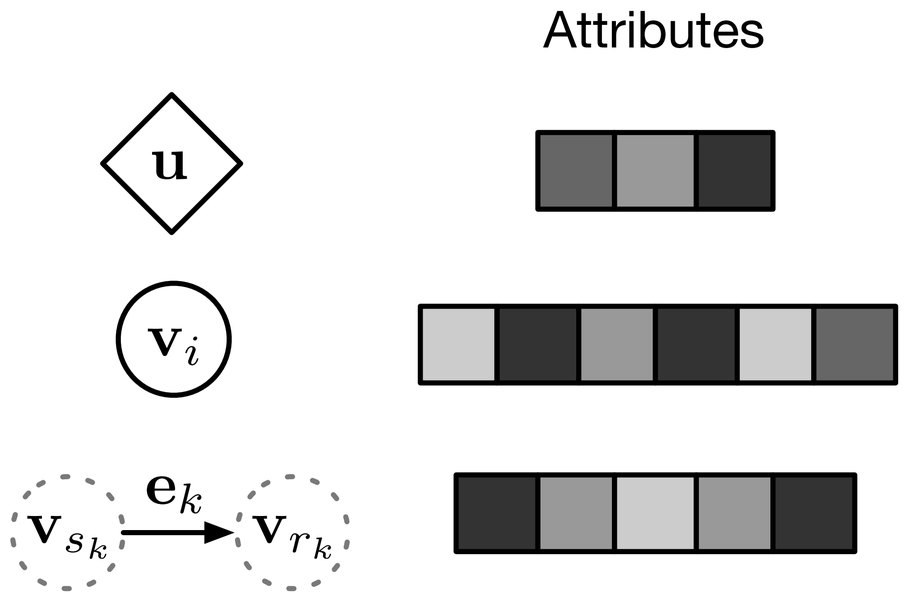
$$m = \sqrt{\sum_{i \in \mathrm{jet}} E_i^2 - p_{x,i}^2 - p_{y,i}^2 - p_{z,i}^2}$$

FORMALIZING A GRAPH

- Directed, attributed multigraph with a global attribute
  - Triplet of global features, node features, and edge features: $(\mathbf{u}, V, E)$ with "receivers" $\boldsymbol{r}$ and "senders" $\boldsymbol{s}$ (graph connectivity)

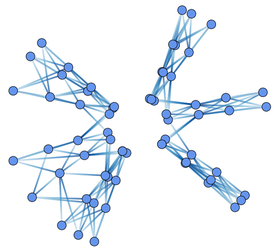


- Graph $G = (\mathbf{u}, V, E)$
  - Global attribute $\mathbf{u}$
  - Node attributes $V = \{\mathbf{v}_i\}$
  - Edge attributes, receivers, senders $E = \{\mathbf{e}_k, r_k, s_k\}$

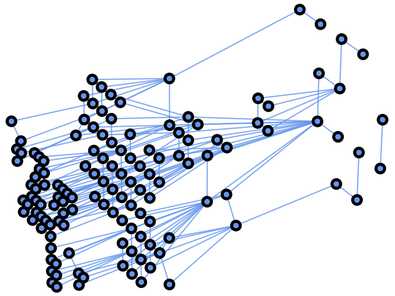


GRAPH TASKS

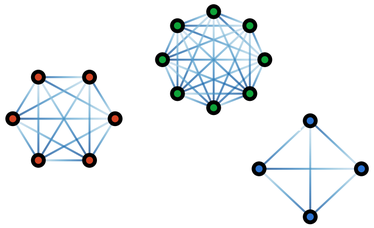
- Edge-level tasks
  - Identify track segments
  - Estimate track parameters
- Node-level tasks
  - Correct cluster energies
  - Identify "pileup" particles



- Graph-level tasks
  - **Jet tagging**
  - Estimate shower energy
  - Signal-to-background event discrimination

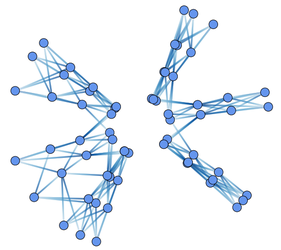


- Node partitioning/pooling tasks
  - **Particle-flow reconstruction**

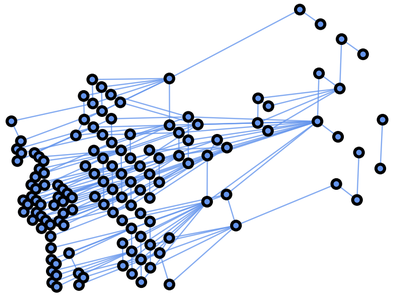


GRAPH TASKS

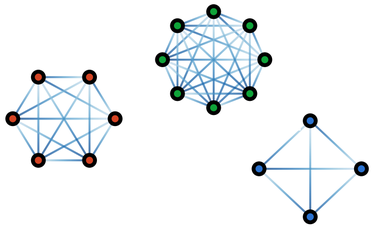
- Edge-level tasks
  - Identify track segments
  - Estimate track parameters
- Node-level tasks
  - Correct cluster energies
  - Identify "pileup" particles



- Graph-level tasks
  - **Jet tagging**
  - Estimate shower energy
  - Signal-to-background event discrimination



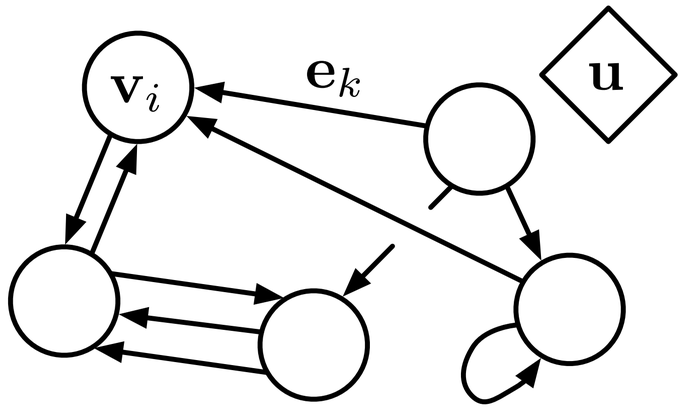
- Node partitioning/pooling tasks
  - **Particle-flow reconstruction**



⇒ GRAPH-TO-GRAPH MAPPINGS

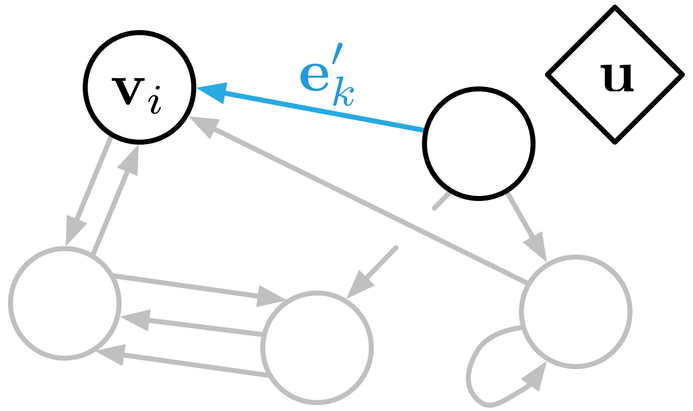
GRAPH NETWORKS*

- One framework for describing GNNs is "Graph Networks" [arXiv:1806.01261]
- GN is a graph-to-graph function approximator
- Inference divided into three parts: edge block, node block, global block



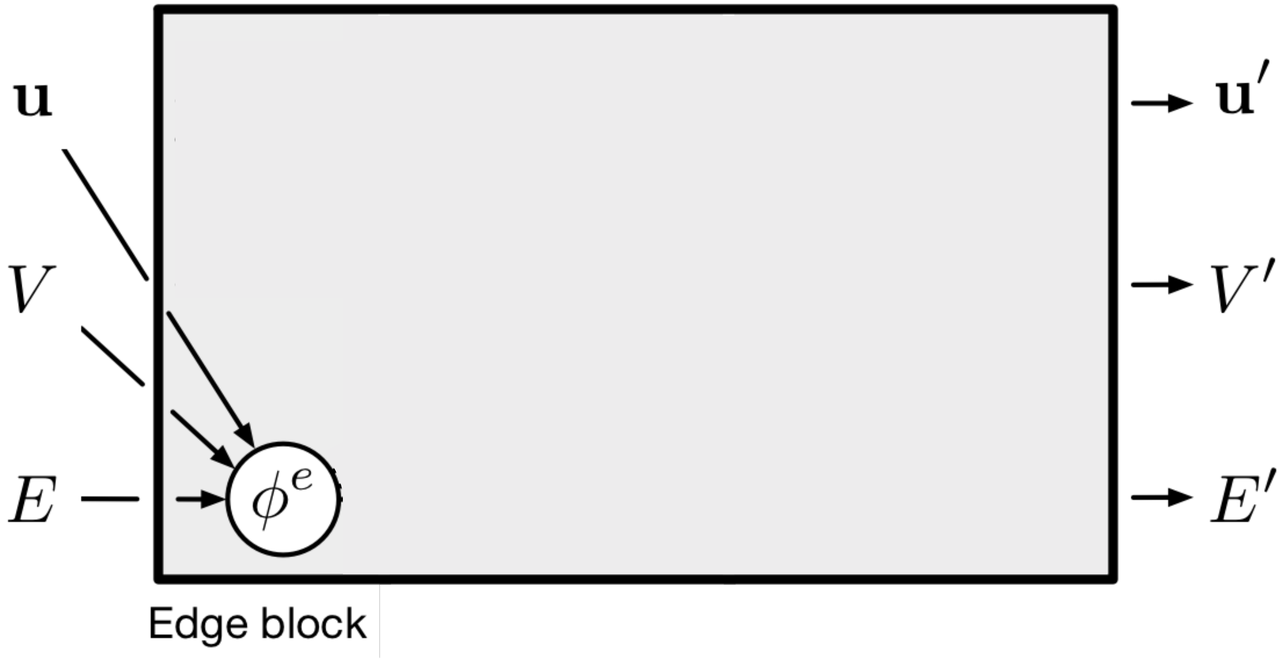
GRAPH NETWORKS*

- One framework for describing GNNs is "Graph Networks" [arXiv:1806.01261]
- GN is a graph-to-graph function approximator
- Inference divided into three parts: edge block, node block, global block



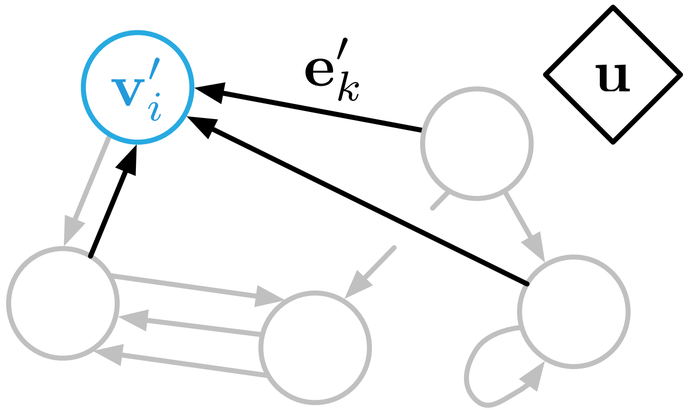
$\mathbf{e}'_k$: message computed for edge k connecting nodes r_k, s_k

$$\mathbf{e}'_k = \phi^e(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u})$$



GRAPH NETWORKS*

- One framework for describing GNNs is "Graph Networks" [arXiv:1806.01261]
- GN is a graph-to-graph function approximator
- Inference divided into three parts: edge block, node block, global block

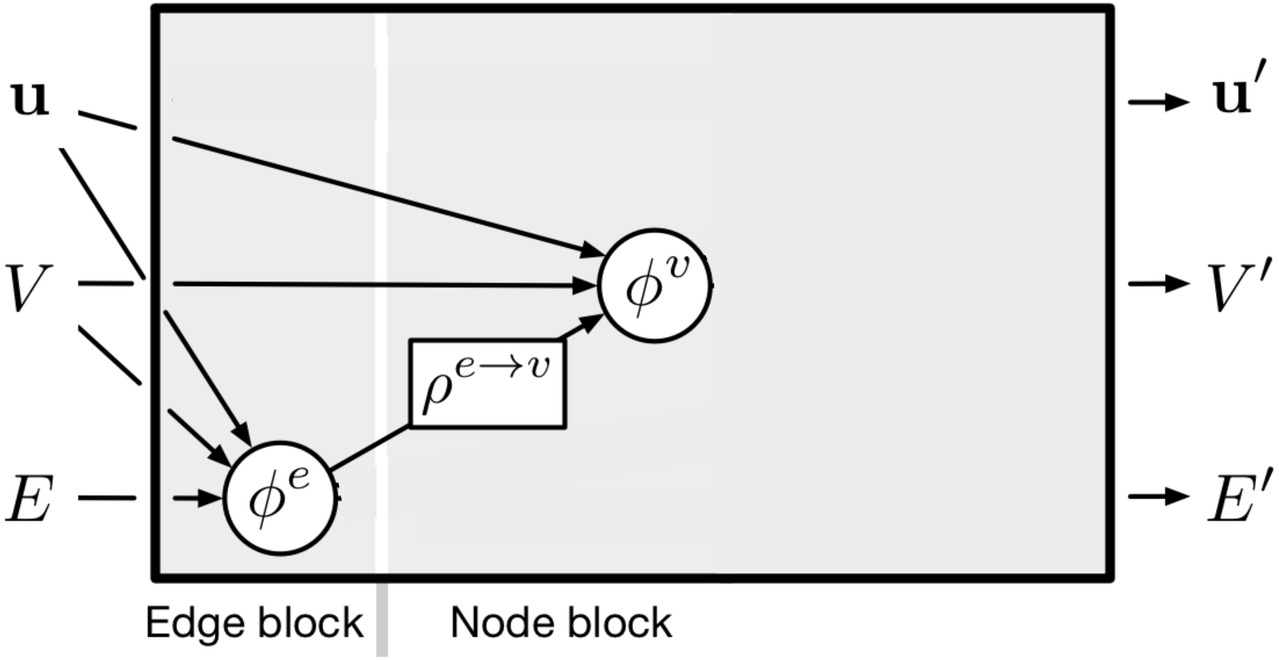


$\mathbf{e}'_k$: message computed for edge k connecting nodes r_k, s_k

$\mathbf{v}'_i$: node feature update based on aggregated messages and previous features

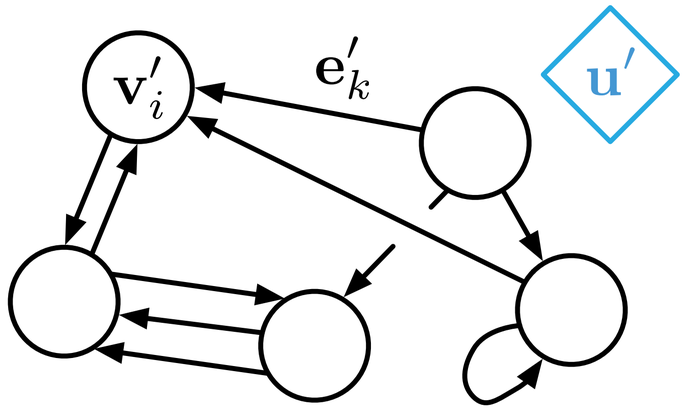
$$\mathbf{e}'_k = \phi^e(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}) \qquad \bar{\mathbf{e}}'_i = \rho^{e\to v}(E'_i)$$

$$\mathbf{v}'_i = \phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right)$$



GRAPH NETWORKS*

- One framework for describing GNNs is "Graph Networks" [arXiv:1806.01261]
- GN is a graph-to-graph function approximator
- Inference divided into three parts: edge block, node block, global block



$\mathbf{e}'_k$: message computed for edge k connecting nodes r_k, s_k

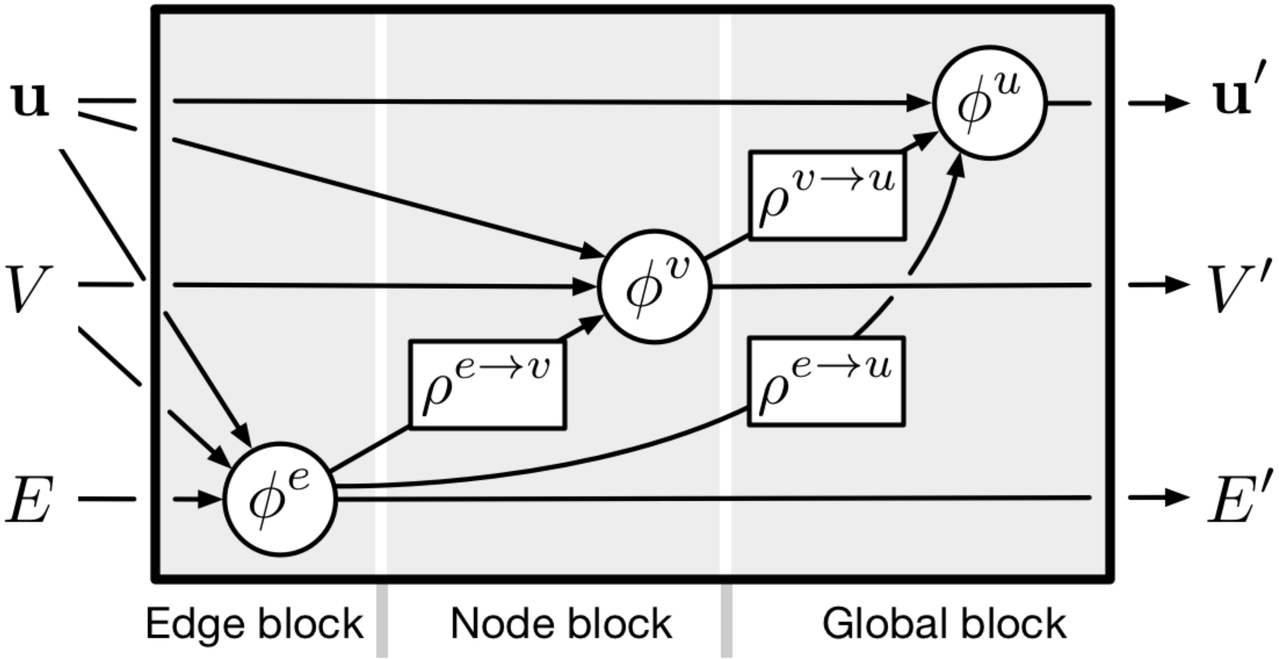
$\mathbf{v}'_i$: node feature update based on aggregated messages and previous features

$\mathbf{u}'$: global feature update based on aggregated, updated node and edge features

$$\mathbf{e}'_k = \phi^e(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}, \mathbf{u}) \qquad \bar{\mathbf{e}}'_i = \rho^{e\to v}(E'_i)$$

$$\mathbf{v}'_i = \phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right) \qquad \bar{\mathbf{e}}' = \rho^{e\to u}(E')$$

$$\mathbf{u}' = \phi^u(\bar{\mathbf{e}}', \bar{\mathbf{v}}', \boldsymbol{u}) \qquad \bar{\mathbf{v}}' = \rho^{v\to u}(V')$$



HIGGS JET TAGGING

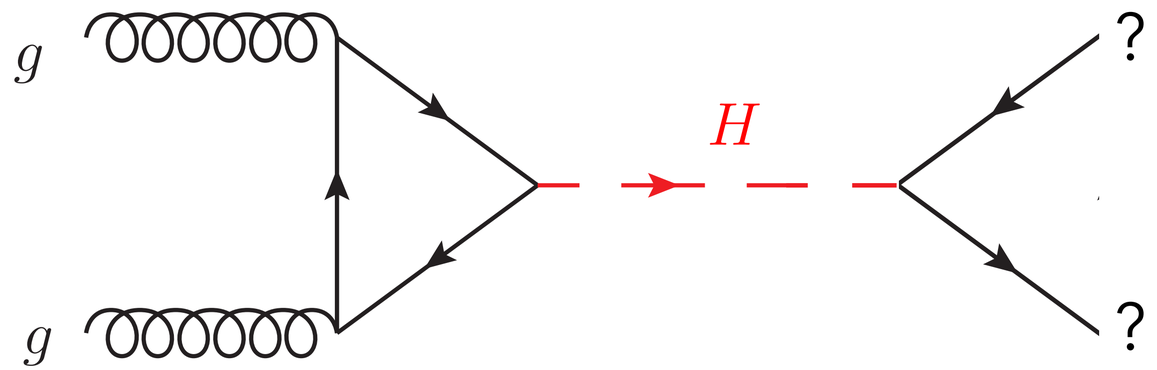
UCSD

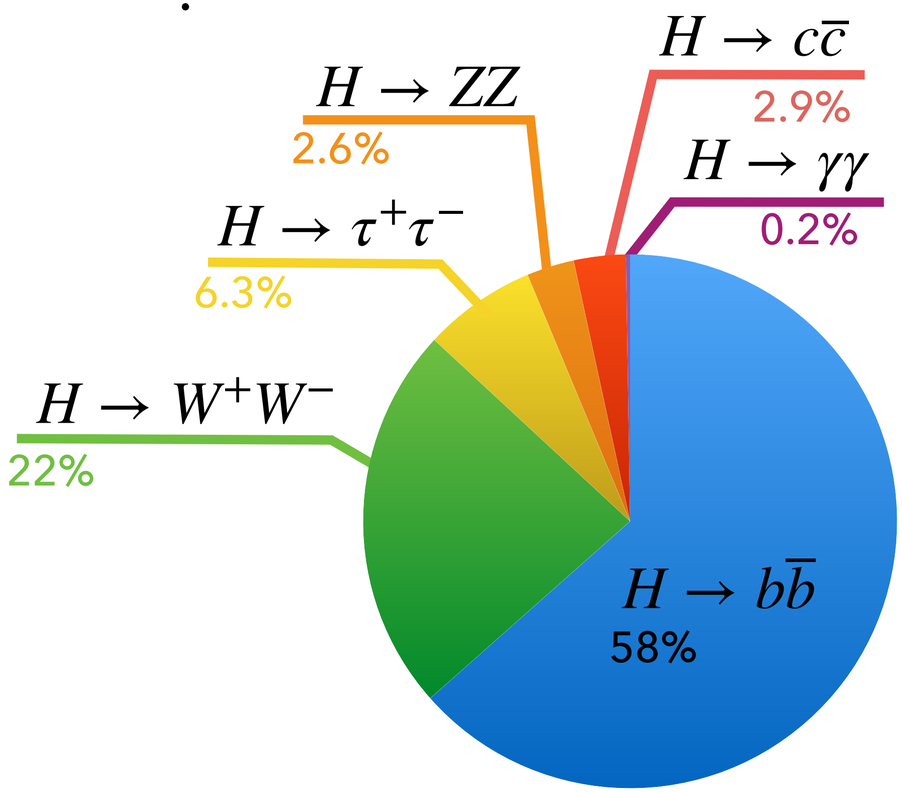
HOW DO WE DETECT HIGGS BOSONS?

produce

detect

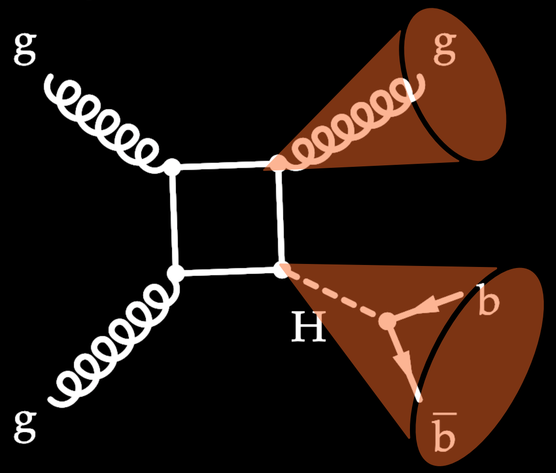
(87%)





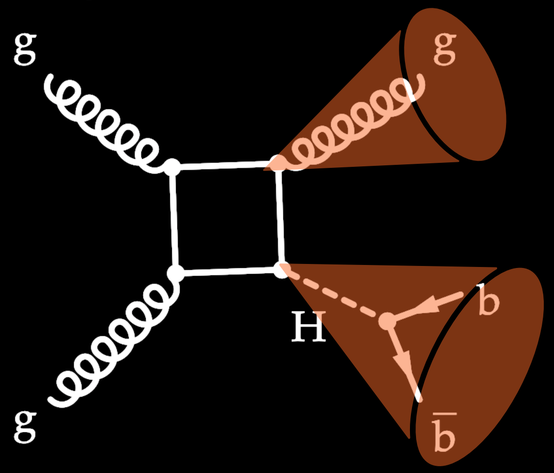
- H→γγ and H→ZZ have small rate but are very "clean"
- H→bb is large, but more difficult due to large backgrounds

Signal:



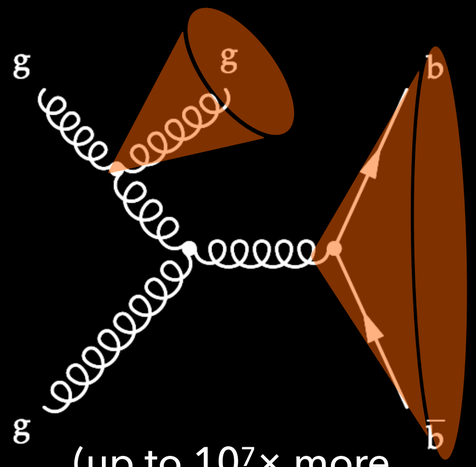
?

Signal:



?

Background:



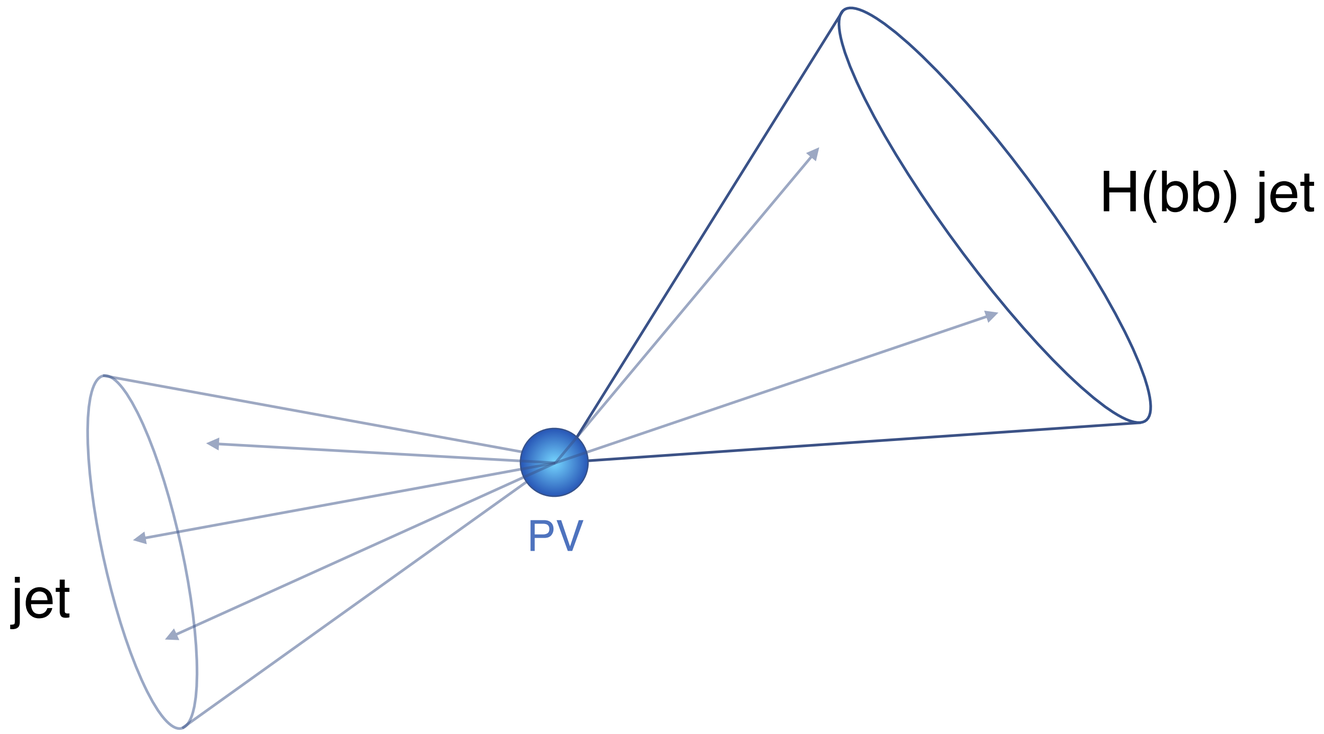
(up to 10^7× more frequent)

?

BASICS OF HIGGS (DOUBLE-B) TAGGING

b hadrons have long lifetimes:
travel O(mm) before decay!

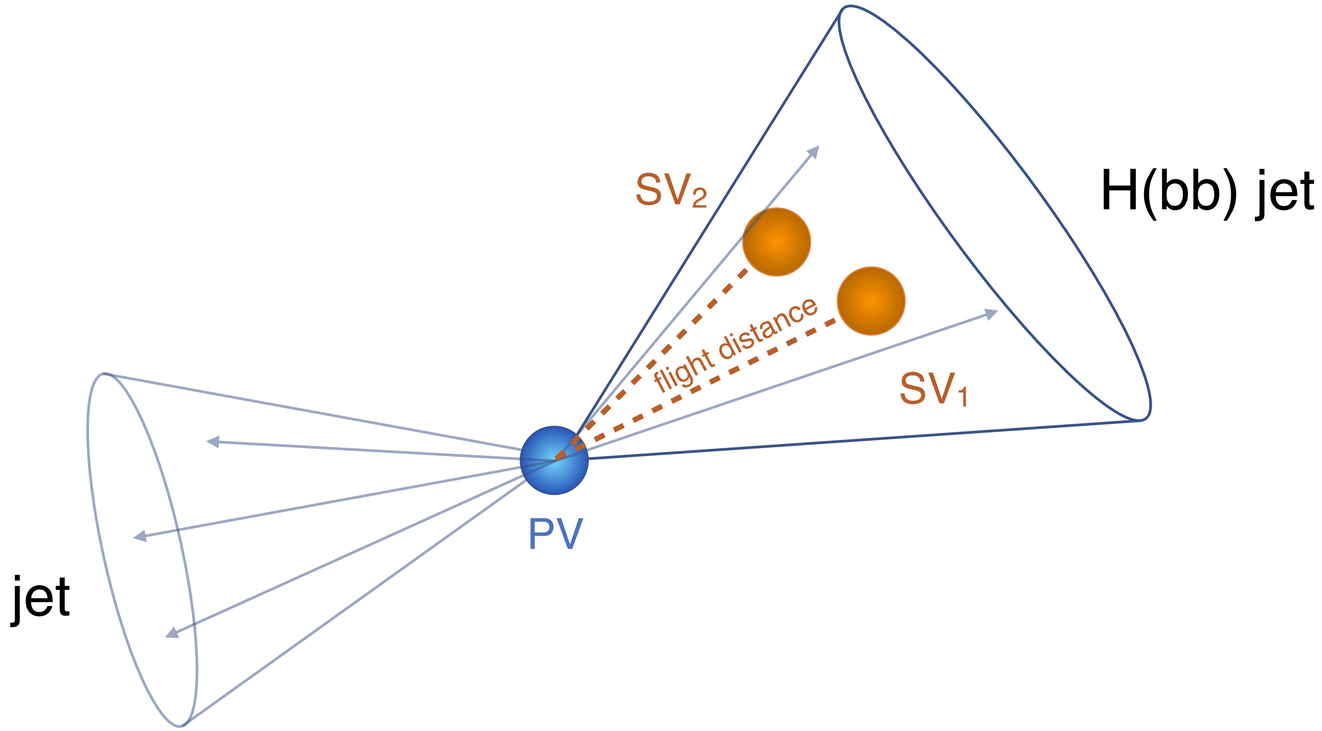
- Handles:



BASICS OF HIGGS (DOUBLE-B) TAGGING

b hadrons have long lifetimes:
travel O(mm) before decay!

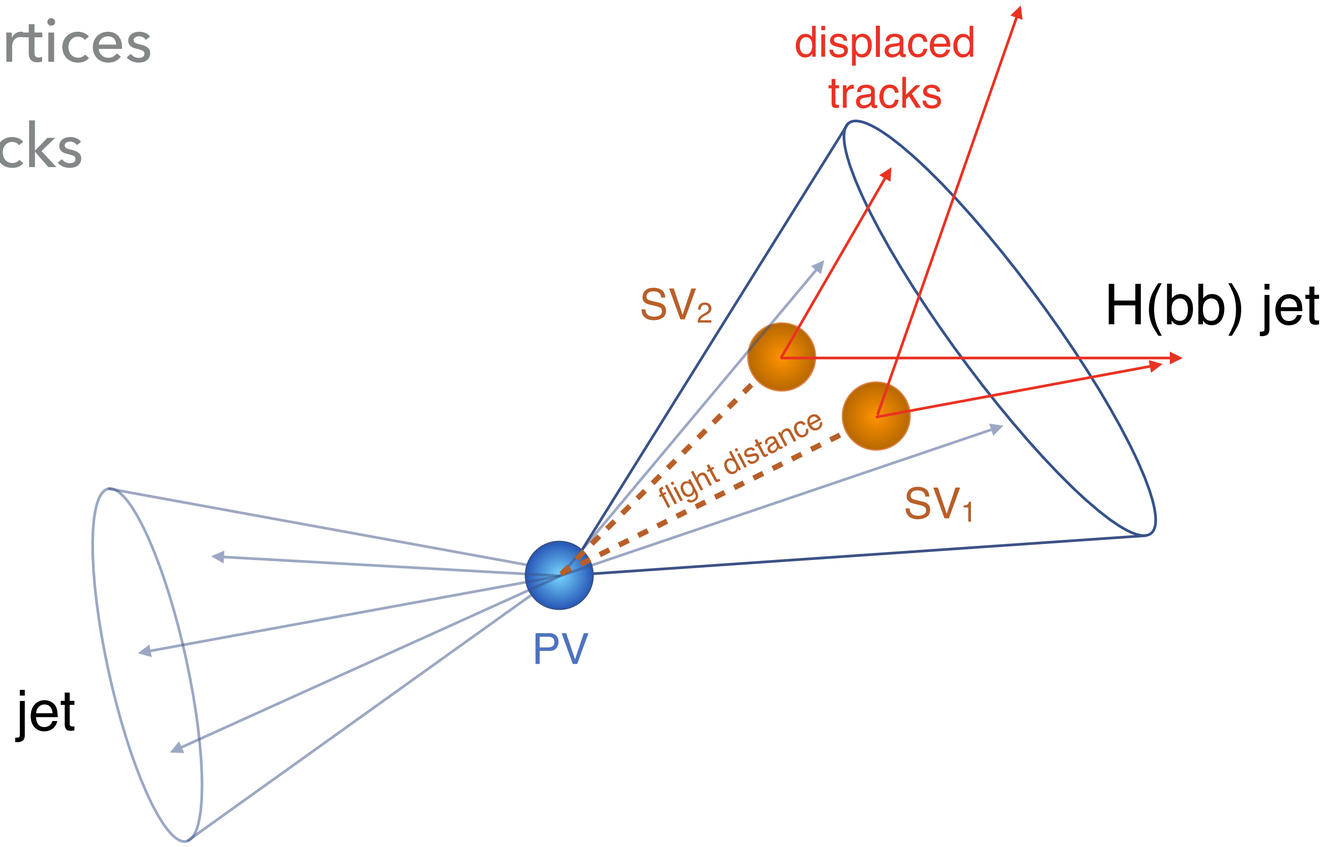
- Handles:
  - secondary vertices



BASICS OF HIGGS (DOUBLE-B) TAGGING

b hadrons have long lifetimes:
travel O(mm) before decay!

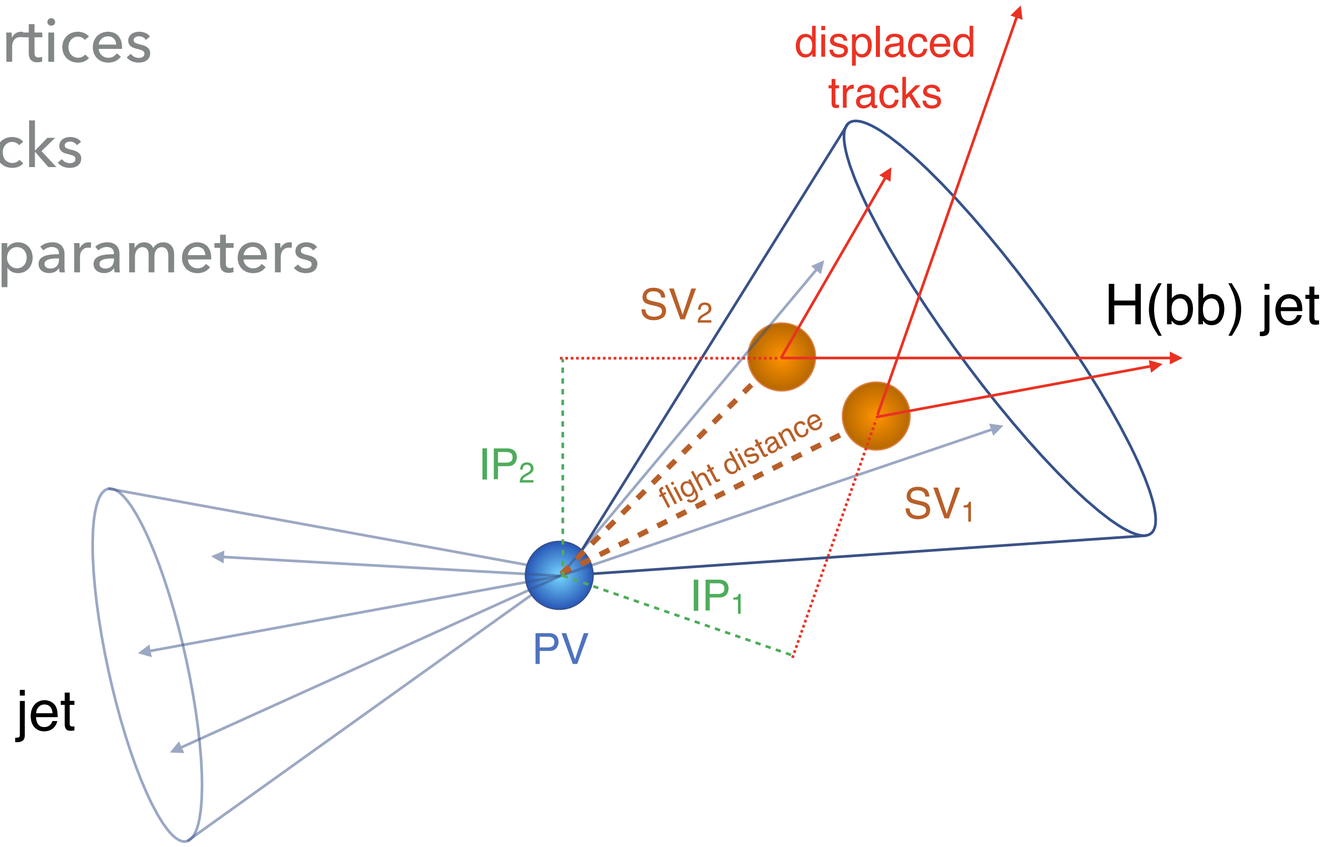
- Handles:
  - secondary vertices
  - displaced tracks



BASICS OF HIGGS (DOUBLE-B) TAGGING

b hadrons have long lifetimes:
travel O(mm) before decay!

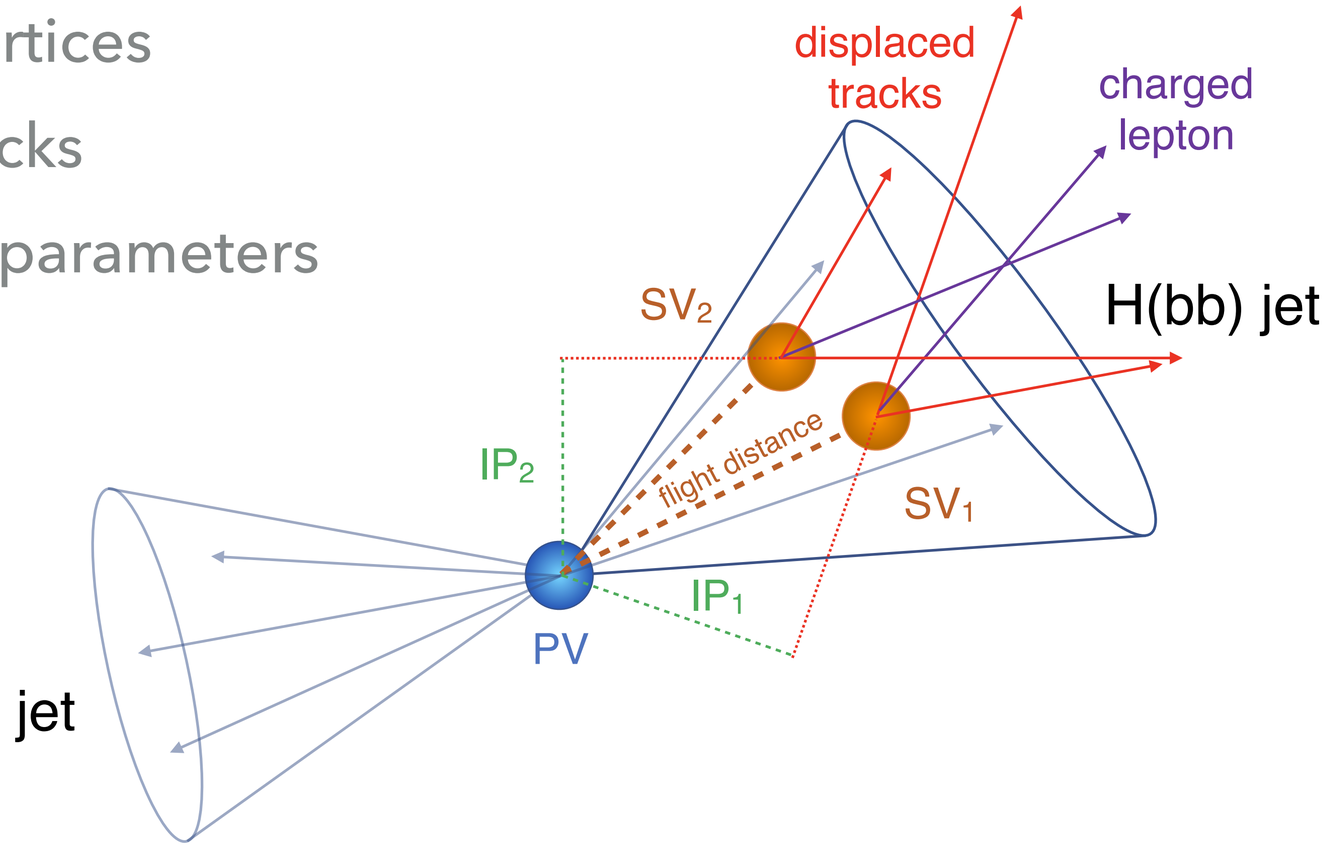
- Handles:
  - secondary vertices
  - displaced tracks
  - large impact parameters



BASICS OF HIGGS (DOUBLE-B) TAGGING

b hadrons have long lifetimes:
travel O(mm) before decay!

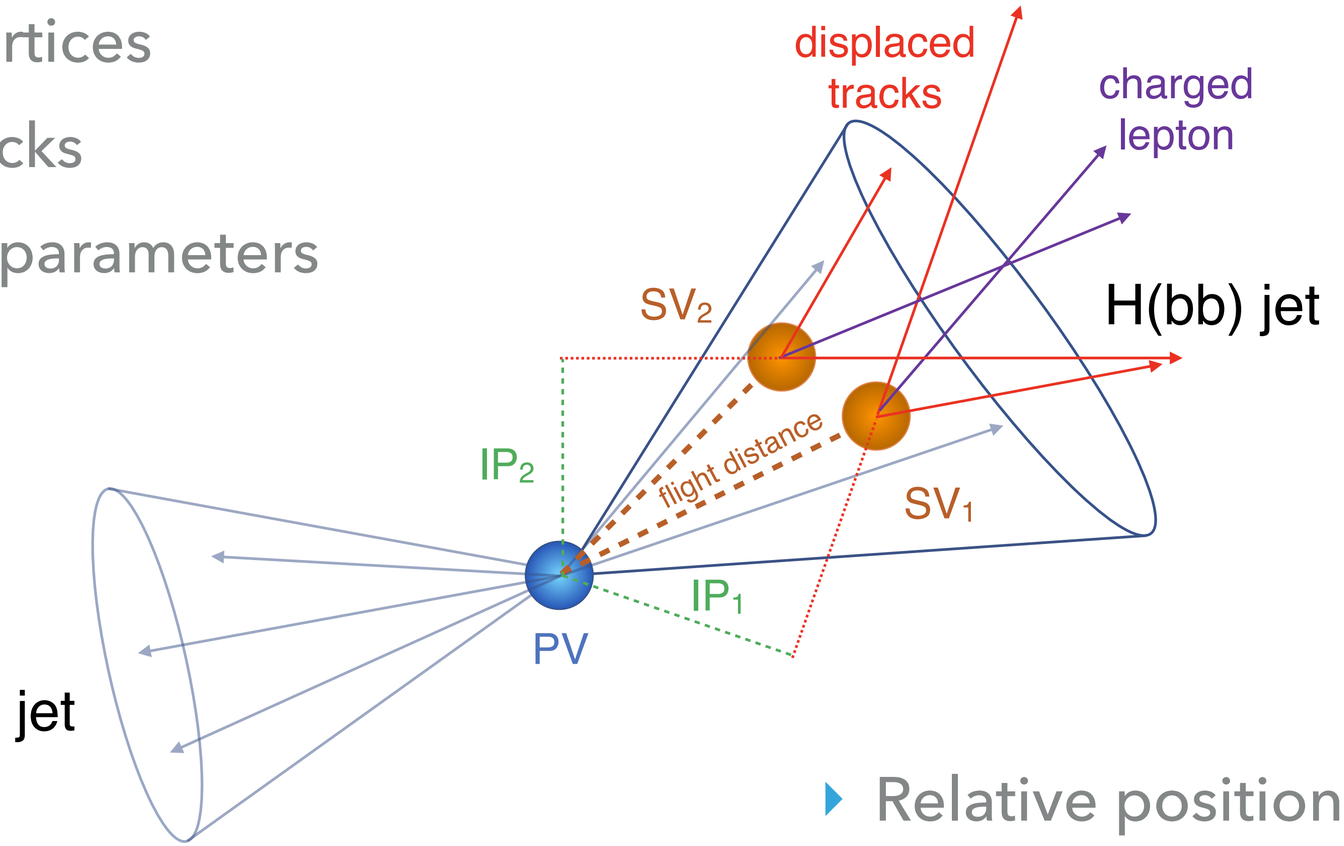
- Handles:
  - secondary vertices
  - displaced tracks
  - large impact parameters
  - soft leptons



BASICS OF HIGGS (DOUBLE-B) TAGGING

b hadrons have long lifetimes:
travel O(mm) before decay!

- Handles:
  - secondary vertices
  - displaced tracks
  - large impact parameters
  - soft leptons

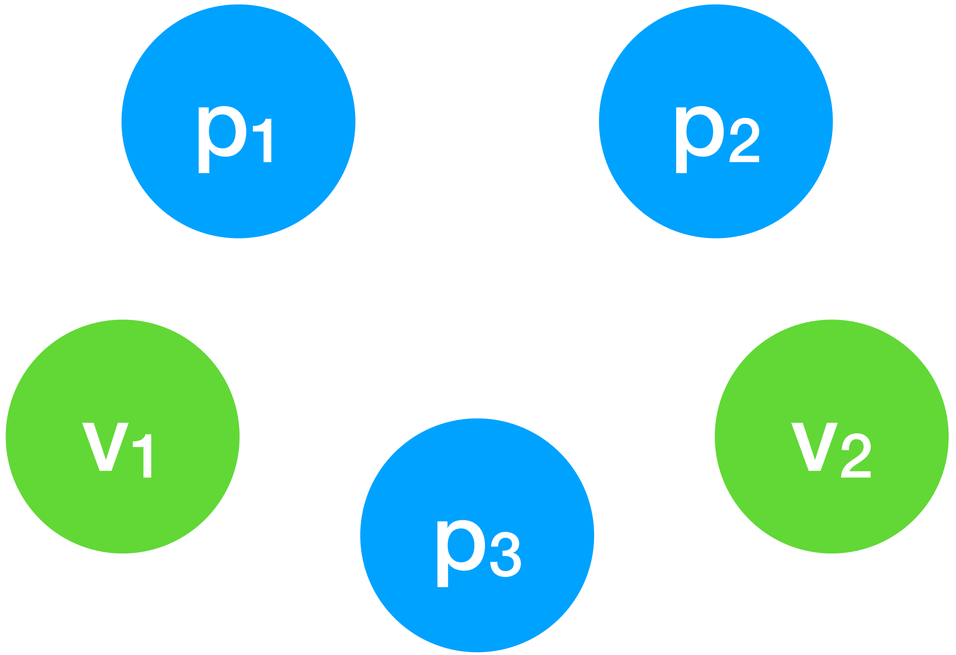


- Relative positions of SVs

PARTICLES AND VERTICES: TWO INPUT GRAPHS

- Particles (i.e. tracks) and vertices are two separate inputs with different feature vectors (*heterogenous graph*)

$$p_i = [p_{\mathrm{T}}^{\mathrm{rel}}, \phi^{\mathrm{rel}}, \eta^{\mathrm{rel}}, \ldots, d_{\mathrm{3D}}, \mathrm{cov}(p_{\mathrm{T}}, p_{\mathrm{T}}), \ldots]$$



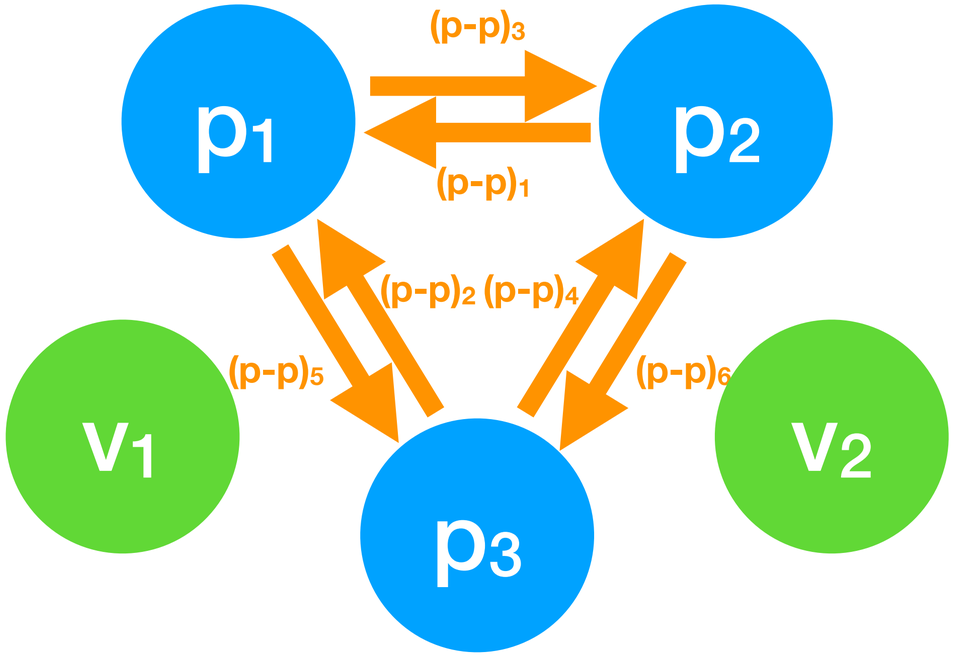
$$v_i = [p_{\mathrm{T}}^{\mathrm{rel}}, \phi^{\mathrm{rel}}, \eta^{\mathrm{rel}}, \ldots, n_{\mathrm{tracks}}, \cos\theta_{\mathrm{PV}}, \ldots]$$

PARTICLES AND VERTICES: TWO INPUT GRAPHS

- Particles (i.e. tracks) and vertices are two separate inputs with different feature vectors (*heterogenous graph*)

- GNNs typically consider a *homogenous graph* (e.g. particle-particle graph)

$$p_i = [p_{\mathrm{T}}^{\mathrm{rel}}, \phi^{\mathrm{rel}}, \eta^{\mathrm{rel}}, \ldots, d_{\mathrm{3D}}, \mathrm{cov}(p_{\mathrm{T}}, p_{\mathrm{T}}), \ldots]$$

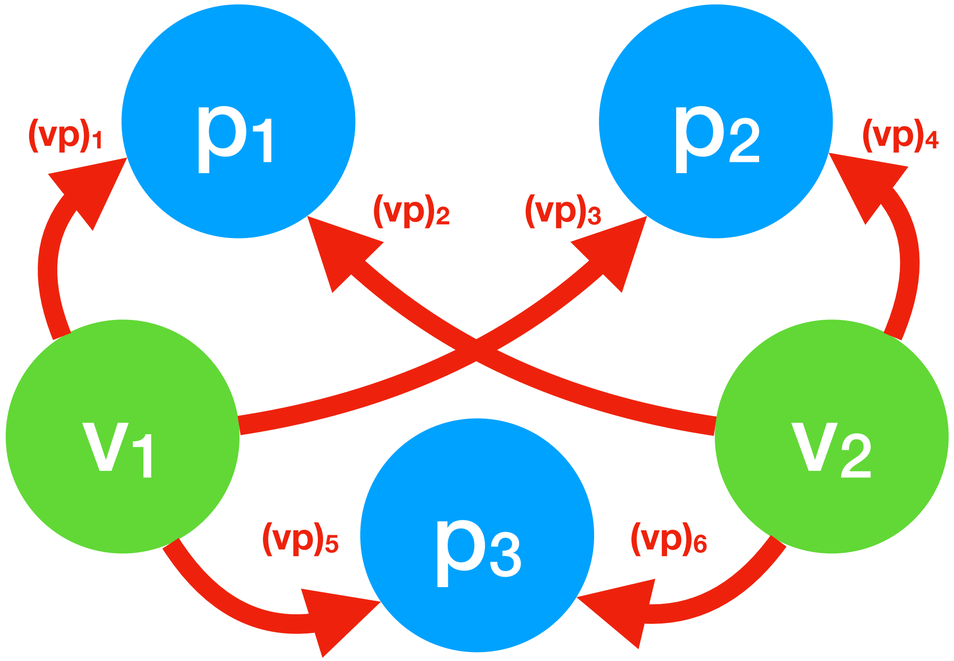


$$v_i = [p_{\mathrm{T}}^{\mathrm{rel}}, \phi^{\mathrm{rel}}, \eta^{\mathrm{rel}}, \ldots, n_{\mathrm{tracks}}, \cos\theta_{\mathrm{PV}}, \ldots]$$

PARTICLES AND VERTICES: TWO INPUT GRAPHS

- Particles (i.e. tracks) and vertices are two separate inputs with different feature vectors (*heterogenous graph*)
- GNNs typically consider a *homogenous graph* (e.g. particle-particle graph)
- Vertex-particle graph can also be considered

$$p_i = [p_{\mathrm{T}}^{\mathrm{rel}}, \phi^{\mathrm{rel}}, \eta^{\mathrm{rel}}, \ldots, d_{\mathrm{3D}}, \mathrm{cov}(p_{\mathrm{T}}, p_{\mathrm{T}}), \ldots]$$

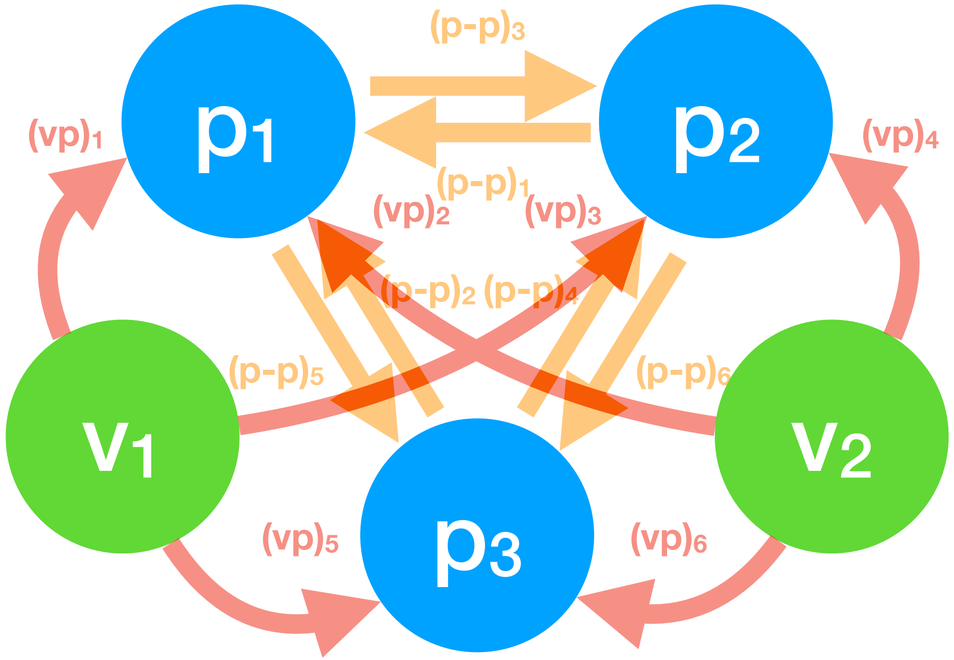


$$v_i = [p_{\mathrm{T}}^{\mathrm{rel}}, \phi^{\mathrm{rel}}, \eta^{\mathrm{rel}}, \ldots, n_{\mathrm{tracks}}, \cos\theta_{\mathrm{PV}}, \ldots]$$

PARTICLES AND VERTICES: TWO INPUT GRAPHS

- Particles (i.e. tracks) and vertices are two separate inputs with different feature vectors (*heterogenous graph*)
- GNNs typically consider a *homogenous graph* (e.g. particle-particle graph)
- Vertex-particle graph can also be considered
- Combined GNN can consider both by constructing two separate graphs

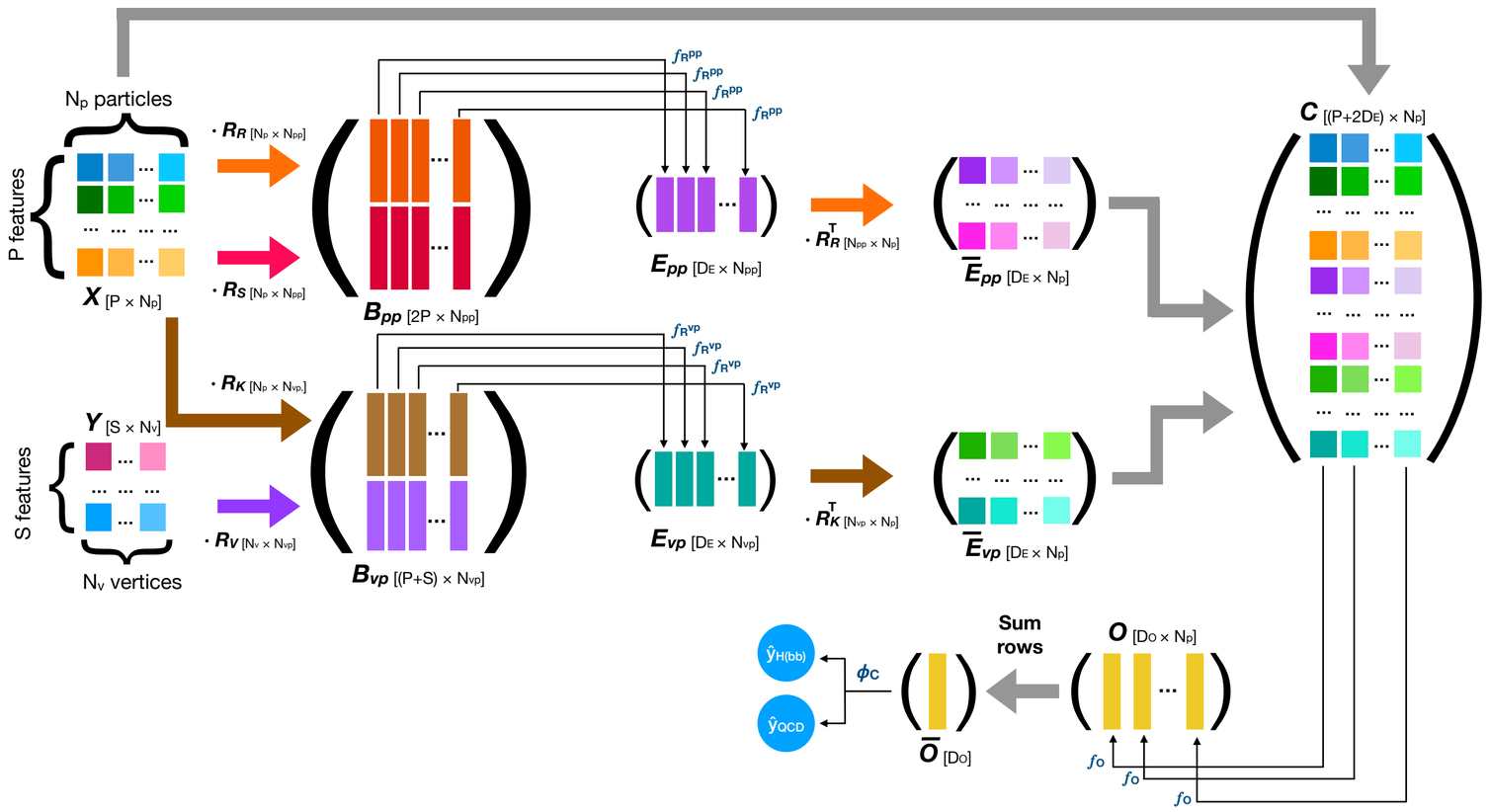
$$p_i = [p_{\mathrm{T}}^{\mathrm{rel}}, \phi^{\mathrm{rel}}, \eta^{\mathrm{rel}}, \ldots, d_{\mathrm{3D}}, \mathrm{cov}(p_{\mathrm{T}}, p_{\mathrm{T}}), \ldots]$$

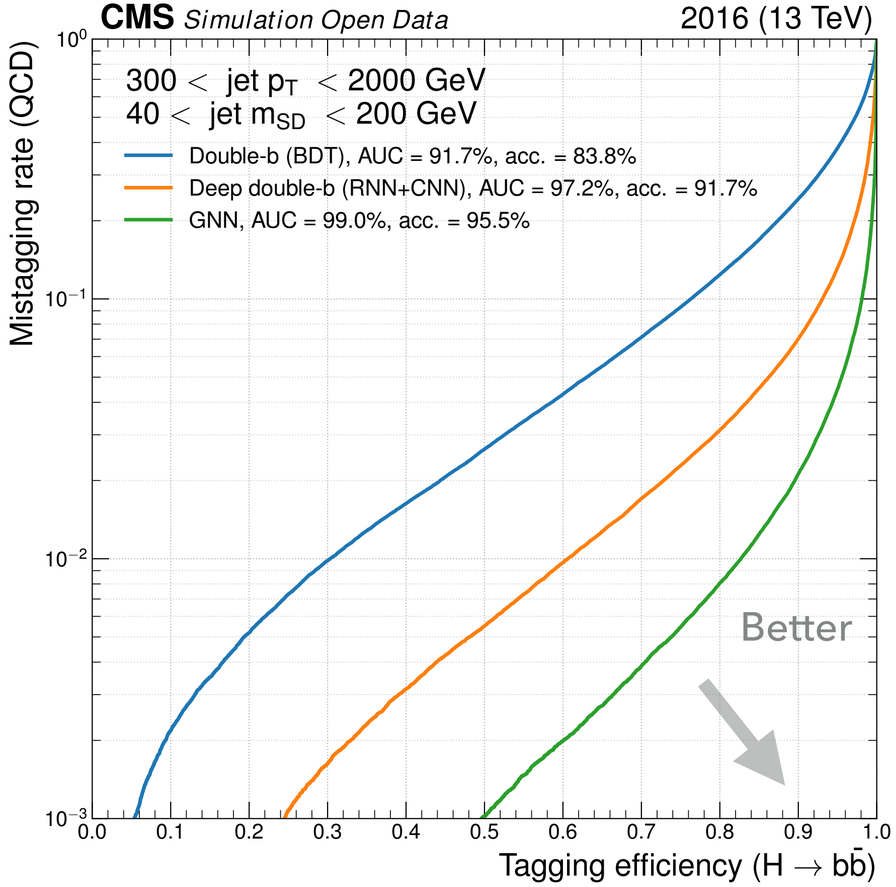


$$v_i = [p_{\mathrm{T}}^{\mathrm{rel}}, \phi^{\mathrm{rel}}, \eta^{\mathrm{rel}}, \ldots, n_{\mathrm{tracks}}, \cos\theta_{\mathrm{PV}}, \ldots]$$

MODEL AND PERFORMANCE

- Edge convolutions for particle-particle and particle-vertex connections update particle features; summed particle features used to predict H(bb) or QCD prob.
- GNN improves on previous methods

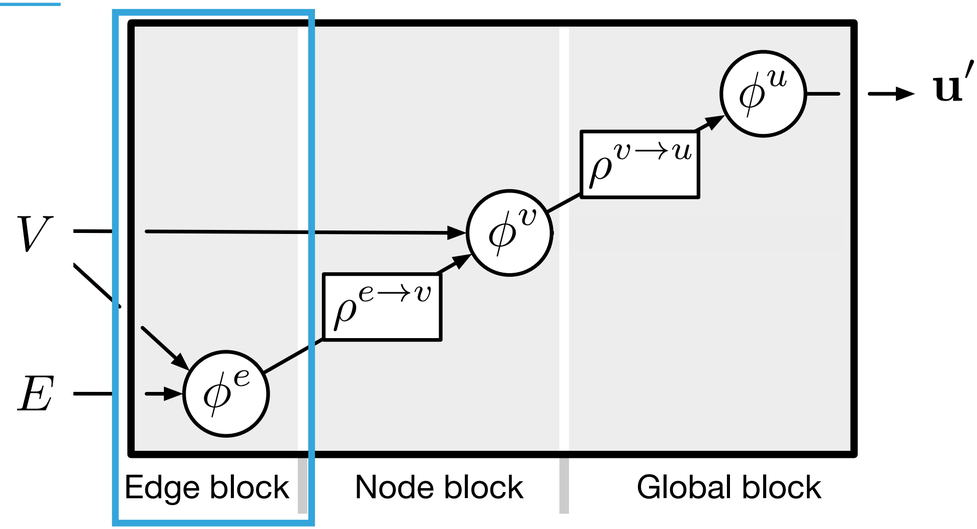




DEMO

- Simplified GNN with PyTorch Geometric: https://jmduarte.github.io/capstone-particle-physics-domain/weeks/08-extending.html

```python
class EdgeBlock(torch.nn.Module):
    def __init__(self):
        super(EdgeBlock, self).__init__()
        self.edge_mlp = Seq(Lin(inputs*2, hidden),
                            BatchNorm1d(hidden),
                            ReLU(),
                            Lin(hidden, hidden))

    def forward(self, src, dest, edge_attr, u, batch):
        out = torch.cat([src, dest], 1)
        return self.edge_mlp(out)
```

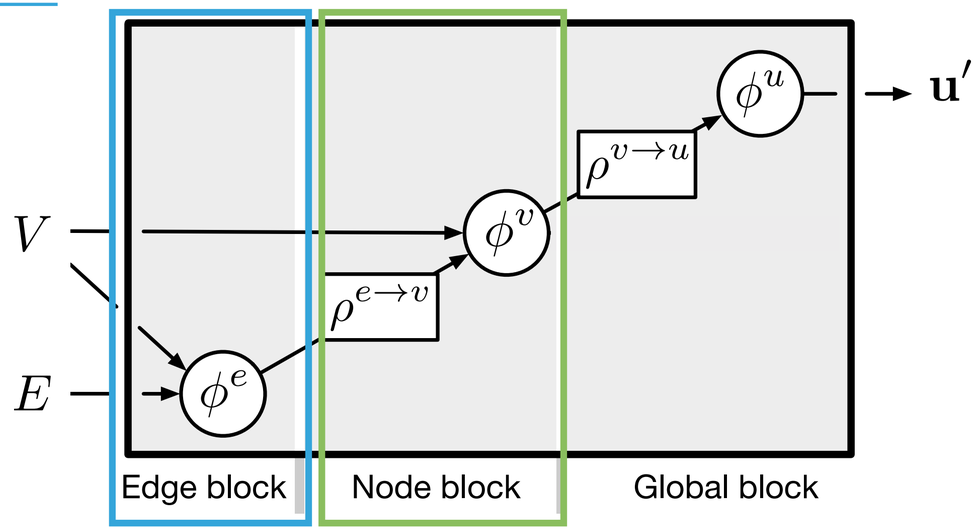


DEMO

- Simplified GNN with PyTorch Geometric: https://jmduarte.github.io/capstone-particle-physics-domain/weeks/08-extending.html

```python
class EdgeBlock(torch.nn.Module):
    def __init__(self):
        super(EdgeBlock, self).__init__()
        self.edge_mlp = Seq(Lin(inputs*2, hidden),
                            BatchNorm1d(hidden),
                            ReLU(),
                            Lin(hidden, hidden))

    def forward(self, src, dest, edge_attr, u, batch):
        out = torch.cat([src, dest], 1)
        return self.edge_mlp(out)
```

```python
class NodeBlock(torch.nn.Module):
    def __init__(self):
        super(NodeBlock, self).__init__()
        self.node_mlp_1 = Seq(Lin(inputs+hidden, hidden),
                              BatchNorm1d(hidden),
                              ReLU(),
                              Lin(hidden, hidden))
        self.node_mlp_2 = Seq(Lin(inputs+hidden, hidden),
                              BatchNorm1d(hidden),
                              ReLU(),
                              Lin(hidden, hidden))

    def forward(self, x, edge_index, edge_attr, u, batch):
        row, col = edge_index
        out = torch.cat([x[row], edge_attr], dim=1)
        out = self.node_mlp_1(out)
        out = scatter_mean(out, col, dim=0, dim_size=x.size(0))
        out = torch.cat([x, out], dim=1)
        return self.node_mlp_2(out)
```



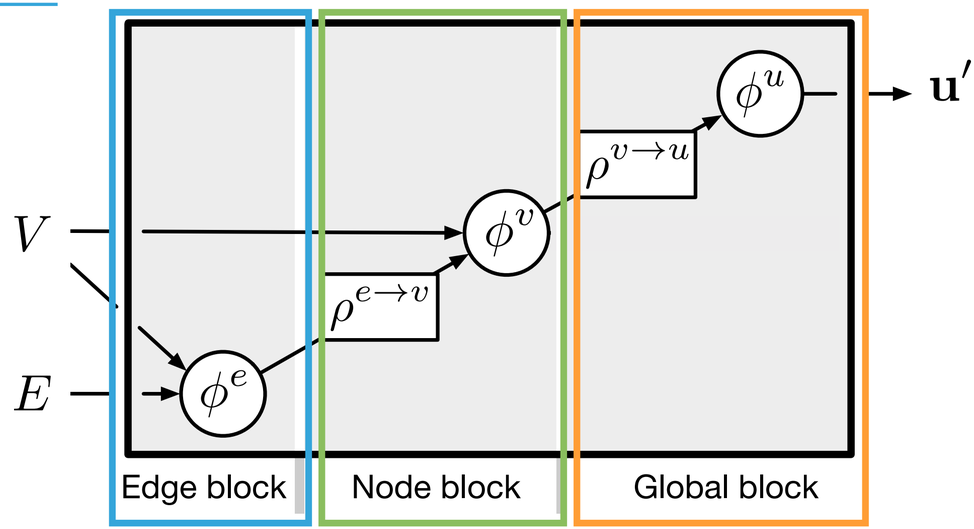
DEMO

- Simplified GNN with PyTorch Geometric: https://jmduarte.github.io/capstone-particle-physics-domain/weeks/08-extending.html

```python
class EdgeBlock(torch.nn.Module):
    def __init__(self):
        super(EdgeBlock, self).__init__()
        self.edge_mlp = Seq(Lin(inputs*2, hidden),
                            BatchNorm1d(hidden),
                            ReLU(),
                            Lin(hidden, hidden))

    def forward(self, src, dest, edge_attr, u, batch):
        out = torch.cat([src, dest], 1)
        return self.edge_mlp(out)
```

```python
class GlobalBlock(torch.nn.Module):
    def __init__(self):
        super(GlobalBlock, self).__init__()
        self.global_mlp = Seq(Lin(hidden, hidden),
                              BatchNorm1d(hidden),
                              ReLU(),
                              Lin(hidden, outputs))

    def forward(self, x, edge_index, edge_attr, u, batch):
        out = scatter_mean(x, batch, dim=0)
        return self.global_mlp(out)
```

```python
class NodeBlock(torch.nn.Module):
    def __init__(self):
        super(NodeBlock, self).__init__()
        self.node_mlp_1 = Seq(Lin(inputs+hidden, hidden),
                              BatchNorm1d(hidden),
                              ReLU(),
                              Lin(hidden, hidden))
        self.node_mlp_2 = Seq(Lin(inputs+hidden, hidden),
                              BatchNorm1d(hidden),
                              ReLU(),
                              Lin(hidden, hidden))

    def forward(self, x, edge_index, edge_attr, u, batch):
        row, col = edge_index
        out = torch.cat([x[row], edge_attr], dim=1)
        out = self.node_mlp_1(out)
        out = scatter_mean(out, col, dim=0, dim_size=x.size(0))
        out = torch.cat([x, out], dim=1)
        return self.node_mlp_2(out)
```



DEMO

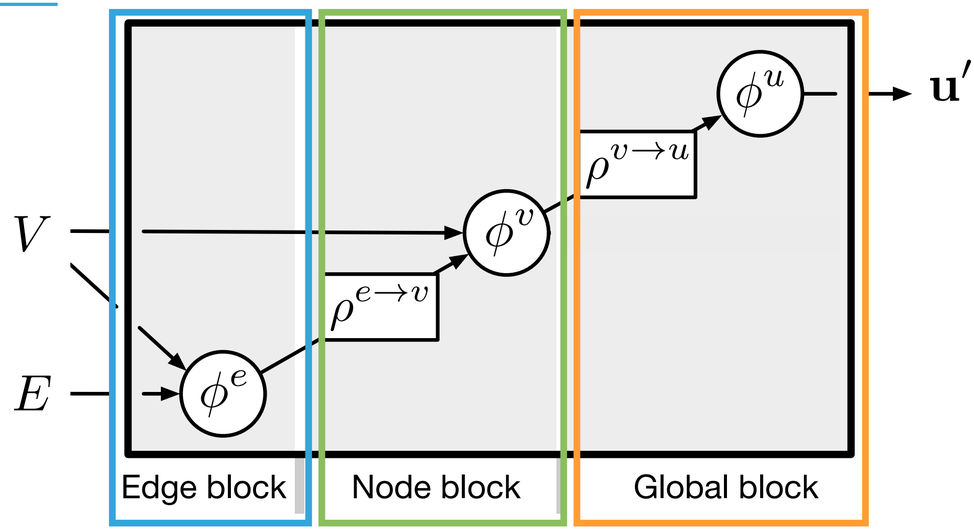
- Simplified GNN with PyTorch Geometric: https://jmduarte.github.io/capstone-particle-physics-domain/weeks/08-extending.html

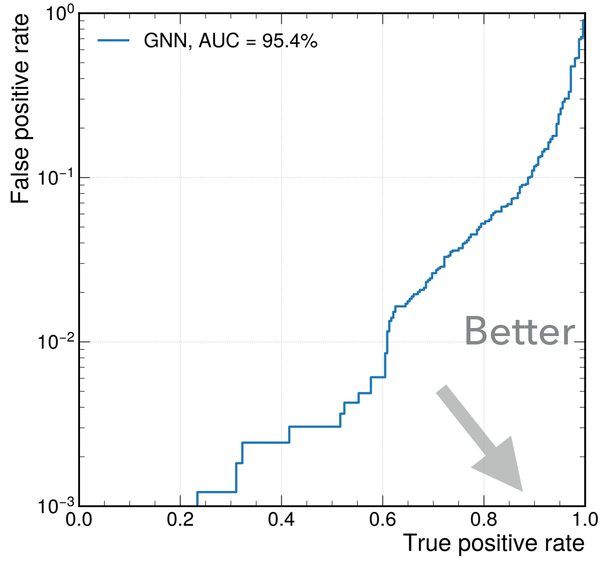
```python
class EdgeBlock(torch.nn.Module):
    def __init__(self):
        super(EdgeBlock, self).__init__()
        self.edge_mlp = Seq(Lin(inputs*2, hidden),
                            BatchNorm1d(hidden),
                            ReLU(),
                            Lin(hidden, hidden))

    def forward(self, src, dest, edge_attr, u, batch):
        out = torch.cat([src, dest], 1)
        return self.edge_mlp(out)
```

```python
class NodeBlock(torch.nn.Module):
    def __init__(self):
        super(NodeBlock, self).__init__()
        self.node_mlp_1 = Seq(Lin(inputs+hidden, hidden),
                              BatchNorm1d(hidden),
                              ReLU(),
                              Lin(hidden, hidden))
        self.node_mlp_2 = Seq(Lin(inputs+hidden, hidden),
                              BatchNorm1d(hidden),
                              ReLU(),
                              Lin(hidden, hidden))

    def forward(self, x, edge_index, edge_attr, u, batch):
        row, col = edge_index
        out = torch.cat([x[row], edge_attr], dim=1)
        out = self.node_mlp_1(out)
        out = scatter_mean(out, col, dim=0, dim_size=x.size(0))
        out = torch.cat([x, out], dim=1)
        return self.node_mlp_2(out)
```

```python
class GlobalBlock(torch.nn.Module):
    def __init__(self):
        super(GlobalBlock, self).__init__()
        self.global_mlp = Seq(Lin(hidden, hidden),
                              BatchNorm1d(hidden),
                              ReLU(),
                              Lin(hidden, outputs))

    def forward(self, x, edge_index, edge_attr, u, batch):
        out = scatter_mean(x, batch, dim=0)
        return self.global_mlp(out)
```

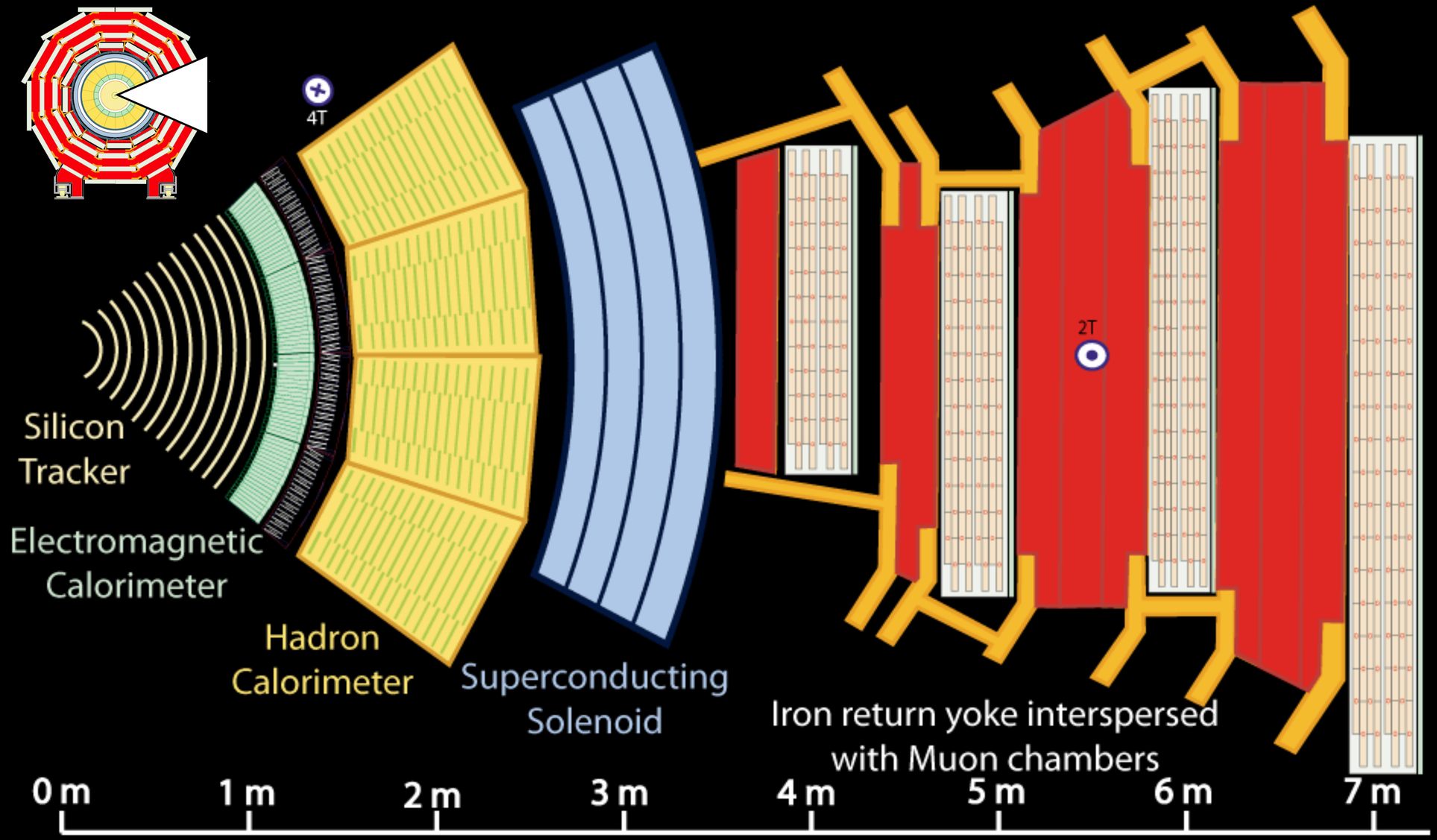
```python
class InteractionNetwork(torch.nn.Module):
    def __init__(self):
        super(InteractionNetwork, self).__init__()
        self.interactionnetwork = MetaLayer(EdgeBlock(), NodeBlock(), GlobalBlock())
        self.bn = BatchNorm1d(inputs)

    def forward(self, x, edge_index, batch):
        x = self.bn(x)
        x, edge_attr, u = self.interactionnetwork(x, edge_index, None, None, batch)
        return u
```

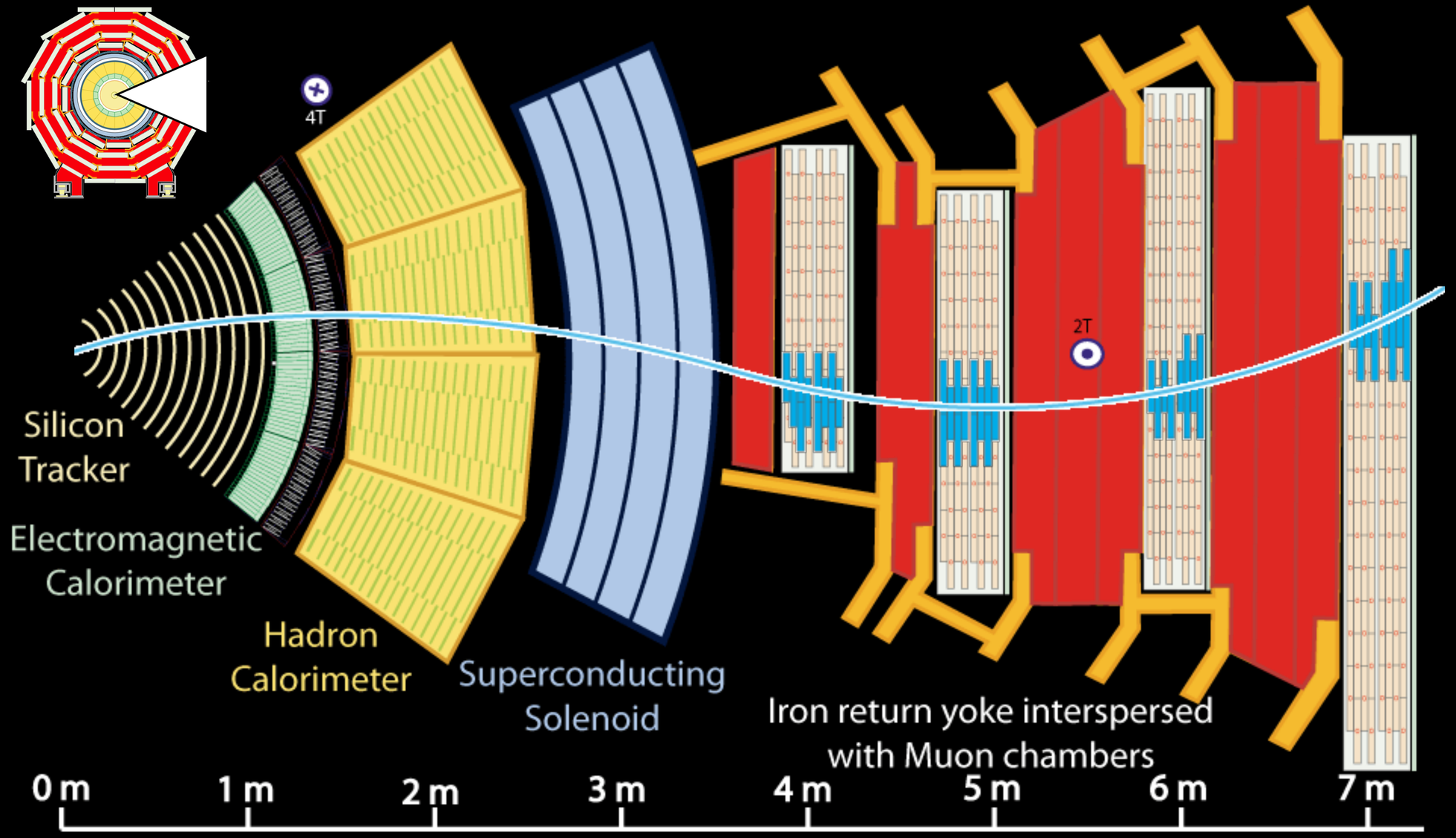


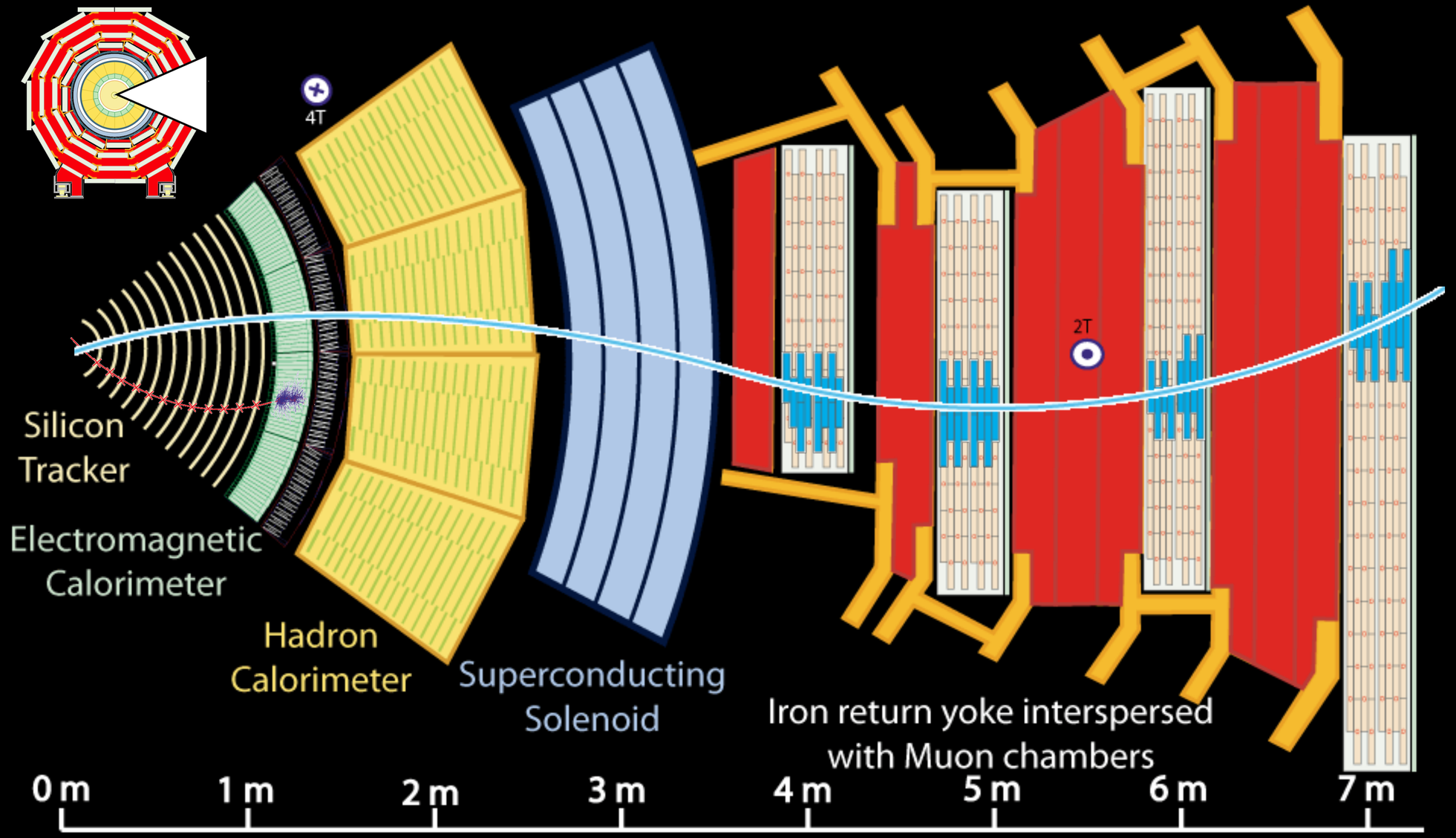


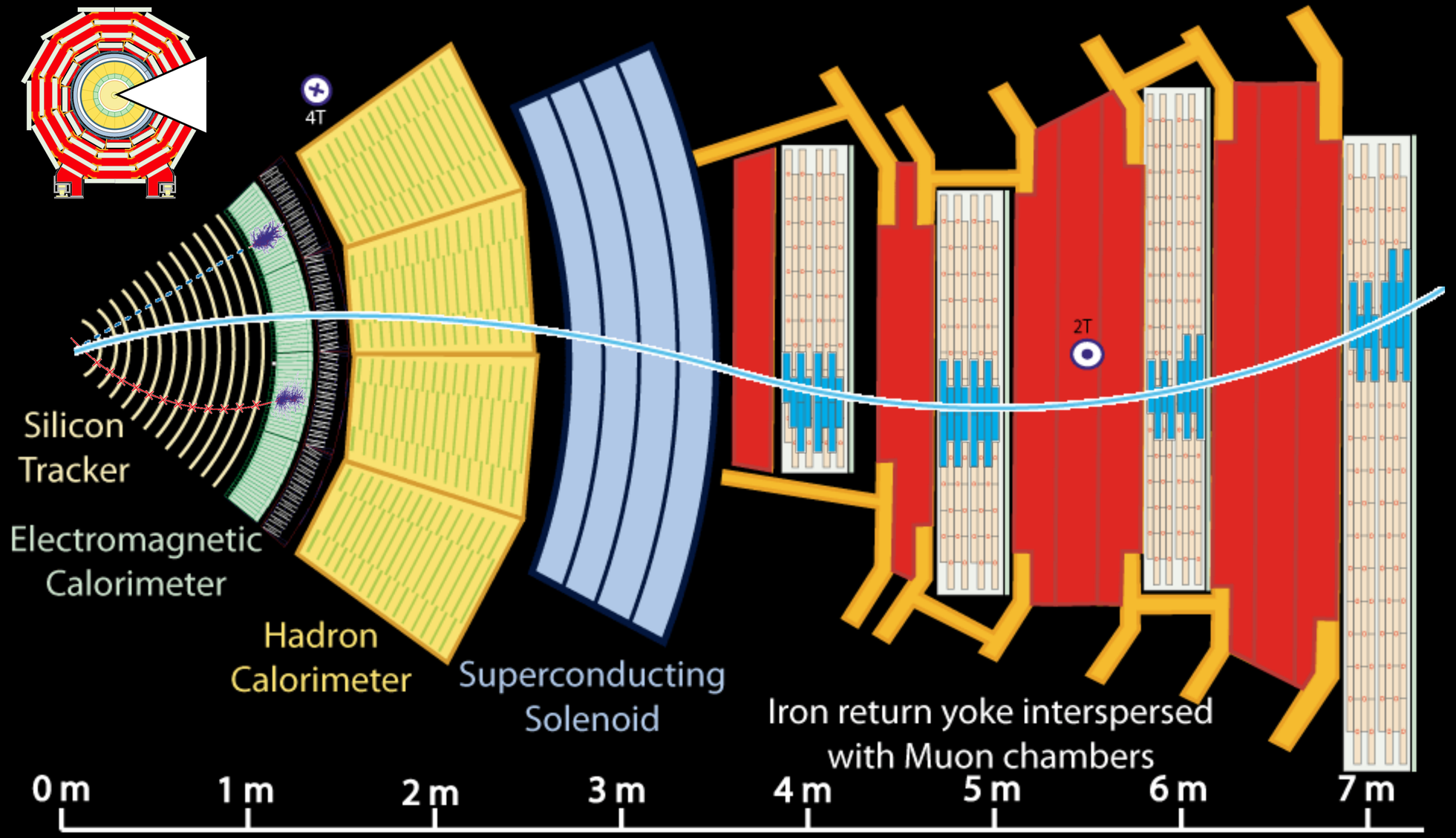
PARTICLE-FLOW

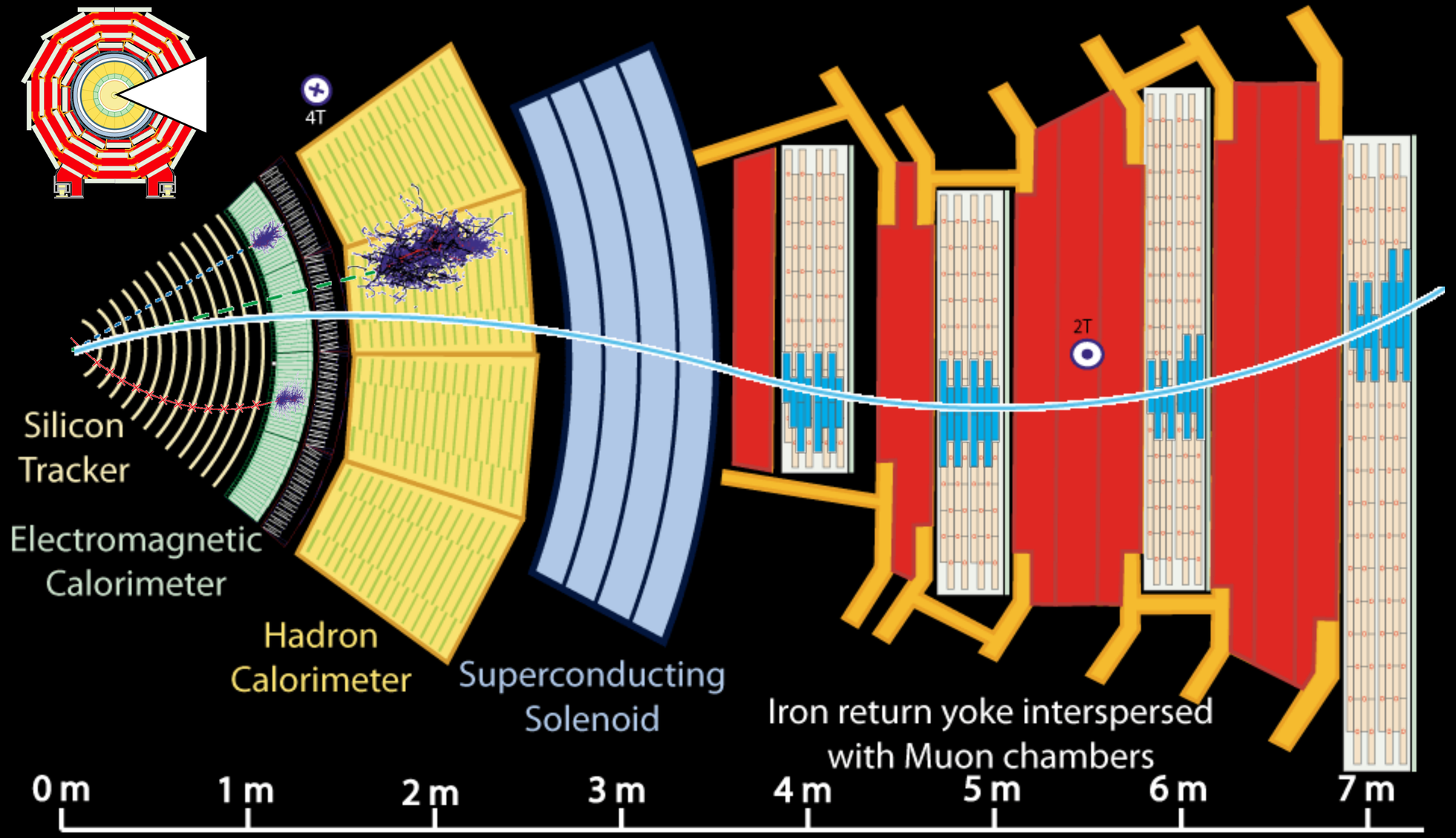
UCSD

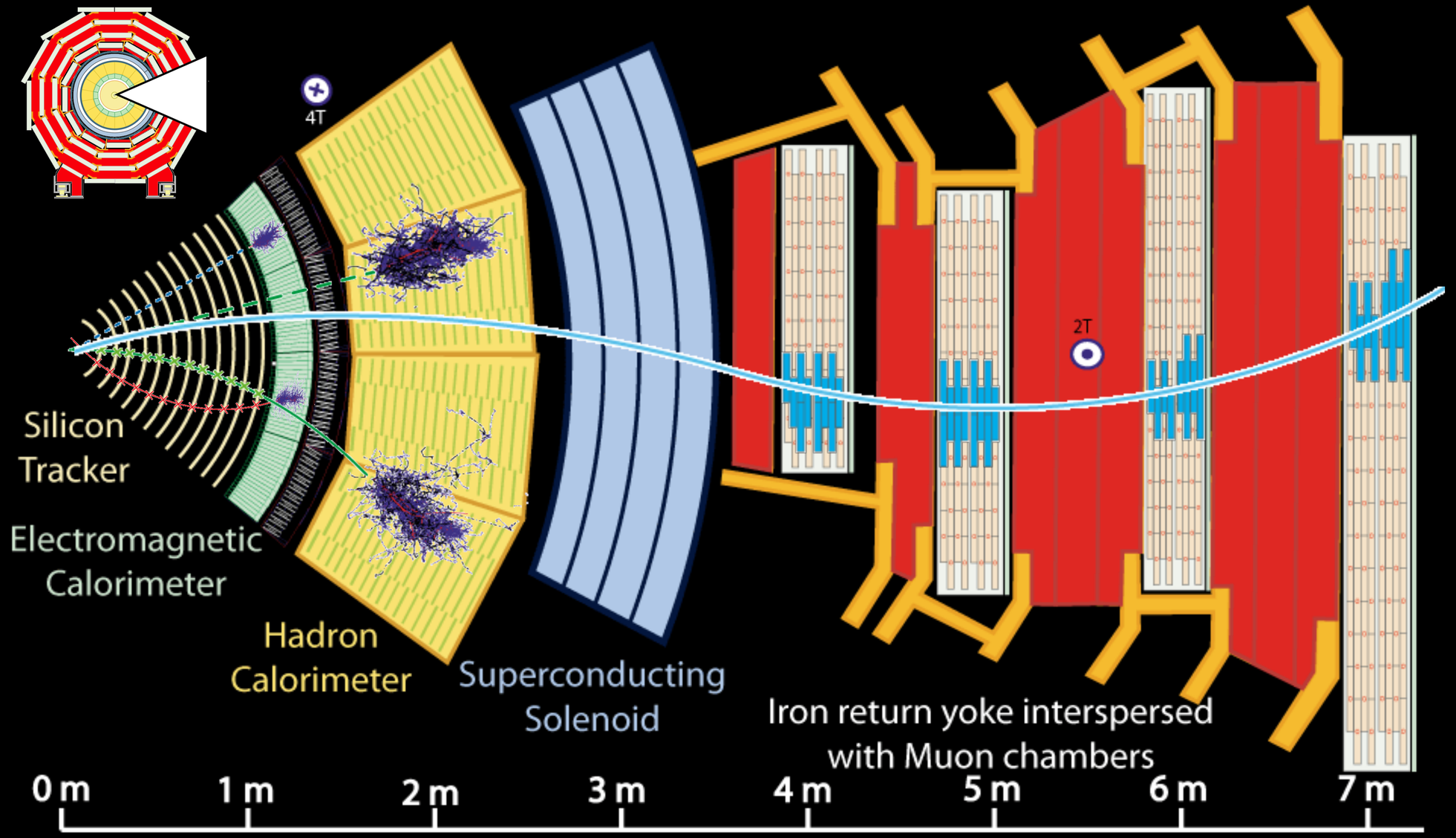

4T
2T
Silicon Tracker
Electromagnetic Calorimeter
Hadron Calorimeter
Superconducting Solenoid
Iron return yoke interspersed with Muon chambers
0 m
1 m
2 m
3 m
4 m
5 m
6 m
7 m
Key:
Muon
Electron
Charged Hadron (e.g. Pion)
Neutral Hadron (e.g. Neutron)
Photon


4T
2T
Silicon Tracker
Electromagnetic Calorimeter
Hadron Calorimeter
Superconducting Solenoid
Iron return yoke interspersed with Muon chambers
0 m
1 m
2 m
3 m
4 m
5 m
6 m
7 m
Key:
Muon
Electron
Charged Hadron (e.g. Pion)
Neutral Hadron (e.g. Neutron)
Photon

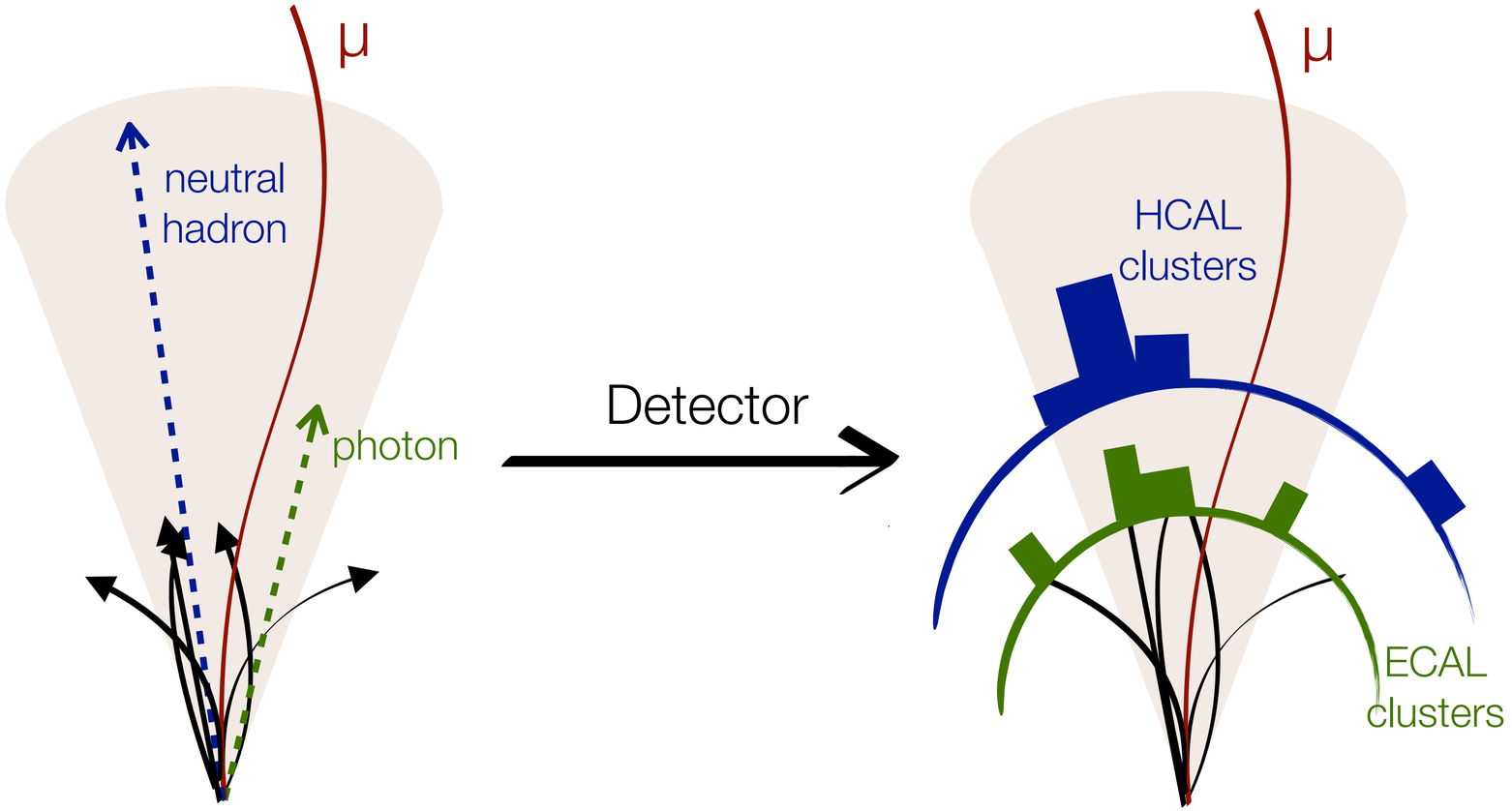

4T
2T
Silicon Tracker
Electromagnetic Calorimeter
Hadron Calorimeter
Superconducting Solenoid
Iron return yoke interspersed with Muon chambers
0 m
1 m
2 m
3 m
4 m
5 m
6 m
7 m
Key:
Muon
Electron
Charged Hadron (e.g. Pion)
Neutral Hadron (e.g. Neutron)
Photon


4T
2T
Silicon Tracker
Electromagnetic Calorimeter
Hadron Calorimeter
Superconducting Solenoid
Iron return yoke interspersed with Muon chambers
0 m
1 m
2 m
3 m
4 m
5 m
6 m
7 m
Key:
Muon
Electron
Charged Hadron (e.g. Pion)
Neutral Hadron (e.g. Neutron)
Photon


4T
2T
Silicon Tracker
Electromagnetic Calorimeter
Hadron Calorimeter
Superconducting Solenoid
Iron return yoke interspersed with Muon chambers
0 m
1 m
2 m
3 m
4 m
5 m
6 m
7 m
Key:
Muon
Electron
Charged Hadron (e.g. Pion)
Neutral Hadron (e.g. Neutron)
Photon

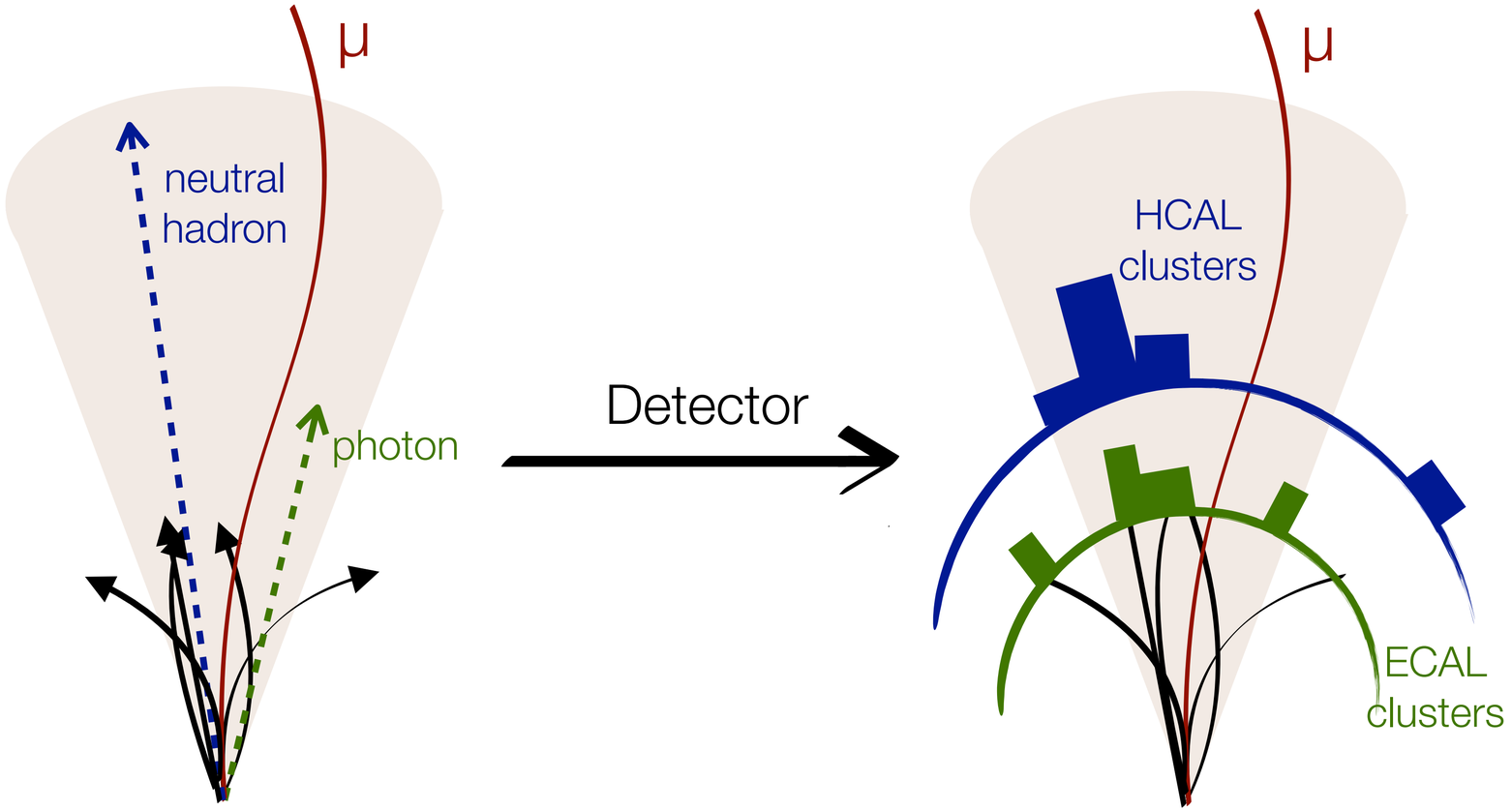

4T
2T
Silicon Tracker
Electromagnetic Calorimeter
Hadron Calorimeter
Superconducting Solenoid
Iron return yoke interspersed with Muon chambers
0 m
1 m
2 m
3 m
4 m
5 m
6 m
7 m
Key:
Muon
Electron
Charged Hadron (e.g. Pion)
Neutral Hadron (e.g. Neutron)
Photon

PARTICLE-FLOW RECONSTRUCTION



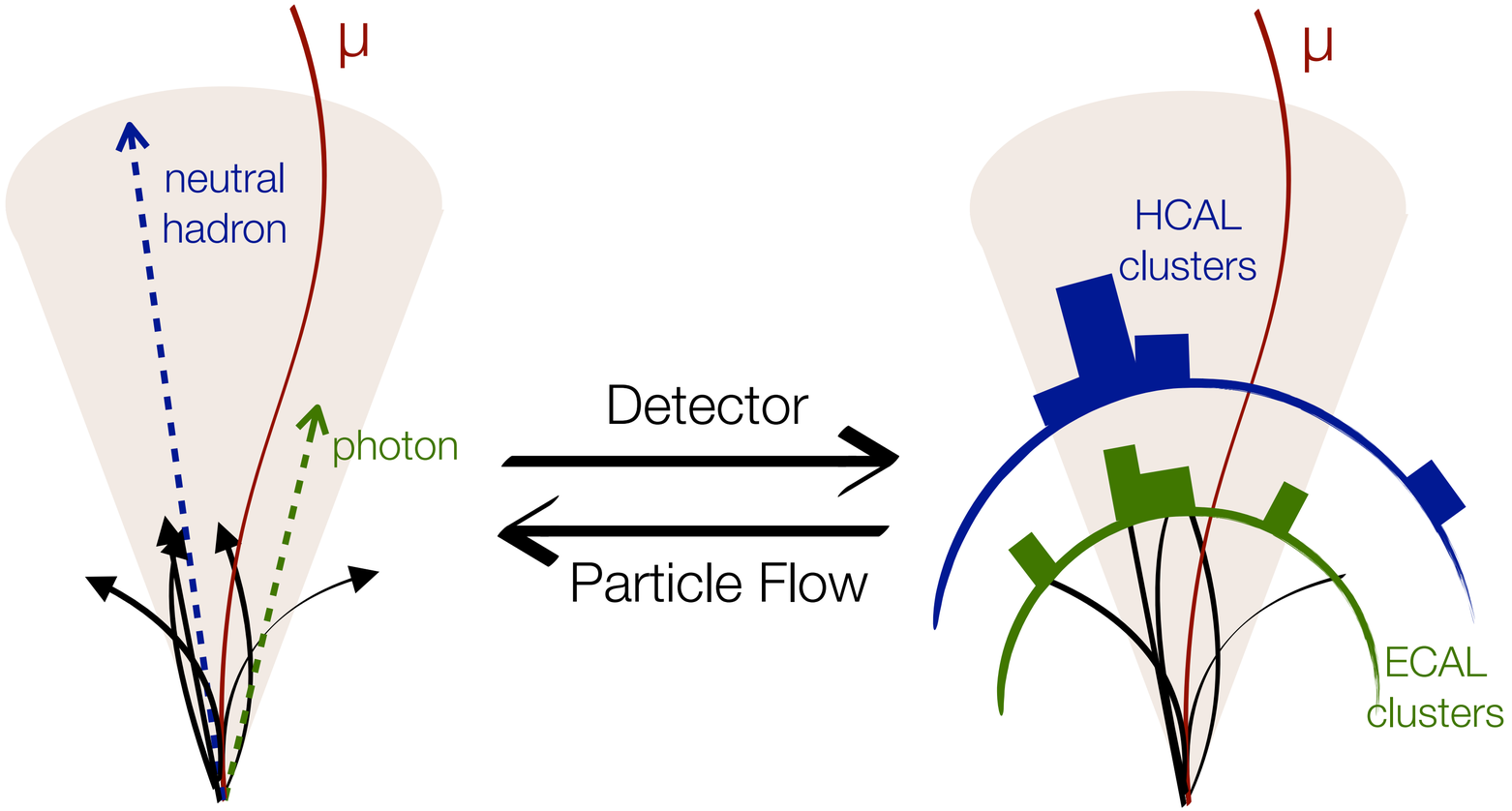
PARTICLE-FLOW RECONSTRUCTION

- Particles interact with detector, leaving energy deposits and tracks



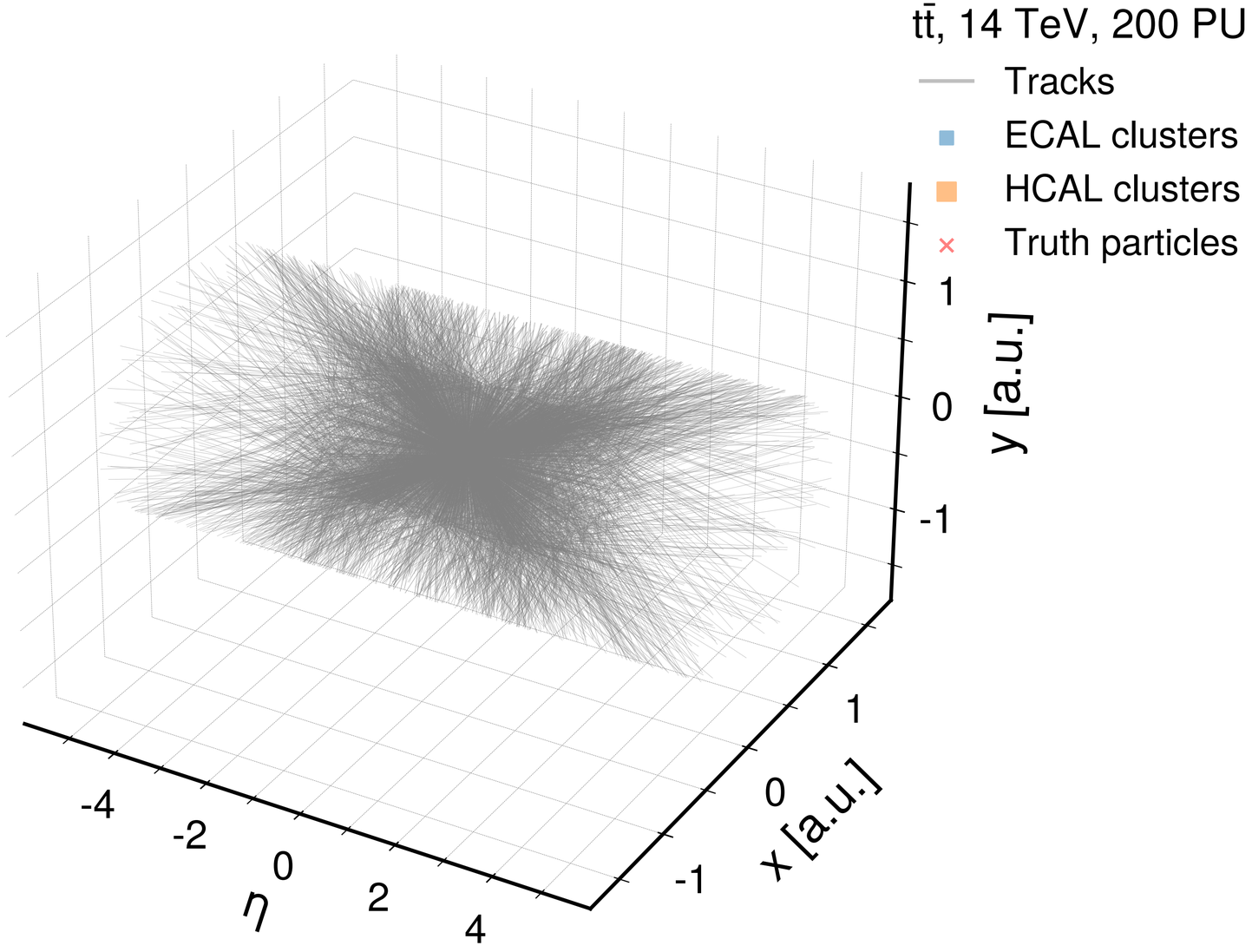
PARTICLE-FLOW RECONSTRUCTION

- Particles interact with detector, leaving energy deposits and tracks
- Efficient combination of info. from complementary detector subsystems to produce a holistic, particle interpretation of the event (that improves on any individual subsystem)



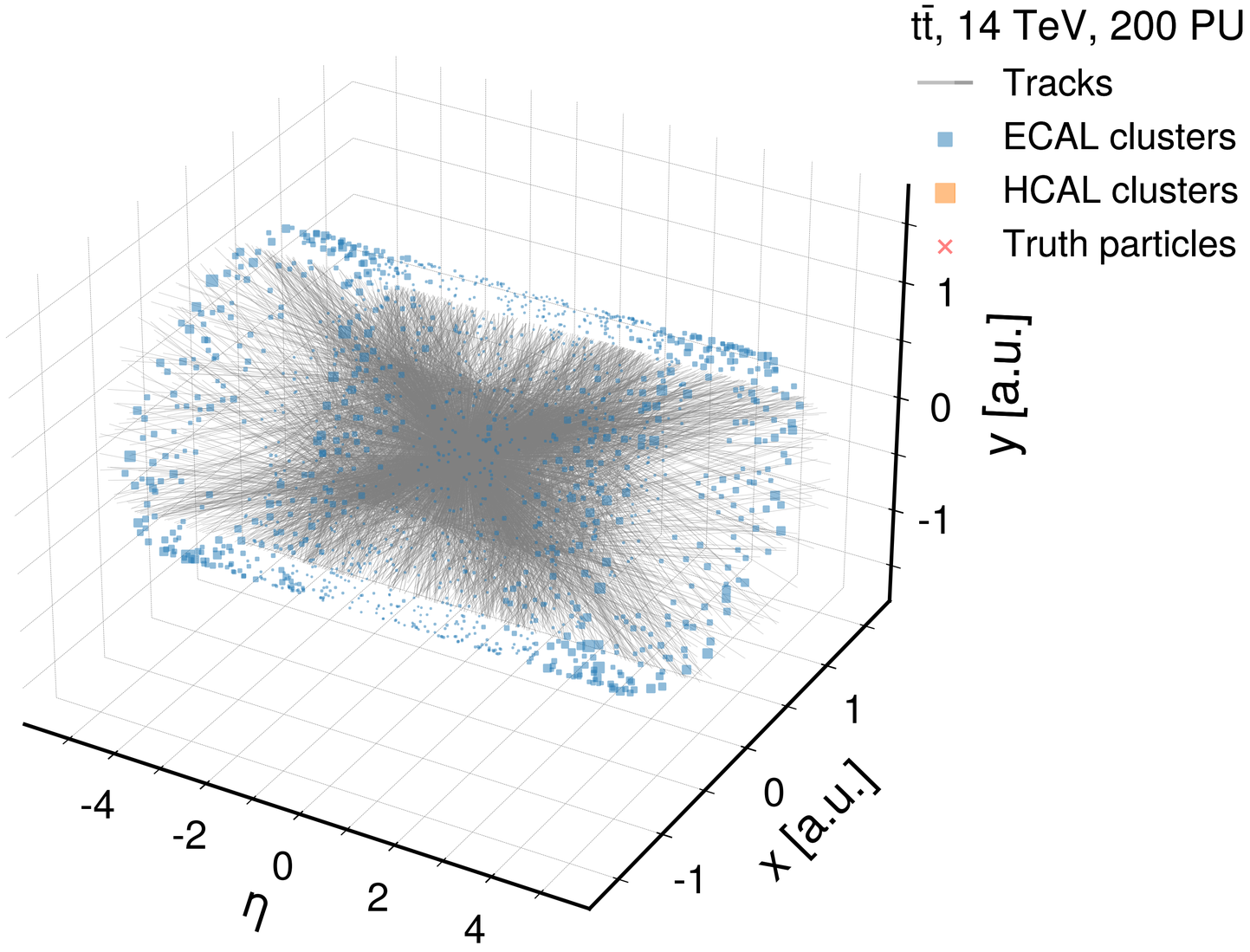
PARTICLE-FLOW AS ML TASK

- Input set (or point could or heterogeneous graph) of tracks,



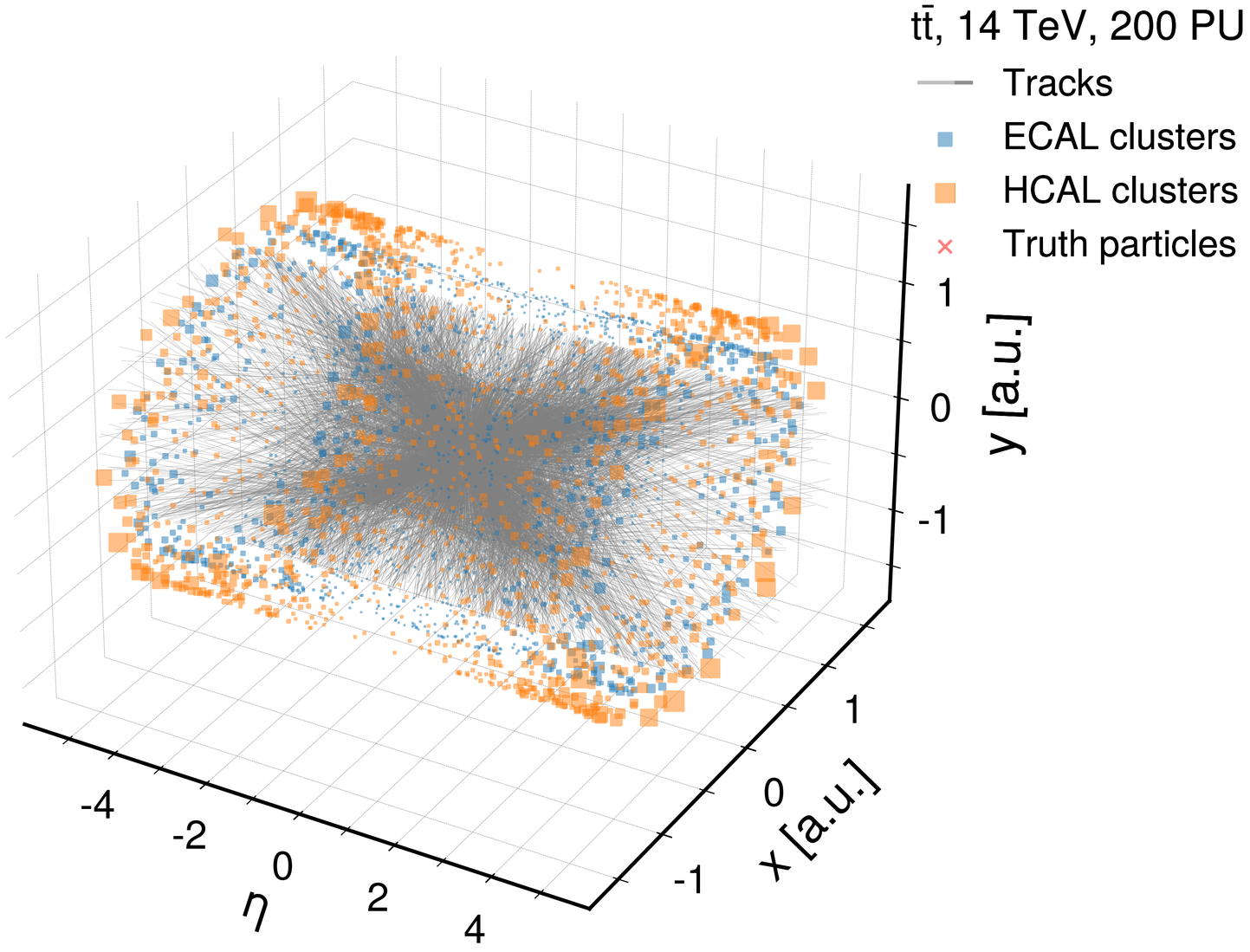
PARTICLE-FLOW AS ML TASK

- Input set (or point could or heterogeneous graph) of tracks, ECAL clusters,



PARTICLE-FLOW AS ML TASK

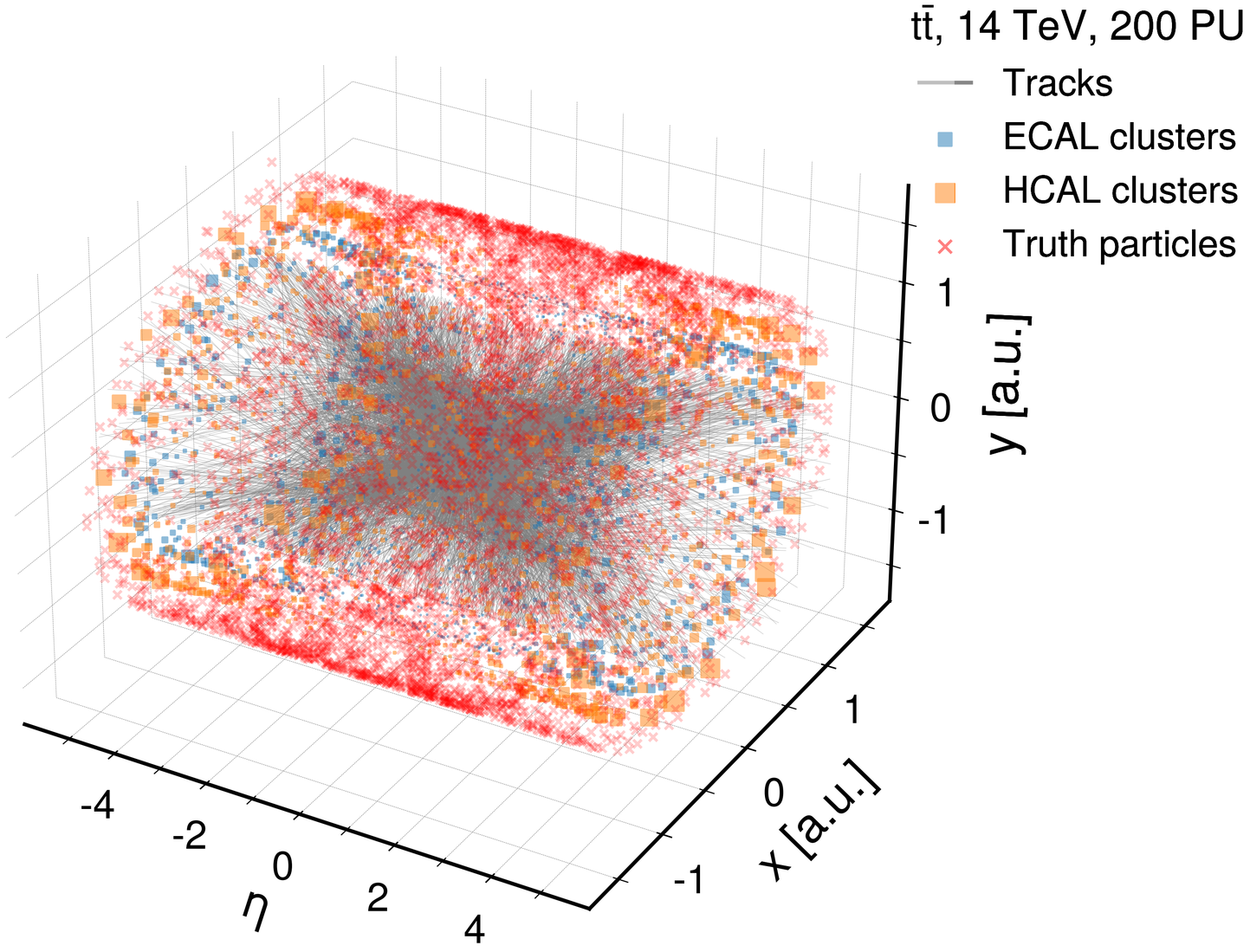
- Input set (or point could or heterogeneous graph) of tracks, ECAL clusters, HCAL clusters



PARTICLE-FLOW AS ML TASK

- Input set (or point could or heterogeneous graph) of tracks, ECAL clusters, HCAL clusters
- Target set of truth particles

$$Y = \{y_i\}$$



PARTICLE-FLOW AS ML TASK

- Input set (or point could or heterogeneous graph) of tracks, ECAL clusters, HCAL clusters
- Target set of truth particles

$$Y = \{y_i\}$$

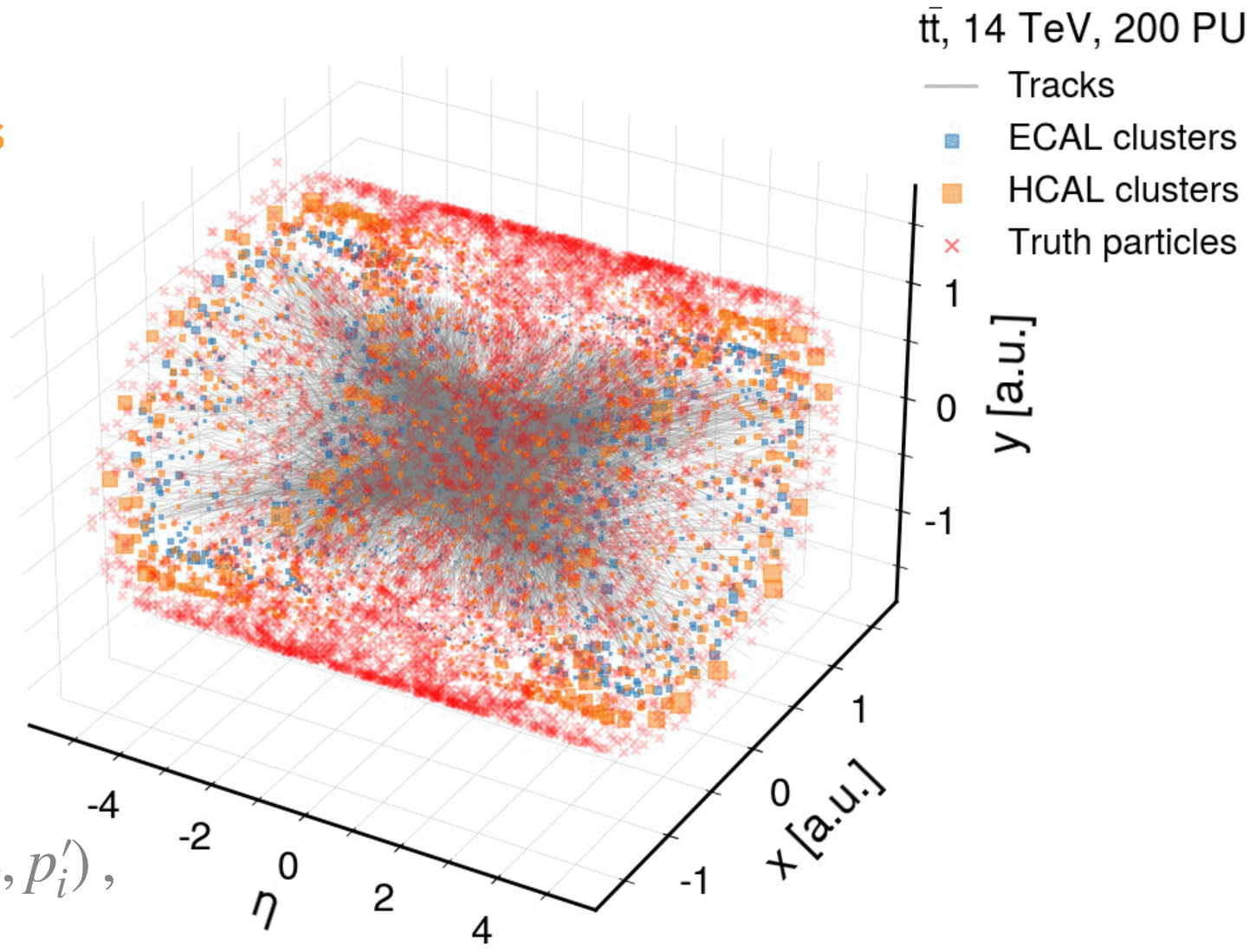
- Goal: construct a mapping

$$\mathcal{U}(X) = Y' \sim Y$$

that minimizes some distance

$$||Y - Y'|| \equiv \sum_{i \in \text{event}} L(y_i, y_i'),$$

$$L(y_i, y_i') \equiv \text{CLS}(c_i, c_i') + \alpha \text{REG}(p_i, p_i'),$$



SET-TO-SET DATA PREPROCESSING

- Formulate set-to-set learning task with data preprocessing

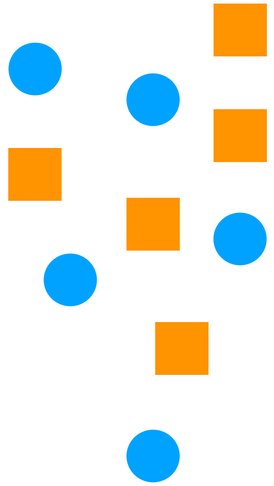
SET-TO-SET DATA PREPROCESSING

- Formulate set-to-set learning task with data preprocessing
- Input set arranged in (arbitrarily ordered) matrix

$$X = \begin{bmatrix} \cdots \\ \text{type}, p_\text{T}, E_\text{ECAL}, E_\text{HCAL}, \eta, \phi, \eta_\text{outer}, \phi_\text{outer}, q \\ \cdots \end{bmatrix}$$

$$\text{type} \in \{\text{track}, \text{cluster}\}$$

Input set $X = \{x_i\}$



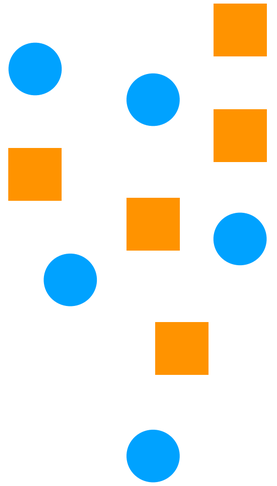
SET-TO-SET DATA PREPROCESSING

- Formulate set-to-set learning task with data preprocessing
- Input set arranged in (arbitrarily ordered) matrix

$$X = \begin{bmatrix} \cdots \\ \text{type}, p_\text{T}, E_\text{ECAL}, E_\text{HCAL}, \eta, \phi, \eta_\text{outer}, \phi_\text{outer}, q \\ \cdots \end{bmatrix}$$

$$\text{type} \in \{\text{track}, \text{cluster}\}$$

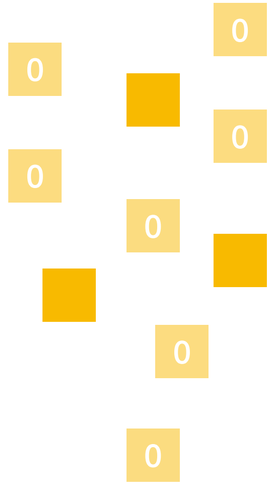
Input set $X = \{x_i\}$



- Target set zero-padded to same size $|X| = |Y|$, with each output particle arranged in same array position as best-matched input element

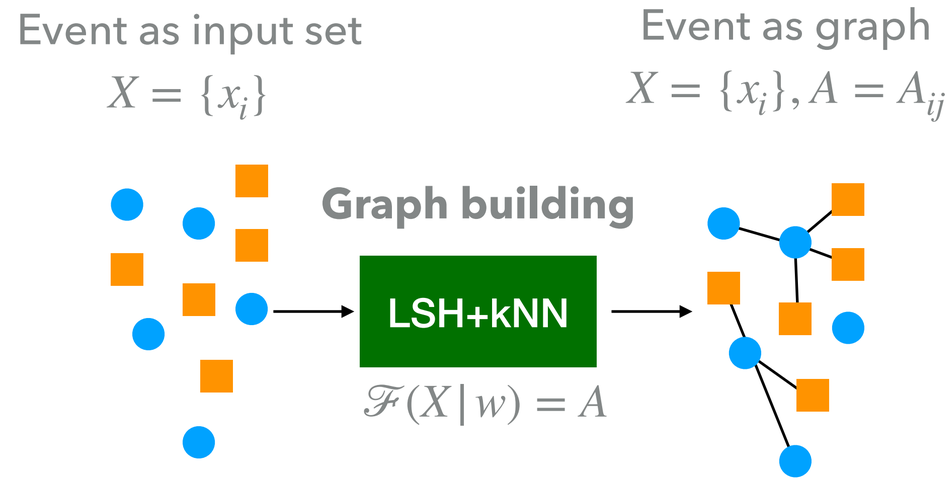
$$Y = \begin{bmatrix} \cdots \\ \text{PID}, p_\text{T}, E, \eta, \phi, q \\ \cdots \end{bmatrix} \quad \text{PID} \in \{\text{none}, \text{charged hadron}, \text{neutral hadron}, \gamma, e^\pm, \mu^\pm\}$$

Target set $Y = \{y_i\}$



GRAPH NEURAL NETWORK APPROACH

- Convert input set to a locally, sparsely connected graph

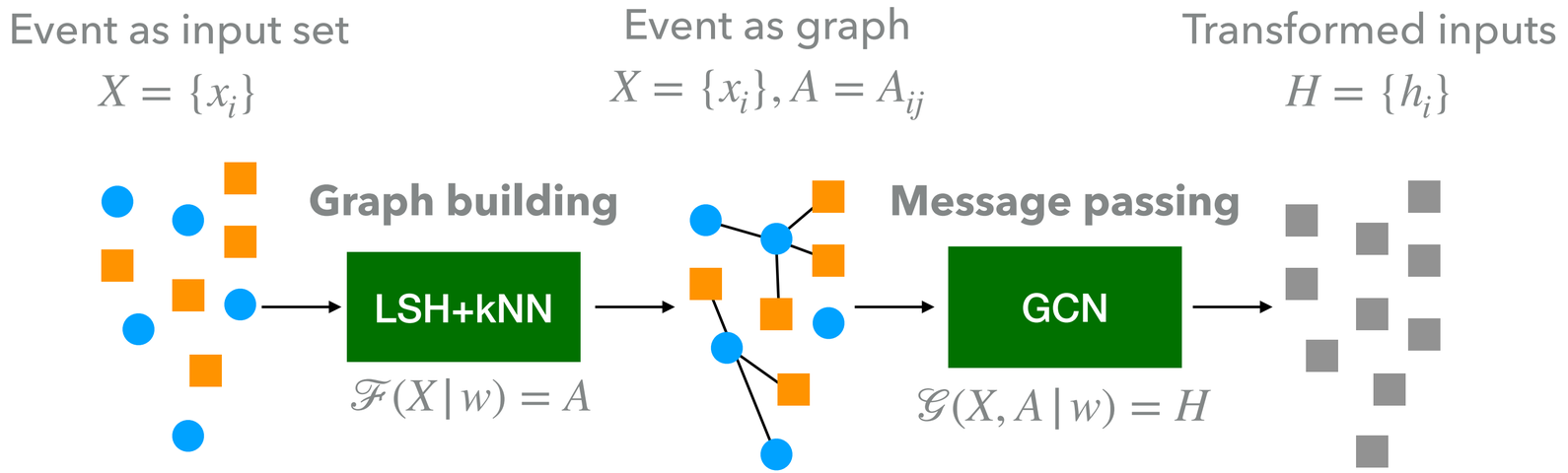


$h_i \in \mathbb{R}^N, N = 256$

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

GRAPH NEURAL NETWORK APPROACH

- Convert input set to a locally, sparsely connected graph
- Message-passing NN to transform features

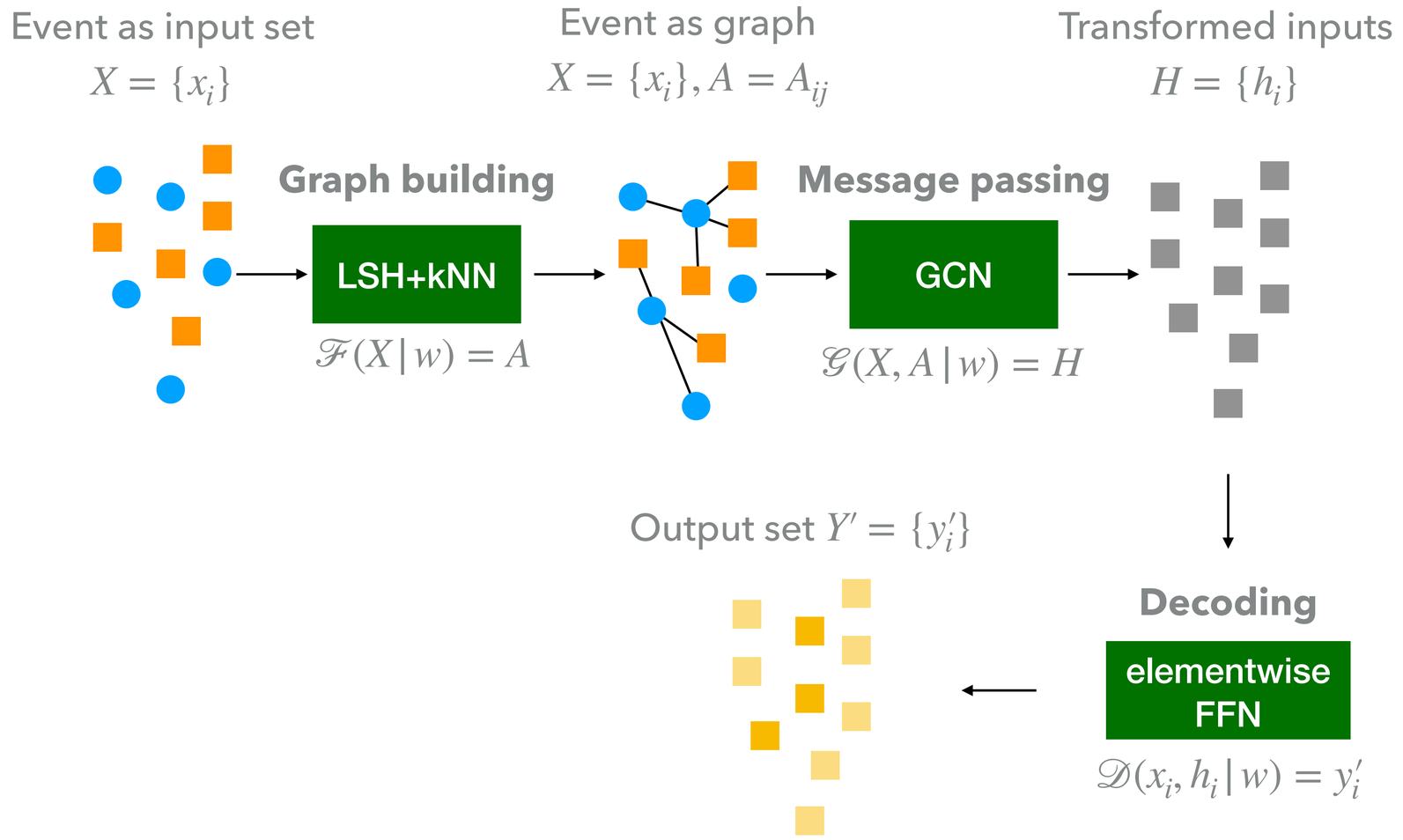


$h_i \in \mathbb{R}^N, N = 256$

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

GRAPH NEURAL NETWORK APPROACH

- Convert input set to a locally, sparsely connected graph
- Message-passing NN to transform features
- Decode transformed inputs elementwise

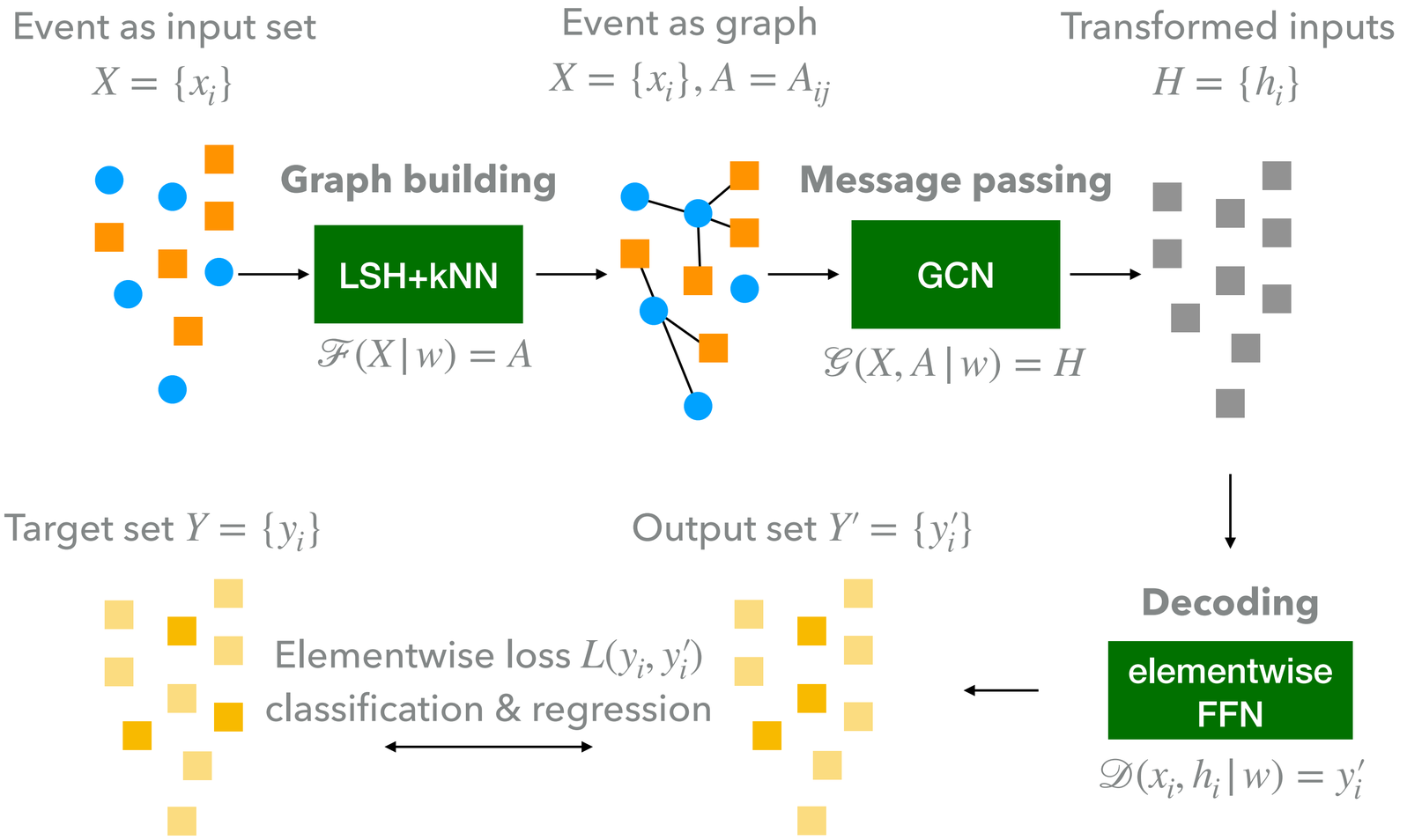


$h_i \in \mathbb{R}^N, N = 256$

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

GRAPH NEURAL NETWORK APPROACH

- Convert input set to a locally, sparsely connected graph
- Message-passing NN to transform features
- Decode transformed inputs elementwise
- (During training) Compare to target set, optimize weights

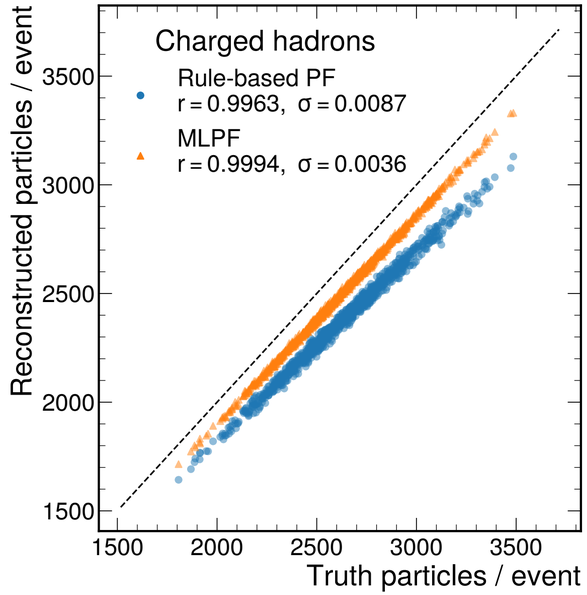


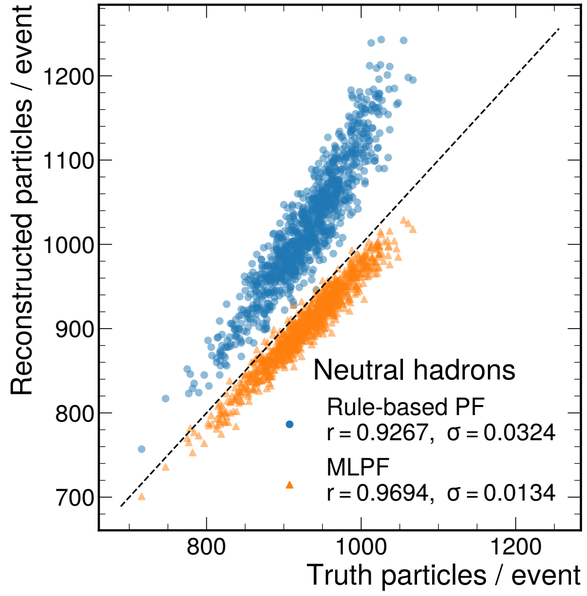
$h_i \in \mathbb{R}^N, N = 256$

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

CLASSIFICATION/IDENTIFICATION PERFORMANCE

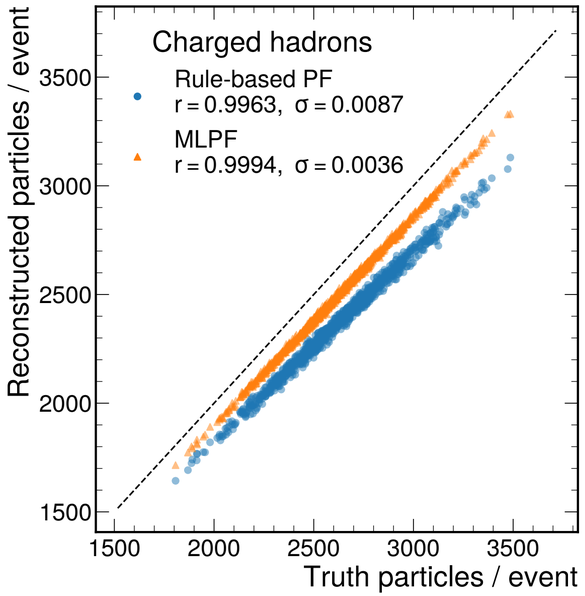
- Particle count prediction: good correlation with true particle count for charged and neutral hadrons

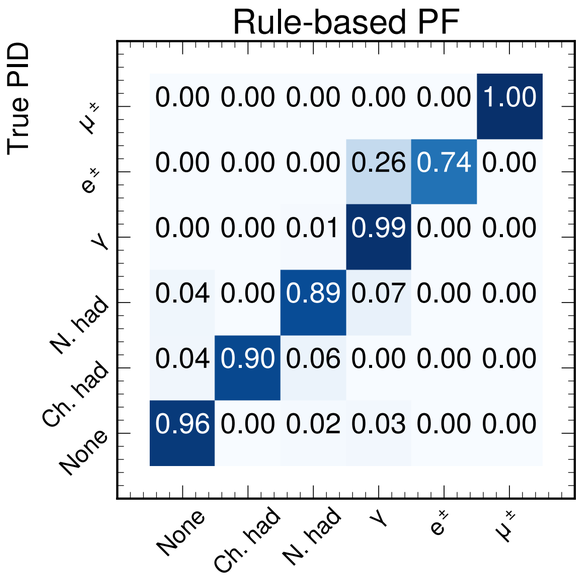


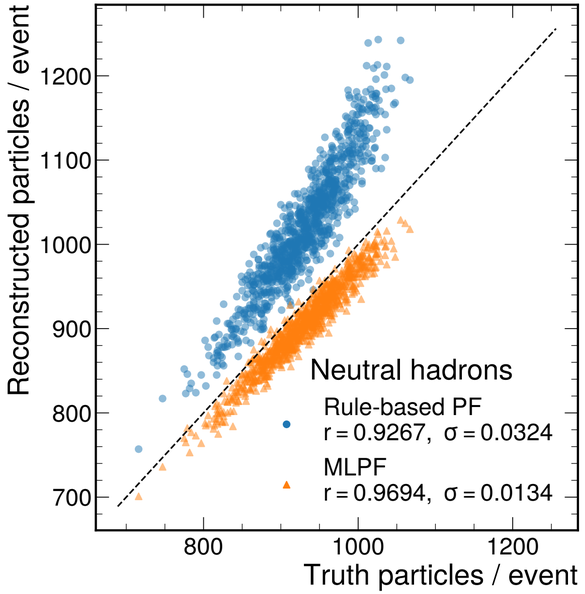


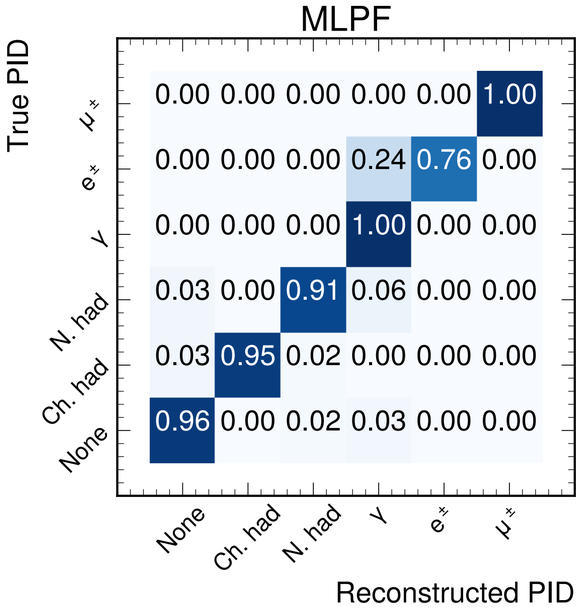
CLASSIFICATION/IDENTIFICATION PERFORMANCE

- Particle count prediction: good correlation with true particle count for charged and neutral hadrons
- Classification performance: MLPF same or better than rule-based PF algorithm



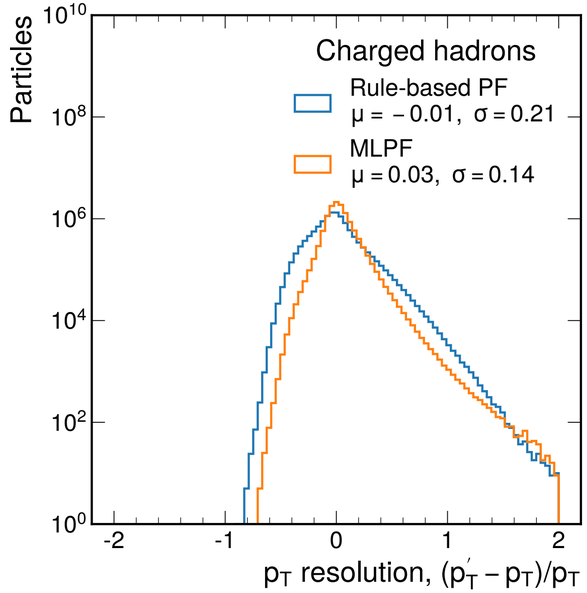


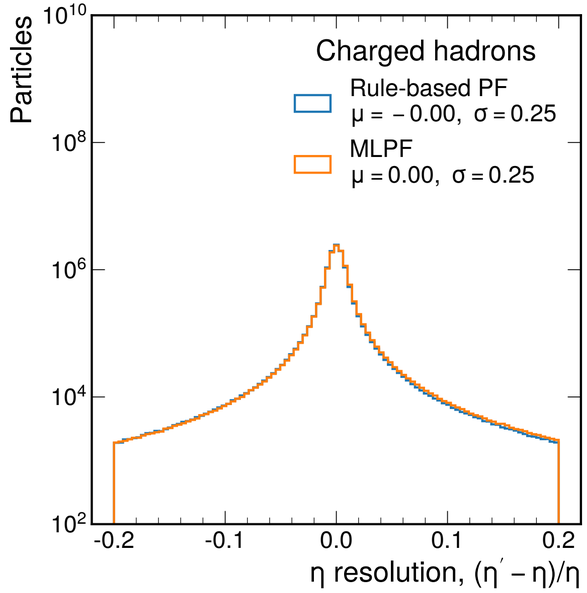




MOMENTUM REGRESSION PERFORMANCE (RESOLUTION)

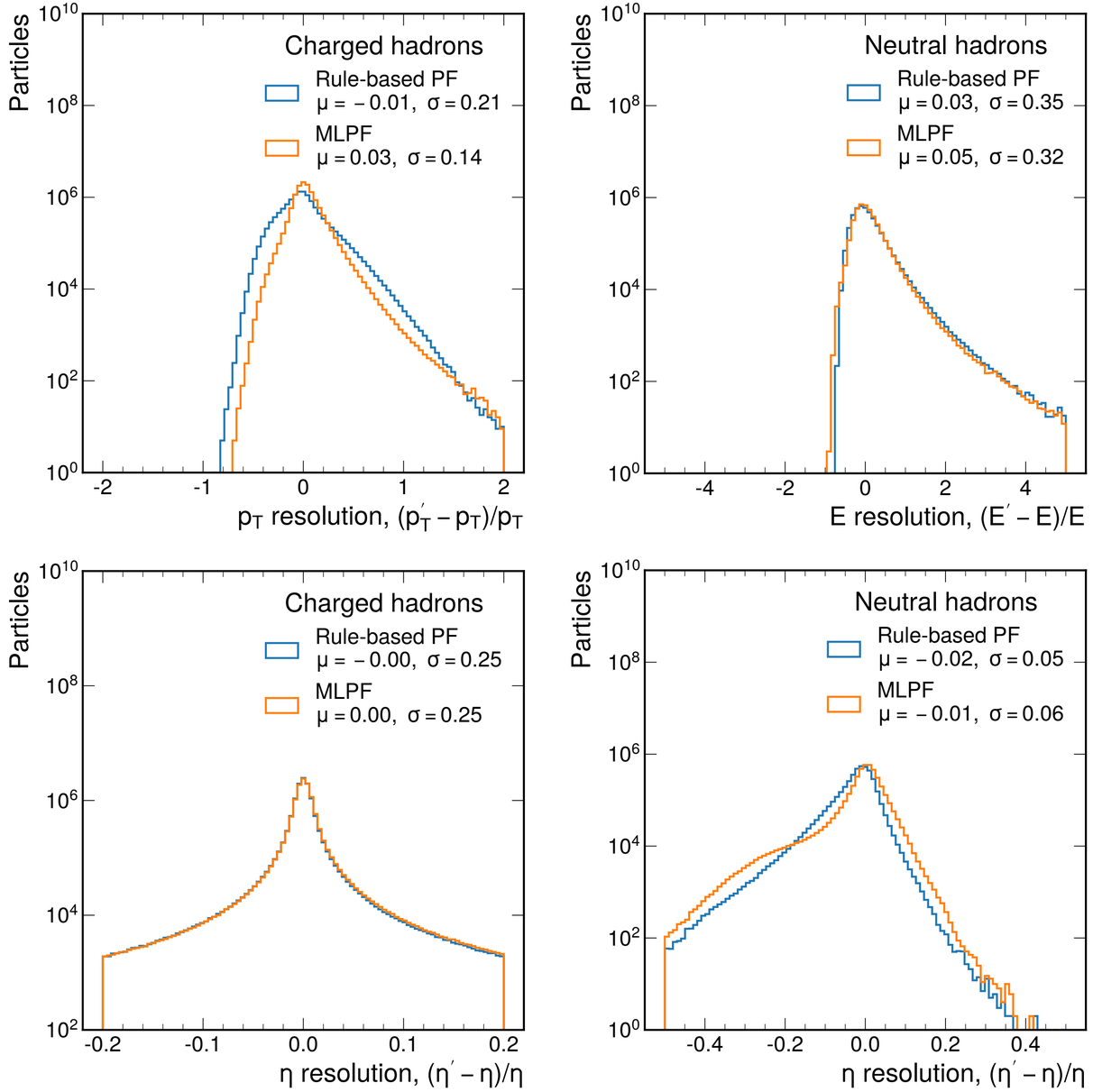
- Charged hadron regression: MLPF improves on p_{T} resolution and comparable η resolution





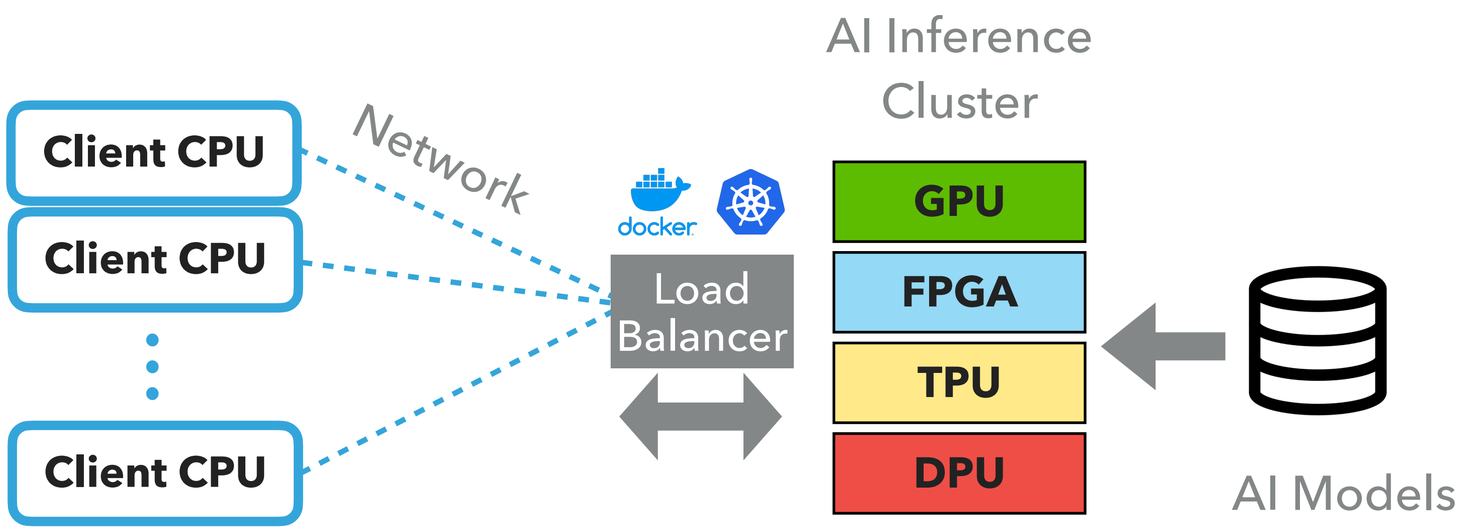
MOMENTUM REGRESSION PERFORMANCE (RESOLUTION)

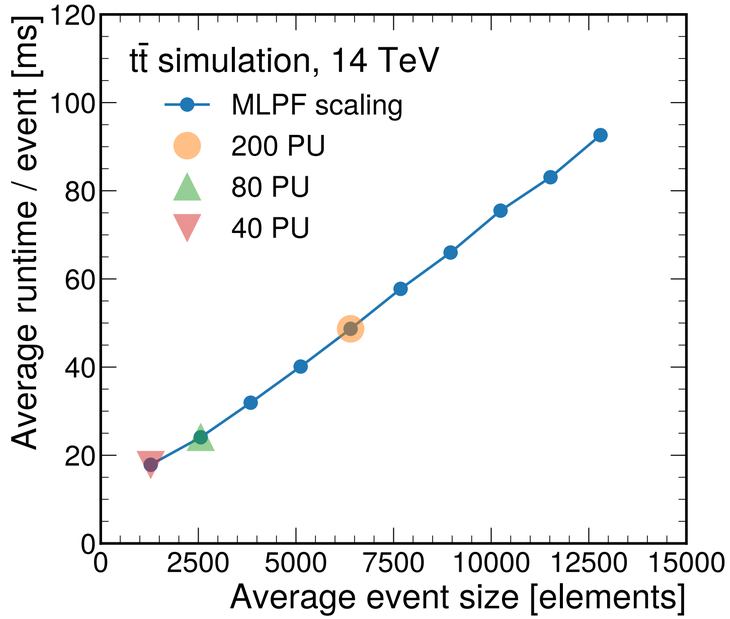
- Charged hadron regression: MLPF improves on p_{T} resolution and comparable η resolution
- Neutral hadron regression: MLPF improves on p_{T} resolution and comparable η resolution

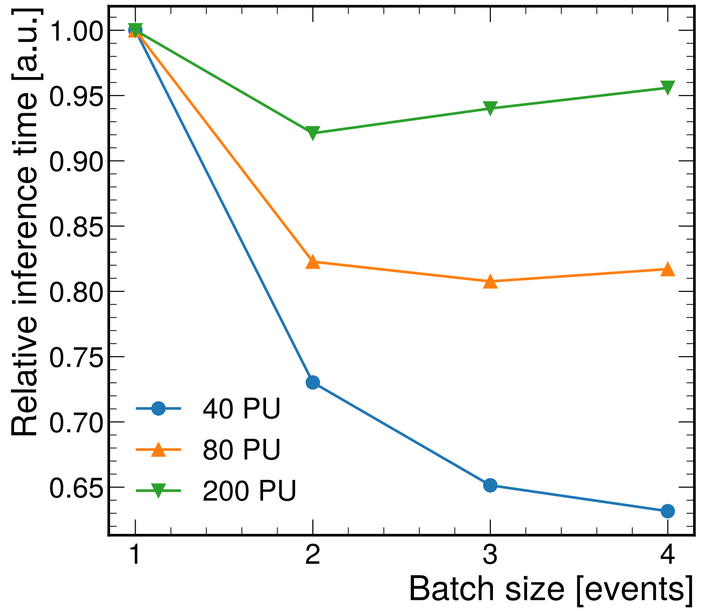


INFERENCE TIME

- Model inference time scales linearly with increasing pileup/event size!
- Increasing batch size also helps
- AI inference cluster can accelerate this



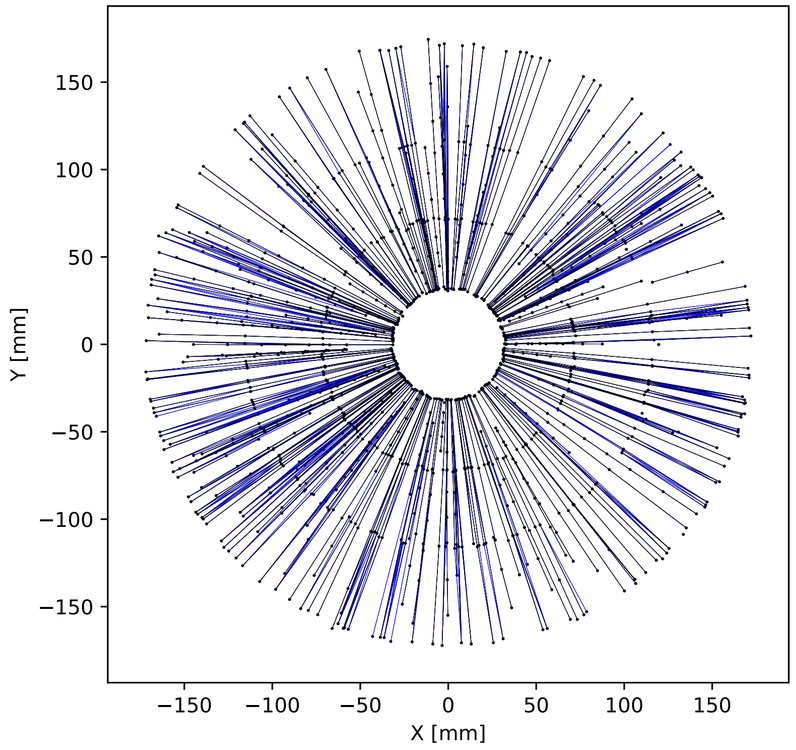


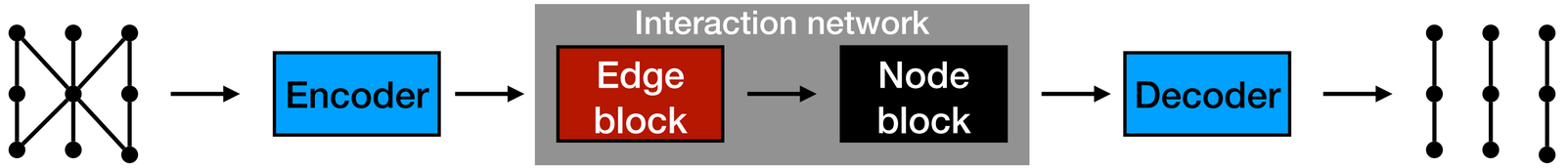


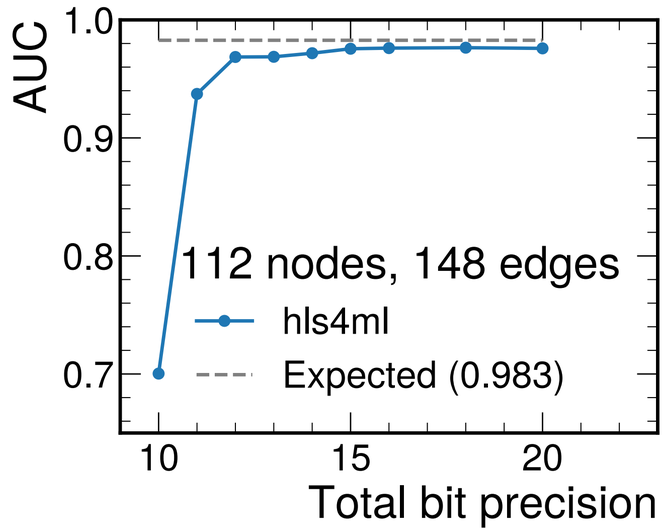
BEYOND

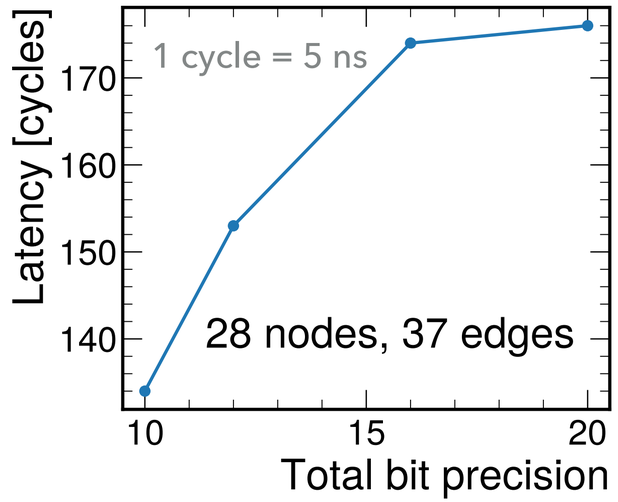
UCSD

GRAPH NEURAL NETWORKS ON FPGAS FOR PARTICLE TRACKING





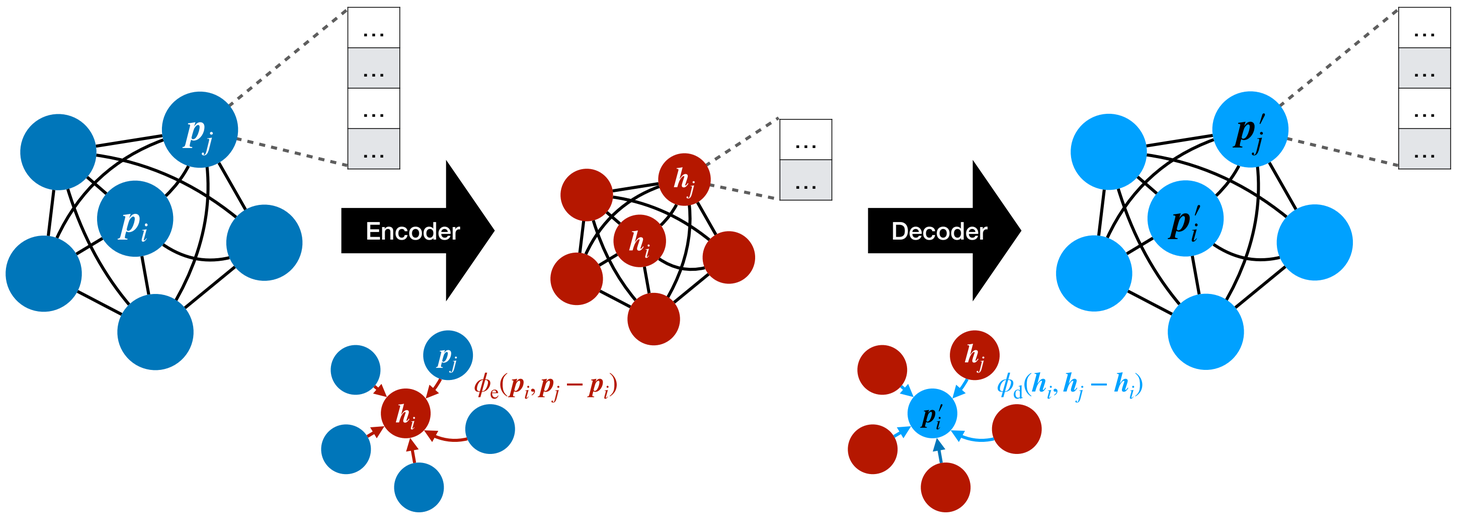


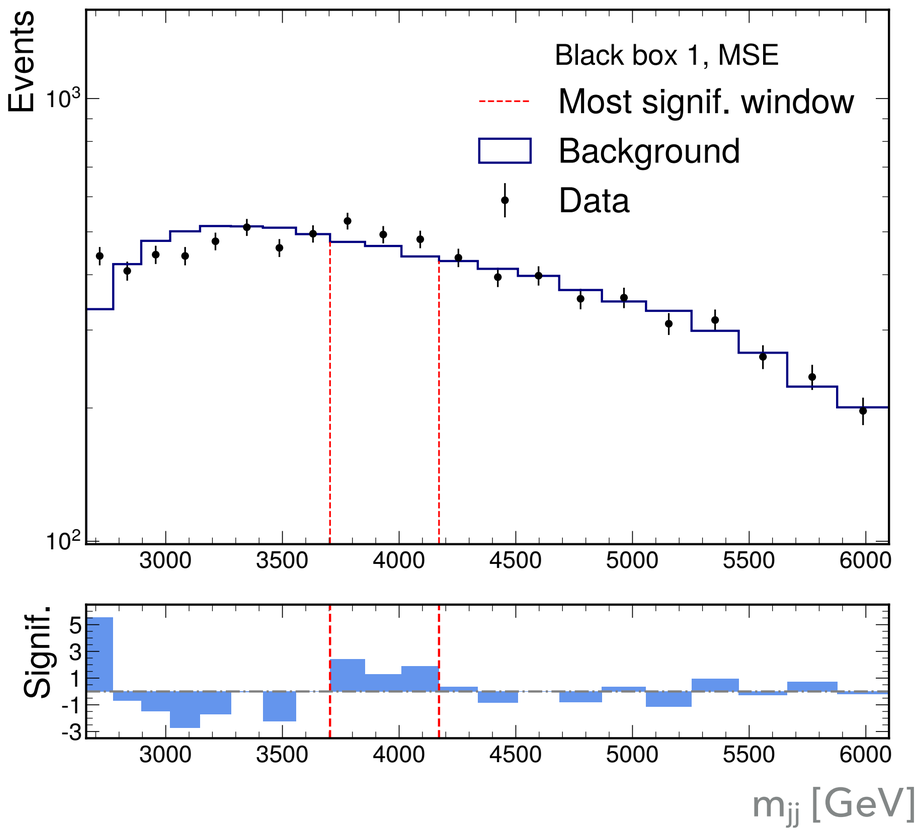


- Implemented a GNN on an FPGA for particle tracking on small sectors
- Found it can complete a tracking task in < 1 microsecond

PARTICLE GRAPH AUTOENCODER FOR ANOMALY DETECTION

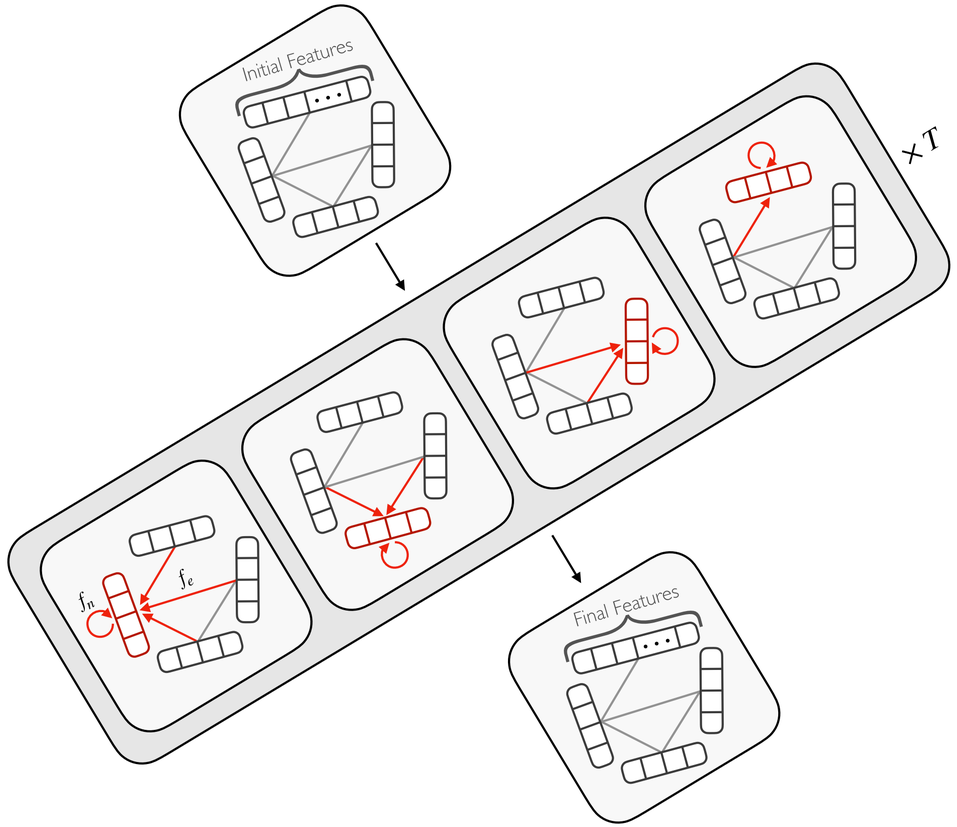
arXiv:2101.08320

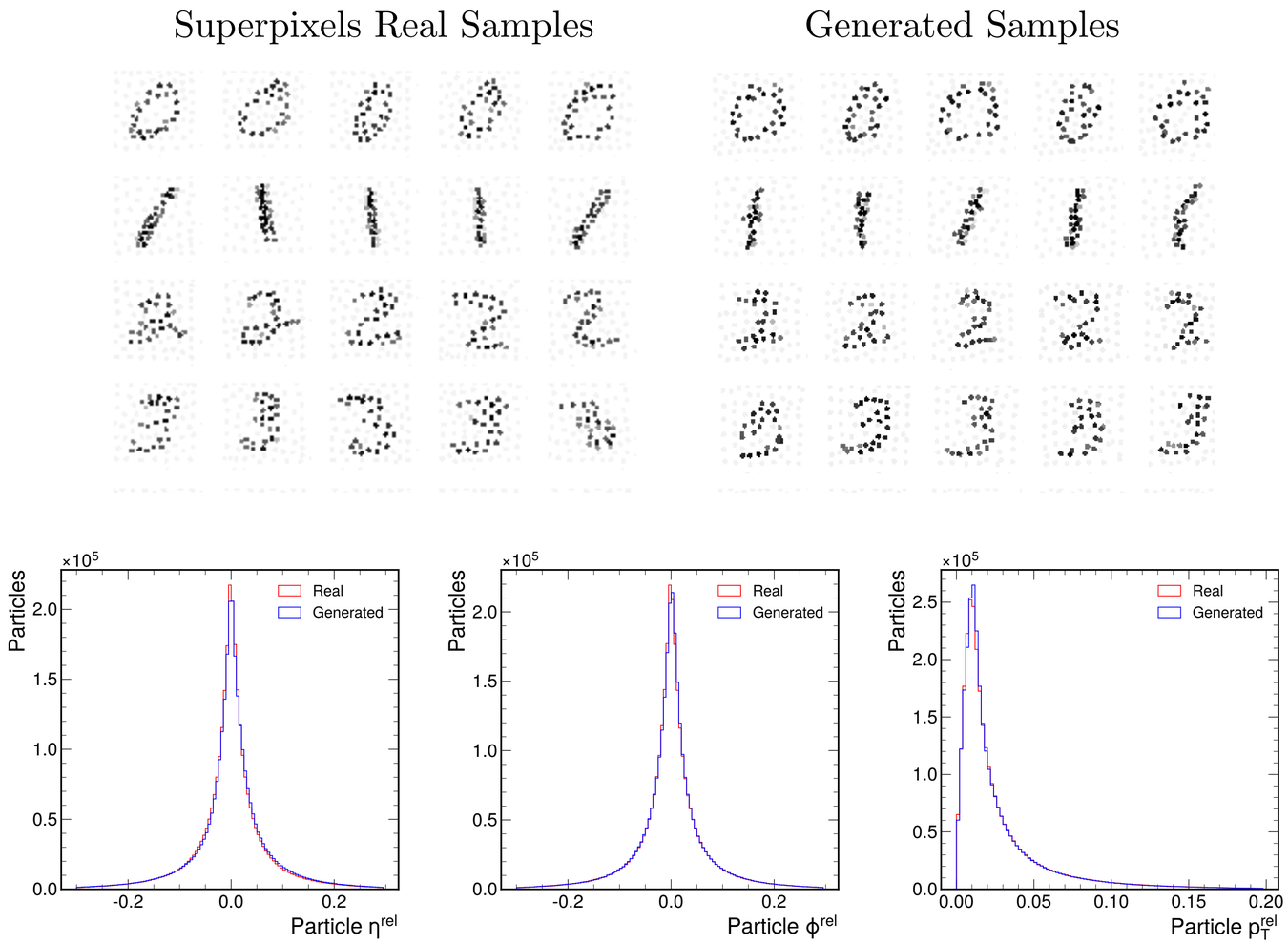




- Autoencoders compress data to a smaller representation and reconstruct it
  - Apply it to particles (E, $\boldsymbol{p}$): train autoencoder on known SM data
- Found to be effective on a mock dataset with a signal injected at 3.8 TeV

GRAPH GENERATIVE ADVERSARIAL NETWORK





- Graph-based generative adversarial network can be used to generate MNIST handwritten digit graphs and particle jet graphs

SUMMARY AND OUTLOOK

- GNNs are NNs that operate on graph-structured data
  - Well suited to HEP data due to its sparsity, irregular geometry, and intercorrelations

SUMMARY AND OUTLOOK

- GNNs are NNs that operate on graph-structured data
  - Well suited to HEP data due to its sparsity, irregular geometry, and intercorrelations
- GNNs are being developed and used for many different tasks in HEP from particle tracking to event classification

SUMMARY AND OUTLOOK

- GNNs are NNs that operate on graph-structured data
  - Well suited to HEP data due to its sparsity, irregular geometry, and intercorrelations
- GNNs are being developed and used for many different tasks in HEP from particle tracking to event classification
- Many new avenues for GNNs are also being explored:
  - End-to-end or “single-stage” inference
  - Unsupervised learning (like compression and anomaly detection)
  - Generative methods for sparse data
  - Accelerated inference in GPUs and FPGAs

REFERENCES

- GNNs:
  - Graph Neural Network Model Survey [doi:10.1109/TNN.2008.2005605]
  - Interaction Network [arXiv:1612.0222]
  - Geometric deep learning review [arXiv:1611.08097]
  - Message Passing NN [arXiv:1704.01212]
  - Dynamic Graph CNN [arXiv:1801.07829]
  - Graph Networks [arXiv:1806.01261]
  - Deep Sets [arXiv:1703.06114]
- Tools:
  - PyTorch Geometric [arXiv:1903.0242]
  - graph_nets [deepmind/graph_nets]
  - DGL [dgl.ai]
  - Spektral [danielegrattarola/spektral]
  - jraph [jraph]
- Physics:
  - Neural message passing for jets [nips_dlps_2017_29]
  - Energy Flow Networks [arXiv:1810.05165]
  - HEP.TrkX GNN [arXiv:1810.06111]
  - GarNet and GravNet [arXiv:1902.07987]
  - ParticleNet [arXiv:1902.08570]
  - JEDI-Net [arXiv:1908.05318]
  - Interaction networks for H(bb) decays [arXiv:1909.12285]
  - ABCNet [arXiv:2001.05311]
  - Object condensation [arXiv:2002.03605]
  - Exa.TrkX GNN [arXiv:2003.11603]
  - Two-point correlation GNN [arXiv:2003.11787]
  - MPNN for HH [arXiv:2005.11086]
  - Graph net for dark showers [arXiv:2006.08639]
  - GNNs in Particle Physics Review [arXiv:2007.13681]
  - Exa.TrkX track seeding and labelling [arXiv:2007.00149]
  - GarNet on FPGAs [arXiv:2008.03601]
  - Jet clustering [arXiv:2008.06064]

JAVIER DUARTE

MIT 8.S50

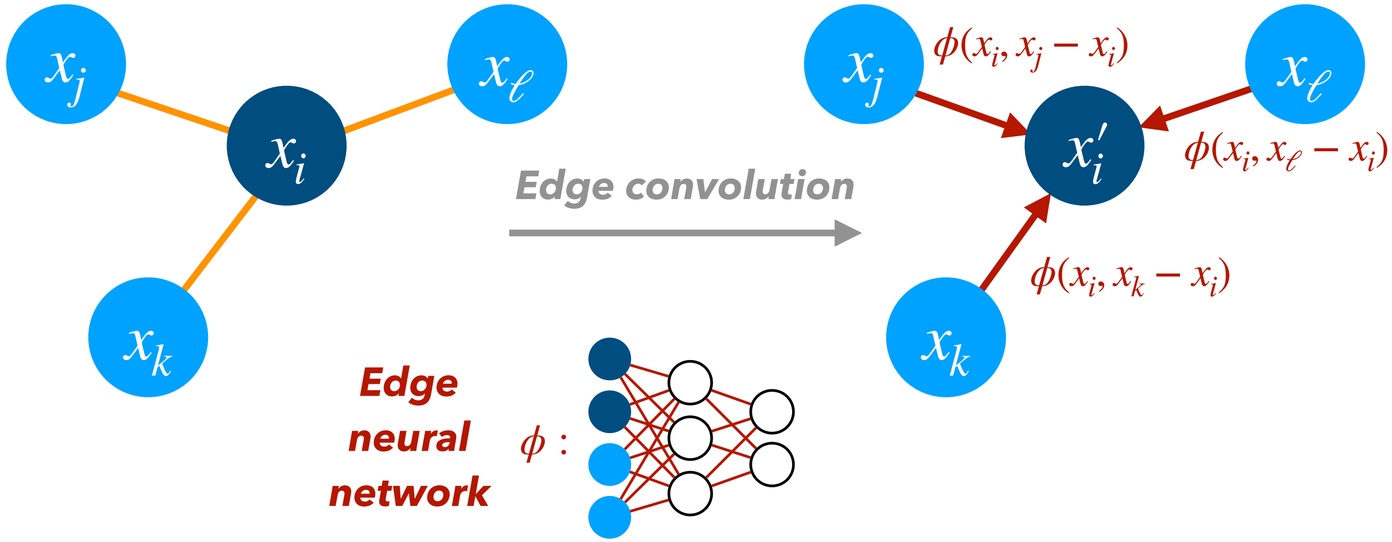
JANUARY 27, 2021

BACKUP

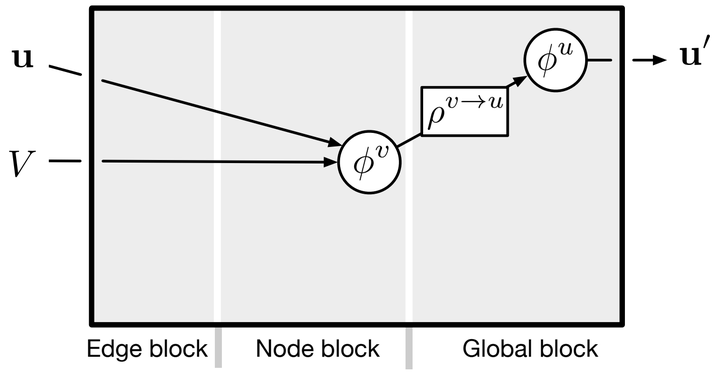
UCSD

CONNECTIONS TO OTHER GNNS

- Edge convolution from DGCNN [arXiv:1801.07829] is a variant of the edge block step (basis of ParticleNet [arXiv:1902.08570])



- Deep Sets [arXiv:1703.06114] does not consider edge features (basis of Energy Flow Network [arXiv:1810.05165])



- Interaction network [arXiv:1612.00222] ignores global features (basis of JEDI-Net, H(bb) tagger [arXiv:1908.05318, arXiv:1909.12285], Exa.TrkX GNN [arXiv:2003.11603])