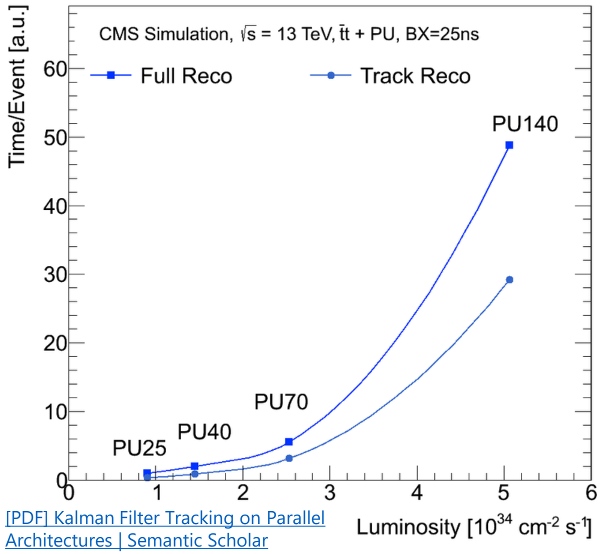
OBJECT CONDENSATION

FOR GNN-BASED PARTICLE TRACKING

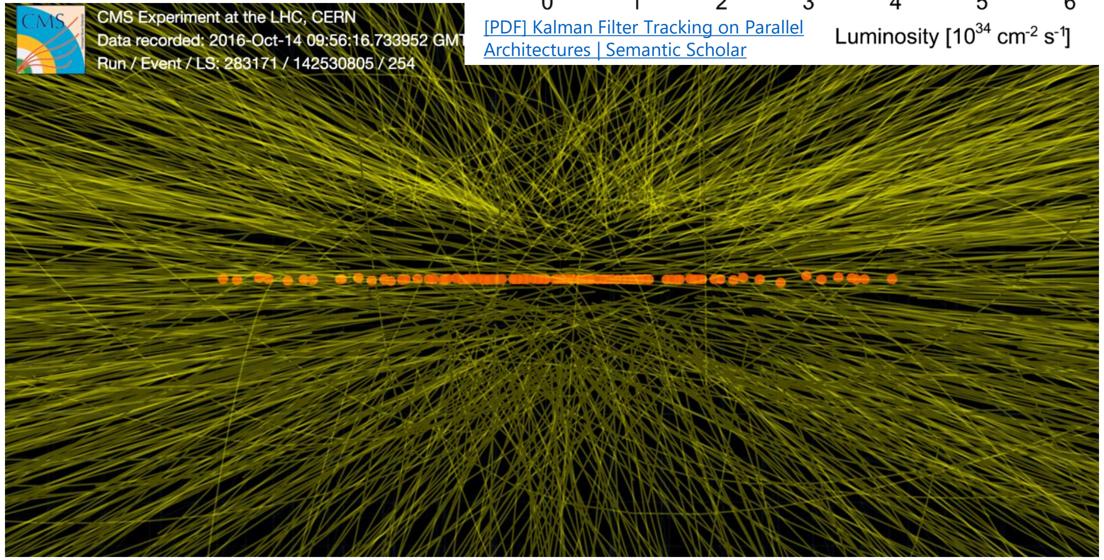
GAGE DEZOORT

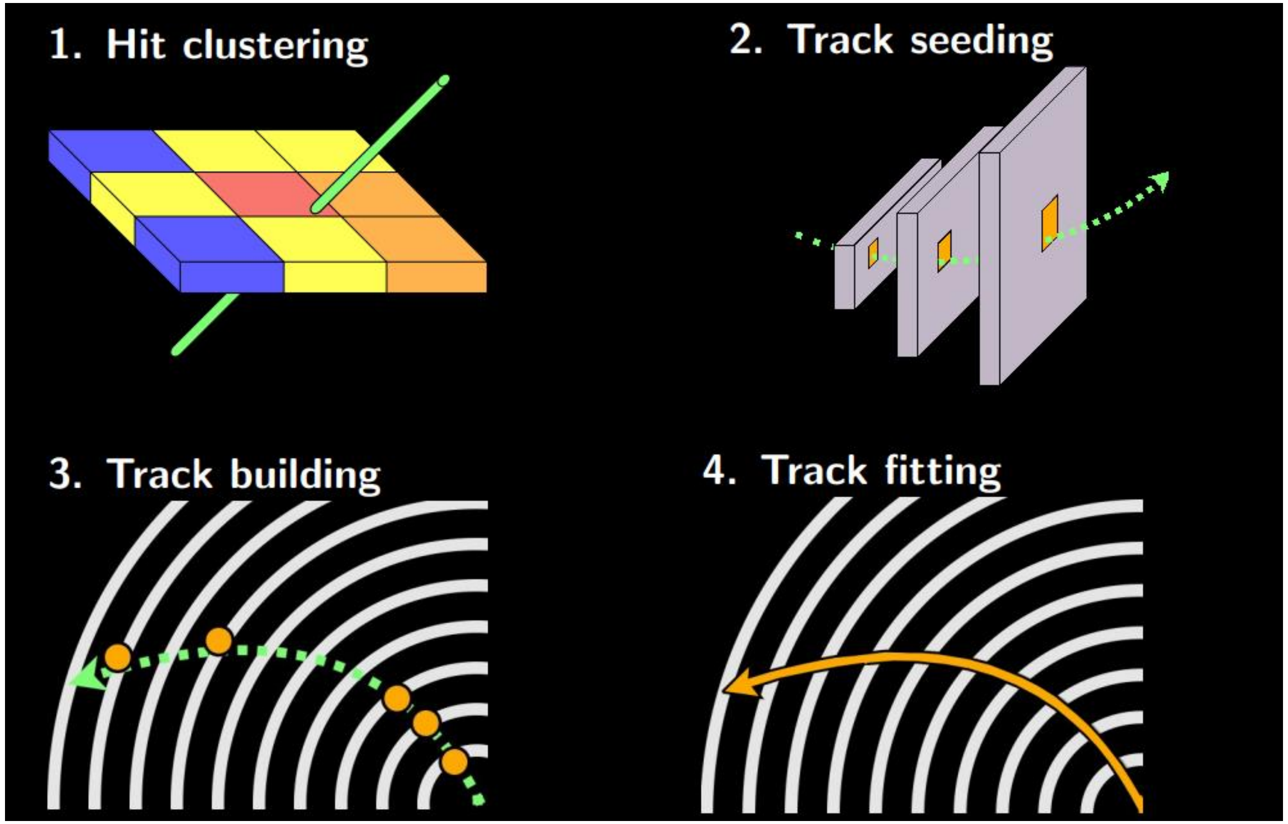
04/27/2022

Traditional tracking algorithms scale worse-than-linearly with increasing pileup



[PDF] Kalman Filter Tracking on Parallel Architectures | Semantic Scholar



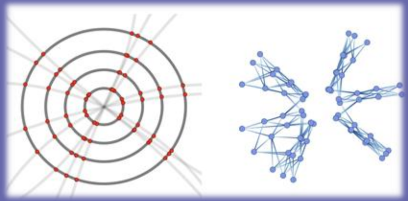


Iterative tracking algorithms are based on the combinatorial Kalman filter (CKF), iteratively extending and fitting tracks from an initial seed

Connecting the dots: applying deep learning techniques in HEP | EP News (cern.ch)

PARTICLE TRACKING

TRADITIONAL TRACKING METHODS

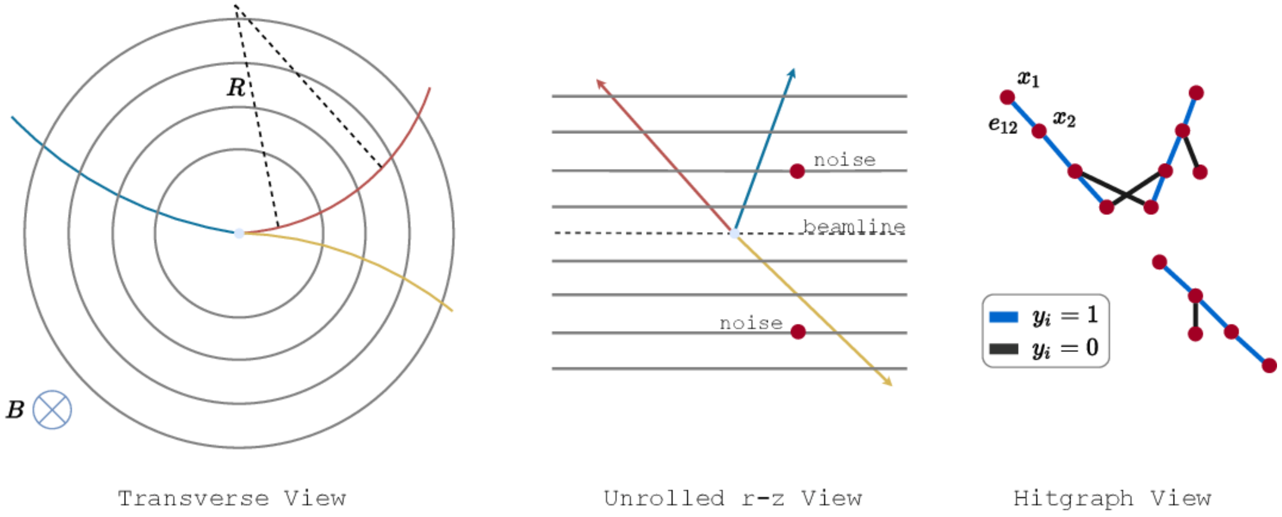


GNN TRACKING

EDGE CLASSIFICATION PARADIGM

Edge Classification Task

- Draw edges to hypothesize various particle trajectories, train a GNN to classify edges

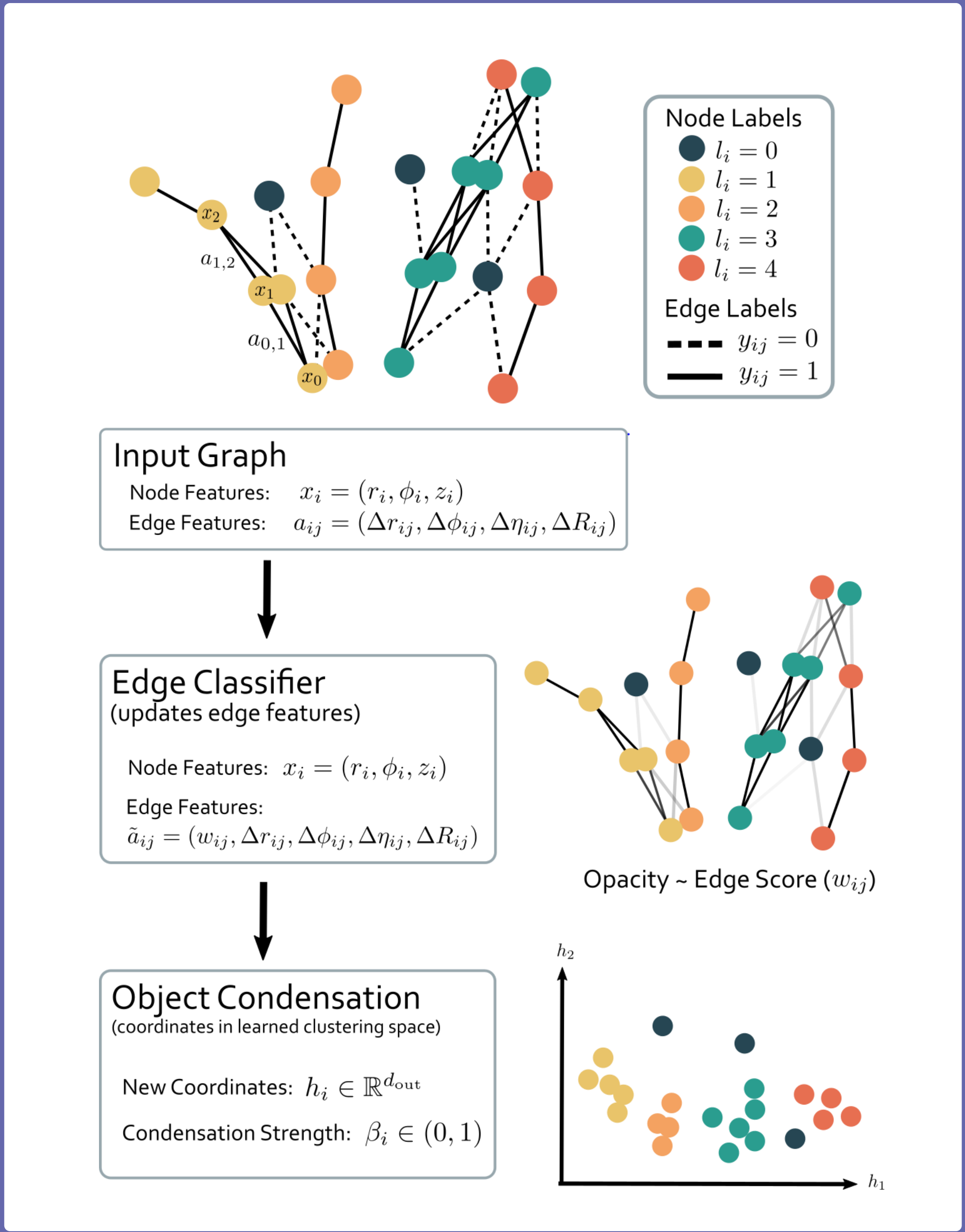


- Use edge weights to produce tracks (i.e. apply a threshold to produce disjointed subgraphs)

- **Key steps** (general to many GNN workflows)
  1) Graph construction from underlying data
  2) GNN inference
  3) Post-processing of GNN predictions

EDGE CLASSIFICATION / OBJECT CONDENSATION

STRATEGY OVERVIEW



GRAPH NEURAL NETWORKS
NEURAL MESSAGE PASSING

Message Passing (MPNN) Layers:
Framework for equivariant graph updates

At each layer *k*, compute messages in each node's neighborhood:

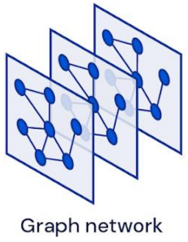
$$\boldsymbol{m}_{uv}^{(k)} = \psi^{(k)}\left(\boldsymbol{h}_u^{(k-1)}, \boldsymbol{h}_v^{(k-1)}, \boldsymbol{e}_{uv}^{(k-1)}\right)$$

Aggregate messages in a permutation-invariant way:

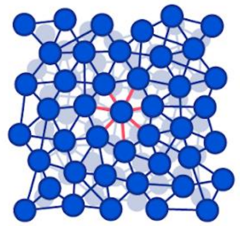
$$\boldsymbol{a}_u^{(k)} = \bigoplus_{v \in N(u)} \boldsymbol{m}_{uv}^{(k)}$$

Update the node's state based on the messages it received:

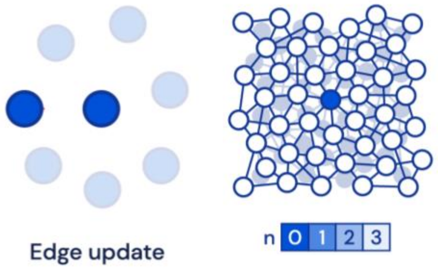
$$\boldsymbol{h}_u^{(k)} = \underbrace{\phi^{(k)}}_{\text{MLP}}\left(\boldsymbol{h}_u^{(k-1)}, \boldsymbol{a}_u^{(k)}\right)$$



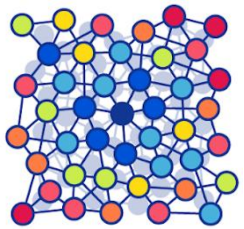
GNN comprised of multiple message passing layers



Input graph



Neural Message Passing



New graph embedding

Figure Source: https://deepmind.com/blog/article/Towards-understanding-glasses-with-graph-neural-networks

GRAPH NEURAL NETWORKS

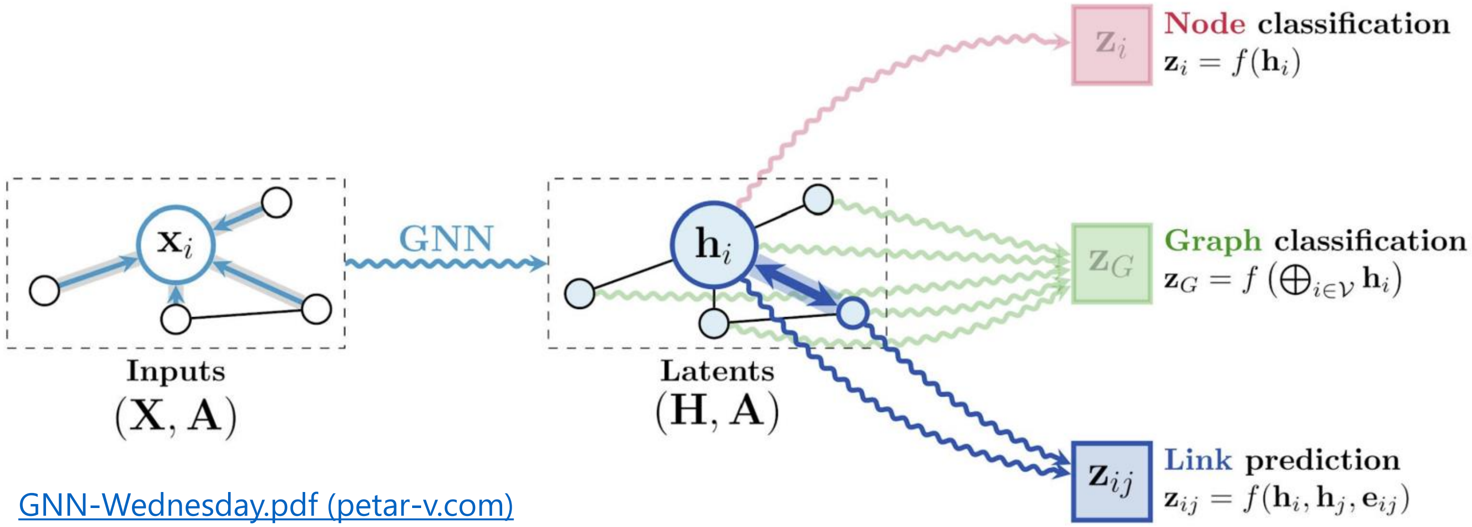
REPEATED MESSAGE PASSING

Generic MPNN Layers:

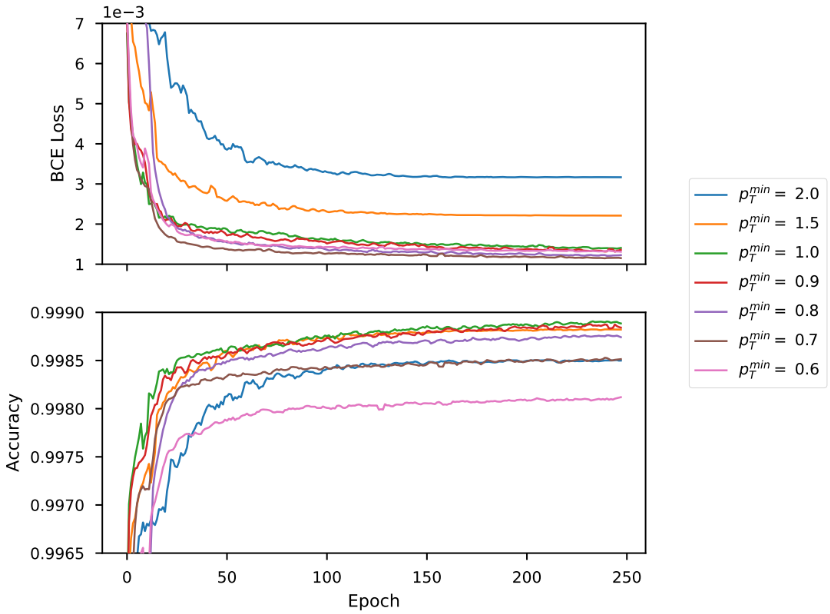
$$\boldsymbol{h}_u^{(k)} = \phi^{(k)}\left[\boldsymbol{h}_u^{(k-1)}, \bigoplus_{v \in N(u)} \psi^{(k)}\left(\boldsymbol{h}_u^{(k-1)}, \boldsymbol{h}_v^{(k-1)}, \boldsymbol{e}_{uv}^{(k-1)}\right)\right]$$

Node Updates: collecting info from each node's *k*-hop neighborhood at the k^{th} layer

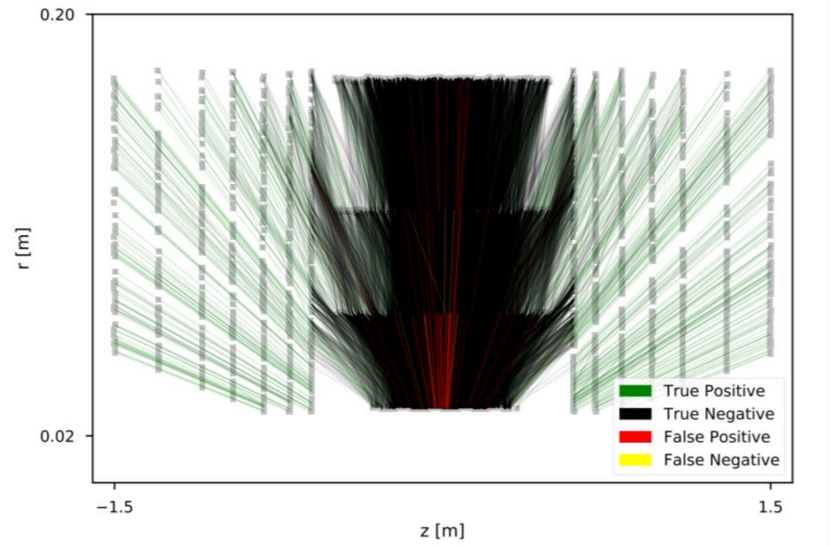
Outputs: node-level, edge-level, or graph-level predictions



GNN-Wednesday.pdf (petar-v.com)



loss/accuracy training curves on a range of graph sizes



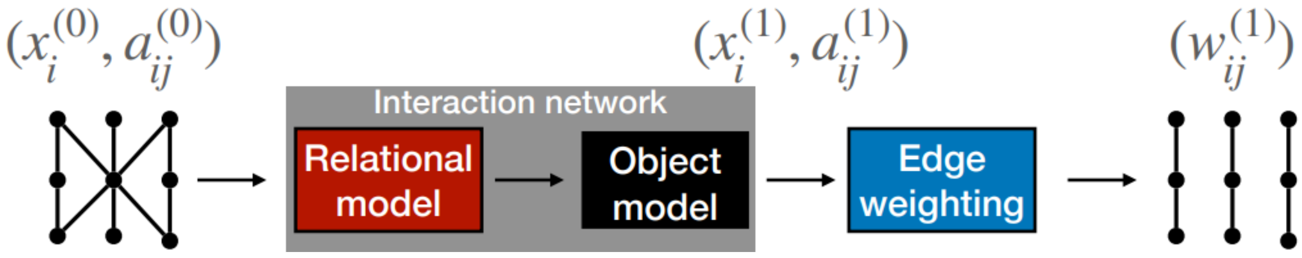
edge classification performance on a single graph

Interaction Networks:

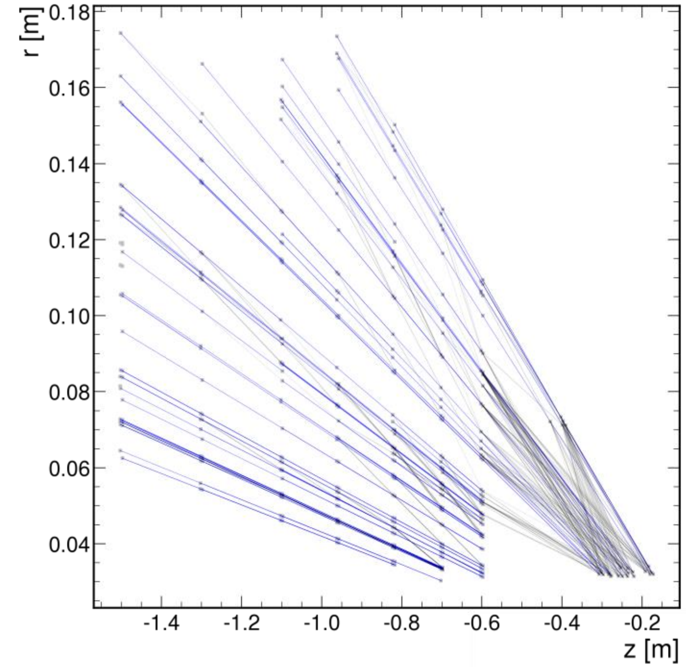
[1612.00222] Interaction Networks for Learning about Objects, Relations and Physics (arxiv.org)

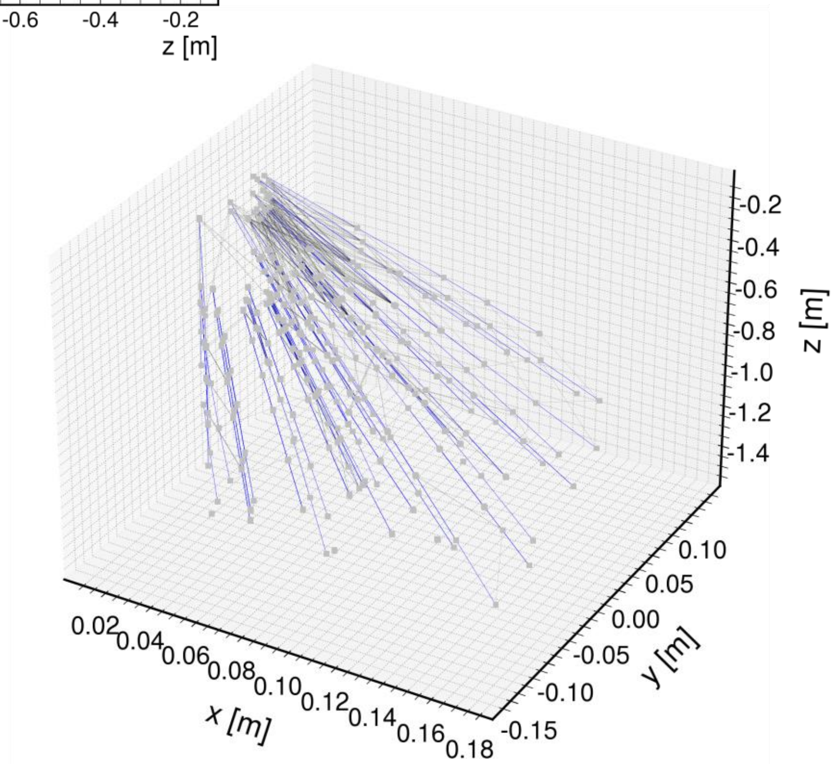
Even a single interaction network layer (depth-1 GNN) can achieve excellent edge classification accuracy

- (**Edge Block**) compute an interaction between two entities
- (**Node Block**) use the interaction to update the state of the receiving node



simple architecture explored in 2103.16701.pdf (arxiv.org)

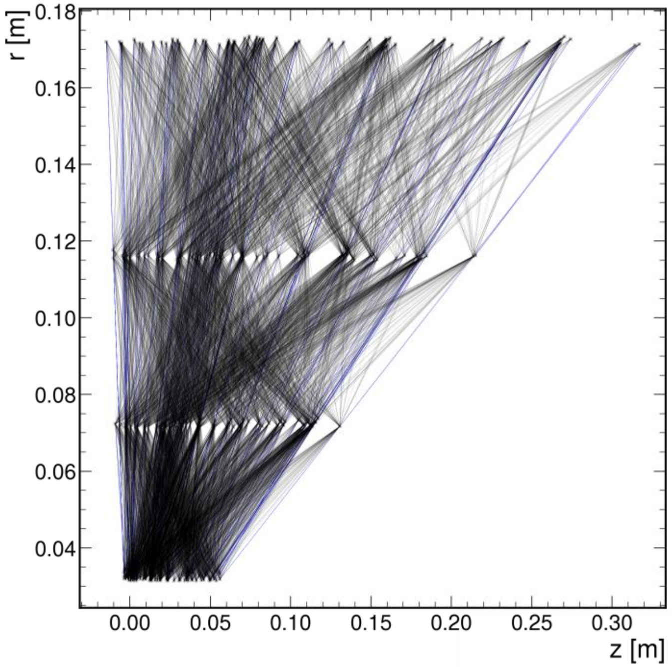


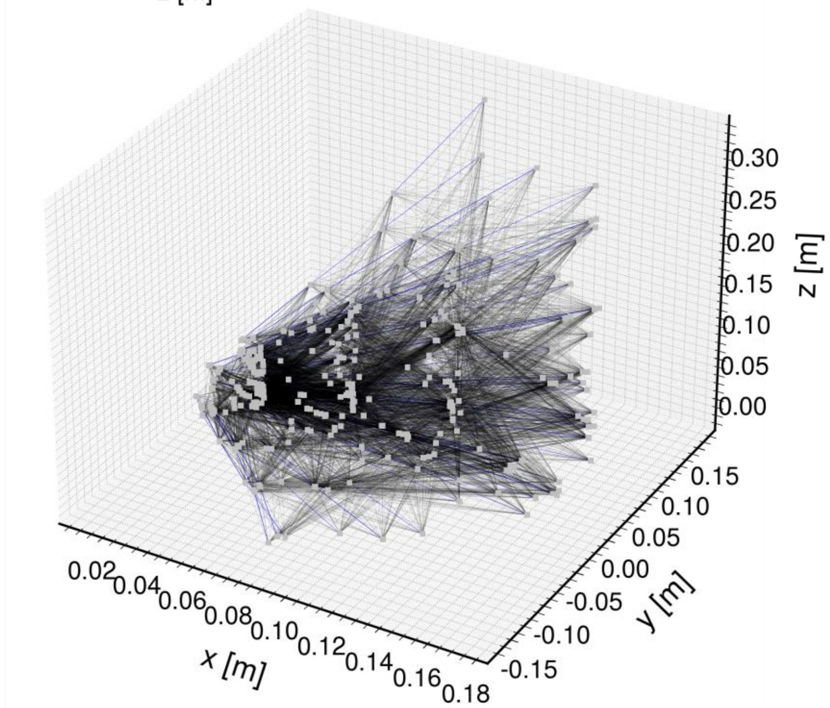


GRAPH CONSTRUCTION

EACH EVENT BROKEN (8X8) PHI-ETA SECTORS

- Truth cuts
  - *track p_T > 1.0 GeV*
  - *remove_noise: true*
- Geometric edge selections:
  - phi_slope < 0.007
  - z0 < 350 mm
  - n_phi_sectors: 8
  - n_eta_sectors: 8
  - phi sector overlap: 0.08
  - eta sector overlap: 0.125





GRAPH CONSTRUCTION

EACH EVENT BROKEN (8X8) PHI-ETA SECTORS

- Truth cuts
  - *track p_T > 1.0 GeV*
  - *remove_noise: true*
- Geometric edge selections:
  - phi_slope < 0.007
  - z0 < 350 mm
  - n_phi_sectors: 8
  - n_eta_sectors: 8
  - phi sector overlap: 0.08
  - eta sector overlap: 0.125

Edge Classifier
(updates edge features)

Node Features: $x_i = (r_i, \phi_i, z_i)$

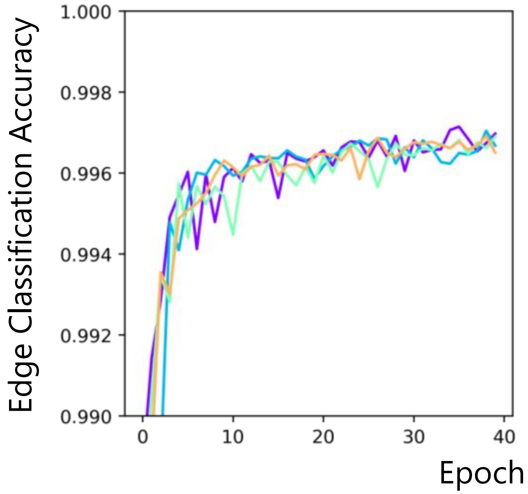
Edge Features:
$\tilde{a}_{ij} = (w_{ij}, \Delta r_{ij}, \Delta\phi_{ij}, \Delta\eta_{ij}, \Delta R_{ij})$

Opacity ~ Edge Score (w_{ij})

- BCE as usual to learn optimal edge weights

$$\mathcal{L}_w(y_j, w_j) = -\sum_{j=1}^{|\mathcal{E}|} \left(y_j \log w_j + (1-y_j)\log(1-w_j)\right)$$

Edge weights converge to high accuracy at intermediate stages of the GNN



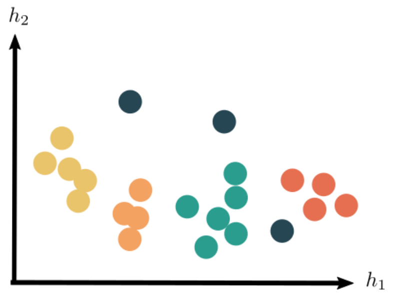
EDGE CLASSIFICATION
BINARY CROSS ENTROPY

OBJECT CONDENSATION
POTENTIAL LOSS + BACKGROUND SUPPRESSION

Object Condensation
(coordinates in learned clustering space)

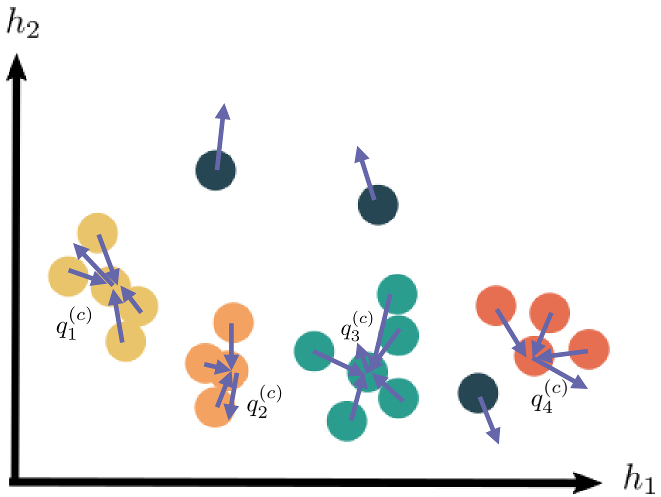
New Coordinates: $h_i \in \mathbb{R}^{d_{\text{out}}}$

Condensation Strength: $\beta_i \in (0, 1)$



- **Predict** condensation "likelihood" ($\beta_i \in (0,1)$) and learned clustering coordinates ($h_i \in \mathbb{R}^{d_h}$)
- At truth level, use charge likelihoods to find "most likely" condensation point per track
- Attract hits belonging to the same track, repulse others (including noise)

Nodes are attracted to their particle's condensation point and repulsed from other particles'

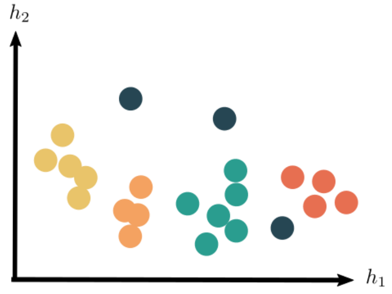


OBJECT CONDENSATION
TOTAL LOSS

Object Condensation
(coordinates in learned clustering space)

New Coordinates: $h_i \in \mathbb{R}^{d_{\text{out}}}$

Condensation Strength: $\beta_i \in (0, 1)$

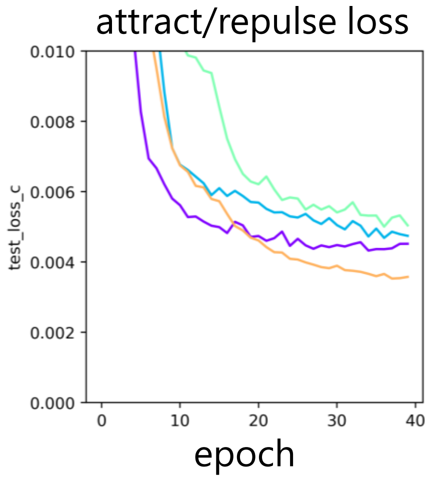


Edge Classification

$$\mathcal{L}_w(y_j, w_j) = -\sum_{j=1}^{|\mathcal{E}|} \left(y_j \log w_j + (1 - y_j)\log(1 - w_j)\right)$$

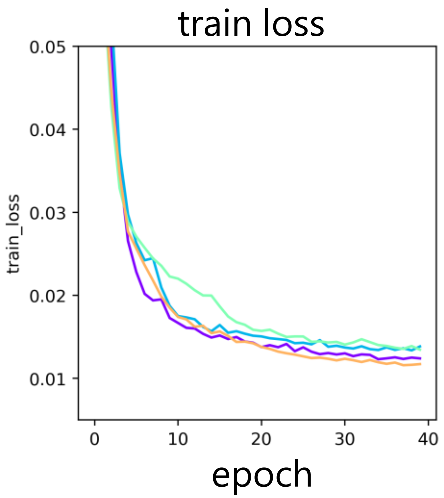
Attraction/Repulsion

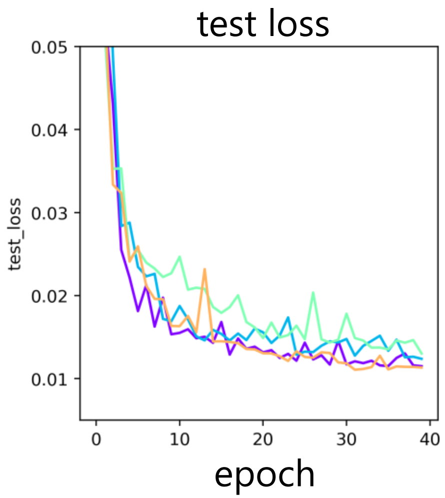
$$\mathcal{L}_V = \frac{1}{|\mathcal{V}|}\sum_{i=1}^{|\mathcal{V}|} q_i \sum_{k=1}^{K}\left(\underbrace{\mathbb{1}_{(l_i=k)} V_k^{\text{attract}}(h_i)}_{\text{(quadratic)}} + \underbrace{(1 - \mathbb{1}_{(l_i=k)}) V_k^{\text{repulse}}(h_i)}_{\text{(hinge)}}\right)$$



Background Suppression → scale by 2.5×10^{-3}

$$\mathcal{L}_\beta = \frac{1}{K}\sum_k (1 - \beta_k^{(c)}) + s_B \frac{\sum_{i=1}^{|\mathcal{V}|} \beta_i \mathbb{1}_{\{l_i=0\}}}{\sum_{i=1}^{|\mathcal{V}|} \mathbb{1}_{\{l_i=0\}}}$$

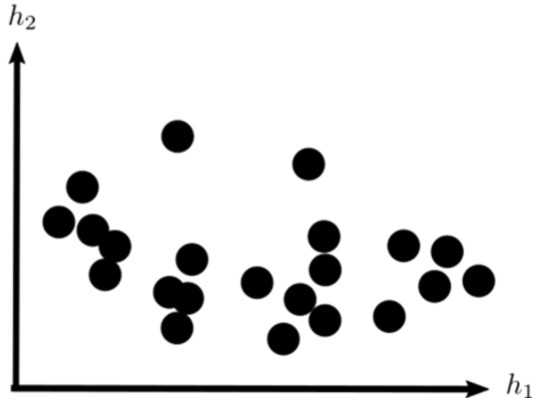




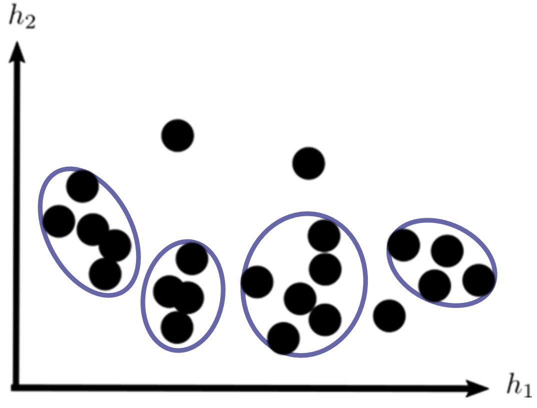
POSTPROCESSING

DBSCAN → TRACK FINDING

- GNN output is the set of hit coordinates in the learned (h_1, h_2) space:



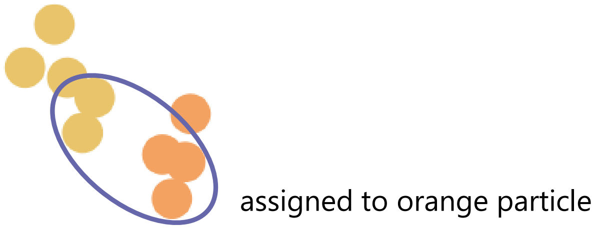
- Need to run DBSCAN to generate cluster labels (clustering parameters are optimized per on graph sector):



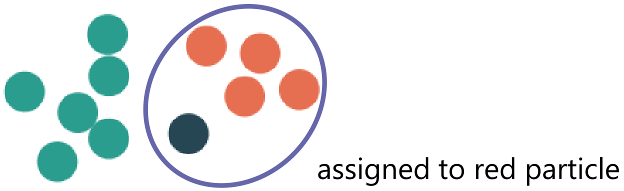
TRACKING EFFICIENCIES
VARIOUS DEFINITIONS

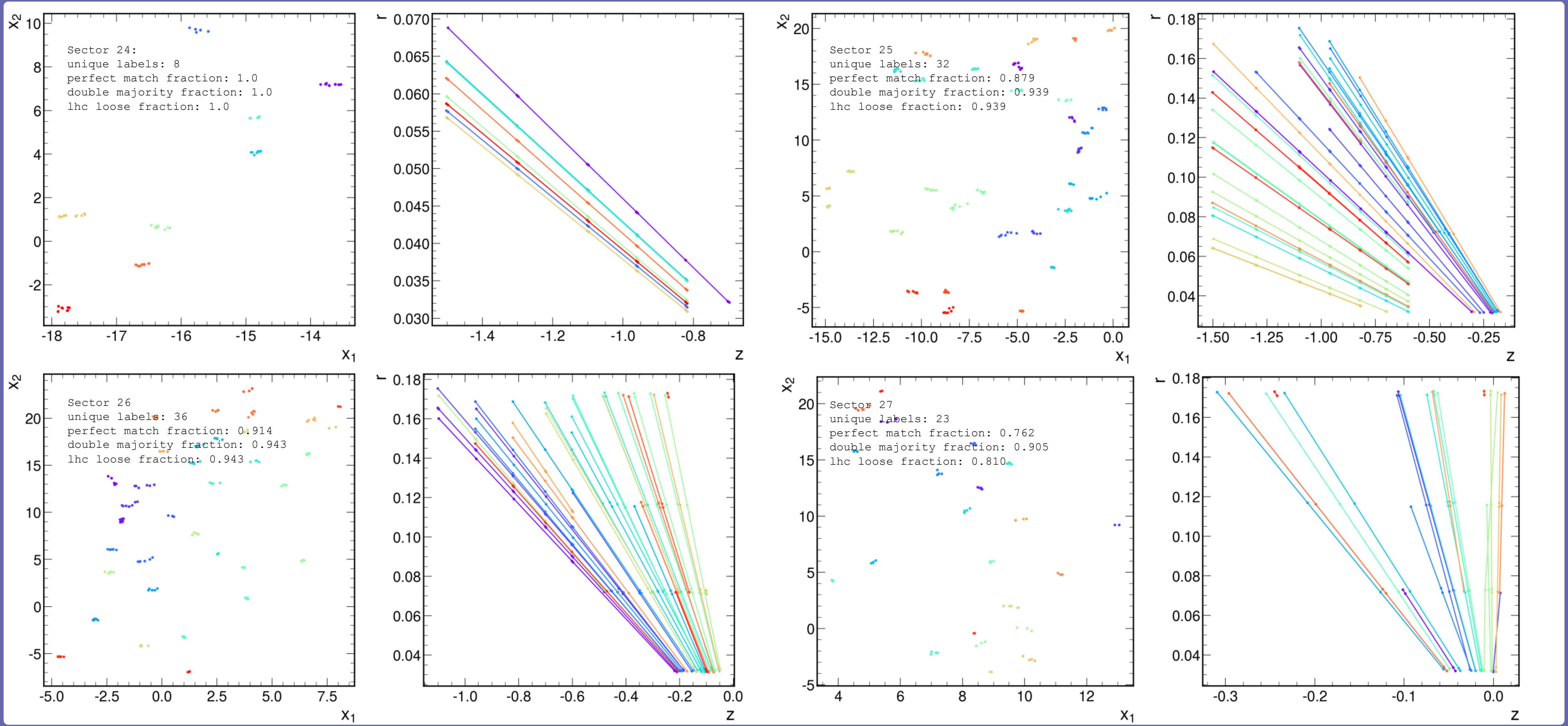
- Perfect Match: fraction of clusters containing every hit associated to a particle and no others

- Double Majority: fraction of clusters comprised of >50% of same-particle hits and containing >50% of that particle's hits



- LHC Loose Match: fraction of clusters comprised of >75% same-particle hits



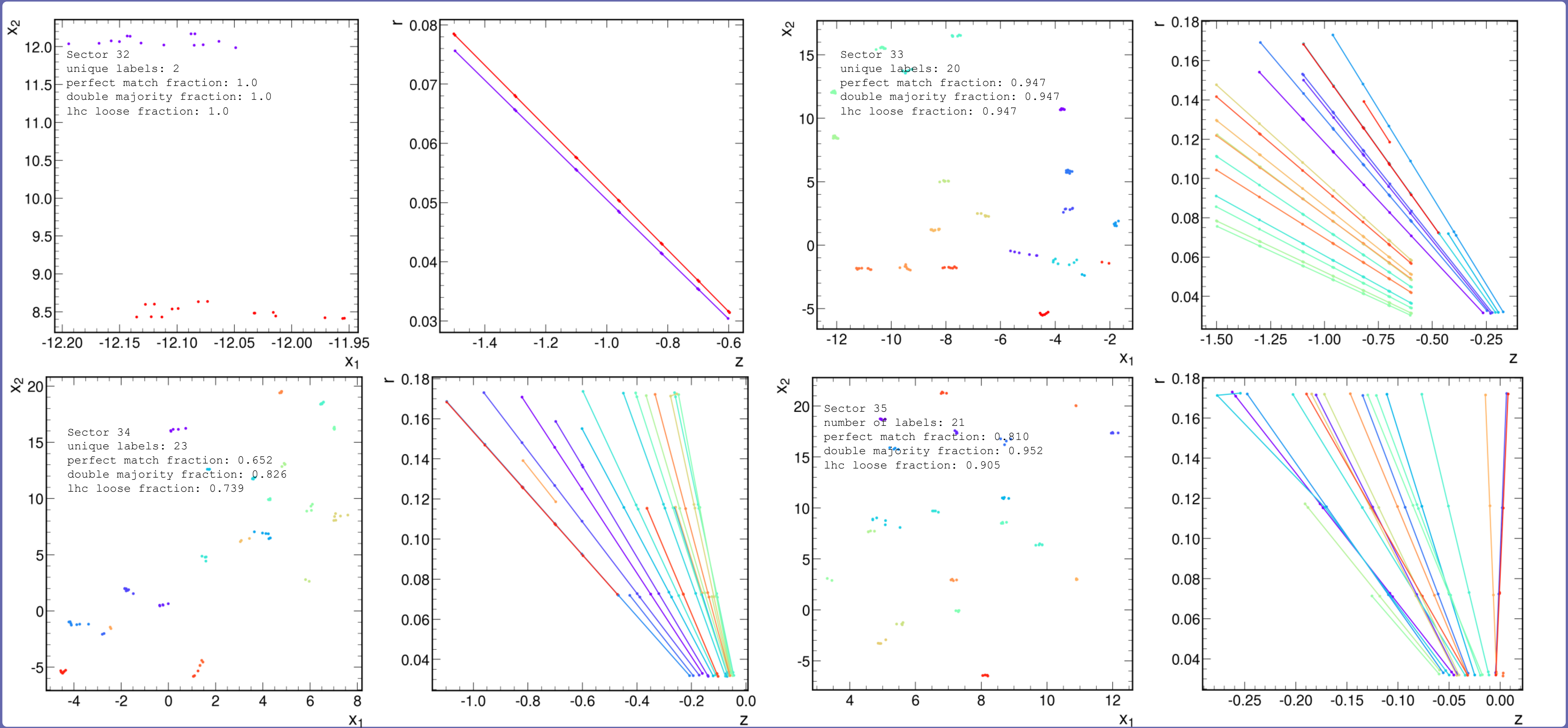


EXAMPLE: EVENT #1127

MODEL 10

```
summary of full event
perfect match fraction: 0.862
double majority fraction: 0.945
lhc loose fraction: 0.906
```

NOTE: Cluster colors are DBSCAN labels, not truth labels!



EXAMPLE: EVENT #1823

MODEL 10

```
summary of full event
perfect match fraction: 0.860
double majority fraction: 0.939
lhc loose fraction: 0.909
```

NOTE: Cluster colors are DBSCAN labels, not truth labels!

TRACKING EFFICIENCIES

AVERAGED ACROSS ~10^4 GRAPHS

- Per-graph summary
  - Perfect Match Fraction: 0.827
  - Double Majority Fraction: 0.932
  - LHC Loose Fraction: 0.890

- Per eta-range:
  - Performance decreases with graph construction purity (decreasing eta)

$\|\eta\|$	LHC Loose Match	Double Majority	Perfect Match	Fake Fraction
$(0, 1.25)$	0.851 +/- 0.070	0.905 +/- 0.058	0.779 +/- 0.099	0.091 +/- 0.072
$(1.25, 2.5)$	0.895 +/- 0.062	0.934 +/- 0.051	0.842 +/- 0.087	0.071 +/- 0.065
$(2.5, 3.75)$	0.939 +/- 0.053	0.966 +/- 0.044	0.884 +/- 0.079	0.083 +/- 0.081
$(3.75, 5)$	0.986 +/- 0.083	0.997 +/- 0.075	0.969 +/- 0.106	0.036 +/- 0.128

Graph construction purity/efficiency isn't consistent among the eta ranges!

CONCLUSIONS

AND FUTURE STEPS

- GNN-based tracking typically involves 1) graph construction, 2) GNN inference (edge classification, object condensation), and 3) postprocessing (track finding)
- Example GNN pipeline based on edge classification and object condensation
  - Object condensation also accommodates track property predictions! → next step
- Future work:
  - Improve graph construction in central barrel region
  - Relax the truth cuts (re-impose noise, zero the p_T cut)
  - Incorporate track parameter predictions
  - Explore dynamic graph construction techniques like GravNet (no edge classification)
  - Full hyperparameter scan over network size/structure