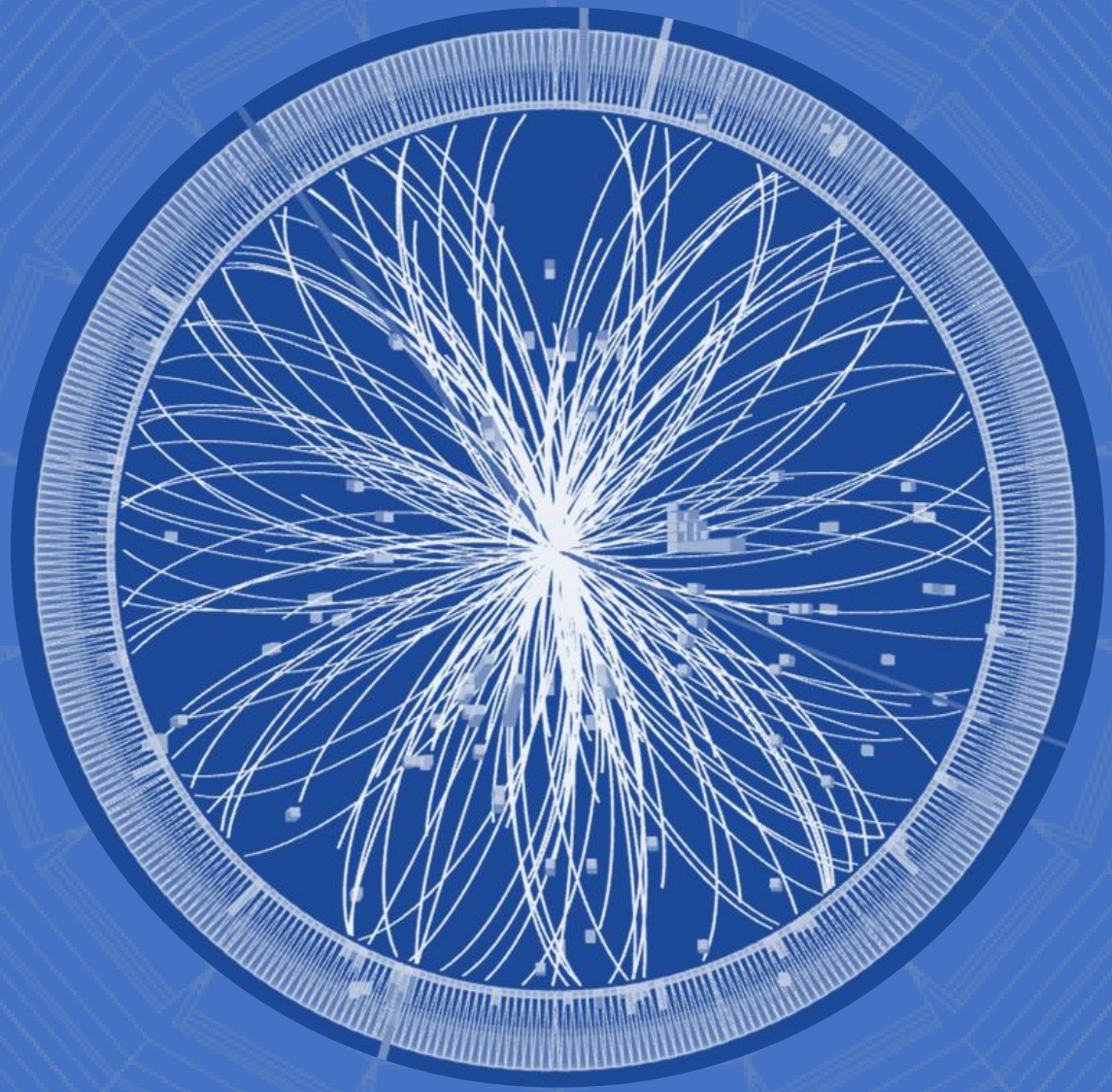
Charged Particle Tracking via Edge-Classifying Interaction Networks



Gage DeZoort*, Savannah Thais, Isobel Ojalvo, Peter Elmer, Javier Duarte, Vesal Razavimaleki, Markus Atkinson, Mark Neubauer

5/21/2021

* jdezoort@princeton.edu

PRINCETON UNIVERSITY

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

UC San Diego

iris hep

Overview:

- Our group studies GNN-based tracking workflows
  - Experimental ML approaches
  - Acceleration via heterogeneous resources
- This work is focused on the Interaction Network GNN architecture adapted to the task of edge classification
- We present a set of measurements at each stage of GNN-based tracking:
  1) Graph Construction
  2) Edge Classification
  3) Track Building

Accelerated GNN Tracking

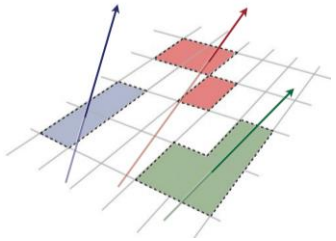
The tracking of charged particles produced in collisions at colliders is a crucial aspect of the science program in the experiments. One of the primary challenges for the HL-LHC is the ability to efficiently, accurately, and rapidly perform tracking in collision events with large interaction pile-up. This project aims to improve charged-particle tracking in the ATLAS and CMS experiments through the use of Geometric Deep Learn methods (particularly Graph Neural Networks (GNNs)) and hardware-based acceleration (currently focused on FPGAs). —

Most current GNN-based approaches to tracking proceed in three distinct stages: graphs are constructed from point cloud of hits in the tracker, the graphs are processed through a GNN to predict a score for each edge (high scores indicate that the edge like belongs to a true particle track, low scores indicate it is a spurious or noise edge), and finally a clustering or graph walk algorithm is used to group the high-scored edges into track candidates. We are studying innovations and optimizations at all three stages of this pipeline. We are also exploring alternate 'one-shot' architectures that are trainable end-to-end and go from point-clouds to track candidates with fit parameters in a single pass.

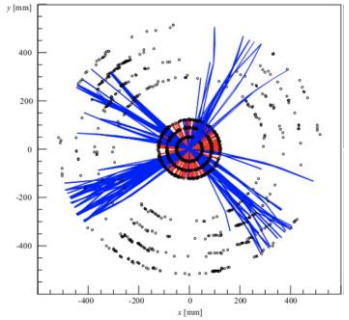
Project homepage: https://iris-hep.org/projects/accel-gnn-tracking.html

Local Reconstruction

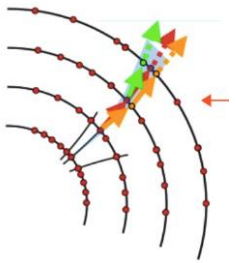
Raw data converted to 3D point cloud of tracker hits



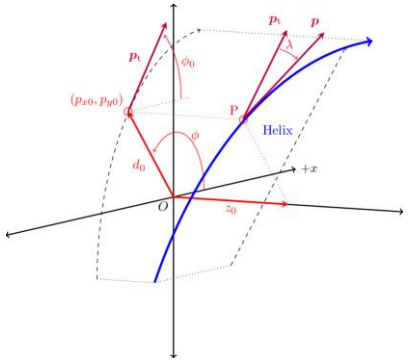
Iterative Tracking



1) **Track Seeding**
Initial track candidates (seeds) built from 2-4 pixel hits



2) **Track Finding**
Tracks extrapolated outward by a Kalman filter, additional hits added



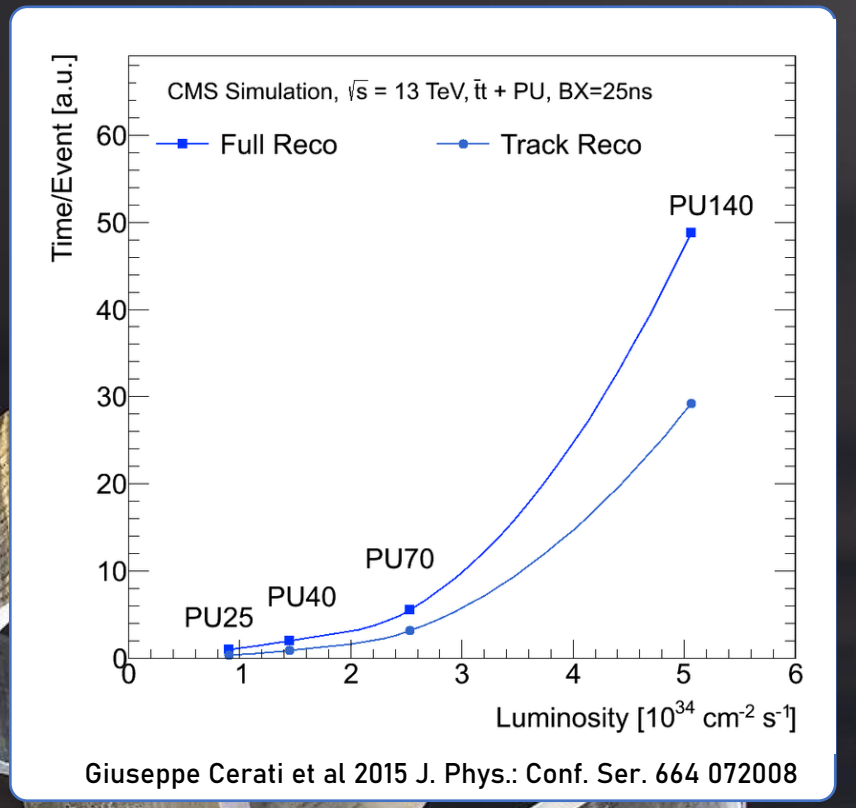
3) **Track Fitting**
Track parameters estimated from each trajectory

4) **Track Selection**
Track quality metrics computed, suboptimal tracks discarded

Charged Particle Tracking

- Tracks provide crucial measurements of charged particle trajectories
- Full track reconstruction occurs offline
- Reduced track reconstruction at the High-Level Trigger (HLT)

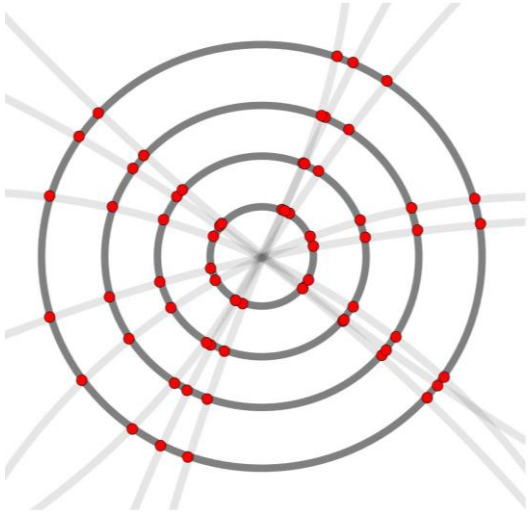
At the HL-LHC, instantaneous luminosity increases to 5×10^{34} $cm^{-2}s^{-1}$, delivering 250 fb^{-1} per year



Giuseppe Cerati et al 2015 J. Phys.: Conf. Ser. 664 072008

High-pileup scenario, at or above <PU> = 140-200

Tracker Data

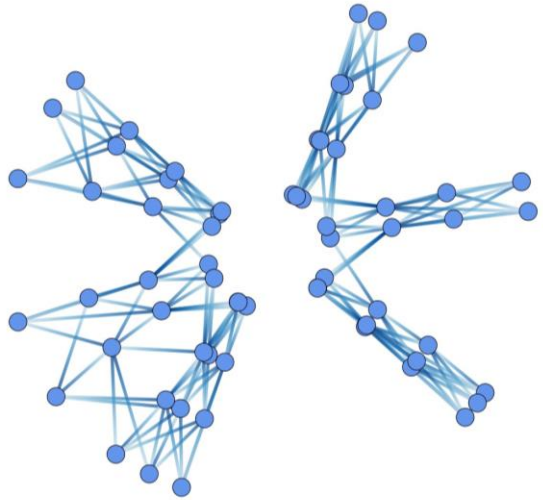


Track Hits

- Event yields N_{hits}
- Indices: [0, 1, 2, … N_{hits}-1]
- Positions: $[(r_0, \phi_0, z_0), (r_1, \phi_1, z_1), \ldots]$

Track Hits as Graphs

Hitgraph



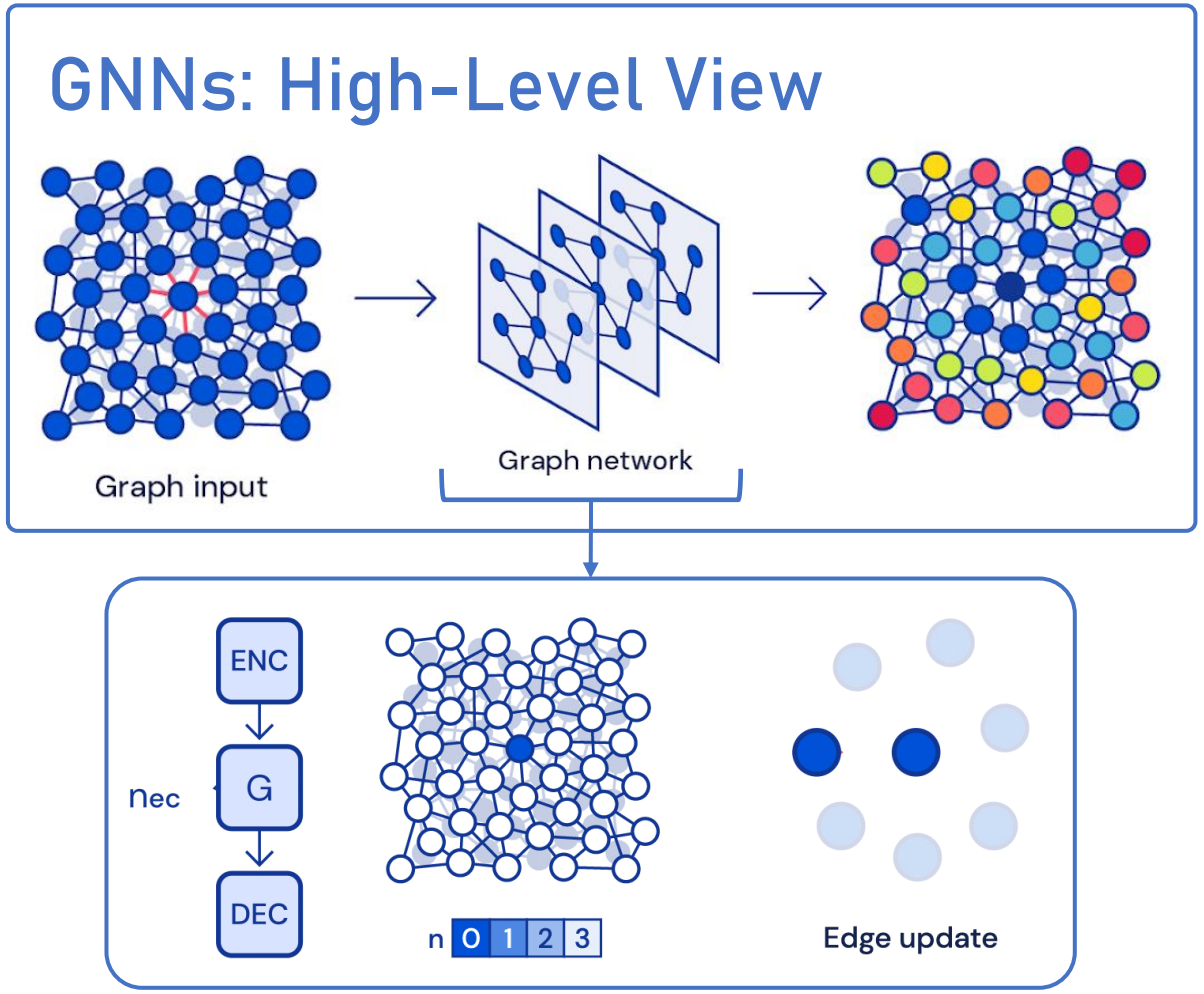
Nodes: hits themselves

- Indices: [0, 1, 2, … N_{hits}-1]
- Features: $[(r_0, \phi_0, z_0), (r_1, \phi_1, z_1), \ldots]$

Edges: track segment hypotheses

- Connect two hits: e_{ij} between hits i and j
- Construct N_{edges} edges for the event
- Each edge is true or false: $\vec{y} \in \{0,1\}^{N_{edges}}$
- Geometric Features: $e_{ij} = [\Delta r_{ij}, \Delta\phi_{ij}, \Delta z_{ij}, \Delta R_{ij}]$

Figure Source: https://arxiv.org/pdf/2012.01249.pdf



In general, GNNs operate in a learned latent space

Information is aggregated across local structure specified edges

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

Graph Neural Networks

- GNNs are well-suited to inference on tracker data:
  - Hits have irregular structure
  - Ability to leverage local geometric information
- GNNs aggregate information in graph-structured data to learn a new embedding
- Subsequent predictions are made on the learned representation of the graph

Three Key Stages of GNN-based Tracking

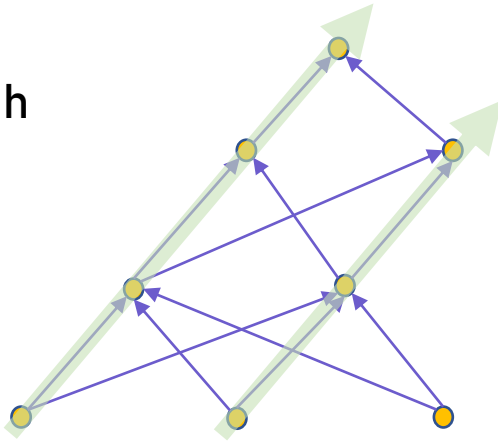
1. Graph Construction:

Convert tracker data to a hitgraph

= nodes (detector hits)

= edges (possible trajectories)

= true particle trajectory

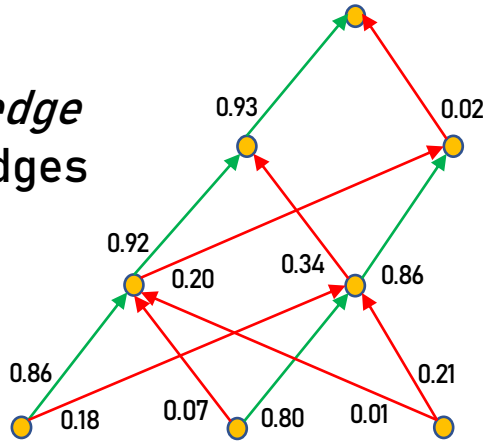


2. GNN Inference:

Edge classification, predict *edge weights* (probabilities that edges are real track segments)

= edge predicted to be true

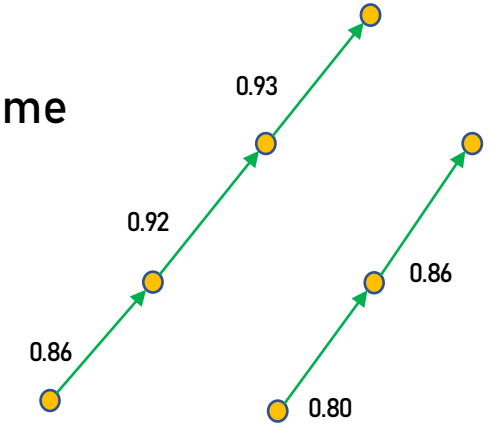
= edge predicted to be false



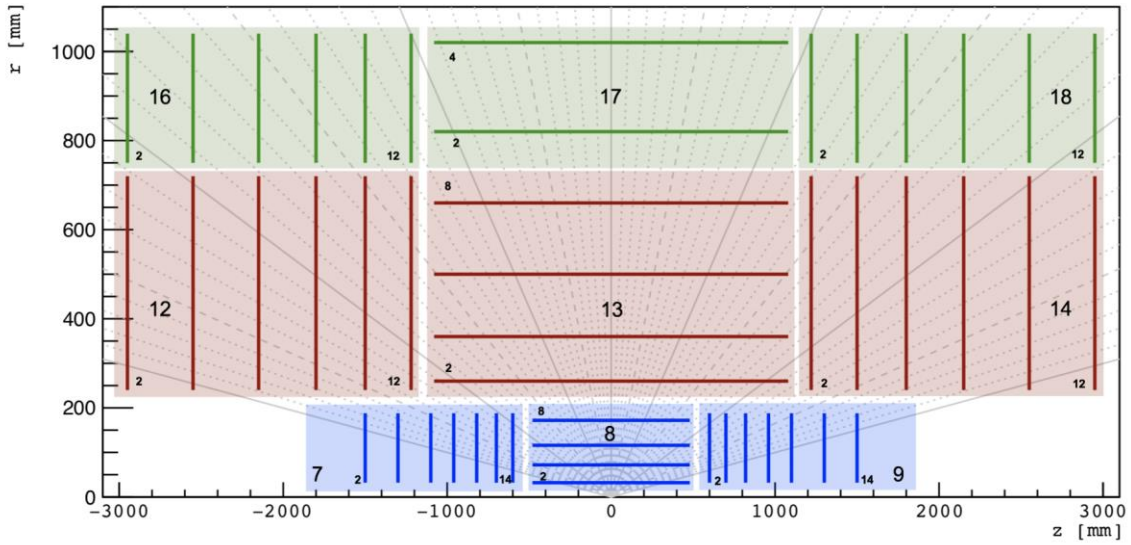
3. Track Building:

Cut edge weights below some threshold, apply clustering algorithm to extract tracks

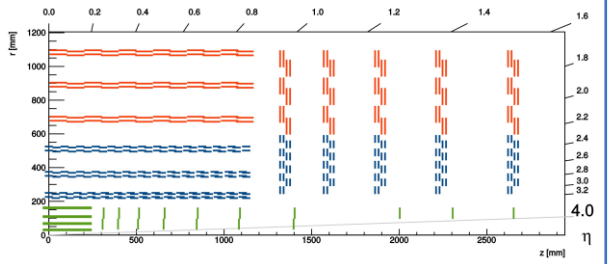
Output: hit clusters



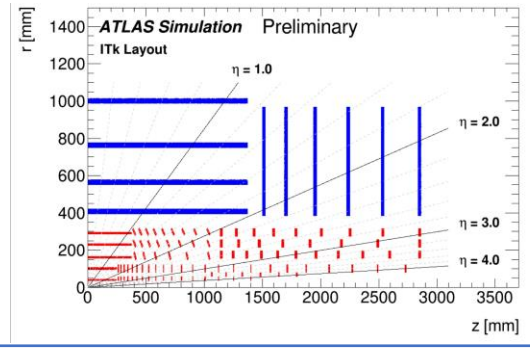
TrackML Generic Tracker



Phase-2 CMS Tracker Upgrade



Phase-II ATLAS ITK Upgrade



TrackML Dataset

- The following experiments are performed on Kaggle's TrackML dataset
- The TrackML detector is a generalized LHC-style tracker
- Events are generated with <PU> = 200
- We focus specifically on the pixel detector:
  - Improving track seeding
  - No hitgraph segmentation

TrackML Pixel Tracker

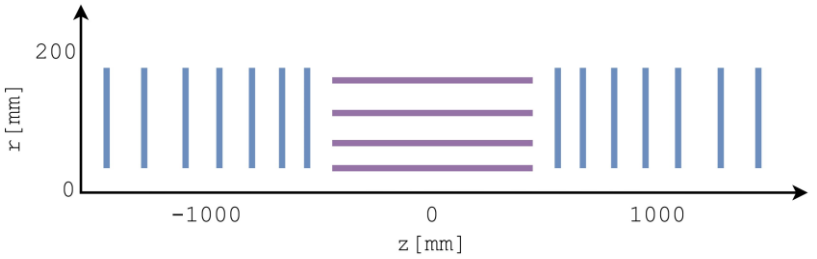
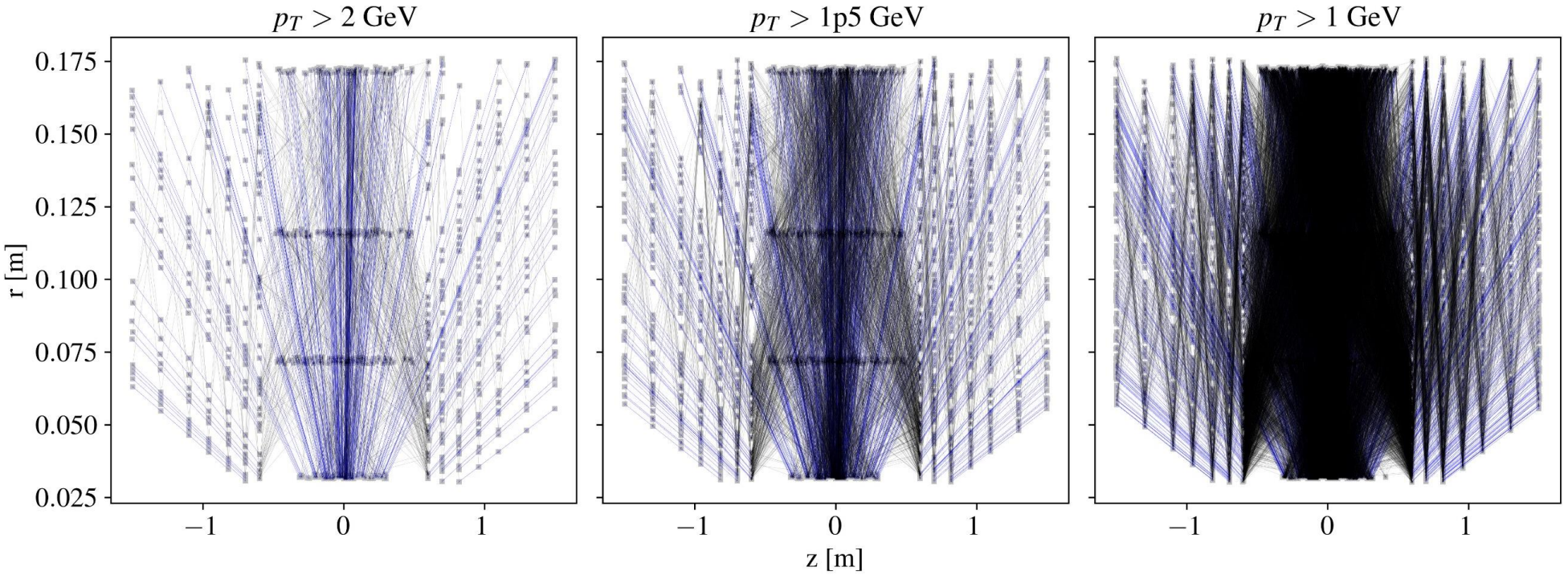


Figure Source: https://www.kaggle.com/c/trackml-particle-identification



p_T threshold used to modulate graph size

Truth labels:

Track Segments
False Edges

1. Graph Construction

- Goal: build graphs in the pixel layers simultaneously maximizing:
  - *Truth efficiency:* fraction of total track segments contained in the graph
  - *Edge efficiency:* ratio of track segments to total edges in the graph
- Key assumptions: 1) no noise hits and 2) one hit per layer per particle

HEP.TrkX+

Pixel barrel+endcap graph construction method

- Loop over pairs of layers, draw edges between hits satisfying geometric constraints on:

$$\phi_{slope}(i,j) = \frac{\phi_j - \phi_i}{\mathrm{r_j} - \mathrm{r}_i} \qquad z_0(i,j) = z_i - r_i \times \frac{z_j - z_i}{r_j - r_i}$$

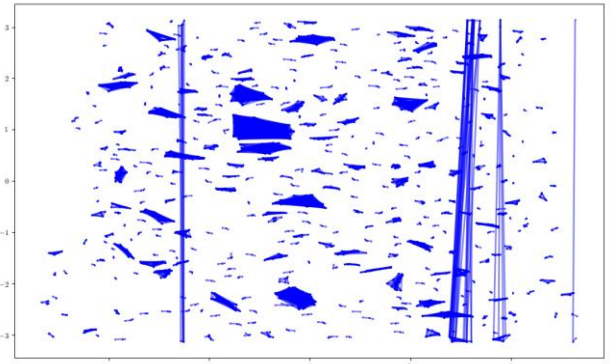
- *Barrel intersection cut:* reject edges between the barrel layers and endcap layers intersecting with an intermediate barrel layer

DBSCAN

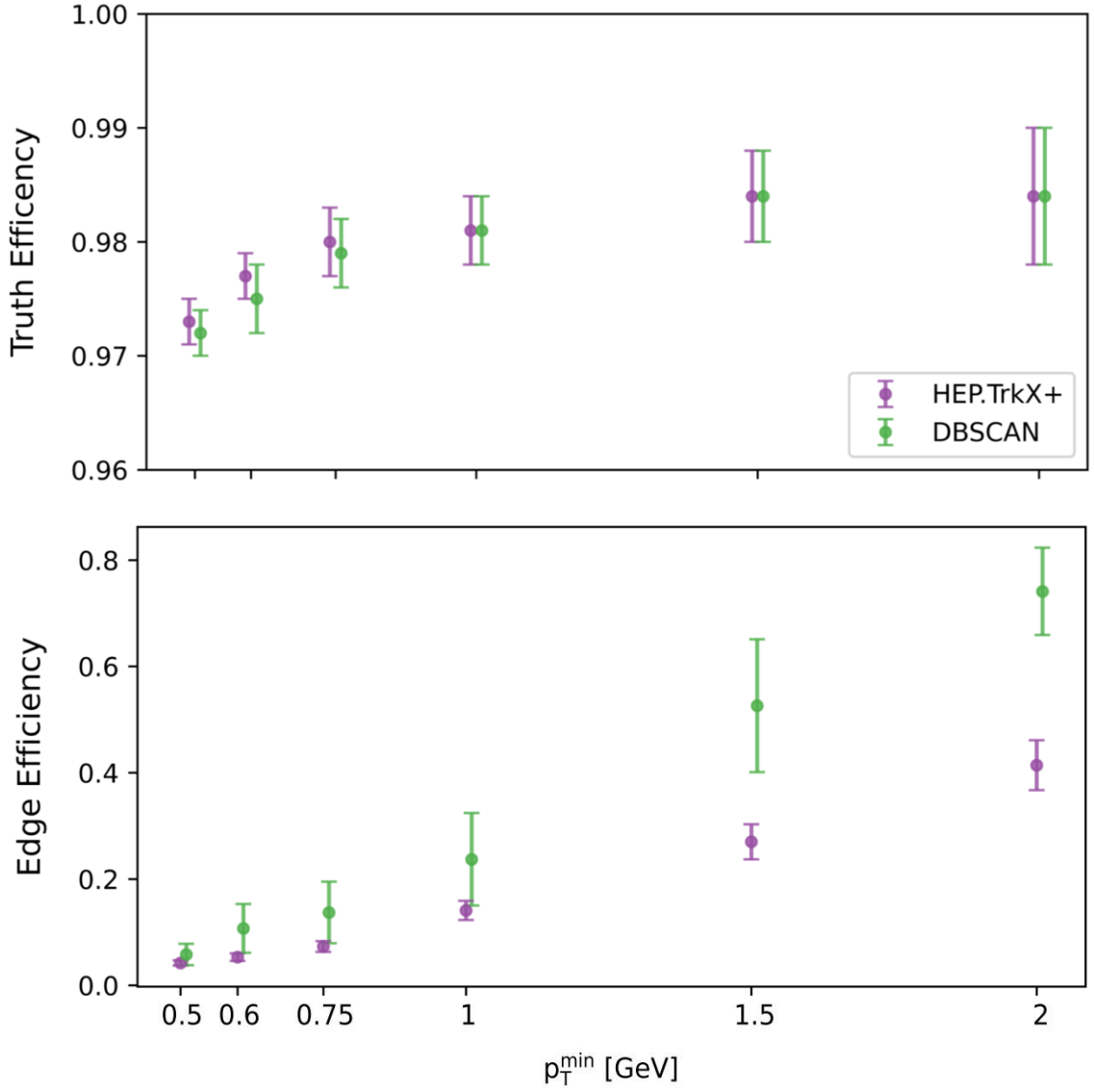
HEP.TrkX+ construction plus requirement that edges correspond to hits in the same DBSCAN η-ϕ cluster

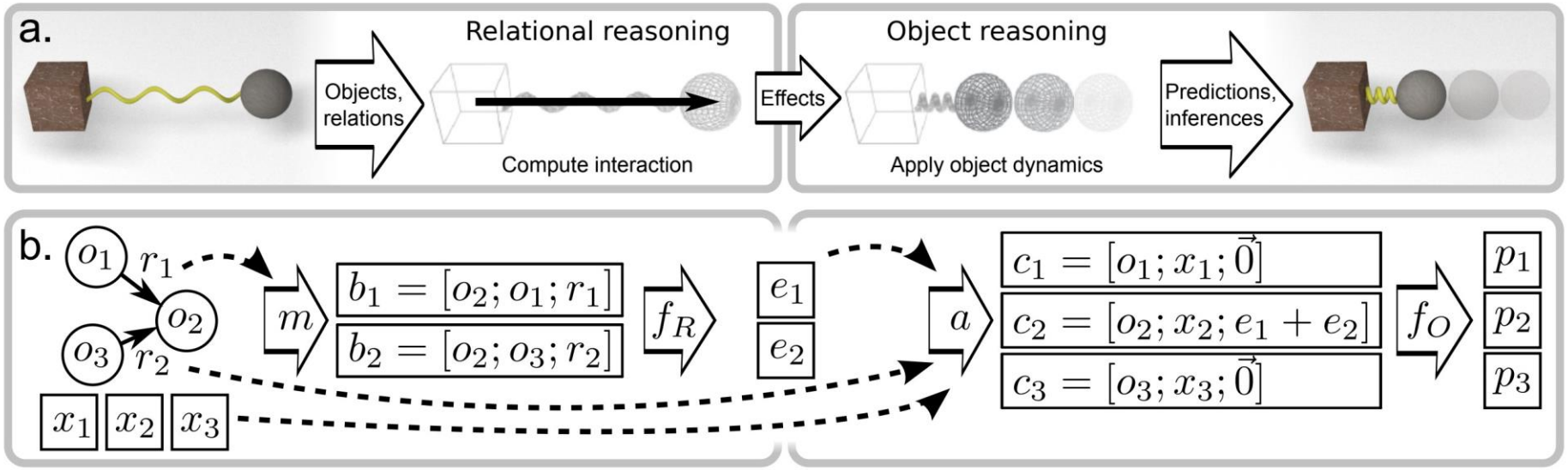
Tracks are observed to be well-localized in η-ϕ space

(η defined w.r.t. spatial coordinates)

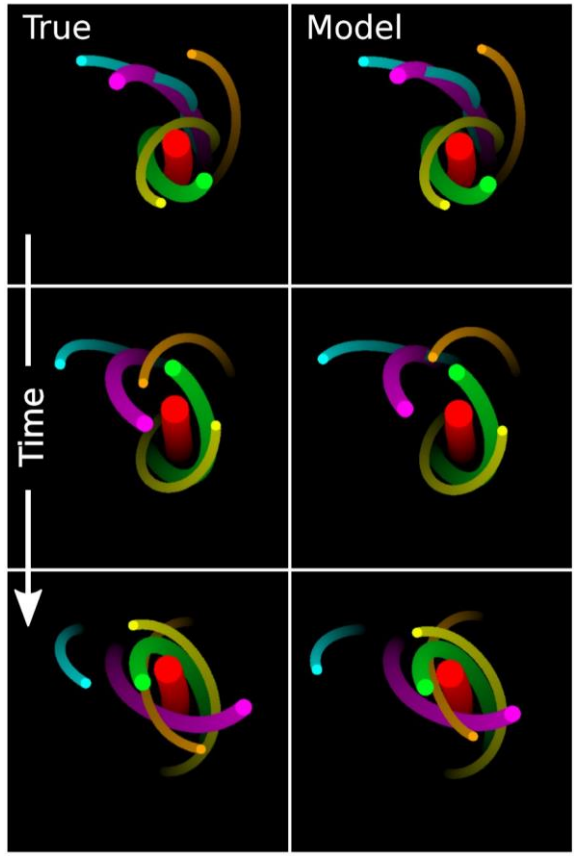


Example of DBSCAN hit clustering





Interaction Networks arXiv:1612.00222

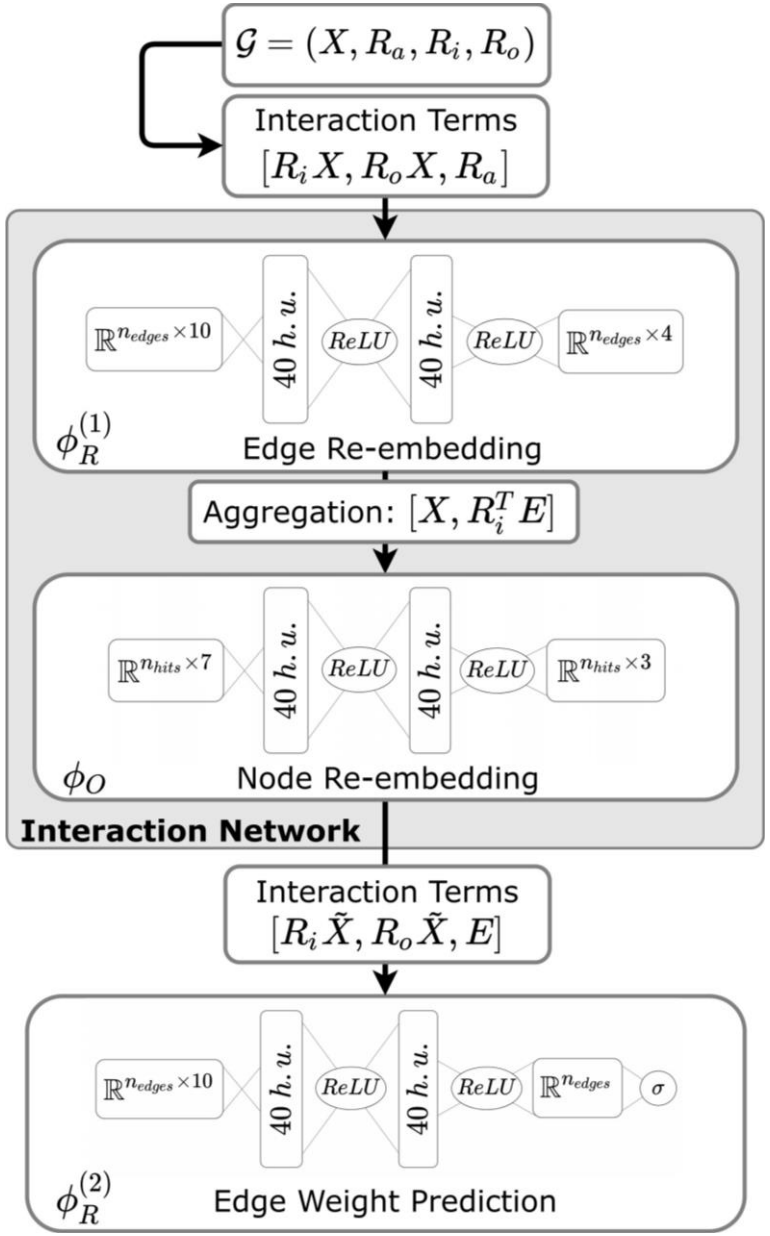


2. Edge Classification

- Adapted the Interaction Network (IN) architecture to the problem of edge classification
  - *Relational Reasoning:* compute interaction effects between objects
  - *Object Reasoning:* aggregate effects, update object features
- Each of these reasoning steps represents a learnable function

GNN Architecture

- Classic IN architecture plus an additional relational reasoning step to produce edge weights
  - 3 NN models, each with separate weights
  - 6448 total trainable parameters
  - Sigmoid constrains output to range (0,1)
- “Block” layout, in general with many iterations of the core IN, is the new paradigm
- Implemented with explicit matrix multiplications in PyTorch

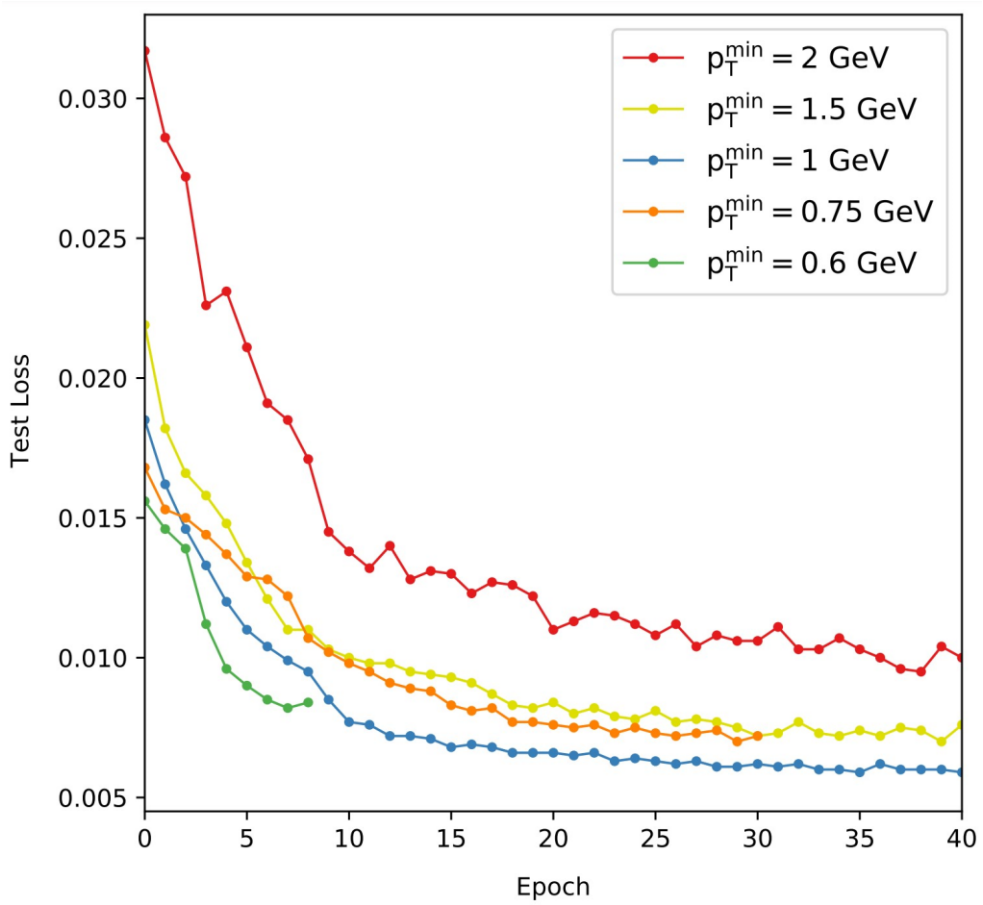


Graph input

Edge Block 1

Node Block

Edge Block 2



Training Specs

- 1000/400/100 train/test/validation split for HEP.TrkX+ graphs at various p_T thresholds
- Adam optimizer with a scheduled learning rate decay

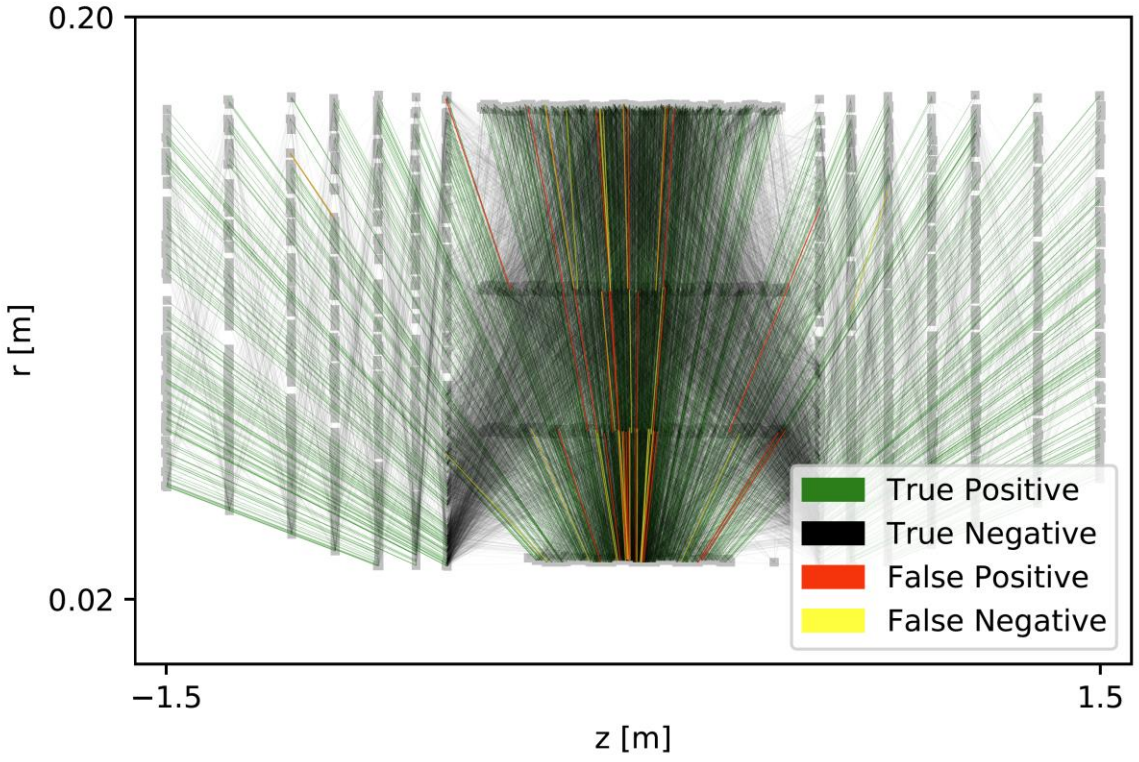
Optimization

- Binary cross-entropy loss function

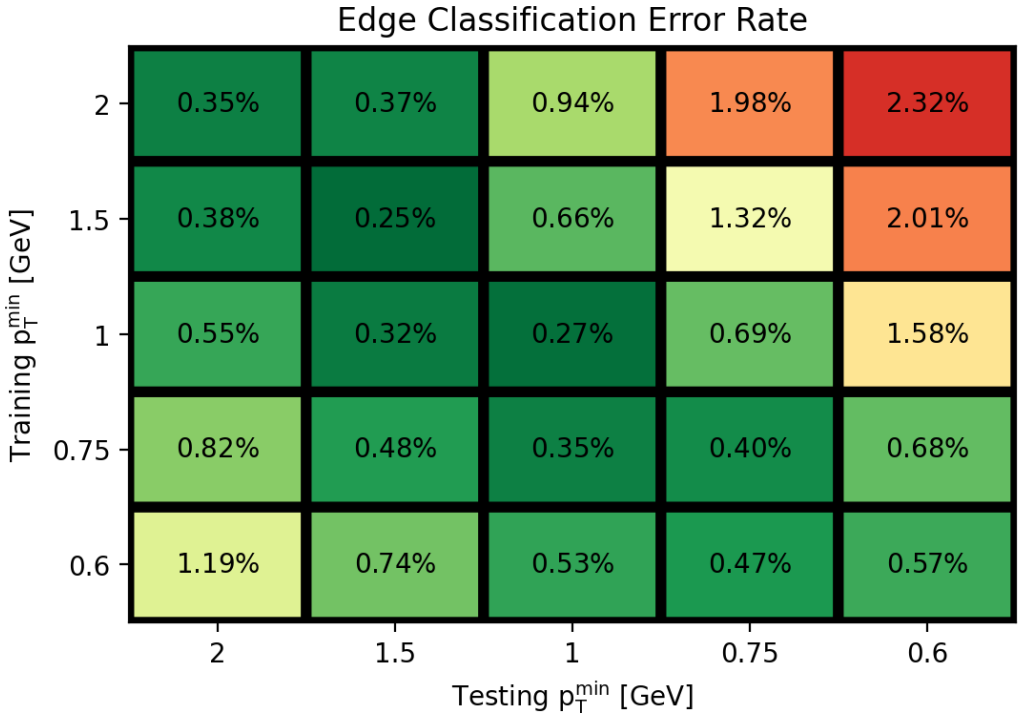
$$\ell\left(y_n, W_n(\mathcal{G})\right) = -\sum_{i=1}^{n_{\text{edges}}} \left(y_i \log w_i + (1 - y_i) \log(1 - w_i)\right)$$

Targets: $y_i = \{0, 1\}$

Predictions: $w_i \in (0, 1)$



p_T^{min} = 1 GeV
Loss: 6.0×10^{-3}
Accuracy: 99.81%
Errors: 57/29,661 edges



Models trained on train_1 sample graphs were tested on 500 graphs from the train_2 sample at various p_T^{min} thresholds

Averaged best inference time at each p_T^{min}

Inference Timing

GPU: Nvidia Titan Xp GPU with 12 GB RAM

CPU: 12-core Intel XeonCPU E5-2650 v4 @ 2.20 GHz

p_T^{min} [GeV]	HEP.TrkX		HEP.TrkX+	
	CPU [ms]	GPU [ms]	CPU [ms]	GPU [ms]
2	1.5 ± 0.2	0.8 ± 0.02	2.9 ± 0.6	0.9 ± 0.03
1.5	2.8 ± 0.1	0.8 ± 0.01	13.1 ± 5.5	1.0 ± 0.2
1	11.1 ± 3.5	1.0 ± 0.1	130.7 ± 60.7	8.0 ± 4.0
0.75	44.6 ± 18.0	2.9 ± 1.3	686.2 ± 281.2	44.3 ± 18.8
0.6	131.0 ± 42.9	8.4 ± 2.7	1954.0 ± 724.7	—
0.5	250.2 ± 77.8	16.0 ± 5.2	1952.0 ± 717.0	—

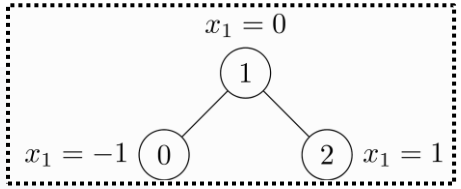
Substantial performance boost from heterogeneous resources like GPUs

Graphs exceed 12 GB during inference

Accelerating Inference

- PyTorch Geometric offers a promising alternative using the MessagePassing base class
  - Sparse edge representation vs. full adjacency matrices
  - Alternative IN representation as a message passing NN

PyTorch Geometric (PyG) Data Class

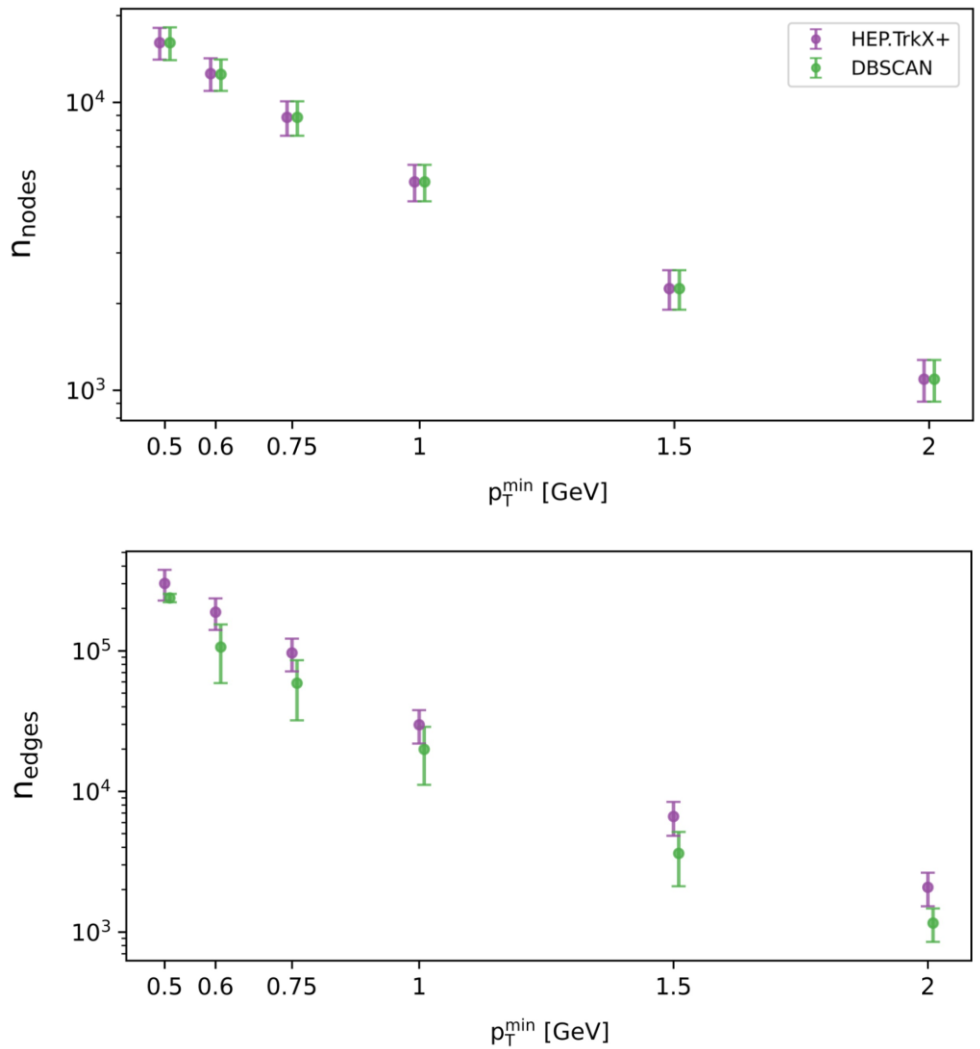



```
import torch
from torch_geometric.data import Data

edge_index = torch.tensor([[0, 1],
                           [1, 0],
                           [1, 2],
                           [2, 1]], dtype=torch.long)
x = torch.tensor([[-1], [0], [1]], dtype=torch.float)

data = Data(x=x, edge_index=edge_index.t().contiguous())
>>> Data(edge_index=[2, 4], x=[3, 1])
```

O(N_{edges}) representation

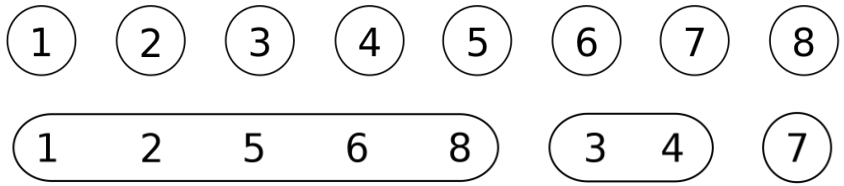
https://pytorch-geometric.readthedocs.io/en/latest/notes/introduction.html

Graph Sizes

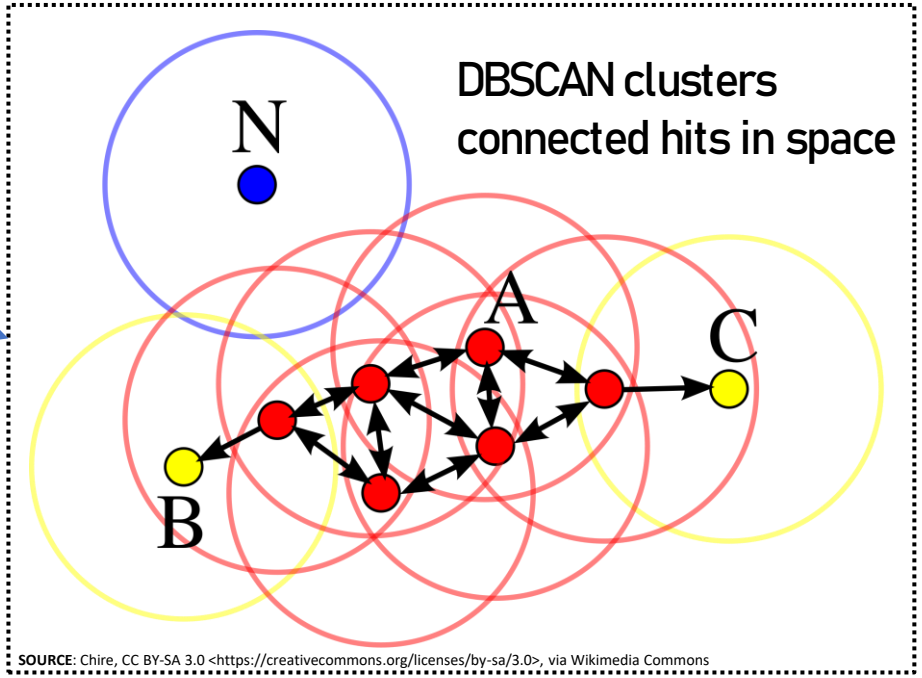


Edge i rejected if edge weight $w_i < \delta^*$, where δ^* is the threshold at which the true positive rate (TPR) equals the true negative rate (TNR) in a validation sample

Unionfind builds disjoint sets via edge-weighted connectivity



SOURCE: 93willy, CC BY-SA 3.0 <https://creativecommons.org/licenses/by-sa/3.0>, via Wikimedia Commons



DBSCAN clusters connected hits in space

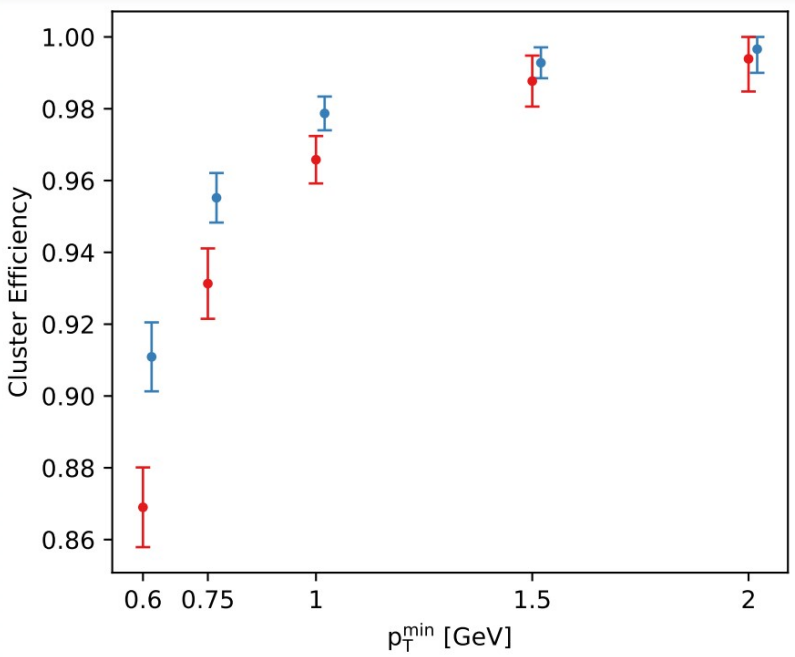
SOURCE: Chire, CC BY-SA 3.0 <https://creativecommons.org/licenses/by-sa/3.0>, via Wikimedia Commons

3. Track Building

- Clustering algorithms leverage GNN predictions to build tracks
  - DBSCAN: iteratively build clusters among neighboring connected points
  - UnionFind: separate disjoint sets to form tracks

https://commons.wikimedia.org/wiki/File:DBSCAN-Illustration.svg

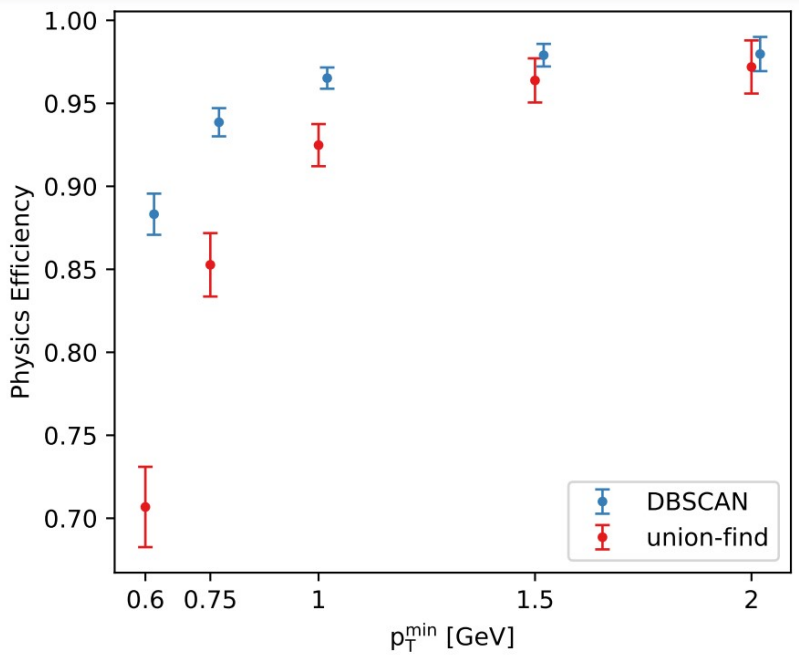
Cluster Efficiency



$$\epsilon_{cluster} = \frac{N_{same\ ID}}{N_{clusters}}$$

$N_{same\ pID}$ = # clusters containing only hits generated by the same particle

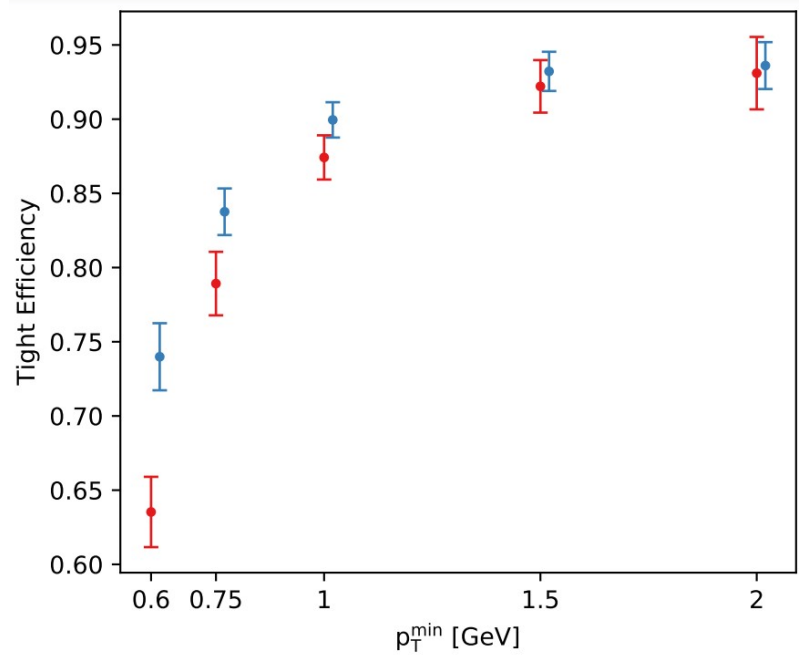
Physics Efficiency



$$\epsilon_{physics} = \frac{N_{>50\%}}{N_{particles}}$$

$N_{>50\%}$ = # clusters containing >50% hits from the same ID, where >50% of that ID's hits are in the cluster

Tight Efficiency



$$\epsilon_{tight} = \frac{N_{all\ hits}}{N_{clusters}}$$

$N_{all\ hits}$ = # clusters containing only hits from the same ID and every hit assigned to that ID

Conclusion

Upcoming Work

Graph Construction

- Explore additional clustering strategies
- Implement on-chip graph construction algorithms

Edge Classification

- Perform full suite of measurements using DBSCAN graphs
- Produce measurements for the PyTorch Geometric message-passing IN implementation

Track Building

- Compare additional clustering algorithms
- Explore learned strategies that take into account the specific values of the edge weights
- Sync measurements with CMS/ATLAS definitions for more direct comparison

Acknowledgements

S.T. and V.R. are supported by IRIS-HEP through the U.S. National Science Foundation (NSF) under Cooperative Agreement OAC-1836650. J.D. is supported by the U.S. Department of Energy (DOE), Office of Science, Office of High Energy Physics Early Career Research program under Award No. DE-SC0021187. G.D. is supported by DOE Award No. DE-SC0007968. We gratefully acknowledge the input and discussion from the Exa.TrkX collaboration.

References:

Tracking

- https://doi.org/10.1016/j.nuclphysbps.2015.09.436
- https://arxiv.org/pdf/1704.07983.pdf
- https://arxiv.org/abs/1901.11198
- https://arxiv.org/pdf/2012.14304.pdf
- https://twiki.cern.ch/twiki/bin/view/CMSPublic/DPGResultsTRK
- https://doi.org/10.1016/j.nuclphysbps.2015.09.162
- https://cds.cern.ch/record/2695003/files/PoSEPS-HEP2019153.pdf

IN Accelerated on FPGAs:

- https://arxiv.org/pdf/2012.01563.pdf

Interaction Network Paper:

- https://arxiv.org/pdf/1612.00222.pdf

Code Used This Paper:

- https://github.com/GageDeZoort/interaction_network_paper

Backup Material:

Message Passing Scheme

$$\mathbf{x}_i^{(k)} = \gamma^{(k)}\left(\mathbf{x}_i^{(k-1)}, \square_{j \in \mathcal{N}(i)}\, \phi^{(k)}\left(\mathbf{x}_i^{(k-1)}, \mathbf{x}_j^{(k-1)}, \mathbf{e}_{j,i}\right)\right)$$

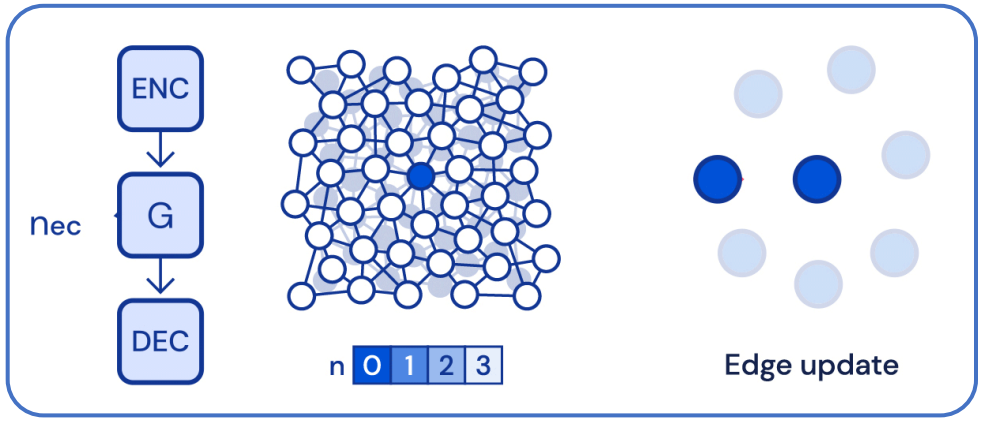
update node embedding

messages aggregated

messages passed within neighborhood

$\square_{j \in \mathcal{N}(i)}$ → sum, mean, max, etc.
(must be permutation invariant)

$\phi^{(k)}$ $\gamma^{(k)}$ → multi-layer perceptrons (MLPs)



https://arxiv.org/pdf/1812.08434.pdf, https://pytorch-

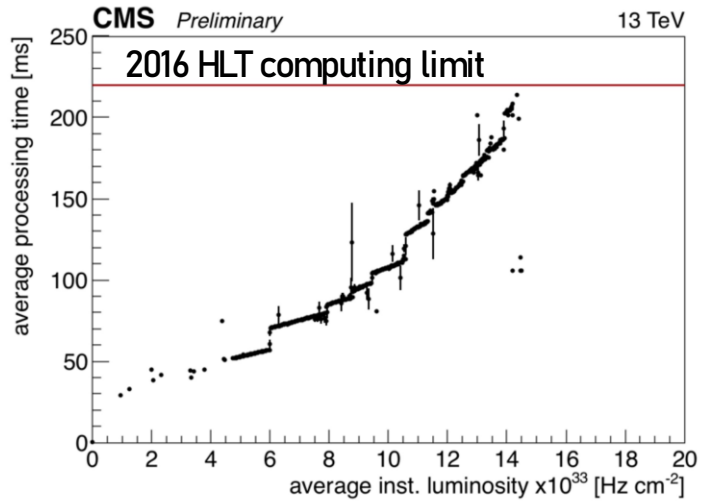
Message Passing GNNs

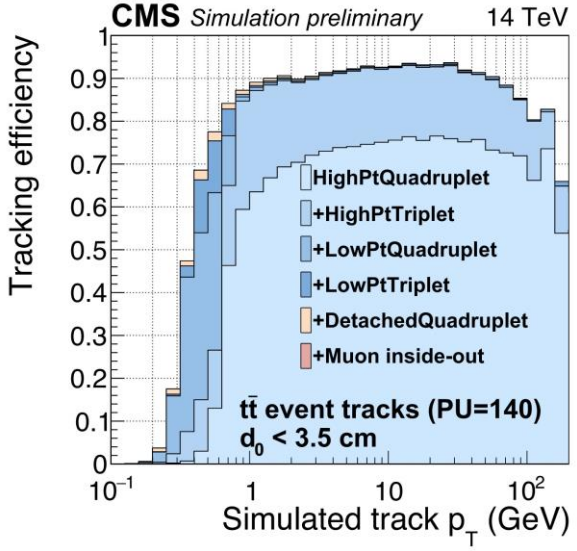
- The message passing neural network (MPNN) framework was proposed to summarize the overlapping behavior of several GNN models
  - See *Neural Message Passing for Quantum Chemistry* (arXiv:1704.01212)

HL-LHC CMS Tracking

- Tracking information will be available at the Level 1 (L1) trigger in the HL-LHC era
  - L1 trigger rates and latency will increase to 750 kHz and 12.5 us
- HLT tracking becomes challenging in high-pileup scenarios
  - During Run 2, HLT tracking took 35% of the reconstruction time
- Seeding should still take place in the Inner Tracker (pixel layers, 4 in the barrel, ~10(x2) in the endcaps)
  - Reduced cluster merging due to high granularity pixels
  - Ability to reconstruct low p_T tracks

Worse-than-linear scaling with pileup at the HLT

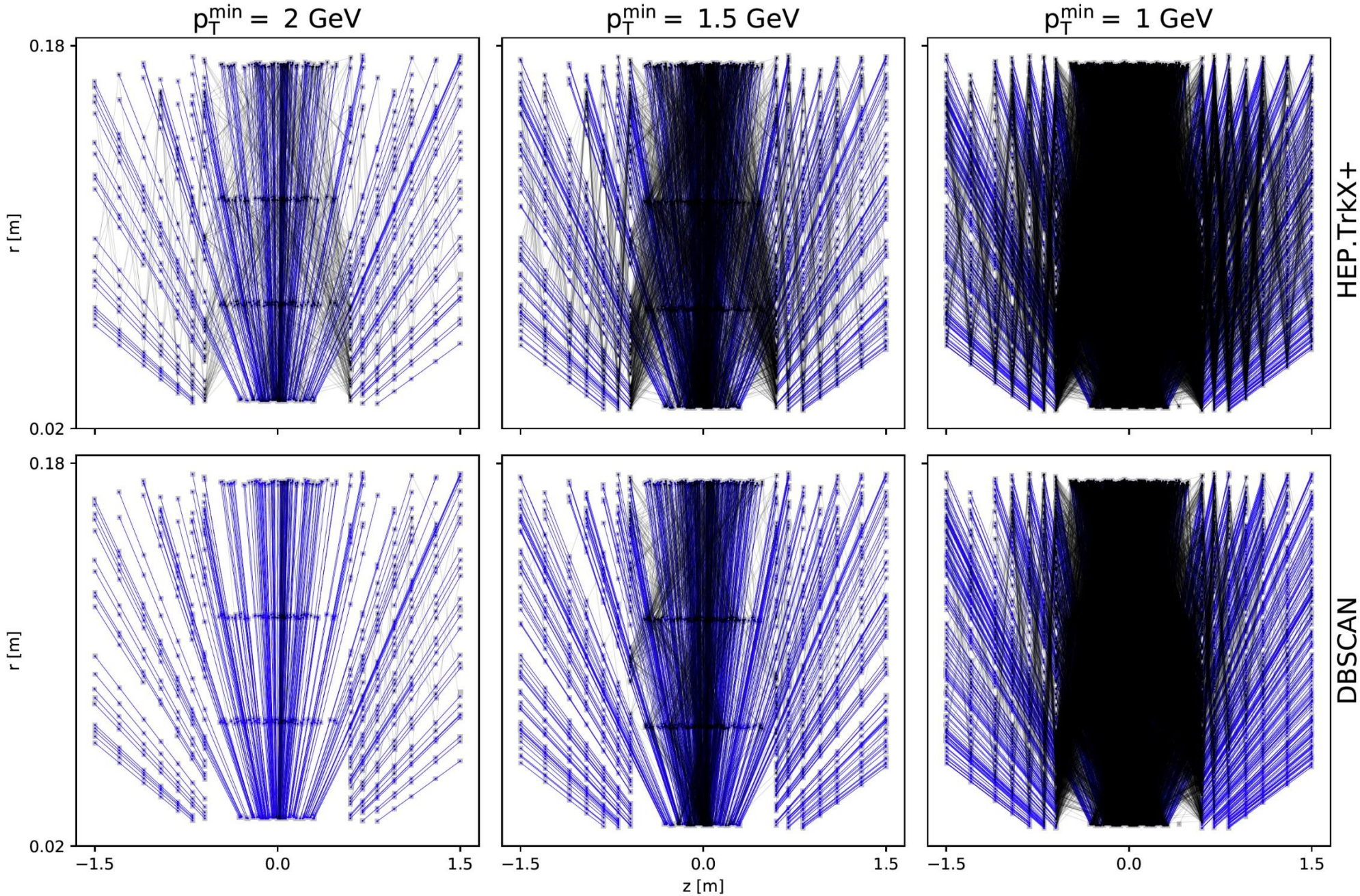


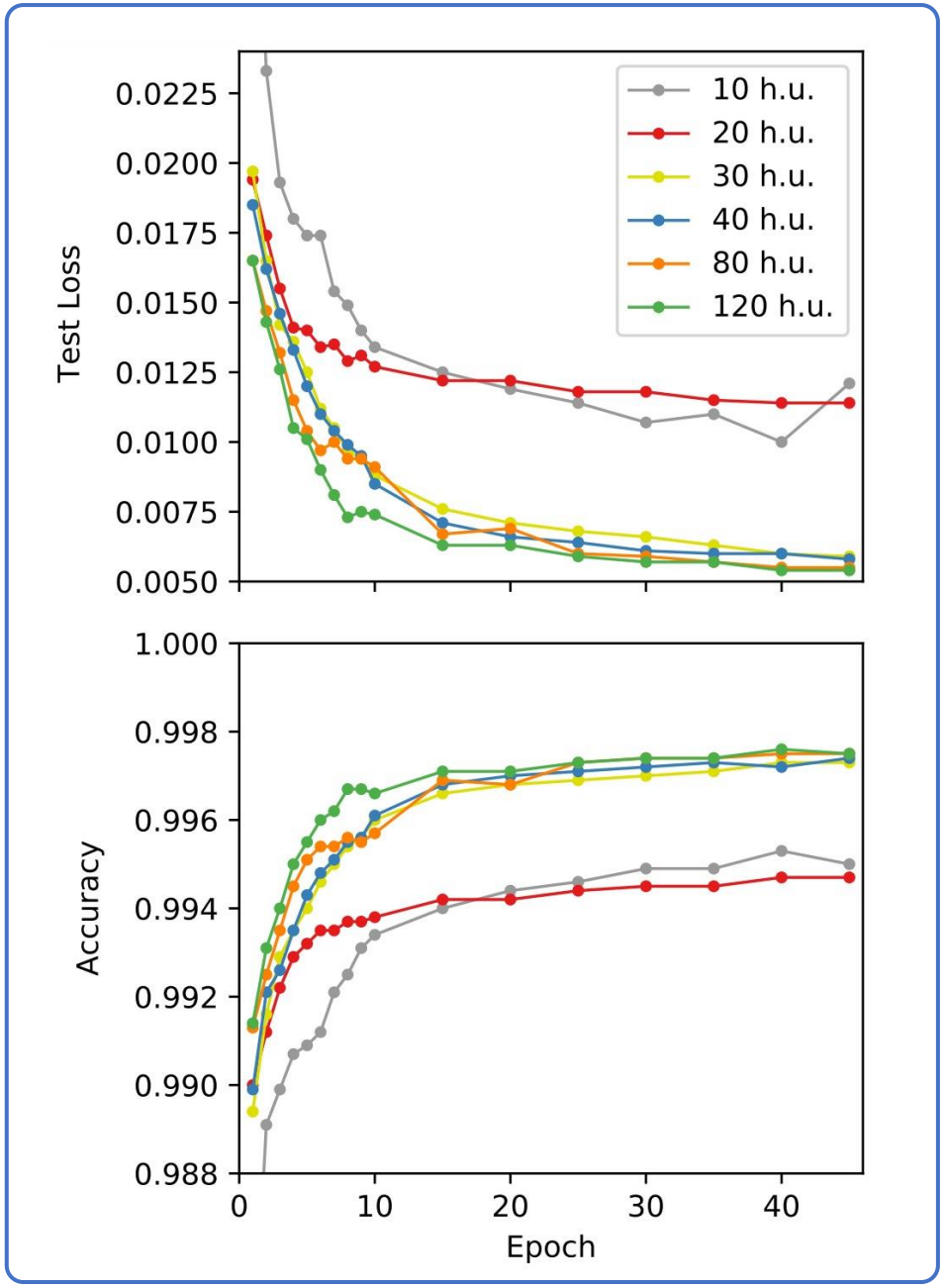


Projected high-pileup tracking efficiency for various seeding strategies

Iteration	Step Name	Seeding	Target Track
0	HighPtQuadruplet	pixel quadruplets	prompt, high p_T
1	HighPtTriplet	pixel triplets	prompt, high p_T
2	DetachedQuadruplet	pixel quadruplets	displaced
3	LowPtQuadruplet	pixel quadruplets	prompt, low p_T
4	LowPtTriplet	pixel triplets	prompt, low p_T
5	Muon inside-out	muon-tagged tracks	muon tracks

https://doi.org/10.1051/epjconf/201715000001
https://arxiv.org/abs/2003.06460


p_T^{min} = 2 GeV
p_T^{min} = 1.5 GeV
p_T^{min} = 1 GeV
HEP.TrkX+
DBSCAN
r [m]
z [m]
0.18
0.02
−1.5
0.0
1.5



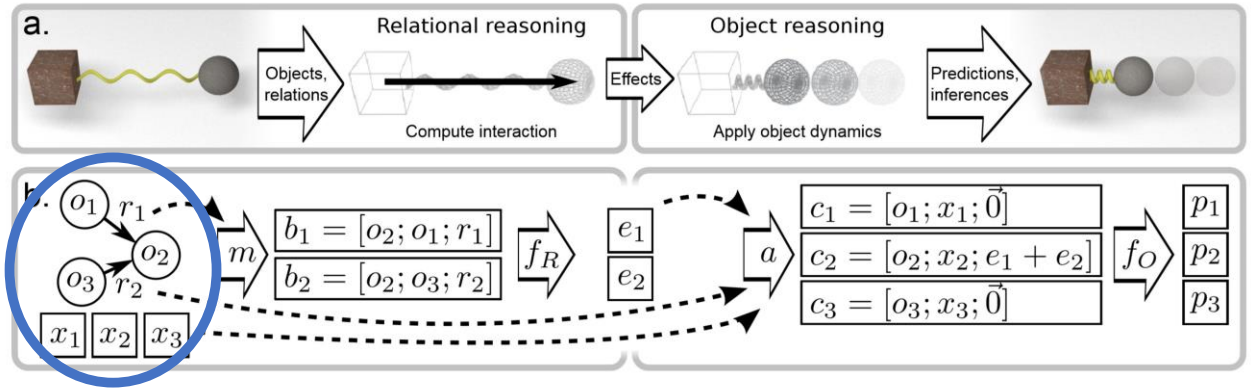
Hyperparameter Scans

- Architecture optimized on 1 GeV graphs; key assumptions:
  - Each MLP has the same number of hidden layers (2)
  - Each MLP has the same number of hidden units in each hidden layer
- Observed a boost in performance after ~30-40 hidden units per layer

IN Matrix Scheme

Begin with an attributed, directed multigraph *G* comprised of N_o objects and N_R relations

- Graph:
  G = <O, R>
- Objects:
  O = $\{o_j\}$ where j = 1, …, N_o
- Relations:
  R = { <i, j, r_k>$_k$ } where k = 1, …, N_R and i ≠ j
- External Effects
  X = $\{x_j\}$ where j = 1, …, N_o



Well-suited for our problem:

$\vec{x} = \{(r_1, \phi_1, z_1), (r_2, \phi_2, z_2), \ldots, (r_N, \phi_N, z_N)\}$

$(\boldsymbol{R}_i)_{hs}$ (segment *s* incoming to hit *h)*
$(\boldsymbol{R}_o)_{hs}$ (segment *s* outgoing from hit *h)*

noise modeling, etc.

Marshalling Function:

Re-arranges objects and relations into interaction terms

$m(G) = [\, OR_r ; OR_s ; R_a \,]$

where $O = \vec{x}^T$

$R_r = (\boldsymbol{R}_i)_{hs}$ (receiver relations)

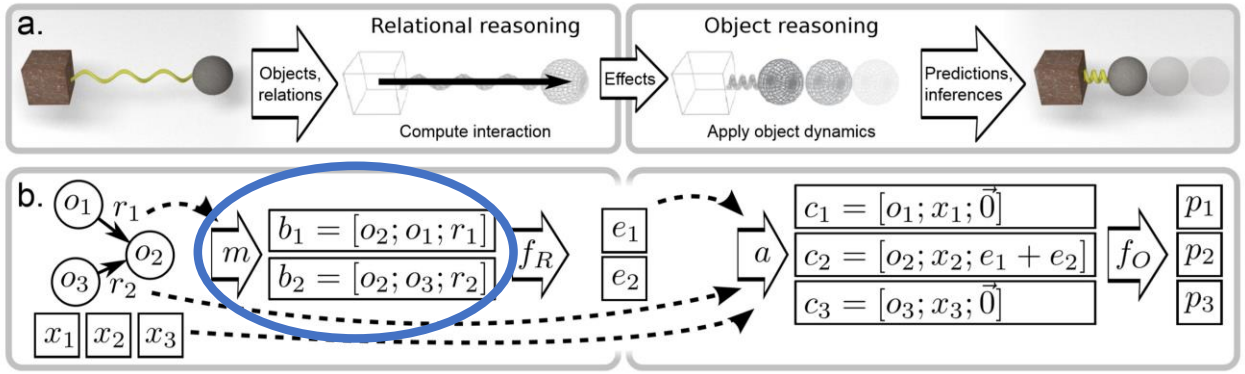
$R_s = (\boldsymbol{R}_o)_{hs}$ (sender relations)

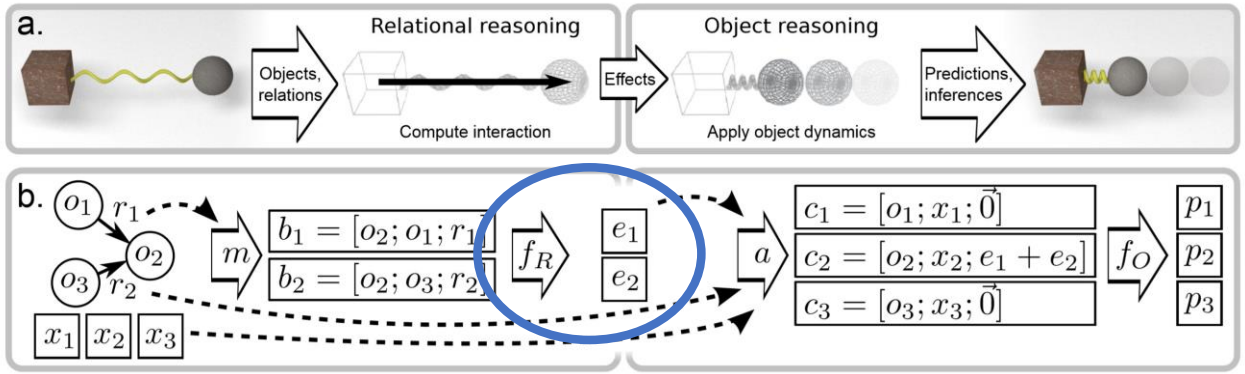
$R_a = \{(a_1), (a_2), \ldots, (a_N)\}, \; a_i = \begin{cases} 1 & same\ layer \\ 0 & cross\ layers \end{cases}$

Relational Network:

Predicts the effect of each interaction by applying a MLP to each interaction term

$\Phi_R(m(G)) = E$





Aggregation:

Sums effects for each receiver, combines with objects and external effects

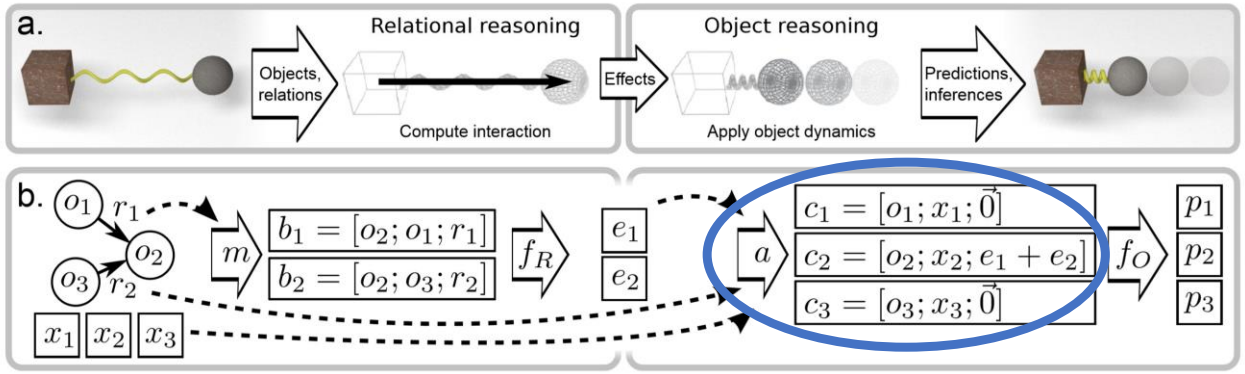
$a(G, X, E) = [O; X; \bar{E}] = C$

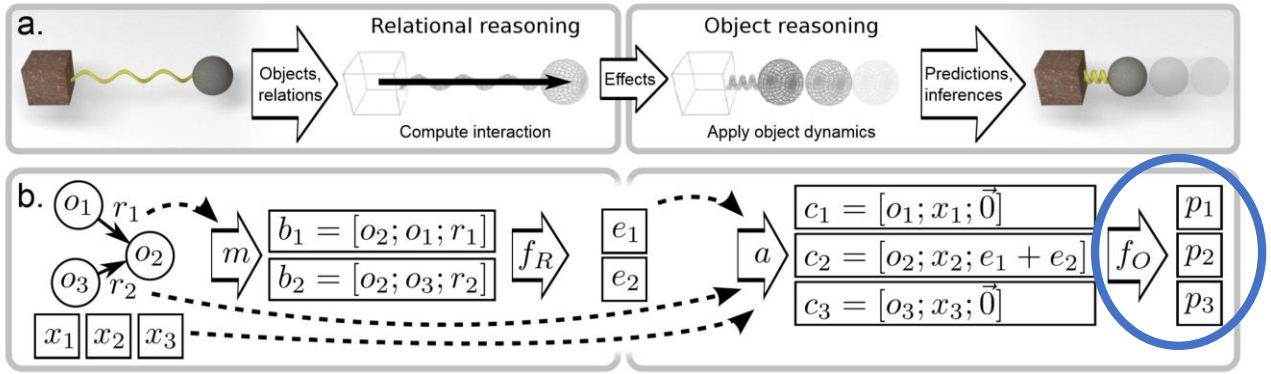
where $\bar{E} = ER_r^T$

Object Model:

Applies a MLP to determine how interactions and dynamics influence the objects

$\Phi_R(C) = P$





Interaction Network:

Applies relational and object models in stages to infer abstract interactions and object dynamics

- Interaction Network:

$IN(G) = \Phi_o(a(G, X, \Phi_R(m(G))))$

- Marshalling:
  $IN(G) = \Phi_o(a(G, X, \Phi_R(B)))$
- Relational Model:
  $IN(G) = \Phi_o(a(G, X, E))$
- Aggregation:
  $IN(G) = \Phi_o(C)$
- Object Model:
  $IN(G) = P$