OPTIMIZATION AND EVALUATION OF EDGE CLASSIFYING GNNS FOR CHARGED PARTICLE TRACKING

Savannah Thais, Markus Atkinson, Gage DeZoort, Javier Duarte, Mark Neubauer, Isobel Ojalvo

ICHEP 2022, Bologna

07/08/2022

UC San Diego

iris hep

ILLINOIS

PRINCETON UNIVERSITY

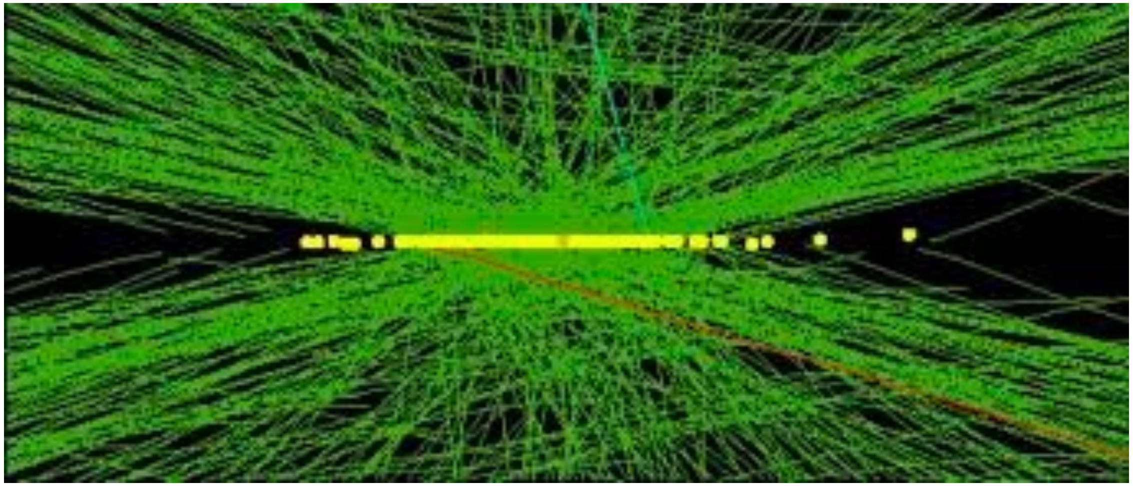
Outline

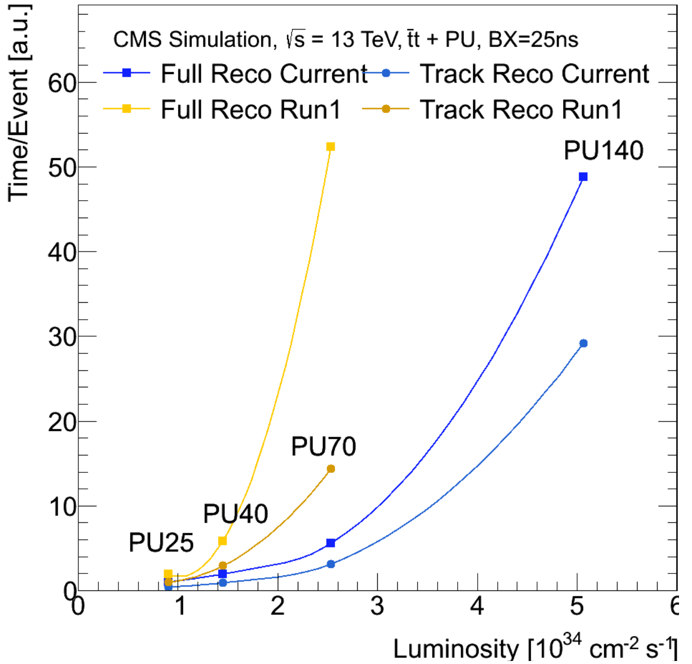
1. Introduction to tracking with GNNs
2. Edge Classifying GNN Architectures
3. Optimization + Experiment Studies
4. Related, Ongoing, and Future Work

Introduction to Tracking with GNNs

Tracking Challenge at HL-LHC

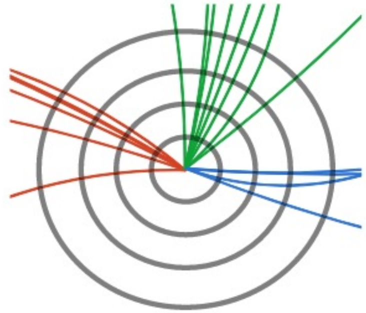
- Tracking is critical for meeting physics goals of LHC
- Tracking is the most computationally intensive reco task
  - Time grows worse than quadratically with increasing number of collisions
  - Additional challenges of overlapping tracks
- Must exploit developments in hardware and software
  - Improved algorithms and data representation
  - Parallelize currently serial algorithms
  - Adapt to modern architectures (GPU, FPGA)

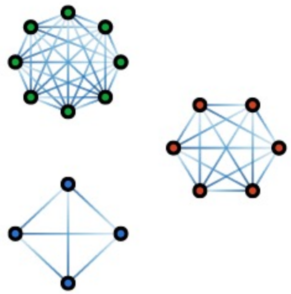


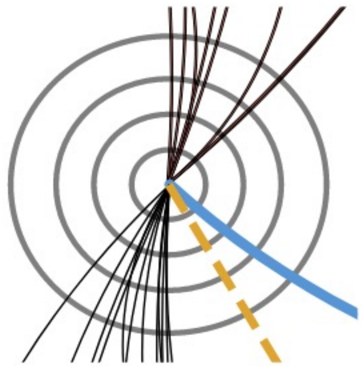


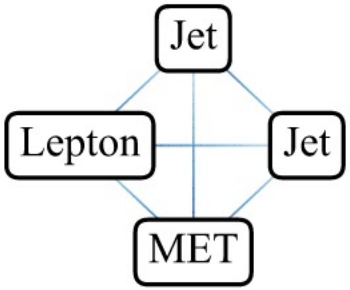
Graphs

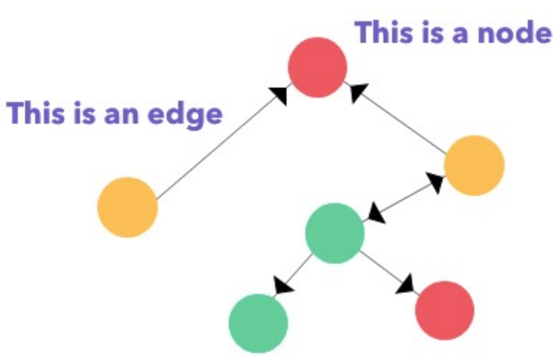
- A graph is a mathematical structure composed of:
  - Nodes: vertices with associated information (spatial coordinates, features, etc)
  - Edges: connections between nodes
    - Can be directed or undirected, can have associated information
  - Graphs can represent many types of relational/geometric data
- Intuitive representation for geometric, structured, variable physics data

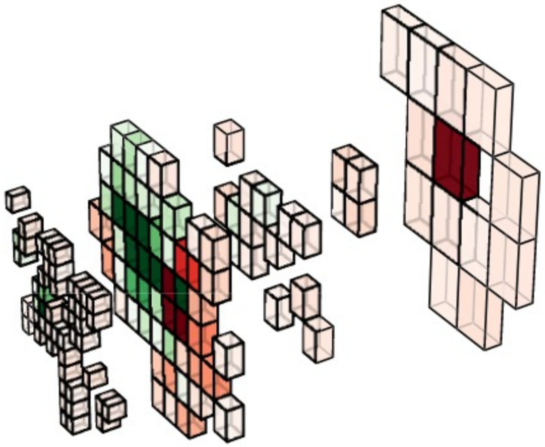












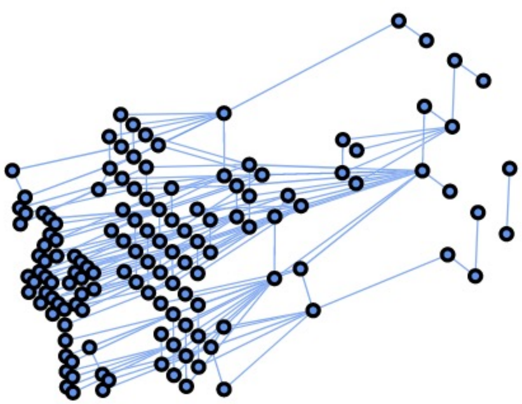
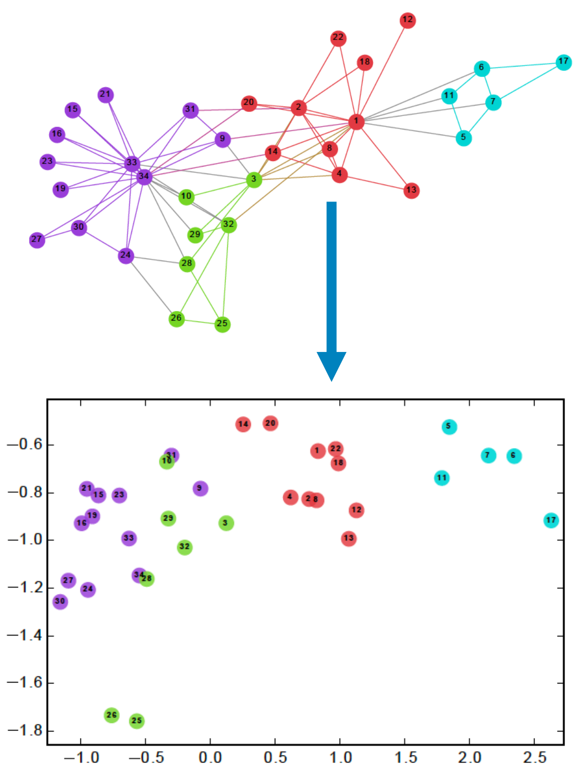


Image source

'Vanilla' Graph Neural Networks

- GNNs learn a smart embedding of the graph structure
- Leverage geometric information by passing and aggregating messages from neighbors
- Practically, W_k and B_k are shallow neural networks applied to a neighborhood based feature set



Initial "layer 0" embeddings are equal to node features

$$\boxed{\mathbf{h}_v^0 = \mathbf{x}_v}$$

$$\boxed{\mathbf{h}_v^k} = \boxed{\sigma}\left(\mathbf{W}_k \boxed{\sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|}} + \mathbf{B}_k \boxed{\mathbf{h}_v^{k-1}}\right), \quad \forall k > 0$$

previous layer embedding of v

kth layer embedding of v

non-linearity (e.g., ReLU or tanh)

average of neighbor's previous layer embeddings

'Vanilla' Graph Neural Networks

Initial "layer 0" embeddings are equal to node features

$$\mathbf{h}_v^0 = \mathbf{x}_v$$

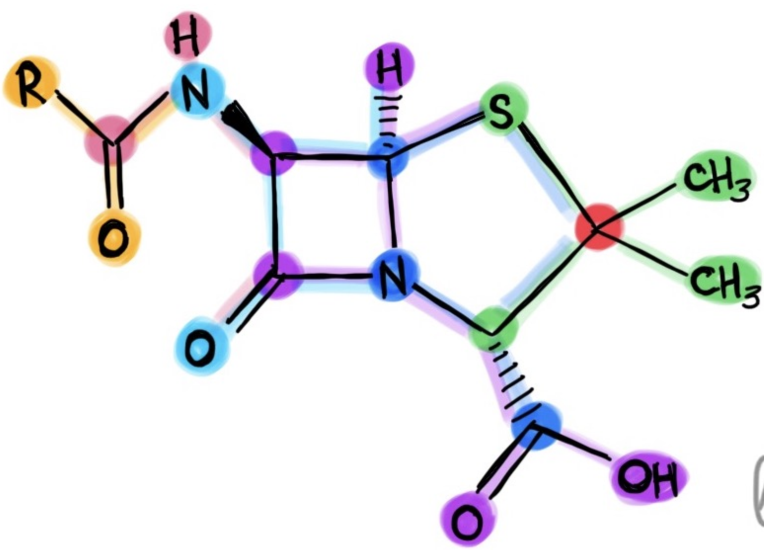
previous layer embedding of v

$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1}\right), \quad \forall k > 0$$

kth layer embedding of v

non-linearity (e.g., ReLU or tanh)

average of neighbor's previous layer embeddings



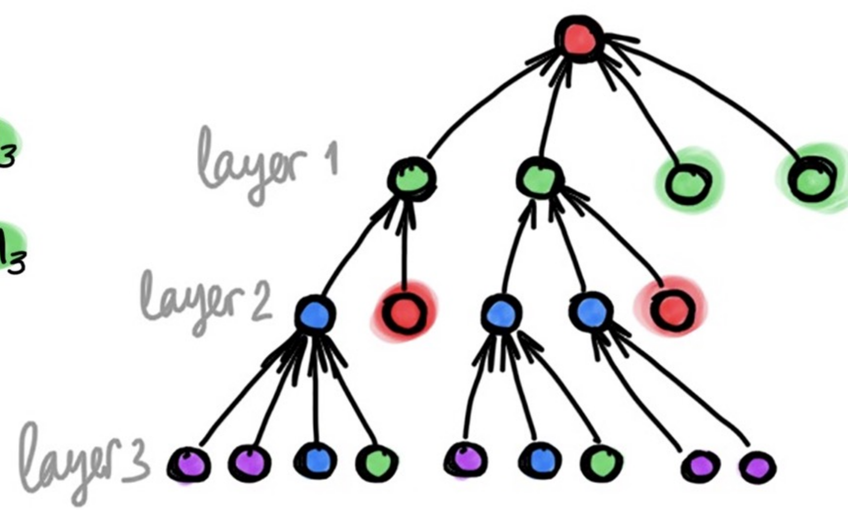
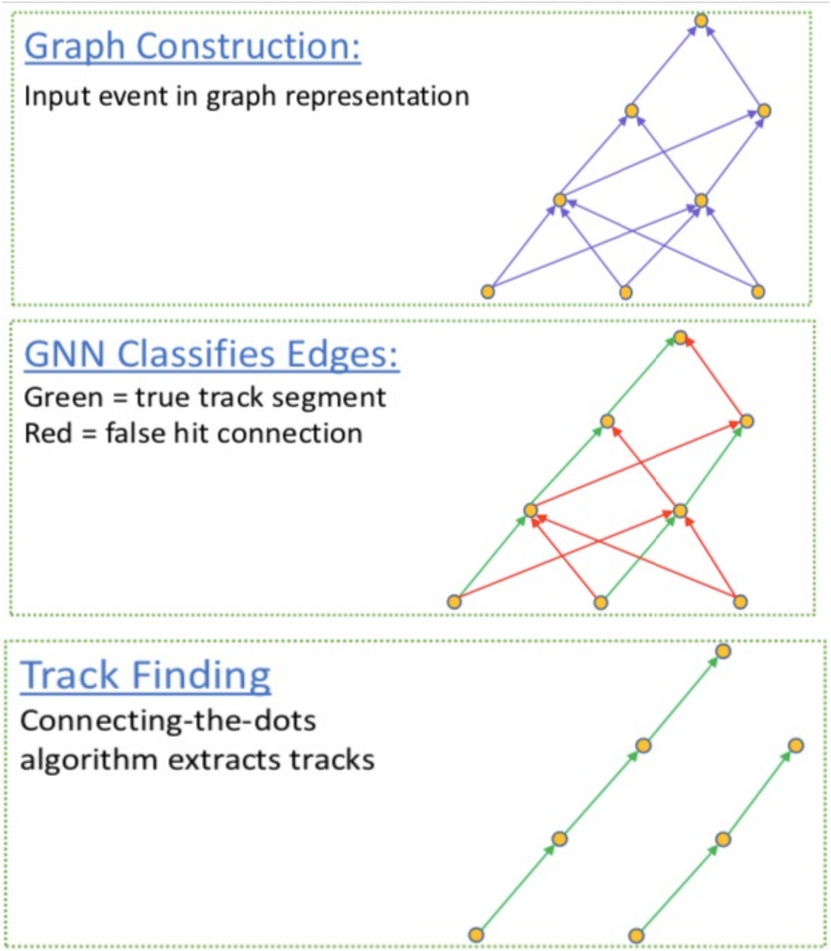


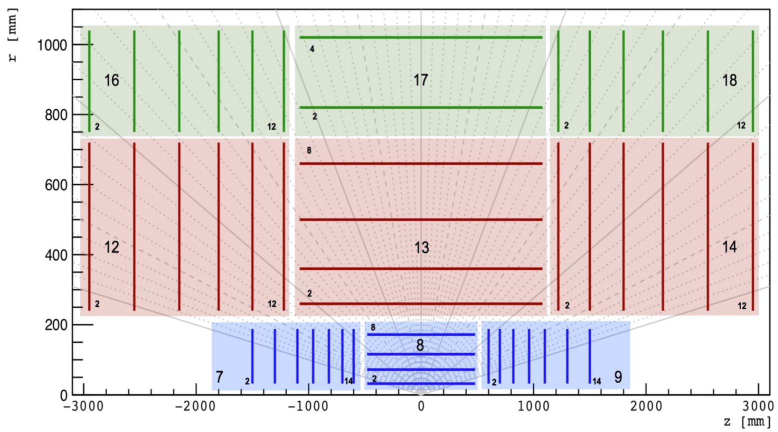
Image source

GNNs for Tracking

Basic procedure

1. Form initial graph from spacepoints/hits (pre-processing)
2. Process with GNN to get probabilities of all edges
3. Apply post-processing algorithm to link edges together into tracks and get parameters





- Many places to improve/innovate
  - Graph construction, architectures, data augmentation…
- Most work shown here uses [TrackML dataset]
  - Open, experiment agnostic
  - 200 PU, silicon semiconductor detector

Edge Classifying GNN Architectures

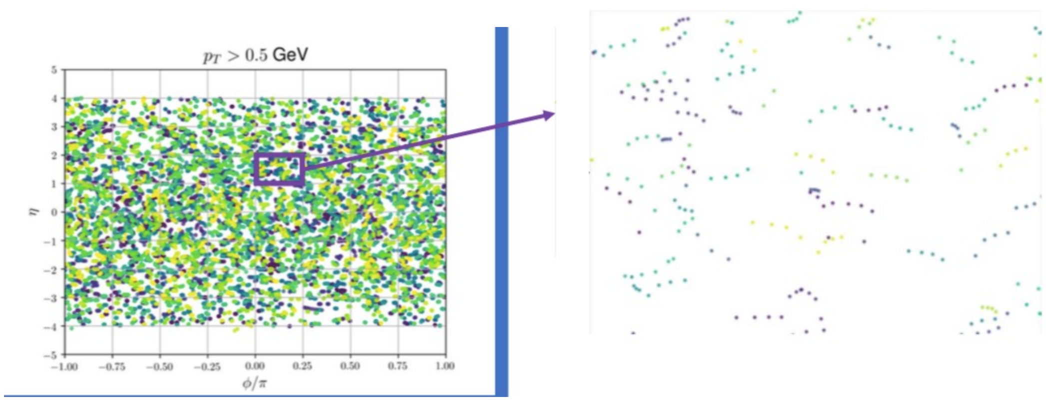
Graph Construction

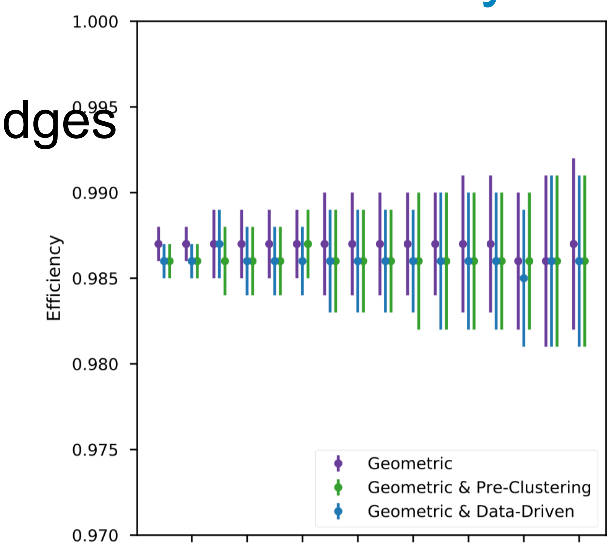
Optimizing graph construction can help GNNs learn effectively

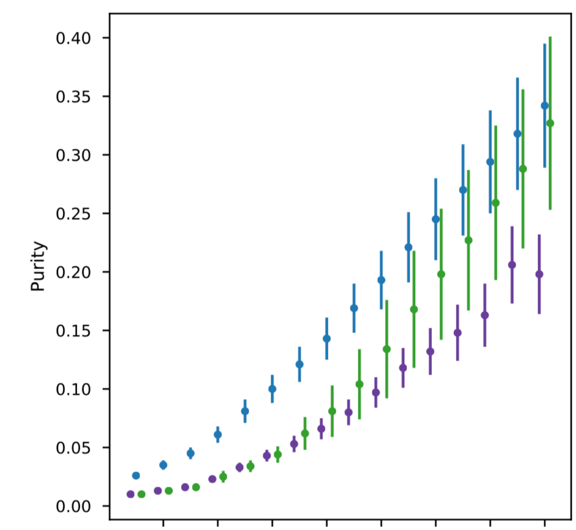
- **Purity**: true edges/all edges
- **Efficency**: true edges in graph/all possible true edges

Current Methods

- **Geometric**: create edges between nodes in adjacent layers within allowed cone
- **Preclustering**: geometric + DBScan in eta-phi space
- **Data driven/module map**: edges allowed between modules that have produced valid track segments in independent sample

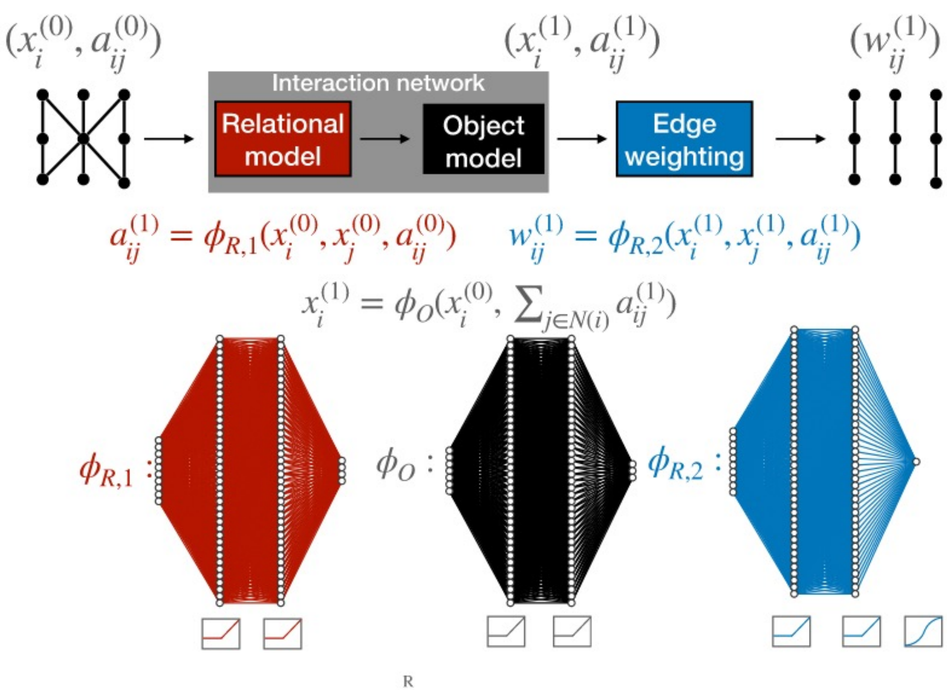


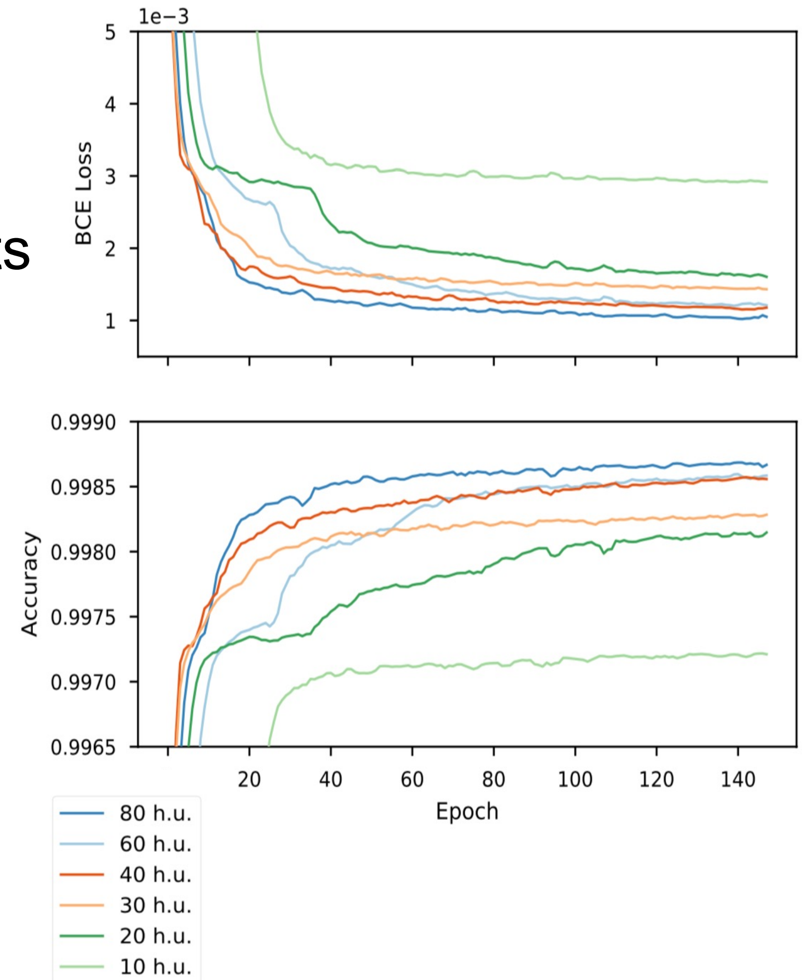




Interaction Network

- Originally developed for next time step predictions of physical systems
- Our implementation adds an additional relational model to predict edge weights
- Includes geometric edge features
- Total of ~6,000 learnable parameters
  - Much smaller than other architectures
  - After hyperparameter scan





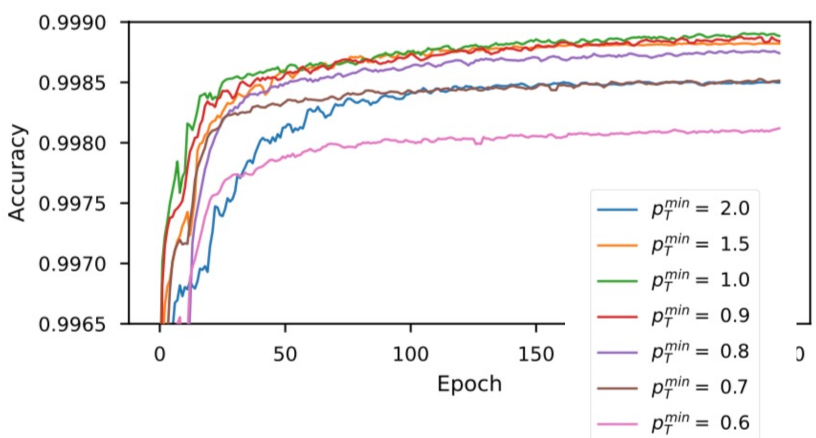
Trained with standard BCE loss

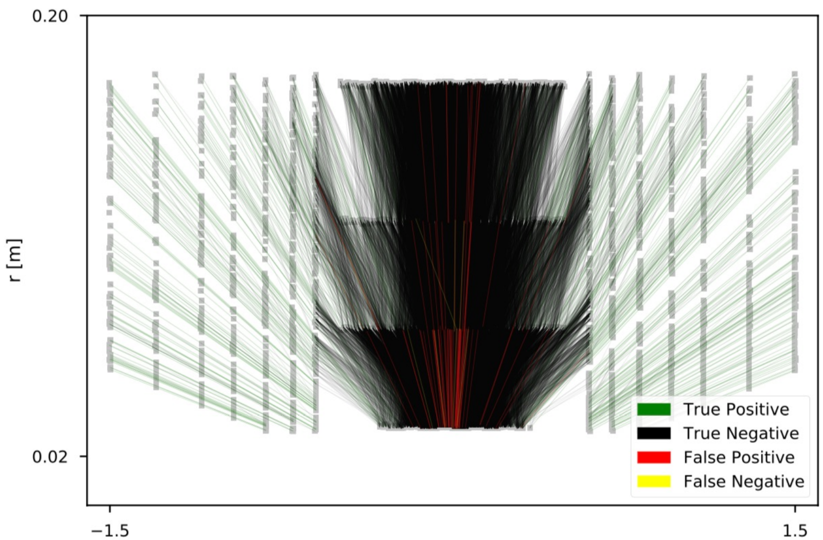
$$\mathcal{L}_w(y_j, w_j) = -\sum_{j=1}^{|\mathcal{E}|} \left(y_j \log w_j + (1 - y_j) \log(1 - w_j)\right)$$

Our Paper, Original Paper

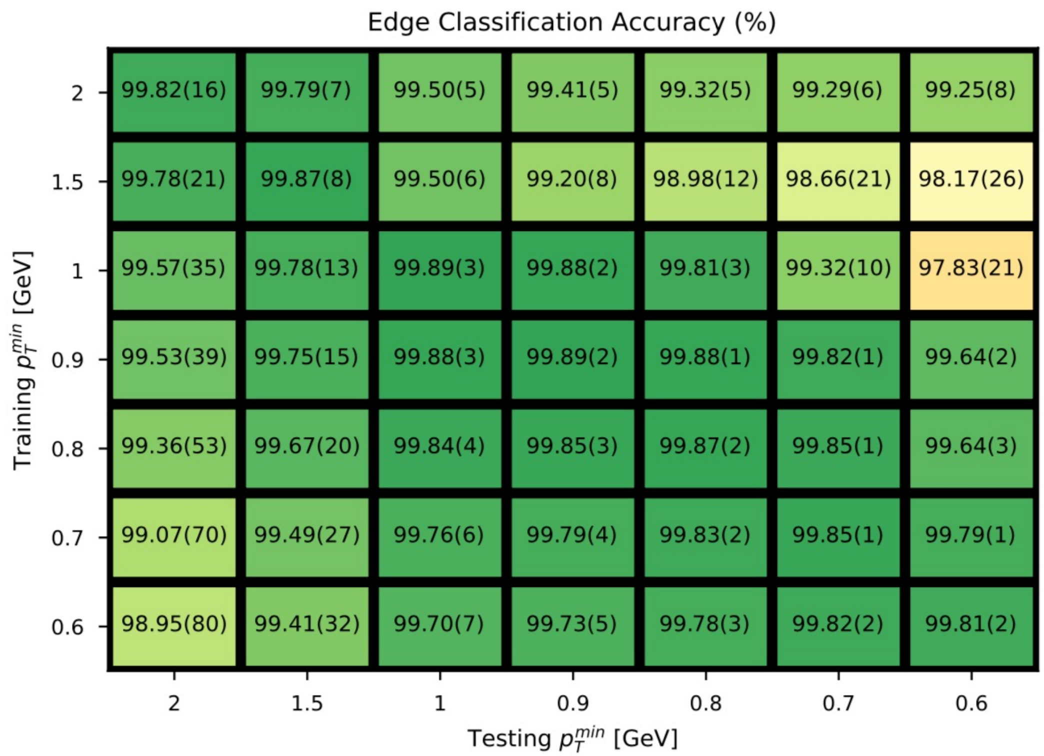
IN Edge Classification Peformance

Models trained and tested on a range of graph pt





pt=1GeV

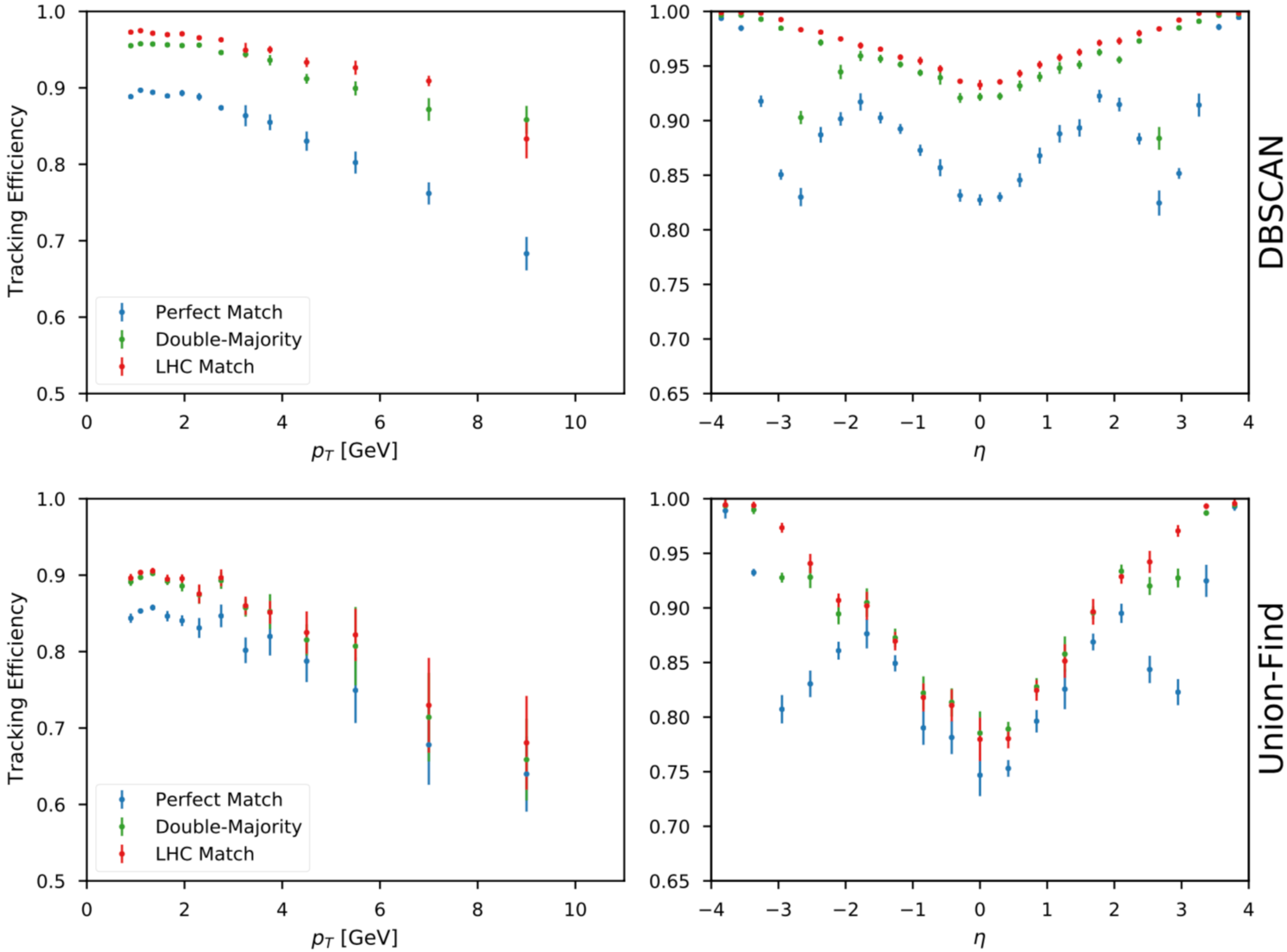


Results:

- 99.9% edge efficiency for matching pt
- 97.8-99.8% edge efficiency for non-matching pt

IN Tracking Performance

Compared 2 methods to group selected edges into track candidates



- **LHC match**: cluster contains >=75% hits from same PID
- **Double-majority**: cluster >= 50% hits from same PID and >=50% of that PIDs hits
- **Perfect match**: cluster contains all hits from 1 PID and only hits from that PID

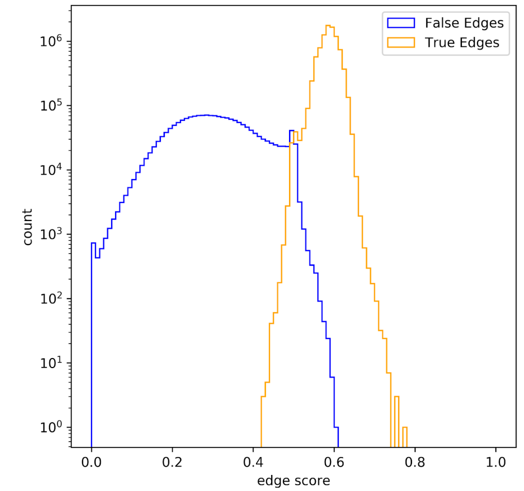
Results: 70-99% tracking efficiency

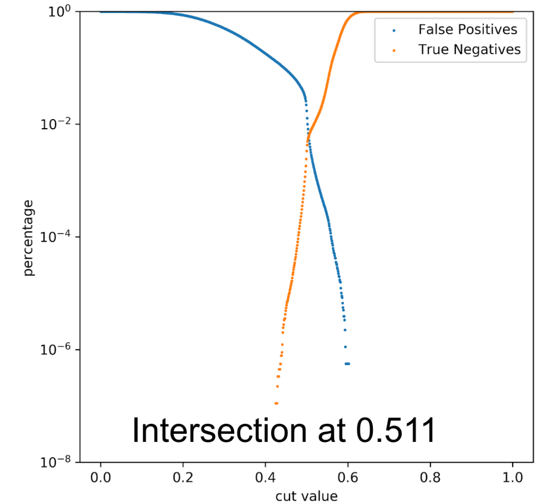
Edge Classifier Network

- Encoder creates an initial embedding of the graph
- Graph modules combine edge and node convolutions
  - Previous graph embeddings are propagated to following modules
  - Total 260k parameters
- Uses phi reflected graphs in training
  - Intuitive data augmentation

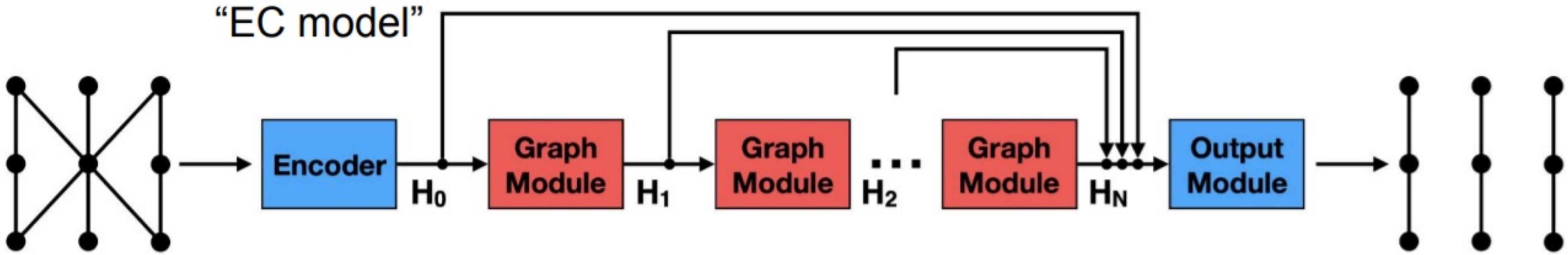
Confusion Matrix

0.9842	0.0037
0.0158	0.9963

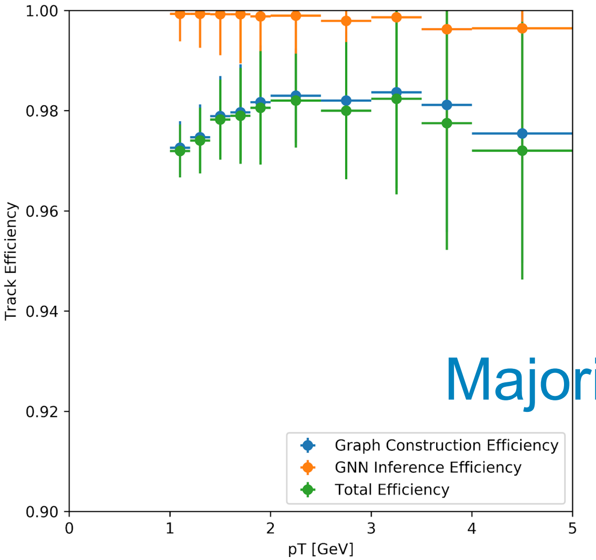


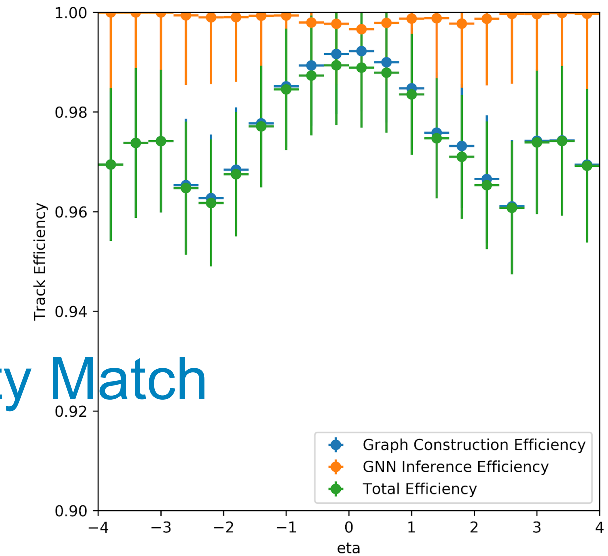


"EC model"



EC Tracking Performance

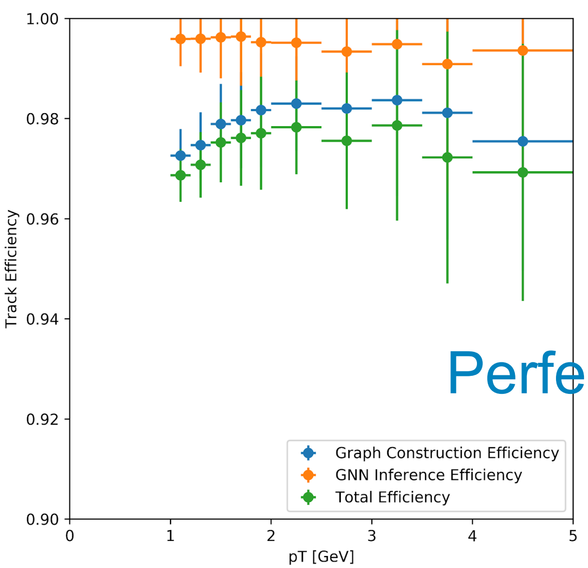


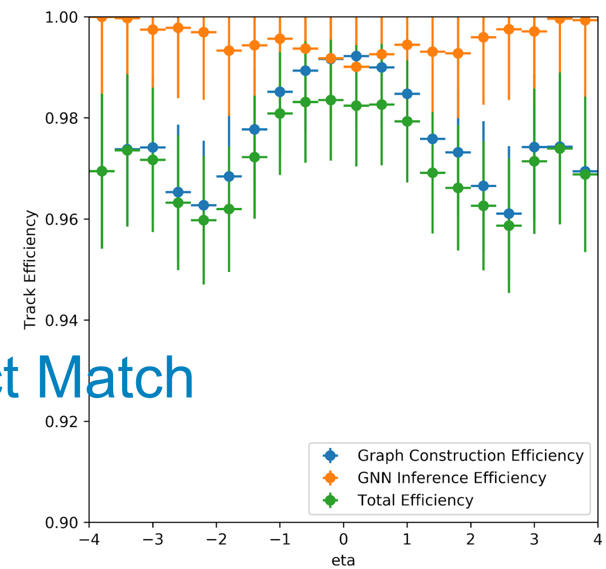


Majority Match

Graph Construction Efficiency	0.977068
GNN Inference Efficiency	0.999101
Total Efficiency	0.976190

- Majority match: cluster contains >= 50% hits from same PID
- Perfect match: cluster contains all hits from 1 PID and only hits from that PID





Perfect Match

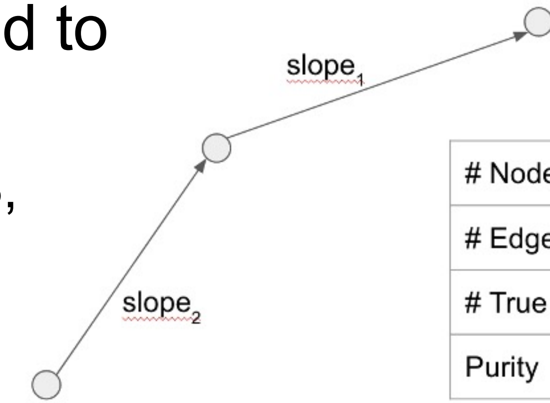
Graph Construction Efficiency	0.977068
GNN Inference Efficiency	0.995663
Total Efficiency	0.972831

Results: 96-98% tracking efficiency (with 1 GeV pt cut)

Optimization + Experiment Studies

Graph Construction Optimization

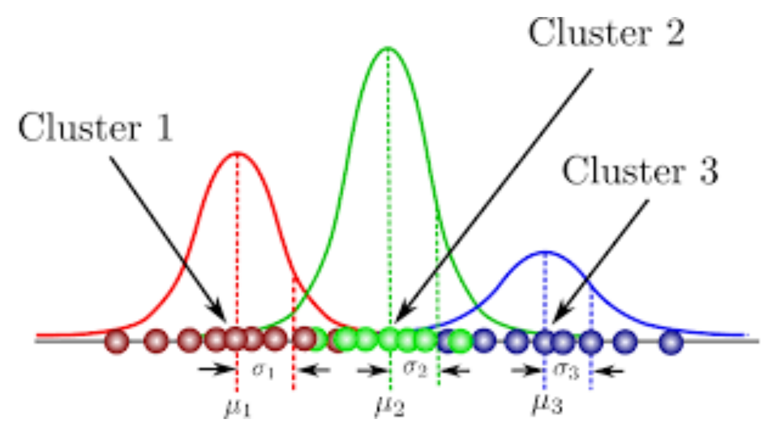
- Can expand module map method to define allowed triplets
  - Optimized cuts: |Δ ϕ-slope| < .00023, |Δ z-slope| < .1
  - ~doubles graph purity!



# Nodes	9950.4
# Edges	220007.1
# True Edges	18275.9
Purity	83.05%

- Studying graph segmentation to better enable parallel processing and resource constrained inference
  - With Gaussian Mixture Models we're able to separate ~60% of tracks into their own clusters during graph construction!

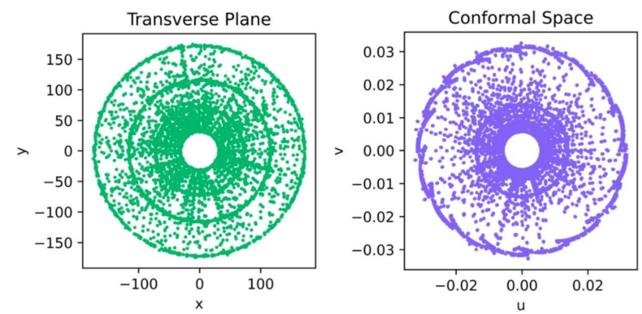
Dataset	Method	$e_{TrackML}\uparrow$	$e_{sc-PDB}\uparrow$	$\chi_{TrackML}\uparrow$	$\chi_{sc-PDB}\uparrow$
DBSCAN	TrackML	0.579	-	0.7424	-
	sc-PDB	-	0.481	-	0.2863
Spectral Clustering	TrackML	0.602	-	0.5968	-
	sc-PDB	-	0.517	-	0.4262
Dynamic kNN	TrackML	0.513	-	0.5079	-
	sc-PDB	-	0.594	-	0.5038
GMM	TrackML	0.735	-	0.8194	-
	sc-PDB	-	0.408	-	0.3920

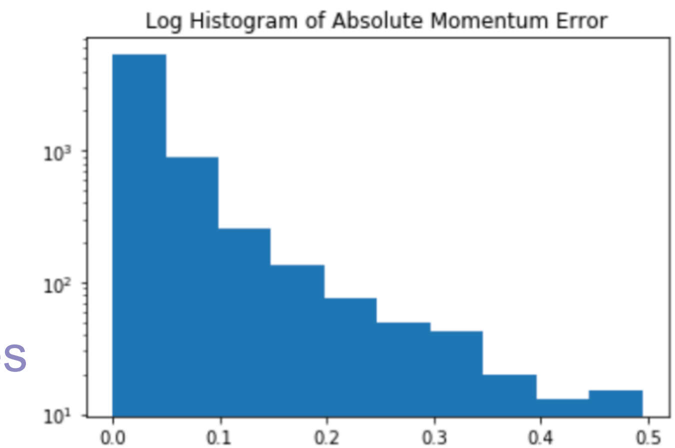


Graph Segmentation Paper

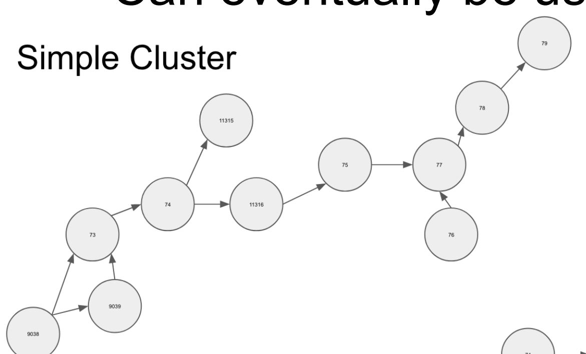
Track Building Optimization

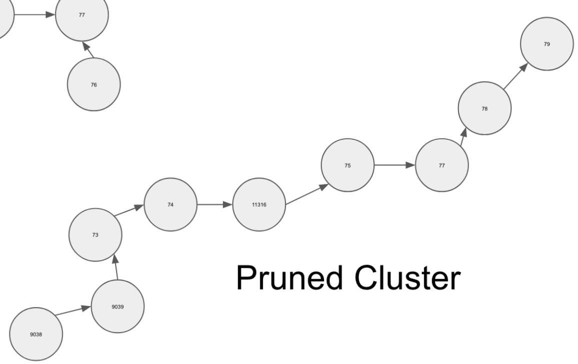
- Walkthrough Method: walkthrough track cluster where nodes have multiple neighbors, find longest path, prune nodes not included in longest path
  - Provides small improvement to tracking efficiency, critical to track fitting
  - Could eventually use pruned nodes to develop additional candidates
- Developing fast conformal space track fitting to further characterize GNN performance
  - Can eventually be used in 'one-shot' architectures





Simple Cluster





Pruned Cluster

Tracks lie on circles in the transverse plane:

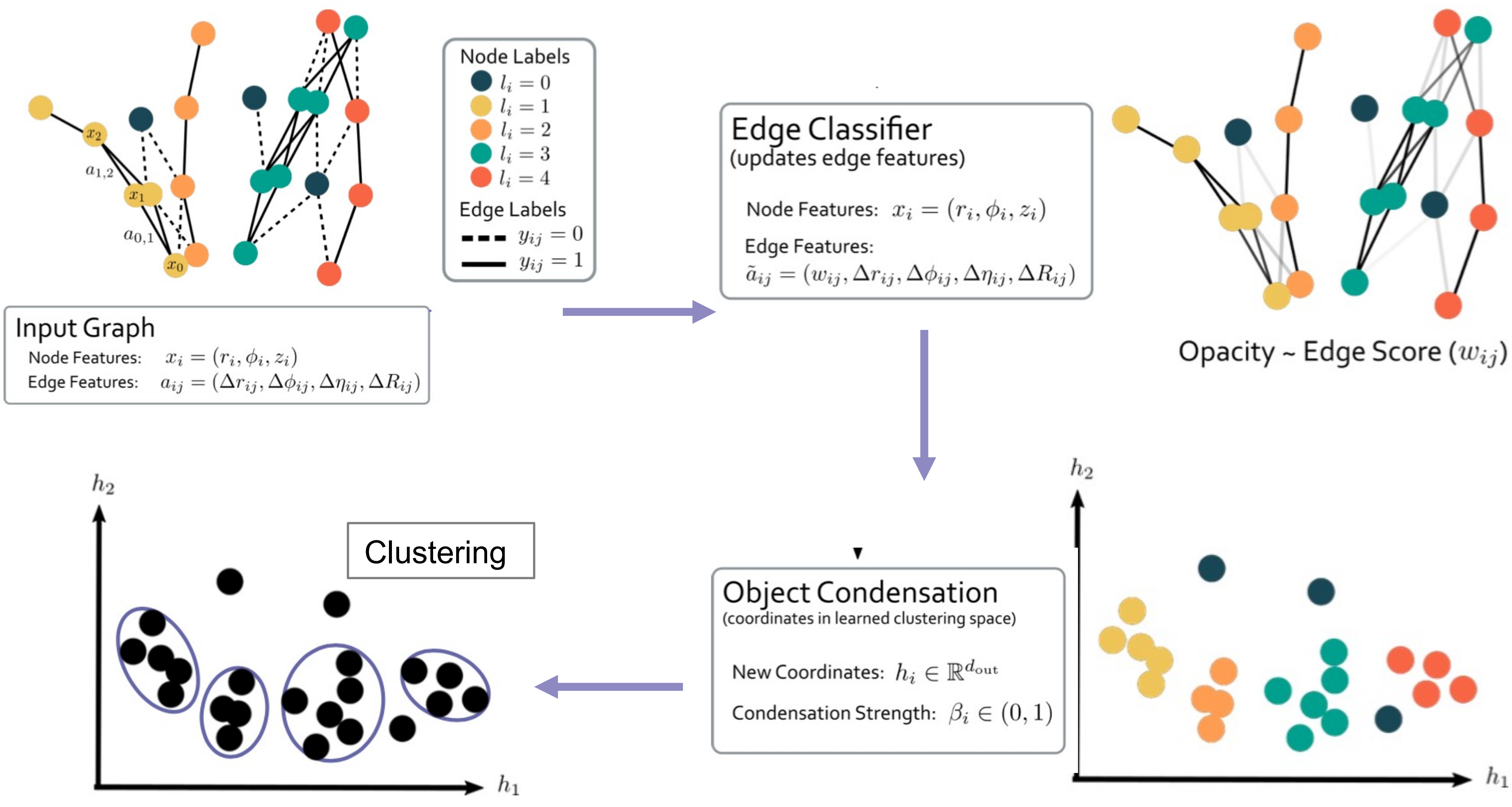
$$R^2 = (x-a)^2 + (y-b)^2$$

A conformal map makes the circles in the x-y plane into straight lines in the u-v plane:

$$u = \frac{x^2}{x^2+y^2} \qquad v = \frac{y^2}{x^2+y^2}$$

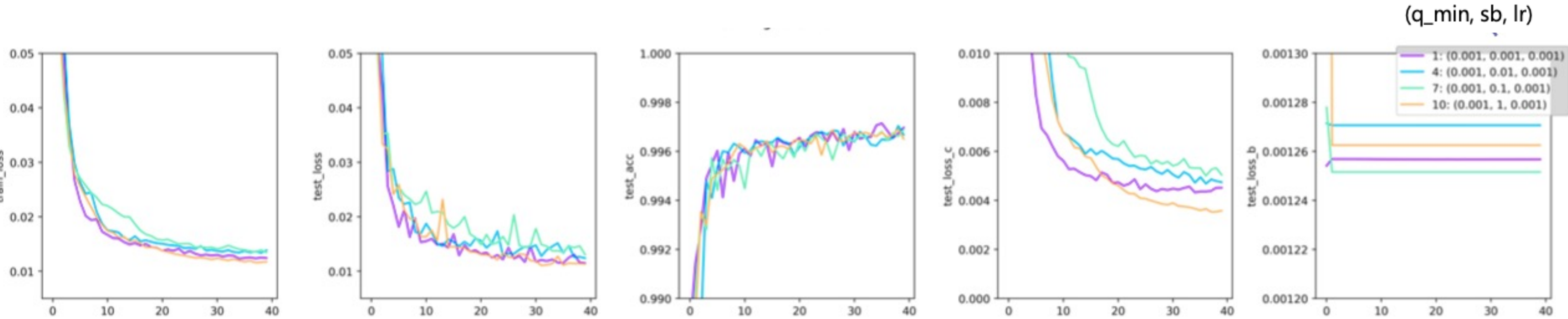
Object Condensation

Can we improve tracking performance of small(er) networks?



O(50k) learnable parameters

Object Condensation: Initial Performance



Train Loss	Test Loss	Edge Classification Accuracy	Attract/Repulse Loss	Background Loss
Smooth convergence, no sign of overtraining	Smooth convergence, no sign of overtraining	Converges to ~99.7% (consistent with arXiv:2103.16701)	Learned clustering coordinates exhibit attract/repulse behavior	Background assignment falls into minimum for best solutions

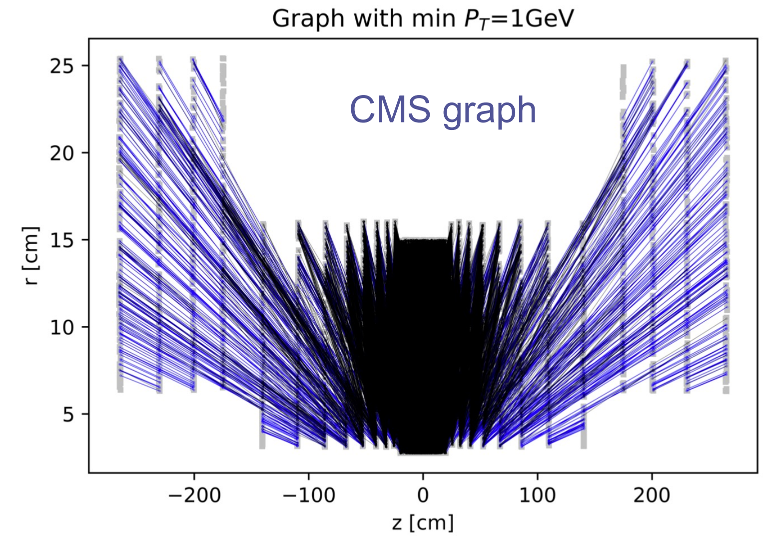
TRACKING EFFICIENCIES
AVERAGED ACROSS ~10^4 GRAPHS

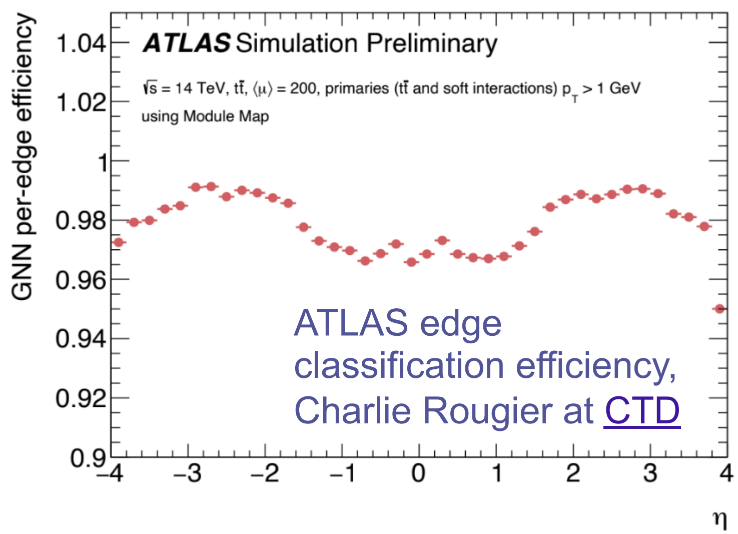
- Per-graph summary
  - Perfect Match Fraction: 0.827
  - Double Majority Fraction: 0.932
  - LHC Loose Fraction: 0.890

$\lvert\eta\rvert$	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

Experiment Integrations

- CMS ML group hosted a hackathon to begin integrating GNN tracking into CMSSW
  - Developed tracker data ntupilizer to dump information
  - Implemented graph building in C++, used Triton to run GNN inference, used existing DBScan implementation to build tracks
- Princeton students working on optimizing IN for CMS data
- UIUC group optimizing EC and IN for ATLAS data
  - Successful initial results obtained, presented within experiment and similar results presented at CTD
  - Has informed planning around EF tracking for HL-LHC





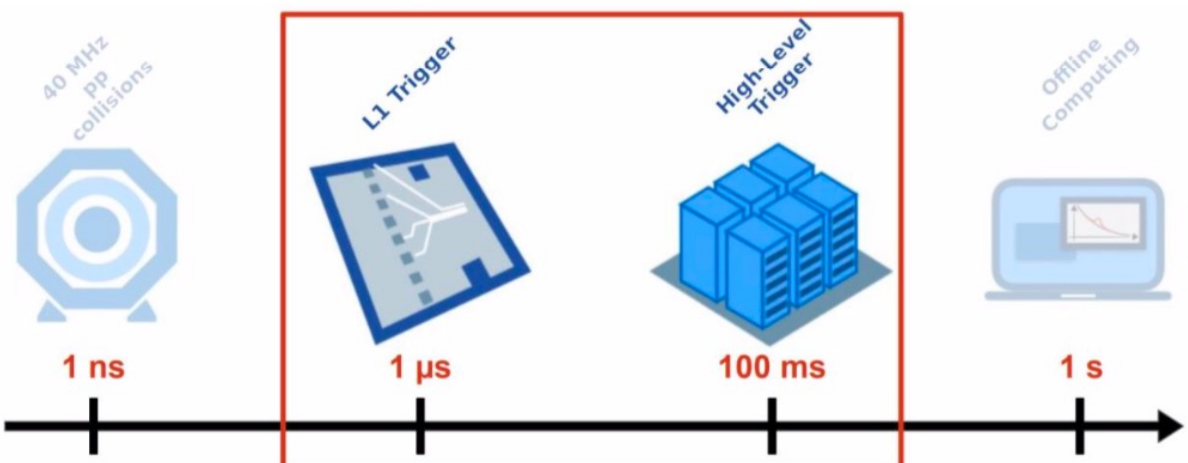
ATLAS edge classification efficiency, Charlie Rougier at CTD

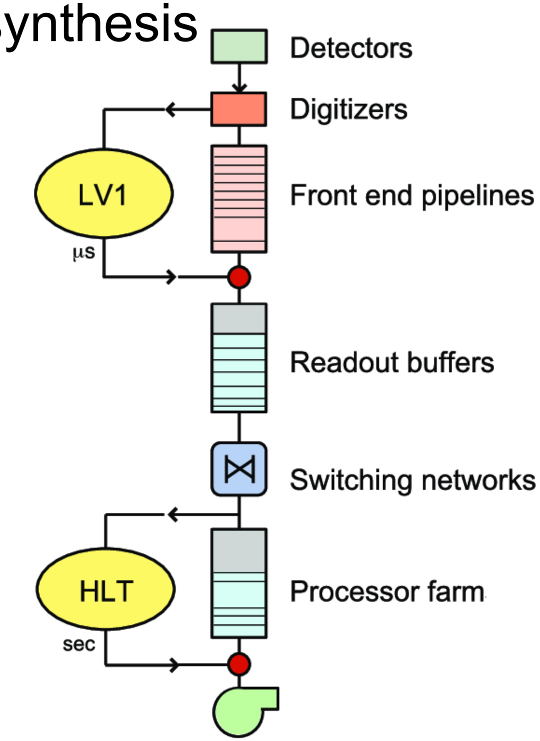
Related, On-going, and Future Work

Accelerated GNN Tracking

Strong interest in accelerating these algorithms with FPGAs

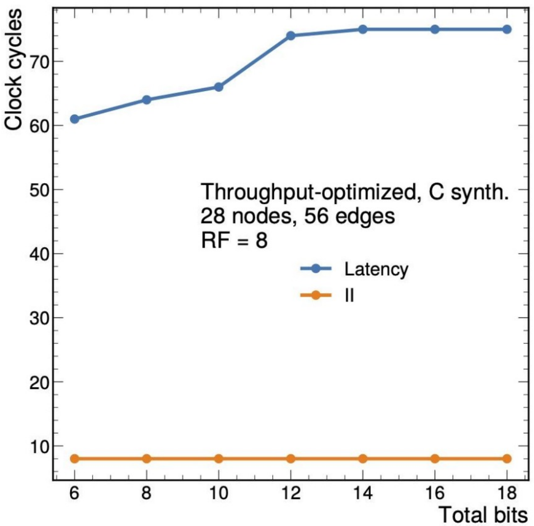
- Reduce compute time and energy utilization
- Possibly enable use at the trigger level in experiments
- Two complimentary acceleration studies
  - Using GNNs directly on hardware via high level synthesis
    - Using HLS4ML framework
    - Potentially suitable for L1
  - Using FPGAs as a co-processor with CPU
    - Potentially suitable for HLT

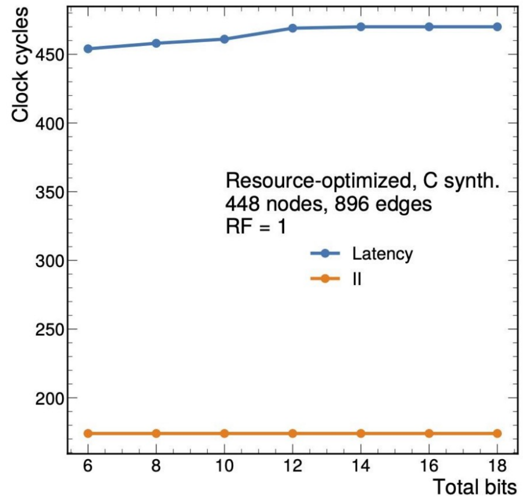


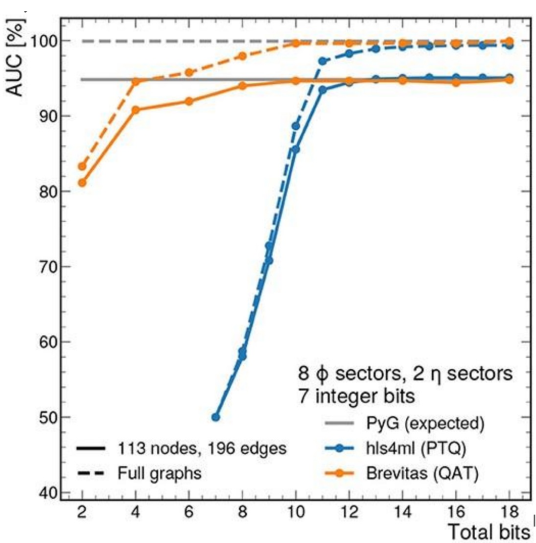


[Our Recent Paper]

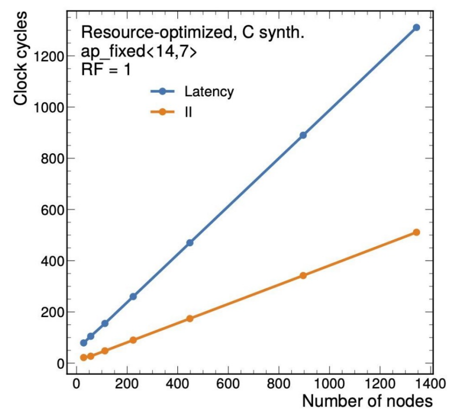
HLS4ML Study



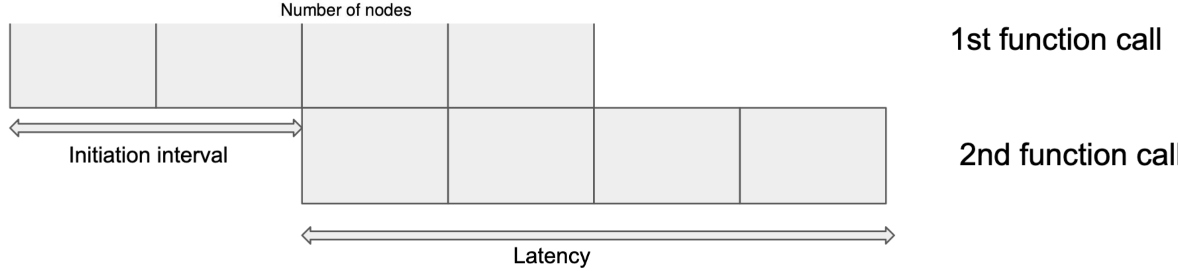




16

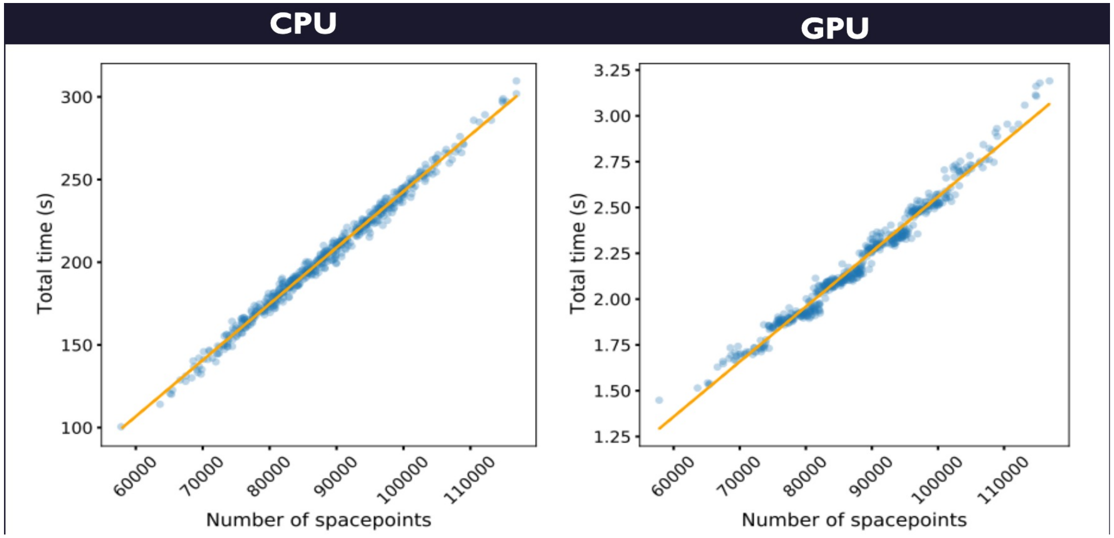


- First hls implementation of GNN blocks!
- Bit precision scan compares physics performance vs resource needs
- Reuse factor controls amount of pipelining
  - Trade-off between latency and resource utilization



Next Steps in Acceleration

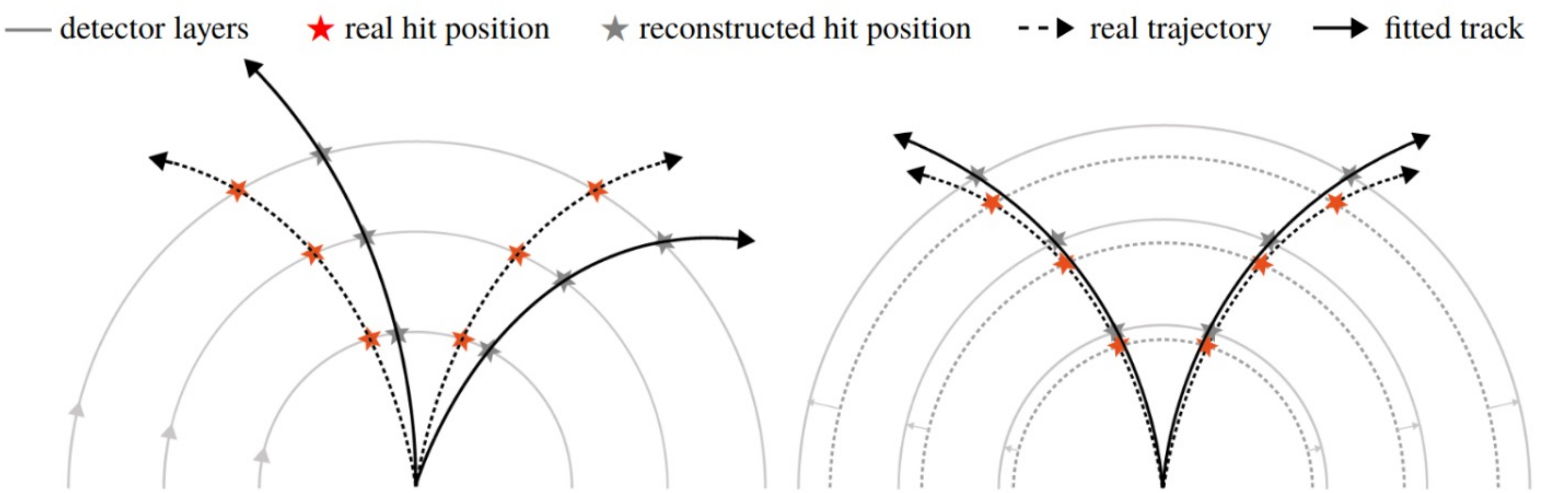
- Throughput optimized implementation achieves <1 μs latency
  - Could be suitable for L1 trigger!
  - Study scaling to larger graphs (currently max 28 nodes/56 edges)
- Need to develop implementations of graph building and track segment linking on accelerators
  - How to handle data flow between different pipeline components
- Complimentary studies on GPU based GNN acceleration
- Many applications of this work to other areas of research and industry!



One Shot Architectures

Can we incorporate track fitting directly into a GNN pipeline?

- Could apply conformal or helical fit after inference
  - Helical fits are resource intensive, conformal fits can be hard to tune
- Could add term for track parameter prediction to loss function
  - Avoids having to actually fit tracks but balancing loss terms can be difficult
  - Particularly interesting for instance segmentation approaches



Conceptual paper on instance segmentation

On-going + Future Tracking Studies

- Optimize parameters of graph construction algorithms
  - Compare different spaces for graph construction
  - Optimize graph segmentation (and post segmentation relinking)
  - Study training on ‘messy’ graphs, inference on ‘clean’ graphs
- Improve existing architectures
  - Include external effects in IN, **improve edge classification in barrel**, conformal space…
  - Alternate shapes for localization in instance segmentation + train end-to-end
  - Explore additional clustering/track building algorithms (include edge weights)
- New ideas
  - Enforce E(3) or other equivariance
  - Add track parameter prediction learning task to existing architectures
  - Alternative architectures (accumulation or message passing nodes, new graph embeddings)
- Further characterize acceleration and potential for use in trigger
  - Full FPGA-based tracking pipeline
  - Use graph segmentation studies for parallelization

Conclusions

- Graphs are a natural representation of particle detector data
- Graph-based learning methods can leverage geometric information for effective reconstruction
  - Many different GNN approaches and architectures can work, important to define cohesive evaluation metrics and benchmarking processes
  - Many techniques/insights from GDL, ML, etc can help improve different components of the pipeline
- GNN inference can be accelerated with dedicated hardware
  - Many tradeoffs to consider
- Geometric deep learning is synergistic with particle physics
  - There are many open questions still, including how to best collaborate and information share with other ML researchers
  - Open datasets can help!
- **Many thanks to my wonderful collaborators!**
  - Gage DeZoort, Javier Duarte, Abdel Elabd, Aneesh Heintz, Vesal Razavimaleki, Isobel Ojalvo, Markus Atkinson, Mark Neubauer, Rajat Sahay, Dominika Krawiec, and the ExaTrkX Collaboration!

Thank you!

Happy to answer any questions!

sthais@princeton.edu @basicsciencesav