GNNS FOR TRACKING

Savannah Thais (on behalf of many others)

CMS ML Forum

09/30/2020

iris hep

Princeton
Research Computing

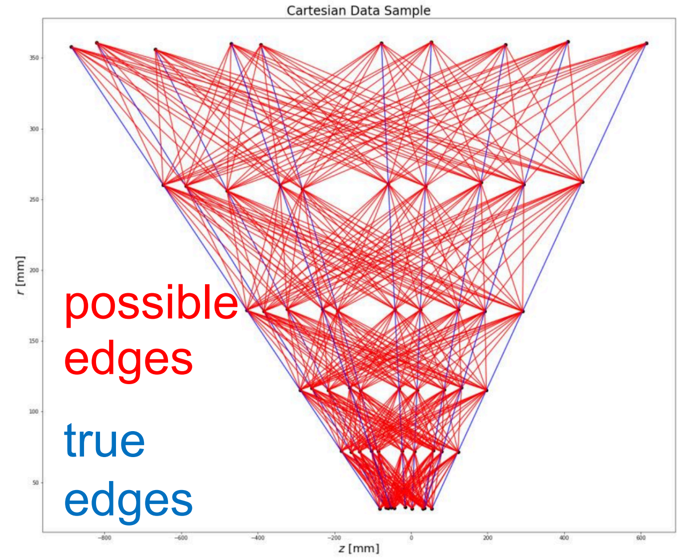
Outline

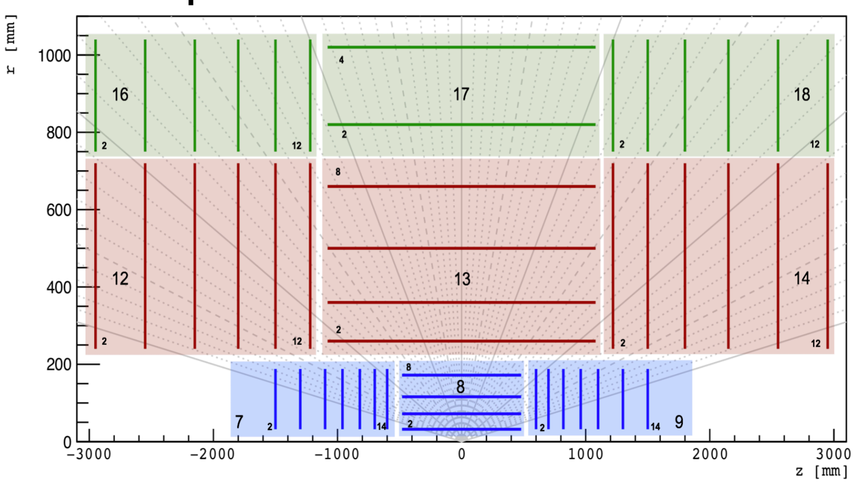
1. Introduction to tracking with GNNs
2. Edge classifiers
3. Data processing
4. New architectures + studies

'Traditional' Tracking with GNNs

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

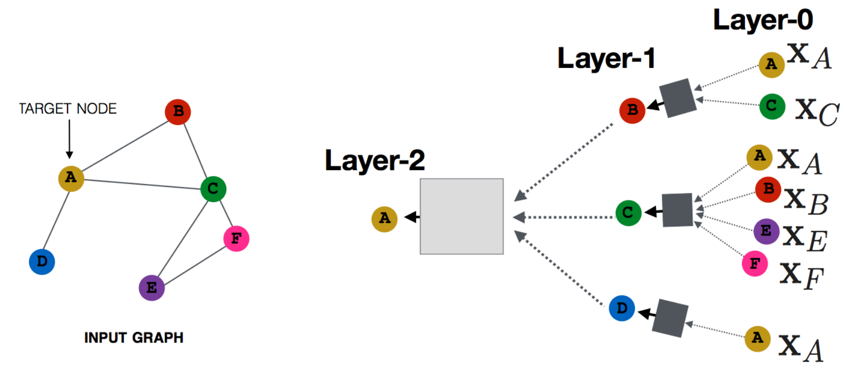


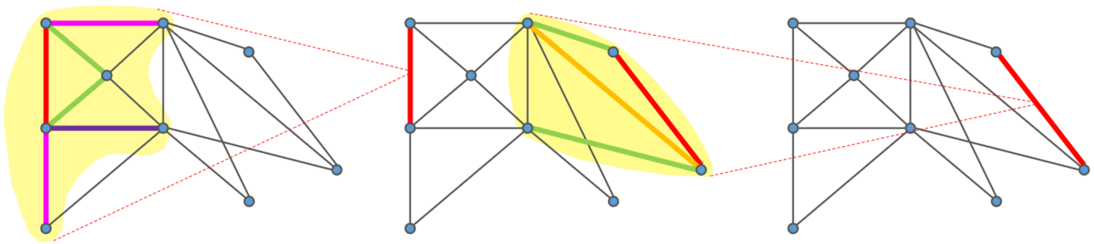


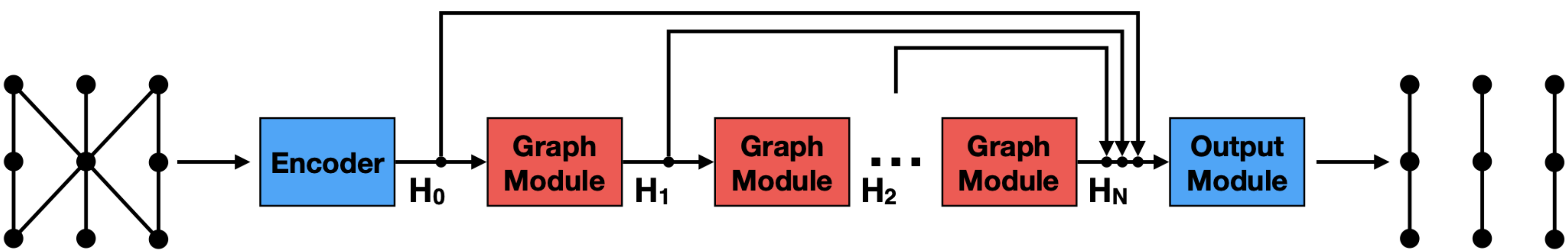
- Many places to improve/innovate
  - Graph construction, architectures, data augmentation…
- Most work shown here uses [TrackML dataset]
  - Open, experiment agnostic
  - Has 'score' functionality to compare models

Edge Classifiers

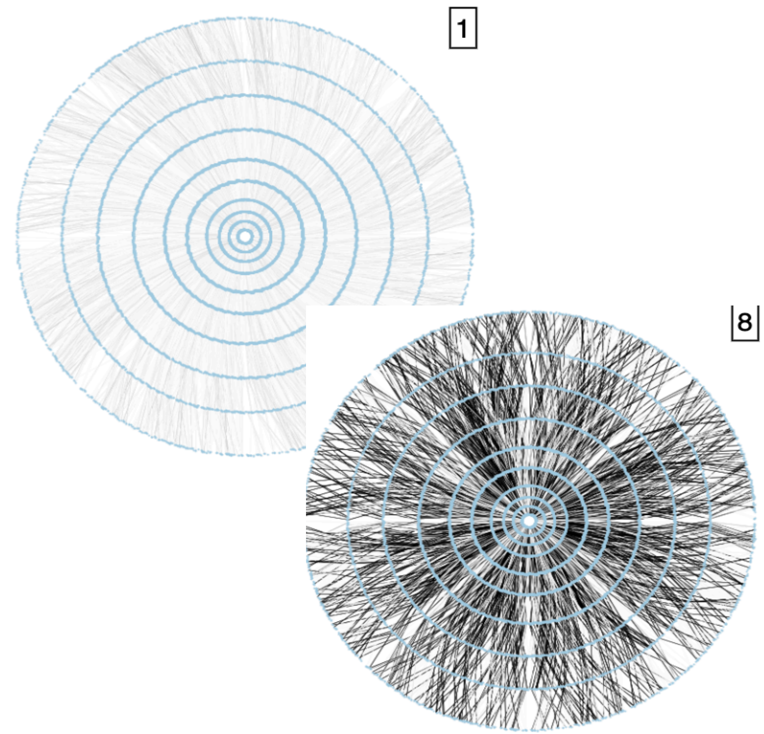
- Graph Modules are core component:
  - Run node and edge convolutions
  - Update features of both
  - Each message passing function is a FCN
- Graph modules are often recursively connected
  - Allows aggregation of progressively more distant information
  - Weights can be shared across modules





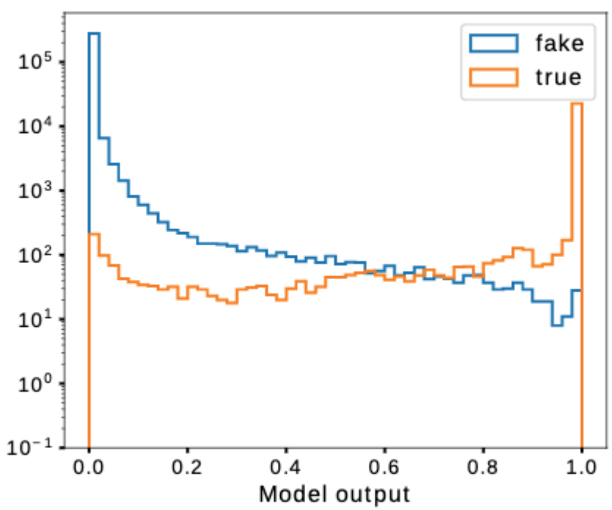


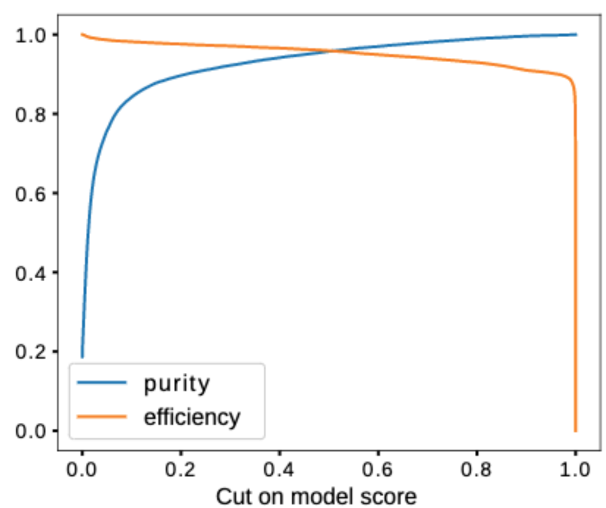
Proof of Principle



NeurIPS 2020 ExaTrkX architecture:

- Node and edge features embedded in latent space
- 8 graph modules with shared weights
- Initial embeddings concatenated at each module
- Each FCN has 128 hidden features and ReLU activation





Results:

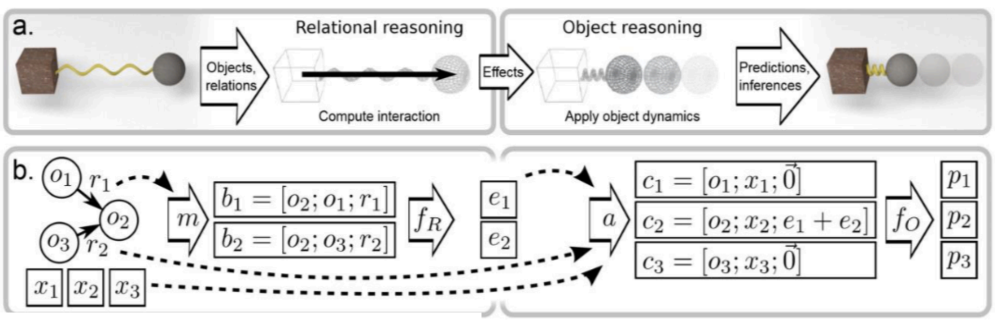
- 95.9% edge efficiency
- ~95% track finding accuracy

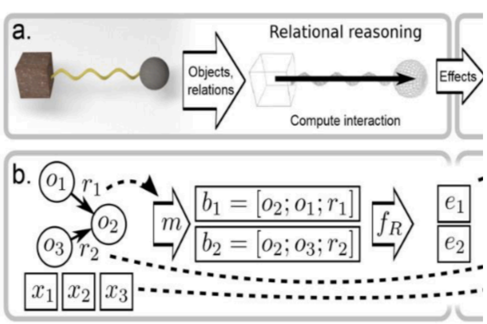
Paper

Interaction Networks

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

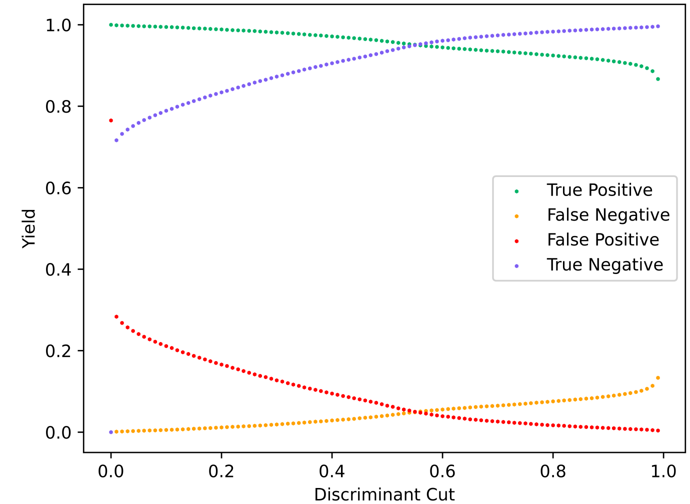
- Relation and object models are FCNs
- Total of 89,400 parameters (smaller than previous architecture)

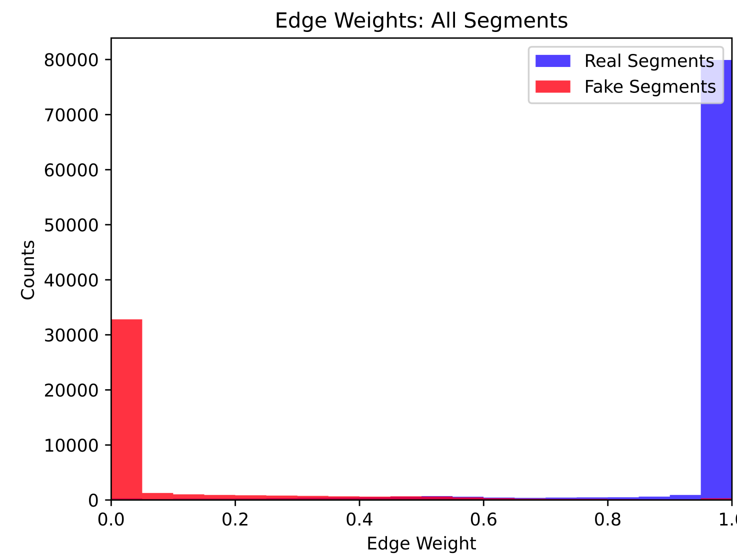




Results:

- 95% edge efficiency
- Tracking efficiency still being measured





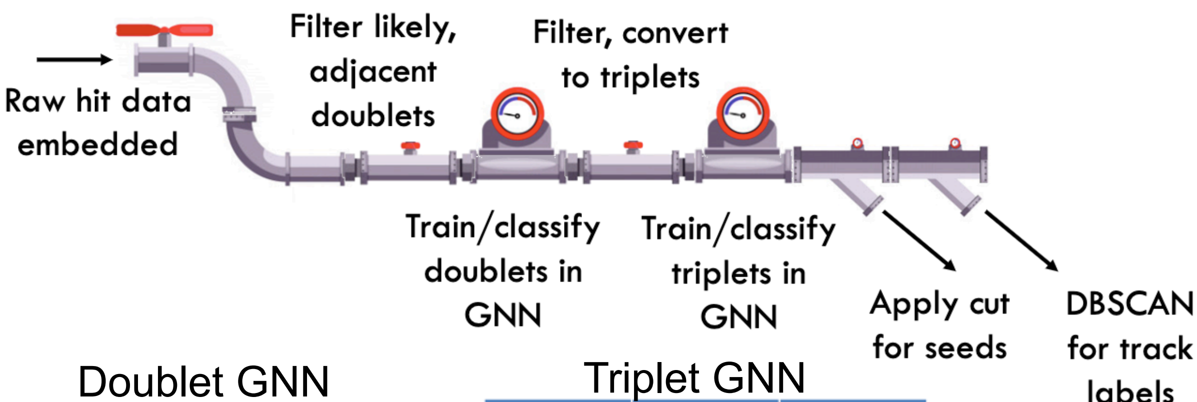
Confusion Matrix: (cut=0.60) [[0.948 0.052]
[0.053 0.947]]

Paper, Recent Talk

Embedding

Improve graph efficiency by embedding features

- Embed features in N-dimensional space where hits from same tracks are close to each other
- Score “target” hit within embedding neighborhood against “seed” hit at center
- Filter by score to create seed-to-target doublets, doublets form the graph
- Can repeat with embedding triplets as edges, creating ’n-plet’ graphs

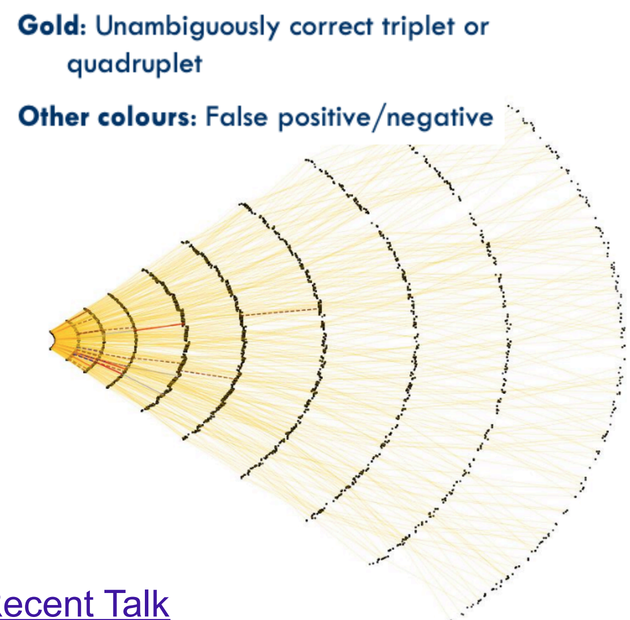


Doublet GNN

Threshold	0.5	0.8
Accuracy	0.9761	0.9784
Purity	0.9133	0.9694
Efficiency*	0.9542	0.9052

Triplet GNN

Threshold	0.5	0.8
Accuracy	0.9960	0.9957
Purity	0.9854	0.9923
Efficiency*	0.9939	0.9850



[Recent Talk]

Graph Construction

Optimizing graph construction can help GNNs learn effectively

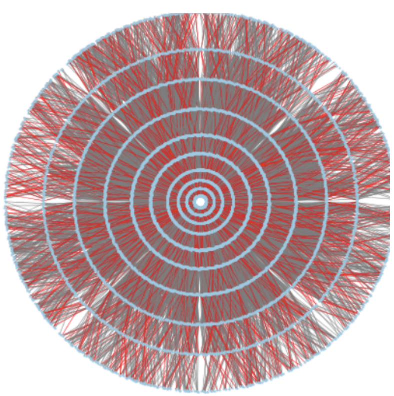
- Edge efficiency: true edges/all edges
- Truth efficiency: true edges in graph/all possible true edges

‘Current’ Methods

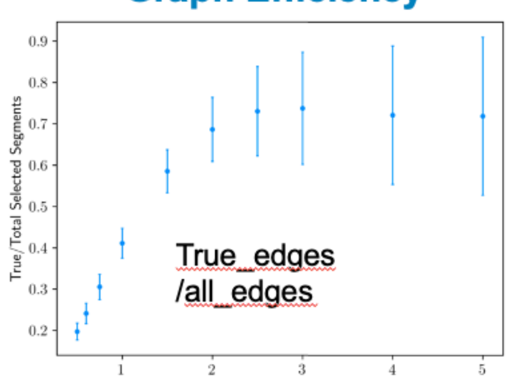
- Layer pairs: create edges between nodes in adjacent layers within a $\Delta\phi/\Delta r$ range
- Layer pairs+: allow edges within a layer
- kNN: form edges between a hit and its k closest neighbors (can customize distance metric)

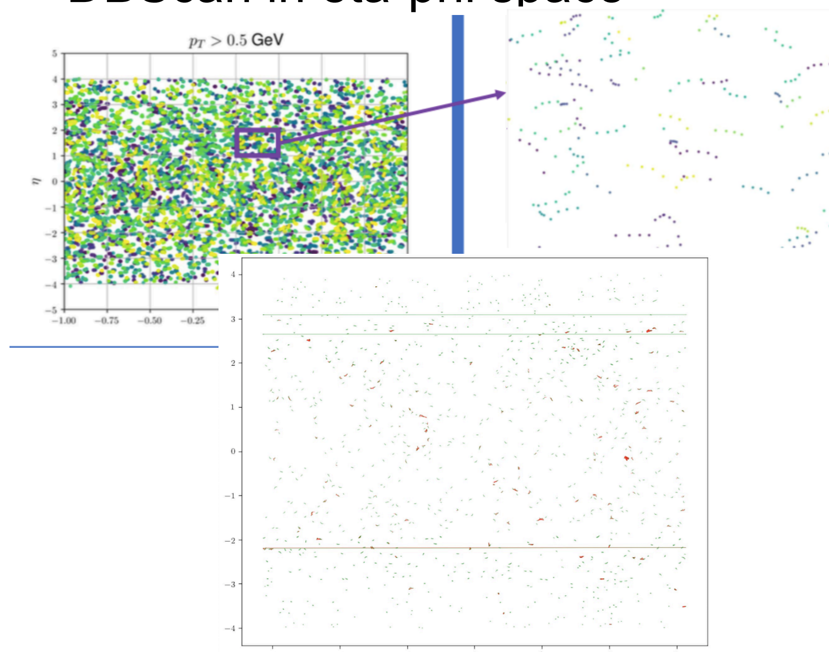
Exploratory Methods

- Dynamic kNN
- Learned clustering
- DBScan in eta-phi space



Graph Efficiency

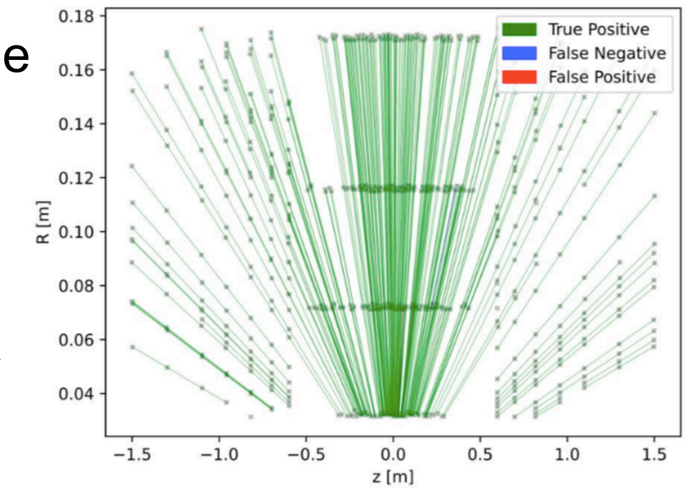




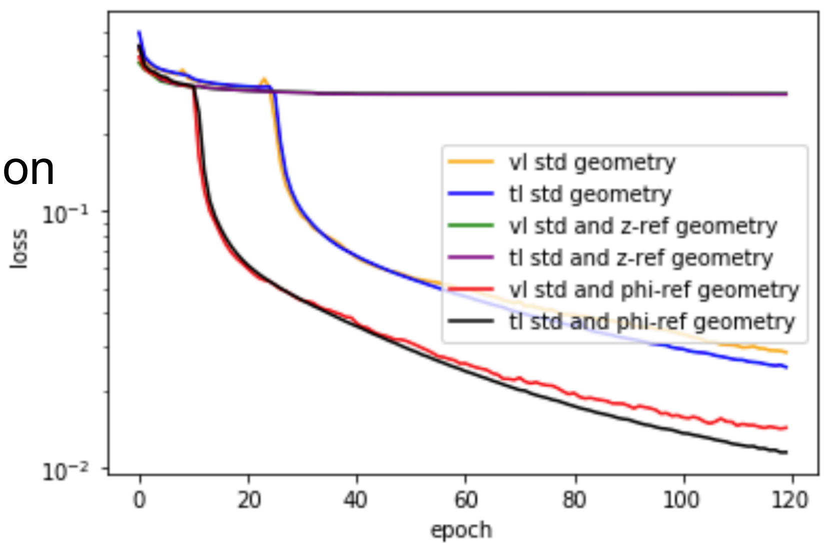
Data Augmentation

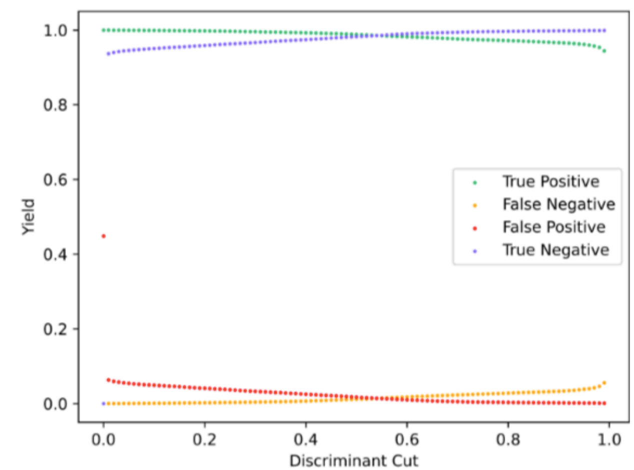
- Including endcaps:
  - Difficult in layer pairs construction due to edge ordering
  - Initial studies in pixel detector only, typically improve edge efficiency
- Dropping layers from graph construction
  - Reduce size of graph while maintaining track finding efficiency
- Applying z and phi reflections
  - Break symmetry of detector to possibly enhance learning

Pixel IN with Endcaps



Data Augmentation with Edge Classifier





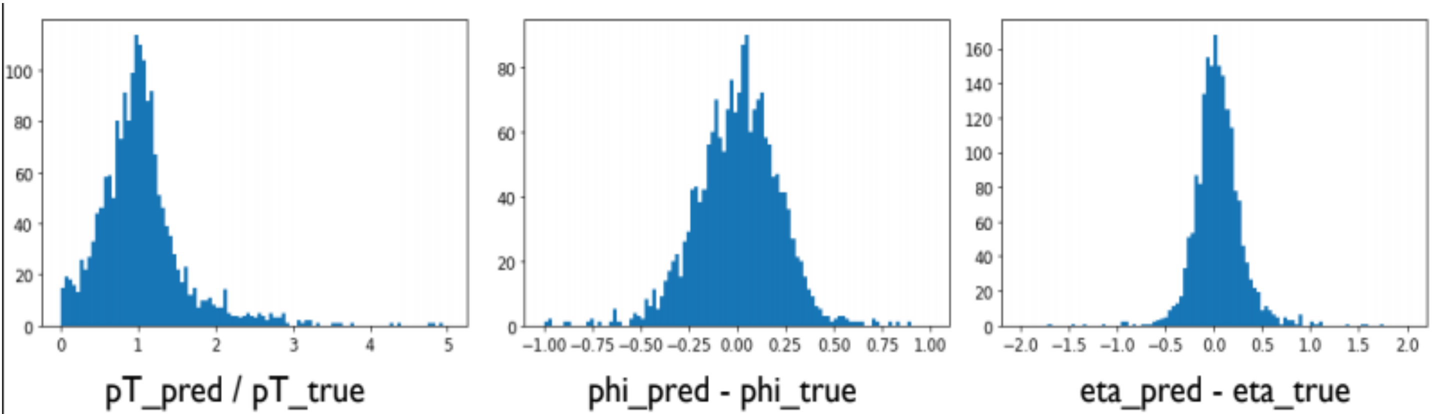
On-going Studies

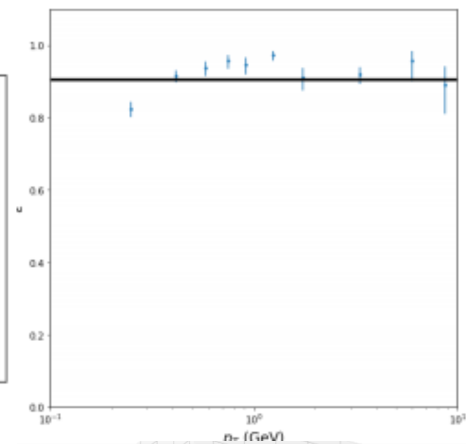
- Optimize parameters of existing graph construction algorithms and explore new ones
- Refine track formation algorithm (currently Union Find or DBScan)
- Improve existing architectures
  - Include external effects in IN, optimize embedding…
- Test other GNN architectures
  - Instance segmentation, GraphSAGE, Spectral Convolutions…
- Additional data augmentation
  - Endcaps for full detector, other transforms…
- New ideas
  - Timing information, one-shot tracking, conformal space…

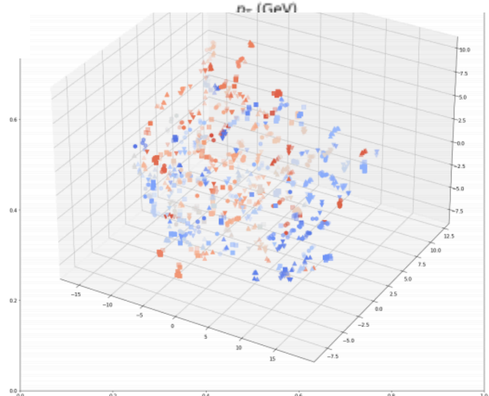
One-Shot Clustering Network

Form graphs, cluster hits, and fit tracks with one algorithm

- DGCNN-based embedding (hinge loss to truth centers in latent space)
- Edge classifier in latent space (cross-entropy binary loss)
- Union-find over edges to form clusters, predict track properties (pt, eta, ph) from clusters (MSE loss)







Initial study on 10 events with 200 tracks each

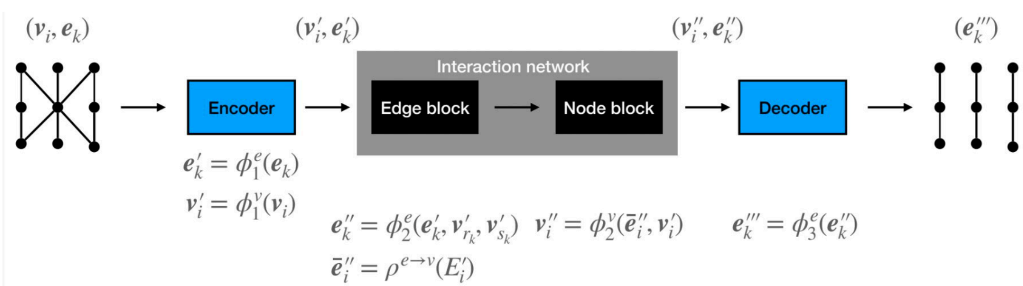
- Track efficiency measurement using maximum IoU
- Fake rate ~20%, truth efficiency ~90%

Recent Talk

Accelerated GNN Tracking

Strong interest in accelerating these algorithms with GPUs or FPGAs

- HLS4ML implemented a 1 iteration version of IN for FPGA
- Princeton group optimizing OpenCL IN on FPGA
- Lindsey Gray implemented Jit functionality for Pytorch geometric
  - Allows serializable GNNs



Latency (cycles)		Latency (absolute)		Initiation Interval (cycles)	
min	max	min	max	min	max
377	377	2.226 us	2.226 us	46	46

== Utilization Estimates

* Summary:

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	2442	-
FIFO	321	-	9350	25516	-
Instance	46	143	177007	826482	-
Memory	-	-	-	-	-
Multiplexer	-	-	-	5418	-
Register	-	-	604	-	-
Total	367	143	186961	859858	0
Available SLR	2160	2760	663360	331680	0
Utilization SLR (%)	16	5	28	259	100
Available	4320	5520	1326720	663360	0
Utilization (%)	8	2	14	129	0

Recent HLS4ML Talk

Conclusions

- GNNs are extremely promising for LHC tracking
  - Geometric data representation with variable number of inputs
- A variety of architectures have been shown to work
  - Focus is now on refining and optimizing
- Graph construction (and embedding) is critical to performance
- Working towards accelerating graph algorithms for use at HL-LHC
  - Possibly at trigger level