GRAPH BASED PARTICLE TRACKING

Savannah Thais

IRIS-HEP Topical Meeting

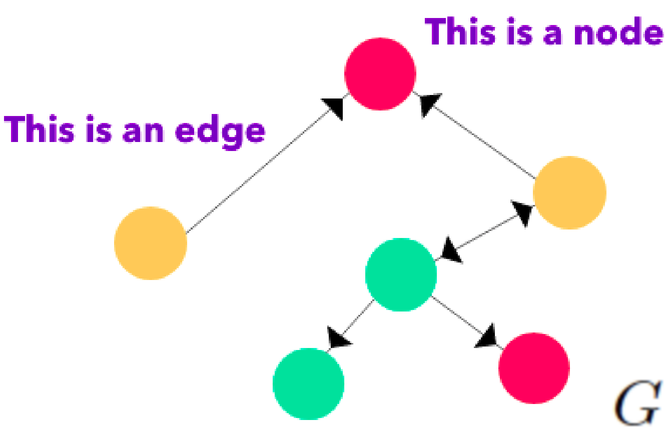
03/01/2021

iris hep

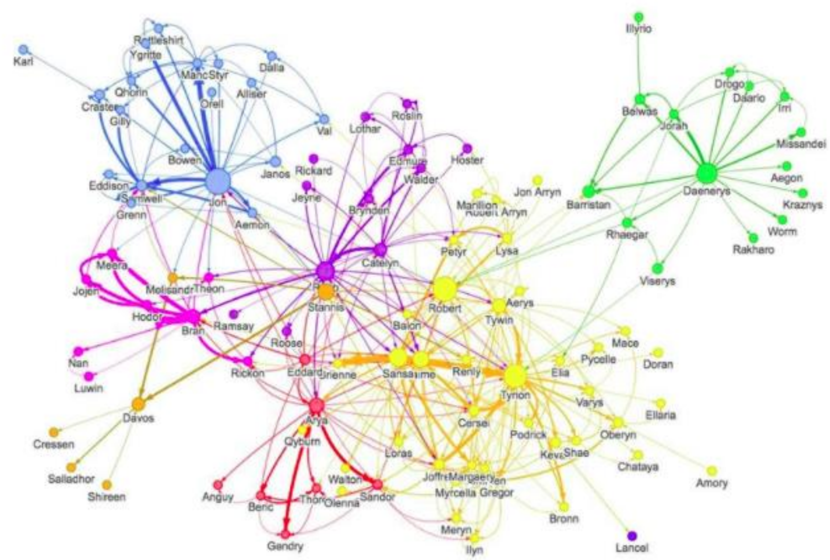
Princeton
Research Computing

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
- Graphs can be multilevel (nodes are encoded graphs)

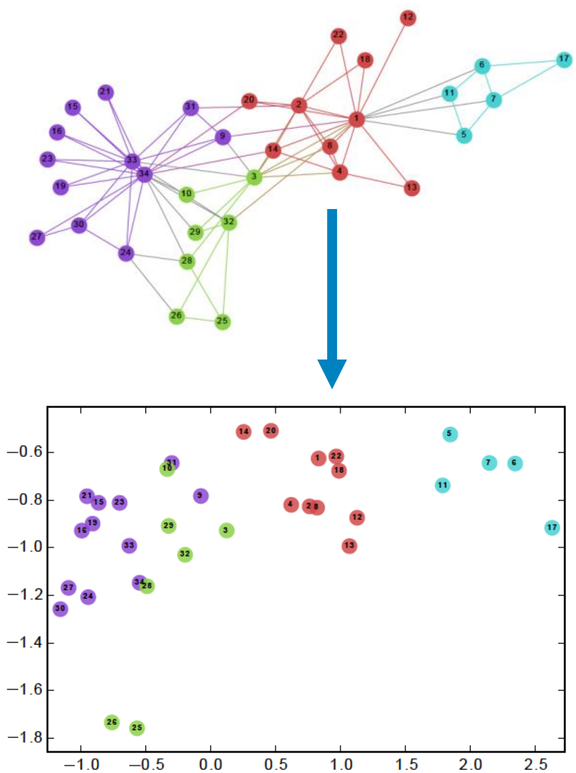


$$G = (V, E)$$



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



Initial "layer 0" embeddings are equal to node features

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

previous layer embedding of v

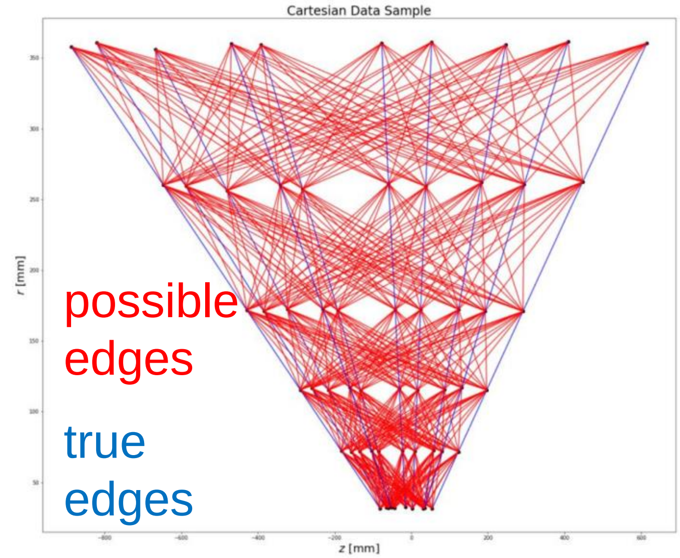
$$\mathbf{h}_v^k = \sigma\left(\mathbf{W}_k \sum_{u \in N(v)} \frac{\mathbf{h}_u^{k-1}}{|N(v)|} + \mathbf{B}_k \mathbf{h}_v^{k-1}\right), \quad \forall k > 0$$

kth layer embedding of v

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

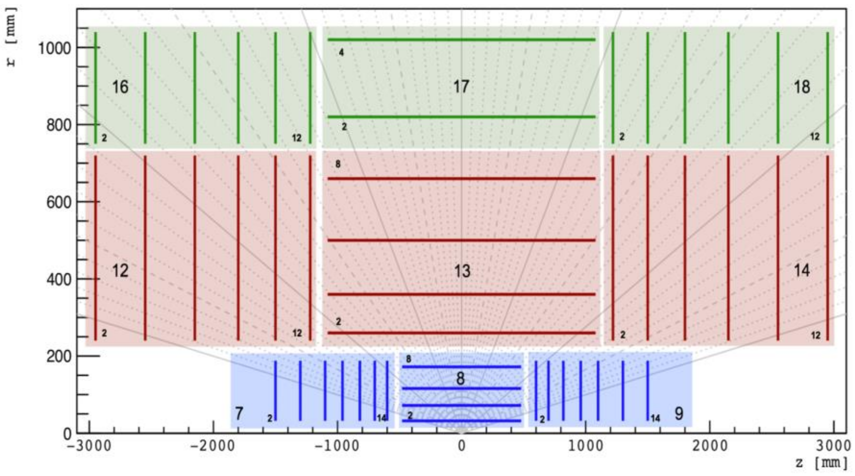
average of neighbor's previous layer embeddings

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

Graph Construction

Optimizing graph construction can help GNNs learn effectively

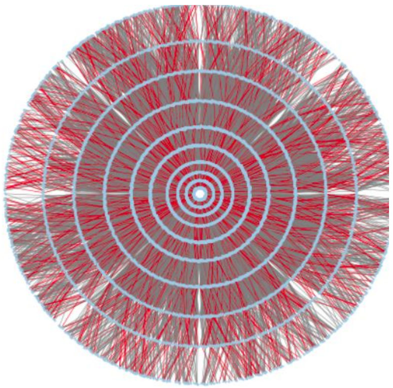
- Edge efficiency: true edges/all edges
- Truth efficiency: true edges in graph/all possible true edges

'Current' Methods

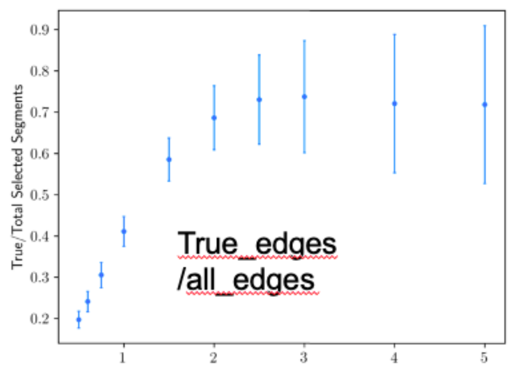
- Layer pairs: create edges between nodes in adjacent layers within a $\Delta\phi/\Delta r$ range
- Layer pairs+: allow edges within a layer
- kNN: form edges between a hit and its k closest neighbors (can customize distance metric)

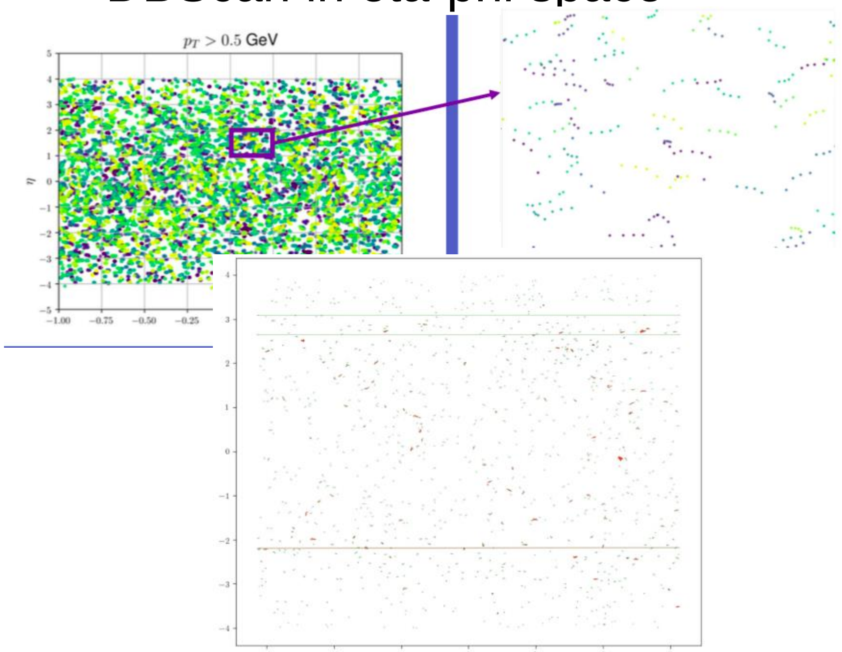
Exploratory Methods

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



Graph Efficiency





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

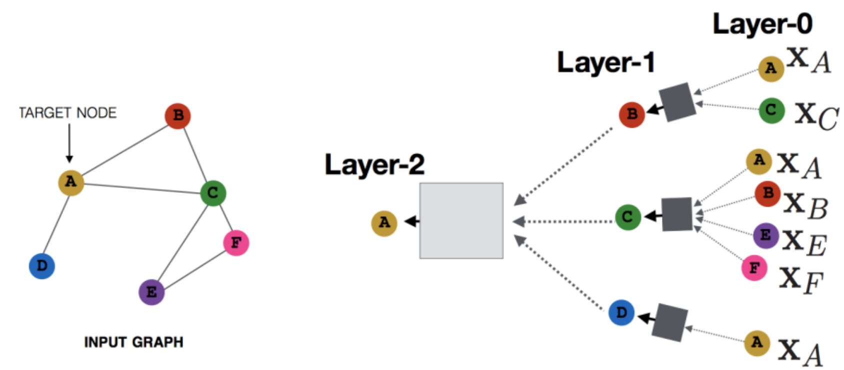
- Layer pairs: create edges between nodes in adjacent layers within a $\Delta\phi/\Delta r$ range
- Layer pairs+: allow edges within a layer
- kNN: form edges between a hit and its k closest neighbors (can customize distance metric)

Exploratory Methods

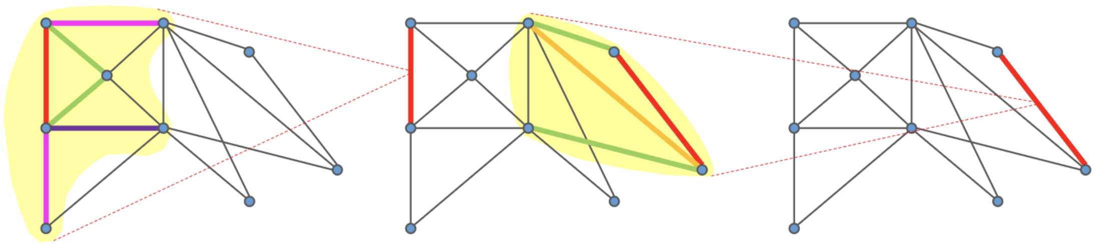
- Dynamic kNN
- Learned clustering
- DBScan in eta-phi space

	HEP.TrkX		**HEP.TrkX+, DBSCAN**		**DBSCAN**	
p_T^{min} [GeV]	ϕ_{slope}	z_0 [m]	ϕ_{slope}	z_0 [m]	ϵ	MinPts
2	6×10^{-4}	0.1	6×10^{-4}	15	0.22	3
1.5	6×10^{-4}	0.1	6×10^{-4}	15	0.18	3
1	6×10^{-4}	0.1	6×10^{-4}	15	0.1	3
0.75	7.63×10^{-4}	0.1	7.63×10^{-4}	25	0.08	3
0.6	7.63×10^{-4}	0.1	7.63×10^{-4}	29.5	0.06	3
0.5	7.63×10^{-4}	0.1	7.63×10^{-4}	29.5	0.05	3

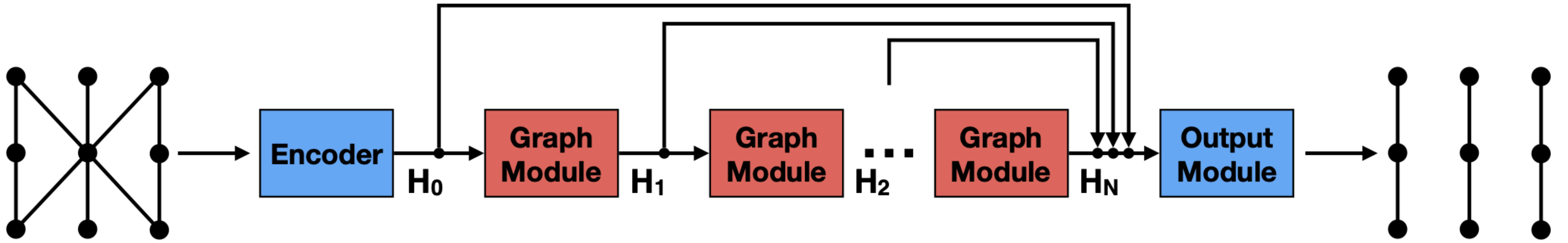
Edge Classifiers



- Graph Modules are core component:
  - Run node and edge convolutions
  - Update features of both
  - Each message passing function is a FCN



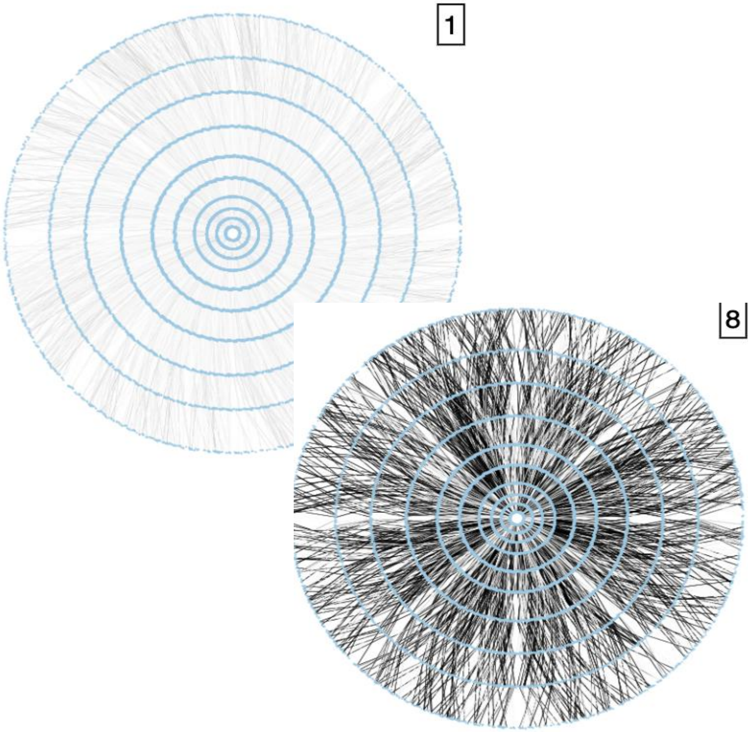
- Graph modules are often recursively connected
  - Allows aggregation of progressively more distant information
  - Weights can be shared across modules

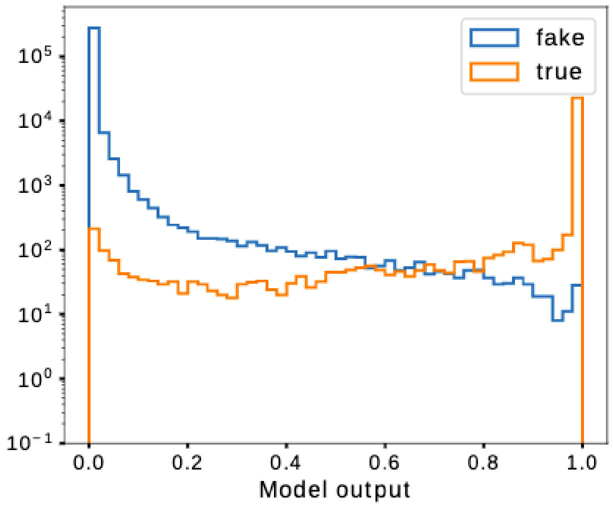


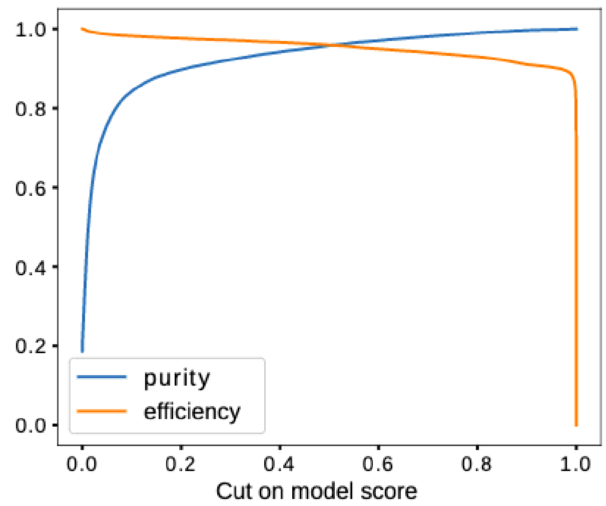
Proof of Principle

NeurIPS 2019 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:

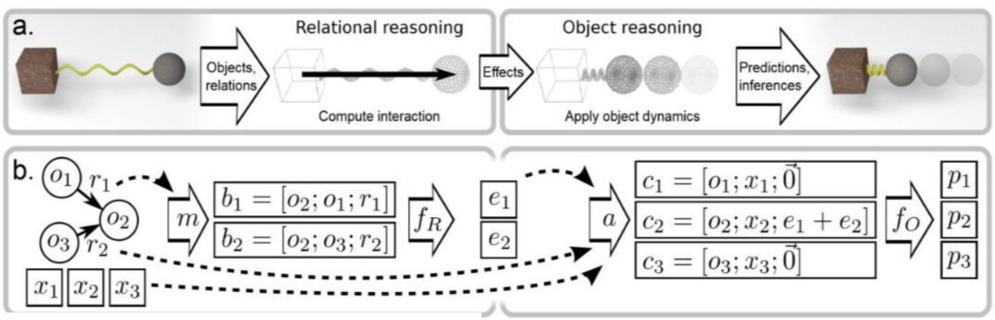
- 95.9% edge efficiency (*true edges/possible*)
- ~95% track finding accuracy (*all edges merged*)

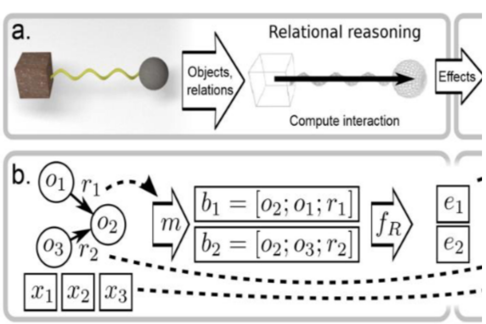
Paper

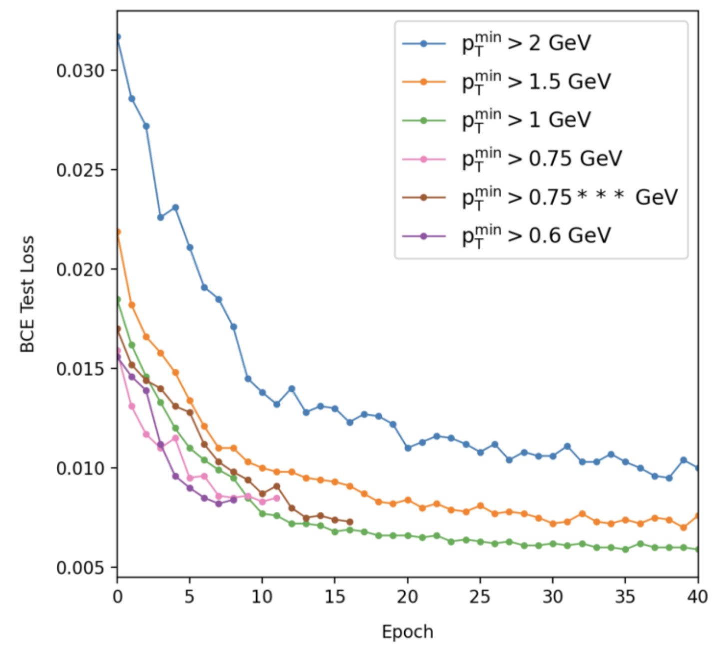
Interaction Networks

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

- Relation and object models are FCNs
- Total of ~10,000 parameters (smaller than previous architecture)







Results:

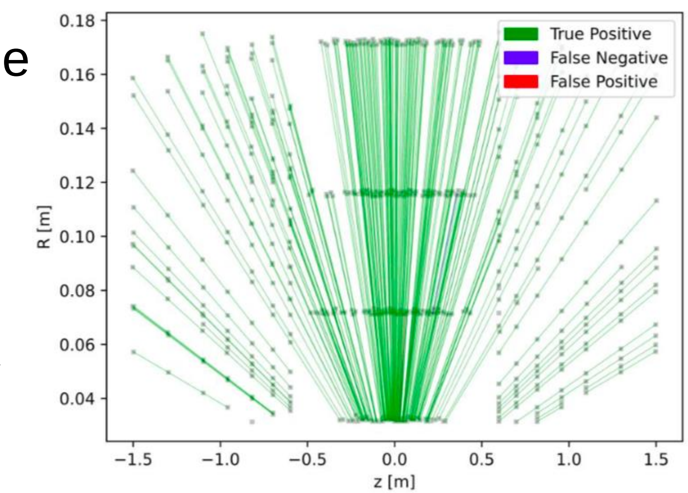
- 99.7% edge efficiency
- 85-94% tracking efficiency

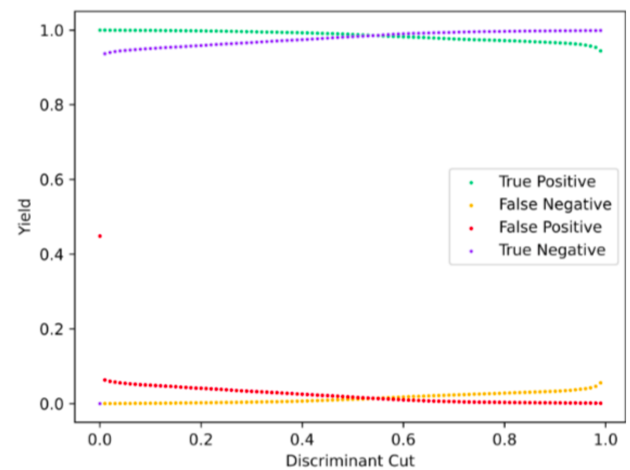
Paper, Recent Talk

Data Augmentation

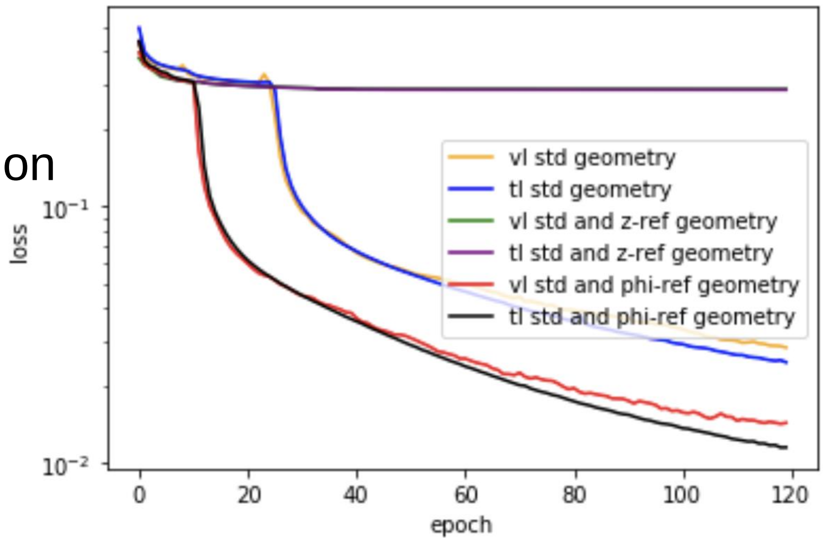
- Including endcaps:
  - Difficult in layer pairs construction due to edge ordering
  - Initial studies in pixel detector only, typically improve edge efficiency
- Dropping layers from graph construction
  - Reduce size of graph while maintaining track finding efficiency
- Applying z and phi reflections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

Pixel IN with Endcaps



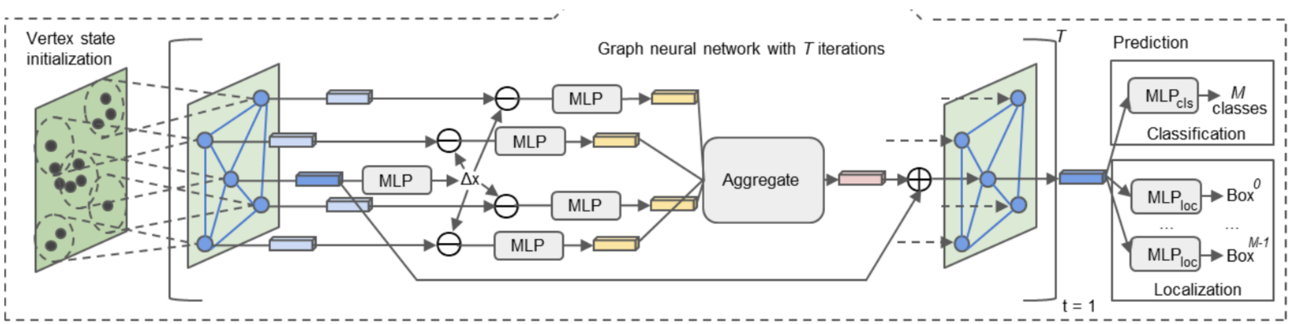


Data Augmentation with Edge Classifier

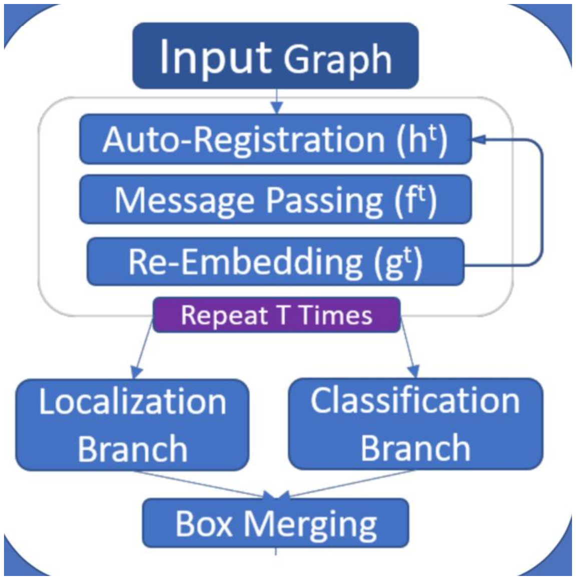


Instance Segmentation GNNs

- Instance segmentation: computer vision task of identifying instances of an object in an image and forming pixel mask
- After message passing, node state vectors are used as input to three branches:
  - Classification branch identifies the node as signal or background
  - Localization branch predicts a bounding box for each node
    - Ellipses merged and scored to create track clusters
  - Tracking branch predicts track parameters



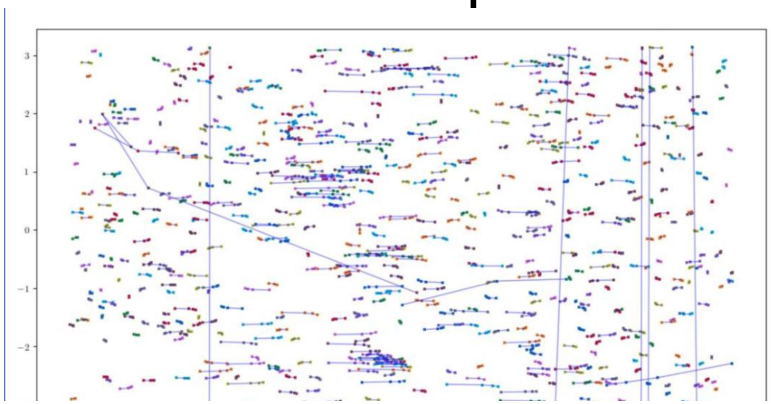
$$\Delta x_i{}^t = MLP_h^t(s_i^t)$$
$$e_{ij}^t = MLP_f^t([x_j - x_i + \Delta x_i{}^t, s_j^t])$$
$$s_i^{t+1} = MLP_g^t(Max(\{e_{ij} \,|(i,j) \in E\})) + s_i^t$$

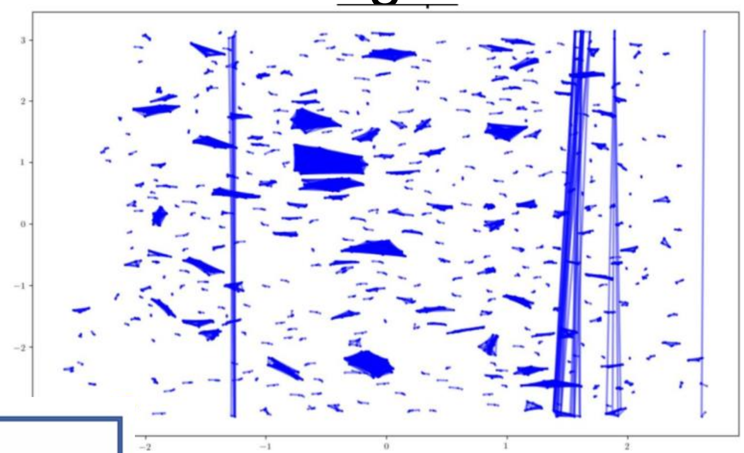


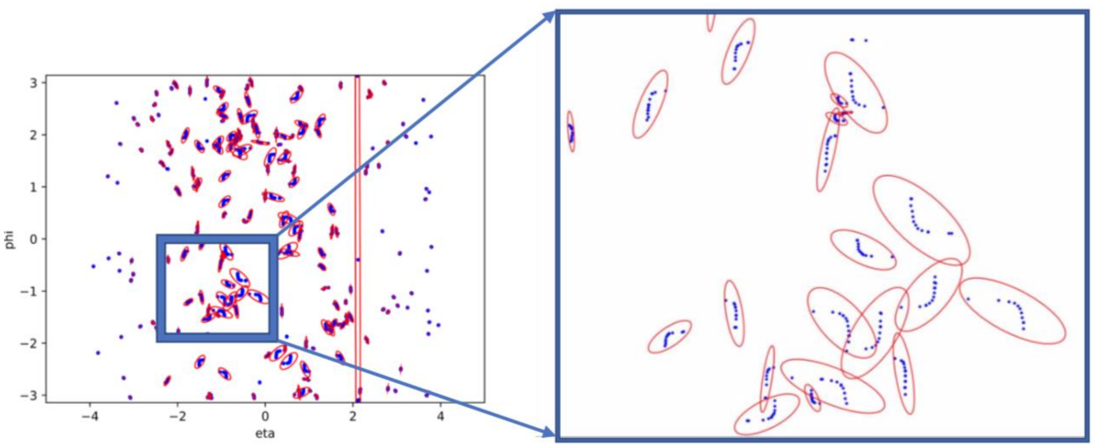
PointGNN Paper

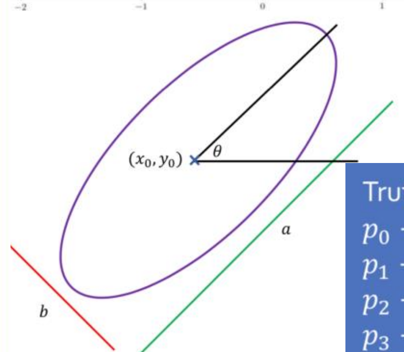
Elliptical Bounding Boxes

- Construct graphs using DBScan in eta-phi space
- Bounding ellipses parameterized with 5 degrees-of-freedom
- Encoded ellipses with each node for training









Truth Encoding (per hit):

$p_0 \rightarrow (u_0 - u_{hit})/0.04$

$p_1 \rightarrow (v_0 - v_{hit})/0.04$

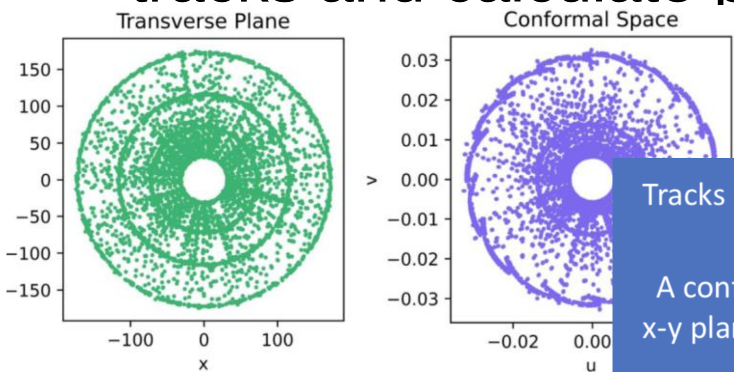
$p_2 \rightarrow \log(a)\,/0.25$

$p_3 \rightarrow \log(b)\,/0.75$

$p_4 \rightarrow (\theta - arctan2(v_{hit}, u_{hit}))/\pi$

Conformal GNNs

- Conformal transformation map tracks to straight lines
  - Can extract track parameters directly from linear fit
- Run instance segmentation GNN in conformal space to find tracks and calculate parameters in a single shot

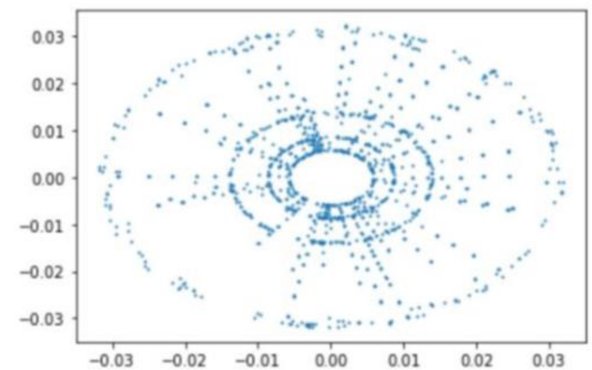


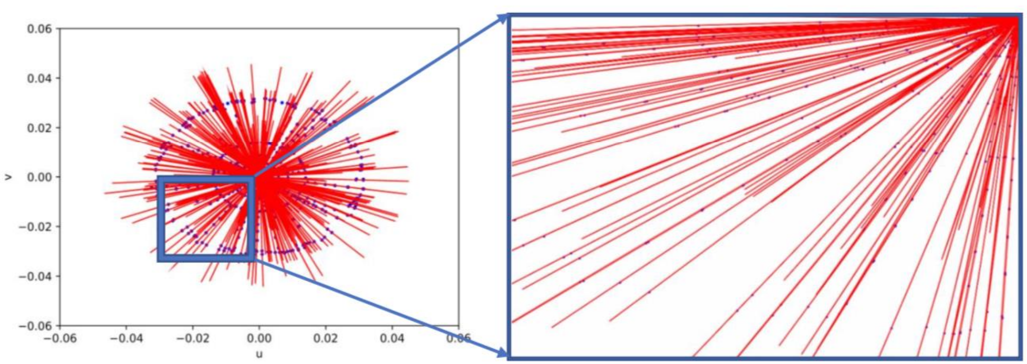
Tracks lie on circles in the transverse plane:

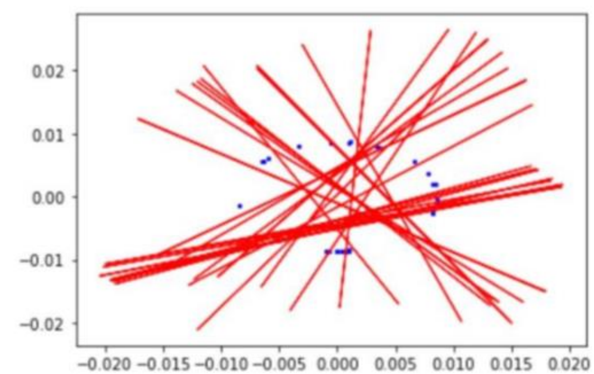
$$R^2 = (x-a)^2 + (y-b)^2$$

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

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



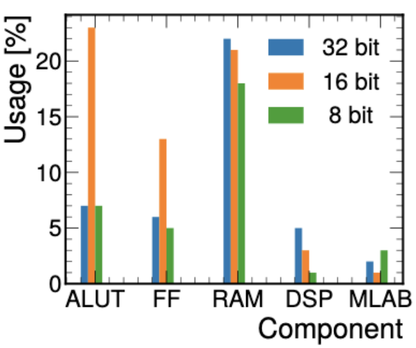


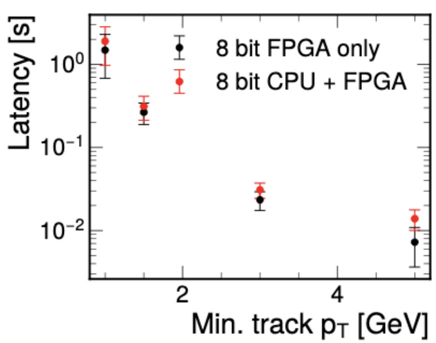


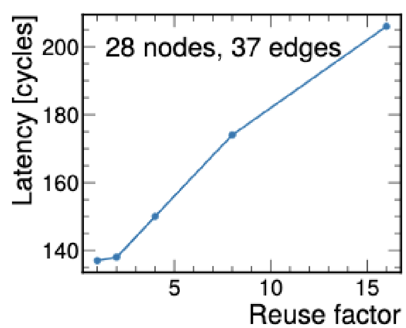
Accelerated GNN Tracking

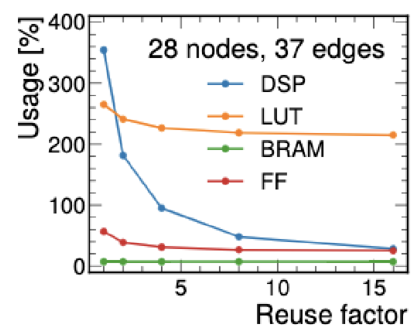
Strong interest in accelerating these algorithms with FPGAs

- HLS4ML implemented a 1 iteration version of IN for FPGA
- Princeton group optimizing OpenCL IN with FPGA as co-processor
  - Bottleneck in data transfer from CPU to FPGA
  - Opportunity for further acceleration in matrix multiplication kernels
  - Also exploring graph construction on FPGA









OpenCL

HLS4ML

Recent Paper

On-going Tracking Studies

- Optimize parameters of existing graph construction algorithms and explore new ones
- Refine track formation algorithm for edge classification architectures
- Improve existing architectures
  - Include external effects in IN, optimize embedding…
- New ideas
  - Timing information, Hough transforms, graph kernels…
- Test performance in LHC experiment environments

Conclusions

- GNNs are a promising method for HL-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
  - Also exploring one-shot tracking architectures
- Graph construction (and embedding) is critical to performance
  - On-going optimization studies (submitted to vCHEP)
- Working towards accelerating graph algorithms for use at HL-LHC
  - Possibly at trigger level

Thank you!

Happy to answer any questions!

sthais@princeton.edu @basicsciencesav