GNNS FOR RECONSTRUCTION AT THE LHC

Savannah Thais

CLARIPHY Topical Meeting

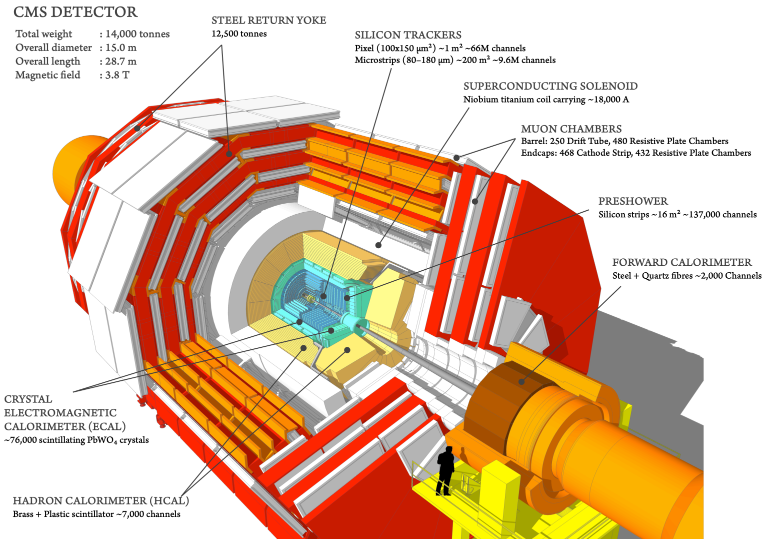
12/04/2020

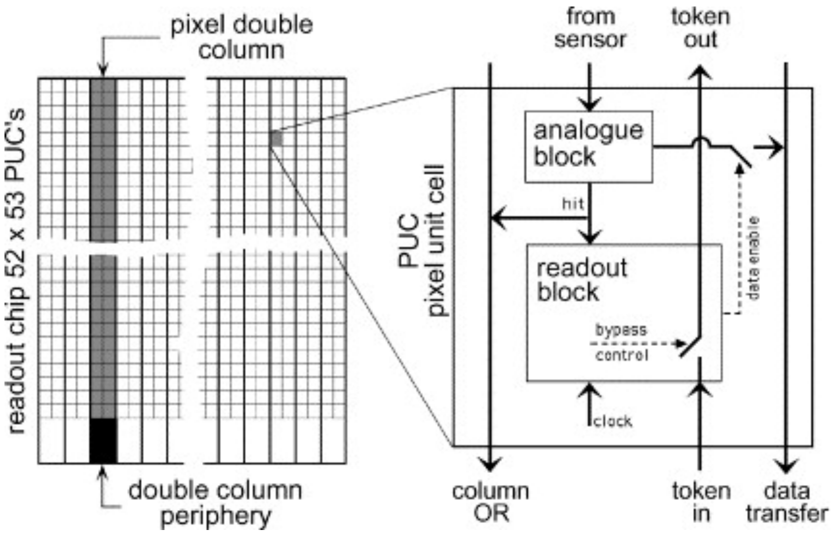
iris hep

Princeton
Research Computing

LHC Data

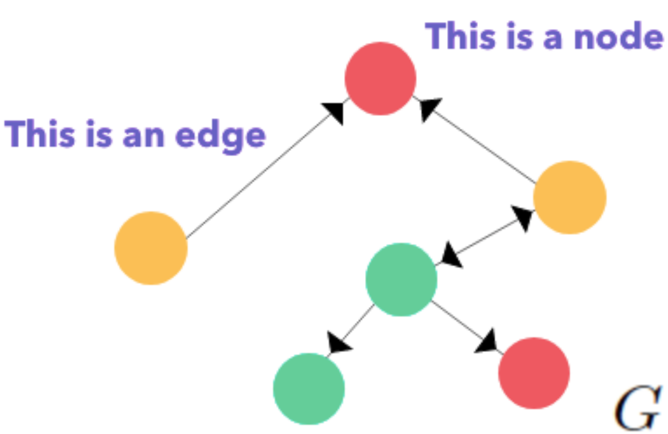
- Collision data measured by dedicated subsystems
  - Quantifies interactions with highly granular detectors
  - Readouts must be reconstructed into particle components (tracks, clusters) then full particle candidates and event information
- Challenge: data is often sparse and not fixed size
  - Traditionally stored in tree structures
  - For ML applications often 'forced' into matrix representations



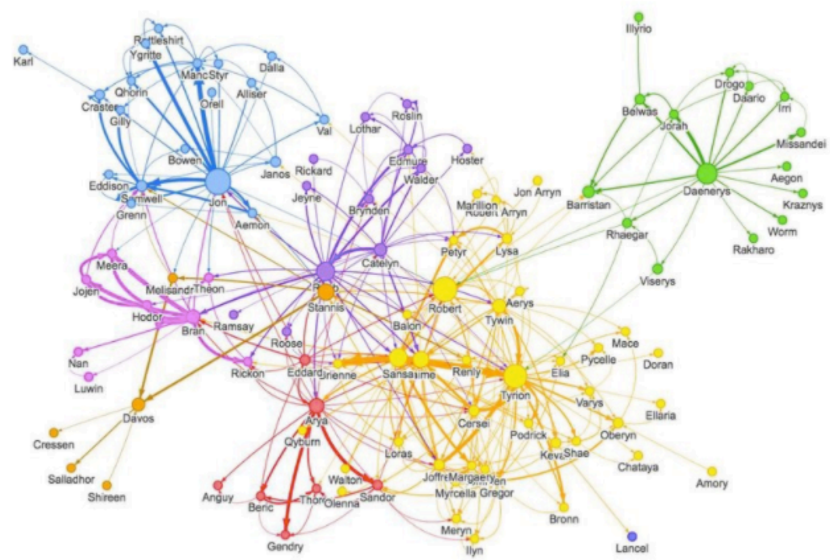


Graphs

- 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)

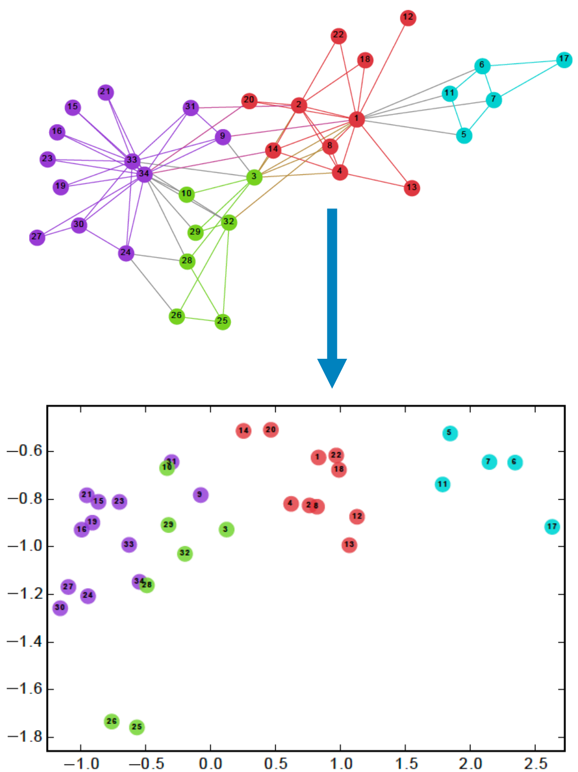


$$G = (V, E)$$



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

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

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

previous layer embedding of v

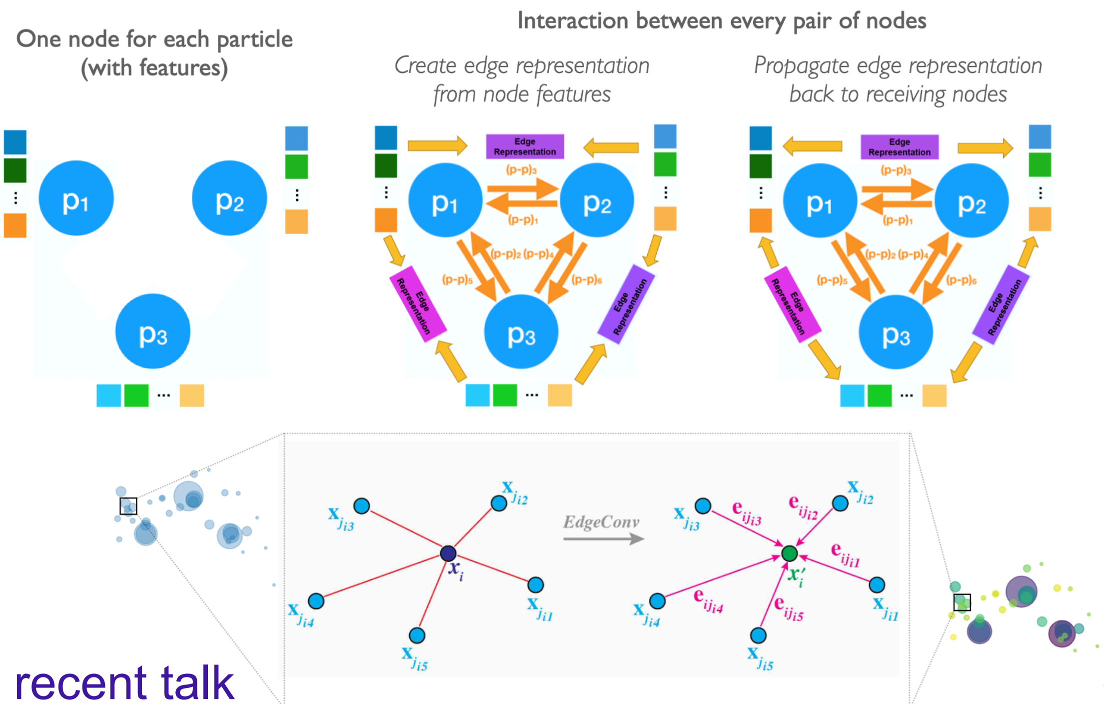
kth layer embedding of v

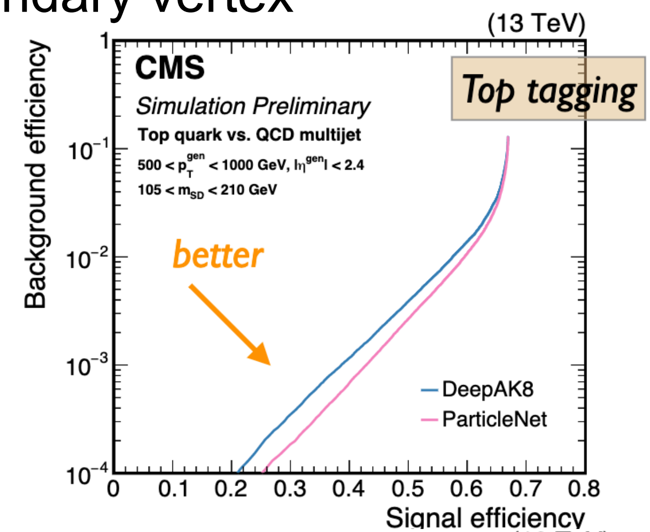
non-linearity (e.g., ReLU or tanh)

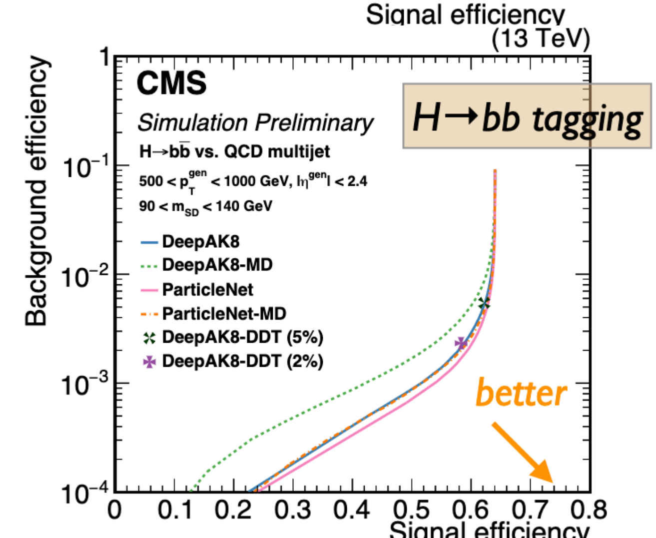
average of neighbor's previous layer embeddings

GNNs for Jet Tagging

- ParticleNet graph convolution architecture
  - Each node is a reconstructed particle or secondary vertex
  - Edges formed by kNN
- Multiclass tagger for t/W/Z/H outperforms current CMS taggers



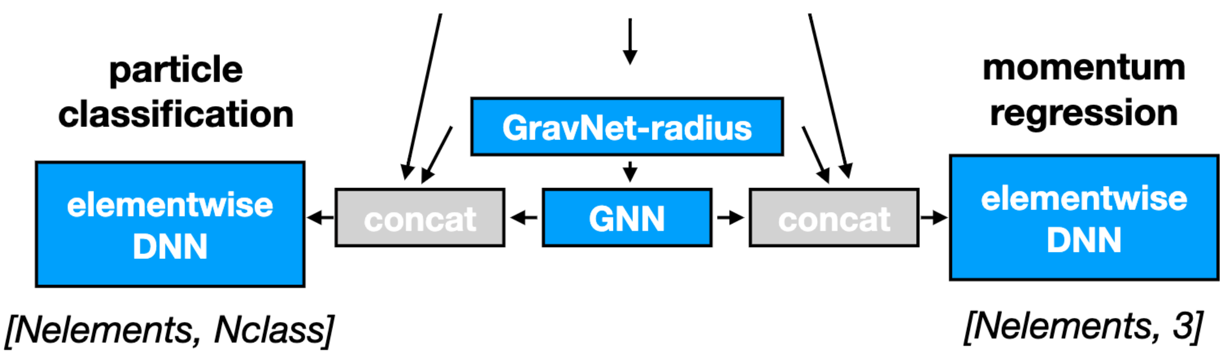


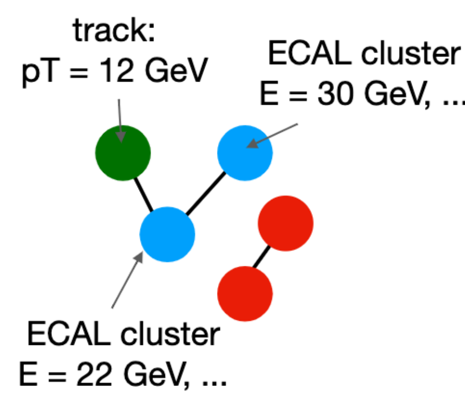


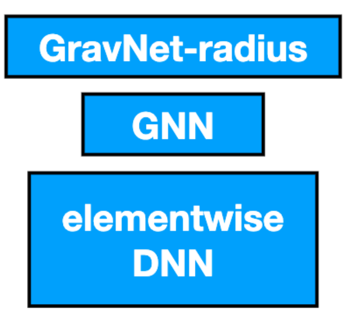
recent talk

GNNs for Particle Reconstruction

- GravNet architecture to form ParticleFlow candidates
  - Each node is a PF element (track, calo clusters, etc)
  - Edges formed by dynamic kNN or radius graph
- Encode with DNN then process with GCN or Highway Networks





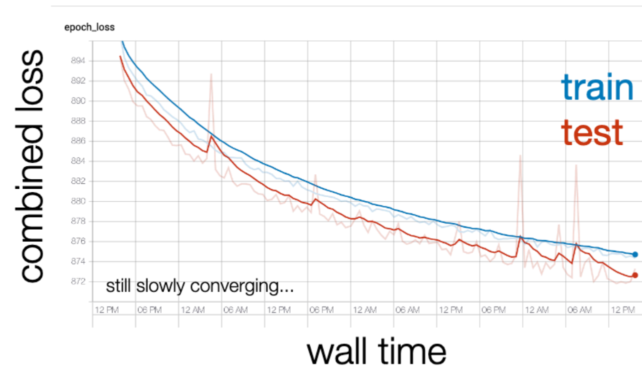


find *N* neighbors in radius *r*

aggregate graph info: SGConv or GATConv

Separate decoding DNNs (2-3 layers, 512 units)

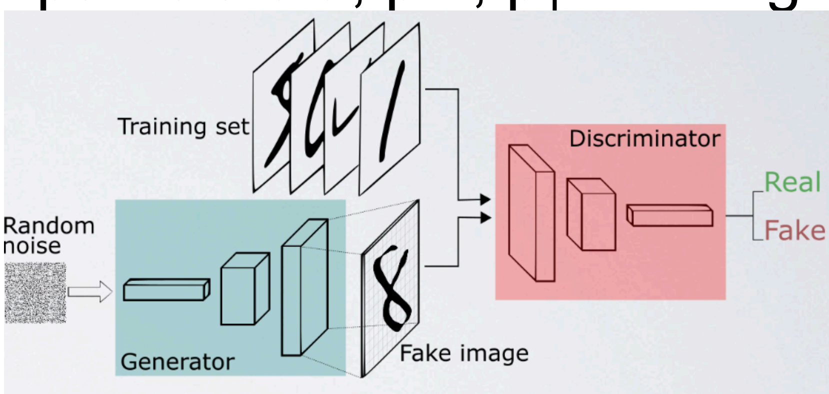
>72h of training on 2x Titan Xp

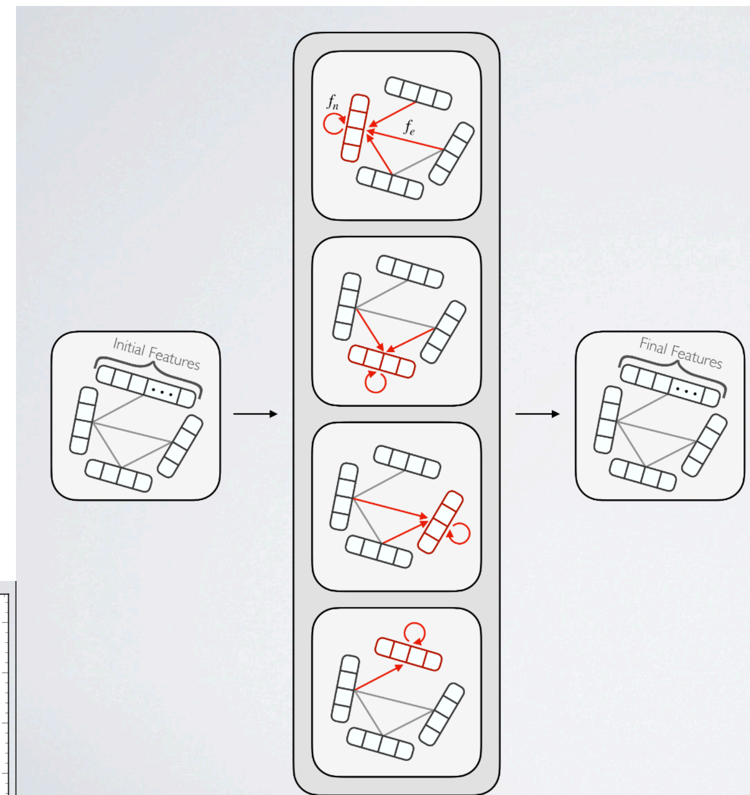


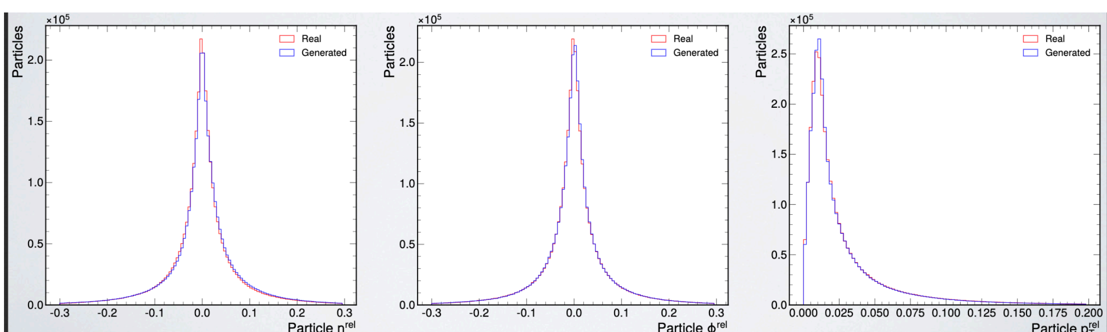
recent talk

GNN GANs

- Graph based Generative Adversarial Network
  - Generator uses message passing on noise graph to recreate input features
  - Discriminator uses message passing to classify graph
- Using jet dataset, can recreate particle eta, phi, p_T with high fidelity





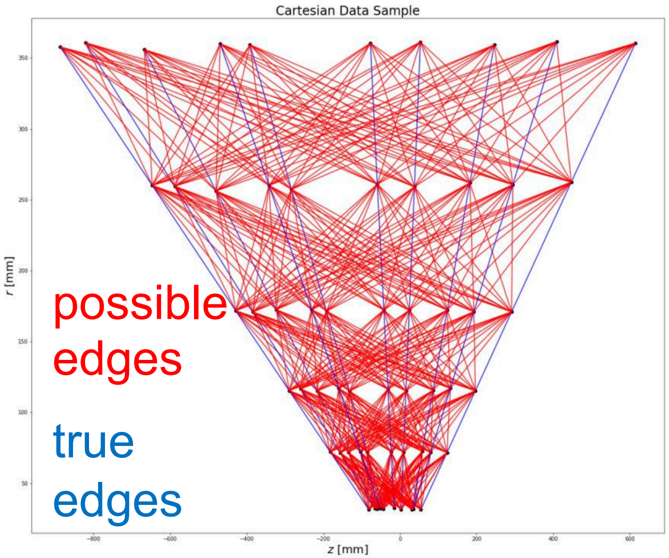


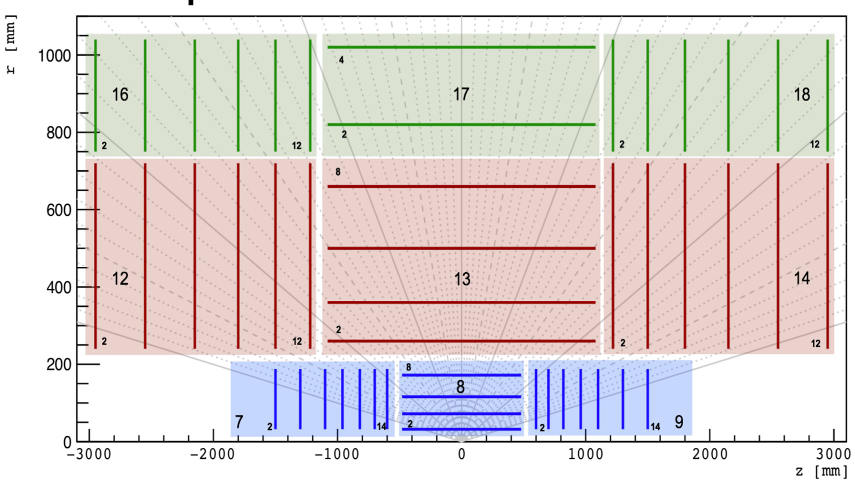
recent talk

GNNs for Tracking

Basic procedure

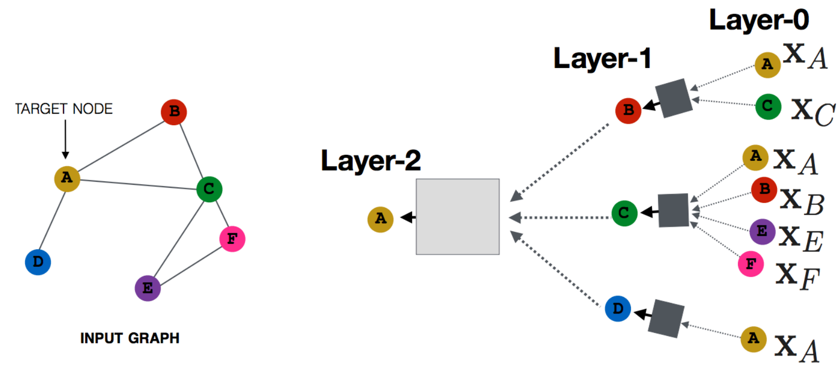
1. Form initial graph from spacepoints/hits (pre-processing)
2. Process with GNN to get probabilities of all edges
3. Apply post-processing algorithm to link edges together into tracks and get parameters



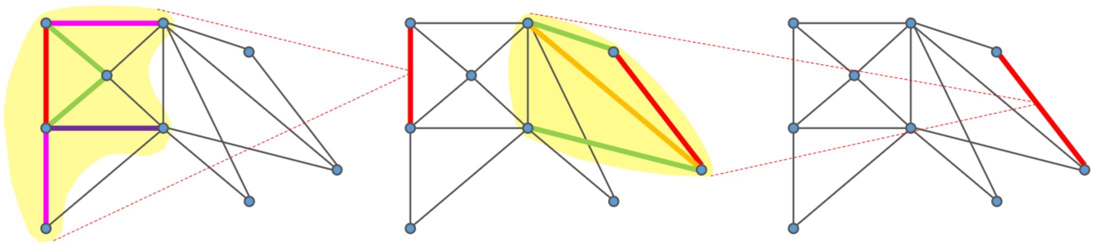


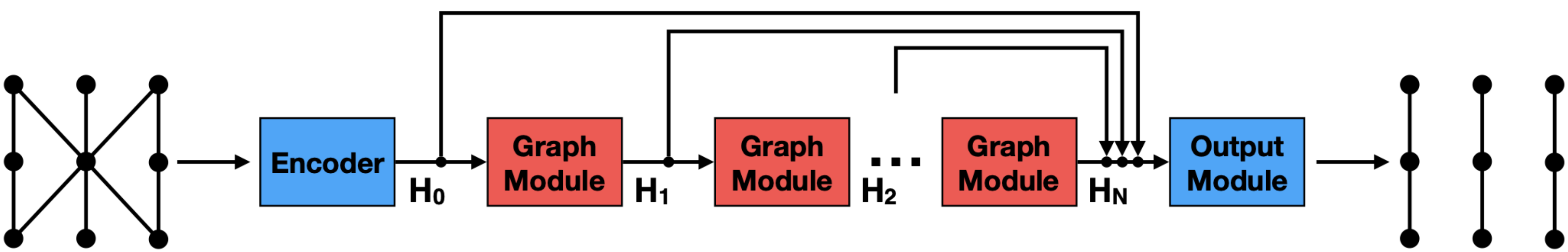
- Many places to improve/innovate
  - Graph construction, architectures, data augmentation…
- Most work shown here uses [TrackML dataset]
  - Open, experiment agnostic
  - 200 PU, silicon semiconductor detector

Edge 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

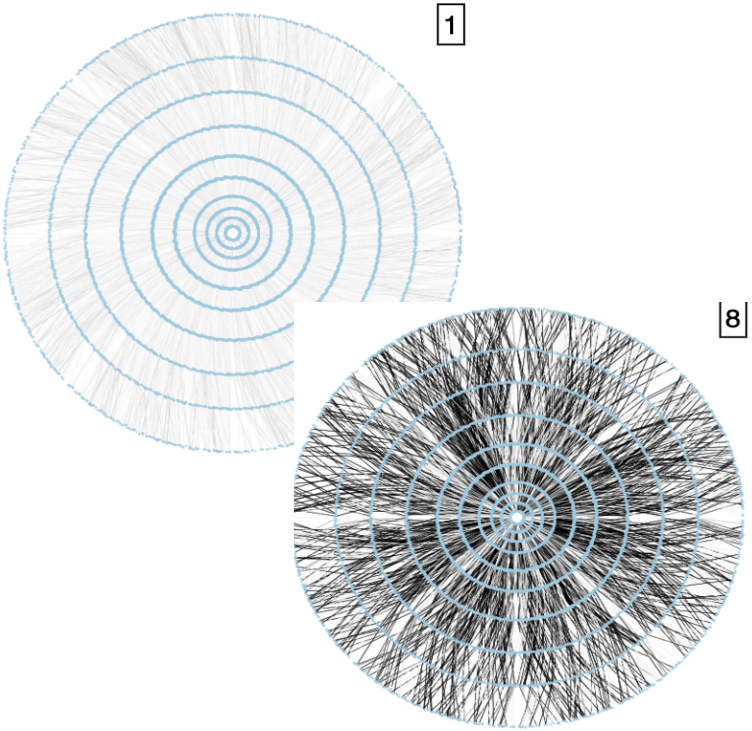


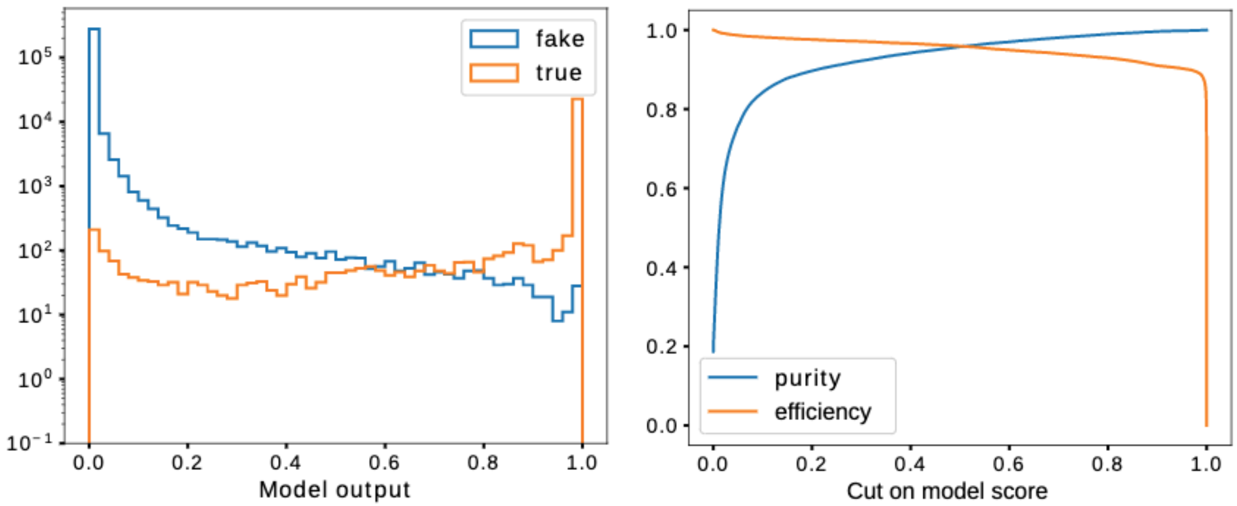


Proof of Principle

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:

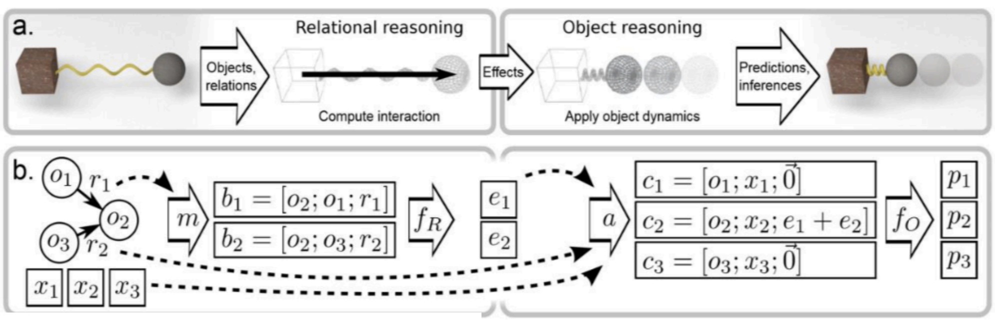
- 95.9% edge efficiency (*true edges/possible*)
- ~95% track finding accuracy (*all edges merged*)

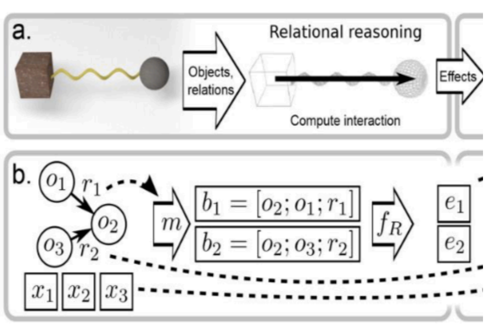
Paper

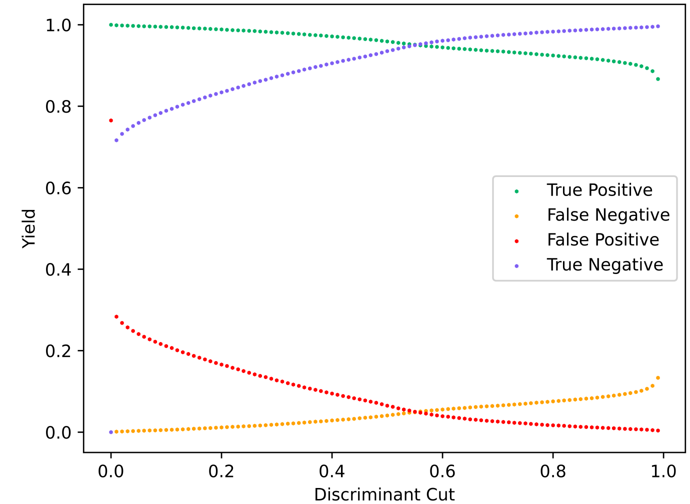
Interaction Networks

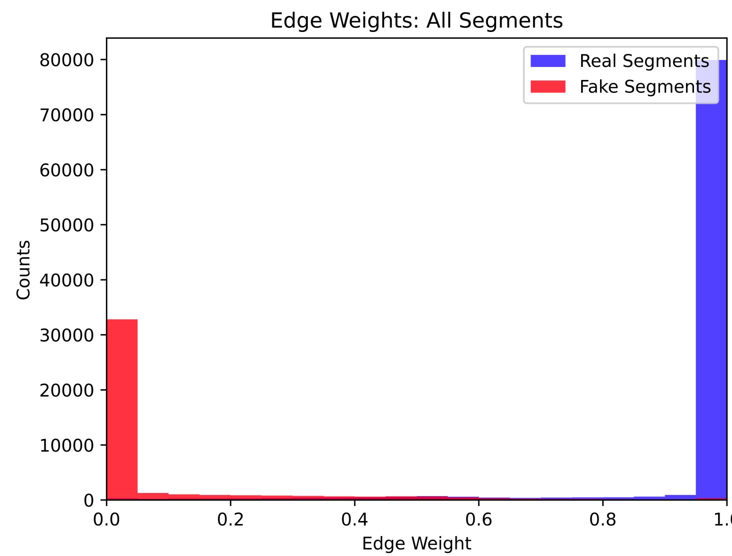
Applies relational and object models in stages to infer abstract interactions and object dynamics

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









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

Results:

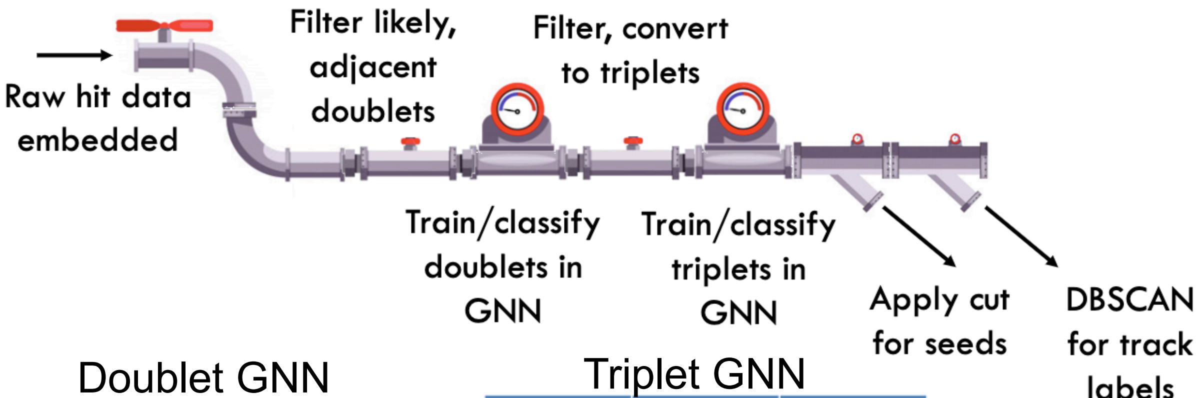
- 95% edge efficiency
- Tracking efficiency still being measured

[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

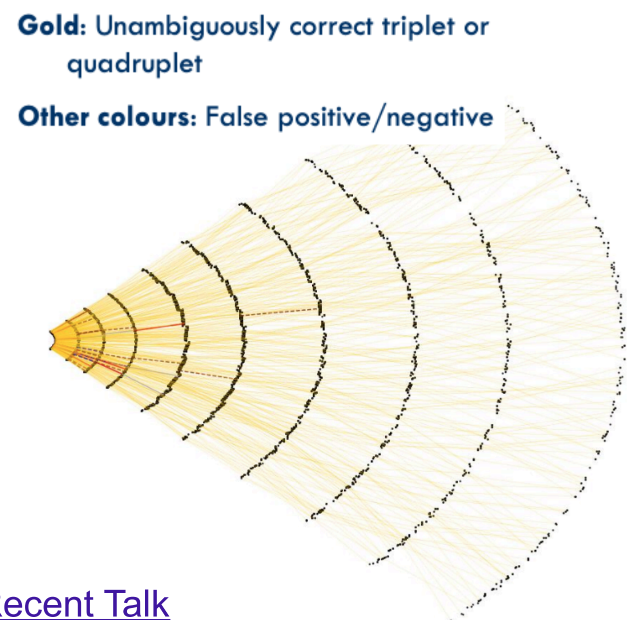


Doublet GNN

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

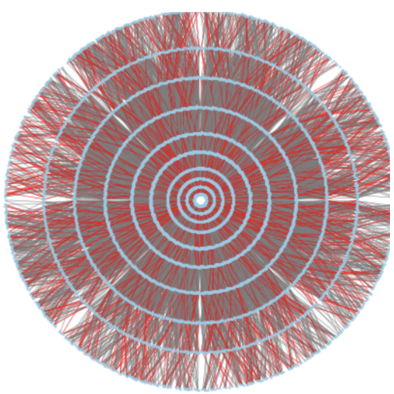
- Edge efficiency: true edges/all edges
- Truth efficiency: true edges in graph/all possible true edges

'Current' Methods

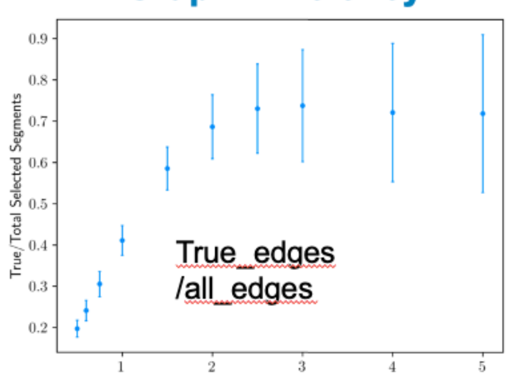
- Layer pairs: create edges between nodes in adjacent layers within a $\Delta\phi/\Delta r$ range
- Layer pairs+: allow edges within a layer
- kNN: form edges between a hit and its k closest neighbors (can customize distance metric)

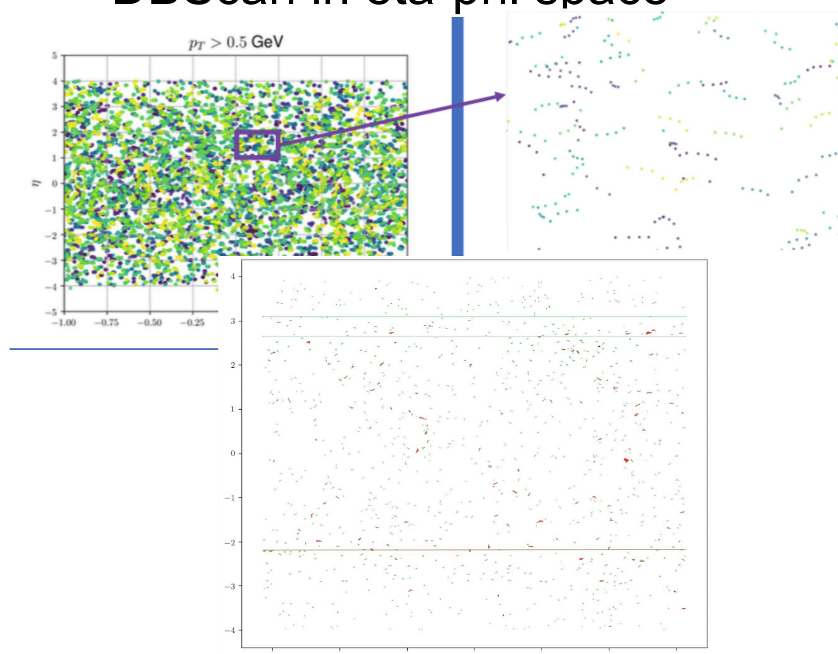
Exploratory Methods

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



Graph Efficiency

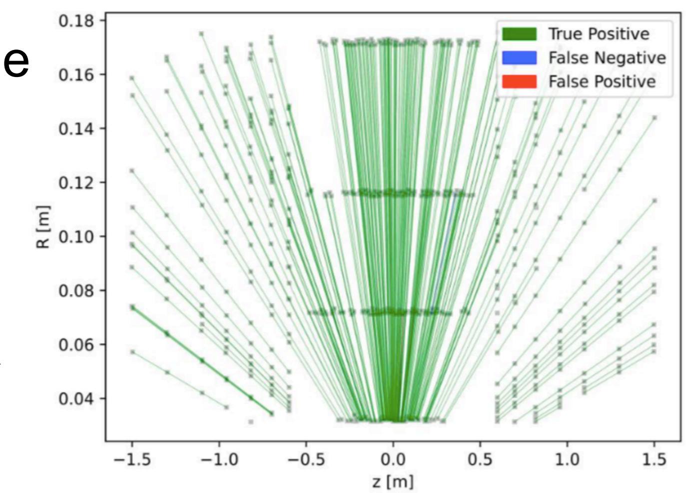


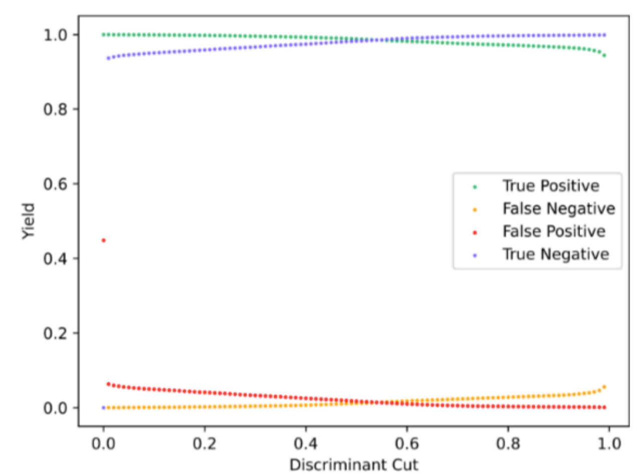


Data Augmentation

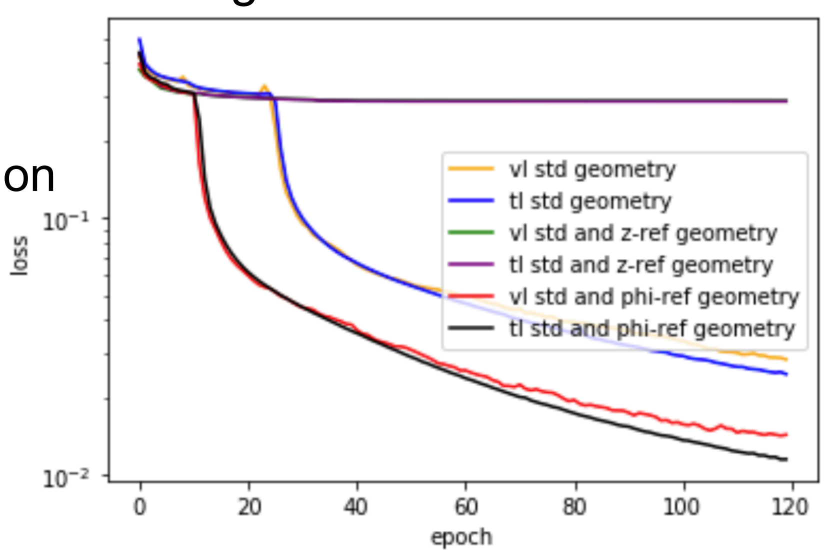
- Including endcaps:
  - Difficult in layer pairs construction due to edge ordering
  - Initial studies in pixel detector only, typically improve edge efficiency
- Dropping layers from graph construction
  - Reduce size of graph while maintaining track finding efficiency
- Applying z and phi reflections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

Pixel IN with Endcaps



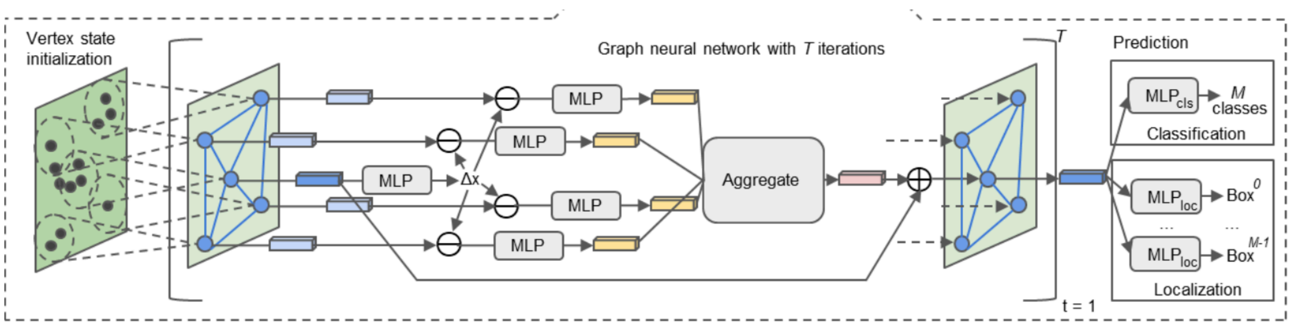


Data Augmentation with Edge Classifier

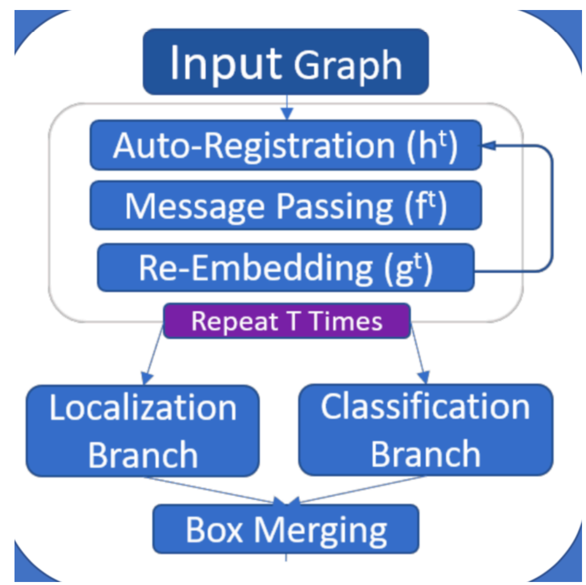


Instance Segmentation GNNs

- 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



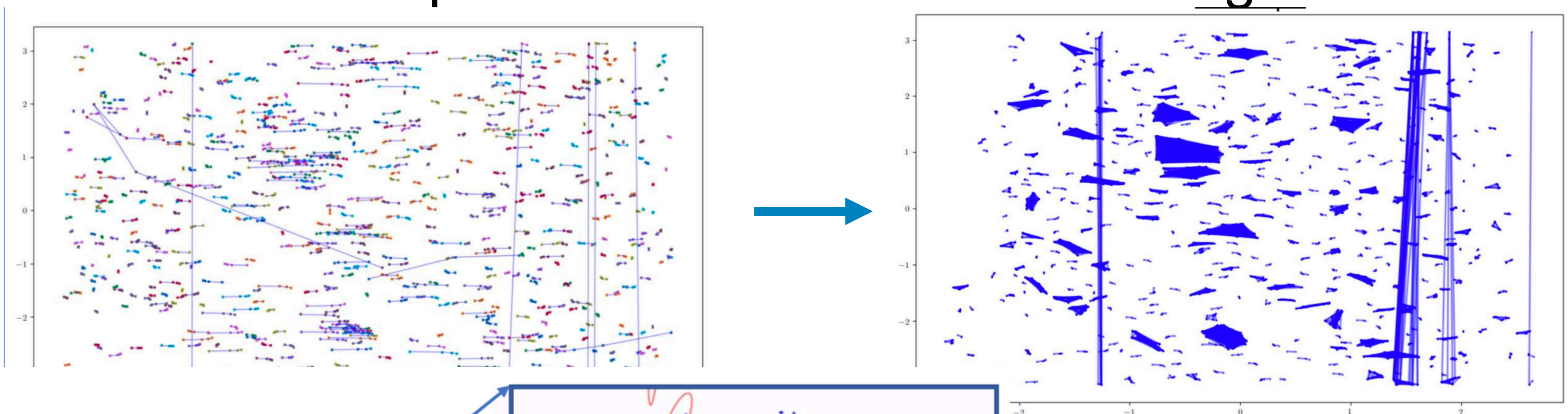
$$\Delta x_i{}^t = MLP_h^t(s_i^t)$$
$$e_{ij}^t = MLP_f^t([x_j - x_i + \Delta x_i{}^t, s_j^t])$$
$$s_i^{t+1} = MLP_g^t(Max(\{e_{ij} \mid (i,j) \in E\})) + s_i^t$$

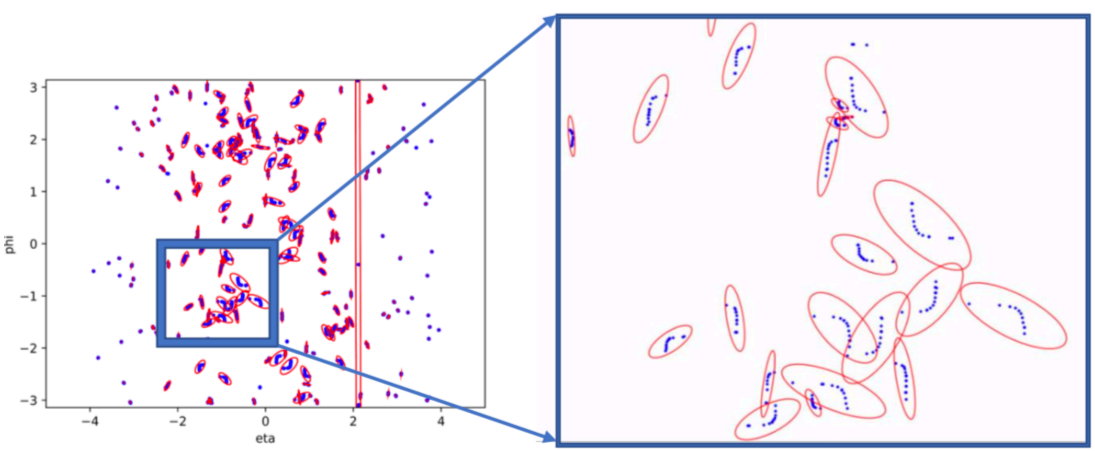


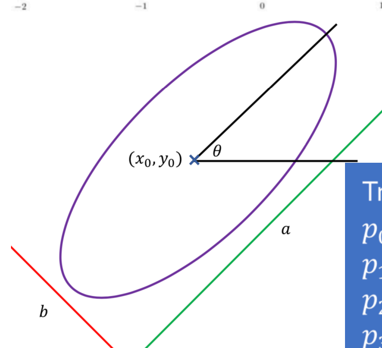
PointGNN Paper

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):

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

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

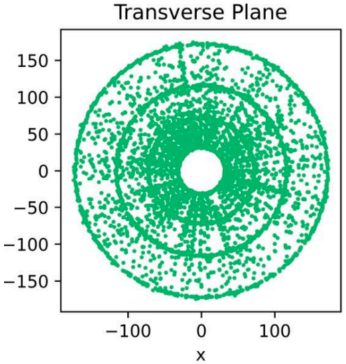
$p_2 \to \log(a)\,/0.25$

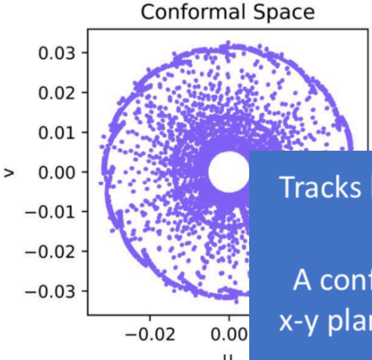
$p_3 \to \log(b)\,/0.75$

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

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



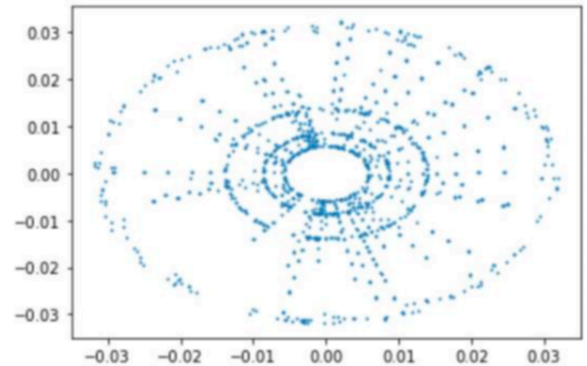


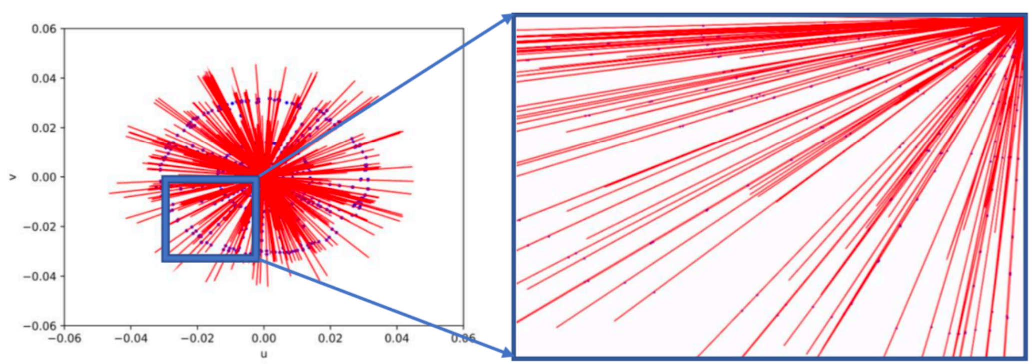
Tracks lie on circles in the transverse plane:

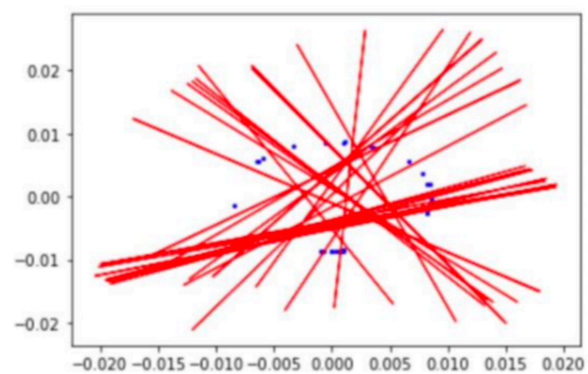
$$R^2 = (x-a)^2 + (y-b)^2$$

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

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







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
- Exploring hardware acceleration with FPGAs

Conclusions

- Graphs are a natural representation of particle detector data
- Graph-based learning methods can leverage geometric information for effective reconstruction
  - Graph classification for jet tagging
  - High-level node features for particle object reconstruction
  - Graphs as nodes for simulation
  - Optimal transport on graphs for event-level analysis (1919)
  - Edge classification and instance segmentation tracking
- Geometric deep learning is synergistic with particle physics
  - Rich and exciting area of research!

Thank you!

Happy to answer any questions!

sthais@princeton.edu @basicsciencesav