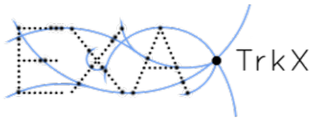
Hierarchical Graph Neural Network

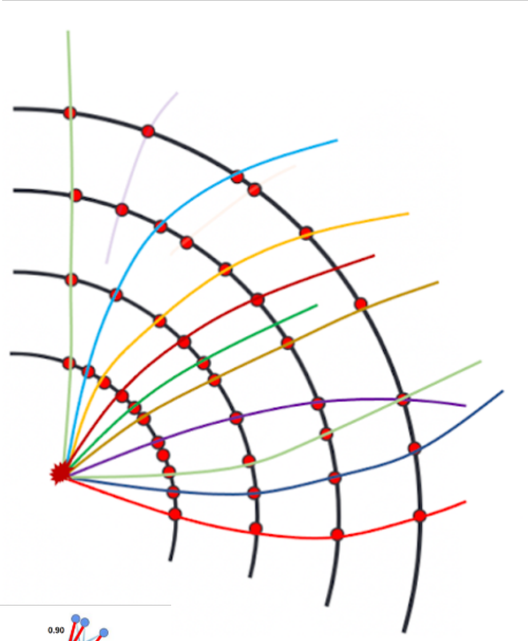
Ryan Liu (UC Berkeley)
Paolo Calafiura, Steven Farrell, Xiangyang Ju, Daniel Murnane (LBNL)

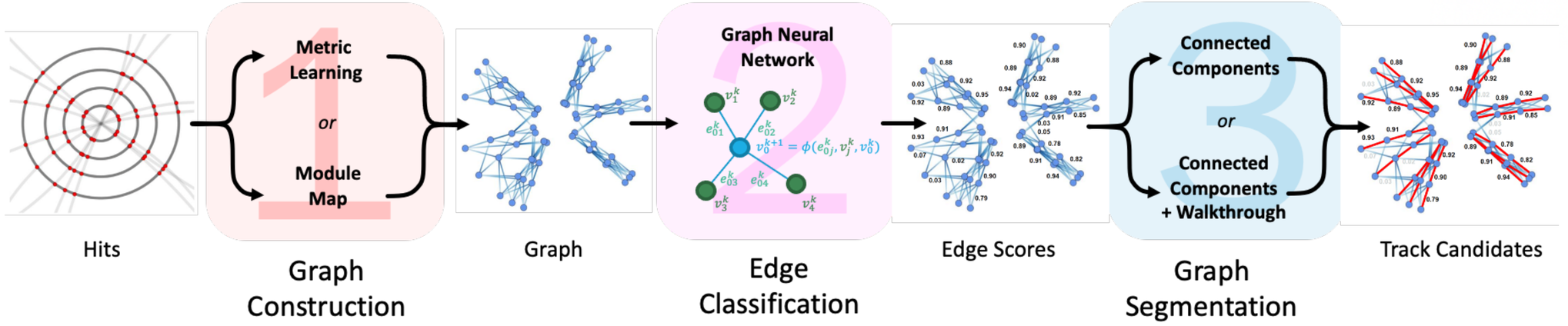


BERKELEY LAB

Track Reconstruction & Exa.TrkX

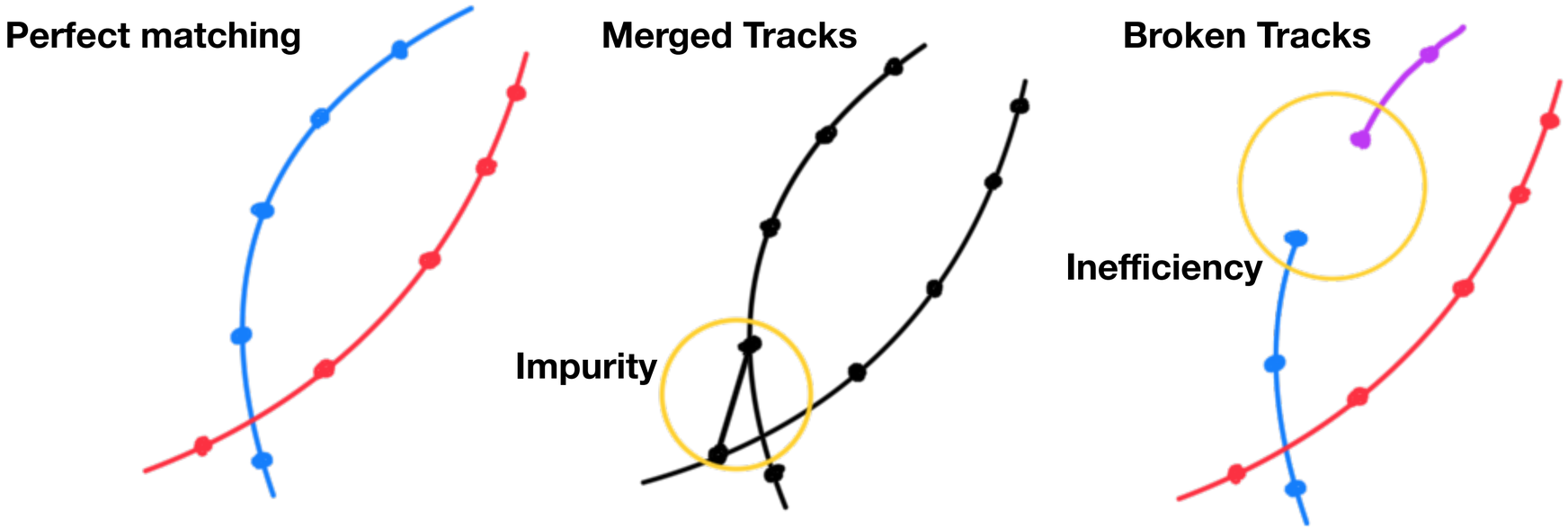
- Proton-proton collisions in the LHC detectors creates numerous particles which travel in helical tracks and produce hits along them.
- The Exa.TrkX pipeline utilizes deep graph learning methods to "connect the dots" into tracks which are crucial for further studies.





Broken and Merged Tracks

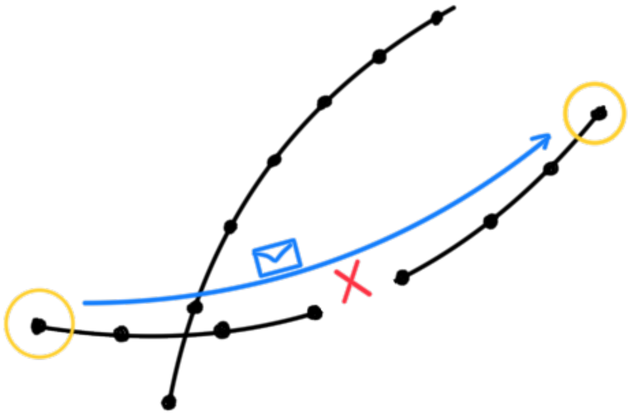
- After the GNN edge classifier, we apply a score cut and perform connected components search/walk through algorithm to get track candidates.
- However, the **inefficiency** and **impurity** in graph construction will give rise to **broken tracks** and **merged tracks**.



Hierarchical GNN

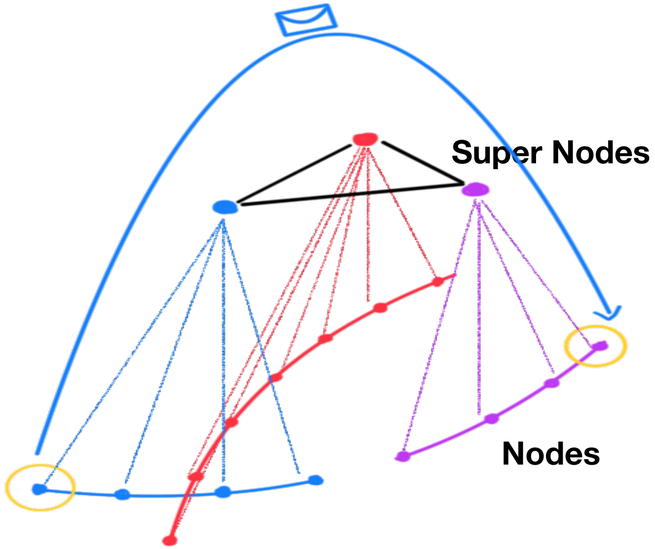
- Could we remedy these defects with GNNs?
- Inefficiency forbids message passing ⇒ hierarchical structure
- Nodes are combined to form super nodes in hierarchical graphs through a pooling algorithm that will be defined later.

Vanilla GNN:
Inefficient graph construction forbids message passing



Pooling

Hierarchical GNN:
Long-distance message passing is possible

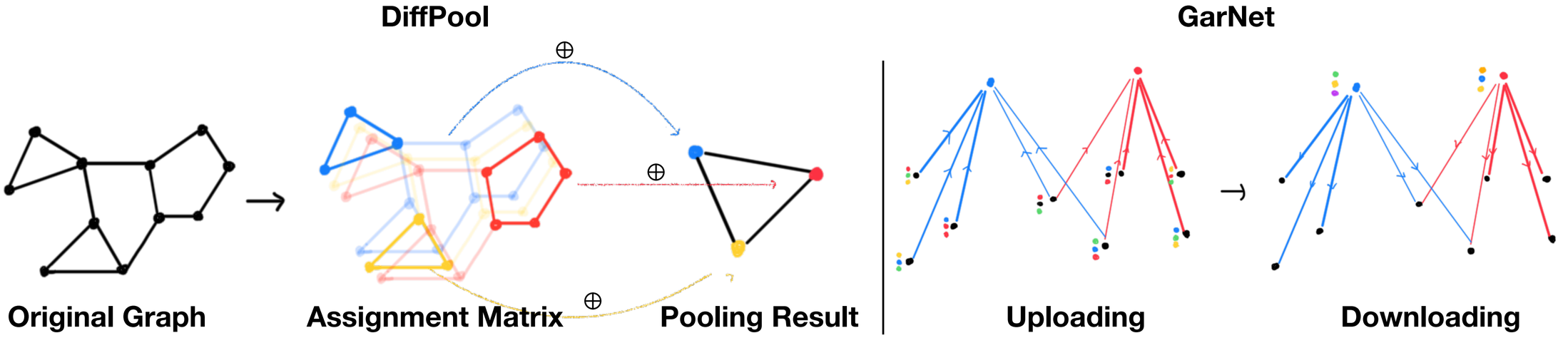


What Do We Need?

- Sparsity: sparsity can guarantee the scalability of the algorithm
  - Defined as the average number of edges that connect to a node (i.e. average connectivity).
  - The complexity of GNN is $O(N)$ if the connectivity is constant.
  - If the pooling is not sparse, then for a fixed pooling ratio k, the complexity is $O(kN^2) = O(N^2)$
- Differentiability: differentiability is crucial for learning purpose
  - Differentiability can enable the model to learn how to cluster instead of using heuristics
- Variable Number of Clusters: events contain indefinite number of tracks
  - Even though we don't know what's the "best" way of pooling the graph, a naive guess is that it pools hits belonging to the same track together. Thus the number of clusters should be variable

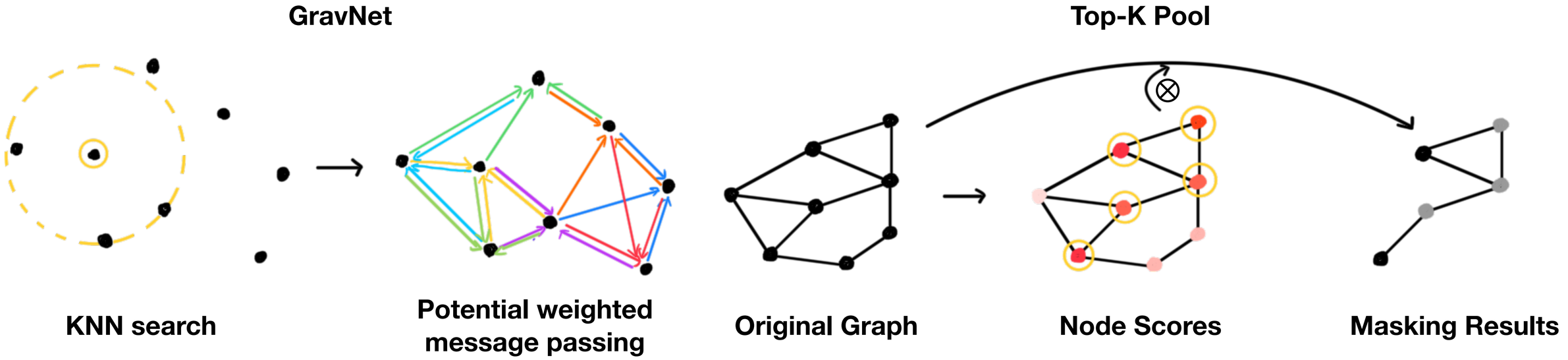
DiffPoll & GarNet

- DiffPool proposes to use GNN generate an **assignment matrix** $S \in \mathbb{R}^{N \times K}$, then aggregate node features $N \in \mathbb{R}^{N \times D}$ to super node features $X = S^T N \in \mathbb{R}^{K \times D}$, and finally create super graph $A' = S^T AS$.
- GarNet replaces the probabilistic assignments with **potential function**. Message passing is done by "uploading" node features and "downloading" aggregator features.
- Inspiration: hierarchical graphs could communicate by $X_{update} = S^T N$ and $N_{update} = SX$



GravNet & Top-K

- GravNet builds a dynamic graph by using **potential weighted edge** and **k-nearest neighbor graph** to guarantee its sparsity. This graph is updated for each iteration.
- Top-K pooling calculates **node scores** and selects **top-k nodes**. This method is made differentiable by putting a **score weighted gate** on pooled features.
- Inspiration: sparsity could be maintained by using KNN search. Differentiability of super edges could be maintained by using differentiable gate.

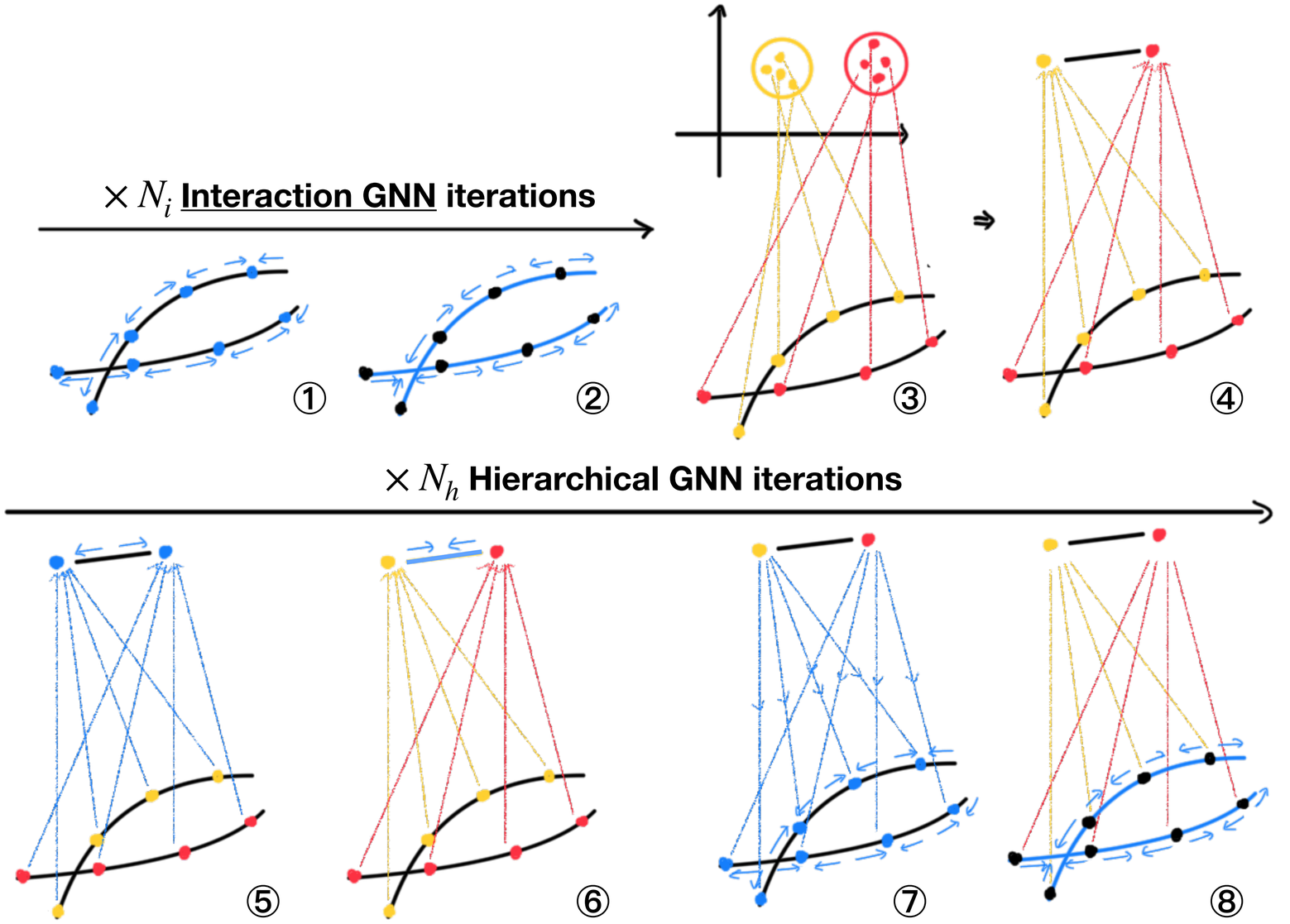


Model Architecture

① Node update
node, adjacent edges
② Edge update
edge, adjacent nodes
③ Embedding & spatial clustering
nodes
④ Aggregating super nodes
nodes, assignment matrix
⑤ Super node update
Super node, adjacent super edges, assigned nodes
⑥ Super edge update
Super edges, adjacent super nodes
⑦ Node update
Node, assigned super nodes, adjacent edges
⑧ Edge update
Edge, adjacent nodes

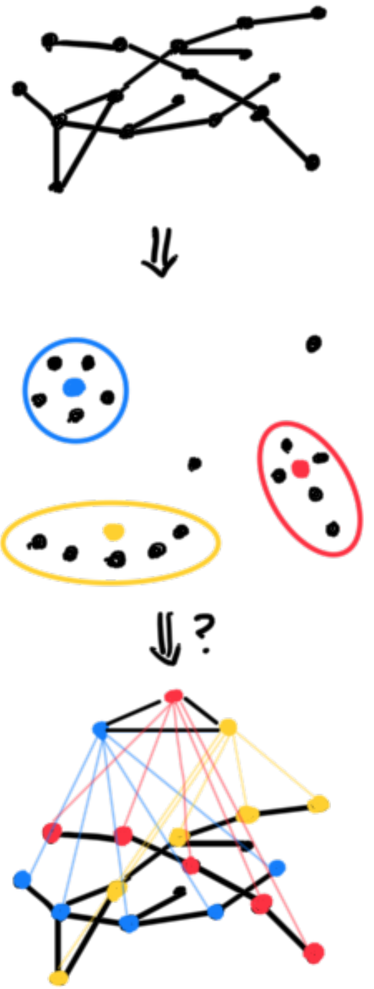
† each update is given by a MLP model

† blue: update in progress



Soften Spatial Clustering

- The requirement of variable number of cluster naturally lead us to **density-based spatial clustering algorithms** such as DBSCAN and HDBSCAN. However, these algorithms produce **hard assignments** and are **non-differentiable**.
- Thus we propose to use the idea of **centroids**:
  1. Given a set of node embeddings $\{n_i\}_{i \in G}$, we apply spatial clustering algorithms to get cluster IDs $\{x_i\}_{i \in G_s}$. Where G_s stands for signal points (not noise) in G.
  2. Compute the **centroids** by $X_i = \frac{1}{N(C_i)} \sum_{x_k = i} n_k$, these are the **super node embeddings**, thus the number of clusters determines number of super nodes.
  3. Construct the **super graph** and **assignments** by using distance weighted algorithms similar to GravNet.



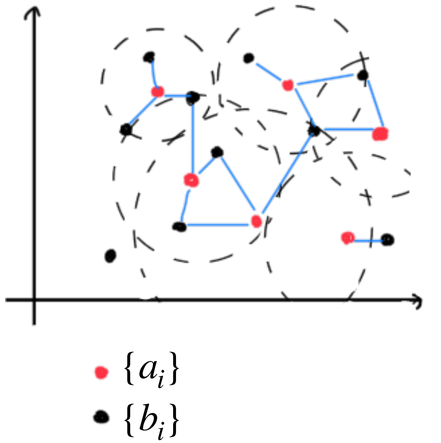
Differentiable Sparse Graph Construction

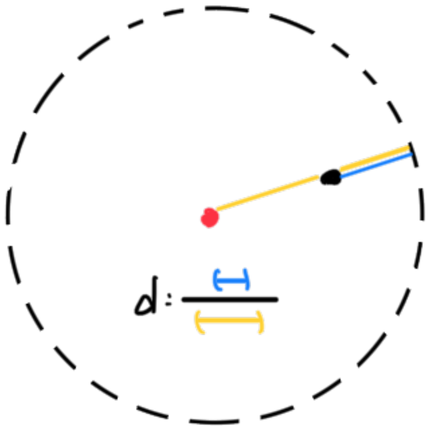
Given two sets of nodes A and B and their embeddings $\{a_i\}_{i\in A}$ $\{b_j\}_{j\in B}$, the bipartite graph (i.e., edges only connect between two parties but not within each of them) of sparsity k (when A and B is identical, the graph) is constructed as follows:

1. Perform **k-nearest neighbor search** and denote the neighbor relative to $i \in A$ as $\mathcal{N}(i)$. The j-th nearest neighbor's index is denoted as $n_i(j)$.

2. Define the **core distance** as $d_{ij} = 1 - \dfrac{|a_i - b_j|}{|a_i - b_{n_i(k)}|}$, which is the normalized distance from the boundary of the neighbor to the embedding vectors.

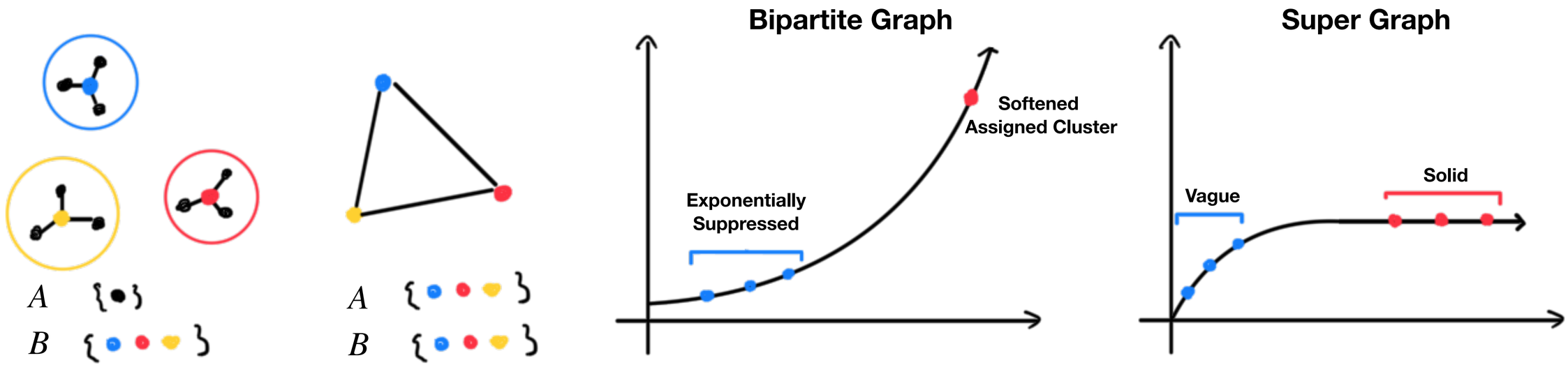
3. The graph is constructed as $G = \left\{(i,j) \,|\, i \in A, j \in \mathcal{N}(i)\right\}$, associated with edge weights $w_{ij} = \dfrac{f(d_{ij})}{\sum_{m\in\mathcal{N}(i)} f(d_{im})}$ with weighting function $f(x)$.





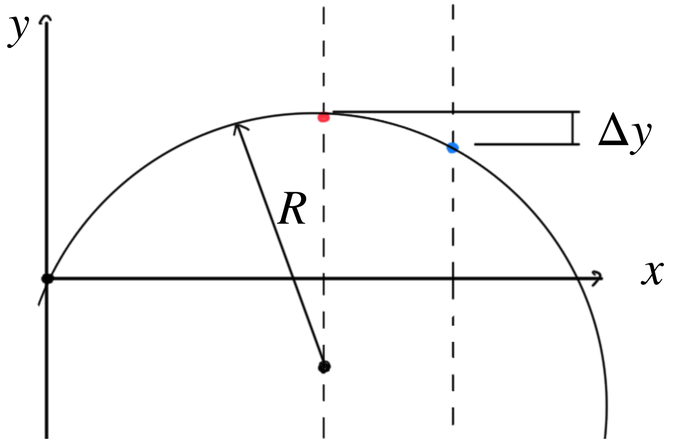
Hierarchical Graph Construction

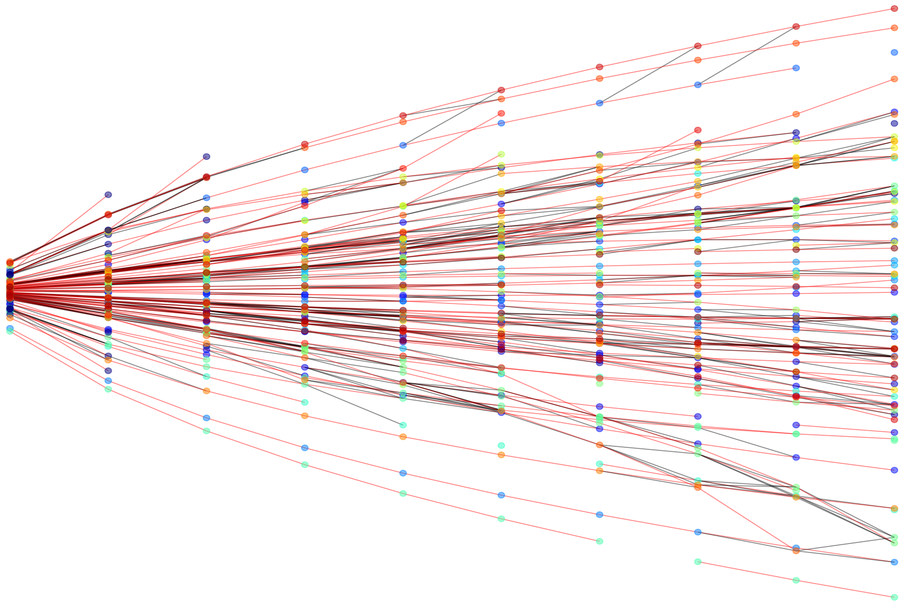
- For cluster assignment, the sets A and B is the **nodes** and **super nodes** respectively. Their embeddings are given by **node embeddings** and the **centroids** computed. We choose weighting function $f(x) = e^{2x}$ to make assignments pseudo single-valued. This will make farther connections exponentially suppressed.
- For super graph construction, the sets A and B are identical, the **super nodes**. Their embeddings are given by the **centroids** computed. We choose weighting function $f(x) = \tanh(2x)$ so that the plateau region can guarantee that "solid" connections will share mostly the same weight.



Toy Dataset

- Randomly generate 2D circular tracks which travels through 10 layer of detectors.
- True edges are sampled from edges that connect two adjacent hits of a particle.
- Fakes edges are sampled from fully connected graph so that the circumcircle radii R are uniformly distributed and that $\Delta y/R < 1/\#\text{layers}$.
- The task is to find all tracks with $R > 1$and have more than 3 hits. Here, R has similar role as p_T since under uniform magnetic field $R \propto p_T$.





Training & Evaluation

- Use Hierarchical GNN with 4 interaction iterations and 6 hierarchical iterations/Interaction GNN with 10 iterations to embed hits into a space and perform spatial clustering to attract track candidates.
- The loss function uses FRNN search to get hinge loss: (here x_i refers to the embedding of node i.)

$$L(x, y) = \begin{cases} |x_i - x_j| & \text{if } (i, j) \text{ is a true edge} \\ 1 - |x_i - x_j| & \text{if } (i, j) \text{ is a fake edge} \end{cases}$$

- Truth is defined as follows:

  True: sequential true edges (the same particle at two adjacent layers)

  Neutral: PID true edges (the same particle)

  Fake: PID fake edges (two different particles)

Results

- The metric used here is the tracking efficiency, defined as:

$$\mathscr{E} = \frac{\#\text{particles that occupied more than 50\% of the assigned cluster}}{\#\text{particles}}$$

- For all runs Hierarchical GNN is performing better than Interaction GNN
- Both of them scale well with number of tracks.
- Hierarchical GNN is more advantageous when the graph has high purity.

90% efficient, 50% pure, 10 layer of detectors

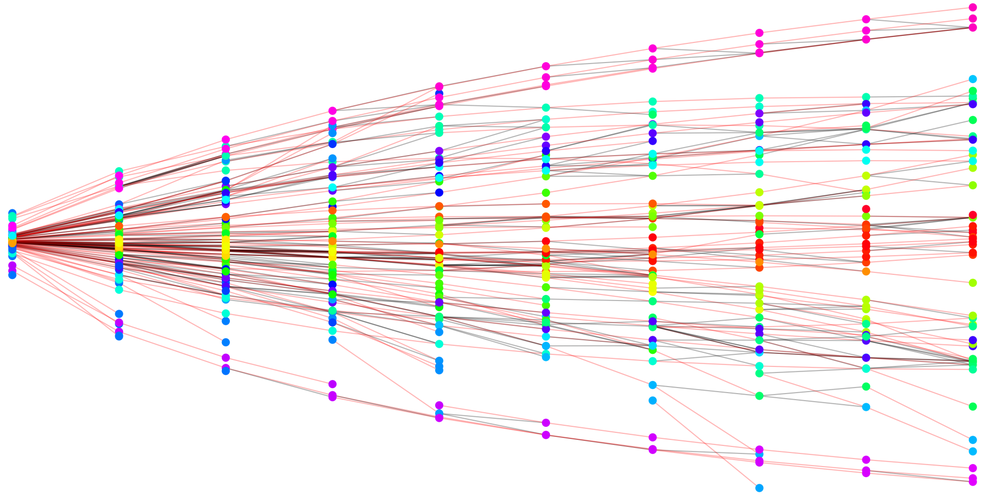
Number of Tracks	100	200	500
Hierarchical	99.34%	99.10%	98.65%
Interaction	99.27%	98.94%	98.50%

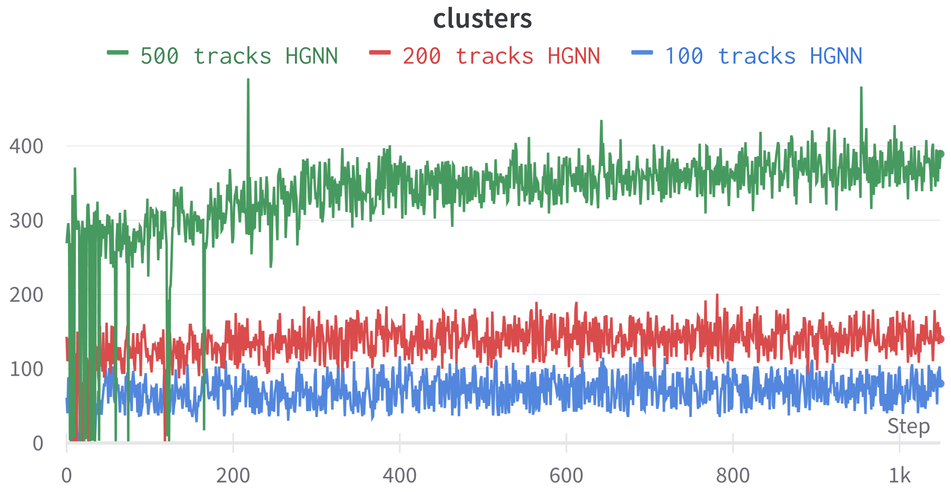
400 tracks, 10 layer of detectors

Hierarchical/Interaction	90% Efficient	80% Efficient
50% Pure	99.40% / 98.98%	97.62% / 96.32%
30% Pure	98.98% / 98.90%	96.23% / 95.88%

What Does the Pooling Pool?

- This is the clustering result obtained during the super graph construction
- The coloring is done by calculating the centroids of each cluster, performing T-SNE to reduce the dimensionality, and finally using colormap to obtain colors.
- Intermediate/Final performance:
  - Tracking efficiency: 44.44% (99.3%)
  - In-cluster purity: 86.03% (98.58%)
  - Fake rate: 48.39% (13.42%)





References

1. Battaglia, P., Pascanu, R., Lai, M., & Jimenez Rezende, D. (2016). Interaction networks for learning about objects, relations and physics. Advances in neural information processing systems, 29.
2. Gao, H., & Ji, S. (2019, May). Graph u-nets. In international conference on machine learning (pp. 2083-2092). PMLR.
3. Qasim, S. R., Kieseler, J., Iiyama, Y., & Pierini, M. (2019). Learning representations of irregular particle-detector geometry with distance-weighted graph networks. The European Physical Journal C, 79(7), 1-11.
4. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Philip, S. Y. (2020). A comprehensive survey on graph neural networks. IEEE transactions on neural networks and learning systems, 32(1), 4-24.
5. Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., & Leskovec, J. (2018). Hierarchical graph representation learning with differentiable pooling. Advances in neural information processing systems, 31.