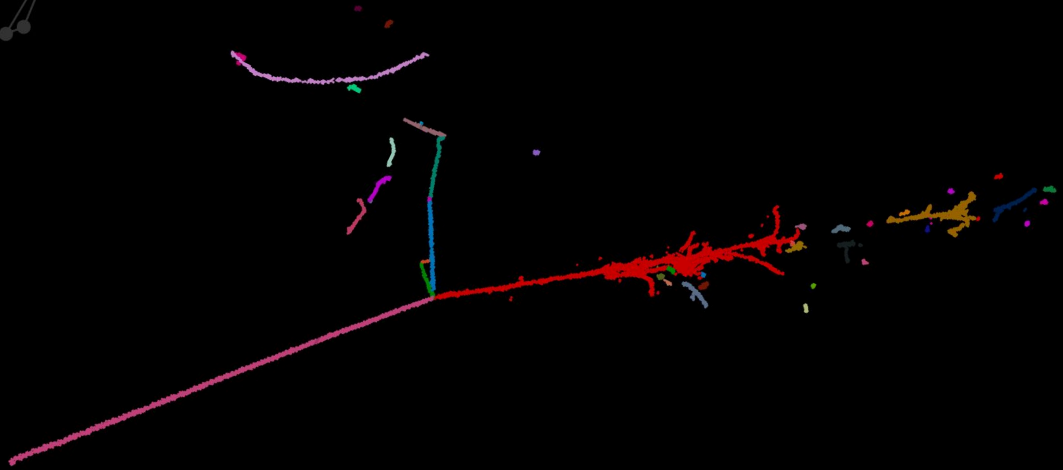
Graph Neural Networks Applications II

SLAC Summer Institute 2023

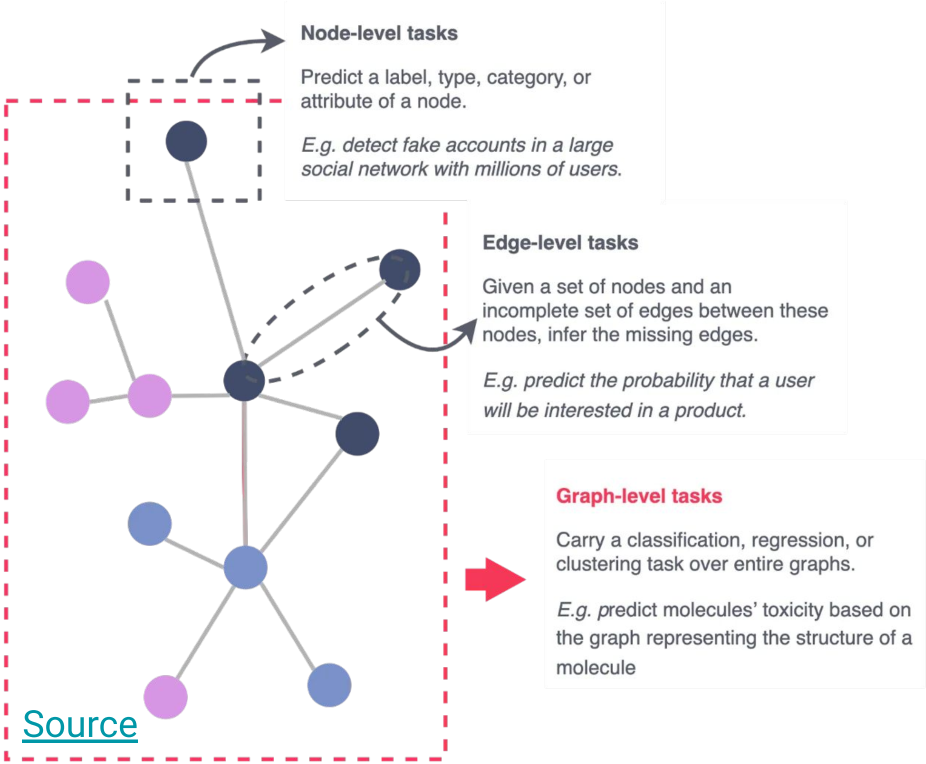
François Drielsma (SLAC)



SLAC NATIONAL ACCELERATOR LABORATORY

Stanford University | U.S. DEPARTMENT OF ENERGY

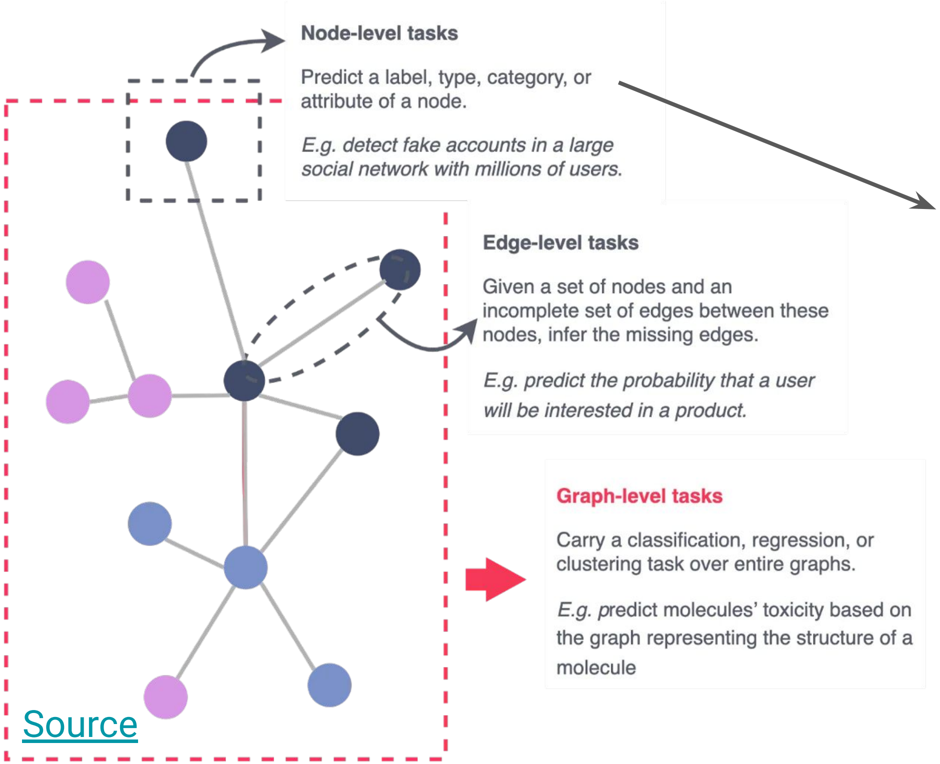
A Tale of Three Papers



[Source]

Graphs are **versatile** beasts

A Tale of Three Papers



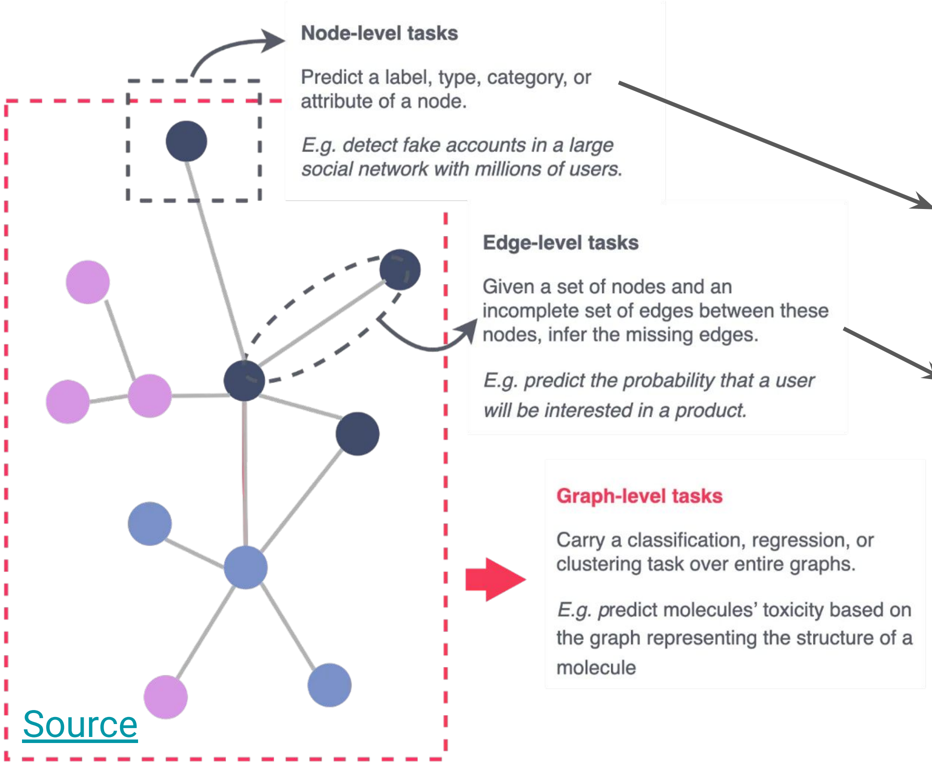
Source

Graphs are **versatile** beasts

Three **examples** at the **IF**:

1. SuperFGD pixel classification (PhysRevD.103.032005)

A Tale of Three Papers

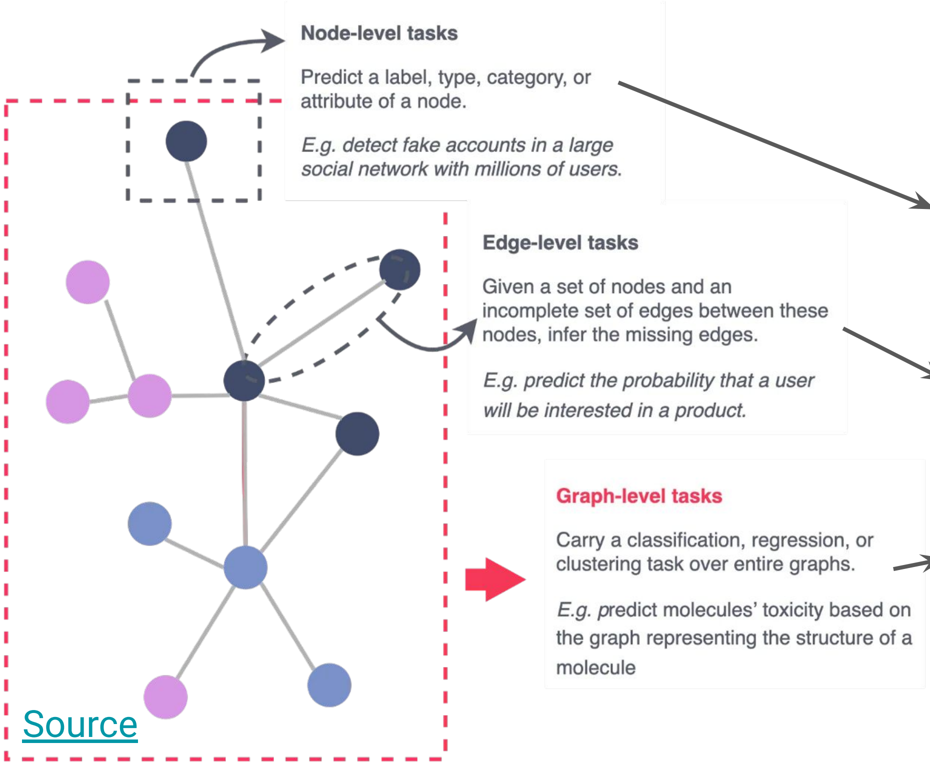


Graphs are **versatile** beasts

Three **examples** at the **IF**:

1. SuperFGD pixel classification (PhysRevD.103.032005)
2. LArTPC particle aggregation (PhysRevD.104.072004)

A Tale of Three Papers



Graphs are **versatile** beasts

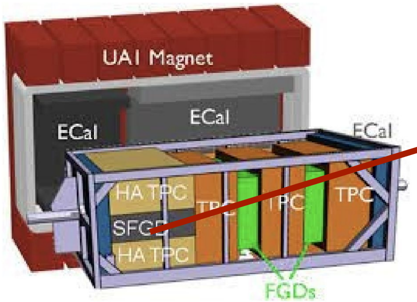
Three **examples** at the **IF**:

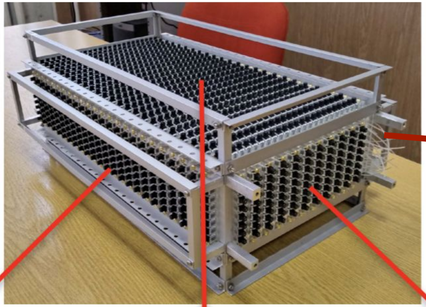
1. SuperFGD pixel classification (PhysRevD.103.032005)
2. LArTPC particle aggregation (PhysRevD.104.072004)
3. IceCube event classification (10.1109/ICMLA.2018.00064)

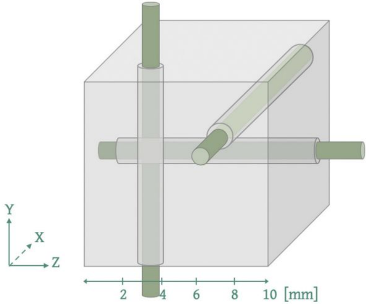
Pixel Classification in T2K’s SuperFGD

T2K’s Super Fine-Grained Detector (SuperFGD)

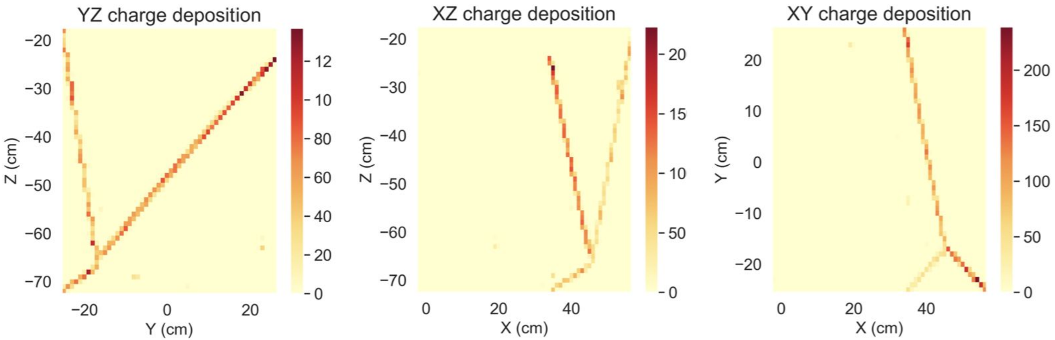
New ND280 configuration





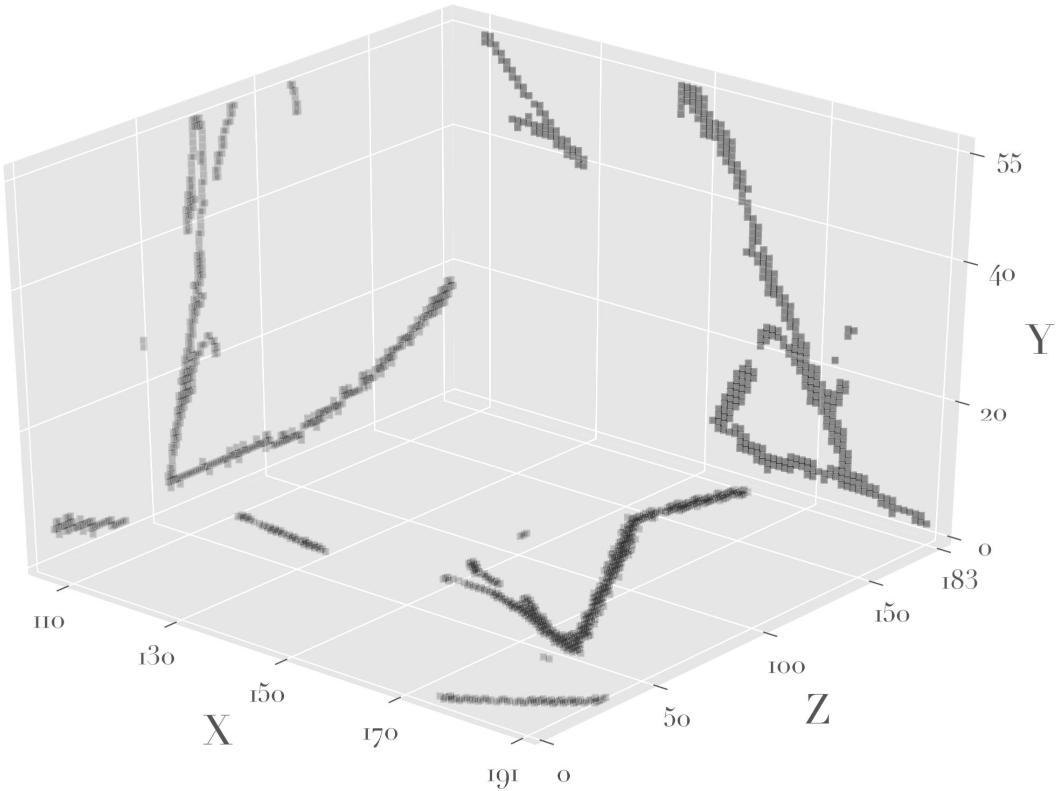


56 × 184 × 192 1cm^3 cubes



Tomographic Reconstruction

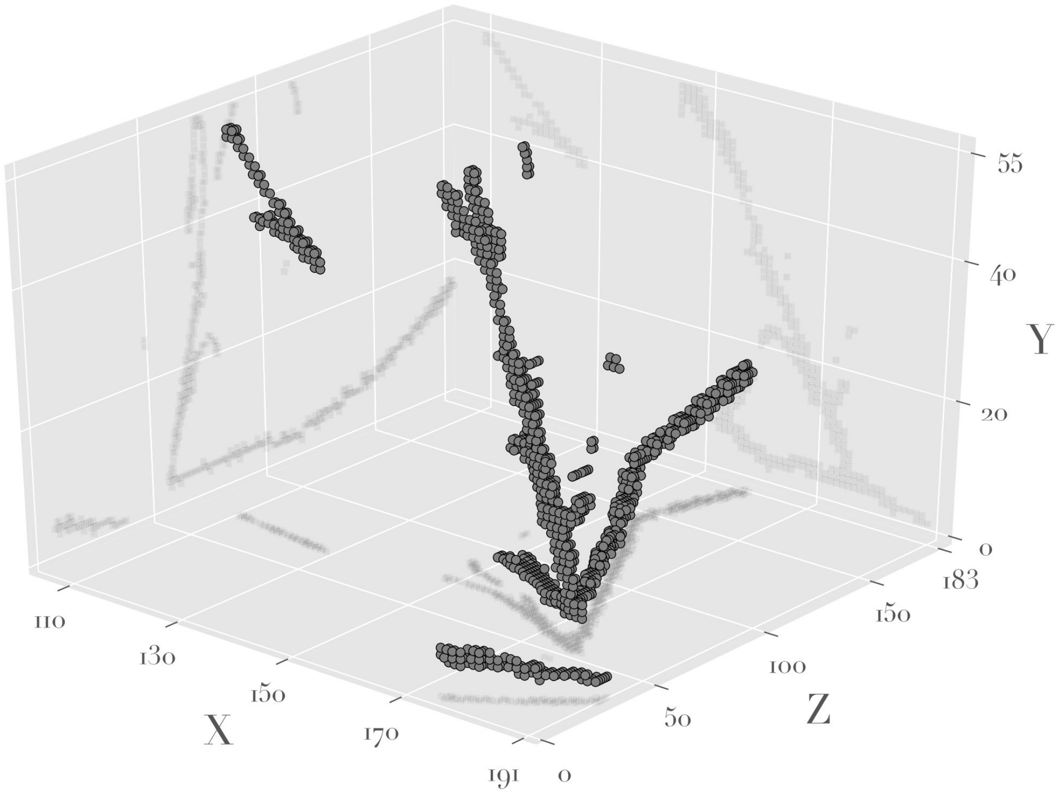
Input: set of three 2D projections



Tomographic Reconstruction

Input: set of three 2D projections

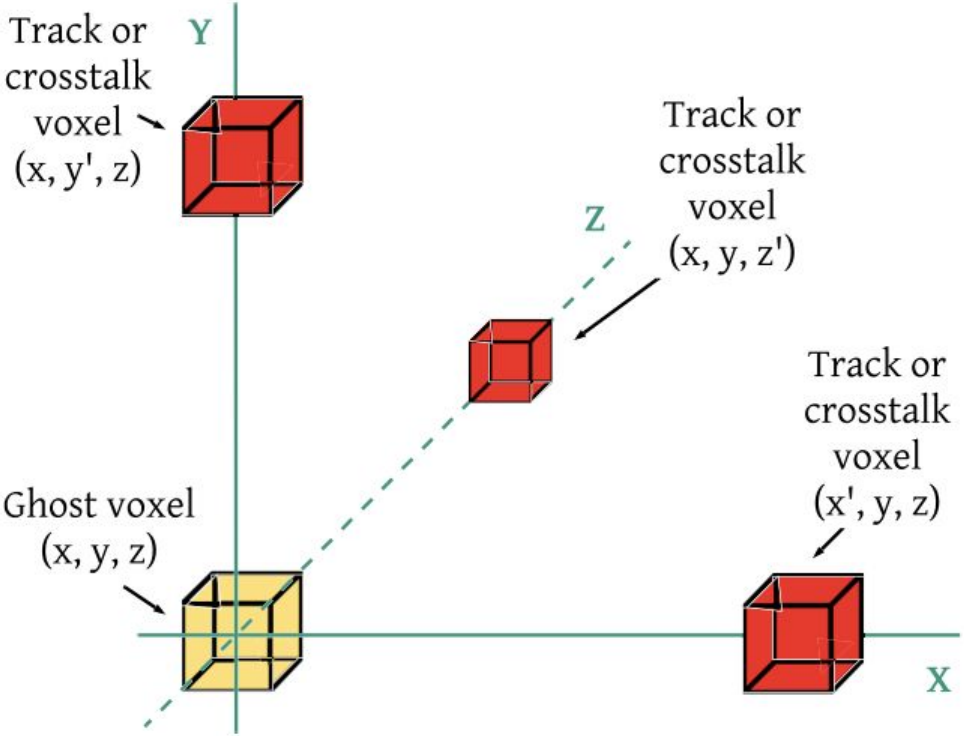
Tomographic reconstruction: make 3D **voxels**



Tomographic Reconstruction

Input: set of three 2D projections

Tomographic reconstruction: make 3D **voxels**

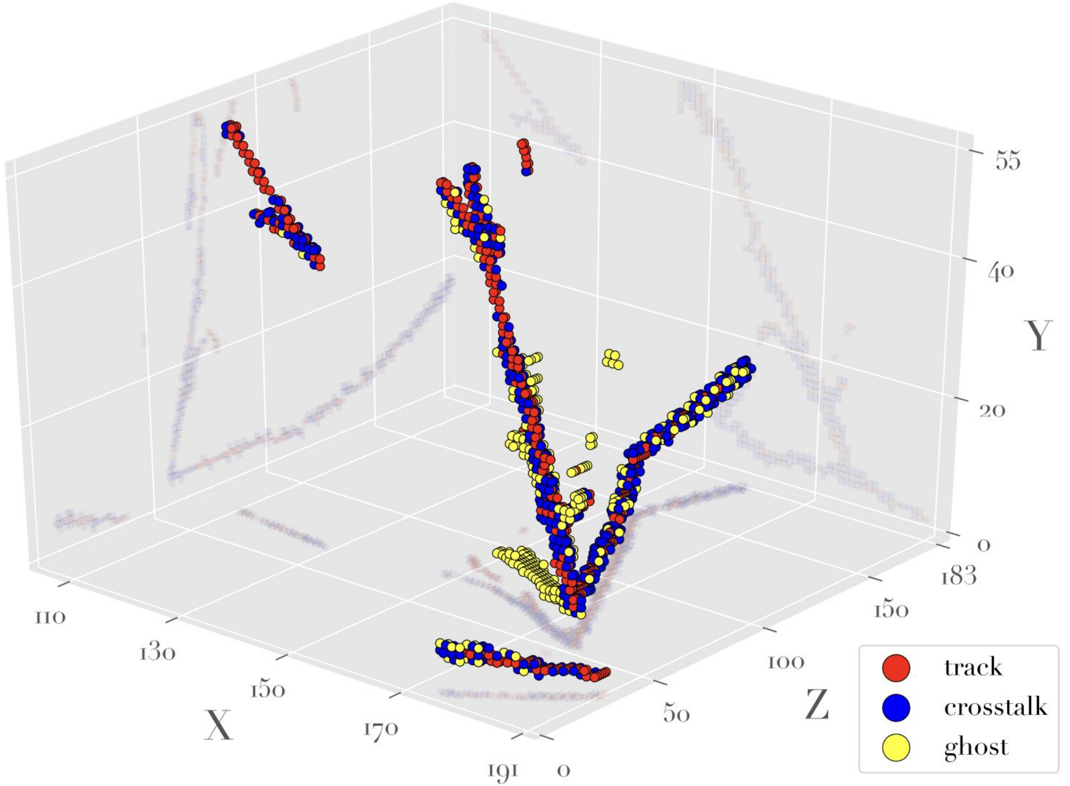


Tomographic Reconstruction

Input: set of three 2D projections

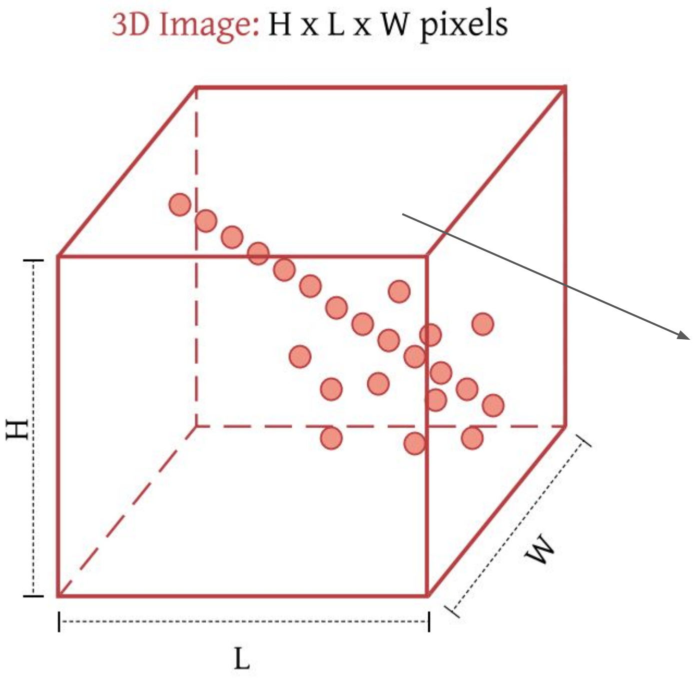
Tomographic reconstruction: make 3D **voxels**

Task: Classify individual voxels into classes, i.e. **semantic segmentation**



Data Representation

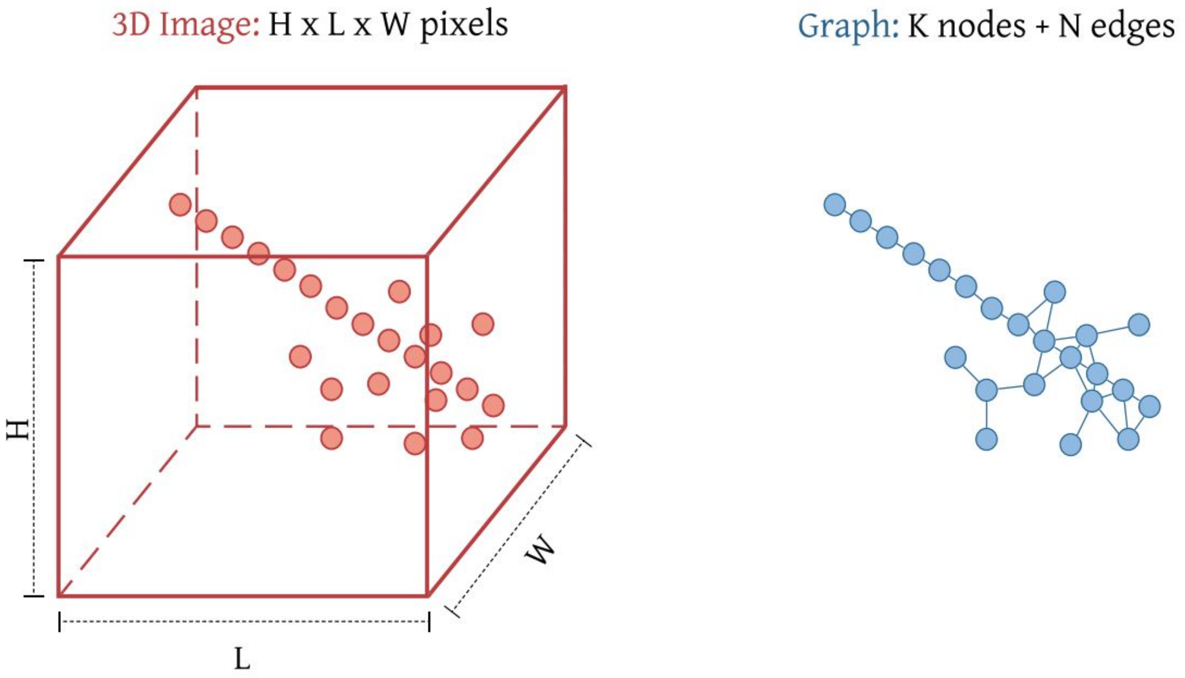
Could use a **Dense 3D CNN**, but… images are **very sparse**



~99.98 % empty!

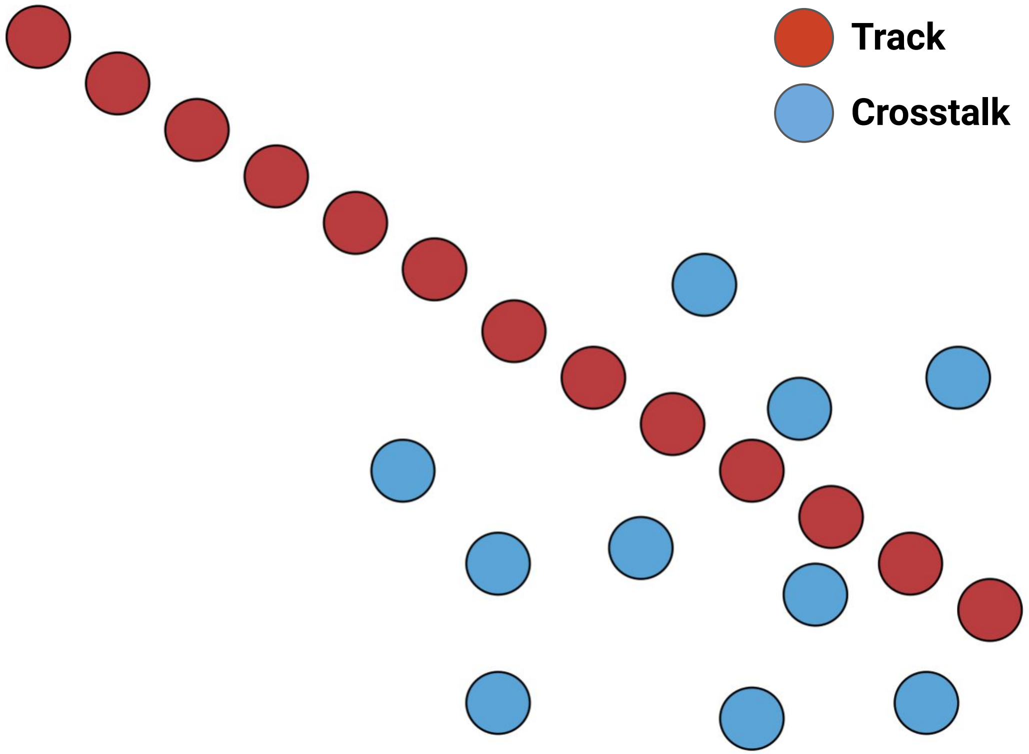
Data Representation

Graphs are a more **efficient representation** of sparse images



Graph Construction

Which **edges** do we **need**?

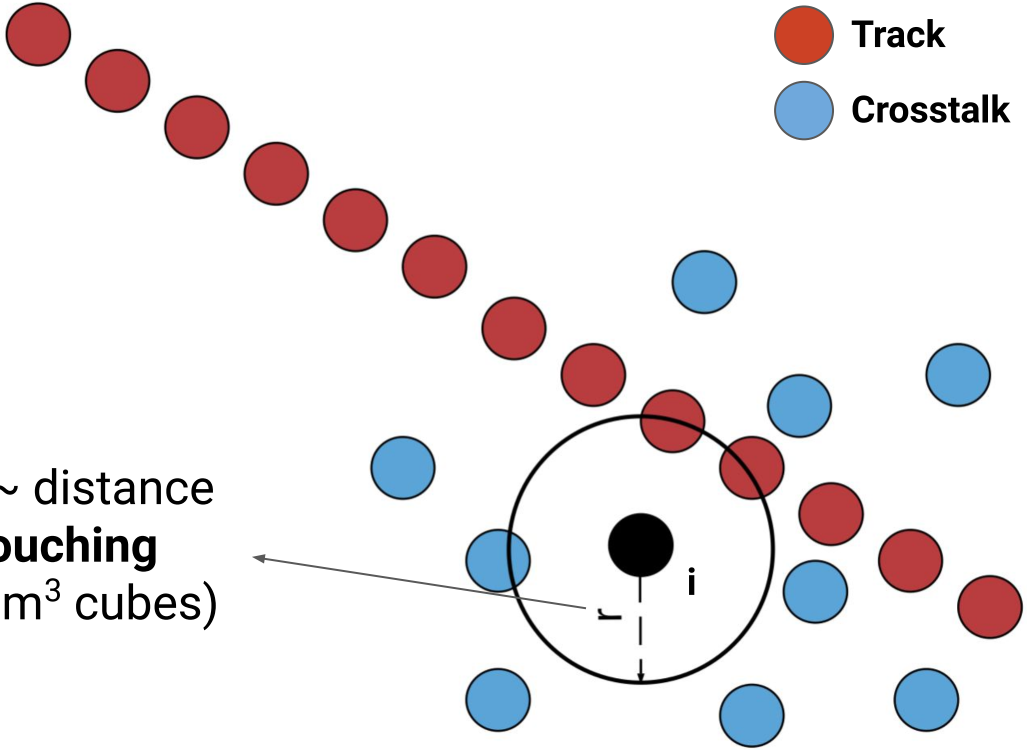


Graph Construction

Which **edges** do we **need**?

Given a **radius**, r, define **neighborhood** of node i

Picked to be **1.75 cm** ~ distance between two **corner-touching** voxel centers (1x1x1 cm^3 cubes)



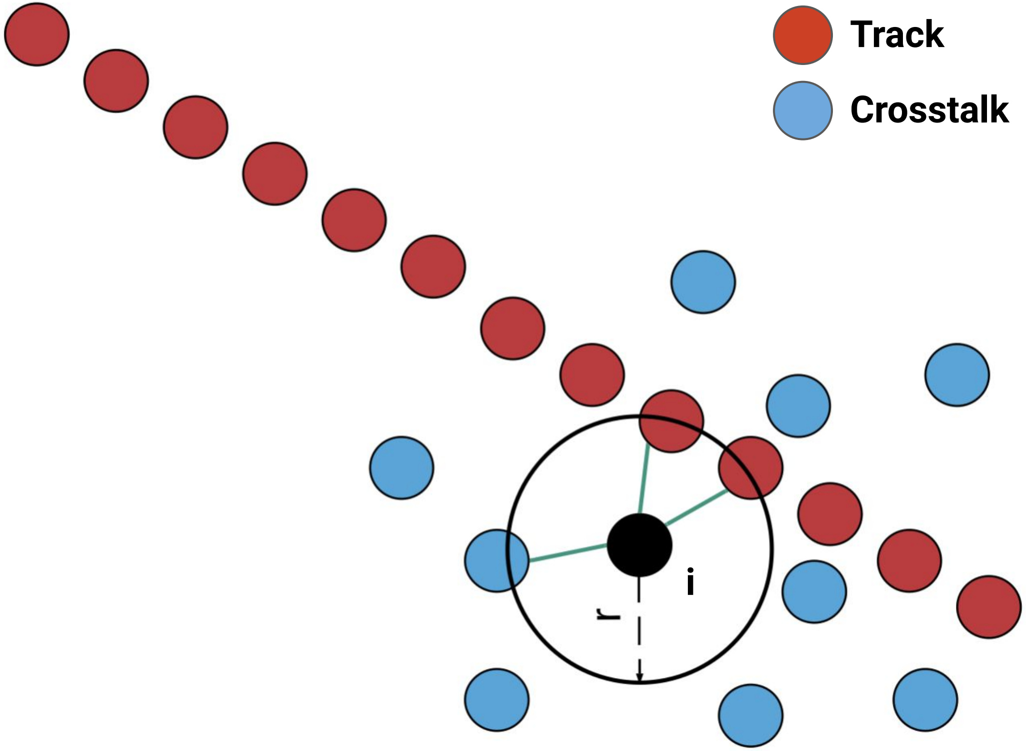
Graph Construction

Which **edges** do we **need**?

Given a **radius**, r, define **neighborhood** of node i

Build **undirected edges** between node and its **neighbors**,

$$\{e_{ij}\}_{j\in\mathcal{N}(i)}$$



Graph Construction

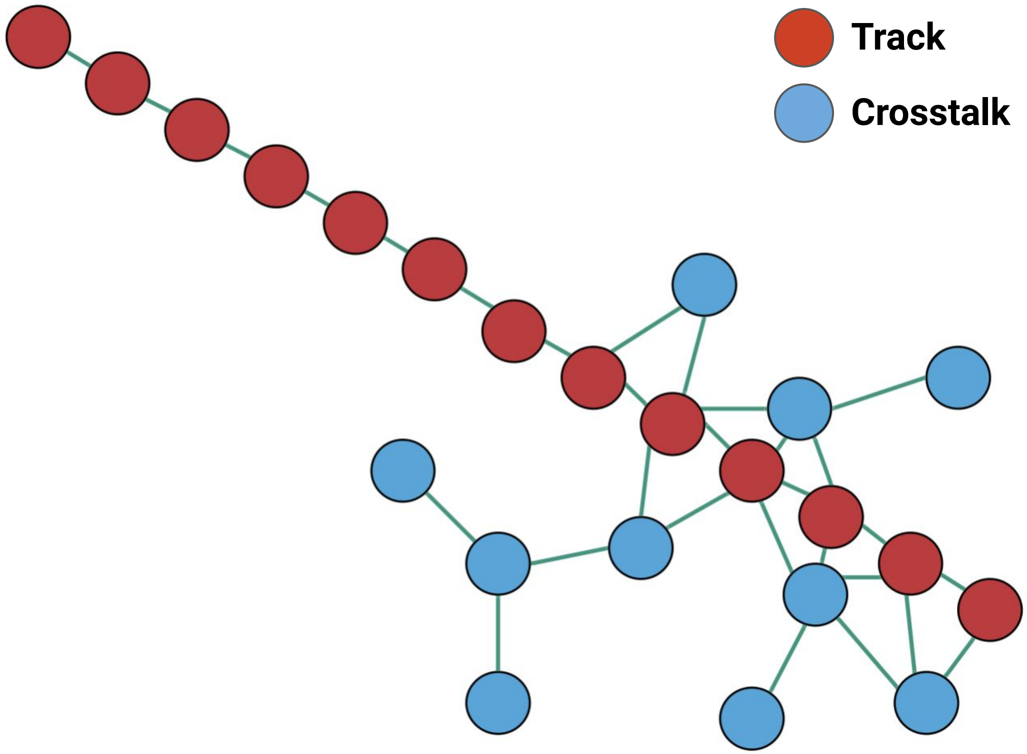
Which **edges** do we **need**?

Given a **radius**, r, define **neighborhood** of node i

Build **undirected edges** between node and its **neighbors**,

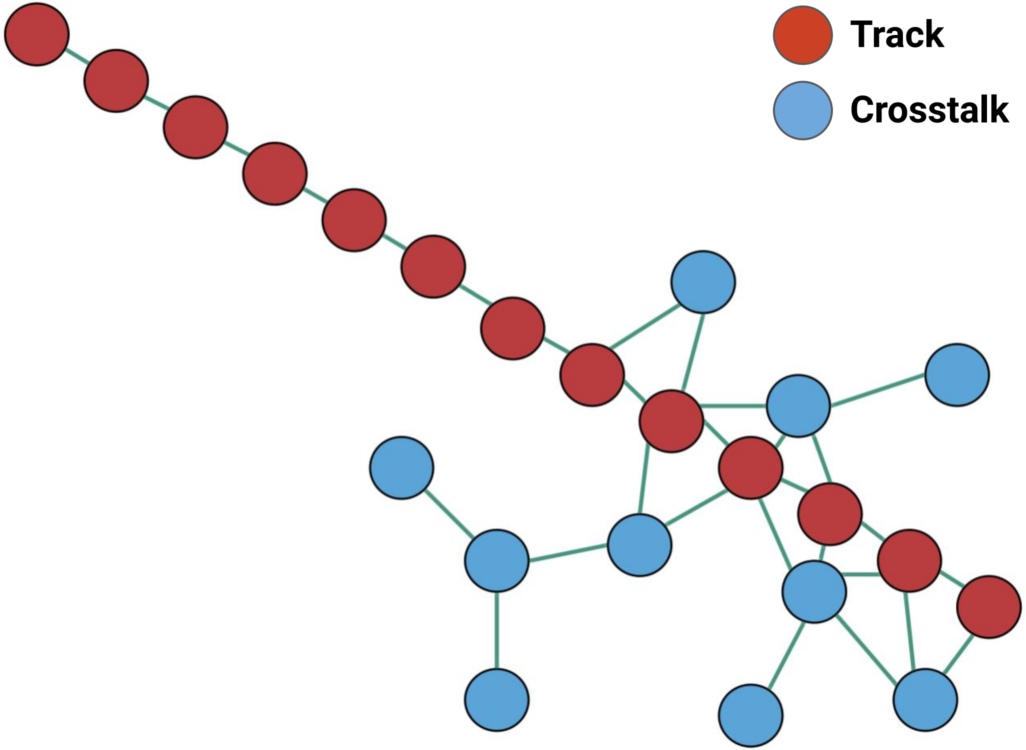
$$\{e_{ij}\}_{j\in\mathcal{N}(i)}$$

Repeat for **all nodes**



Node embedding

What **information** to pass to the graph?

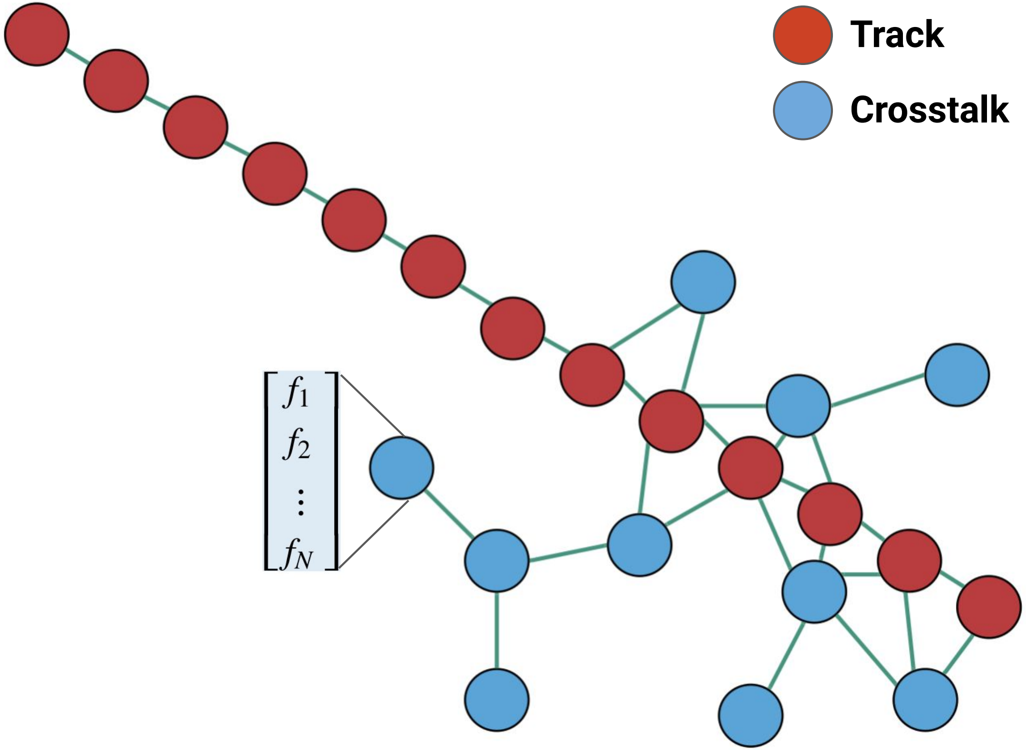


Node embedding

What **information** to pass to the graph?

Nodes (space points) are encoded as **vectors** of 25 **features**, e.g.

- Number of **photons**
- Fiber **multiplicity**
- Light **fluctuation** between planes
- . . .

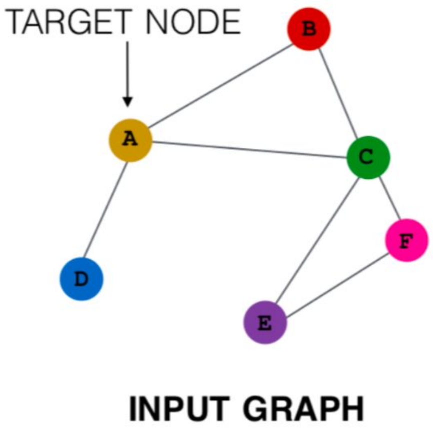


Graph Convolutions (Graph-SAGE)

How is information **communicated**?

- Neighborhood = computation graph

arXiv:1706.02216

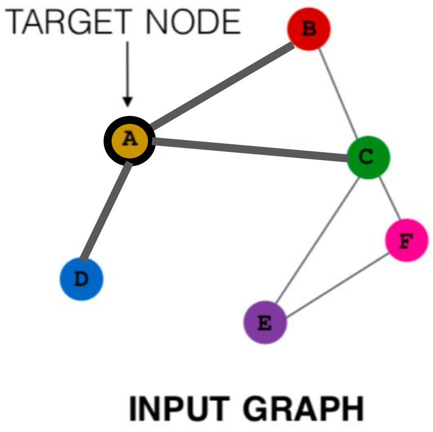


Graph Convolutions (Graph-SAGE)

How is information **communicated**?

- Neighborhood = computation graph

arXiv:1706.02216

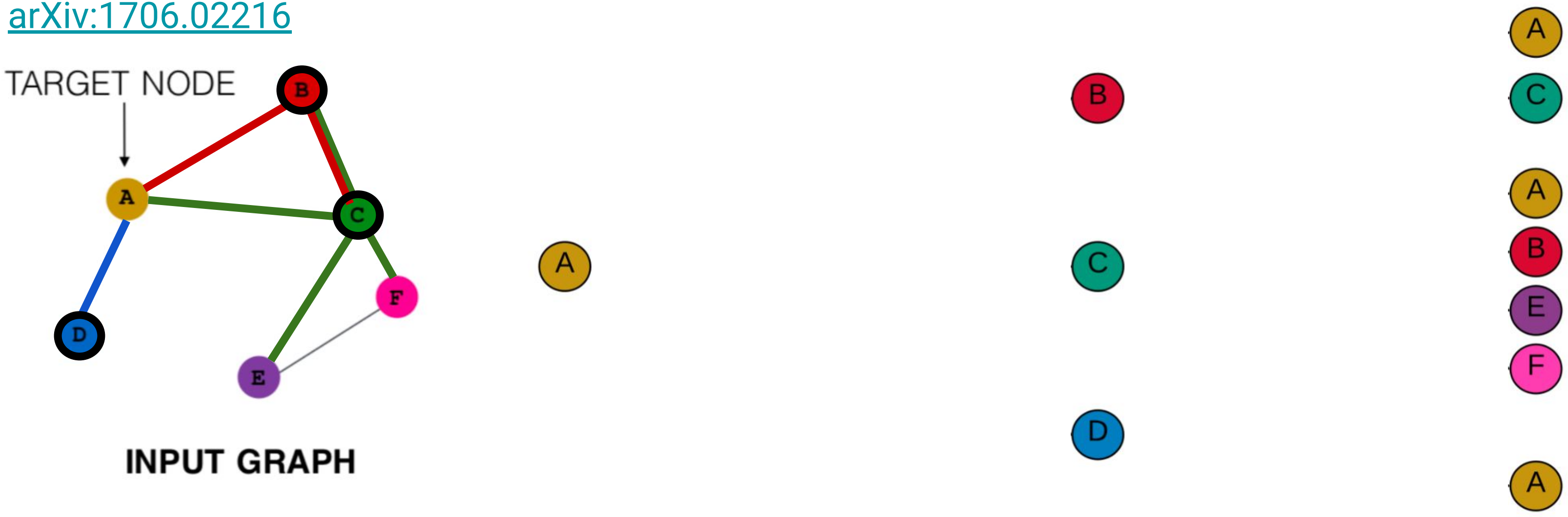


Graph Convolutions (Graph-SAGE)

How is information **communicated**?

- Neighborhood = computation graph

arXiv:1706.02216



Graph Convolutions (Graph-SAGE)

How is information **communicated**?

- Neighborhood = computation graph

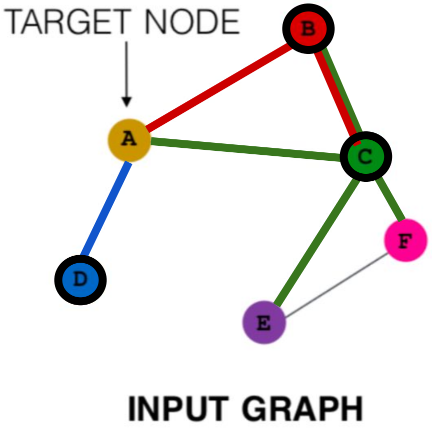
arXiv:1706.02216

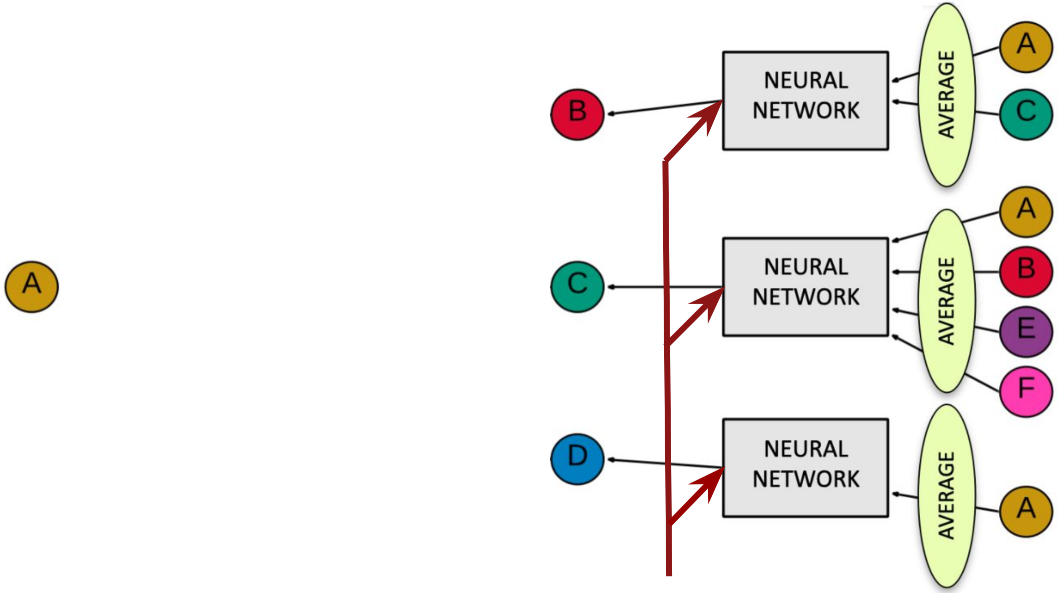
Graph Convolutions (Graph-SAGE)

How is information **communicated**?

- Neighborhood = computation graph

arXiv:1706.02216





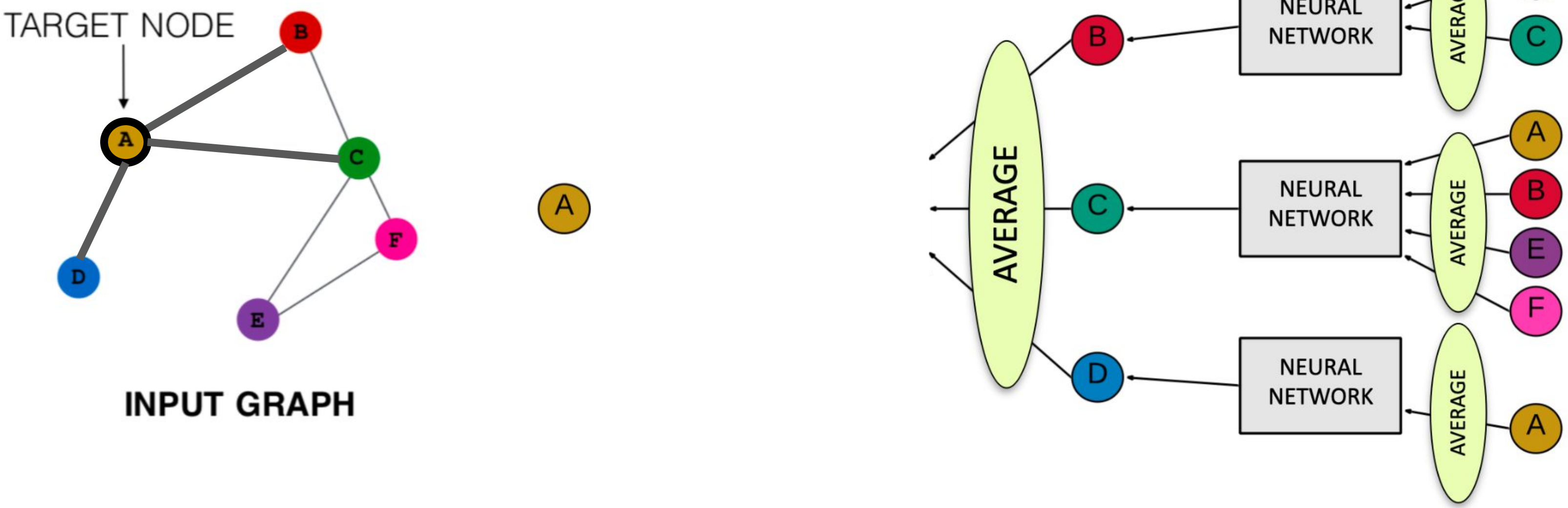
Shared (depth 2)

Graph Convolutions (Graph-SAGE)

How is information **communicated**?

- Neighborhood = computation graph

arXiv:1706.02216

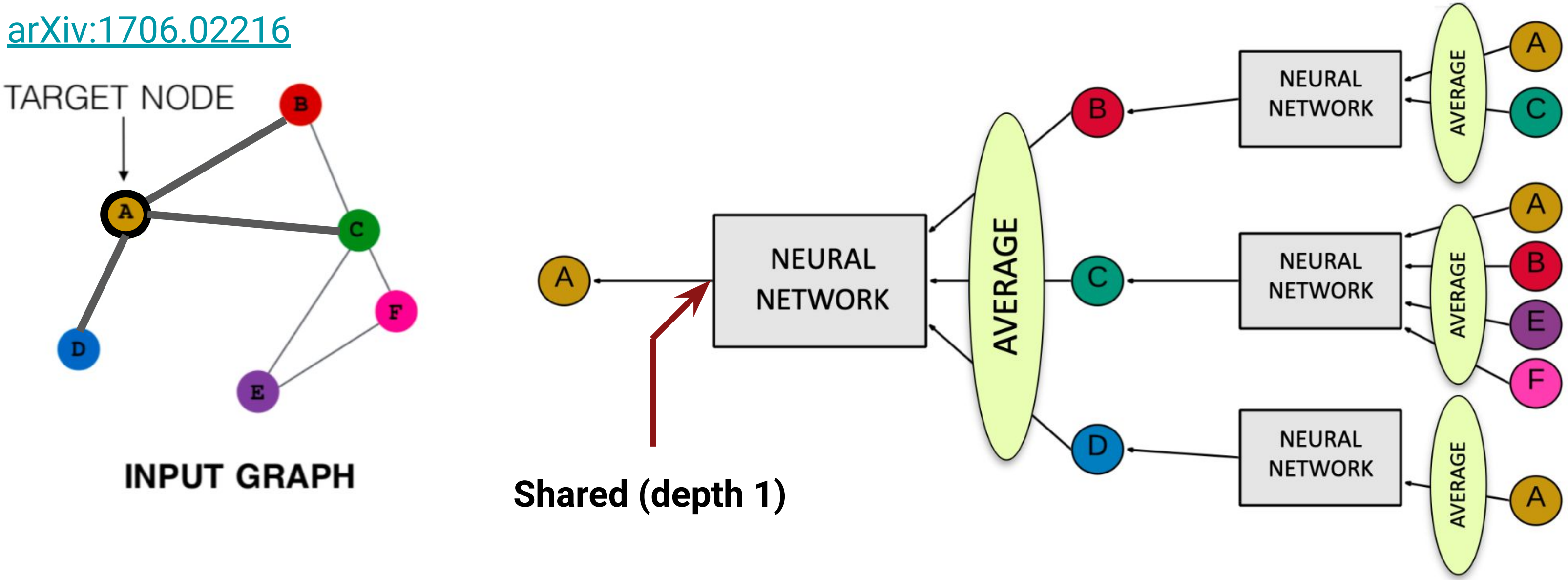


Graph Convolutions (Graph-SAGE)

How is information **communicated**?

- Neighborhood = computation graph

arXiv:1706.02216

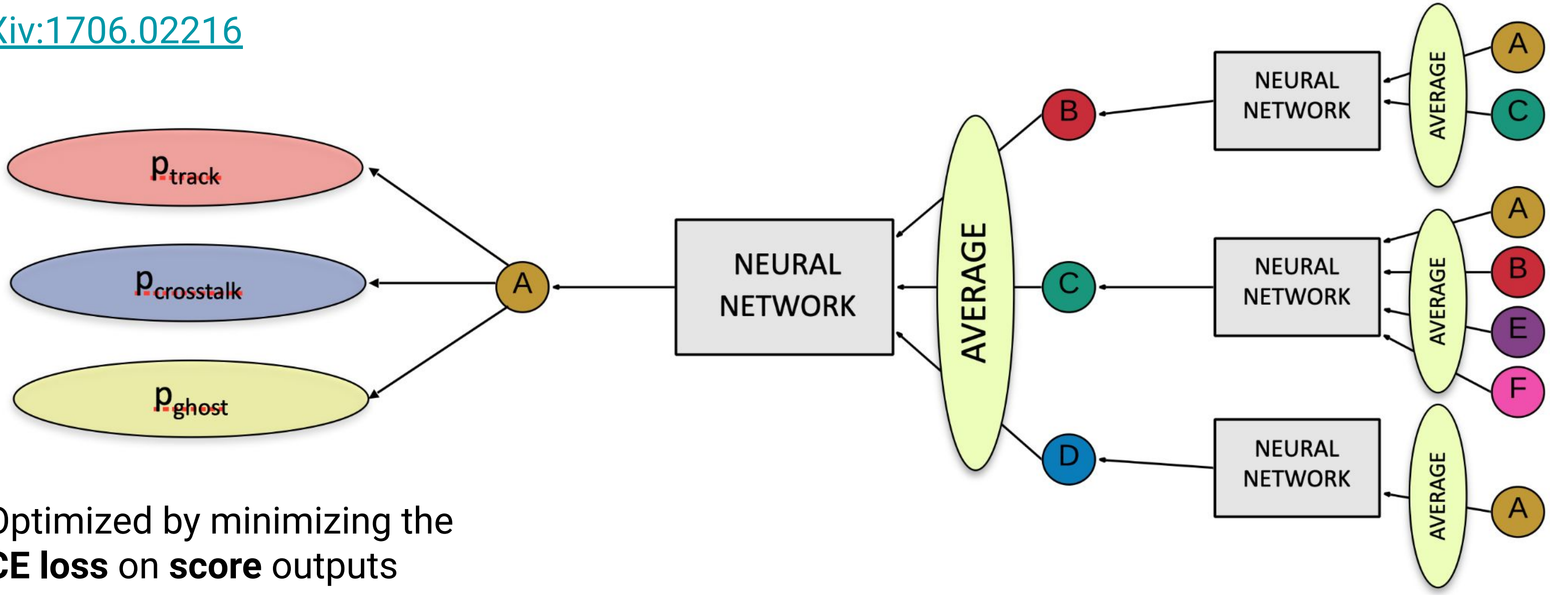


Graph Convolutions (Graph-SAGE)

How is information **communicated**?

- Neighborhood = computation graph

[arXiv:1706.02216](https://arxiv.org/abs/1706.02216)



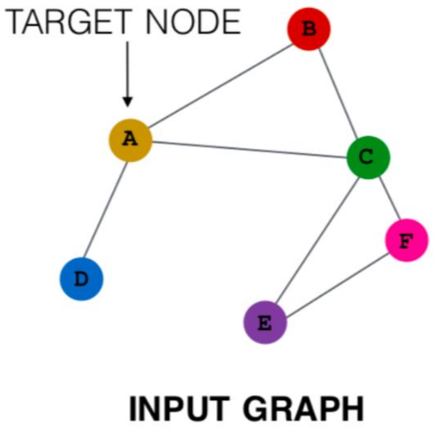
Optimized by minimizing the
CE loss on **score** outputs

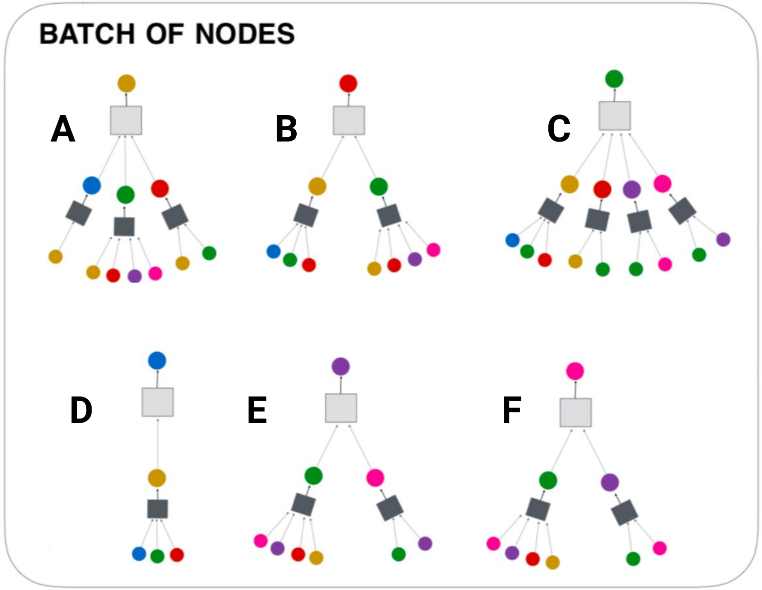
Graph Convolutions (Graph-SAGE)

How is information **communicated**?

- Neighborhood = computation graph

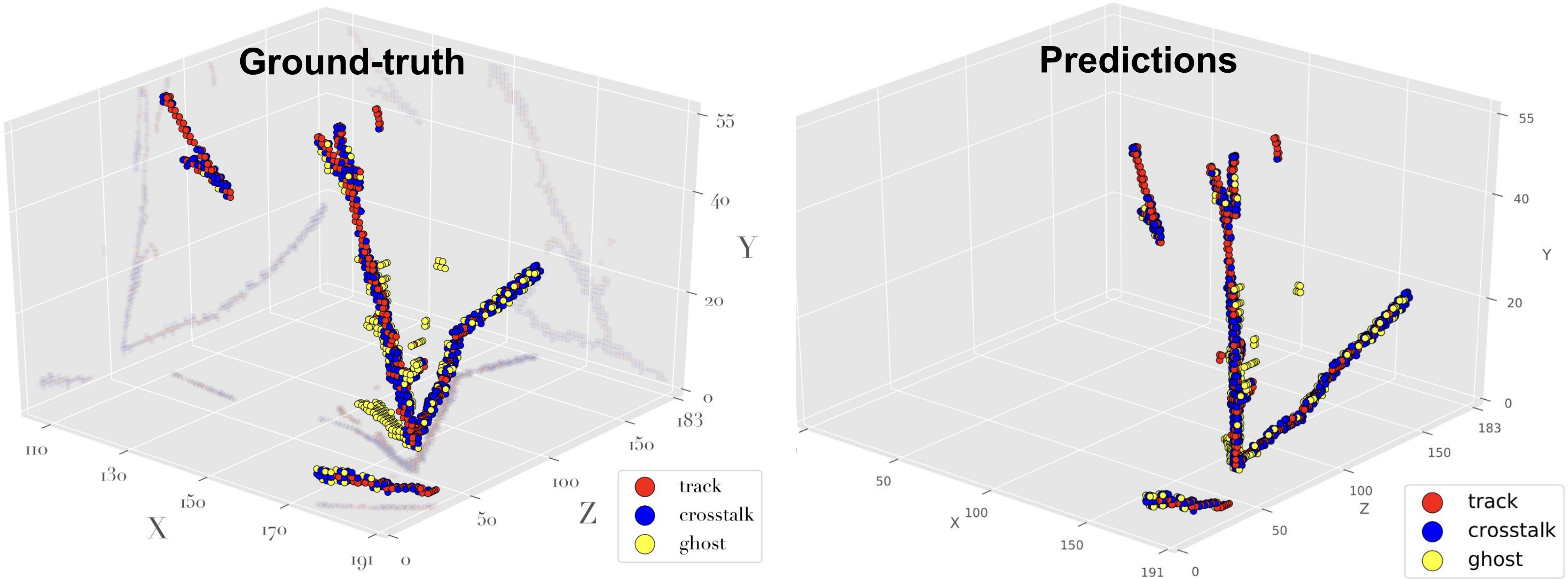
[arXiv:1706.02216](https://arxiv.org/abs/1706.02216)





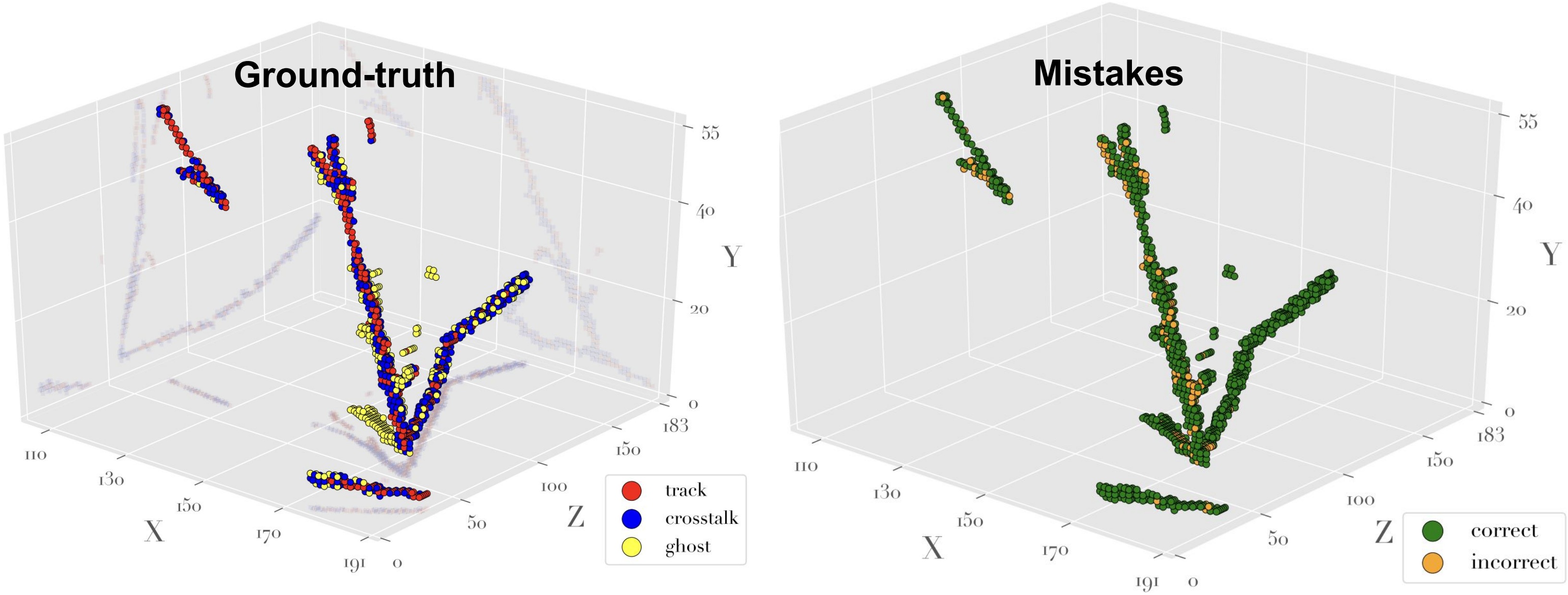
Example Predictions

Ground-truth **labels** vs **predictions**



Example Predictions

Ground-truth **labels** vs **predictions**

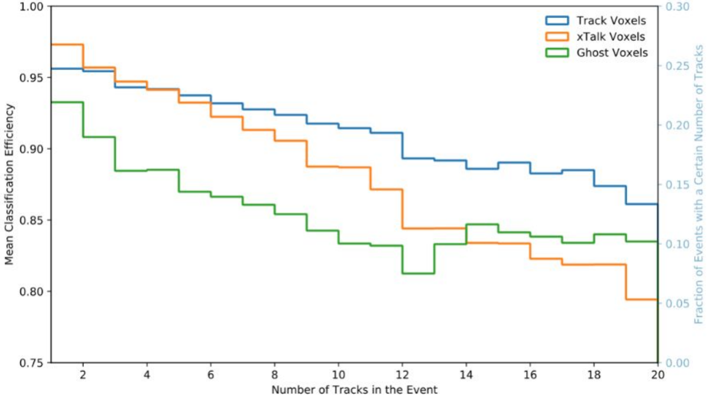


Performance

Quality metrics as a function of **track multiplicity**

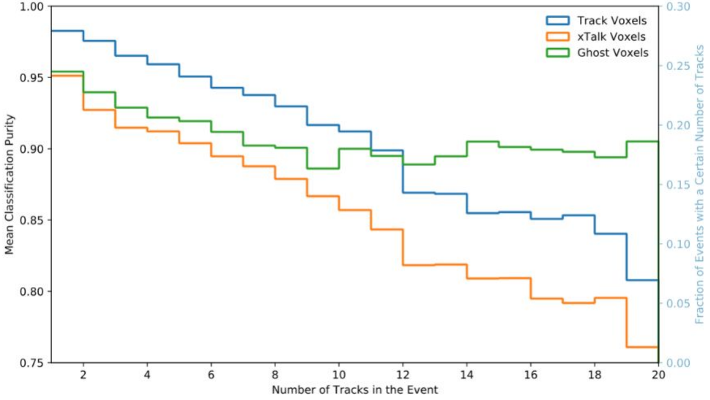
Efficiency

$$E_i = \frac{\#(\mathrm{pred} = \mathrm{label} = i)}{\#(label = i)}$$



Purity

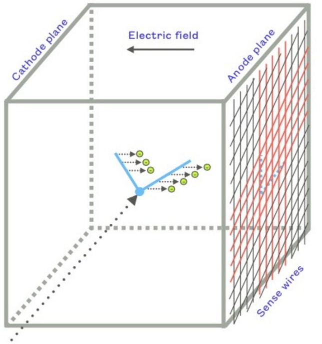
$$P_i = \frac{\#(\mathrm{pred} = \mathrm{label} = i)}{\#(pred = i)}$$

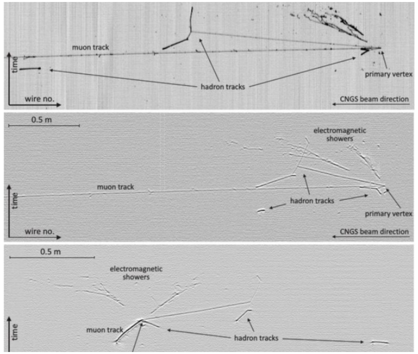


Particle Aggregation in LArTPCs

Liquid Argon Time-Projection Chambers

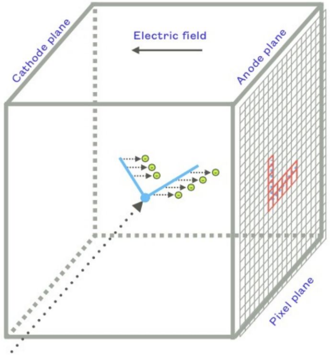
Wire TPC (2D)

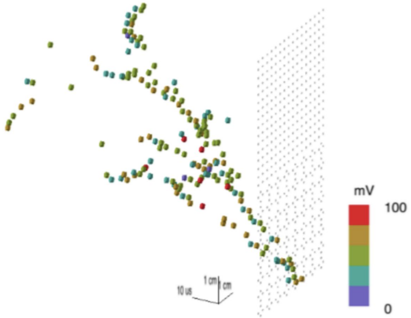




ICARUS, arXiv:1210.5089

Pixel TPC (3D)





LArPIX, arXiv:1808.02969

LArTPC are at the center stage of **beam ν physics** in the US

Short Baseline Neutrino program

- µBooNE, **ICARUS**, **SBND**

DUNE long-baseline experiment

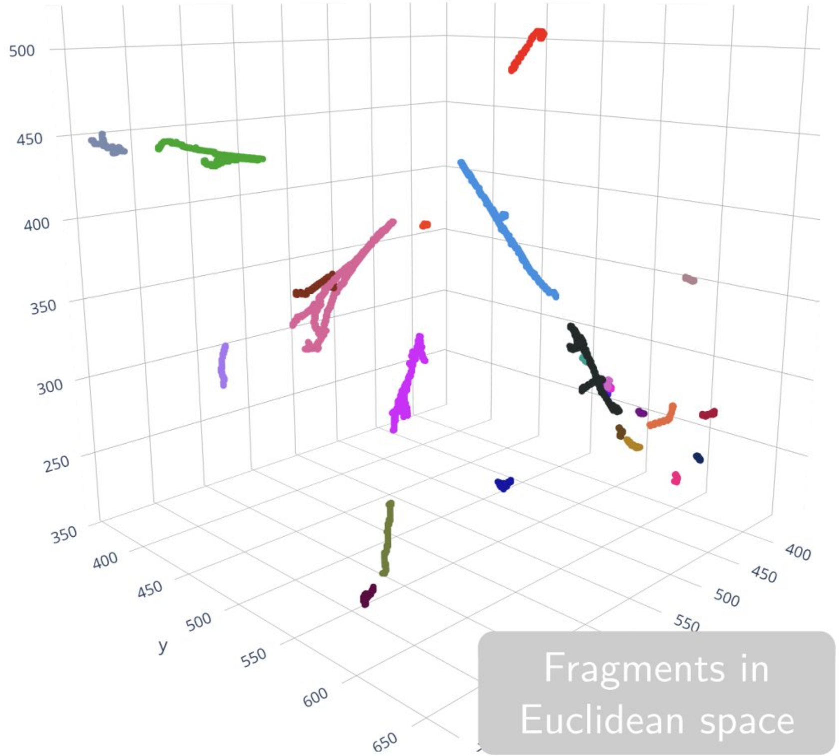
- **Wire**: DUNE FD
- **Pixel**: **DUNE ND-LAr**

Advantages:

- **Detailed:** O(1) mm resolution, precise calorimetry
- **Scalable**: Up to tens of kt

Aggregation

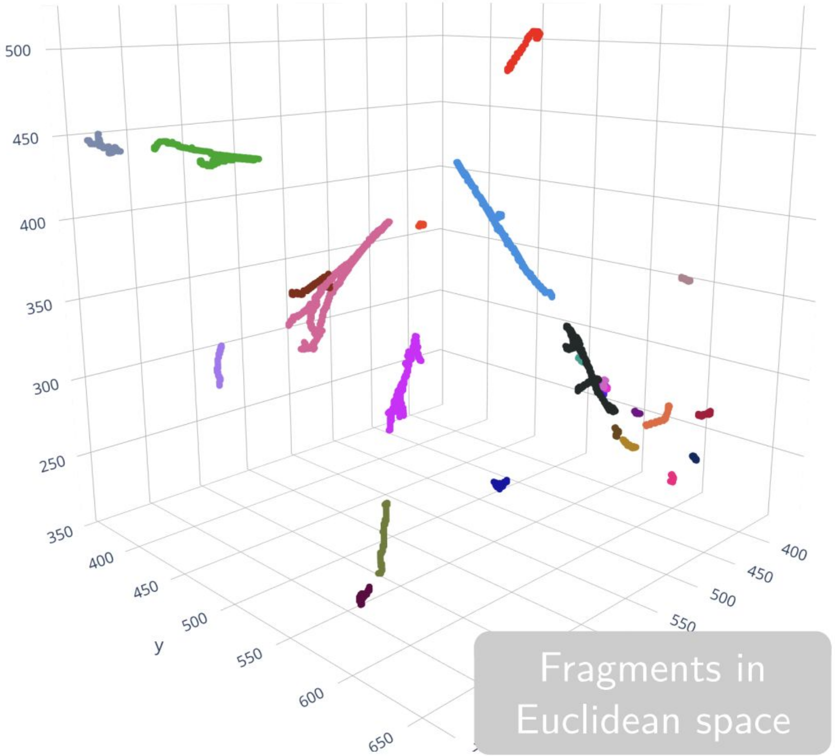
Input: set of particle fragments formed upstream, e.g. **shower fragments**



Aggregation

Input: set of particle fragments formed upstream, e.g. **shower fragments**

Aggregation: build **particles** which fragments belong to

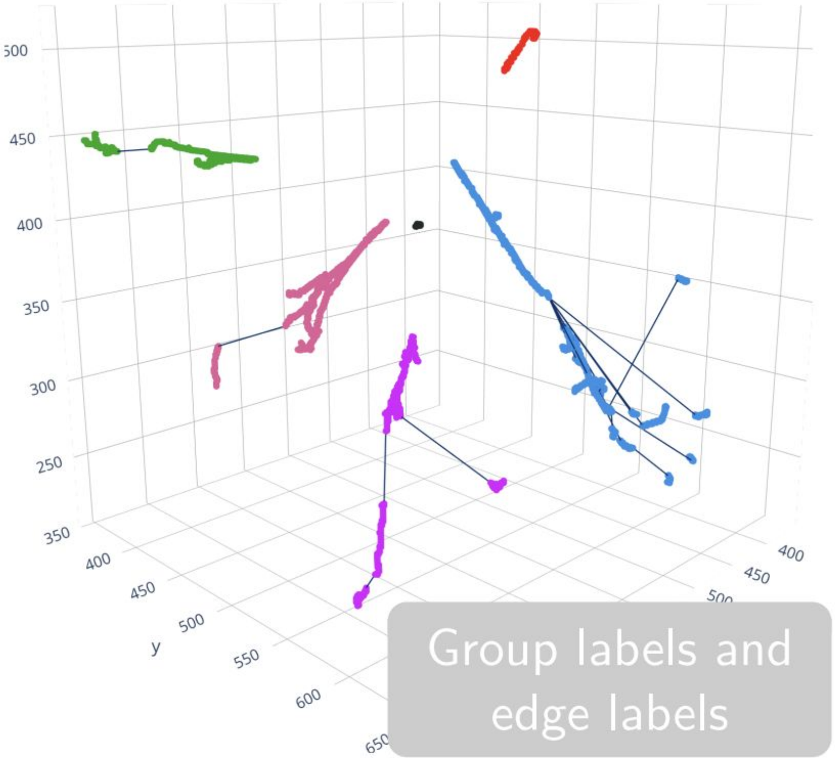


Aggregation

Input: set of particle fragments formed upstream, e.g. **shower fragments**

Aggregation: build **particles** which fragments belong to

Task: Find **connections** between fragments that belong to the same particle



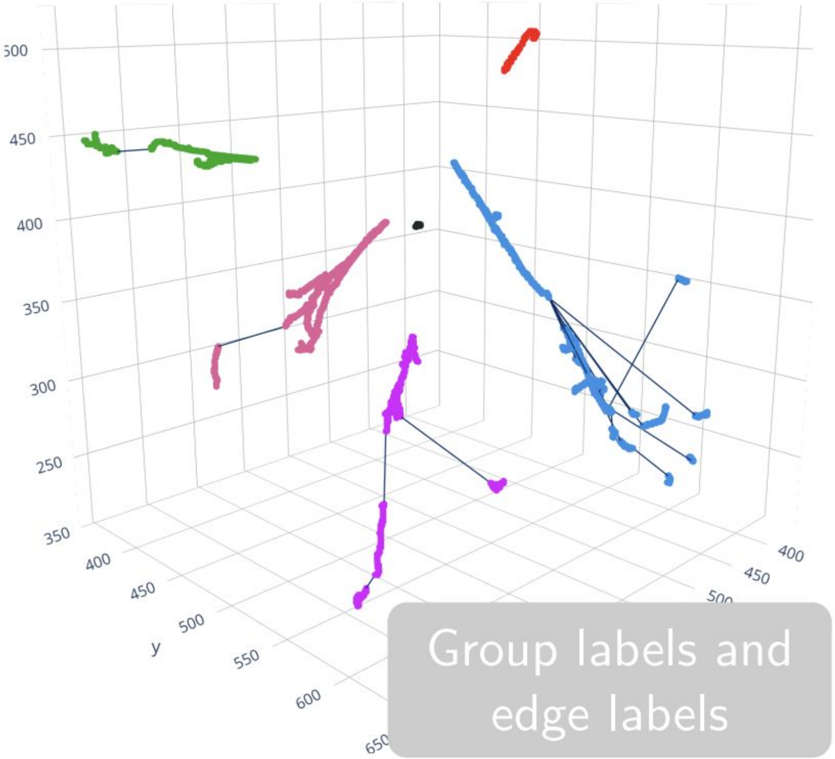
Aggregation

Input: set of particle fragments formed upstream, e.g. **shower fragments**

Aggregation: build **particles** which fragments belong to

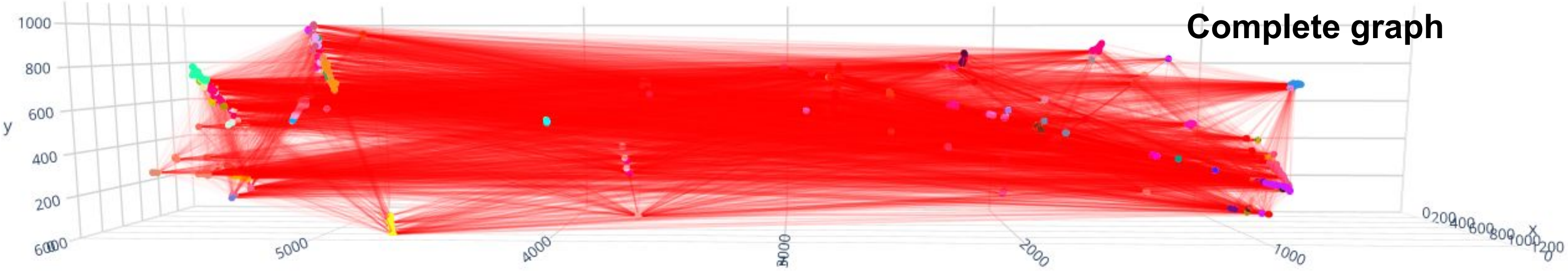
Task: Find **connections** between fragments that belong to the same particle

→ Obvious **graph structure**



Graph Construction

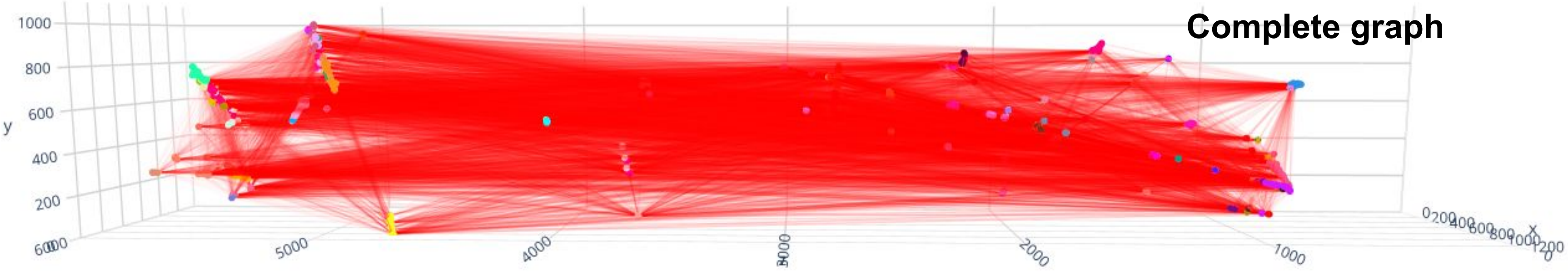
Joining **all possible** pairs of fragments **impractical**



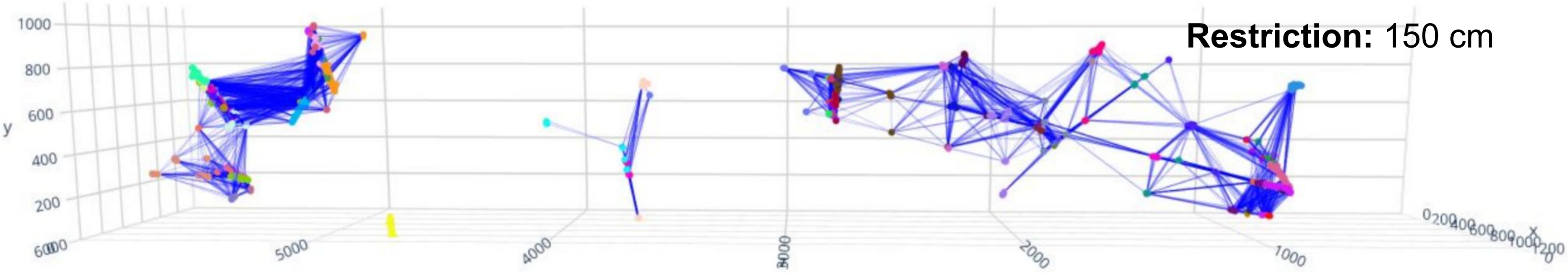
Graph Construction

Choose **natural** distance scale (γ mean free path in LAr ~20 cm)

Complete graph

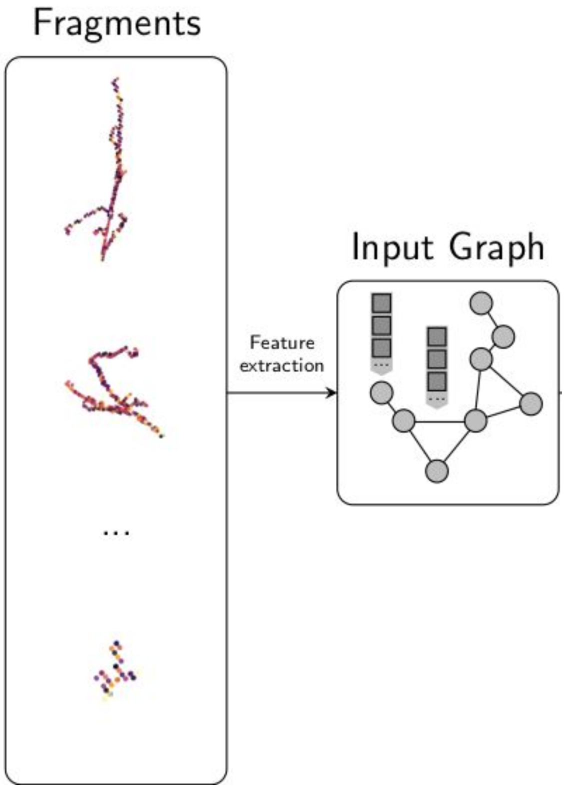


Restriction: 150 cm



Graph Representation

We now **care** about both **nodes** and **edges** in the graph

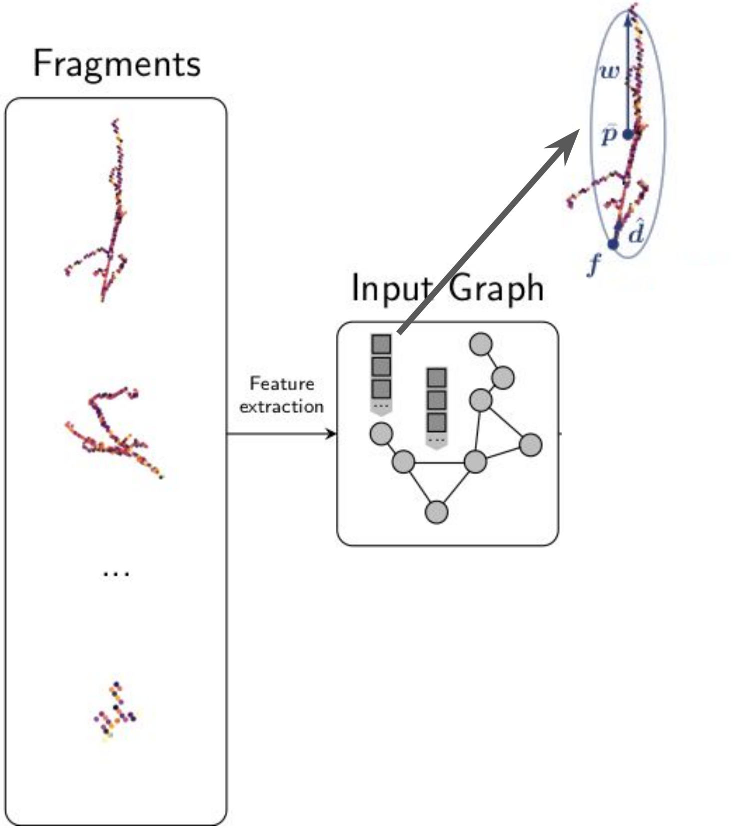


Graph Representation

We now **care** about both **nodes** and **edges** in the graph

Node features:

- Centroid
- Covariance matrix
- Start point/direction
- . . .

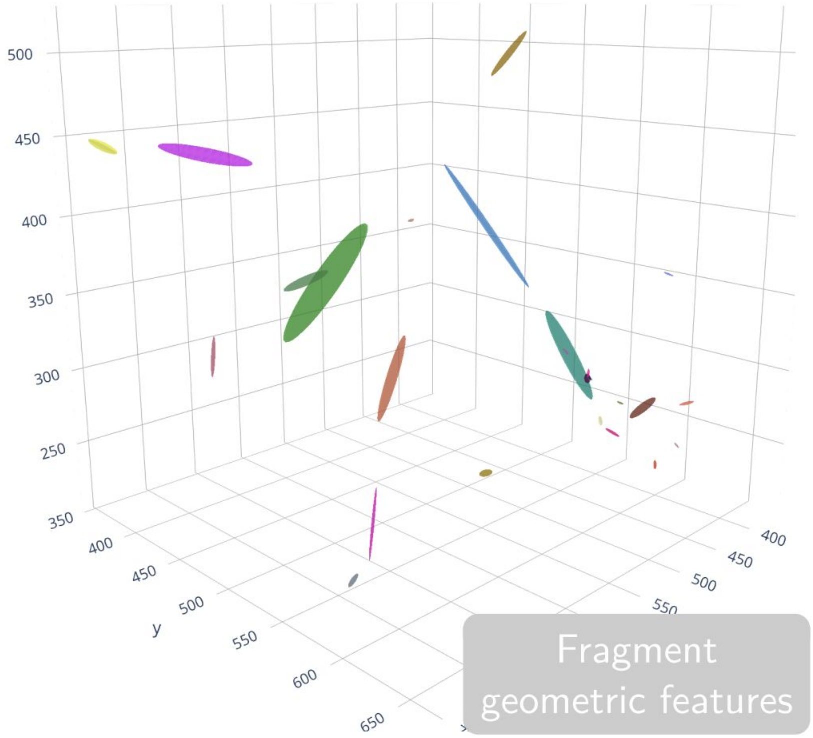


Graph Representation

We now **care** about both **nodes** and **edges** in the graph

Node features:

- Centroid
- Covariance matrix
- Start point/direction
- . . .



Graph Representation

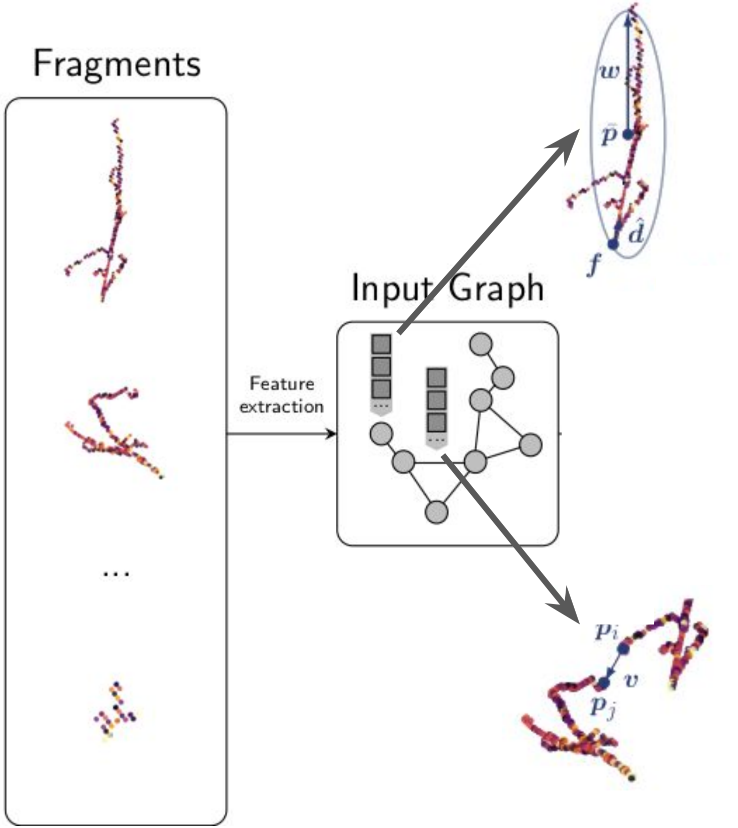
We now **care** about both **nodes** and **edges** in the graph

Node features:

- Centroid
- Covariance matrix
- Start point/direction
- . . .

Edge features:

- Displacement vector
- . . .



Graph Representation

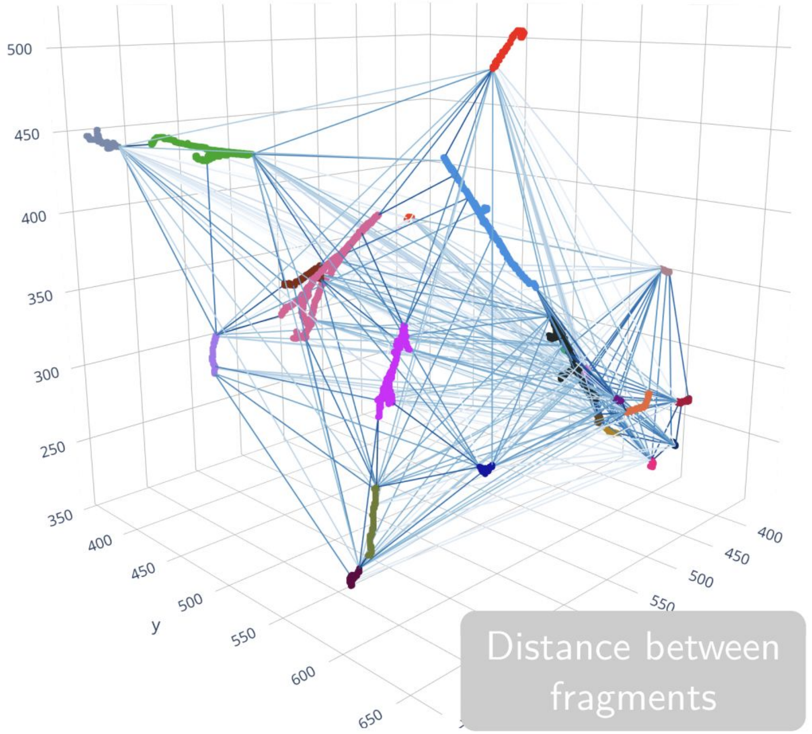
We now **care** about both **nodes** and **edges** in the graph

Node features:

- Centroid
- Covariance matrix
- Start point/direction
- . . .

Edge features:

- Displacement vector
- . . .



Message passing

Two **feature update steps**

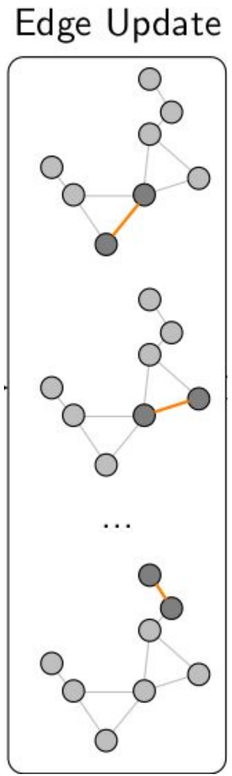
1. Edge update

$$\mathbf{e}'_{ij} = \phi_\Theta(\mathbf{x}_i,\, \mathbf{x}_j,\, \mathbf{e}_{ij})$$

Neural network

Edge features

Node features



Message passing

Two **feature update steps**

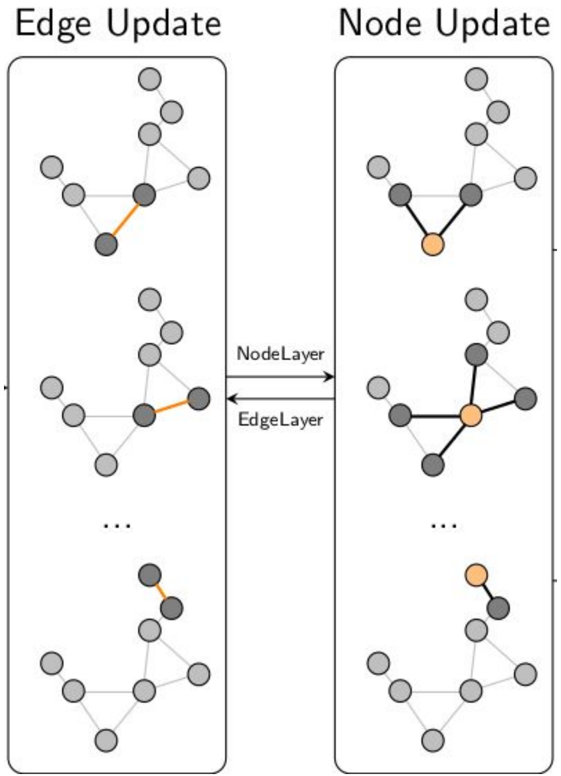
1. **Edge update**

$$\mathbf{e}'_{ij} = \phi_\Theta(\mathbf{x}_i,\, \mathbf{x}_j,\, \mathbf{e}_{ij})$$

2. **Node update**

$$\mathbf{m}_{ji} = \chi_\Theta(\mathbf{x}_j,\, \mathbf{e}_{ji})$$

Message of addressed from node j to node i



Message passing

Two **feature update steps**

1. **Edge update**

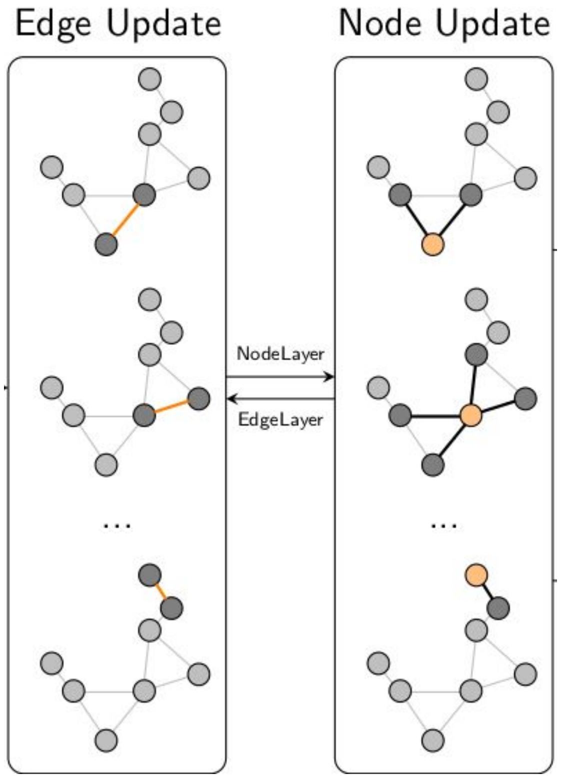
$$\mathbf{e}'_{ij} = \phi_\Theta(\mathbf{x}_i, \mathbf{x}_j, \mathbf{e}_{ij})$$

2. **Node update**

$$\mathbf{m}_{ji} = \chi_\Theta(\mathbf{x}_j, \mathbf{e}_{ji})$$

$$\mathbf{x}'_i = \psi_\Theta(\mathbf{x}_i, \square_{j\in\mathcal{N}(i)}\mathbf{m}_{ji})$$

Aggregator function: mean, max, sum, etc.



Message passing

Two **feature update steps**

1. **Edge update**

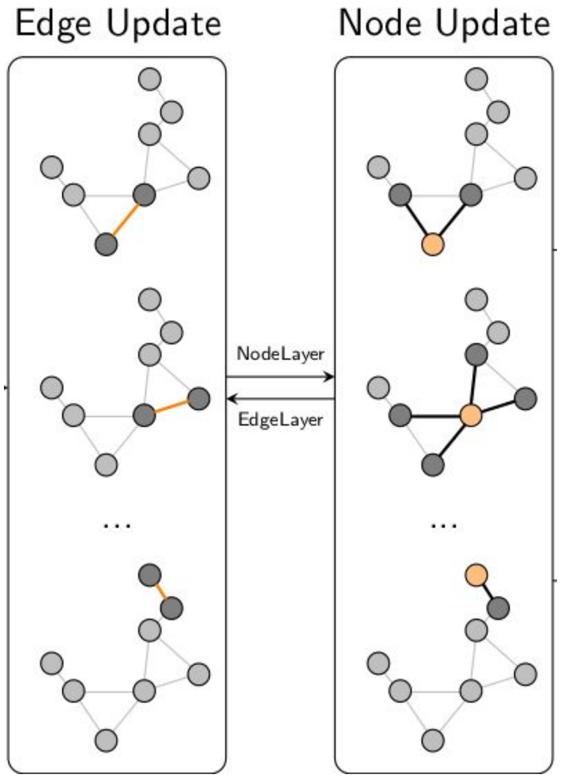
$$\mathbf{e}'_{ij} = \phi_\Theta(\mathbf{x}_i,\, \mathbf{x}_j,\, \mathbf{e}_{ij})$$

2. **Node update**

$$\mathbf{m}_{ji} = \chi_\Theta(\mathbf{x}_j,\, \mathbf{e}_{ji})$$

$$\mathbf{x}'_i = \psi_\Theta(\mathbf{x}_i,\, \square_{j\in\mathcal{N}(i)}\mathbf{m}_{ji})$$

Repeat **n** times (depth)



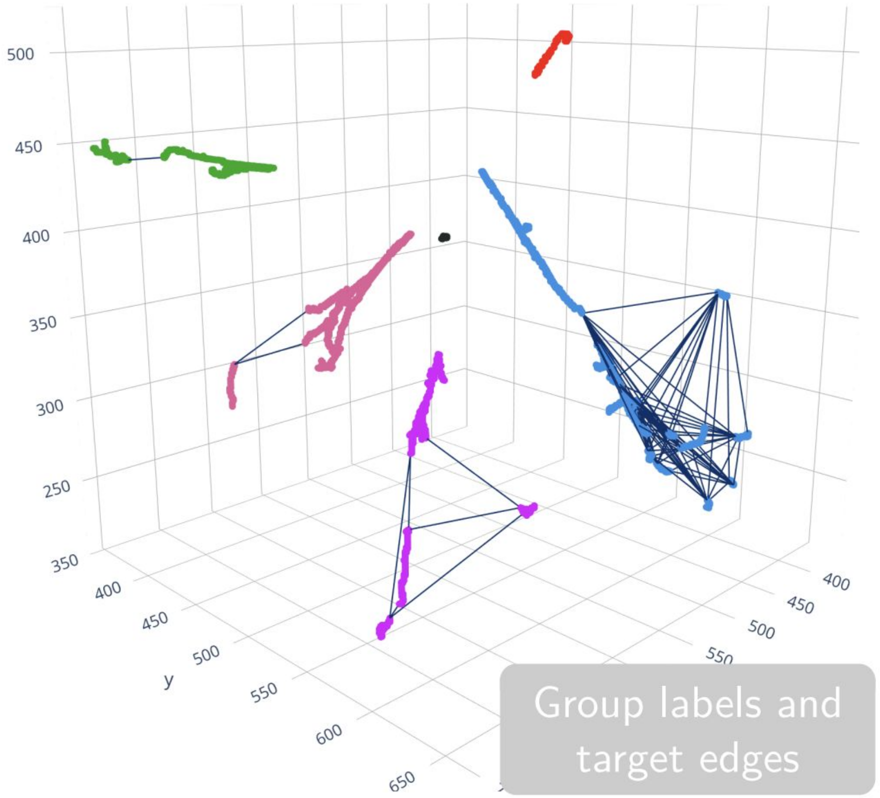
Graph Objective

Output: edge scores, s_{ij}

Goal: assign edge labels

$$a_{ij} = \delta_{g_i, g_j}$$

g_i the group of fragment i.



Graph Objective

Output: edge scores, s_{ij}

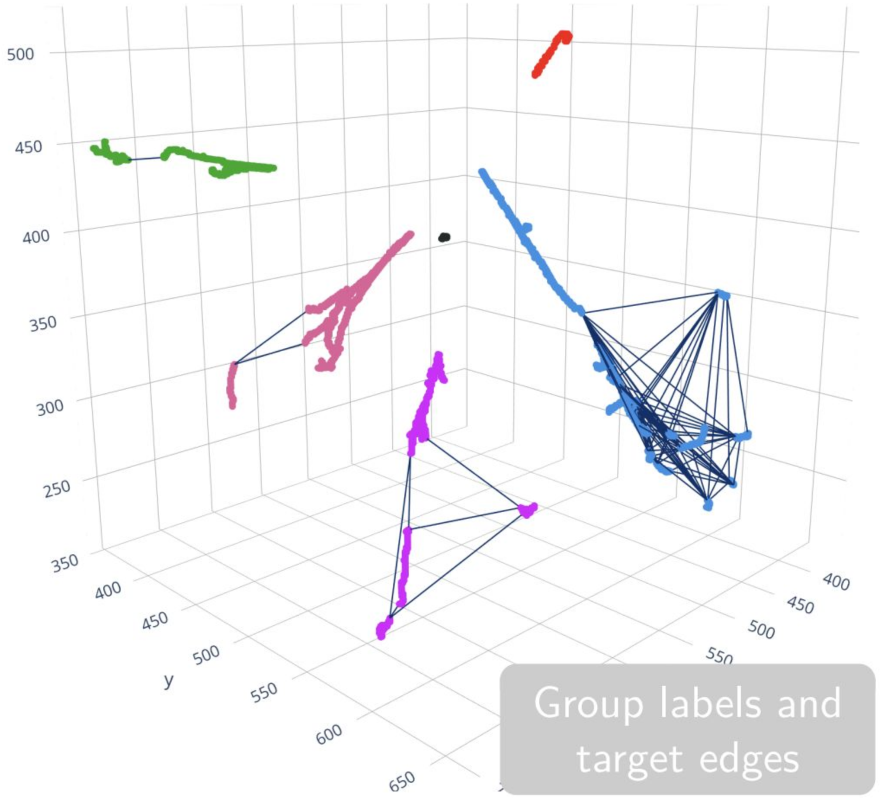
Goal: assign edge labels

$$a_{ij} = \delta_{g_i, g_j}$$

g_i the group of fragment i.

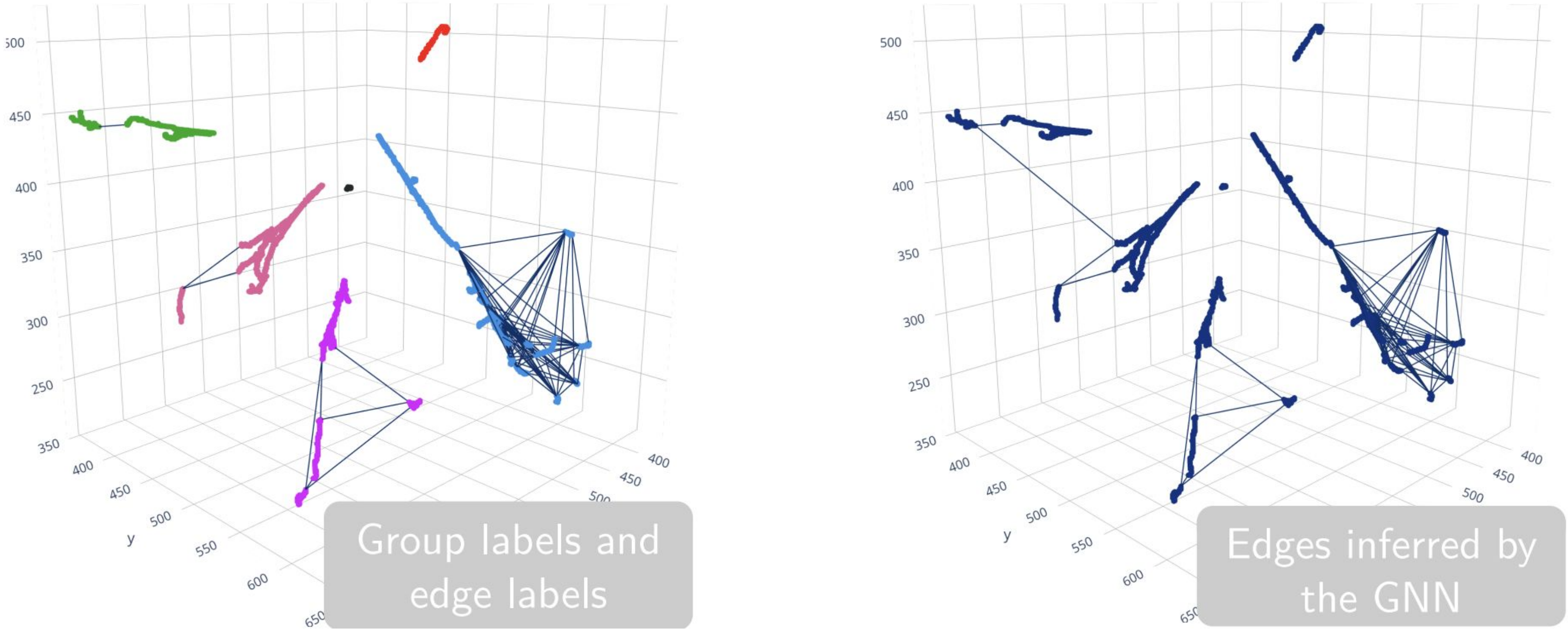
Loss: cross-entropy

$$\mathcal{L} = -\frac{1}{N_e} \sum_{(i,j)\in E} \left[a_{ij} \ln(s_{ij}) + (1 - a_{ij}) \ln(1 - s_{ij})\right]$$



Graph Construction

Find **connected components** and it's a done deal? **Not quite**…



Edge Selection

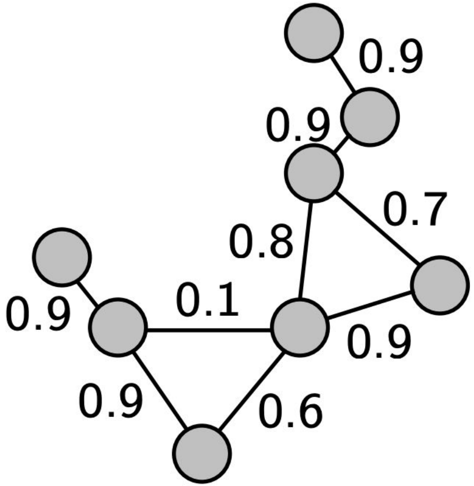
The GNN gives you a list of **edge scores**, not a partition

For the **best partition**, ĝ, we must select edges which minimizes the **partition CE loss**

Classification at the partition level!

$$\mathcal{L}_{\hat{g}} = -\frac{1}{N_e} \sum_{(i,j)\in E} \Big[\delta_{\hat{g}_i,\hat{g}_j} \ln(s_{ij}) + (1-\delta_{\hat{g}_i,\hat{g}_j}) \ln(1-s_{ij})\Big]$$

Edge scores



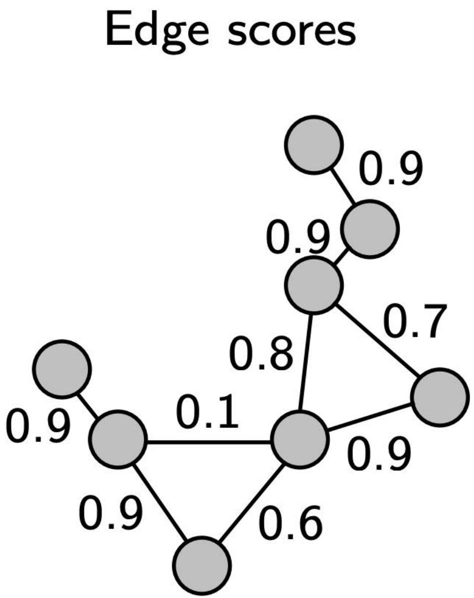
Edge Selection

Brute-force try all partitions?

Absolutely not...

Bell number: number of possible partition of a set of N objects

- B_8 ~ 4140 permutations
- B_{20} ~ 5 x 10^13 permutations
- . . .
- Could be hundreds of fragments

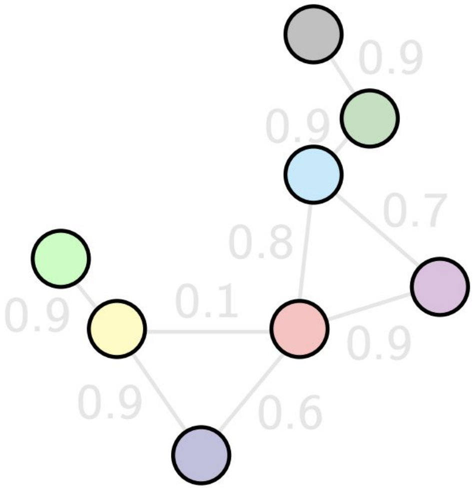


Edge Selection

Instead, **iterate:**

1. Compute partition **loss** for the empty graph

Empty graph

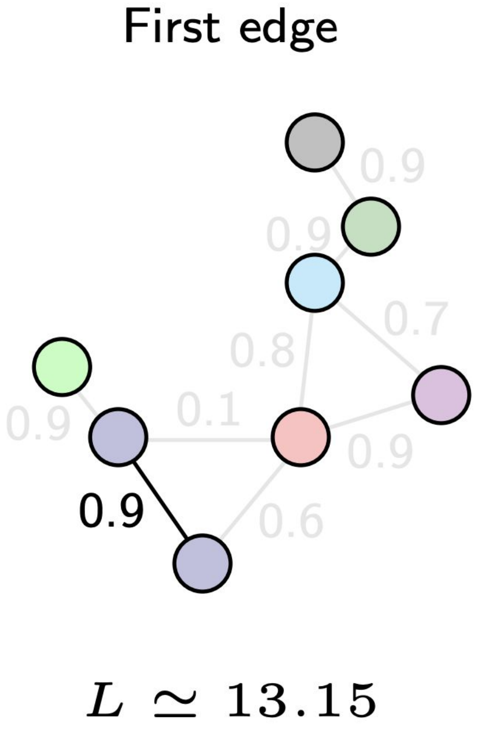


$$L \simeq 15.35$$

Edge Selection

Instead, **iterate:**

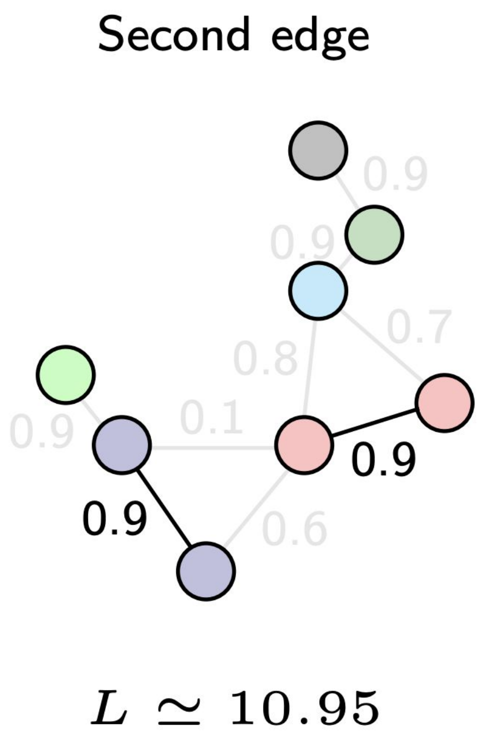
1. Compute partition **loss** for the empty graph
2. Add the **most likely edge**, compute loss again
3. If $L_{n+1} < L_n$, **update partition**



Edge Selection

Instead, **iterate:**

1. Compute partition **loss** for the empty graph
2. Add the **most likely edge**, compute loss again
3. If $L_{n+1} < L_n$, **update partition**
4. Repeat until the next best edge has $s_{ij} < 0.5$

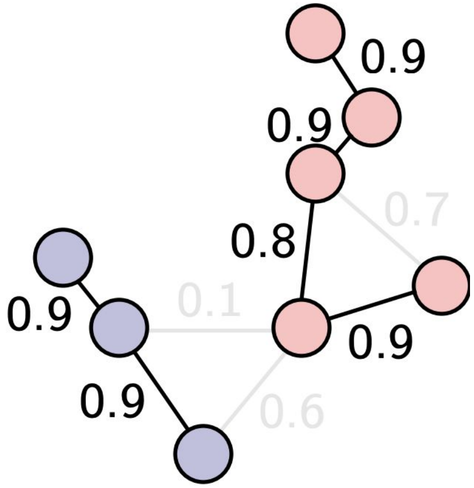


Edge Selection

Instead, **iterate:**

1. Compute partition **loss** for the empty graph
2. Add the **most likely edge**, compute loss again
3. If $L_{n+1} < L_n$, **update partition**
4. Repeat until the next best edge has $s_{ij} < 0.5$

Optimized partition



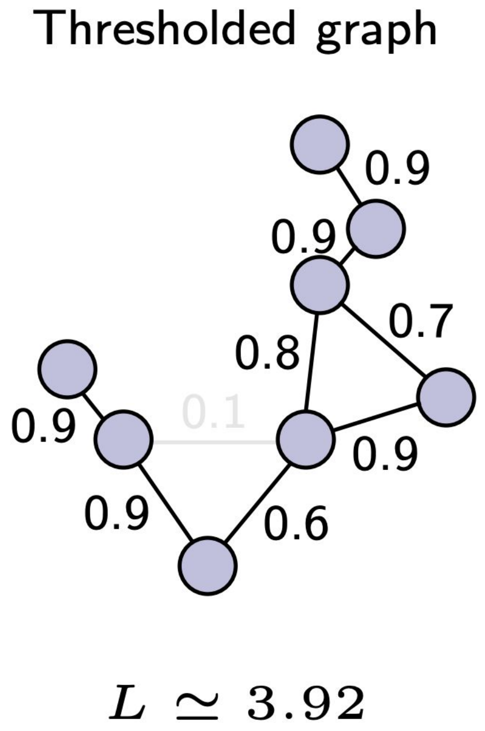
$L \simeq 2.13$

Edge Selection

Instead, **iterate:**

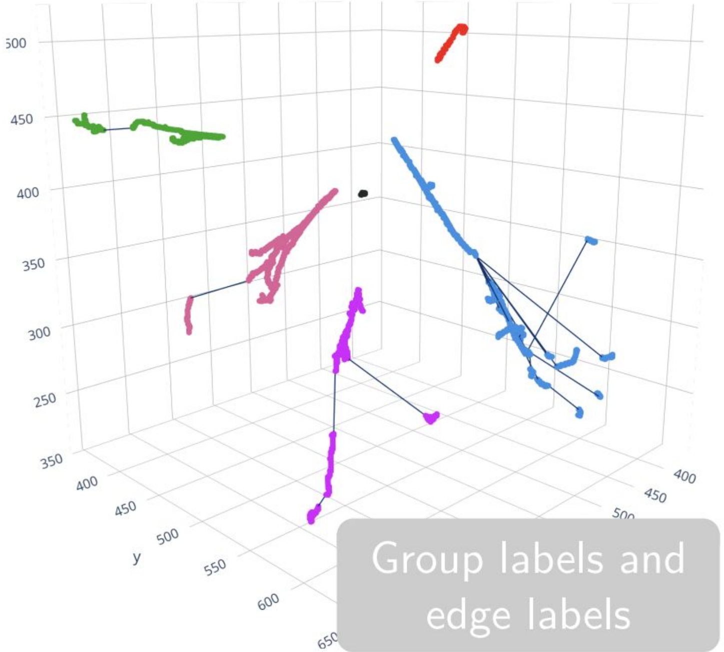
1. Compute partition **loss** for the empty graph
2. Add the **most likely edge**, compute loss again
3. If $L_{n+1} < L_n$, **update partition**
4. Repeat until the next best edge has $s_{ij} < 0.5$

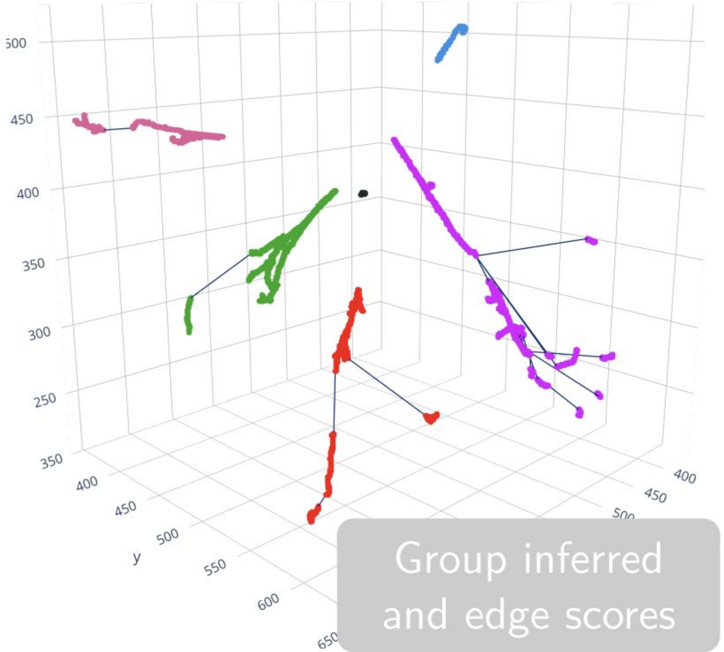
Better than edge thresholding!



Edge Selection

This automatically gets rid of spurious positive edges!





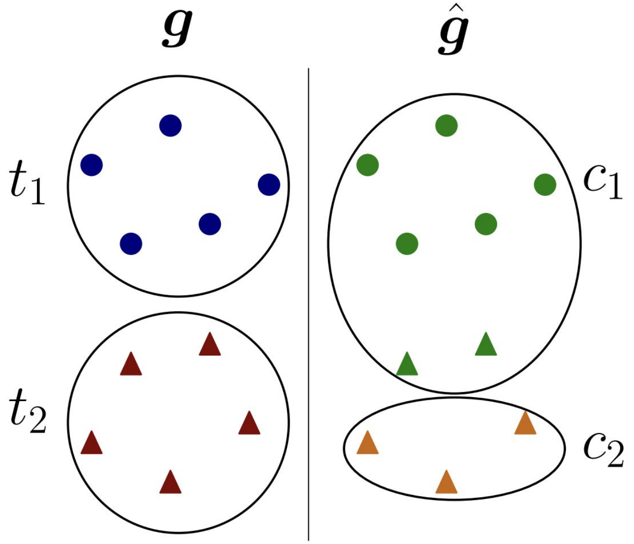
Clustering Metrics

Quantifying clustering accuracy is not trivial

- There is no single-voxel accuracy (cluster ID is not fixed)

Three metrics we use:

- **Efficiency**:
  - $\frac{1}{N_t}\sum_i^{N_t} \max_j \#(t_i \cap c_j)/\#t_i$



Eff = 0.5*(5/5 + 3/5) = 0.8

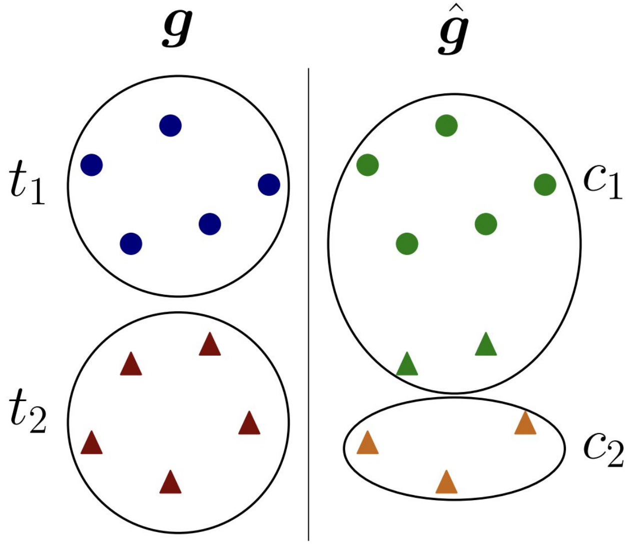
Clustering Metrics

Quantifying clustering accuracy is not trivial

- There is no single-voxel accuracy (cluster ID is not fixed)

Three metrics we use:

- **Efficiency**:
  - $\frac{1}{N_t}\sum_i^{N_t} \max_j \#(t_i \cap c_j)/\#t_i$
- **Purity**:
  - $\frac{1}{N_r}\sum_j^{N_r} \max_i \#(t_i \cap c_j)/\#c_j$



Pur. = 0.5*(5/7 + 3/3) ~ 0.86

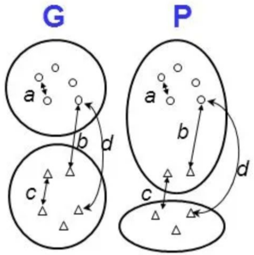
Clustering Metrics

Quantifying clustering accuracy is not trivial

- There is no single-voxel accuracy (cluster ID is not fixed)

Three metrics we use:

- **Efficiency**:
  - $\frac{1}{N_t}\sum_i^{N_t} \max_j \#(t_i \cap c_j)/\#t_i$
- **Purity**:
  - $\frac{1}{N_r}\sum_j^{N_r} \max_i \#(t_i \cap c_j)/\#c_j$
- **Adjusted Rand Index (ARI)**

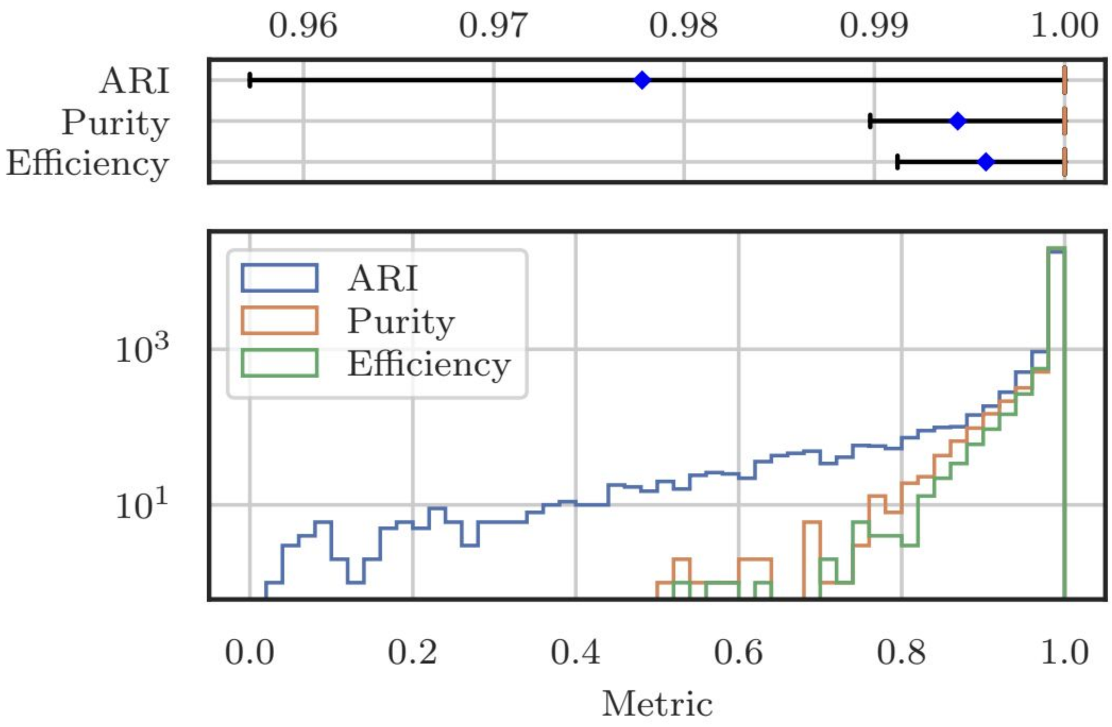


Agreement: a, d
Disagreement: b, c

$$RI(P,G) = \frac{a+d}{a+b+c+d}$$

$$ARI = \frac{RI - E(RI)}{1 - E(RI)}$$

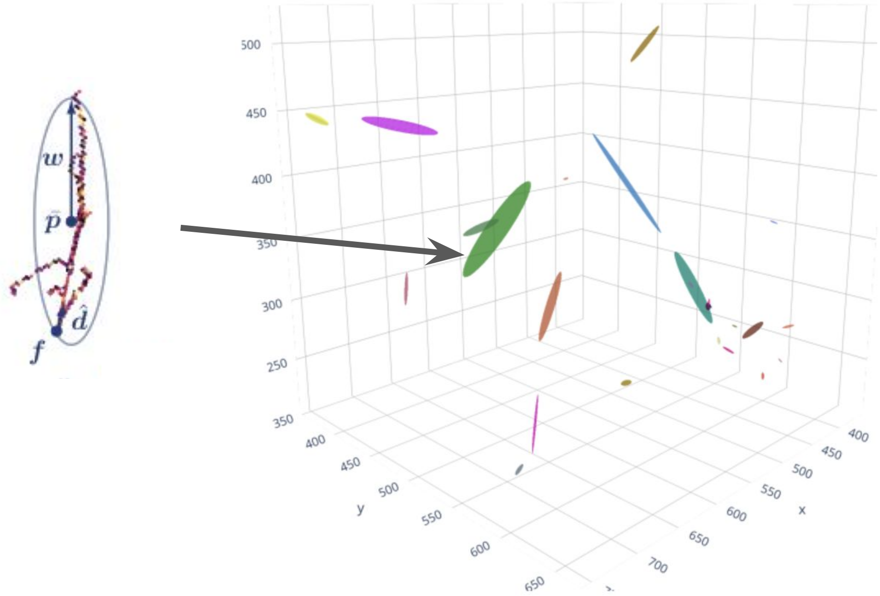
Shower Clustering Performance



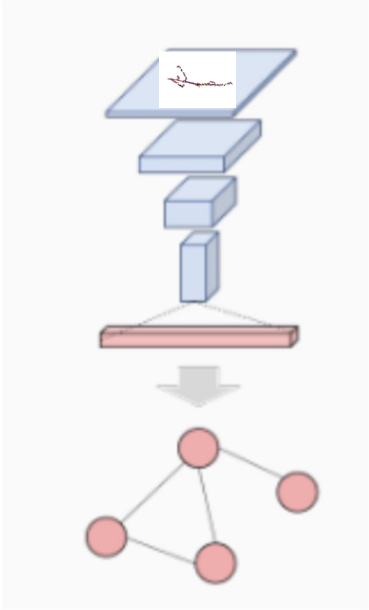
Optimization

Node encoding: hand-engineered or automatic?

Geometric

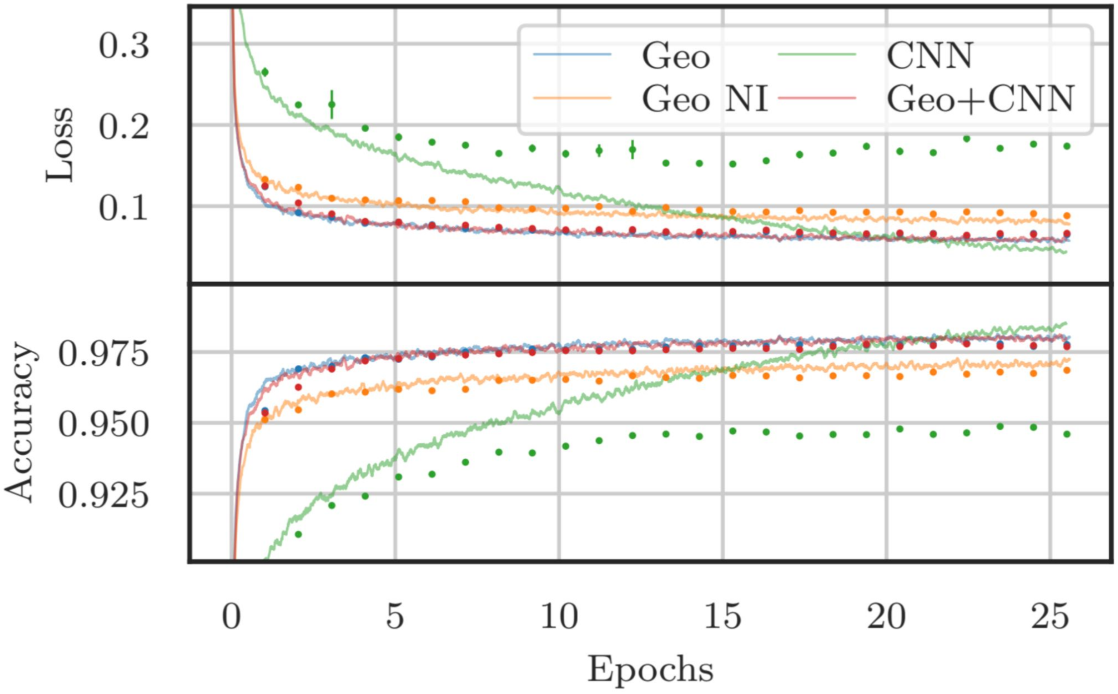


CNN



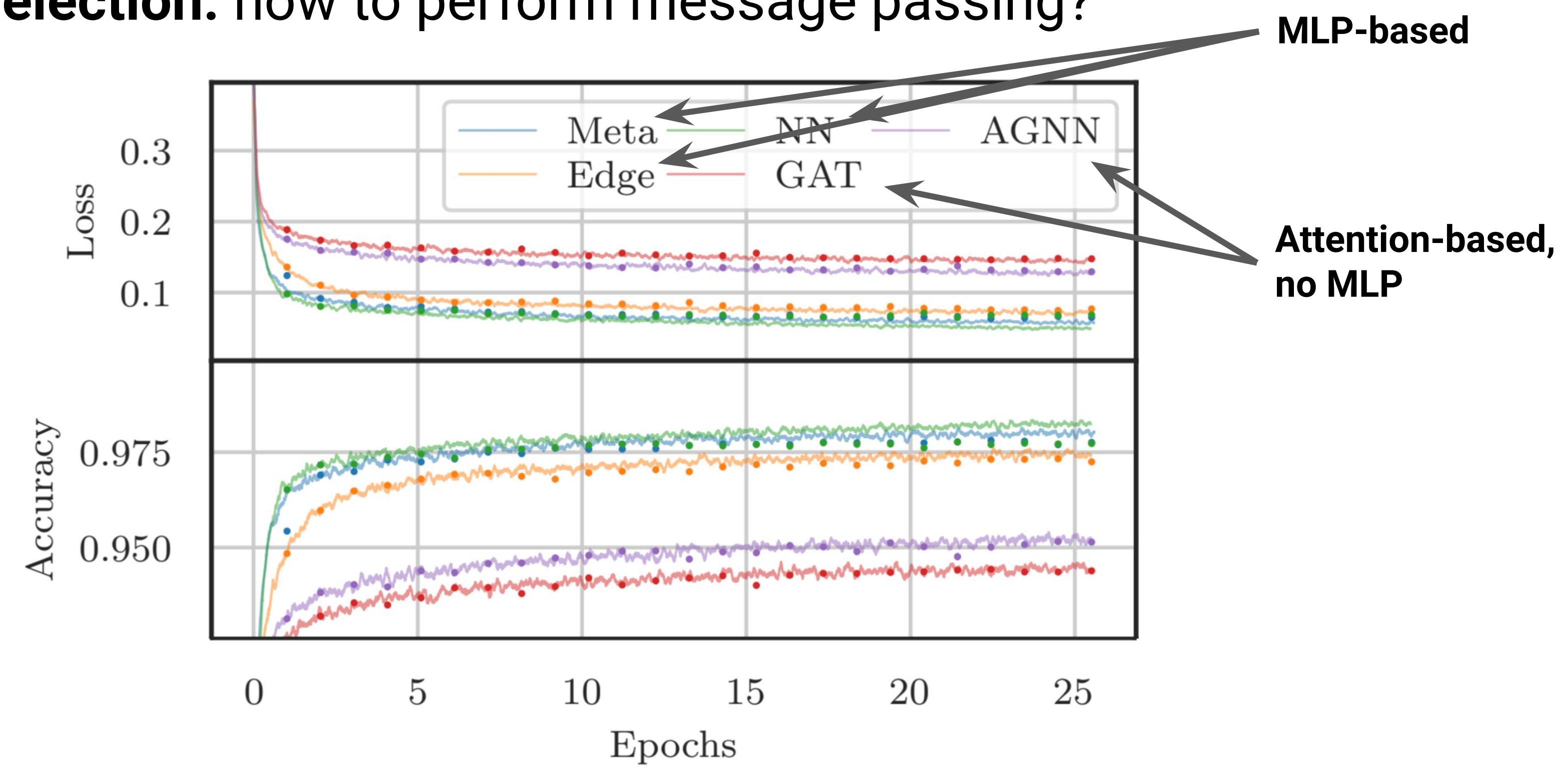
Optimization

Node encoding: automatic or hand-engineered?



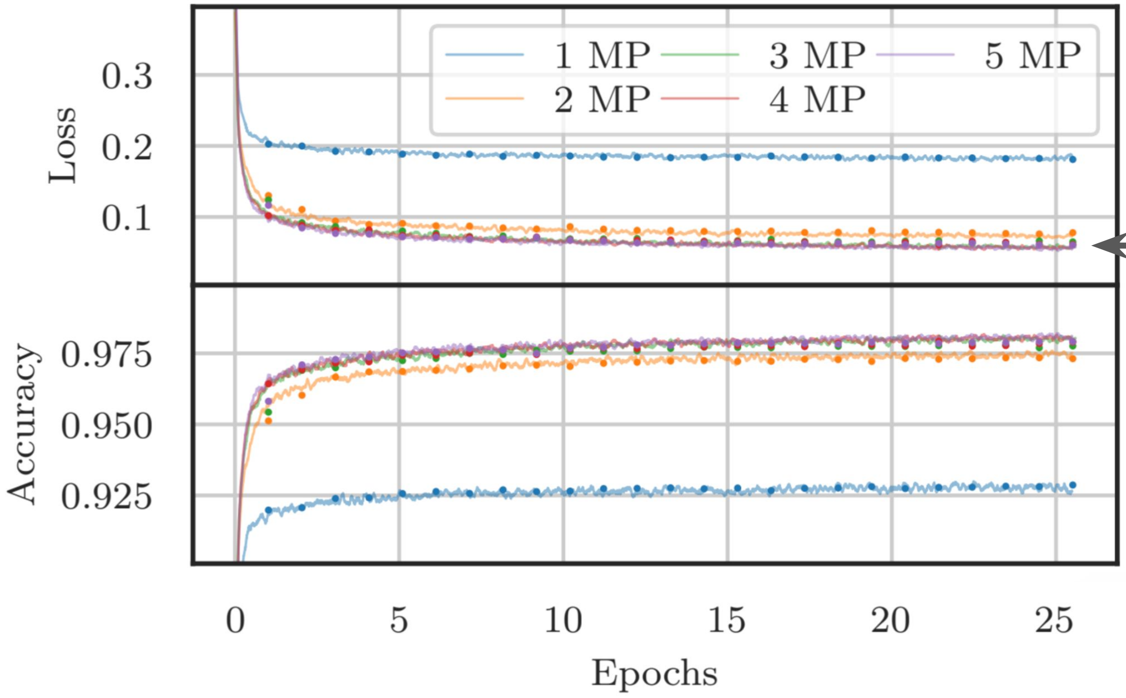
Optimization

Model selection: how to perform message passing?



Optimization

Model selection: how many layers do we need?

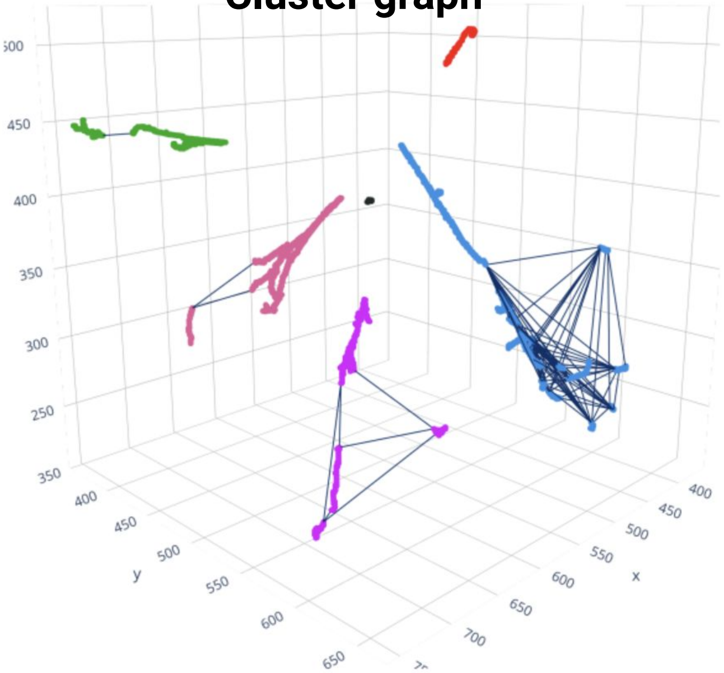


No gain beyond 3 layers

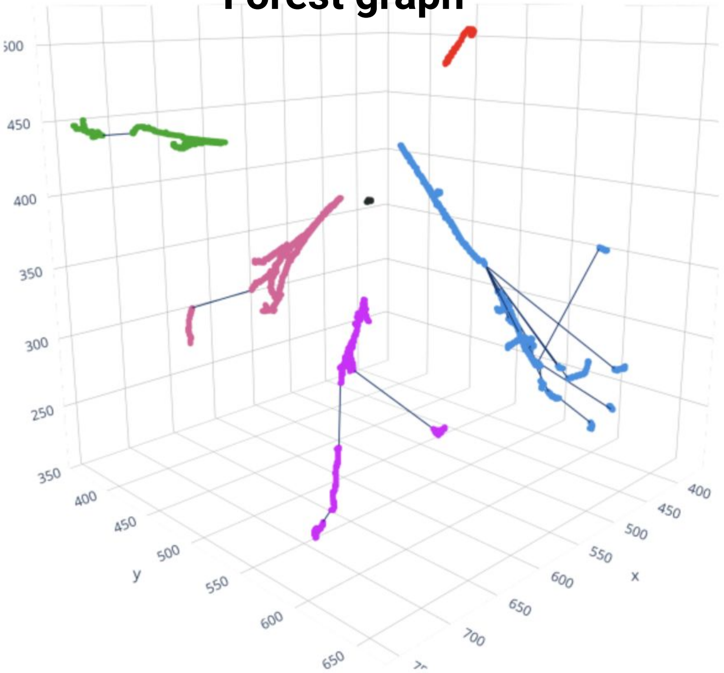
Optimization

Target selection: What are we trying to predict?

Cluster graph

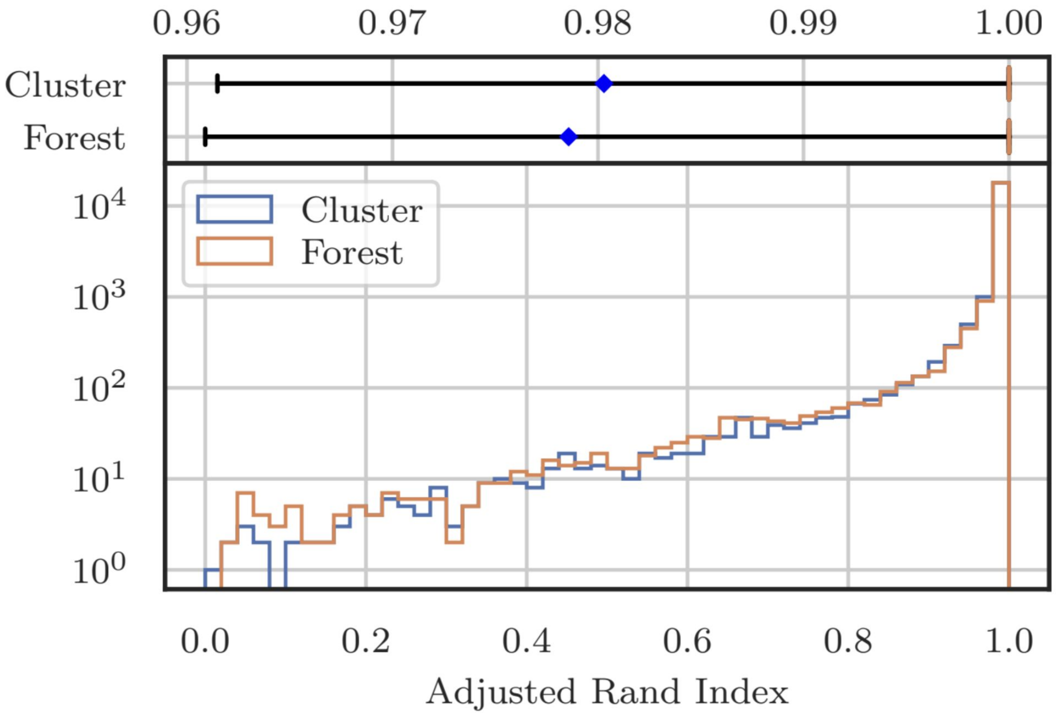


Forest graph



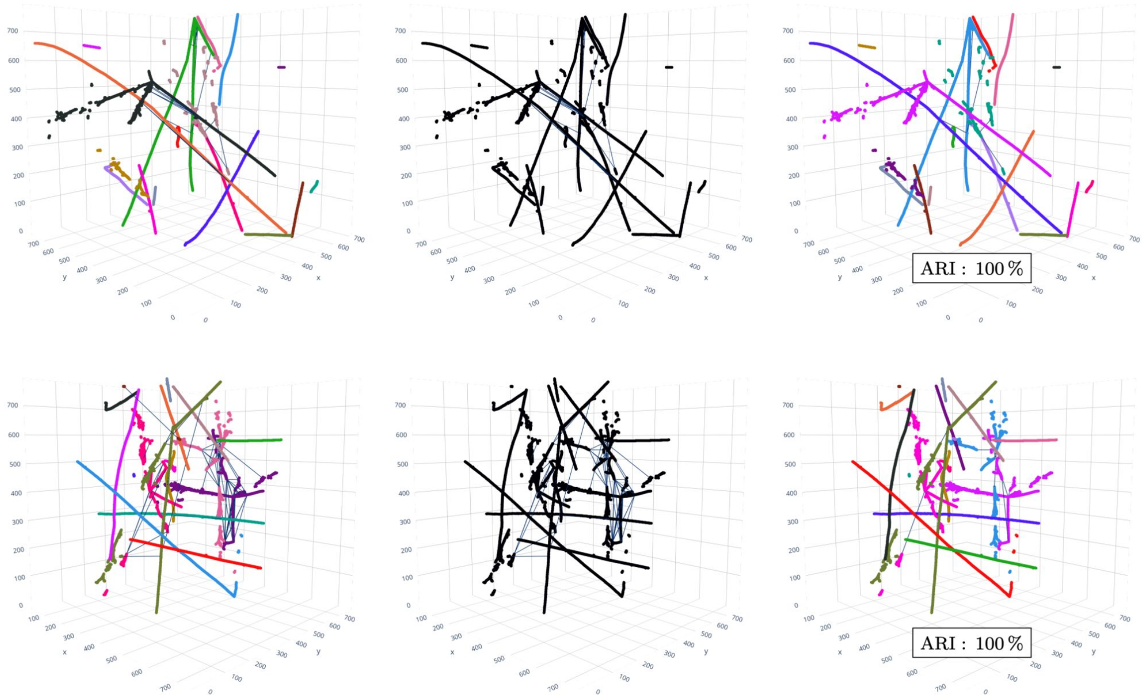
Optimization

Target selection: What are we trying to predict?

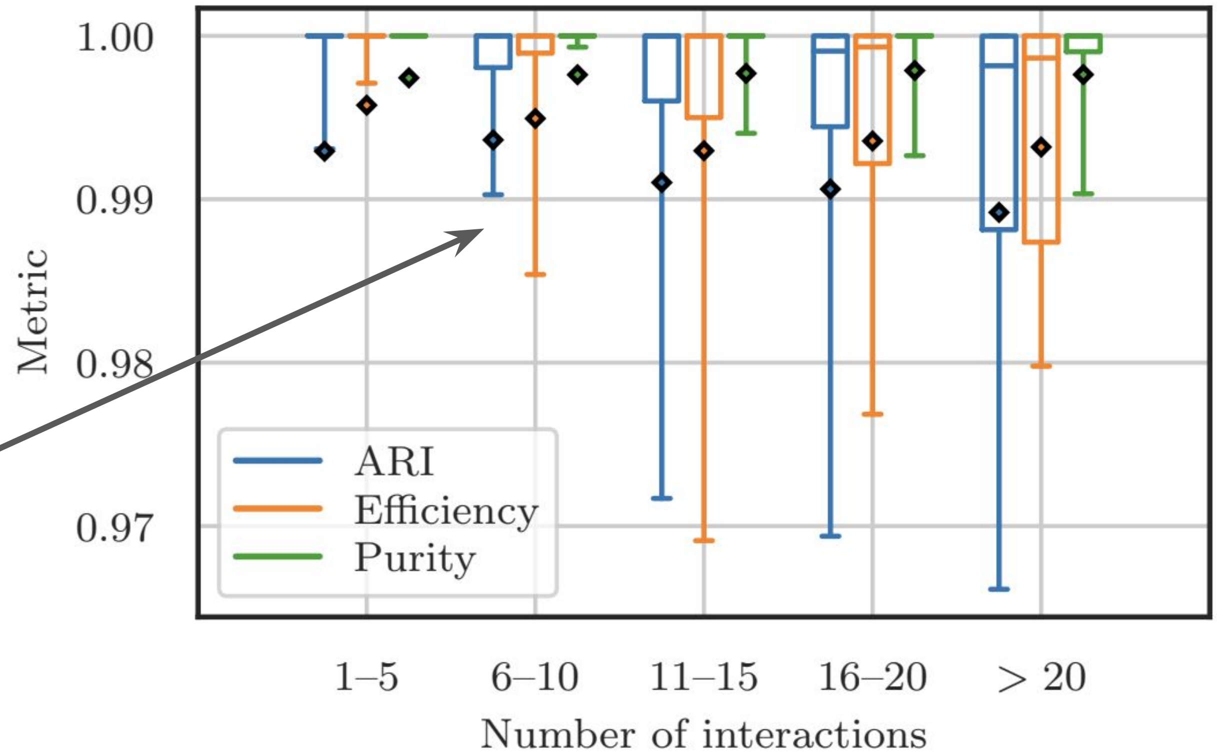


Multi-Purpose Algorithm

Can be **repurposed** for other aggregation tasks, e.g. **interactions**



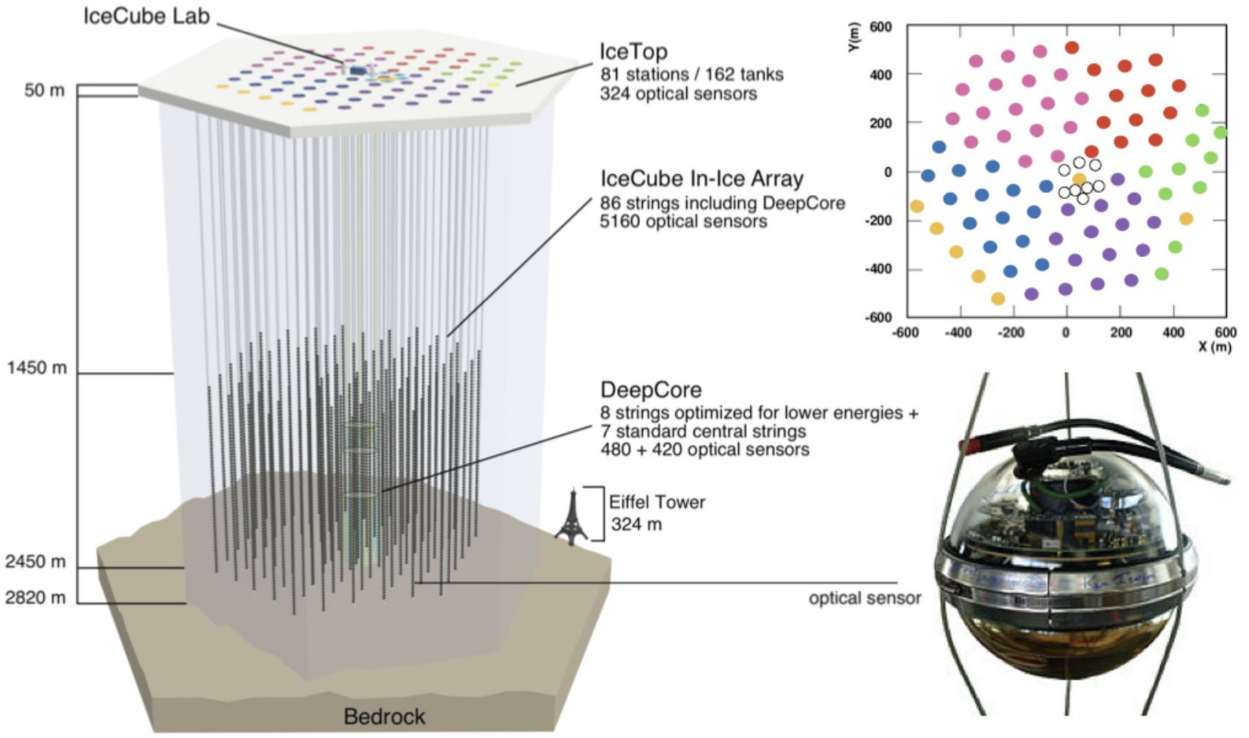
Interaction Clustering Performance



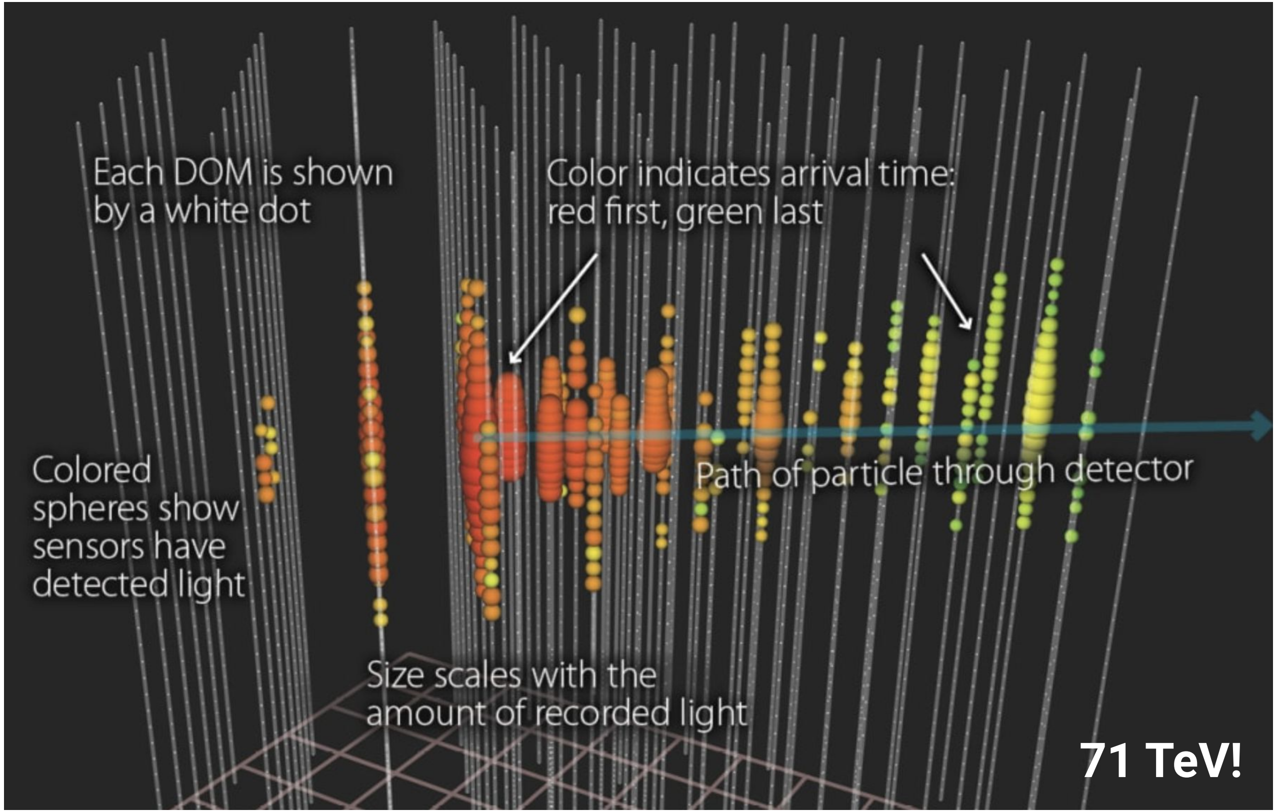
DUNE-ND pile-up level!

Whole Event Classification in IceCube

IceCube Detector

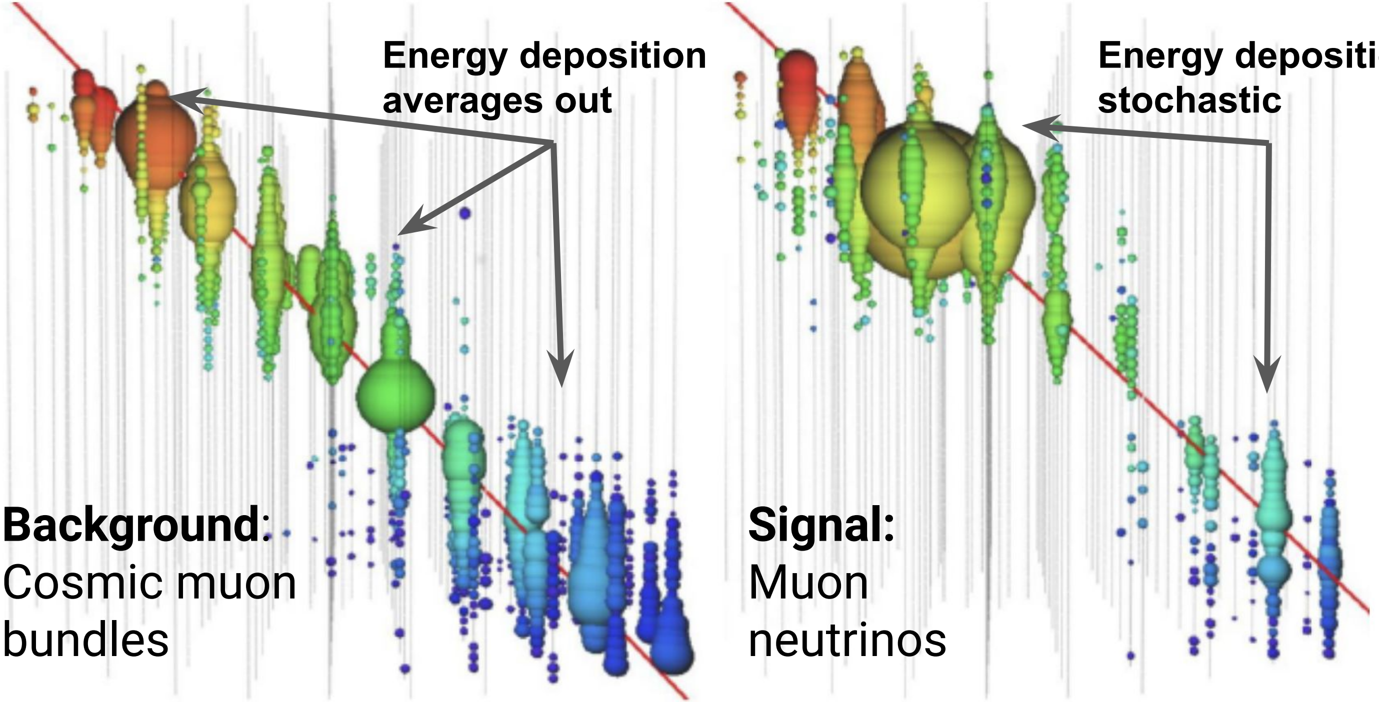


IceCube Events



Signal vs Background Separation

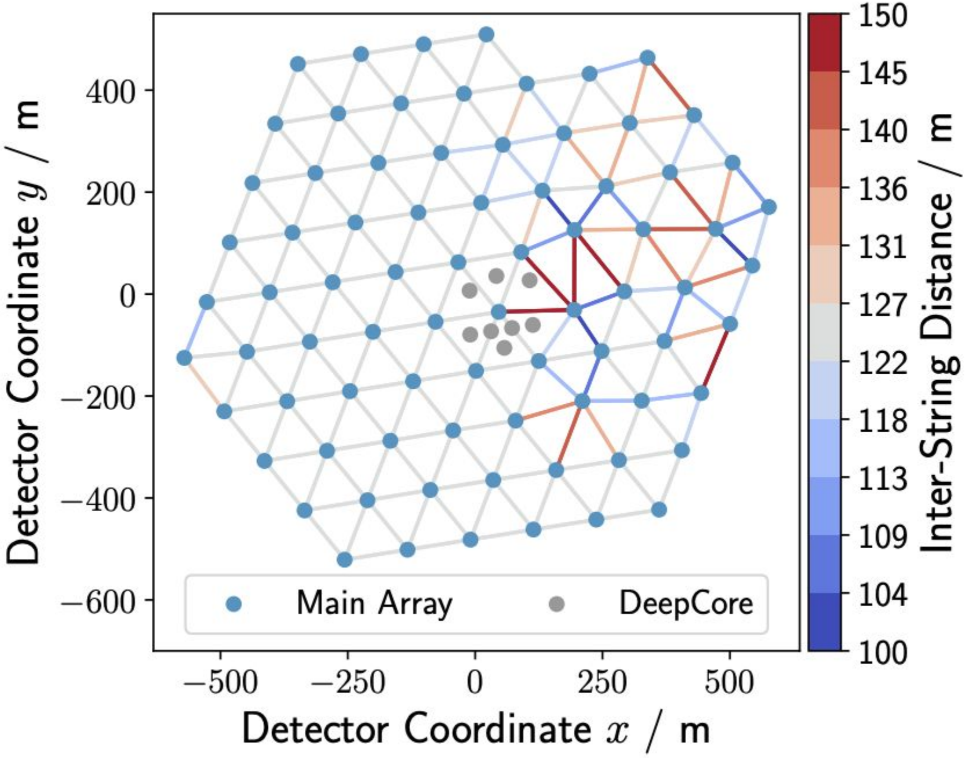
0.6 TeV -> 100 TeV



Data Representation

CNNs not the best choice for this data:

- **Hexagonal** and irregular layout
- DOM **pitch** different in x, y and z
- **DeepCore** strings

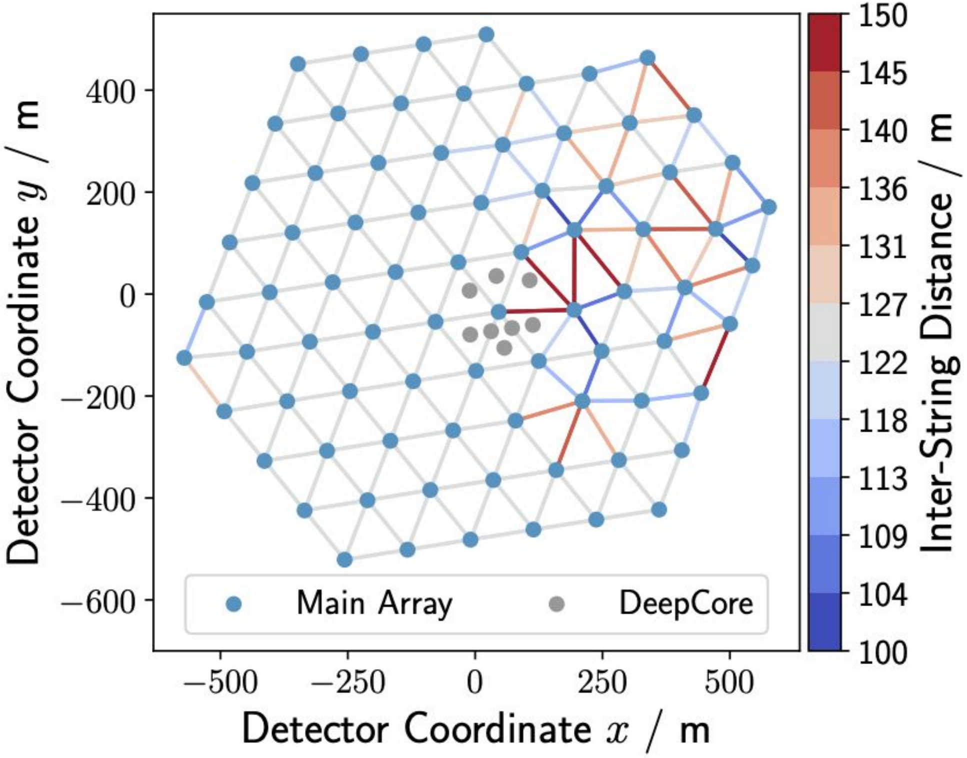


Data Representation

CNNs not the best choice for this data:

- **Hexagonal** and irregular layout
- DOM **pitch** different in x, y and z
- **DeepCore** strings

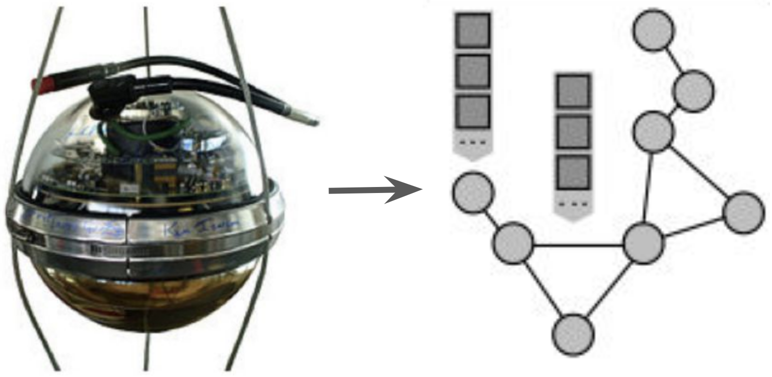
Graphs can accommodate all this!



Graph Representation

Node features:

- Position (x, y, z)
- Total PE (first hit, all hits)
- First time over threshold



Graph Representation

Node features:

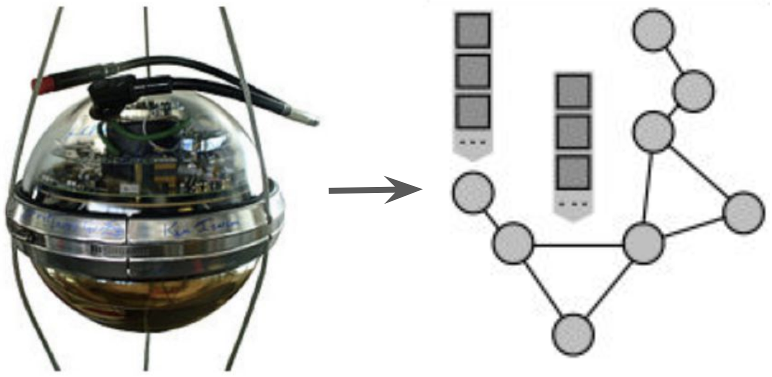
- Position (x, y, z)
- Total PE (first hit, all hits)
- First time over threshold

Edge set:

- Complete, weighted graph

$$d_{ij} = \exp(-||\mathbf{x}_j - \mathbf{x}_i||^2/\sigma^2)$$

Weight prop. to distance



Graph Representation

Node features:

- Position (x, y, z)
- Total PE (first hit, all hits)
- First time over threshold

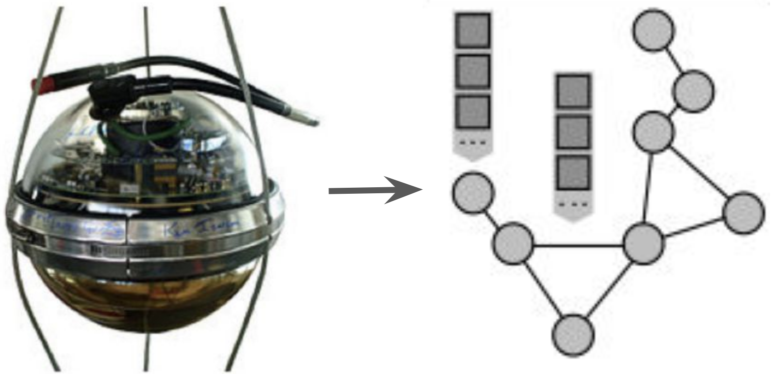
Edge set:

- Complete, weighted graph

$$d_{ij} = \exp(-||\mathbf{x}_j - \mathbf{x}_i||^2/\sigma^2)$$

$$a_{ij} = \frac{\exp(-d_{ij})}{\sum_k \exp(-d_{ik})}$$

Adjacency matrix element (normalized in [0, 1])



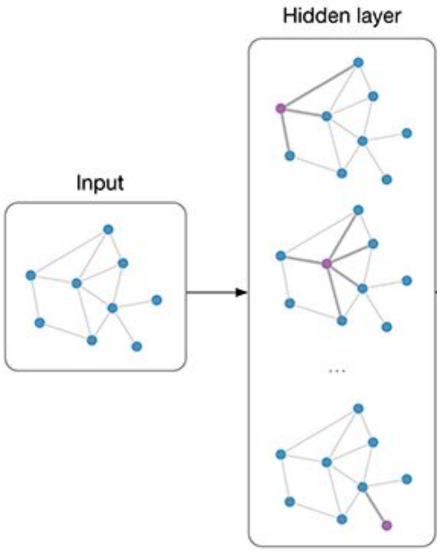
Feature extraction

Graph convolutions:

Learnable parameters

$$\mathbf{GConv}(\mathbf{X}^{(t)}) = [\mathbf{A}\mathbf{X}^{(t)}, \mathbf{X}^{(t)}]\mathbf{W}^{(t)} + \mathbf{b}^{(t)}$$

Weighted adjacency matrix



Feature extraction

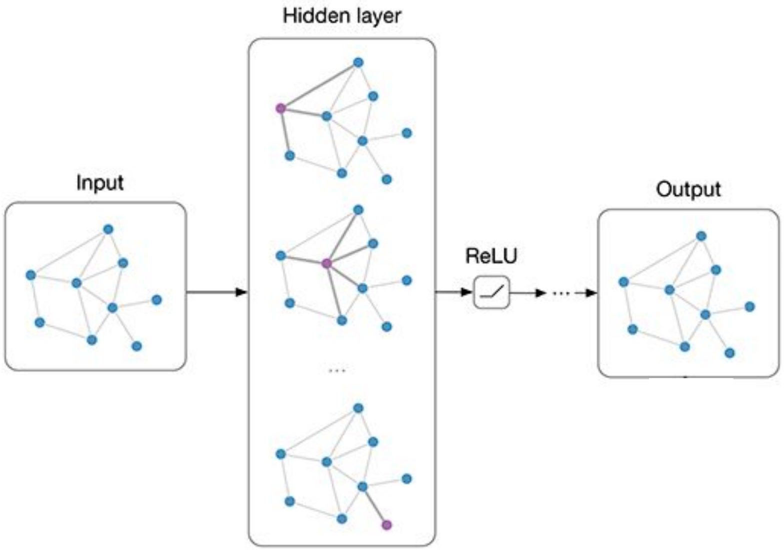
N times

Graph convolutions:

$$\mathbf{GConv}(\mathbf{X}^{(t)}) = [\mathbf{A}\mathbf{X}^{(t)}, \mathbf{X}^{(t)}]\mathbf{W}^{(t)} + \mathbf{b}^{(t)}$$

Activation:

$$\begin{aligned}\mathbf{X}^{(t+1)} = [&\mathbf{ReLU}(\mathbf{GConv}(\mathbf{X}^{(t)})),\\ &\mathbf{GConv}(\mathbf{X}^{(t)})]\end{aligned}$$



Feature extraction

Graph convolutions:

$$\mathbf{GConv}(\mathbf{X}^{(t)}) = [\mathbf{A}\mathbf{X}^{(t)}, \mathbf{X}^{(t)}]\mathbf{W}^{(t)} + \mathbf{b}^{(t)}$$

Activation:

$$\mathbf{X}^{(t+1)} = [\mathbf{ReLU}(\mathbf{GConv}(\mathbf{X}^{(t)})), \mathbf{GConv}(\mathbf{X}^{(t)})]$$

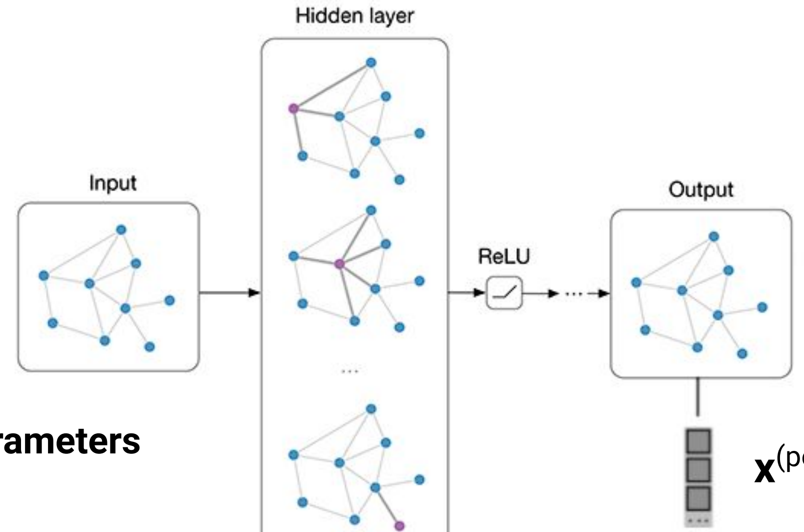
Pooling:

$$\mathbf{x}^{(\mathrm{pool})} = \sum_{i=1}^{n} \mathbf{X}_i^{(T)}$$

Learnable parameters

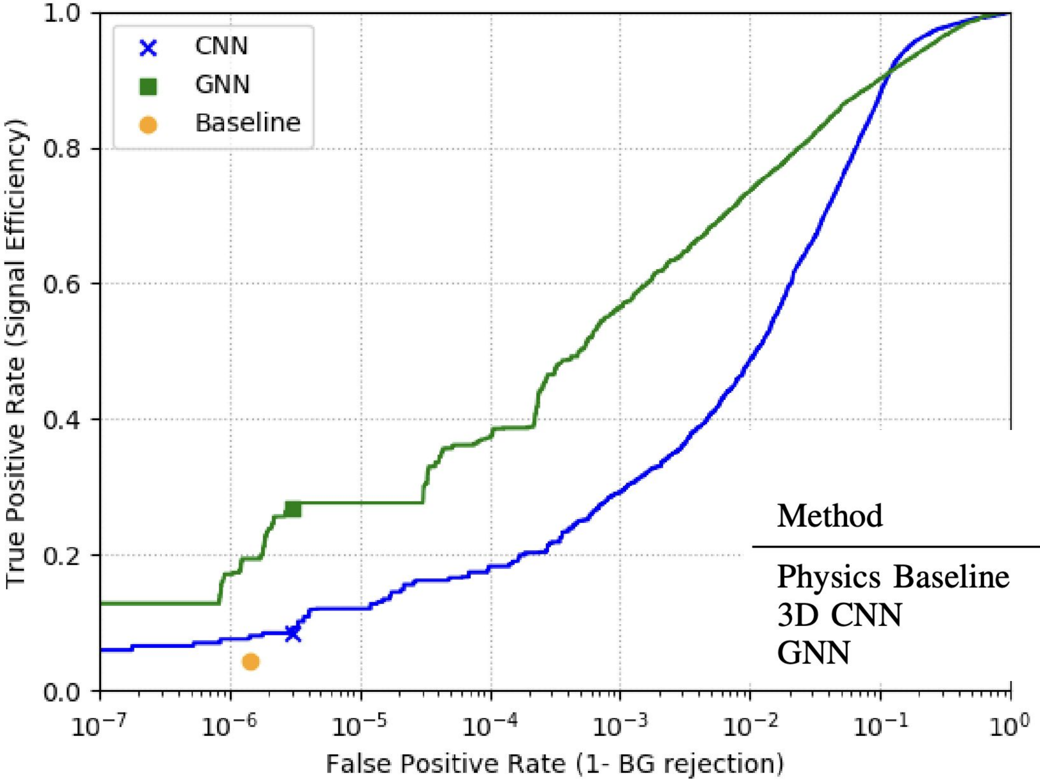
$$\hat{y} = \mathrm{Sigmoid}(\mathbf{x}^{\mathrm{pool}} \cdot \mathbf{w}^{\mathrm{pool}} + b^{\mathrm{pool}})$$

Graph-wide score in [0,1]



$\mathbf{x}^{(\mathrm{pool})}$

Background Rejection Performance



S/B: ~10^-6

Baseline: cut based

CNN: ResNet-18

GNN: This work

	# events per year		
Method	Signal	Background	Signal:Noise
Physics Baseline	0.922	0.934	0.987
3D CNN	1.815	1.937	0.937
GNN	**5.772**	1.937	**2.980**

Conclusions

Graphs can represent **anything**, way beyond rasterized images

- Endless **applications** in HEP and the **Intensity Frontier**
- We've barely scratched the surface! **Exciting times…**