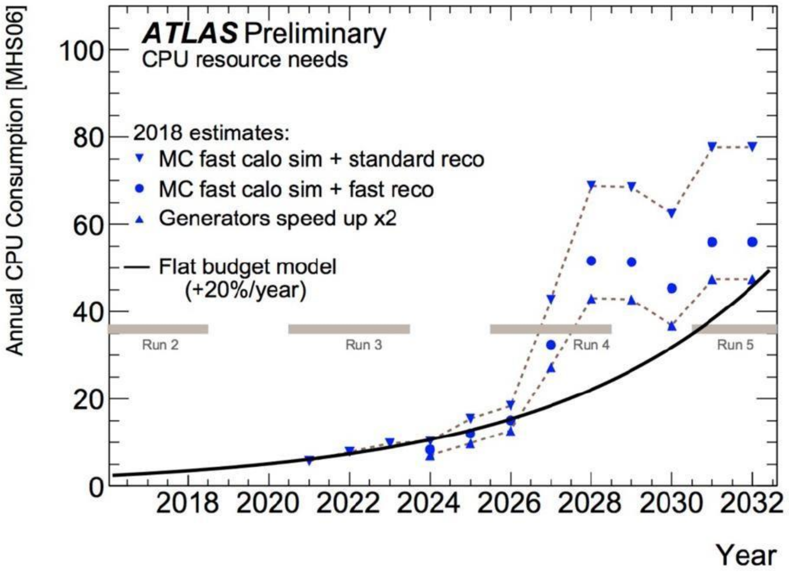
TRACK RECONSTRUCTION

with Metric Learning & GNNs

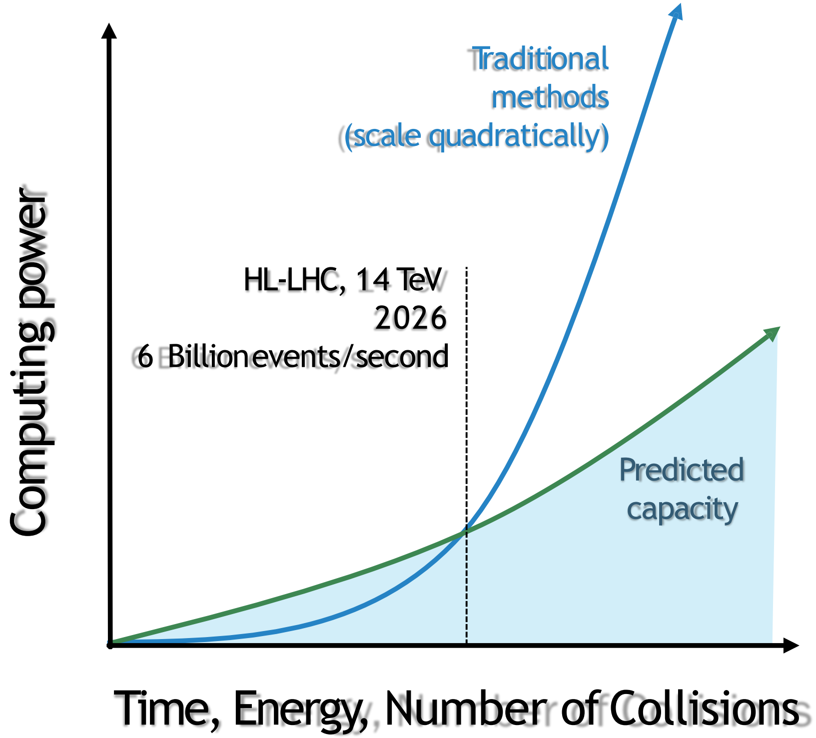
Leilan Zhang
zhangleilan@gmail.com

WHY MACHINE LEARNING FOR TRACKING?

High-luminosity scaling problem, means we need *something* to compliment traditional tracking algorithms, but why graphs?



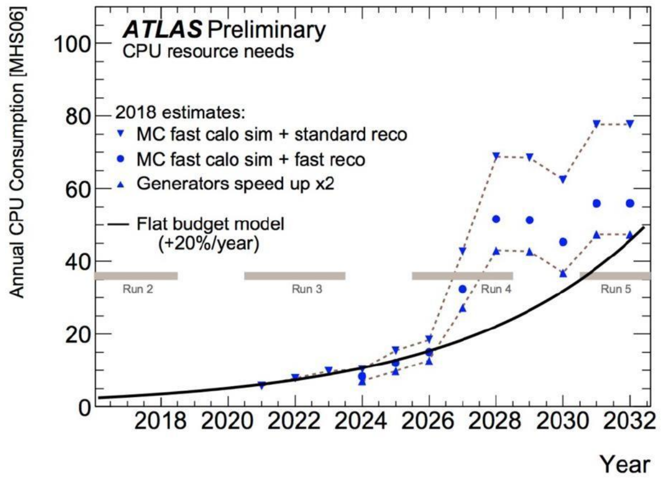
In other words...

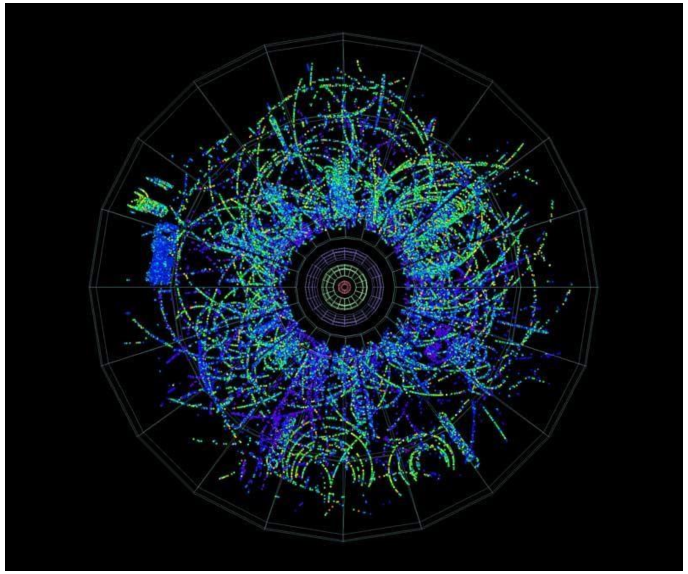


WHY GRAPHS SPECIFICALLY?

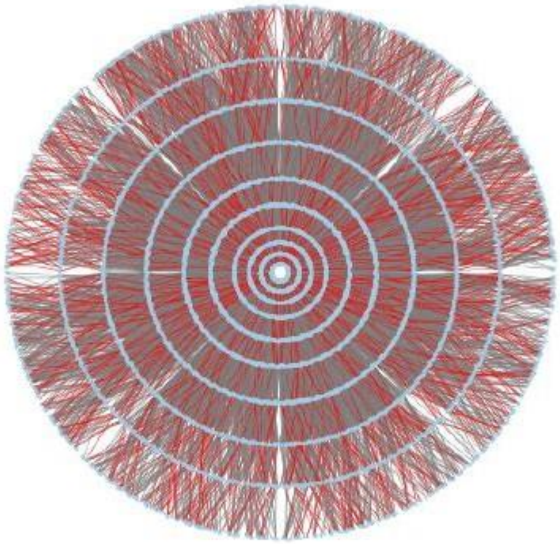
High-luminosity scaling problem, means we need *something* to compliment traditional tracking algorithms,
but why graphs?

Graphs can capture inherent sparsity of much physics data

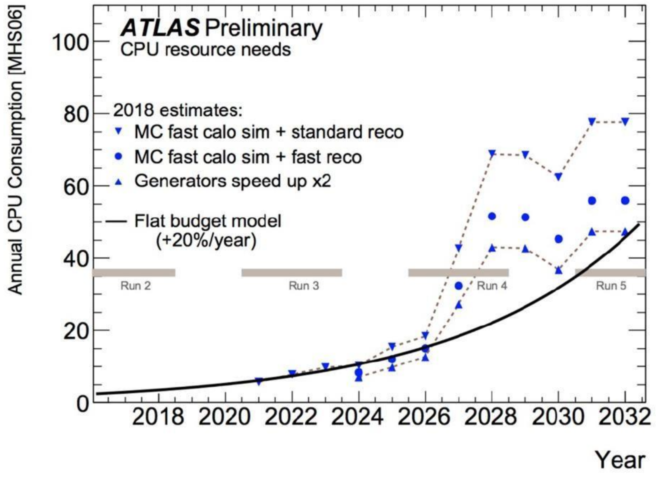




Hits to graphs



WHY GRAPHS?



High-luminosity scaling problem, means we need *something* to compliment traditional tracking algorithms, but why graphs?

Graphs can capture inherent sparsity of much physics data

Graphs can capture the manifold and relational structure of much physics data

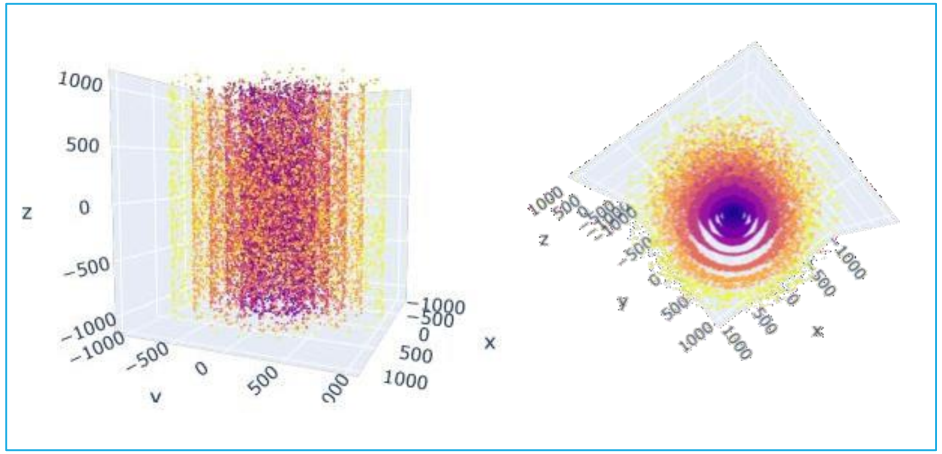
Conversion to and from graphs can allow manipulation of dimensionality

Graph Neural Networks are booming (i.e. wouldn't be talking about graphs if there weren't a wealth of classic algorithms and NN models for graph data)

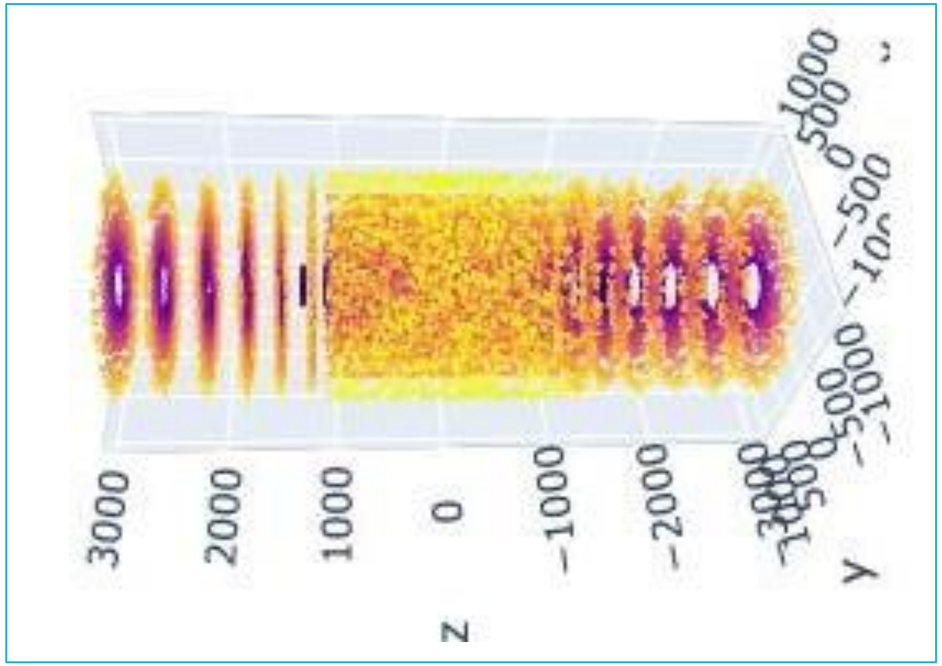
Industry research and investment means good outlook for software and hardware optimised for graphs

THE PHYSICAL PROBLEM

- “TrackML Kaggle Competition” dataset
- Generated by HL-LHC-like tracking (ACTS) **simulation**
- 9000 events to train on
- Each event has up to 100,000 layer hits from around 10,000 particles
- Layers can be hit multiple times by same particle (“duplicates”)
- Non-particle hits present (“noise”)



Barrel



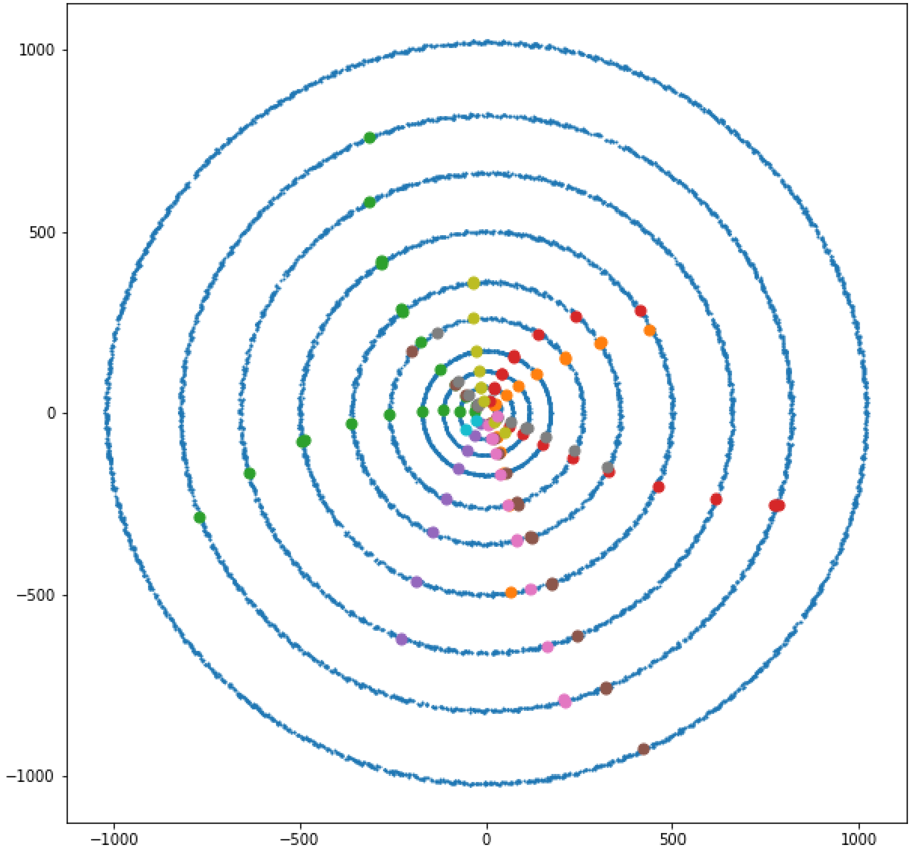
Full Detector

THE PHYSICAL PROBLEM

- Need to construct hit data into graph data, i.e. nodes and edges
- Can use geometric heuristics (have used in past: ~45% efficiency, 5% purity)
- To improve performance, use learned embedding construction
- Ideal final result is a “TrackML score”

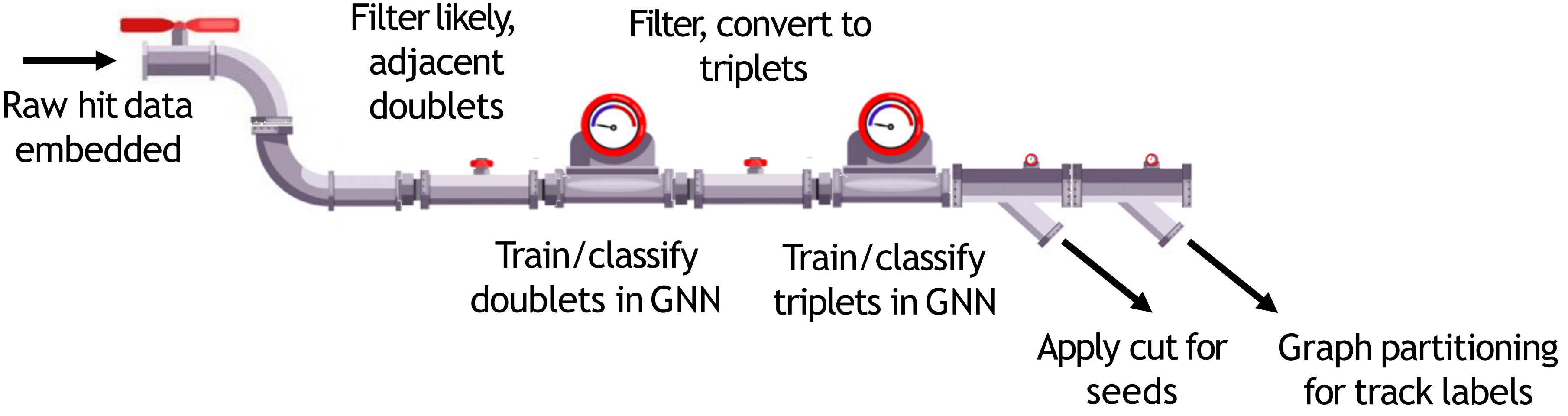
$$S \in [0,1]$$

- All hits belonging to same track labelled with same unique label $\Rightarrow$ $S = 1$

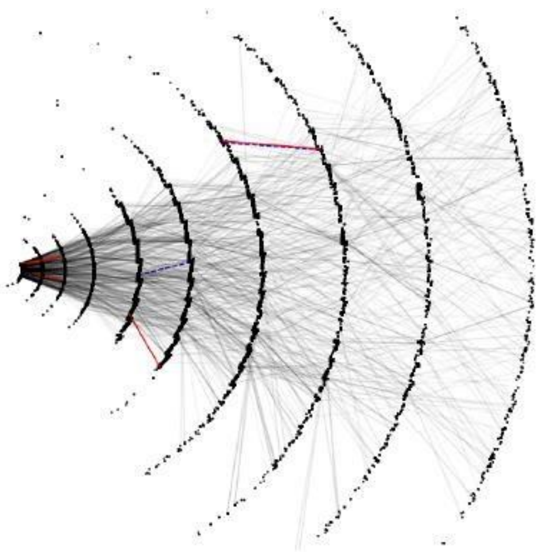


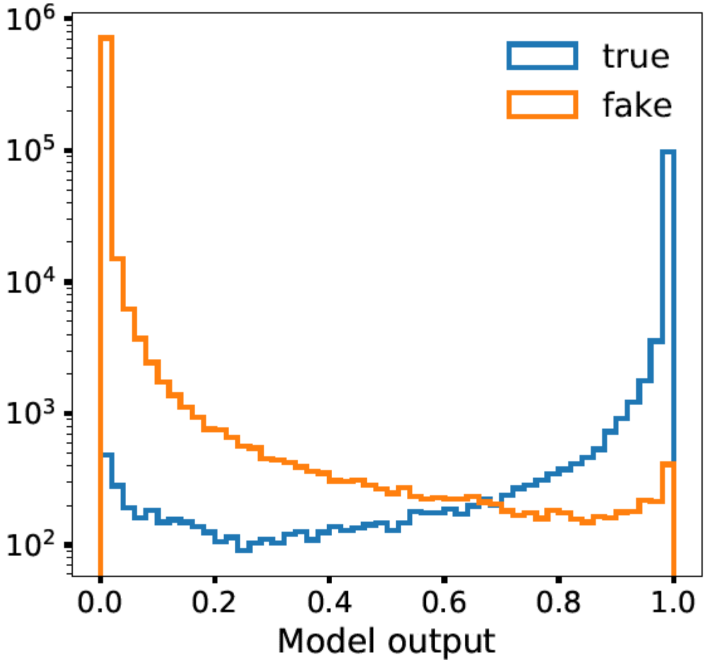
TRACKING PIPELINE

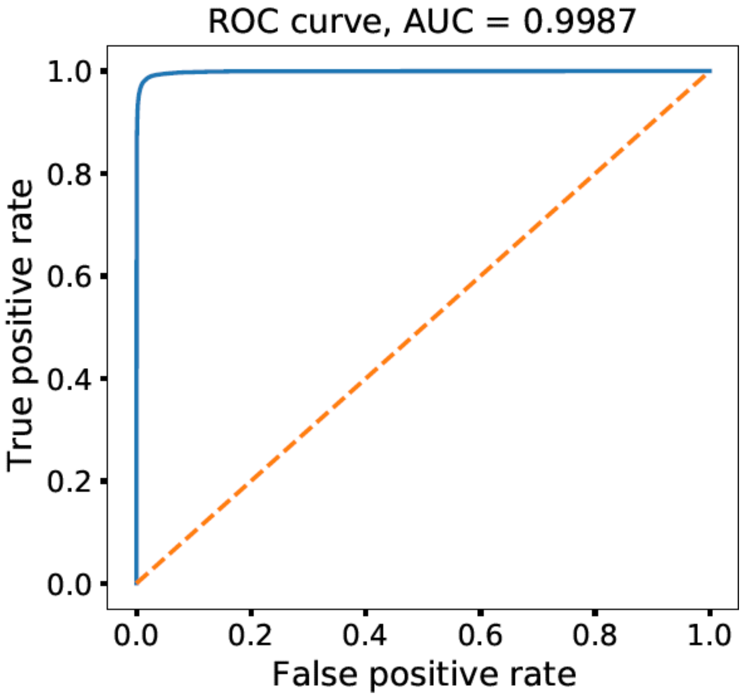
1. Metric Learning
2. Doublet GNN
3. (Optional) Triplet GNN
4. DBSCAN ❼ TrackML score



SPOILER ALERT
GRAPH NEURAL NETWORK PERFORMANCE

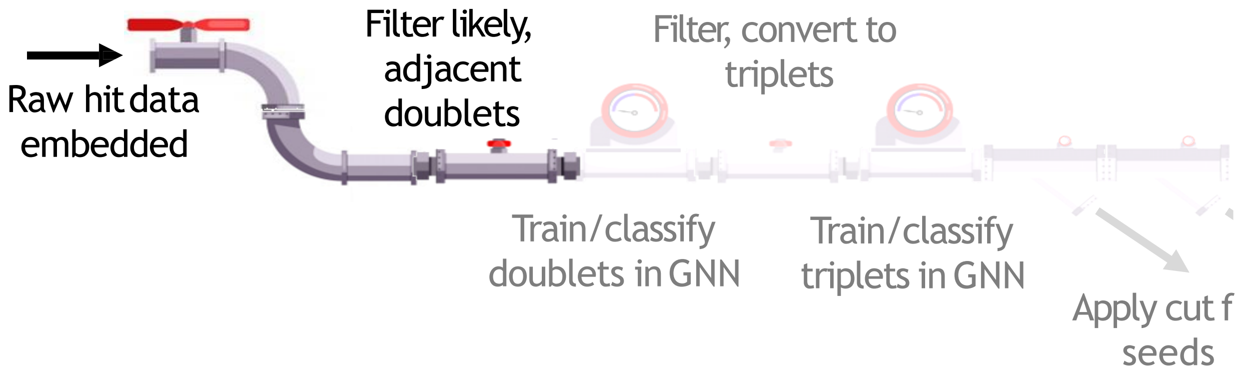




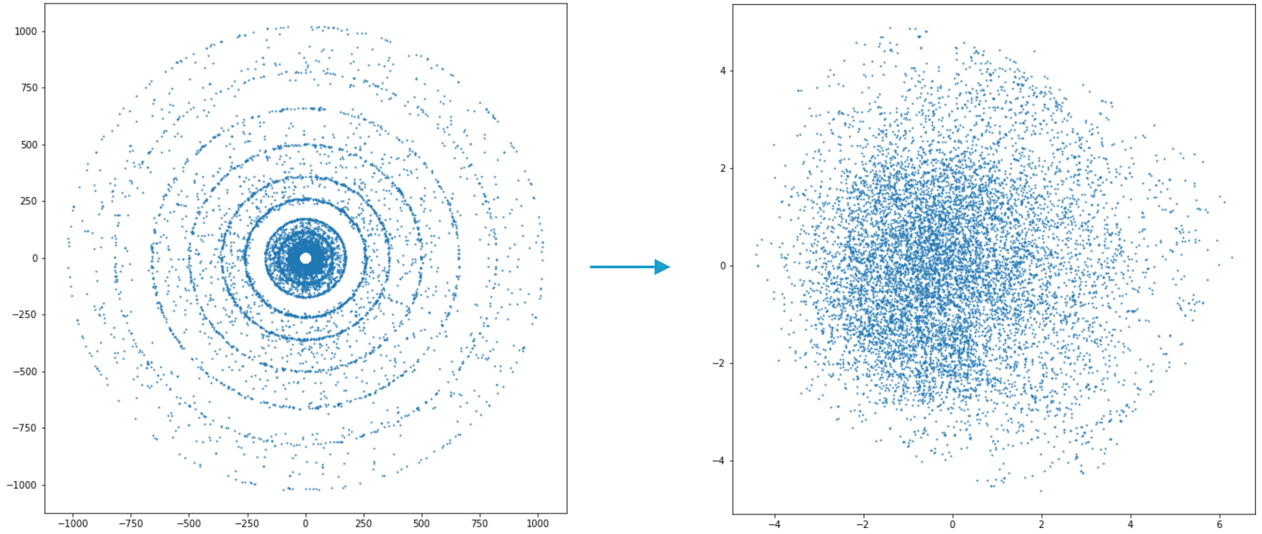


Full detector

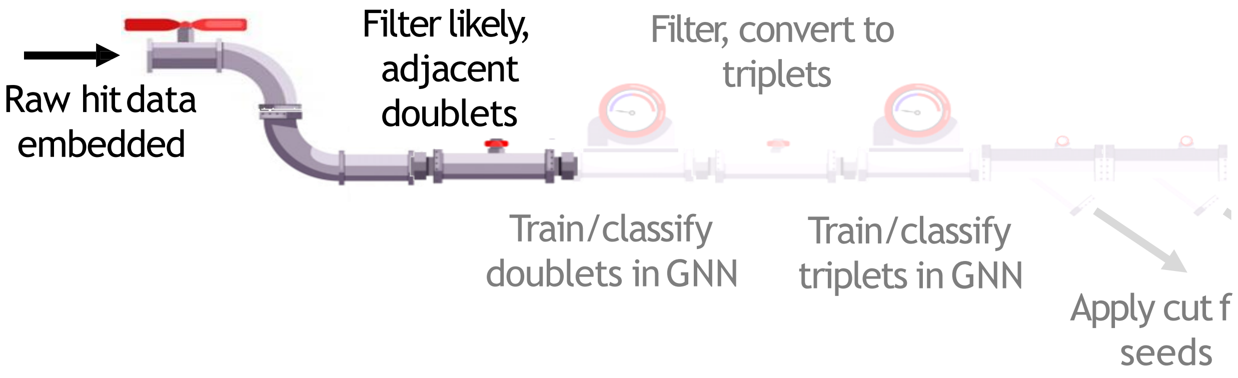
METRIC LEARNING



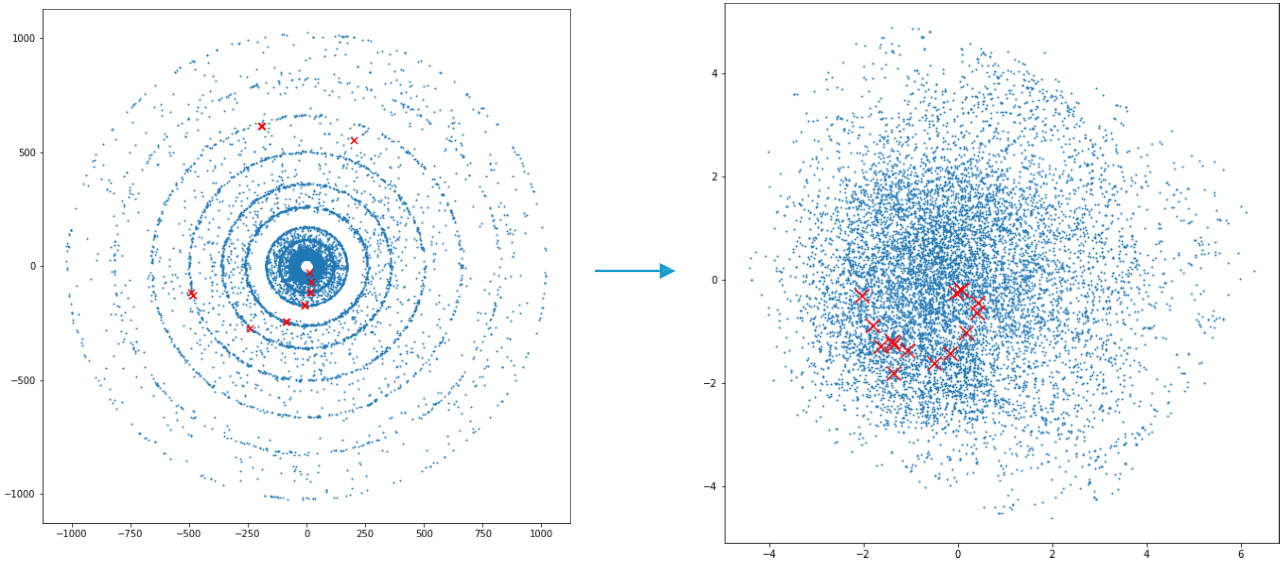
1. For all hits in barrel, embed features (co-ordinates, cell direction data, etc.)
into N-dimensional space



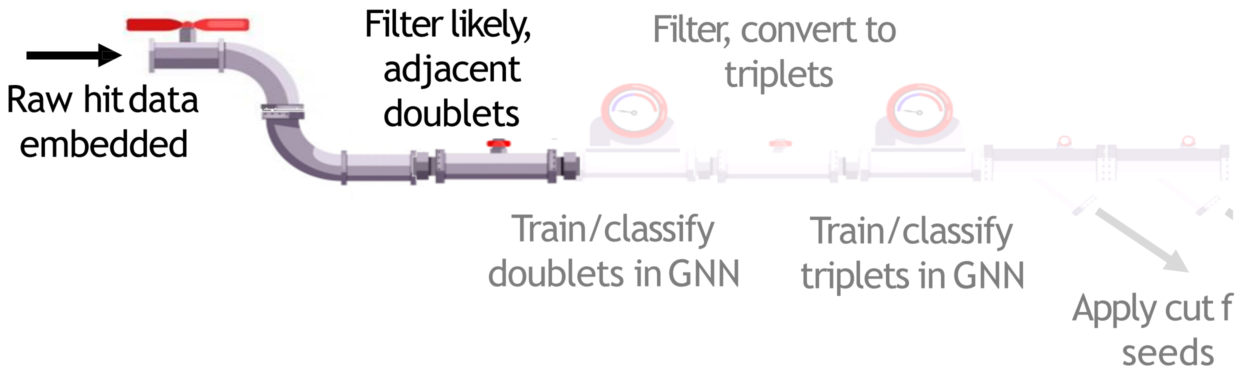
METRIC LEARNING



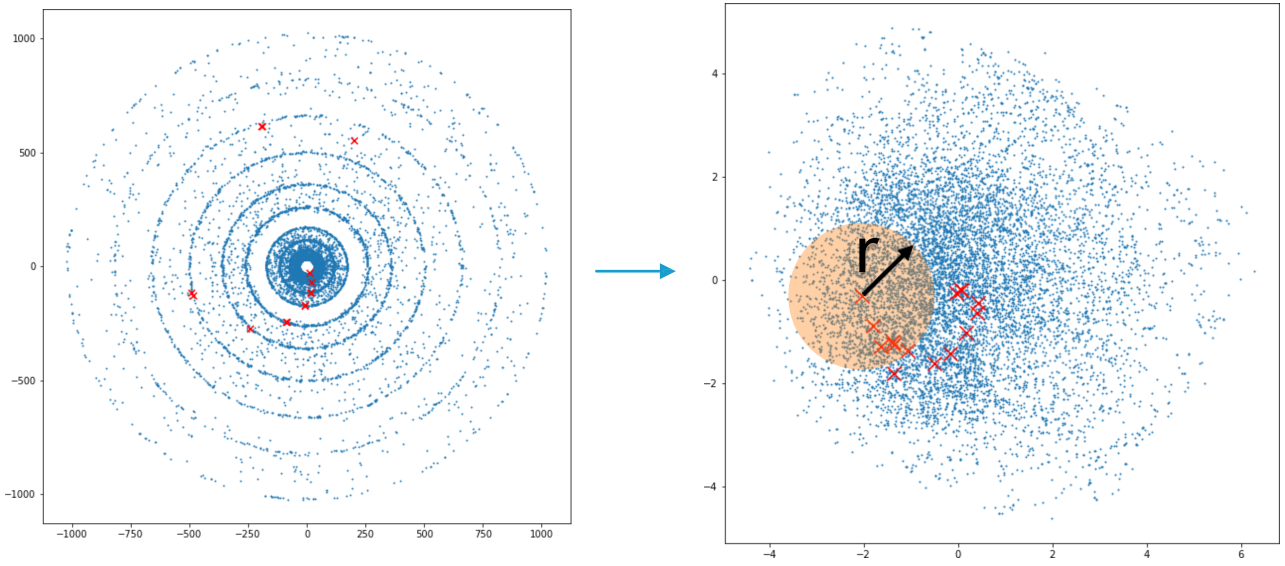
1. For all hits in barrel, embed features (co-ordinates, cell direction data, etc.)
   into N-dimensional space
2. Associate hits from same tracks as close in N-dimensional distance (close = within Euclidean distance *r*)



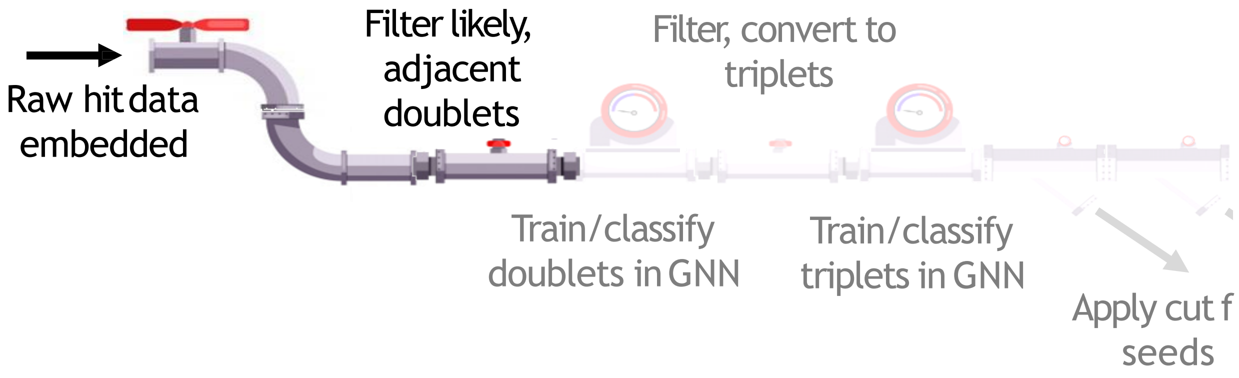
METRIC LEARNING



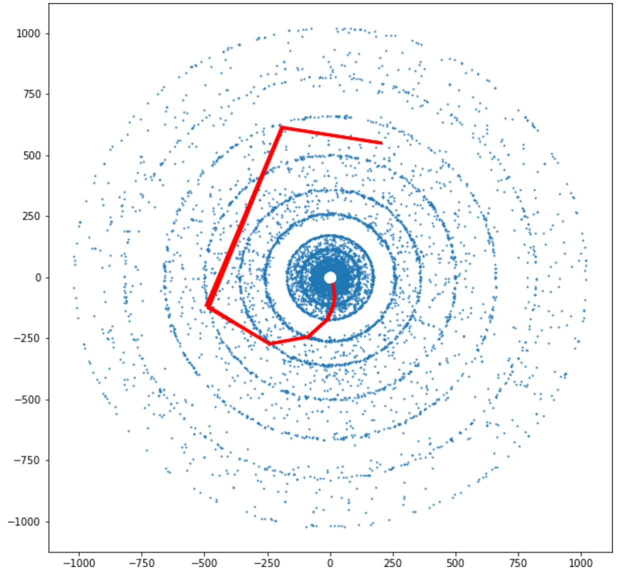
1. For all hits in barrel, embed features (co-ordinates, cell direction data, etc.)
   into N-dimensional space
2. Associate hits from same tracks as close in N-dimensional distance (close = within Euclidean distance *r*)



METRIC LEARNING



1. For all hits in barrel, embed features (co-ordinates, cell direction data, etc.)
   into N-dimensional space
2. Associate hits from same tracks as close in N-dimensional distance
3. Score each “neighbour” hit *within embedding neighbourhood* against
   the “target” hit at centre = Euclidean distance



METRIC LEARNING LOSS FUNCTION

“Comparative” hinge loss

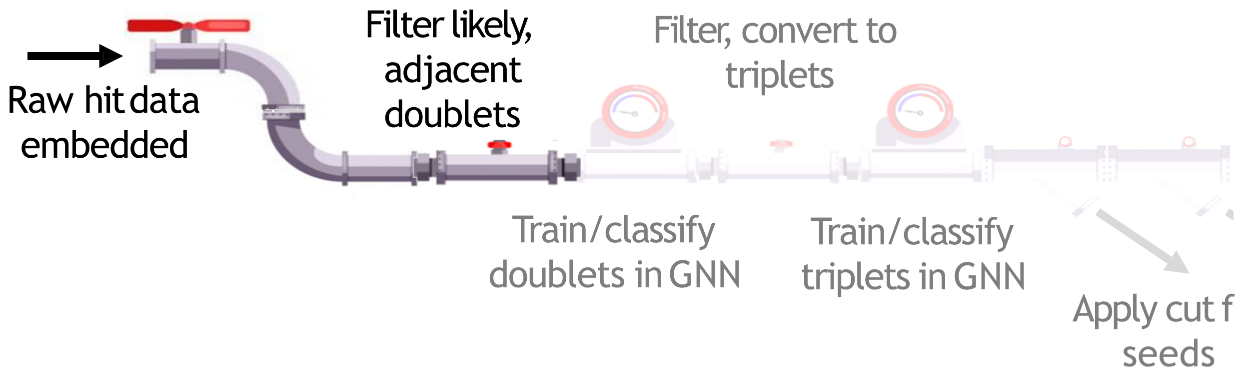
Negatives examples punished for being in margin radius

Positives examples punished for being outside margin radius (Δ)

$$l_n = \begin{cases} x_n, & \text{if } y_n = 1, \\ \max\{0, \Delta - x_n\}, & \text{if } y_n = -1, \end{cases}$$

Pair list (l_n), with associated Euclidean distances (x_n).

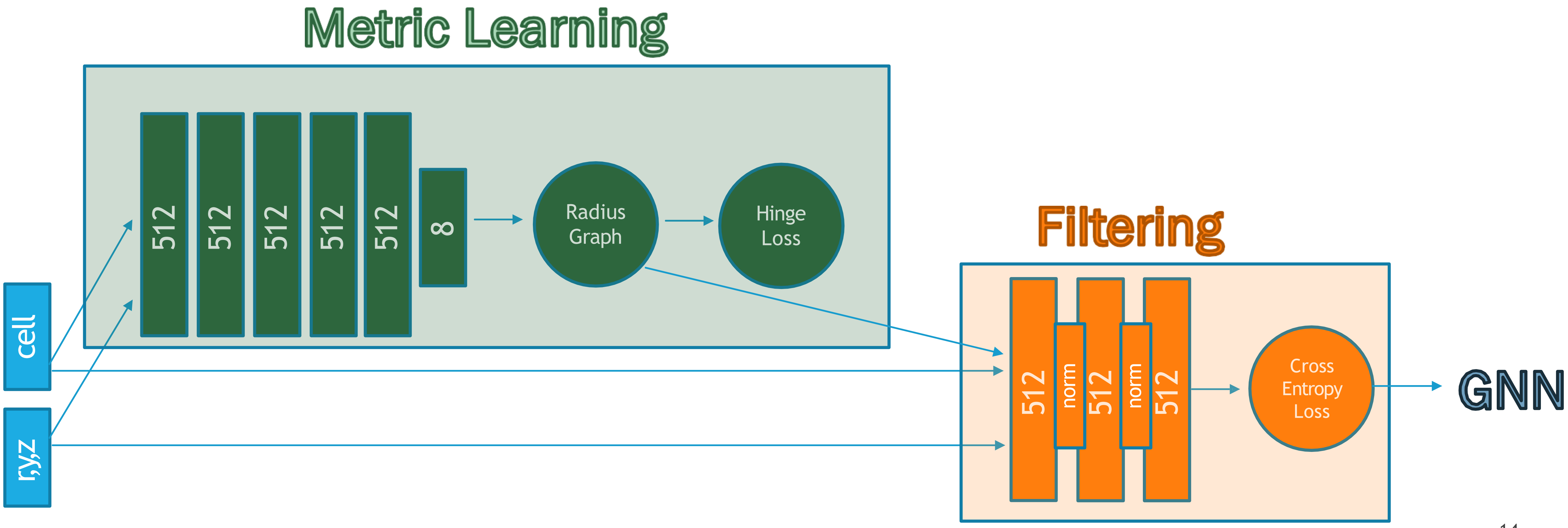
Train with random pairs with only (r,ϕ,z): 0.3 - 0.5% purity @ 96% efficiency.



METRIC LEARNING
FILTERING OUT EASY FAKES

Number of neighbour pairs out of embedding space: O(10 million)

We can apply an MLP to the concatenated pair features to reduce number of pairs (i.e. "edges") to O(3 million)



REFINING "TRUE NEIGHBOURS"

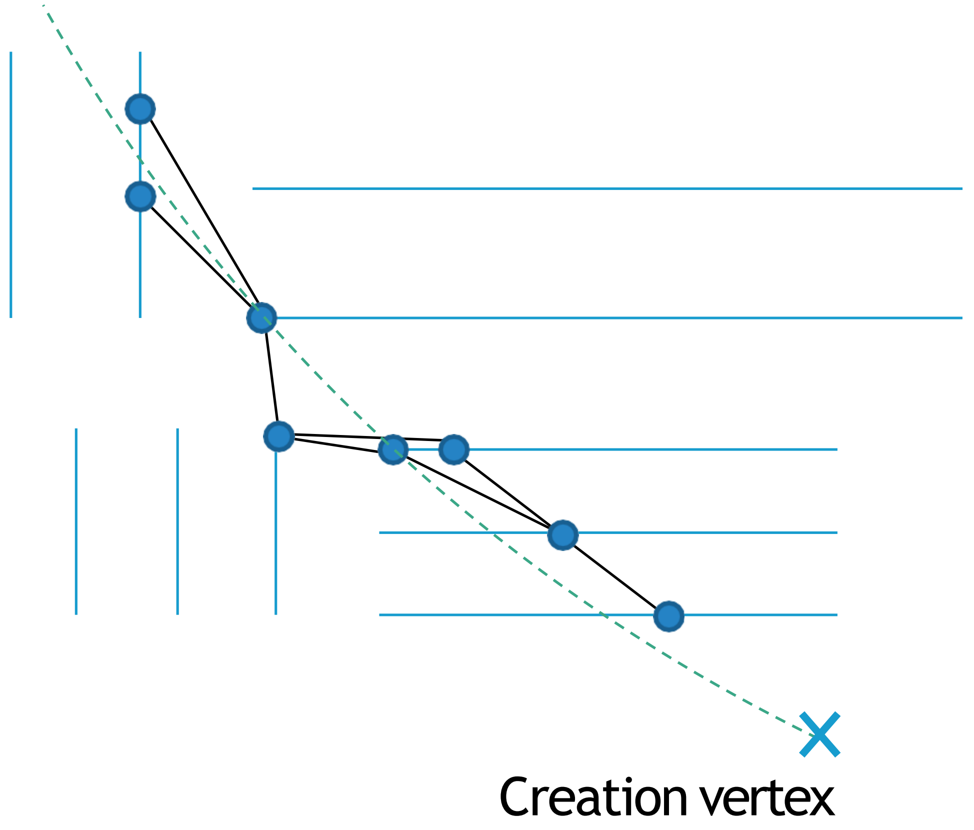
GEOMETRY-FREE GRAPH-TRUTH

1.For each particle, order hits by increasing distance from creation vertex,

$$R = \sqrt{x^2 + y^2 + z^2}$$

2. Group by shared layers

3.Connect all combinations from layer L_i to L_{i+1}, where $R_{i-1} < R_i < R_{i+1}$



FILTERING
TOWARDS REALISTIC TRACKING

Regime	GraphTruth purity @ 99% efficiency	ParticleTruth purity @ 99% efficiency
Vanilla	6.3%	7.8%
Cell info	8.3%	12.8%
Cell info, layer+batch norm	14.0%	17.4%
Graph size	O(1 million edges)	O(3 million edges)

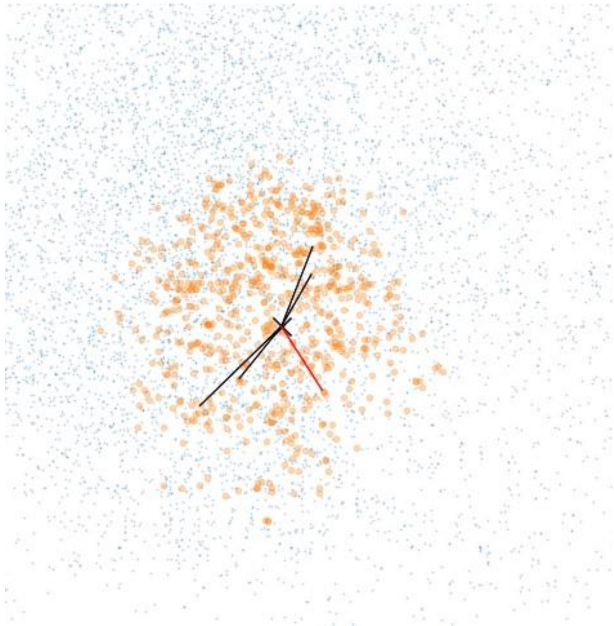
Remember

GraphTruth:
Truth is defined as edges between closest hits in track, on different layers

ParticleTruth:
Truth is defined as any edge connecting hits with the same Particle ID (PID)

FILTERING
TOWARDS REALISTIC TRACKING

Does it work? Let’s check an example:



Pretty good!

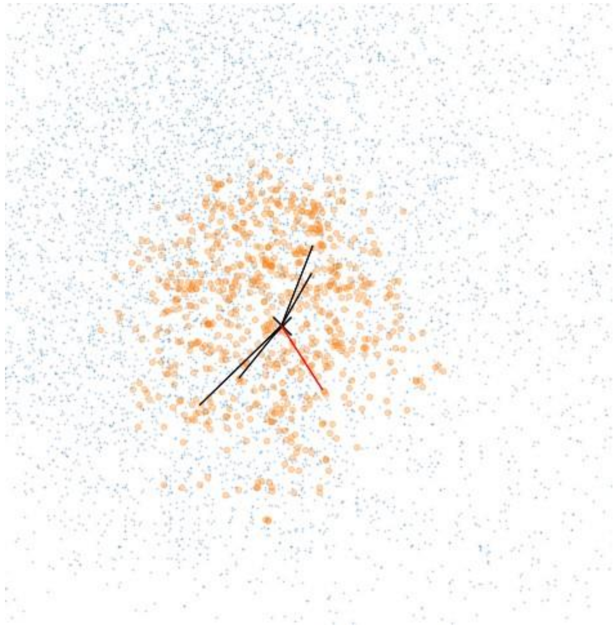
True positive

False positive

No false negatives

FILTERING
TOWARDS REALISTIC TRACKING

Does it work? Let's check an example:



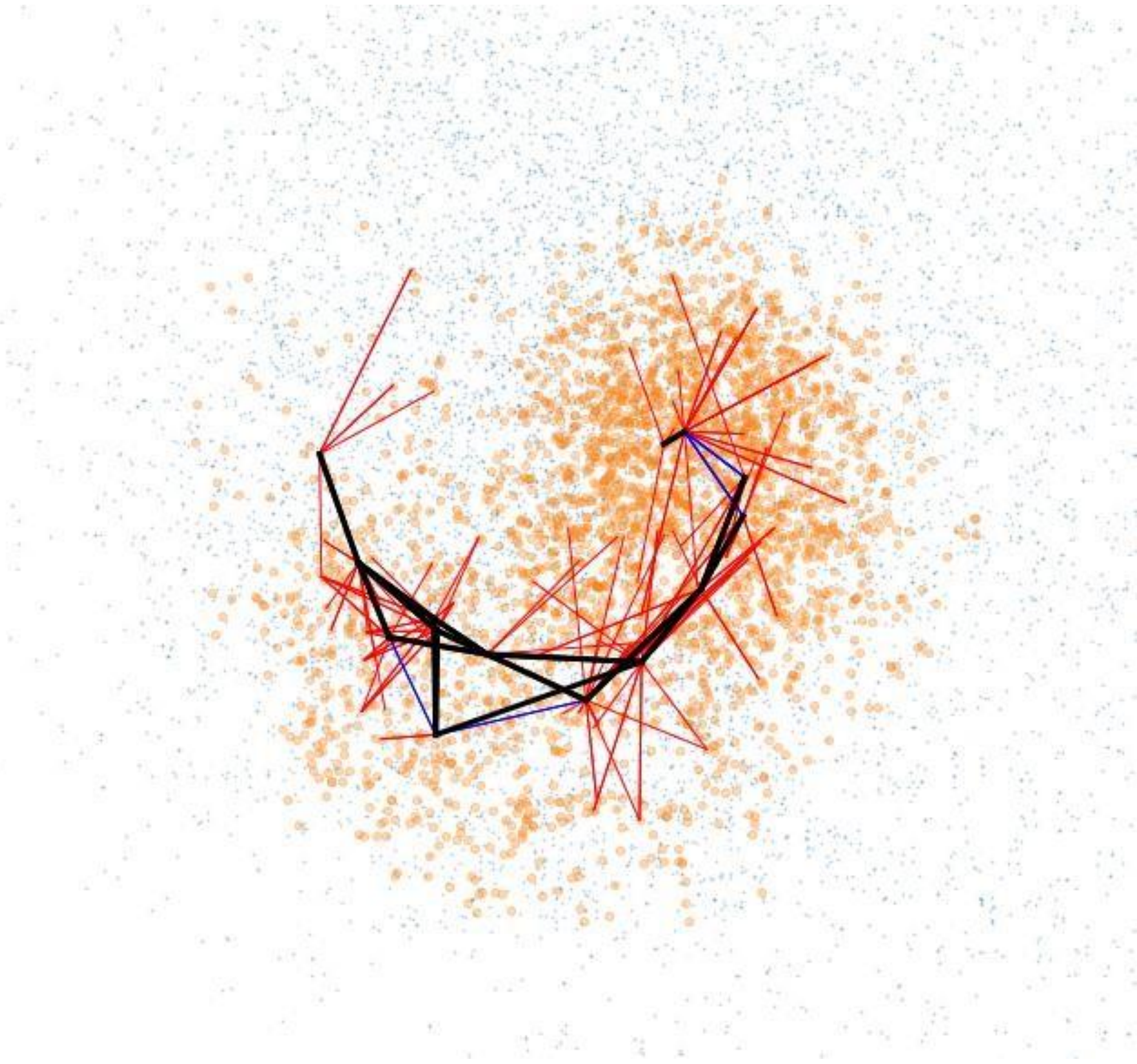
Not quite as good..

True positive

False positive

False negatives

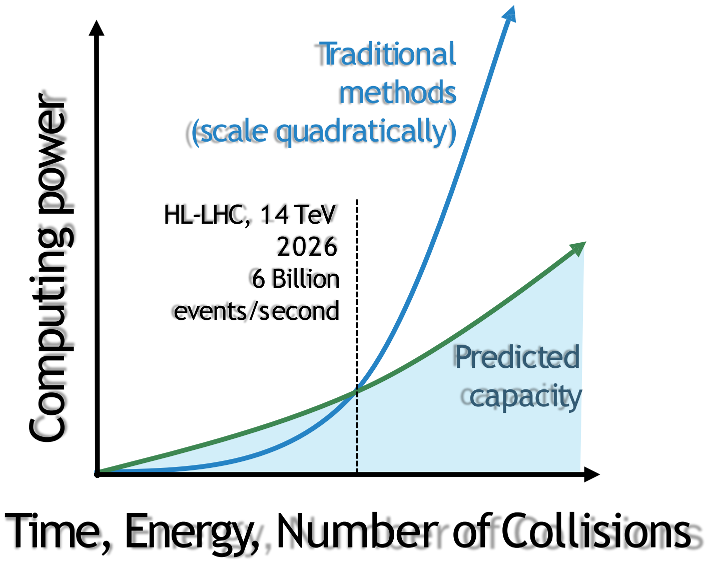
This is where GNN comes in

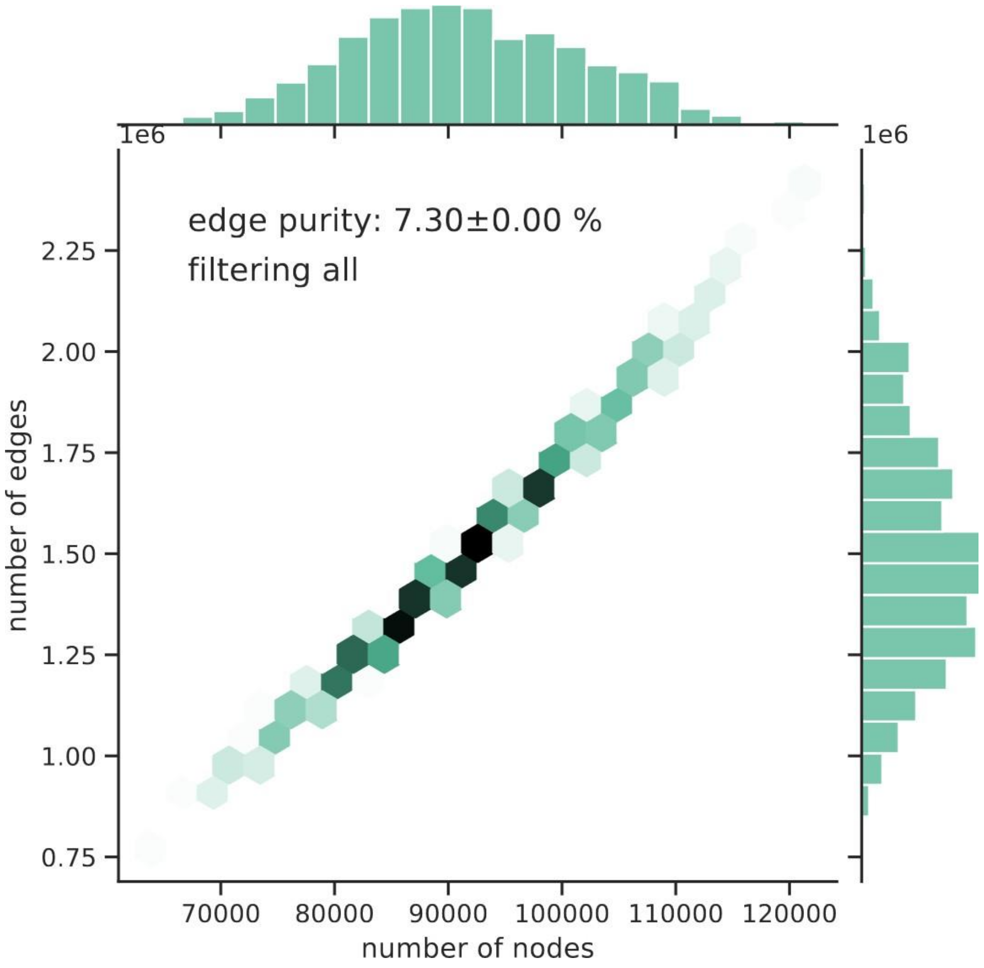


FILTERING
TOWARDS REALISTIC TRACKING

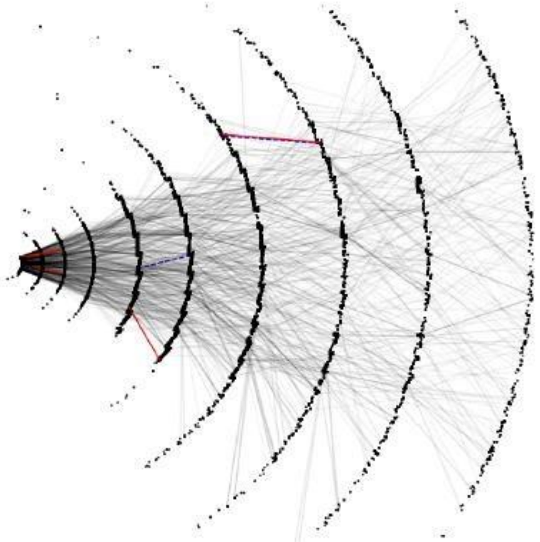
We get sub-quadratic scaling of number of edges with number of hits

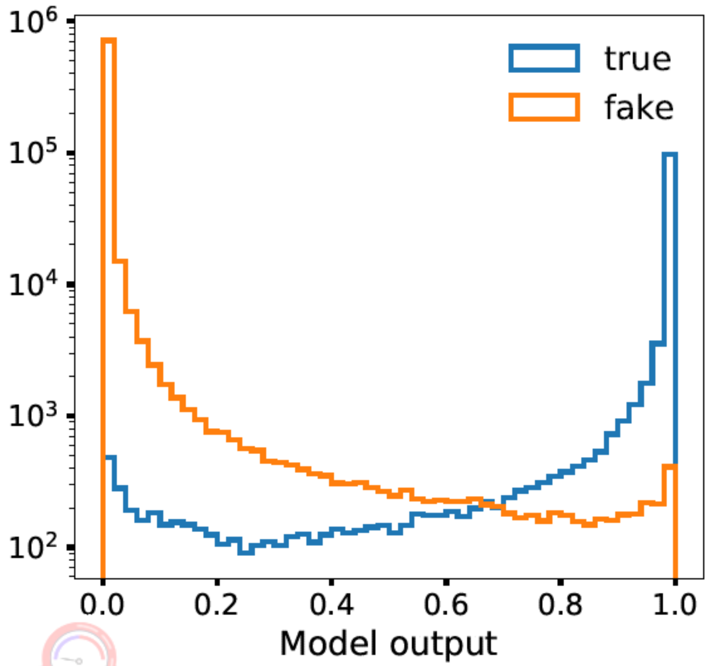
(I.e. We are on track to beating the curve)

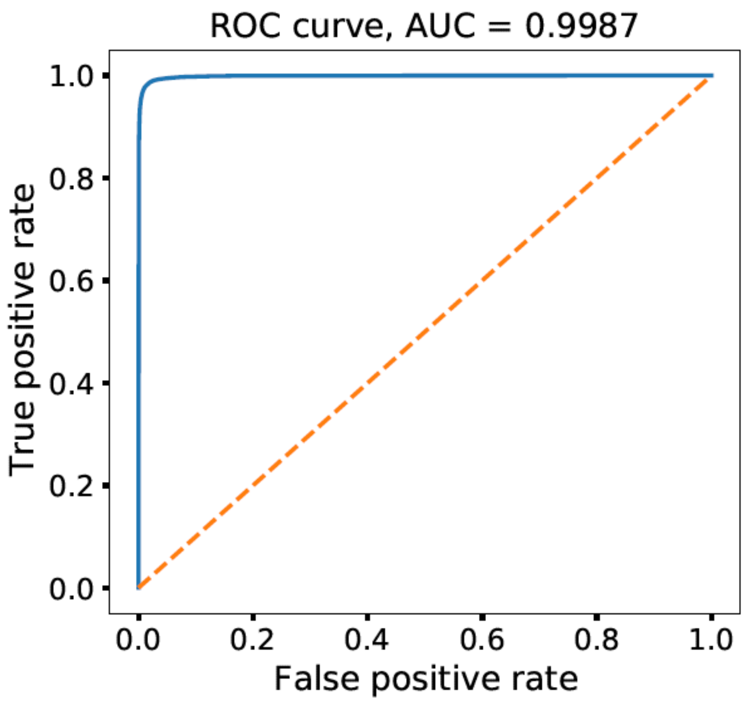


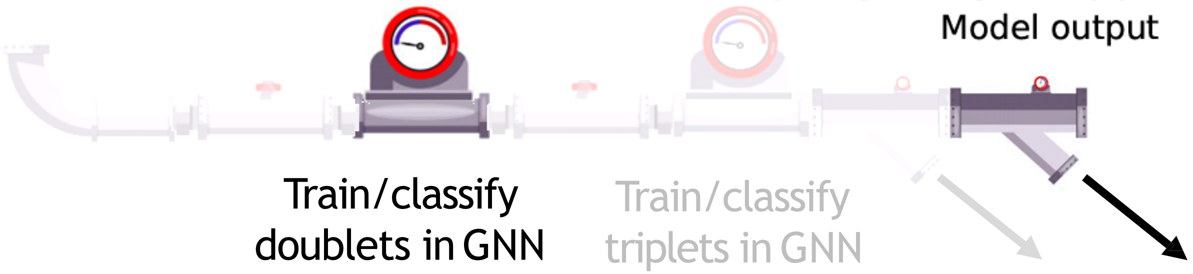


AFTER GRAPH CONSTRUCTION
GRAPH NEURAL NETWORK PERFORMANCE

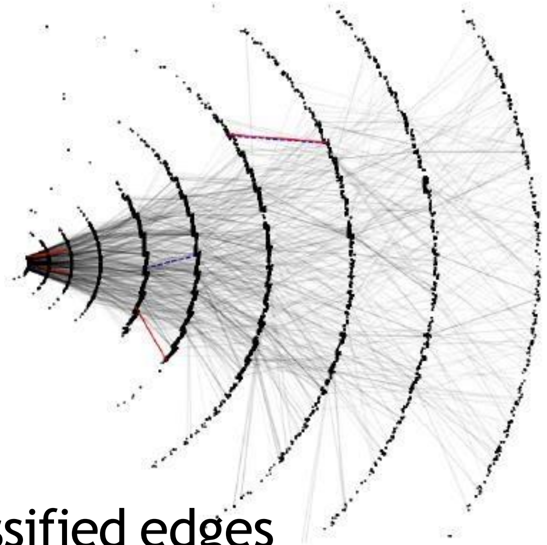






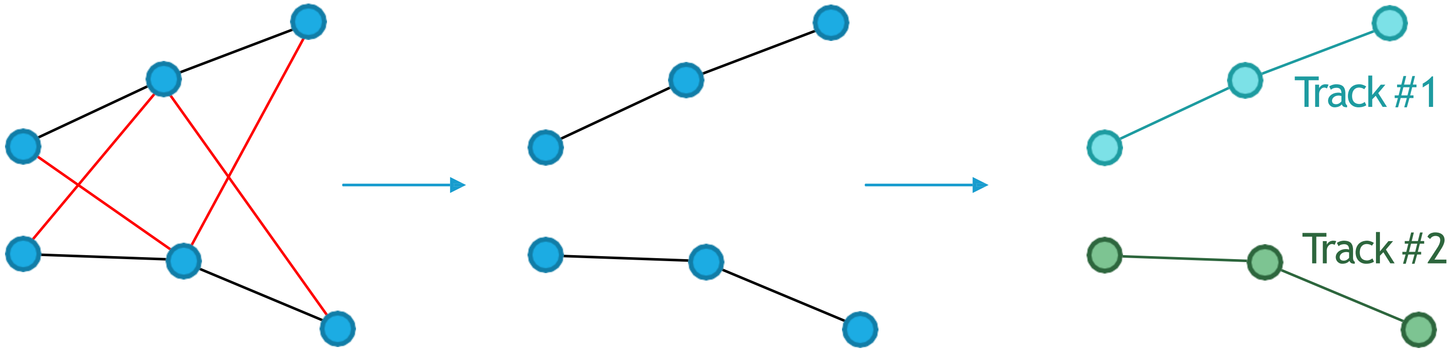


AFTER GRAPH CONSTRUCTION
GRAPH NEURAL NETWORK PERFORMANCE



Classified edges

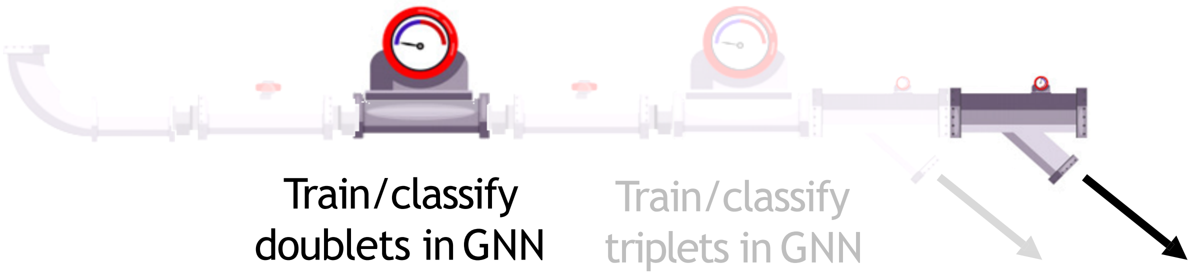
=



Classified edges

Ignore cut edges

Labelled connected components



METRIC LEARNING & GRAPH NEURAL NETWORK PIPELINE

SUMMARY

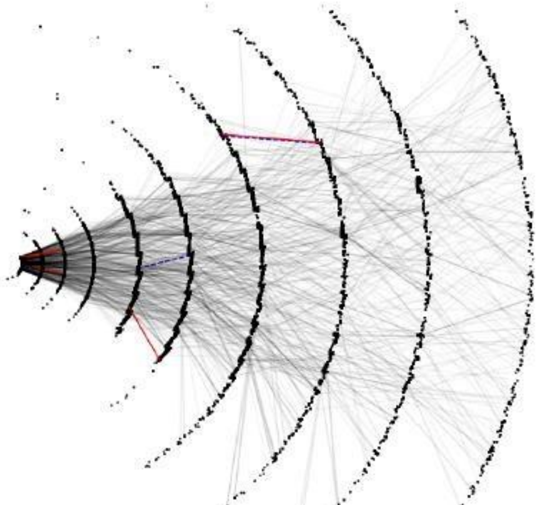
We handle full detector, noise, geometry-free inference, distributed training, with care

Can learn embedding space without layer information, provided we equip training with hard negative mining, cell information, warmup

Can run GNN of full event, provided we equip training with gradient checkpointing, mixed precision

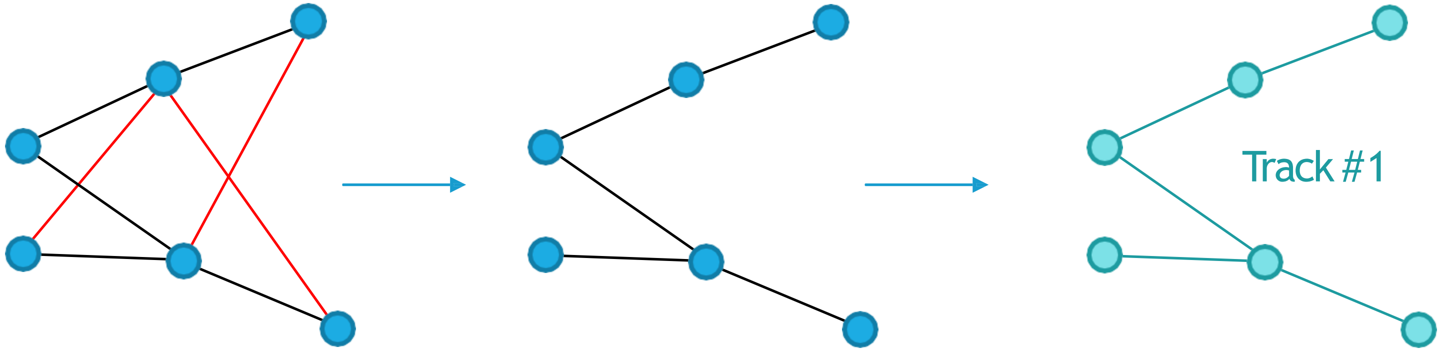
Can include noise without re-training, at a small (~20%) penalty to purity

BUT…
IT'S NOT ALL SUNSHINE AND ROSES



If there's a misclassified edge…

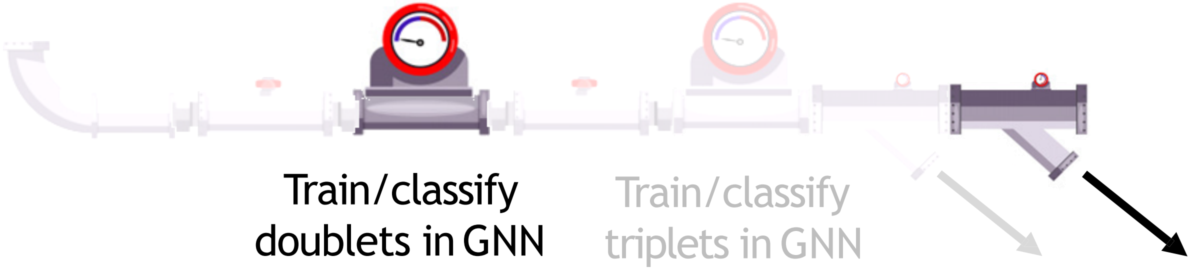
=



Classified edges

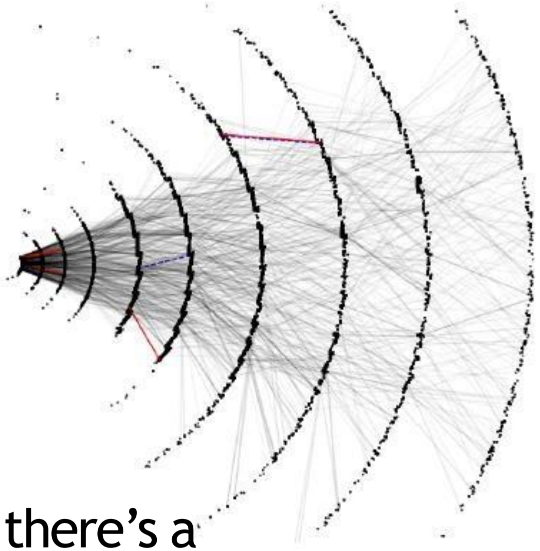
Ignore cut edges

Labelled connected components



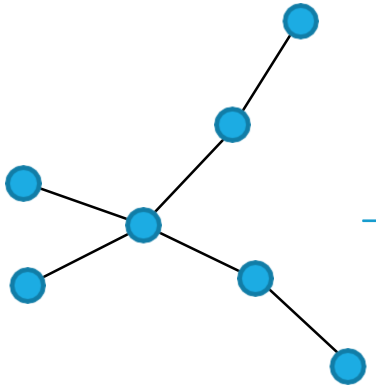
ALSO…

IN REALITY HITS CAN BE COMPOSED OF MULTIPLE PARTICLES



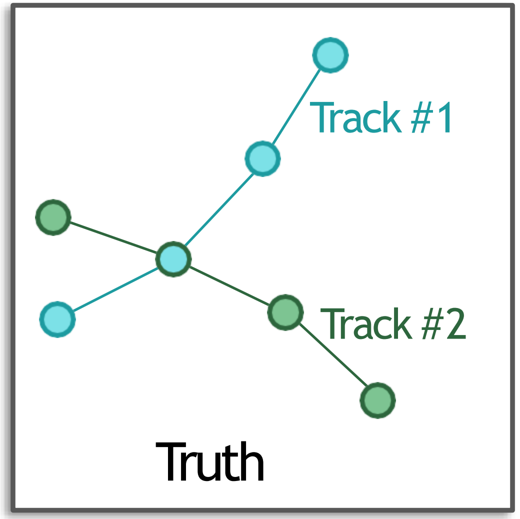
If there’s a misclassified edge…

=



Classified edges

Labelled connected components



Train/classify doublets in GNN

Train/classify triplets in GNN

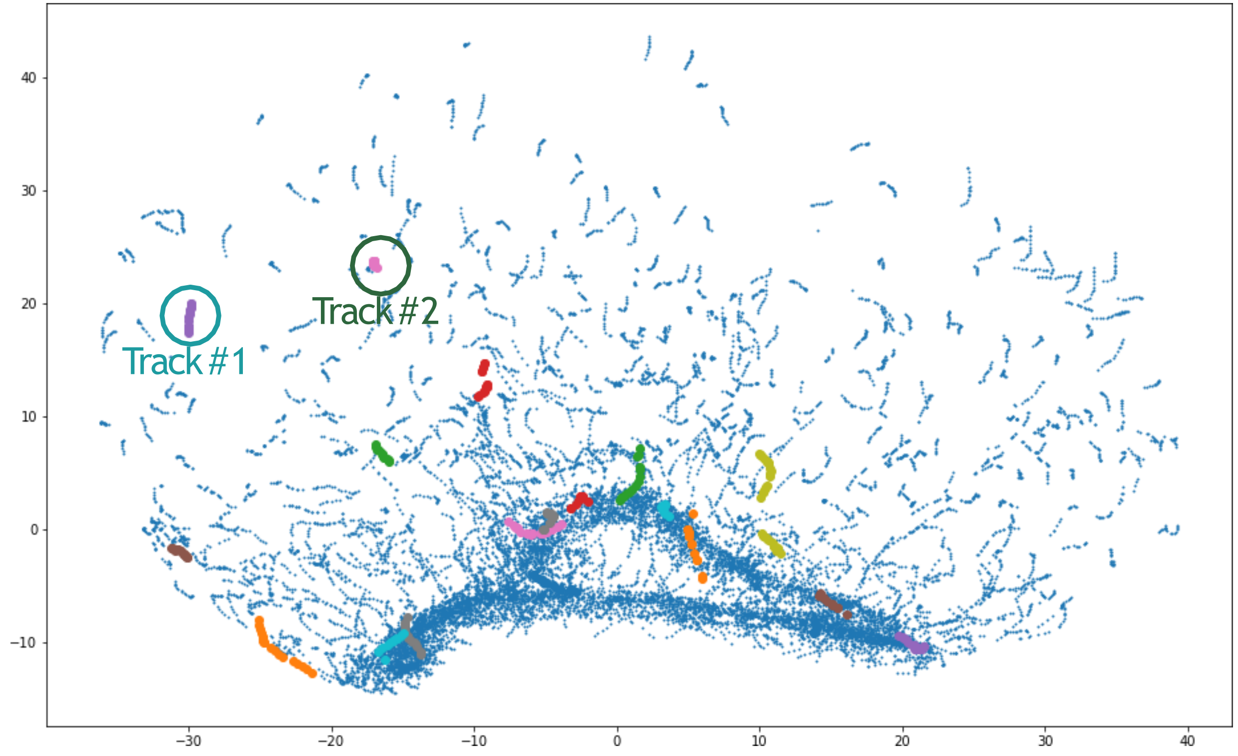
Apply cut for seeds

Graph partitioning for track labels

WHAT ARE WE REALLY TRYING TO DO HERE?

Goal: Assign labels to hits

- If GNN had 100% efficiency and 100% purity, we could apply hard cut, get connected components, and get perfect track labelling
- Can we train to a final goal of grouping nodes/hits into clusters
- We could use metric learning and label tracks using neighbour method

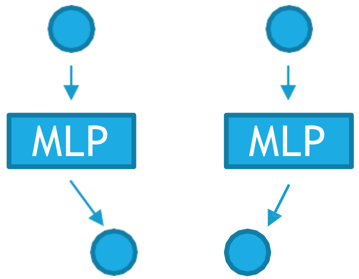


WHAT DO WE HAVE AT OUR DISPOSAL?

Metric Learning

Powered by:
Embedded space & hinge loss

Benefit:
Similarity search

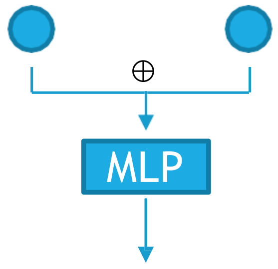


$$\mathcal{L} = |x'_1 - x'_2|$$

Pair Classification

Powered by:
Concatenation of hit features & cross entropy loss

Benefit:
Expressive parameterisation

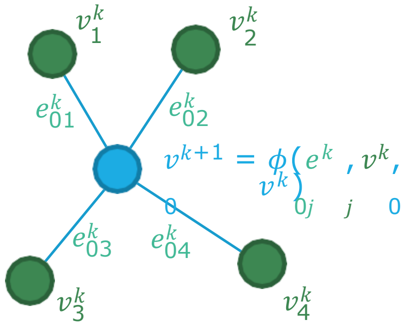


$$\mathcal{L} = f(x_1, x_2)$$

Graph Neural Network

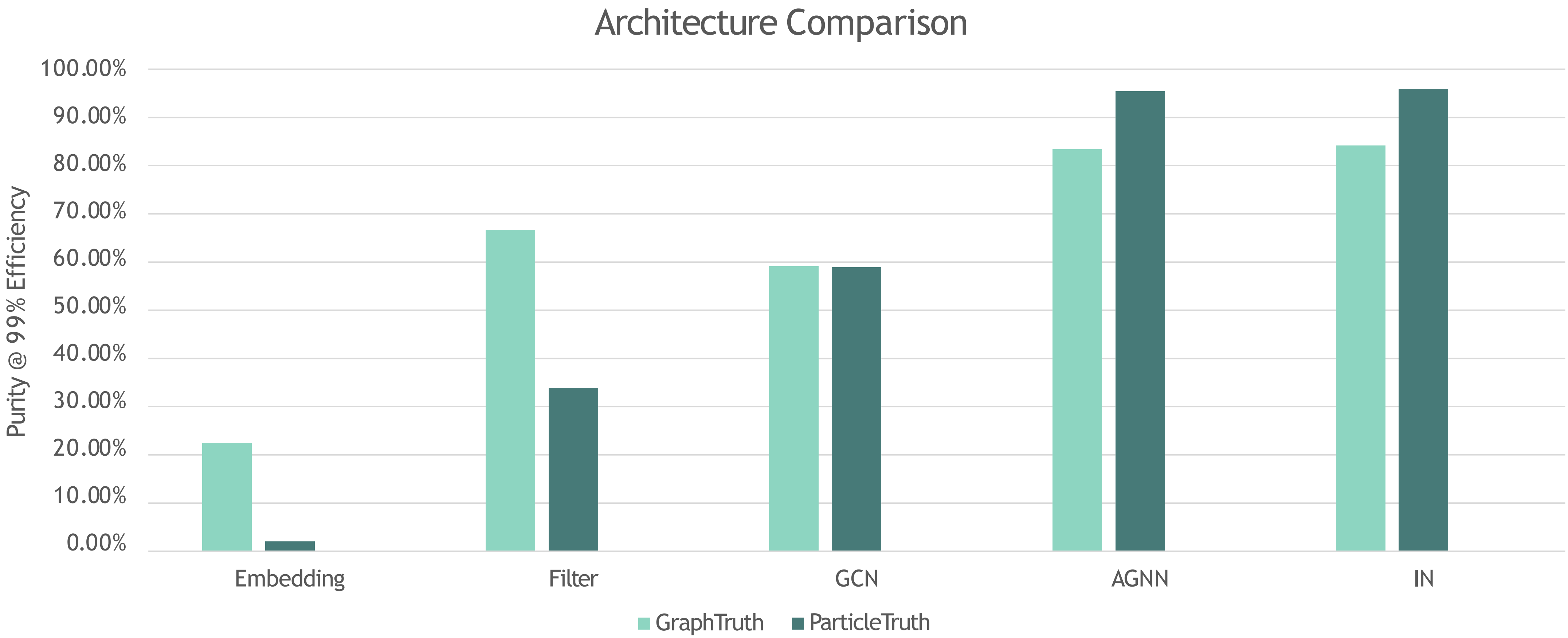
Powered by:
Message passing & attention mechanism

Benefit:
High accuracy



$$\mathcal{L} = f(v_1, v_{knn})$$

HOW DO THEY STACK UP?

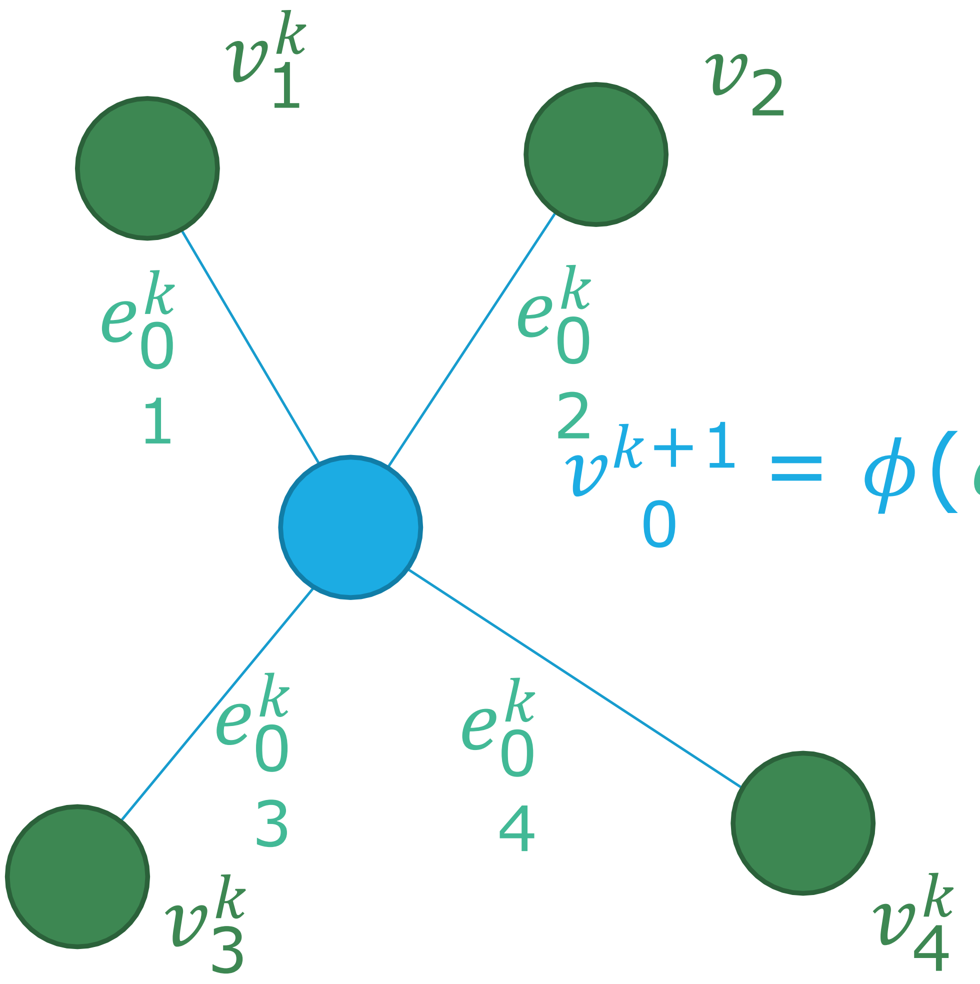


GCN: Graph Convolution Network
AGNN: Attention Graph Neural Network
INN: Interaction Network

GRAPH NEURAL NETWORKS

ARCHITECTURES

MESSAGE PASSING



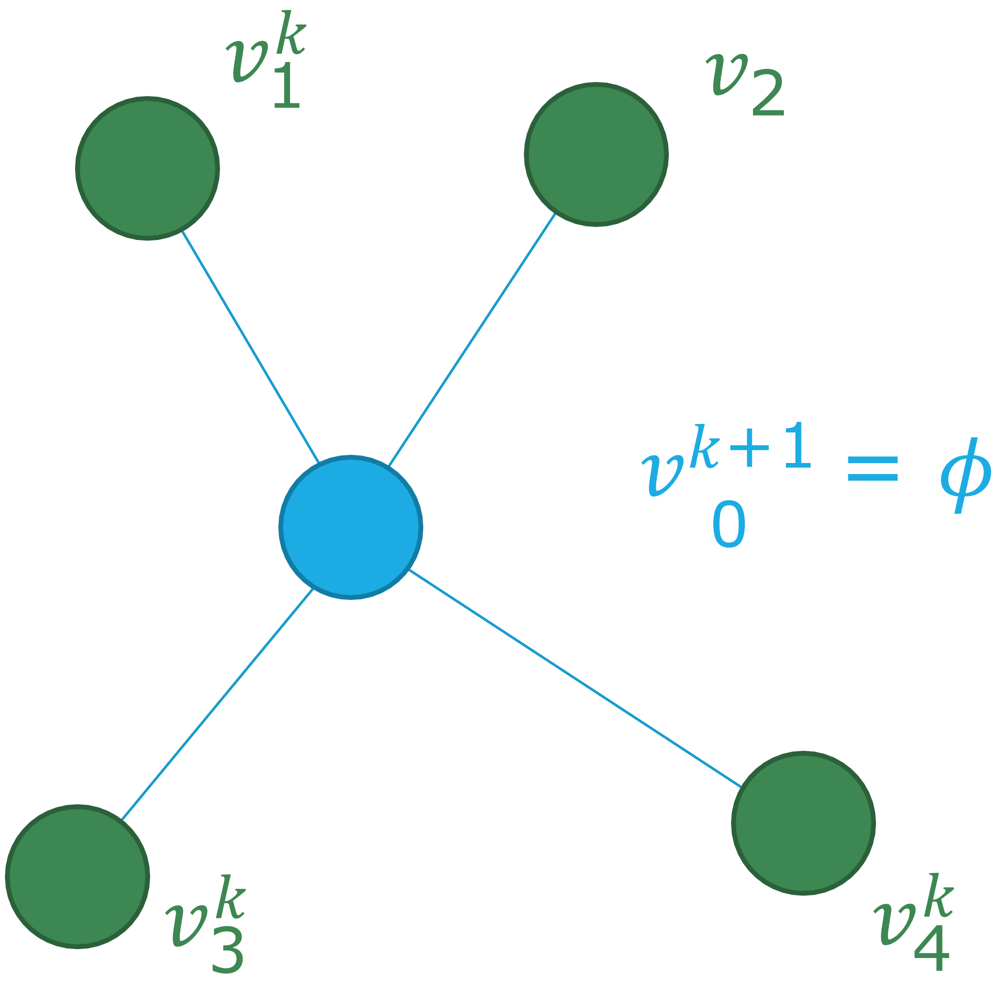
$$v_0^{k+1} = \phi(e_{0j}^k, v_j^k, v_0^k)$$

v_i^k node features
e_{ij}^k edge features
at iteration k

GRAPH NEURAL NETWORKS
ARCHITECTURES

GRAPH CONVOLUTION NETWORK

Kipf & Welling "Semi-Supervised Classification with Graph Convolutional Networks" *arXiv:1609.02907* (2016).



v_i^k node features at iteration k

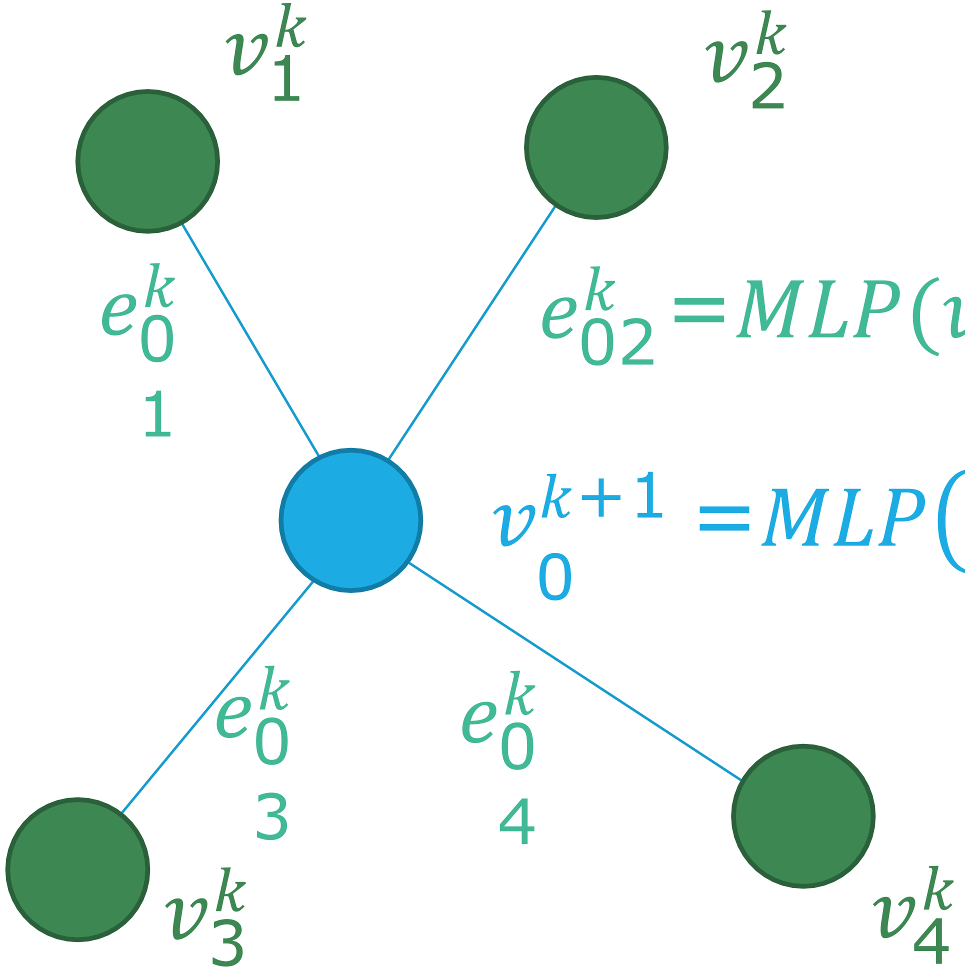
$$v_0^{k+1} = \phi(v_j^k, v_0^k)$$

GRAPH NEURAL NETWORKS
ARCHITECTURES

v_i^k node features
e_{ij}^k edge score [0, 1]
at iteration k

ATTENTION GNN

Veličković, Petar, et al. "Graph attention networks" *arXiv preprint arXiv:1710.10903* (2017).



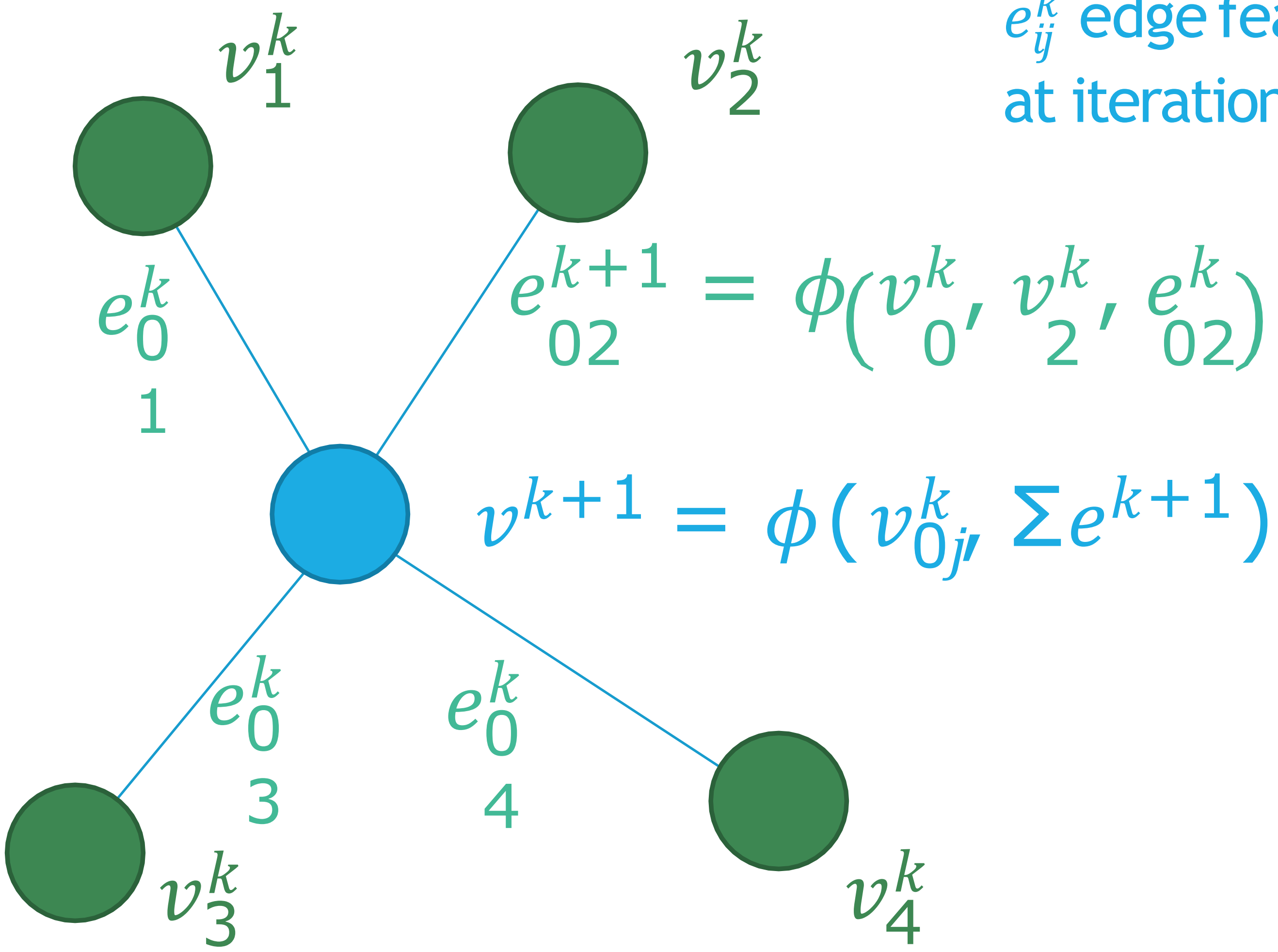
$$e_{02}^k = MLP(v_0^k, v_2^k)$$

$$v_0^{k+1} = MLP\left(\left[\left(\sum e_{0j}^k\, v_j^k\right), v_0^k\right]\right) + v_0^k$$

GRAPH NEURAL NETWORKS ARCHITECTURES

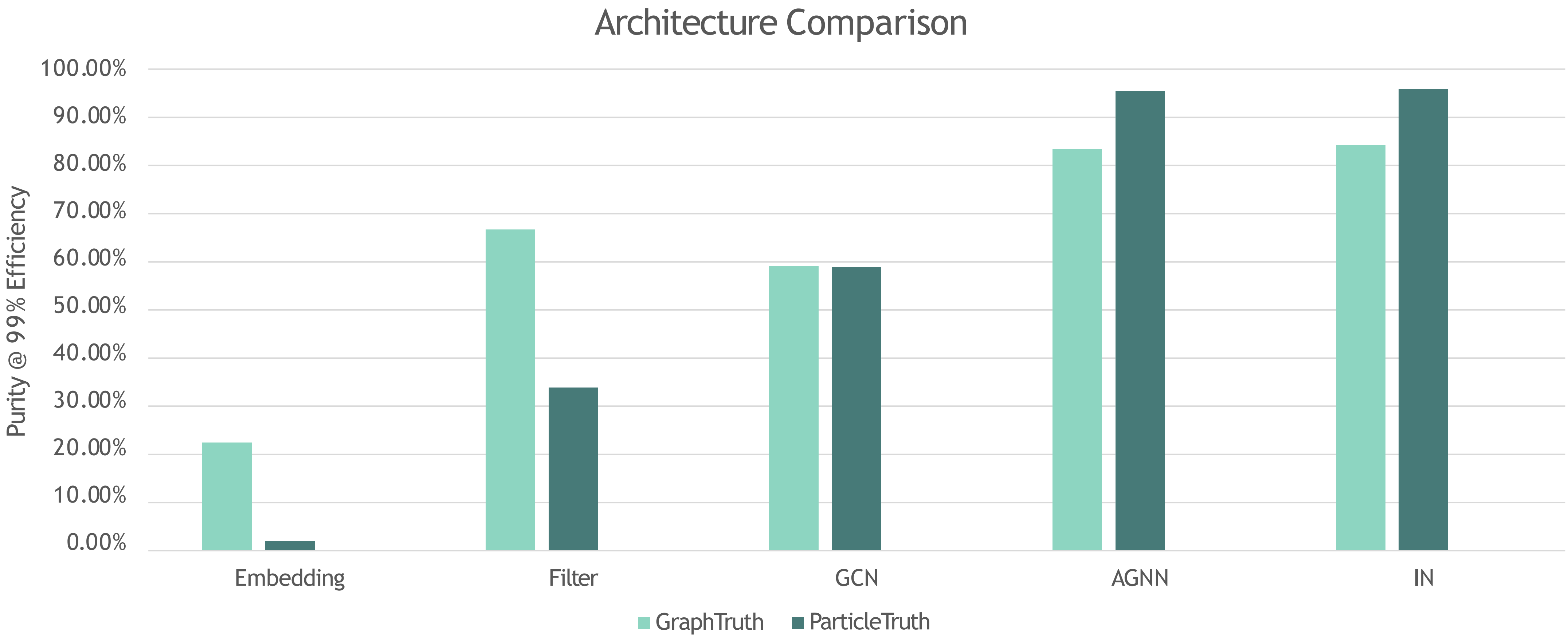
v_i^k node features
e_{ij}^k edge features
at iteration k

INTERACTION NETWORK



Battaglia, Peter, et al. "Interaction networks for learning about objects, relations and physics." *Advances in neural information processing systems*. 2016.

HOW DO THEY STACK UP?



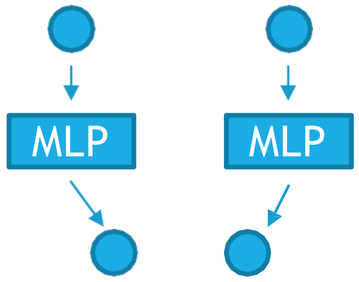
GCN: Graph Convolution Network
AGNN: Attention Graph Neural Network
INN: Interaction Network

WHAT DO WE HAVE AT OUR DISPOSAL?

Metric Learning

Powered by:
Embedded space & hinge loss

Benefit:
Similarity search

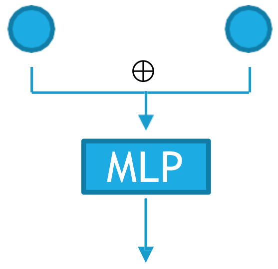


$$\mathcal{L} = |x'_1 - x'_2|$$

Pair Classification

Powered by:
Concatenation of hit features & cross entropy loss

Benefit:
Expressive parameterisation

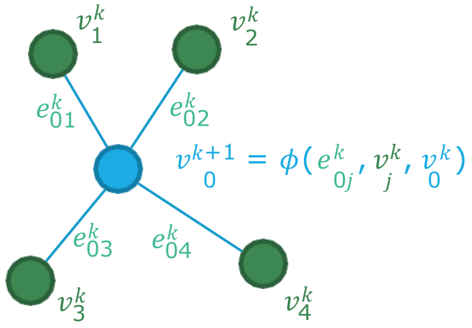


$$\mathcal{L} = f(x_1, x_2)$$

Graph Neural Network

Powered by:
Message passing & attention mechanism

Benefit:
High accuracy



$$\mathcal{L} = f(v_1, v_{knn})$$

WHAT DO WE HAVE AT OUR DISPOSAL?

Metric Learning

Powered by:
Embedded space & hinge loss

Benefit:
Similarity search

+

Pair Classification

Powered by:
Concatenation of hit features & cross entropy loss

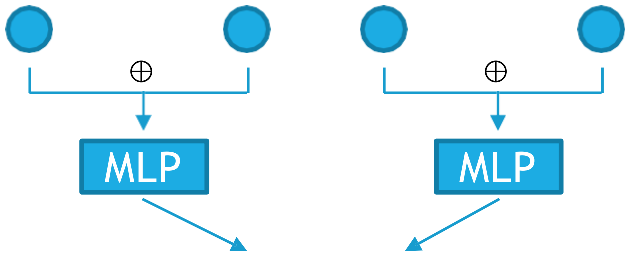
Benefit:
Expressive parameterisation

Graph Neural Network

Powered by:
Message passing & attention mechanism

Benefit:
High accuracy

PairEmbedding



$$\mathcal{L} = |f(x_1, x_2) - f(x_3, x_4)|$$

WHAT DO WE HAVE AT OUR DISPOSAL?

Metric Learning

Powered by:
Embedded space & hinge loss

Benefit:
Similarity search

+

Pair Classification

Powered by:
Concatenation of hit features & cross entropy loss

Benefit:
Expressive parameterisation

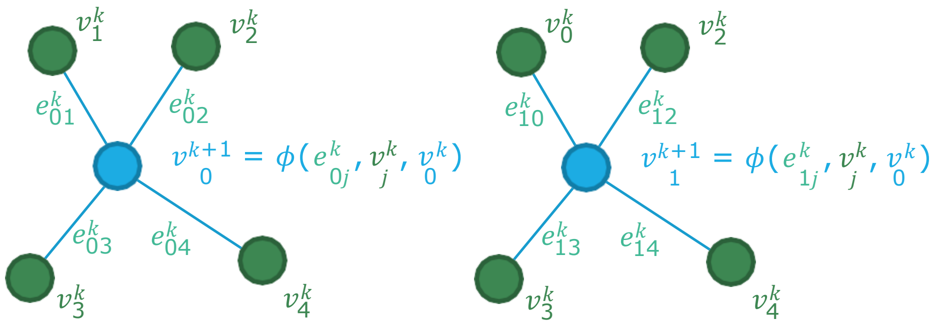
+

Graph Neural Network

Powered by:
Message passing & attention mechanism

Benefit:
High accuracy

GNNNodeEmbedding

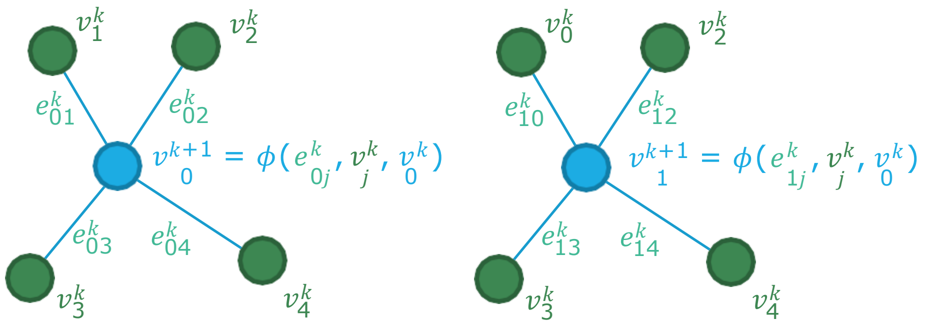


$$\mathcal{L} = |f(v_0, v_{knn}) - f(v_1, v_{knn})|$$

WHAT DO WE HAVE AT OUR DISPOSAL?

Metric Learning		Pair Classification		Graph Neural Network
Powered by: Embedded space & hinge loss	+	Powered by: Concatenation of hit features & cross entropy loss	+	Powered by: Message passing & attention mechanism
Benefit: Similarity search		Benefit: Expressive parameterisation		Benefit: High accuracy

GNNNodeEmbedding



$$\mathcal{L} = |f(v_0, v_{knn}) - f(v_1, v_{knn})|$$

WHAT DO WE HAVE AT OUR DISPOSAL?

Metric Learning

Powered by:
Embedded space & hinge loss

Benefit:
Similarity search

+

Pair Classification

Powered by:
Concatenation of hit features & cross entropy loss

Benefit:
Expressive parameterisation

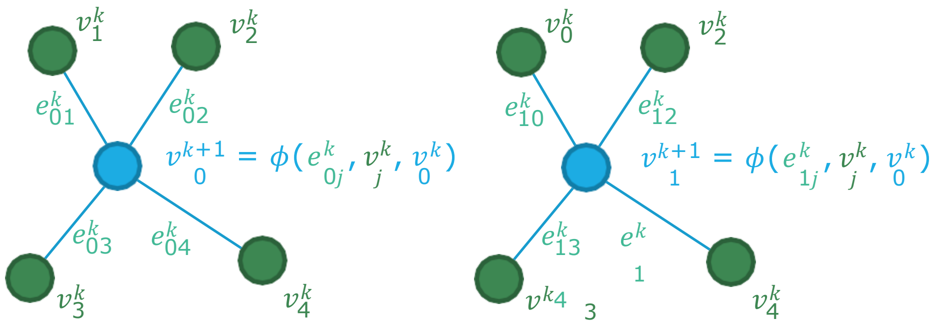
+

Graph Neural Network

Powered by:
Message passing & attention mechanism

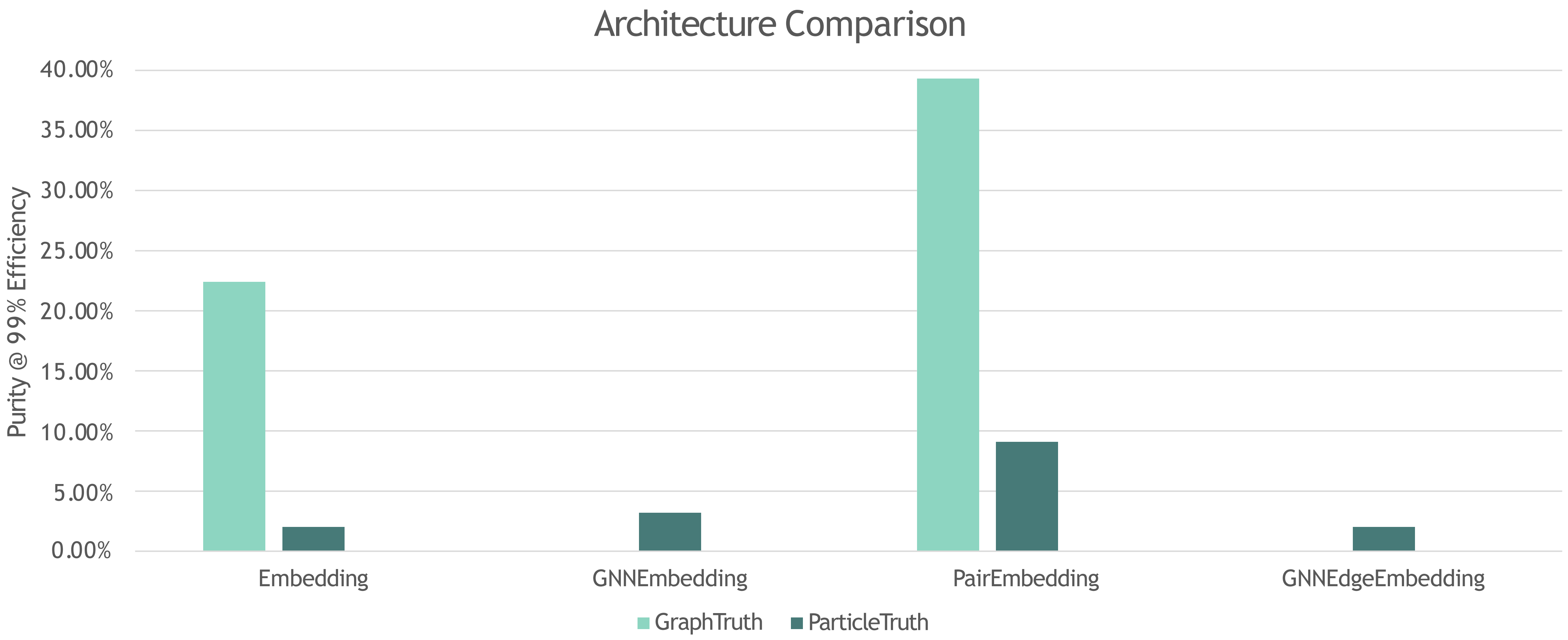
Benefit:
High accuracy

GNNEdgeEmbedding



$$\mathcal{L} = |f(e_0) - f(e_1)|$$

DO THESE HYBRIDS WORK?



WHY DOESN'T GRAPH METRIC LEARNING WORK (YET)?

Still running HPO, may simply be an issue of learning rate, choice of dimensionality, etc.

But best performance still obtained by 1 graph iteration

I.e. Message passing and/or attention is not contributing informative features

Intuition: It should work. GNN edge classification has superior performance, utilising message passing and attention. Metric learning uses the same architecture, only with a different final task