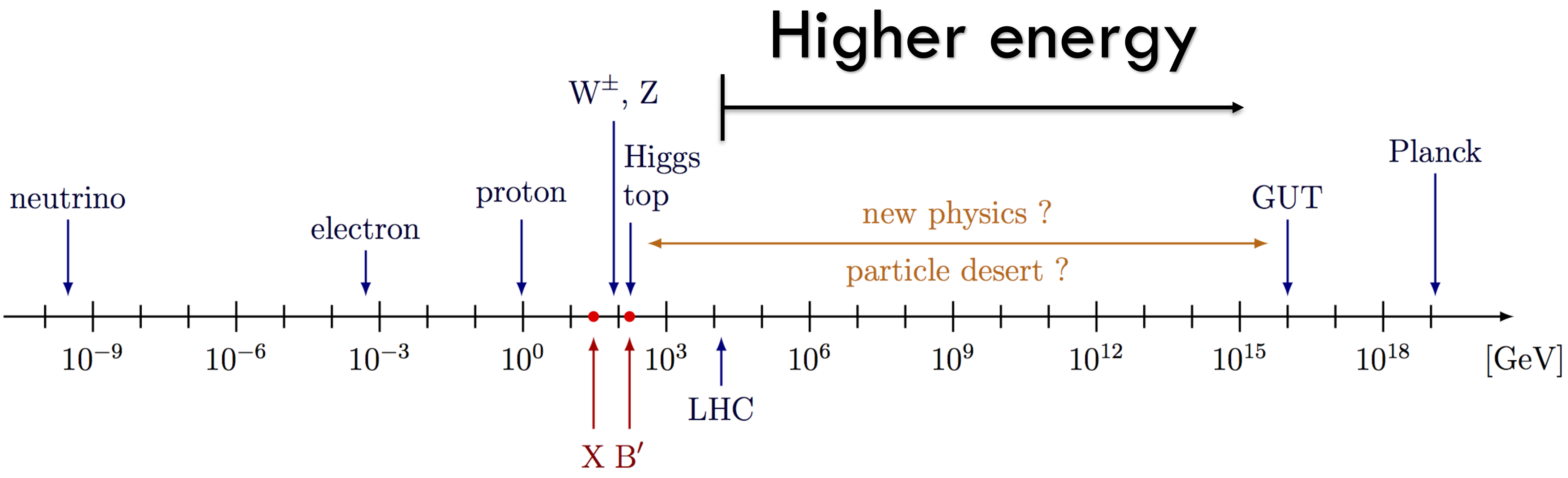
Machine Learning for Particle Tracking

ExaTrkX @ Berkeley Lab

Paolo Calafiura (PI), Nicholas Choma,
Steve Farrell, Xiangyang Ju,
Daniel Murnane (*ExaTrkX*)
Zachary Marshall (*ATLAS*)

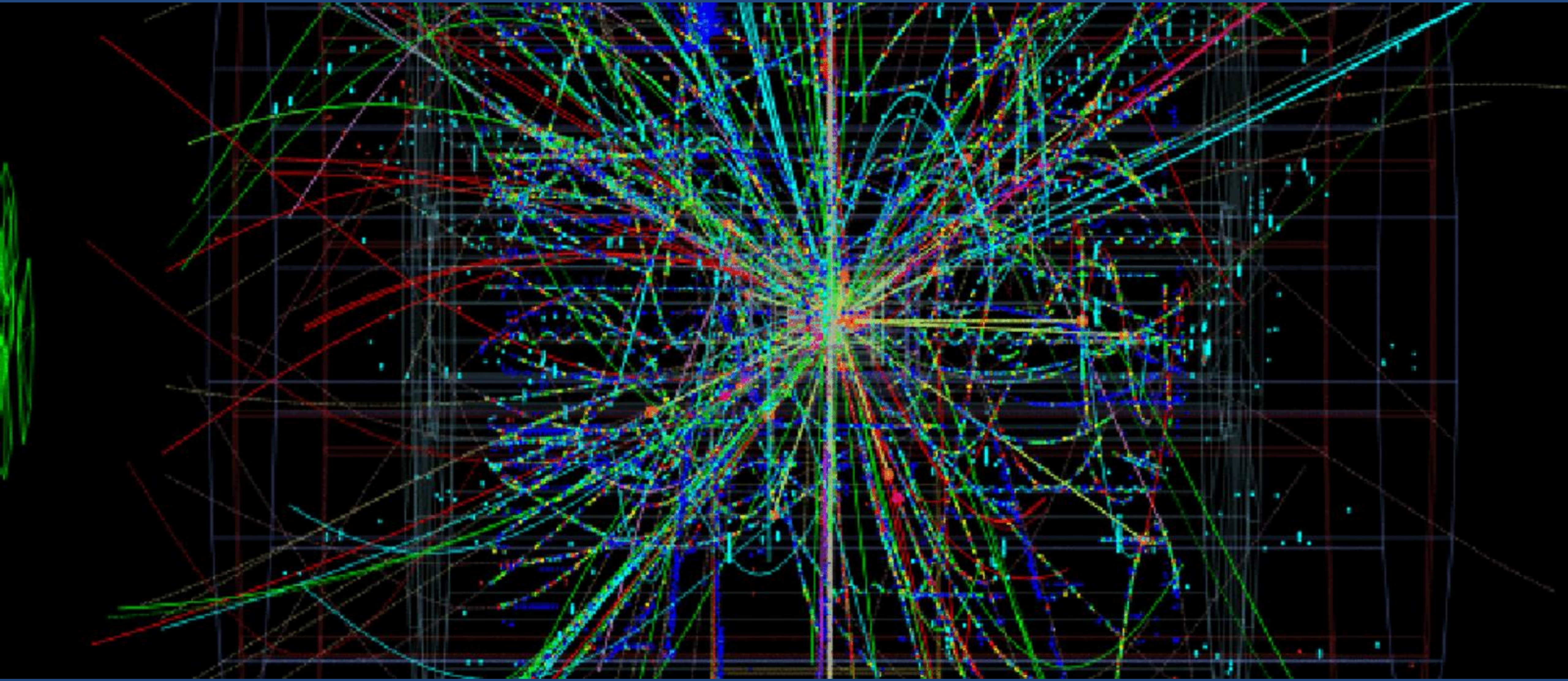
WHY COLLIDERS? New physics lives at higher energies



Discovering new physics is getting harder and harder

1. How do we discover new physics?
2. The “tracking problem” of particle physics
3. Tracking is hard, and getting harder
4. Graphs are a natural representation of tracks
5. GNNs and other ML approaches to tracking
6. The road to fully learned tracks

The usual story: Smashing atoms into millions of pieces is useful



New physics needs collisions…

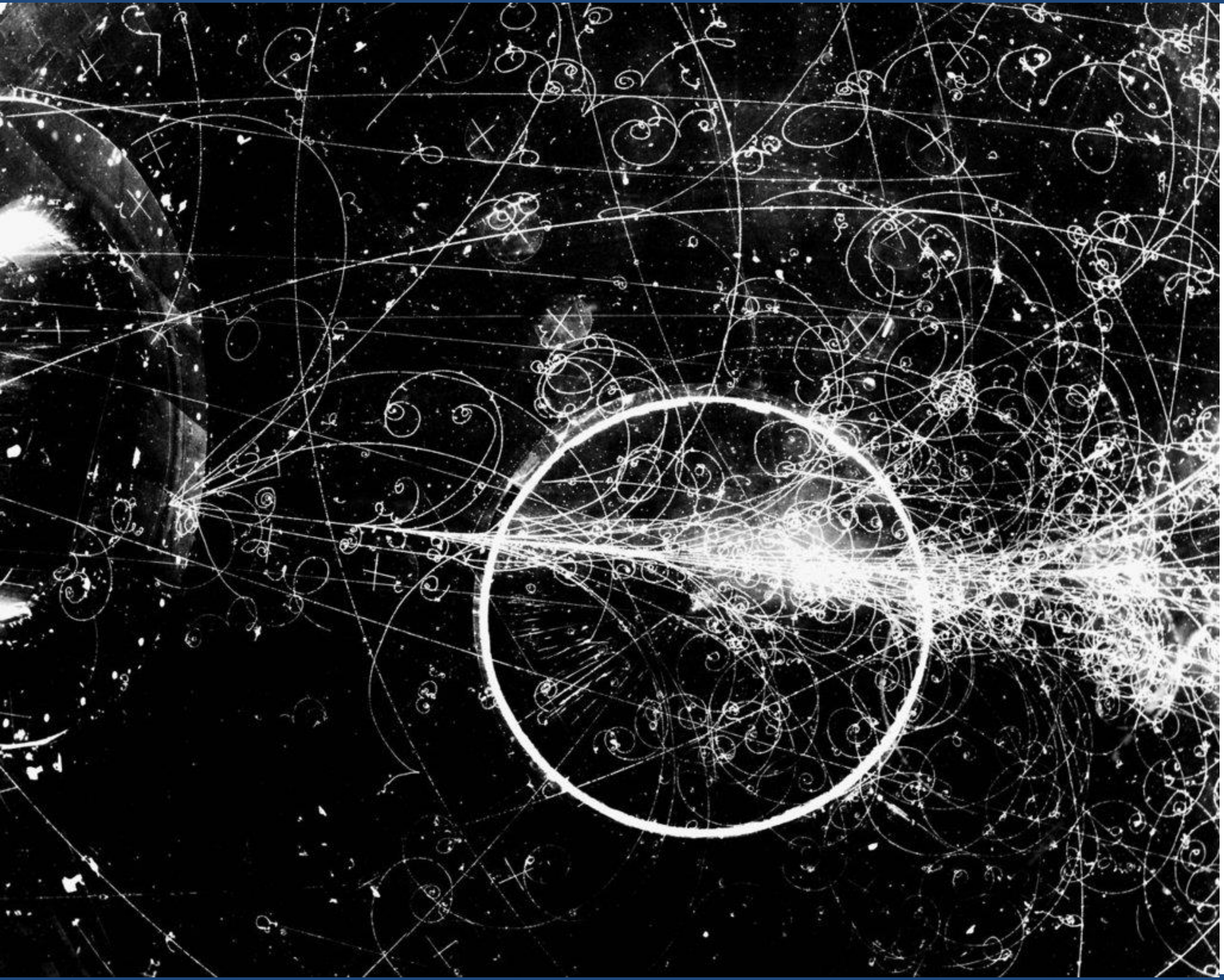
- Higgs boson (LHC),
- Quarks (SLAC, Fermilab), and
- Neutrino mass (Super-Kamiokande)

Discovered with collisions

- Supersymmetry,
- Composite Higgs,
- Dark matter,
- Leptoquarks,
- W/Z prime, and
- Axions

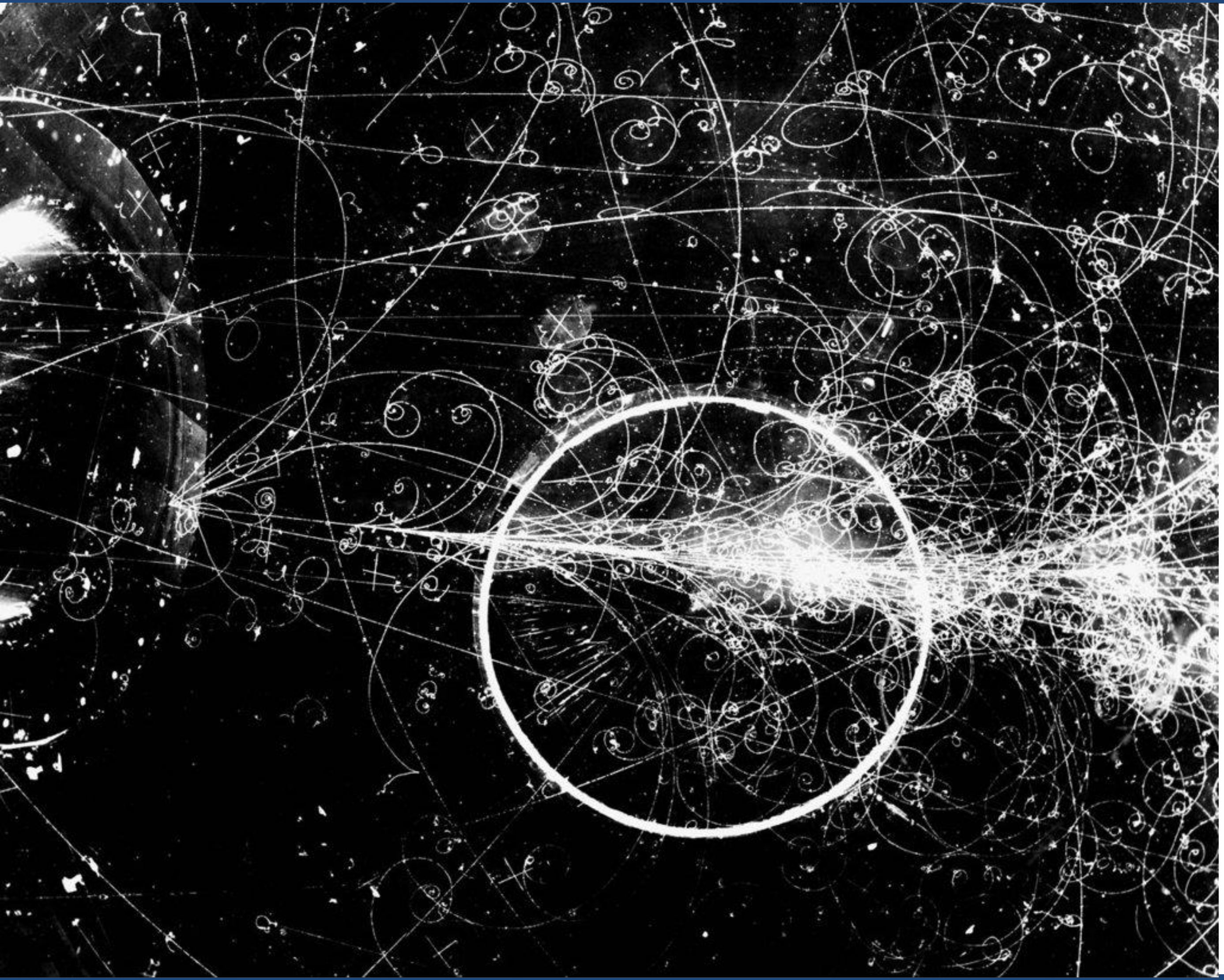
Could be discovered with collisions

… but collisions are messy



- High energy collisions bring huge numbers of particles (unfortunately)
- Want to see the particles coming out of the collisions, which we can get from the curves (“tracks”) moving through a magnetic field

… but collisions are messy



- High energy collisions bring huge numbers of particles (unfortunately)
- Want to see the particles coming out of the collisions, which we can get from the curves (“tracks”) moving through a magnetic field
- Why not just watch the particles curving directly?
- Every observation/measurement affects particle track
- We need to observe the tracks as little as possible


Imagine solving a
jigsaw puzzle

(with your eyes closed)

And every time…

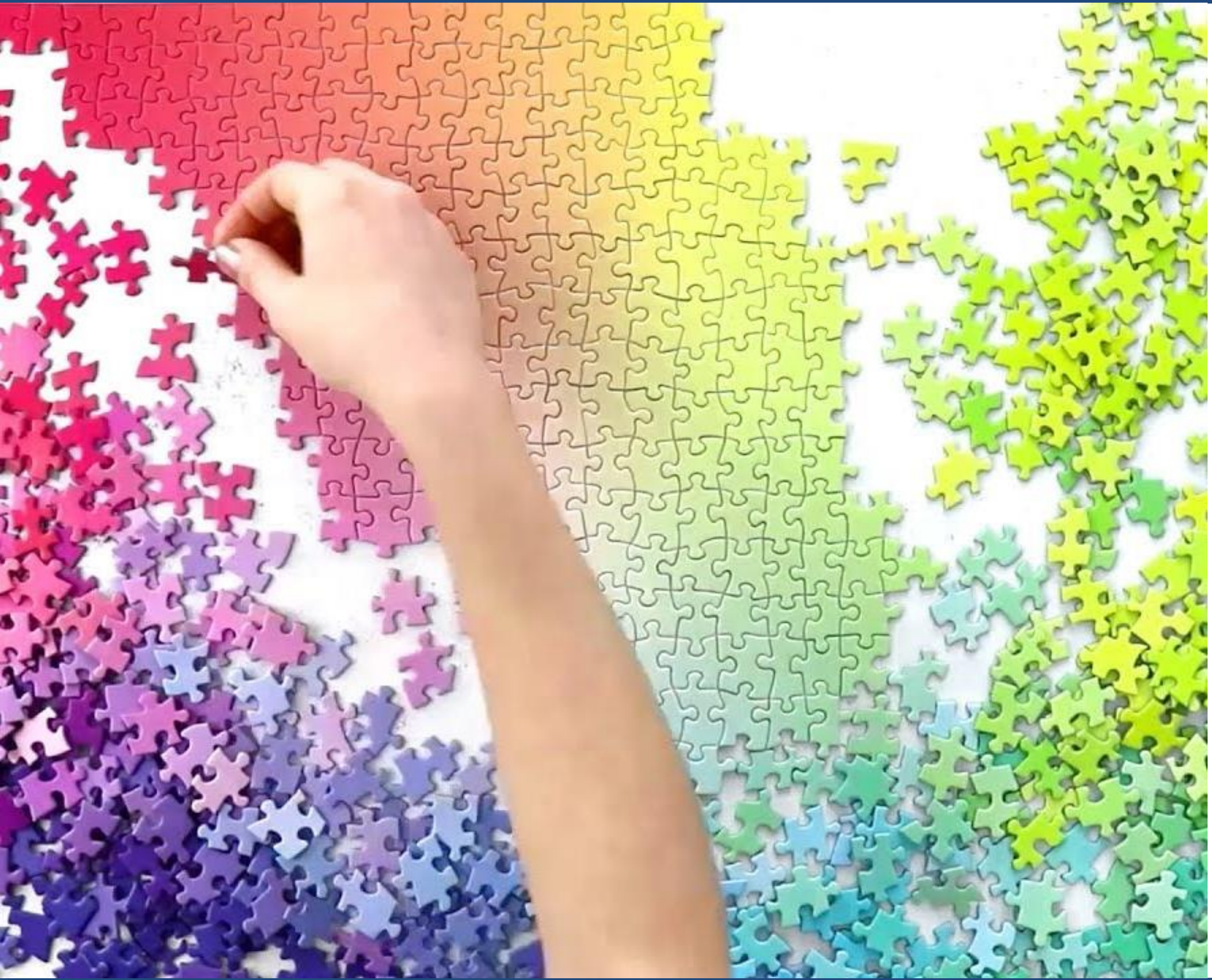

you peak…

the puzzle becomes…


more...

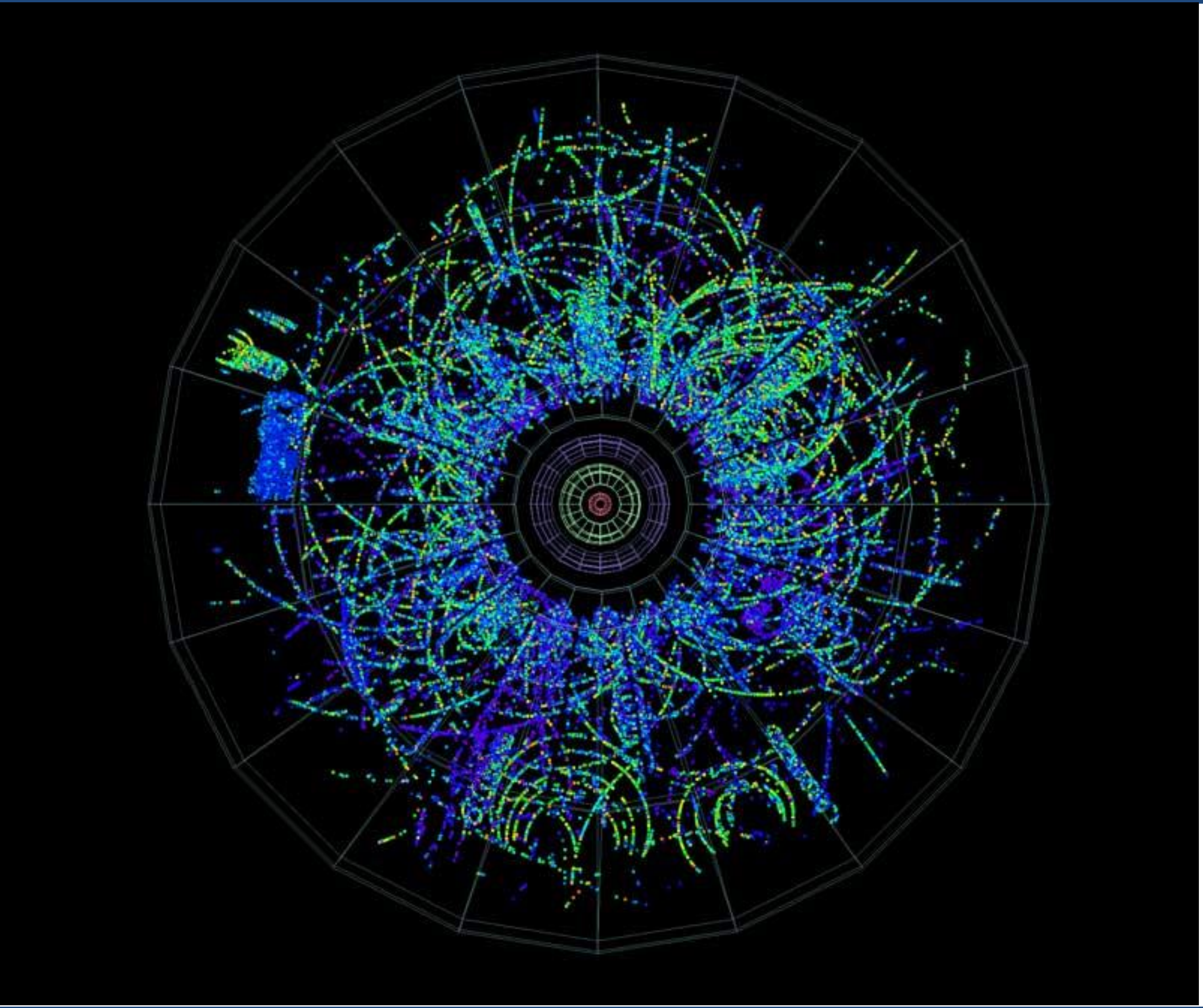

and more...

…complicated



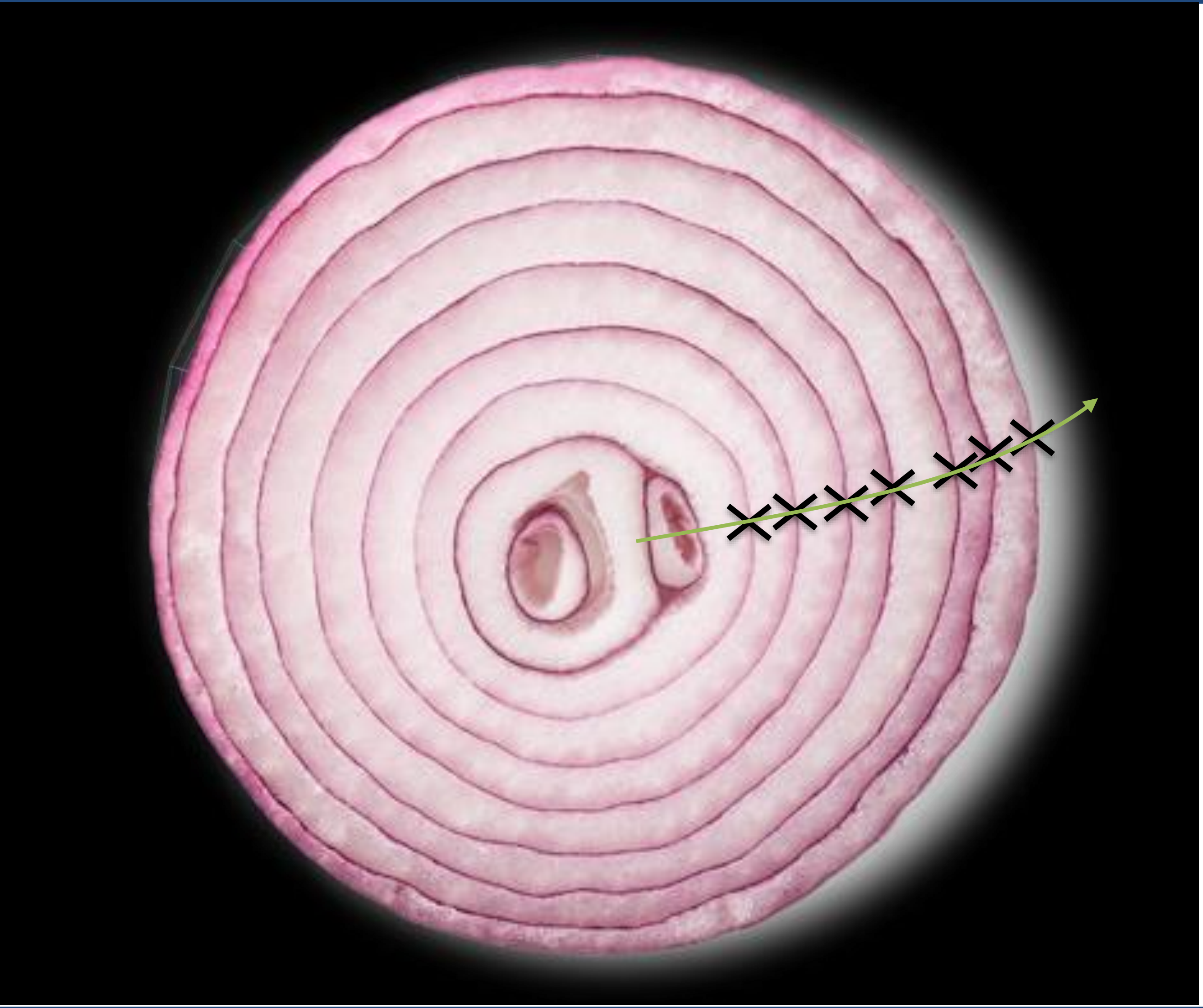
- New physics requires high energy and high precision
- This implies carefully tracking millions of particles per event through the (as-few-as-possible) layers of a detector
- Each collision comprises of dozens of events
- Each second produces tens of millions of collisions

The jigsaw leads to the "Tracking Problem" of new physics



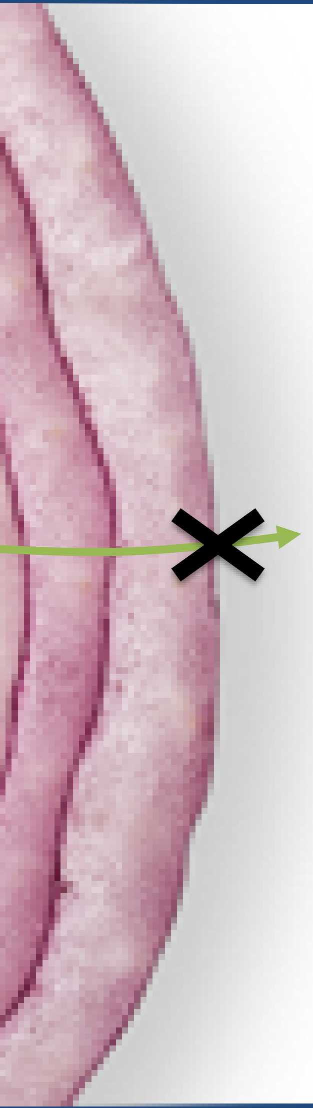
- New physics requires high energy and high precision
- This implies carefully tracking millions of particles per event through the (as-few-as-possible) layers of a detector
- Each collision comprises of dozens of events
- Each second produces tens of millions of collisions

The jigsaw leads to the "Tracking Problem" of new physics



- New physics requires high energy and high precision
- This implies carefully tracking millions of particles per event **through the (as-few-as-possible) layers of a detector**
- Each collision comprises of dozens of events
- Each second produces tens of millions of collisions

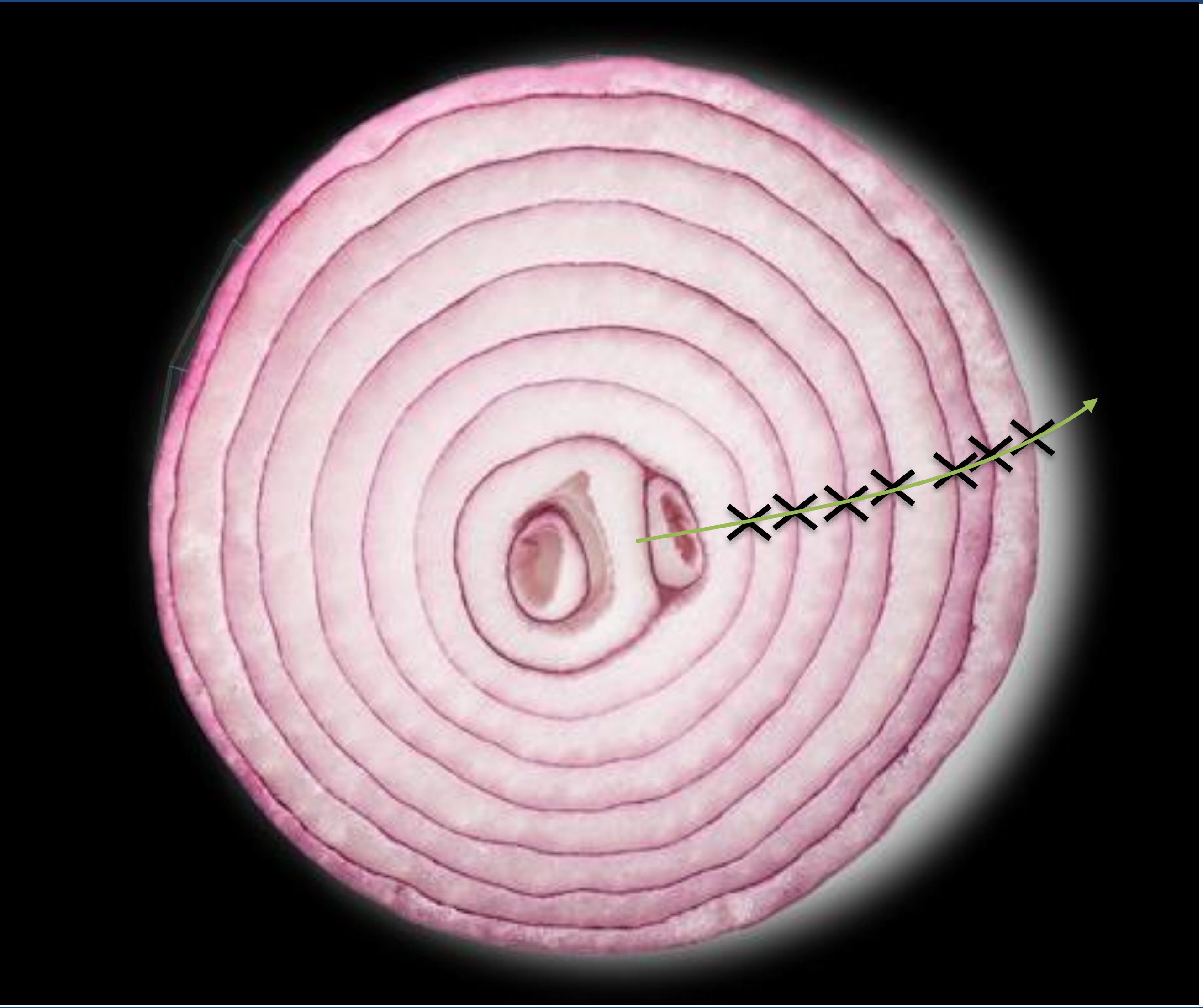
The jigsaw leads to the “Tracking Problem” of new physics



A particle interacting with a layer is a “hit”

- New physics requires high energy and high precision
- This implies carefully tracking millions of particles per event **through the (as-few-as-possible) layers of a detector**
- Each collision comprises of dozens of events
- Each second produces tens of millions of collisions

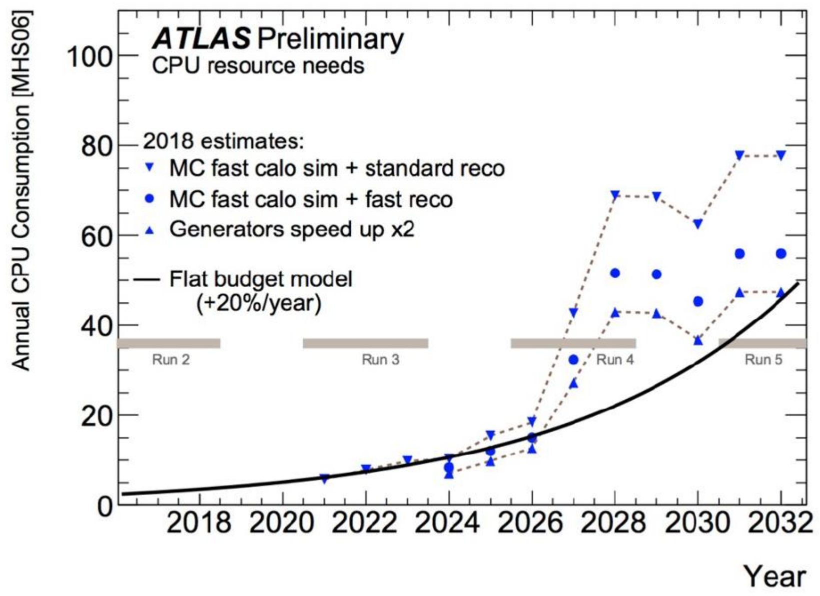
The jigsaw leads to the “Tracking Problem” of new physics



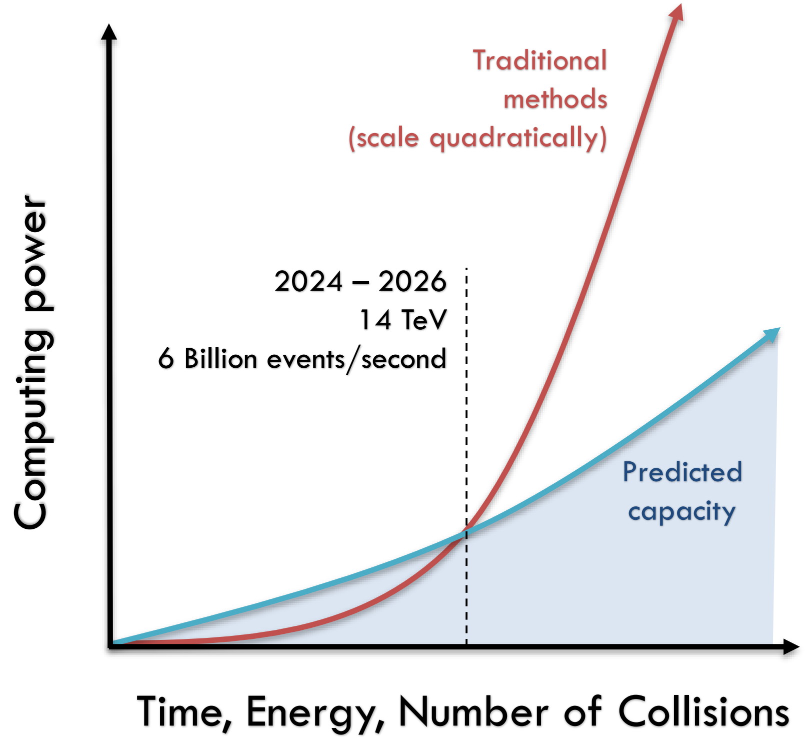
We need a fast, high-accuracy method to connect hits into tracks to determine the types and energies of particles coming out of every event

Current techniques will* not work on next-gen colliders

Standard doom-and-gloom plot



In other words…



*probably

Collider physics has 4 steps

1. Observe hits on layers
2. Join hits into track
3. Convert track into particle information
4. (Dis)prove supersymmetry

We want to build tracks

1. Observe hits on layers

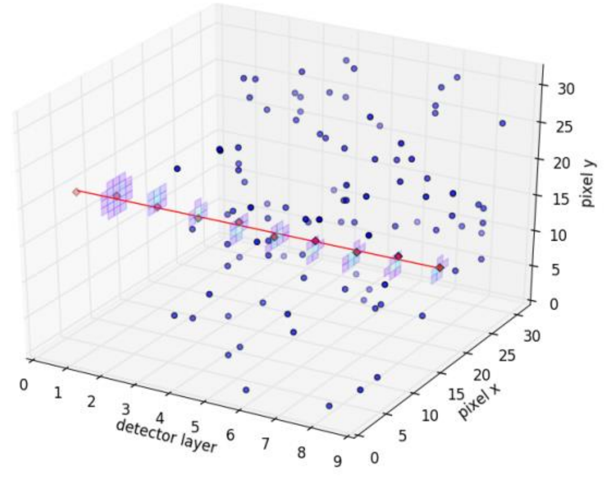
2. Join hits into track

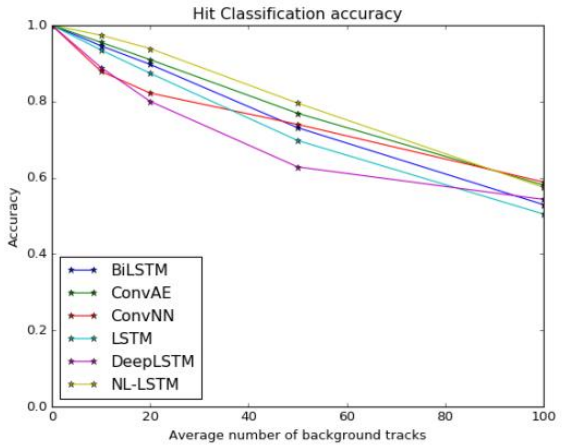
This is our focus

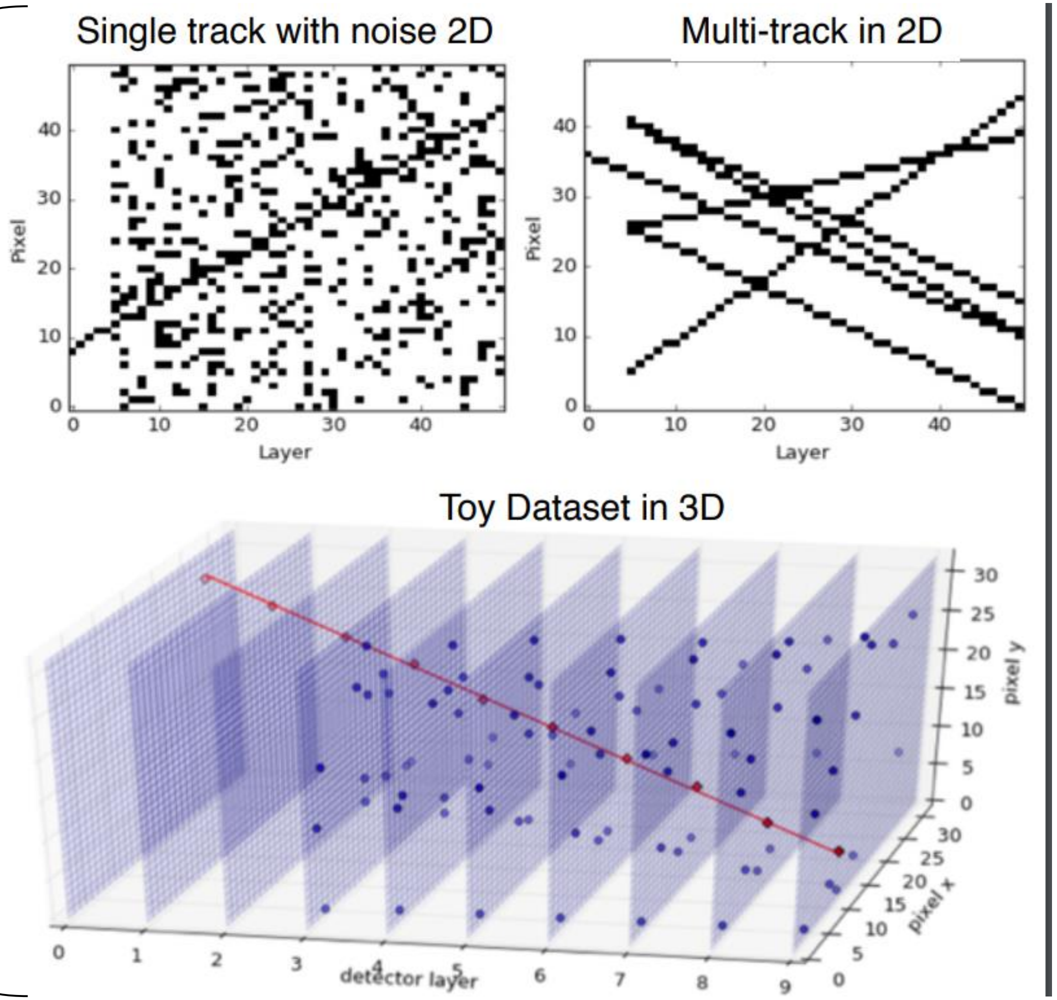
3. Convert track into particle information
4. (Dis)prove supersymmetry

Previous ML Approaches

- Tracks as **images (CNN)**
- Tracks as **sequences of points (LSTM)**



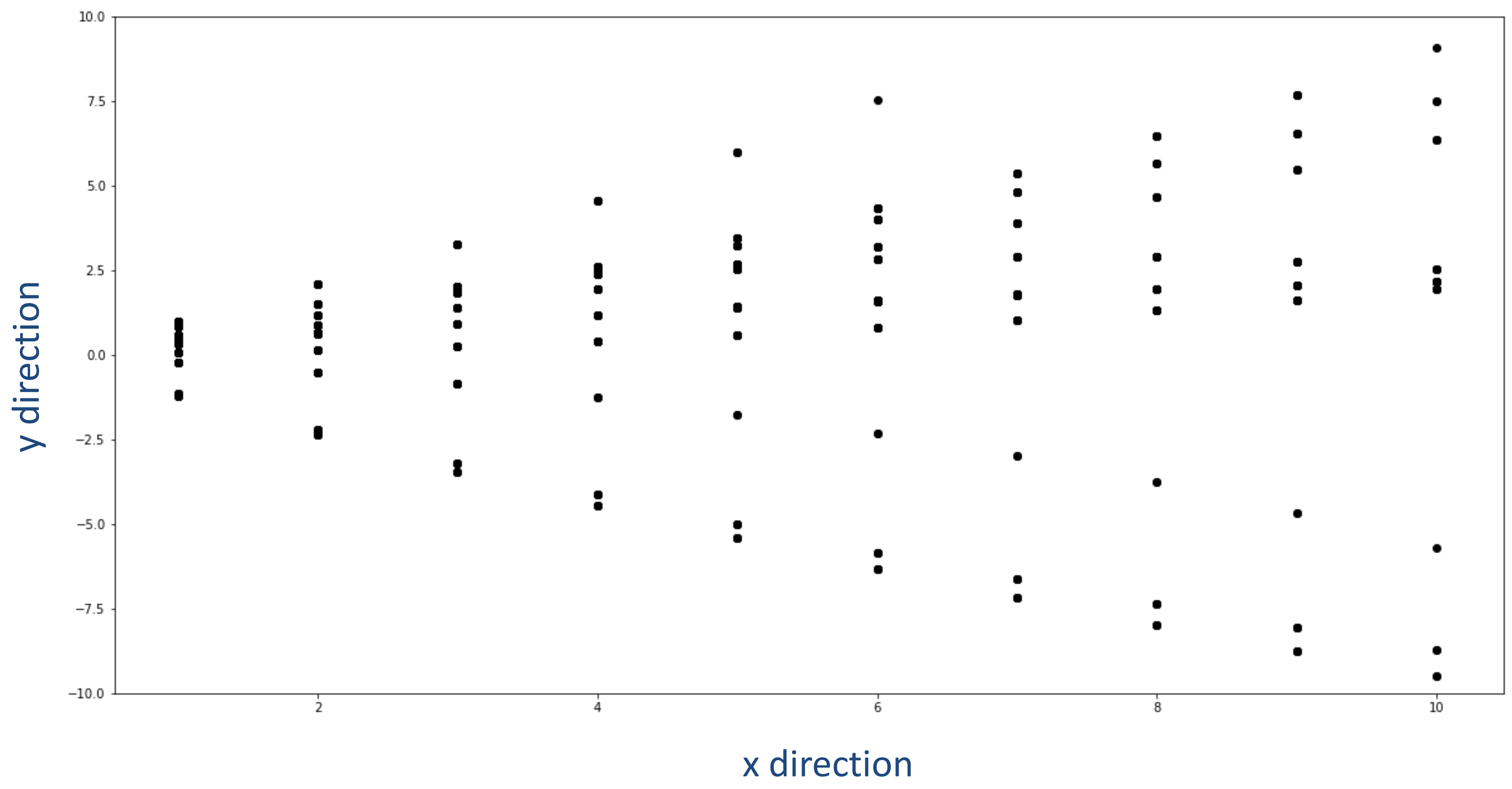




Graphs are a natural way to represent tracks

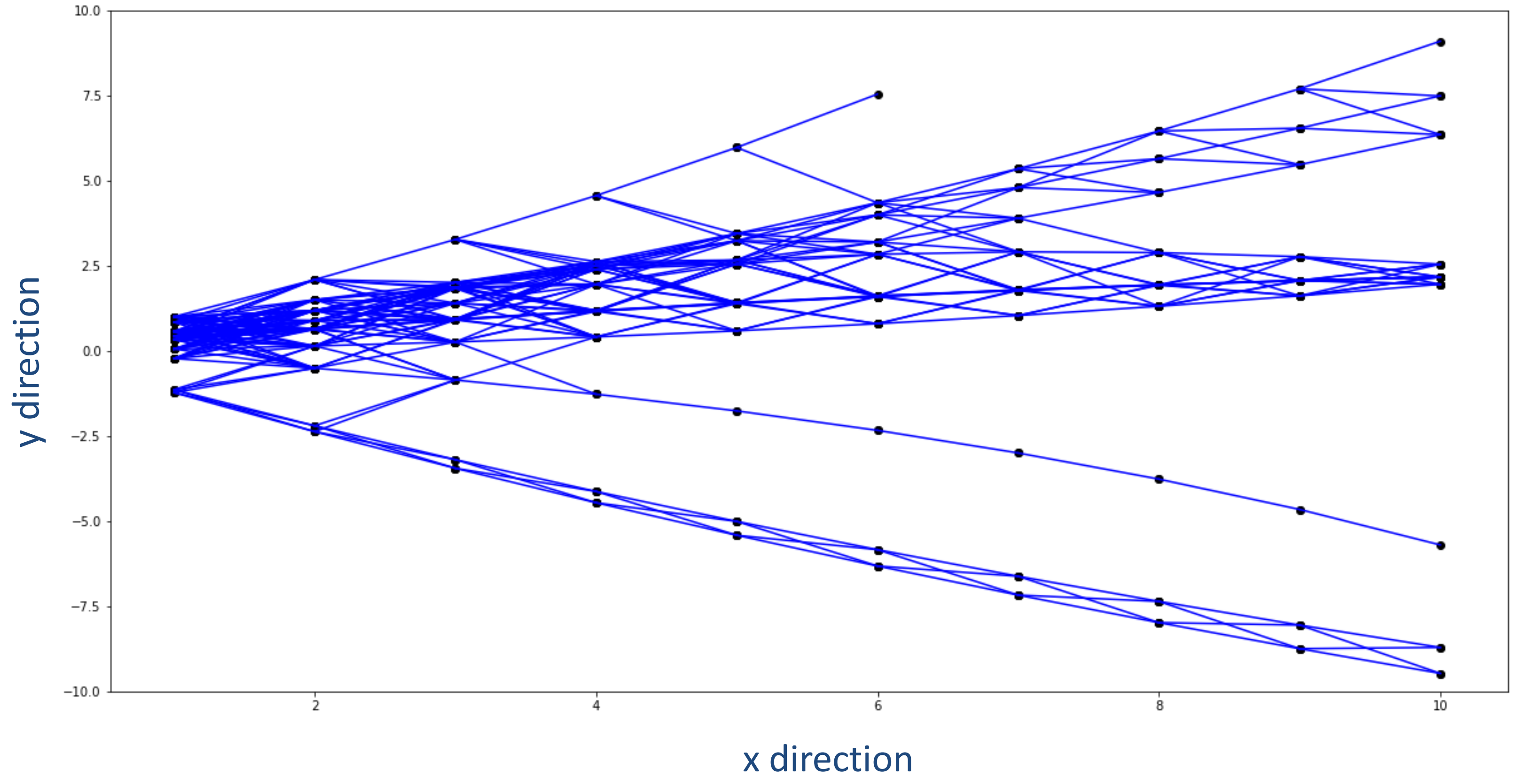
- We have a collection of hits
- Want to “Connect the Dots”
- A natural way is to represent the problem is as a graph

Graphs are a natural way to represent tracks



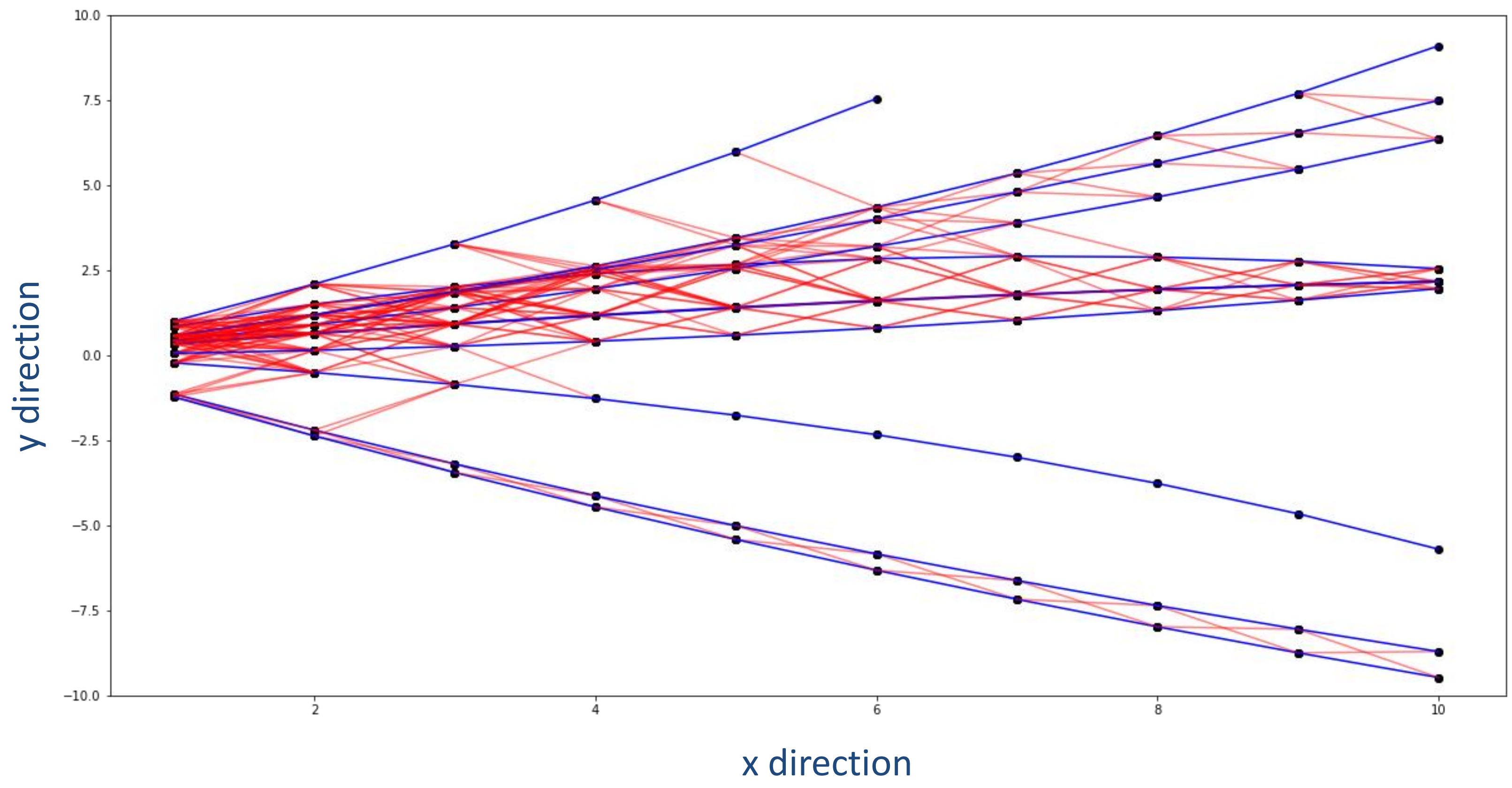
Some toy data…

Graphs are a natural way to represent tracks



Join the hits in some clever/dumb way…

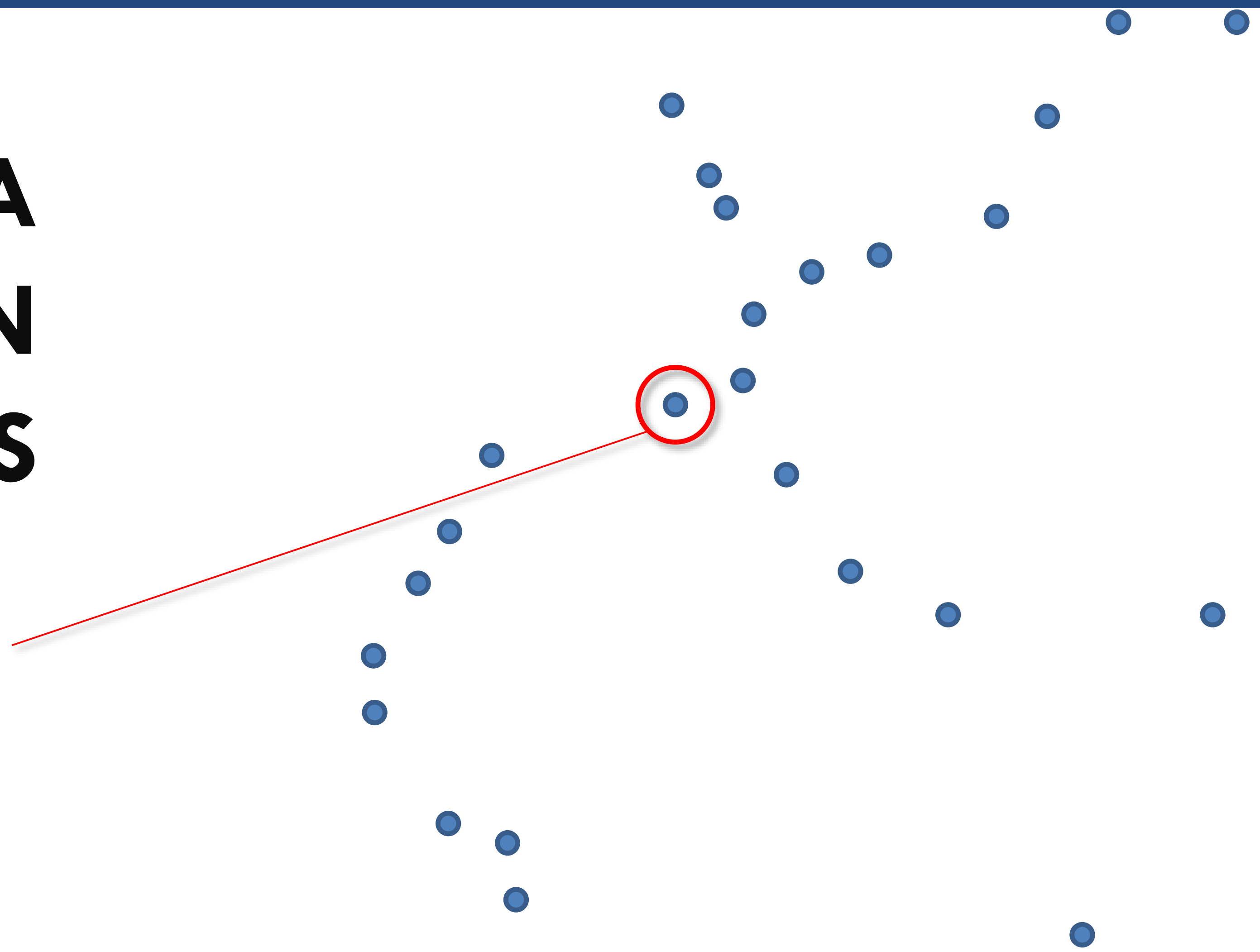
Graphs are a natural way to represent tracks



The tracks should be in here

What is a graph?

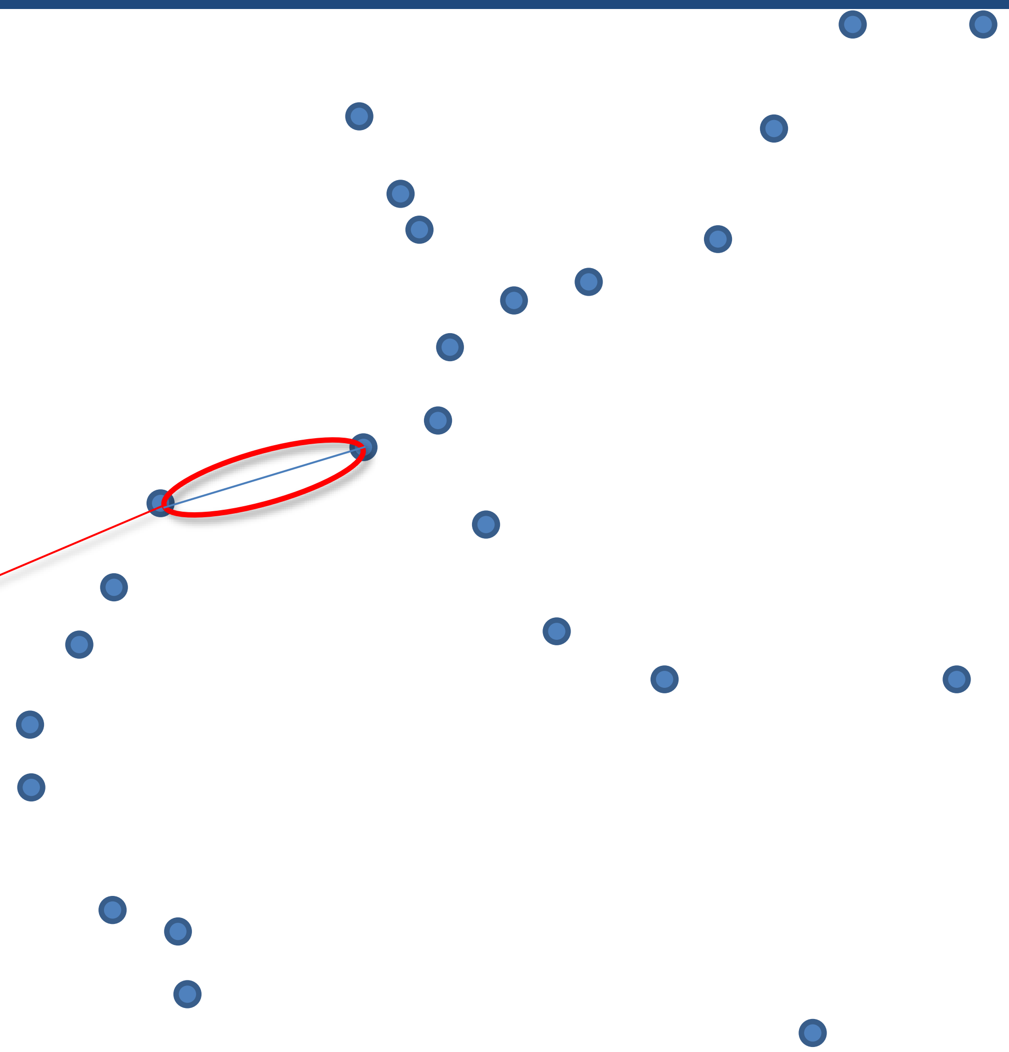
A COLLECTION OF NODES



NODE

What is a graph?

AND EDGES

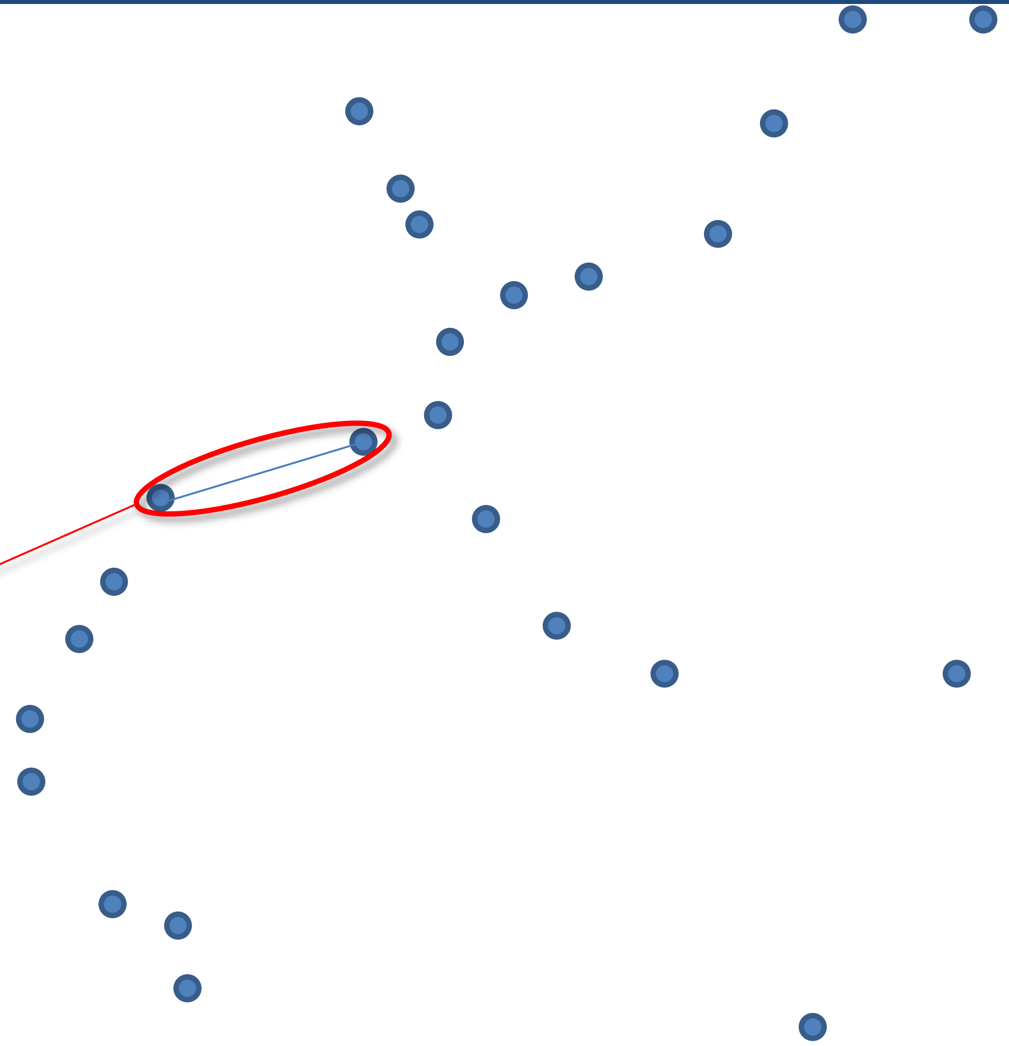


EDGE

What is a graph?

NODES + EDGES = DOUBLETS

DOUBLET



What is a graph?

NODES CAN HAVE FEATURES

NODE FEATURE
e.g. “West Oakland”

What is a graph?

EDGES CAN HAVE FEATURES

EDGE FEATURE
e.g. “Under Maintenance – Single Track”

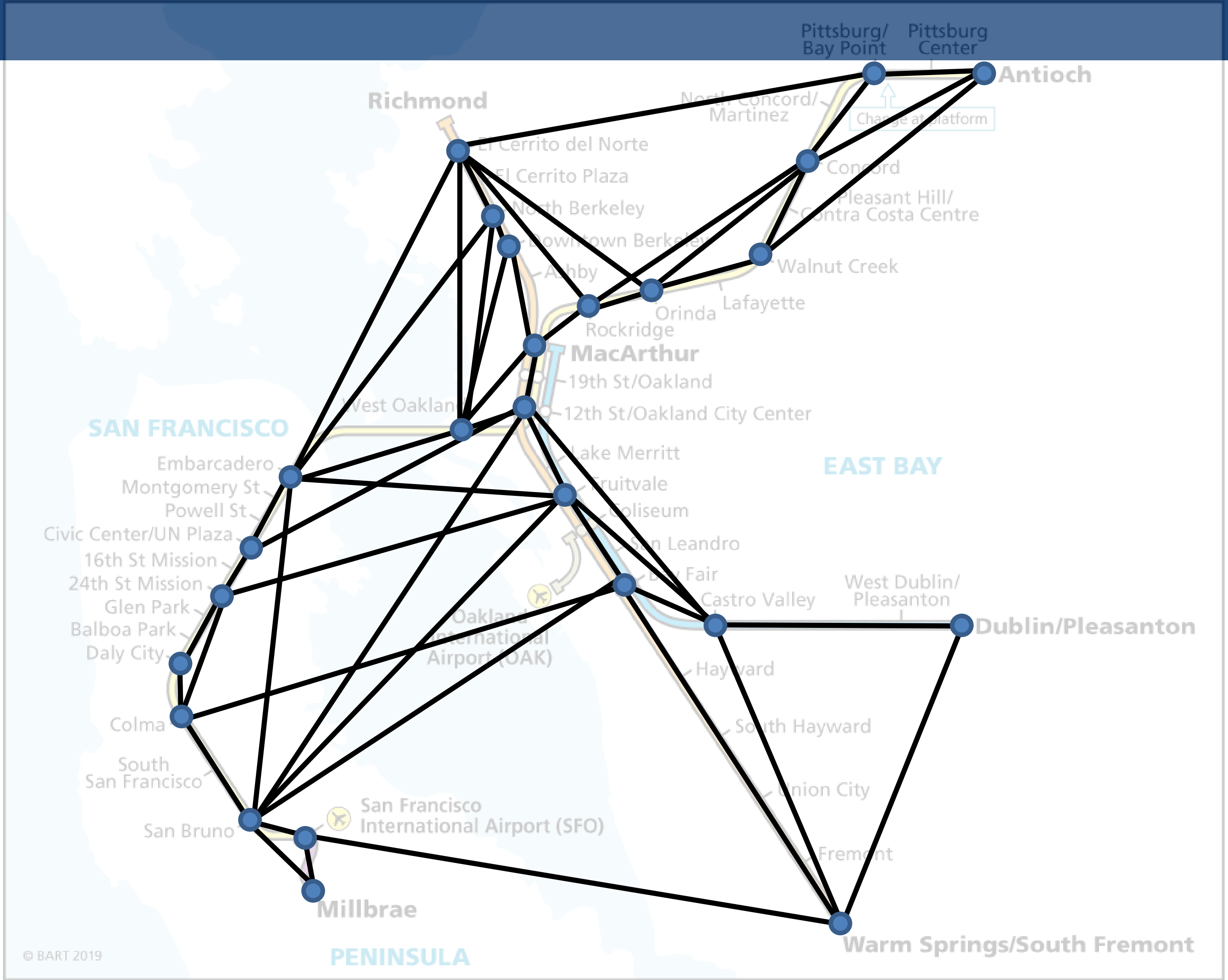
What is a graph?

THE WHOLE GRAPH CAN HAVE FEATURES

GRAPH FEATURE
e.g. “Sunday Timetable”

The tracking problem can be considered edge classification

Join the nodes in some dumb/clever way…

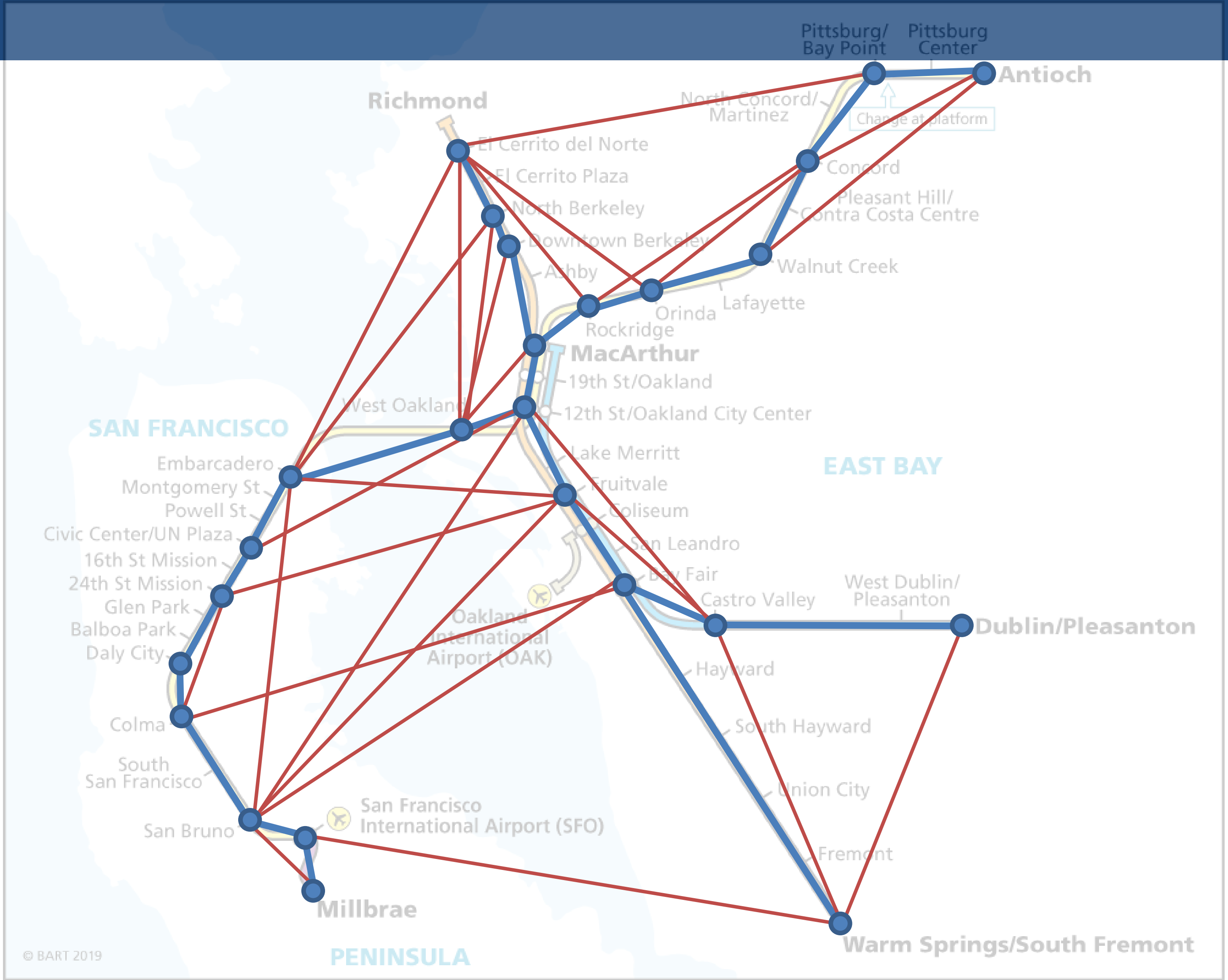


The tracking problem can be considered edge classification

Classify edges with score between [0,1]

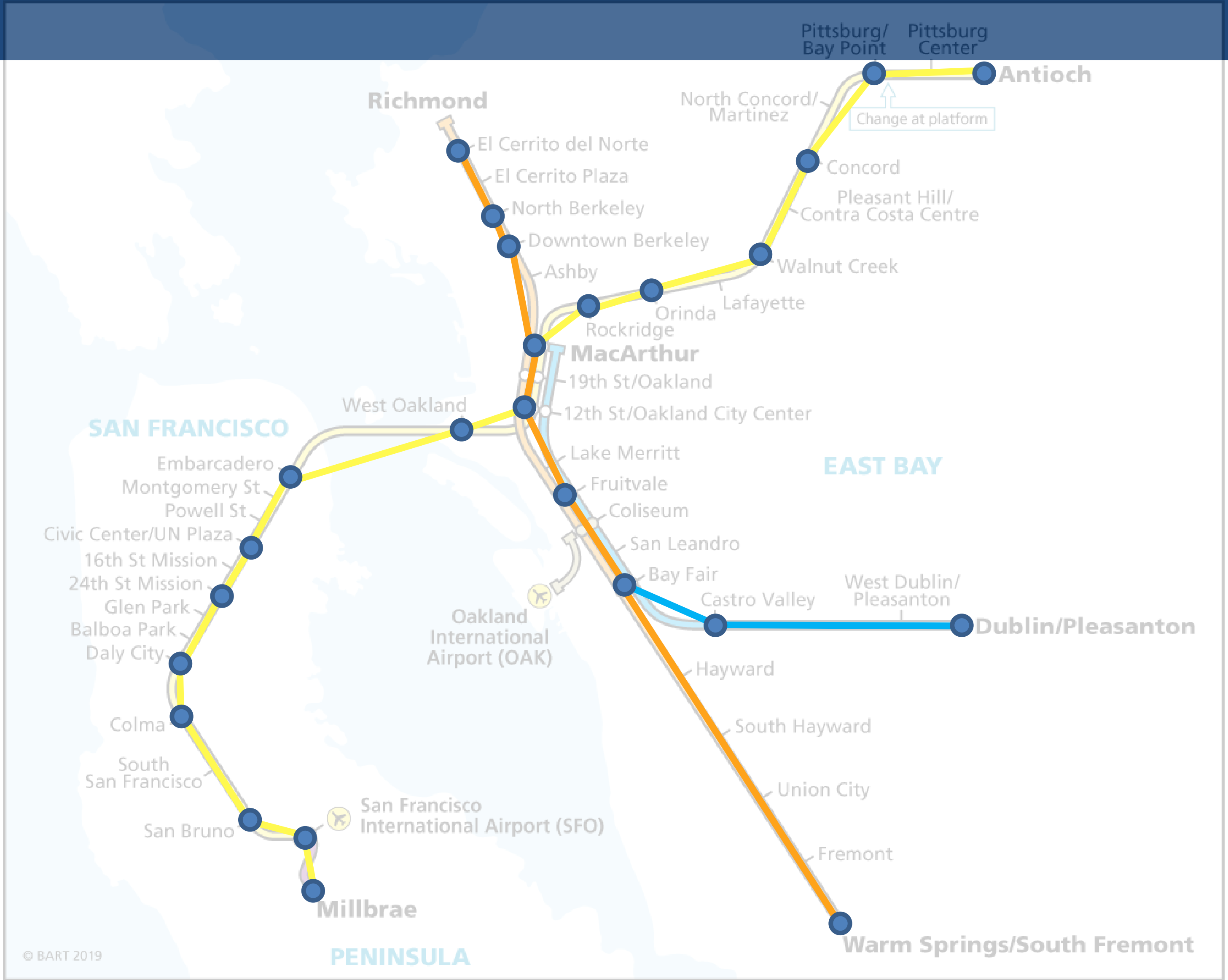
score > cut: true

score < cut: fake



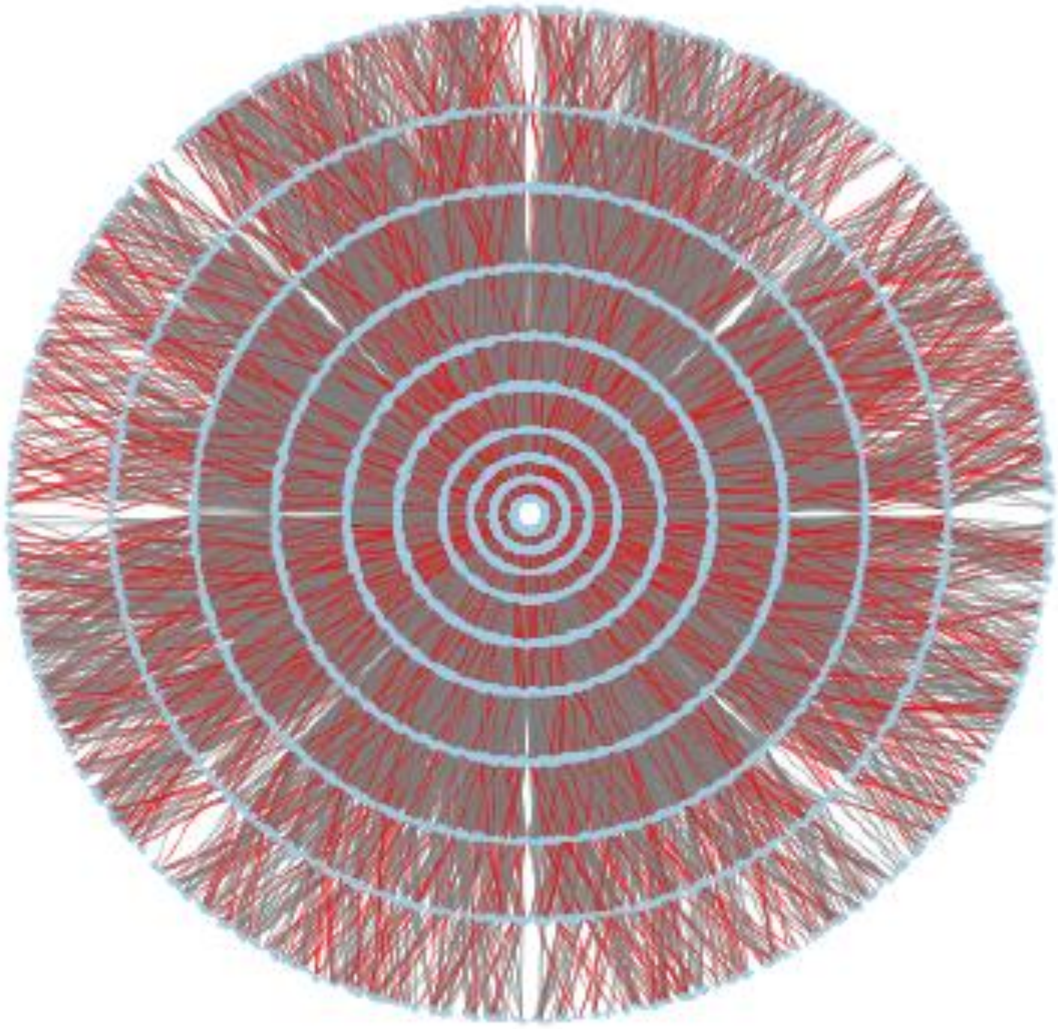
The tracking problem can be considered edge classification

Ultimate goal is to connect doublets into tracks



Realistic tracking is complicated

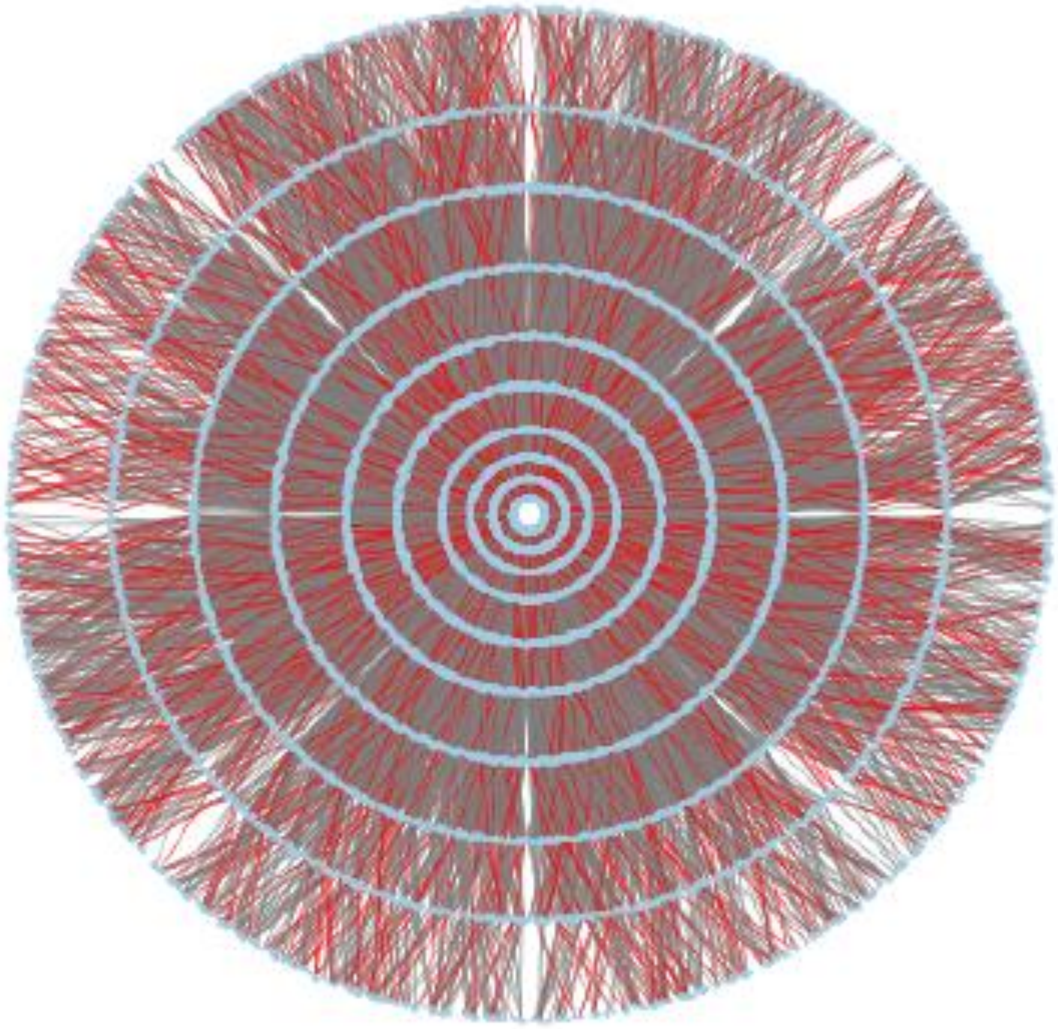
Ultimate goal is to connect doublets into tracks



Realistic tracking is complicated

Dataset

- "TrackML Kaggle Competition" dataset
- Generated by simulation
- 8000 collisions to train on
- Each collision has up to 100,000 hits of around 10,000 particles



Enter the Graph Neural Network

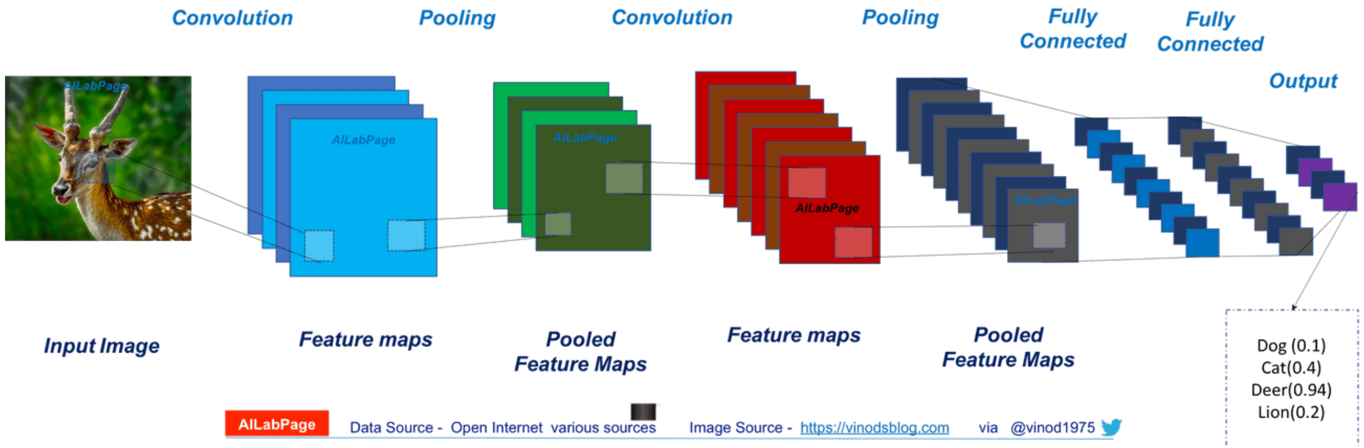
1. Takes graph features (node-level, edge-level and graph-level)
2. Performs transformations on those features
3. Runs through a neural net
4. Returns graph predictions (node-level, edge-level or graph-level)

Is a *generalisation* of Convolutional Neural Net (aka deep learning)

Convolutional neural nets in 13 seconds

- Simply connecting every piece of information in a big equation does not always produce good predictions
- "Convolving" (i.e. combining complex chunks of information into simpler chunks that can be trained upon) can reveal "high-level features"

DNN
with
CNNs



A GNN generalises a CNN

- Convolutions with matrices really just connect neighbours in 2-D space
- A GNN connects neighbours in N-D
- Not necessarily flat – the geometry is determined by edge and node features, and edges between nodes

Passing information around the graph gives it learning power

- Can make a node “aware” of its neighbours by concatenating the neighbouring hidden features
- Iterating this neighbourhood learning passes information around the graph

Passing information around the graph gives it learning power

Message passing

+

Attention mechanism

=

Excellent prediction performance

AI/ML usage in ExaTrkX @ Berkeley – The Lay of the Land

- **Doublet classification** (Steve & Xiangyang): MPNN, AGNN
- **Triplet classification** (Daniel): Concatenated AGNN
- **Doublet classification for building** (Nick): Embedded space + Doublet MLP
- **End-to-End Track Classification** (Nick): Embedded Clustering + GNN
- **Doublet classification** (Xiangyang): Layer-pair MLPs
- **Distributed training** (Steve)
- **Architecture exploration & Node regression** (Daniel): Other GNN convolutions and aggregations, track parameter regression

AI/ML usage in ExaTrkX @ Berkeley – The Lay of the Land

- **Doublet classification** (Steve & Xiangyang): MPNN, AGNN
- **Triplet classification** (Daniel): Concatenated AGNN
- **Doublet classification for seeding** (Nick): Embedded Clustering + Doublet MLP

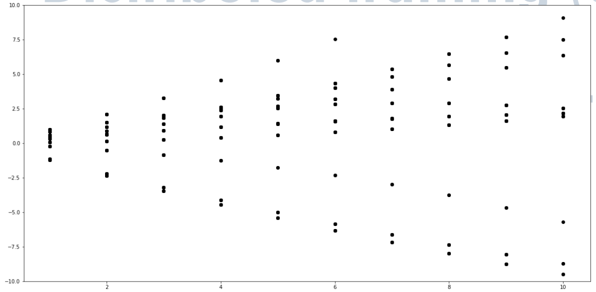
Will cover in next few slides

- **End-to-End Track Classification** (Nick): Embedded Clustering + GNN
- **Doublet classification** (Xiangyang): Layer-pair MLPs
- **Distributed training** (Steve)
- **Architecture exploration & Node regression** (Daniel): Other GNN convolutions and aggregations

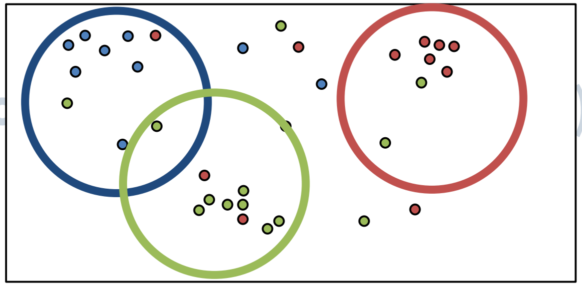
AI/ML usage in ExaTrkX @ Berkeley – The Lay of the Land

- **Doublet classification** (Steve & Xiangyang): MPNN, AGNN
- **Triplet classification** (Daniel): Concatenated AGNN
- **Doublet classification for building** (Nick): Embedded space + Doublet MLP
- **End-to-End Track Classification** (Nick): Embedded Clustering + GNN
- **Doublet classification** (Xiangyang): Layer-pair MLPs
- **Distributed training** (Steve)
- **...ion & No...**...): Other GN...

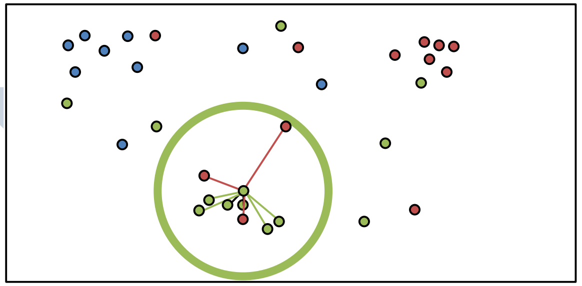
3D space



Latent space

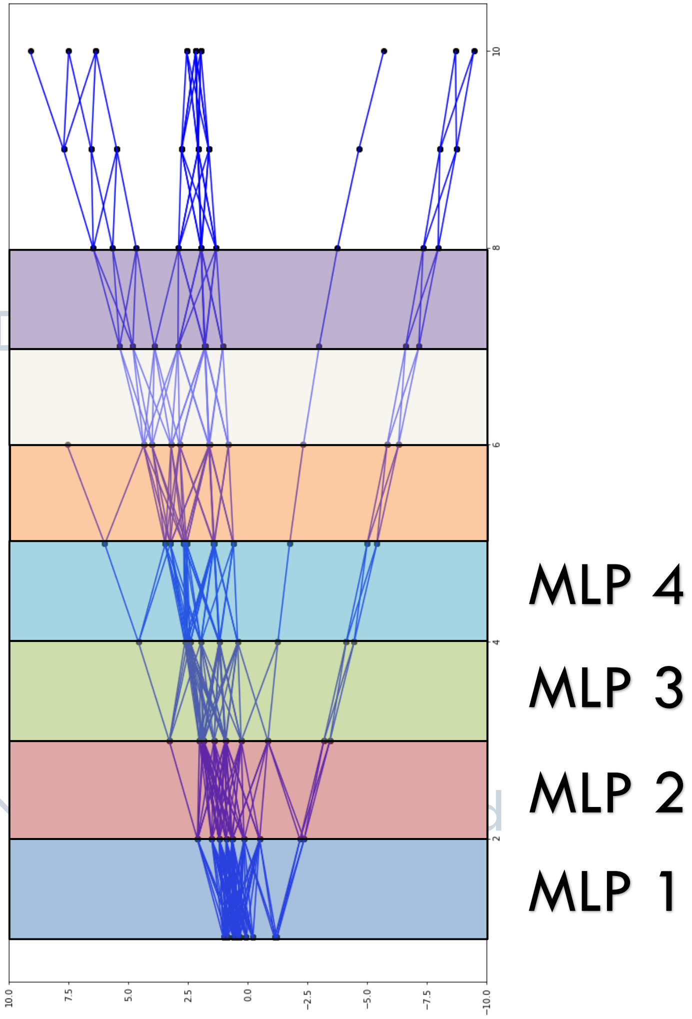


MLP in ball



AI/ML usage in ExaTrkX @ Berkeley – The Lay of the Land

- **Doublet classification** (Steve & Xiangyang): MPNN, AGNN
- **Triplet classification** (Daniel): Concatenated AGNN
- **Doublet classification for building** (Nick): Embedded space + D
- **End-to-End Track Classification** (Nick): Embedded Clustering +
- **Doublet classification** (Xiangyang): Layer-pair MLPs
- **Distributed training** (Steve)
- **Architecture exploration & Node regression** (Daniel): Other GN d aggregations



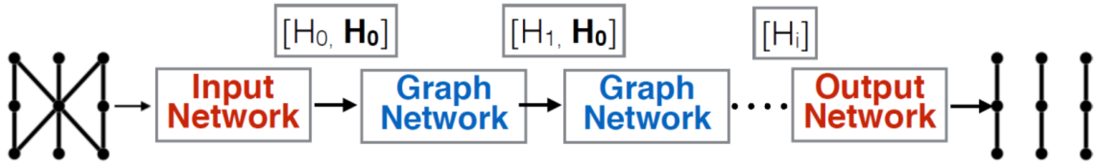
AI/ML usage in ExaTrkX @ Berkeley – The Lay of the Land

- **Doublet classification** (Steve & Xiangyang): MPNN, AGNN
- **Triplet classification** (Daniel): Concatenated AGNN
- **Doublet classification for building** (Nick): Embedded space + Doublet MLP
- **End-to-End Track Classification** (Nick): Embedded Clustering + GNN
- **Doublet classification** (Xiangyang): Layer-pair MLPs

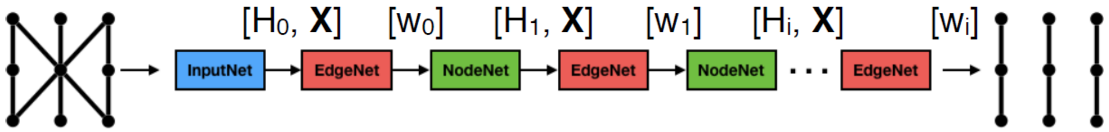
Self-explanatory?

- **Distributed training** (Steve)
- **Architecture exploration & Node regression** (Daniel): Other GNN convolutions and aggregations, track parameter regression

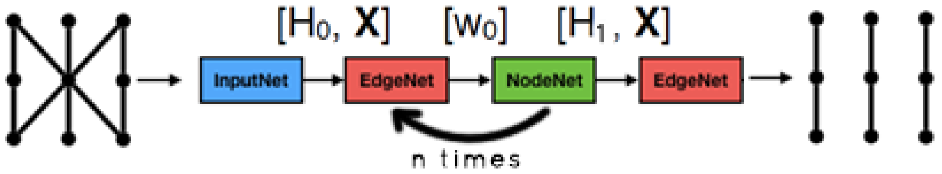
GNN Edge prediction architecture



- Message Passing



- Attention Message Passing

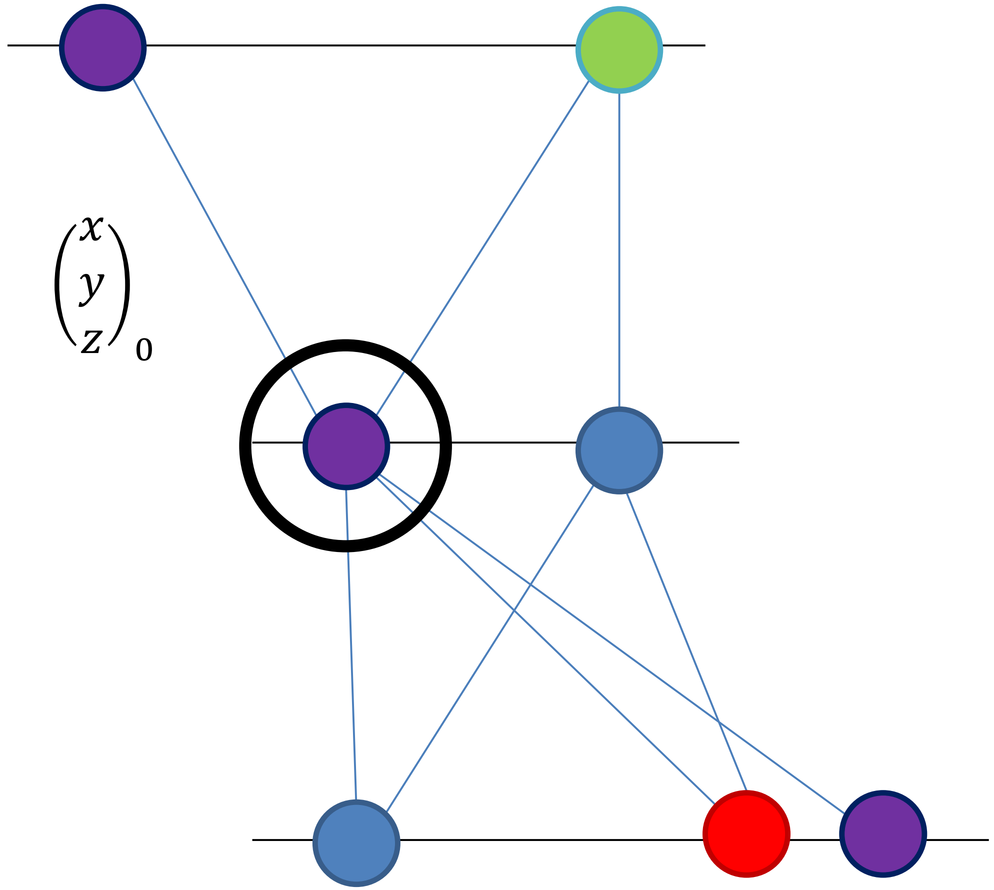


- Attention Message Passing with Recursion

Edge prediction architecture

- **Input node features**
- Hidden node features
- Hidden edge features
- Edge score
- Attention aggregation
- New hidden node features
- New hidden edge features
- New edge score

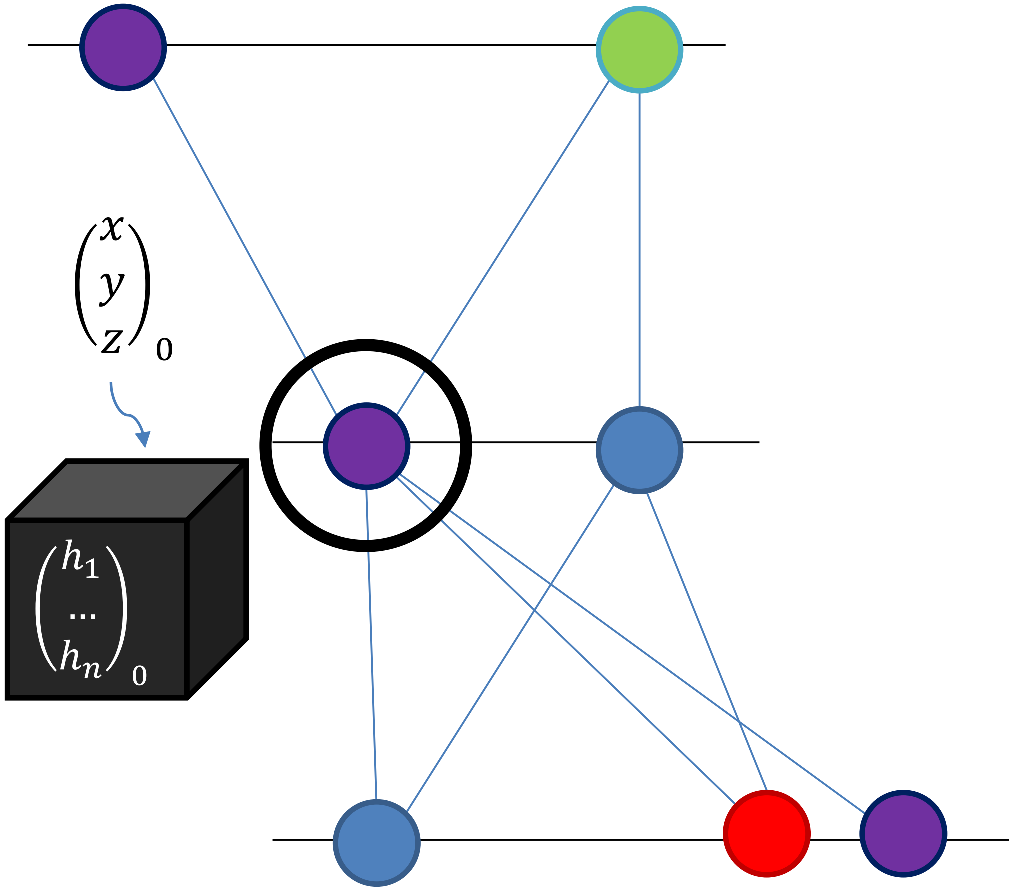
x n iterations
(hyperparameter)



Edge prediction architecture

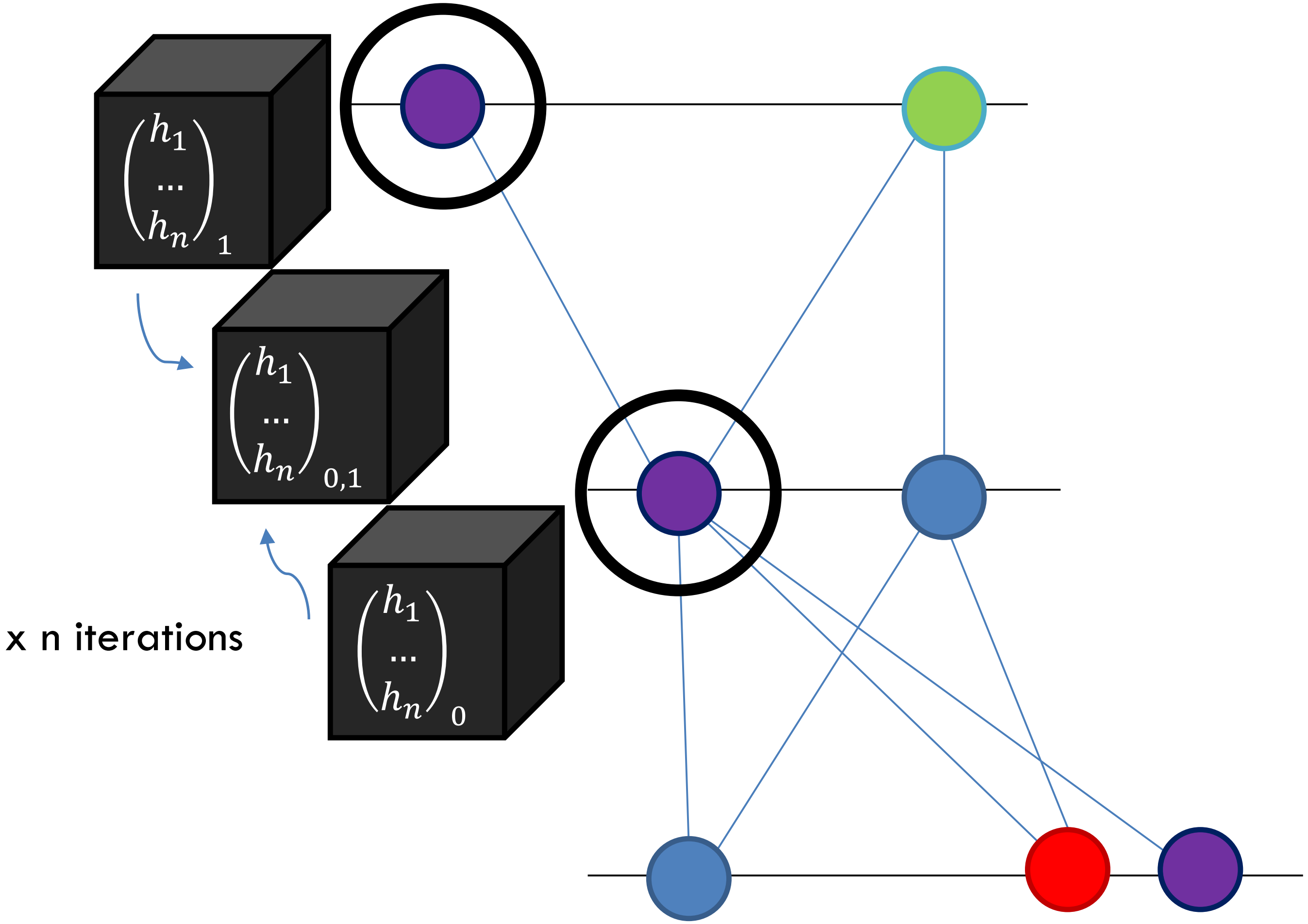
- Input node features
- **Hidden node features**
- Hidden edge features
- Edge score
- Attention aggregation
- New hidden node features
- New hidden edge features
- New edge score

x n iterations



Edge prediction architecture

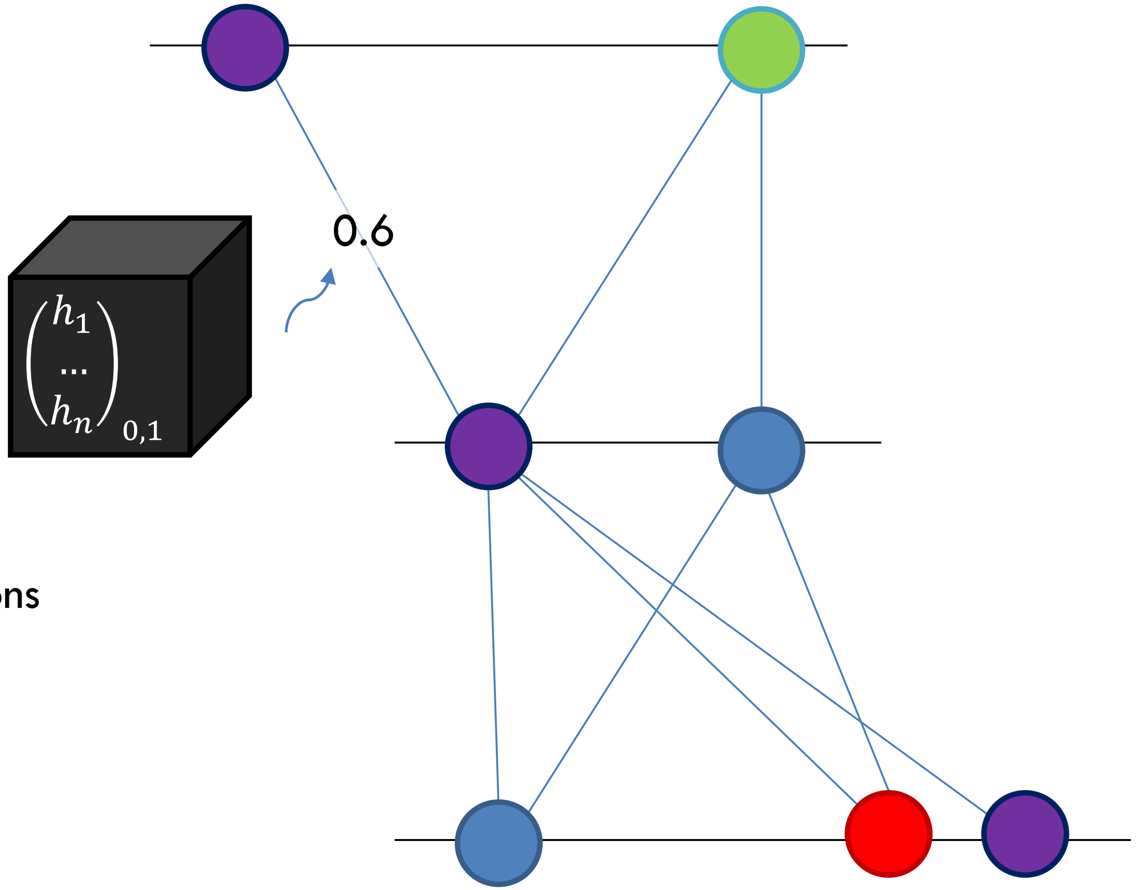
- Input node features
- Hidden node features
- **Hidden edge features**
- Edge score
- Attention aggregation
- New hidden node features
- New hidden edge features
- New edge score



Edge prediction architecture

- Input node features
- Hidden node features
- Hidden edge features
- **Edge score**
- Attention aggregation
- New hidden node features
- New hidden edge features
- New edge score

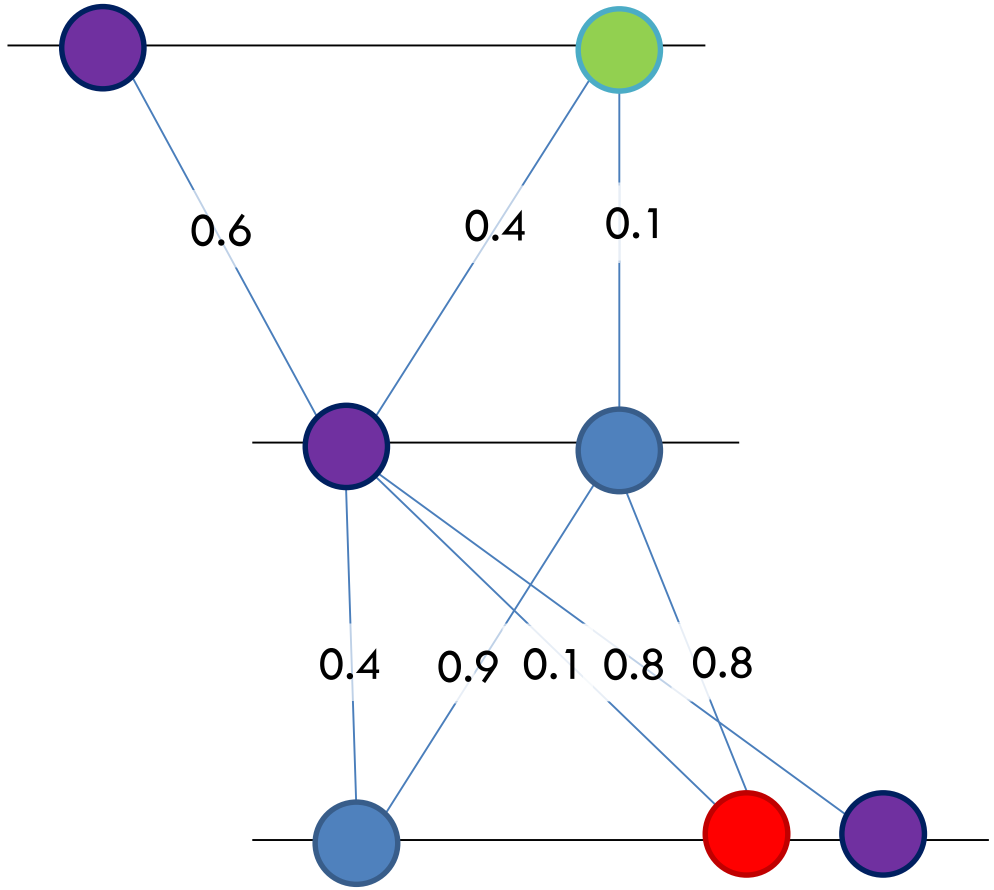
x n iterations



Edge prediction architecture

- Input node features
- Hidden node features
- Hidden edge features
- **Edge score**
- Attention aggregation
- New hidden node features
- New hidden edge features
- New edge score

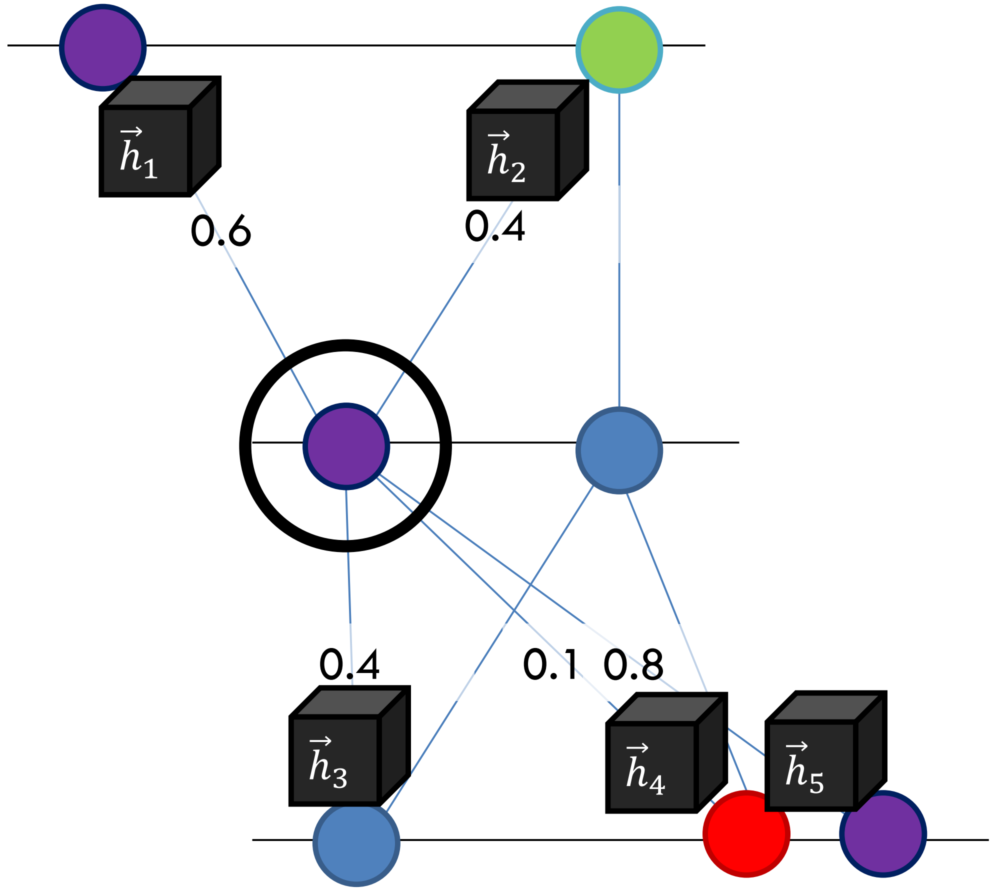
x n iterations



Edge prediction architecture

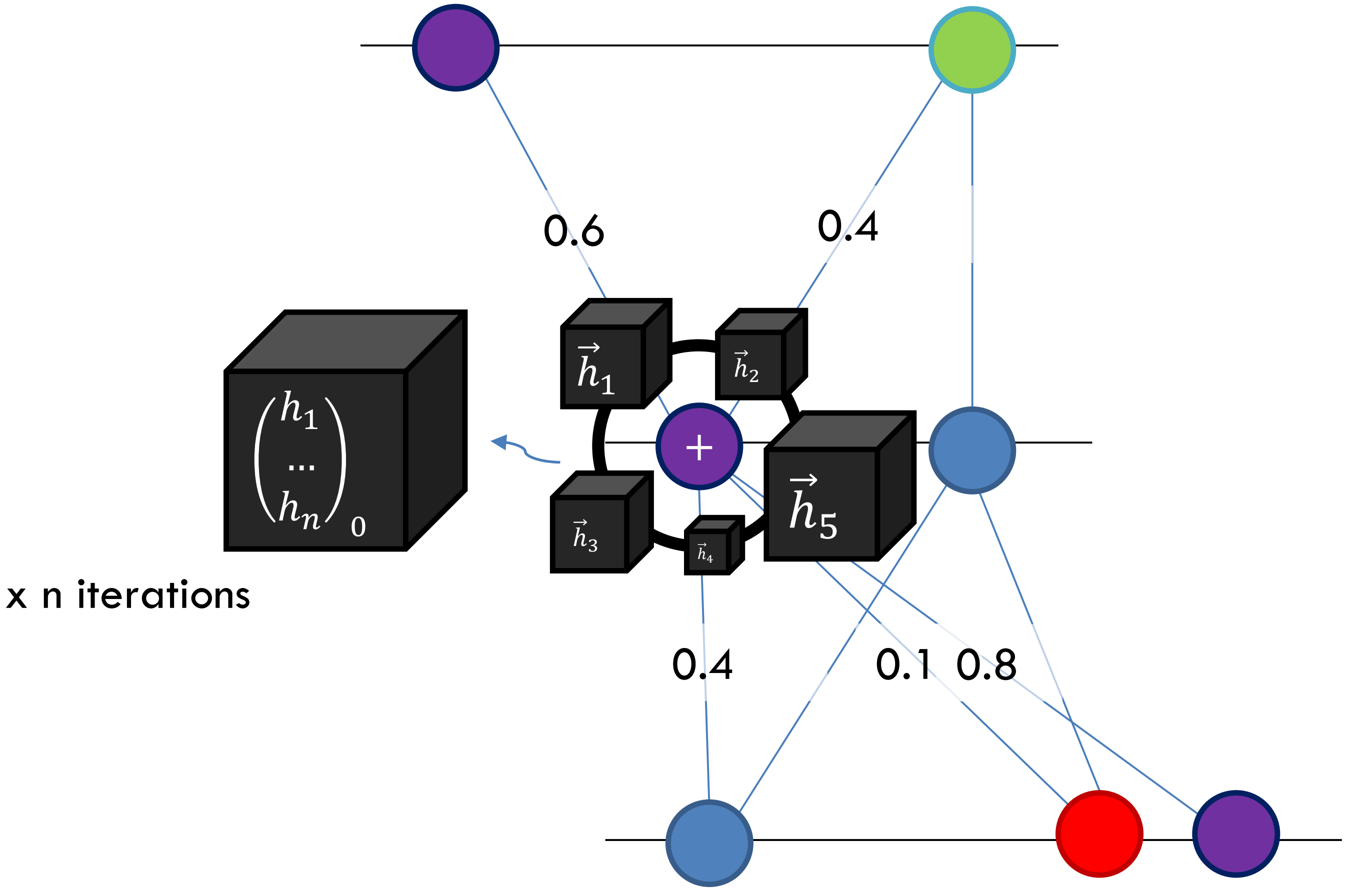
- Input node features
- Hidden node features
- Hidden edge features
- Edge score
- **Attention aggregation**
- New hidden node features
- New hidden edge features
- New edge score

x n iterations



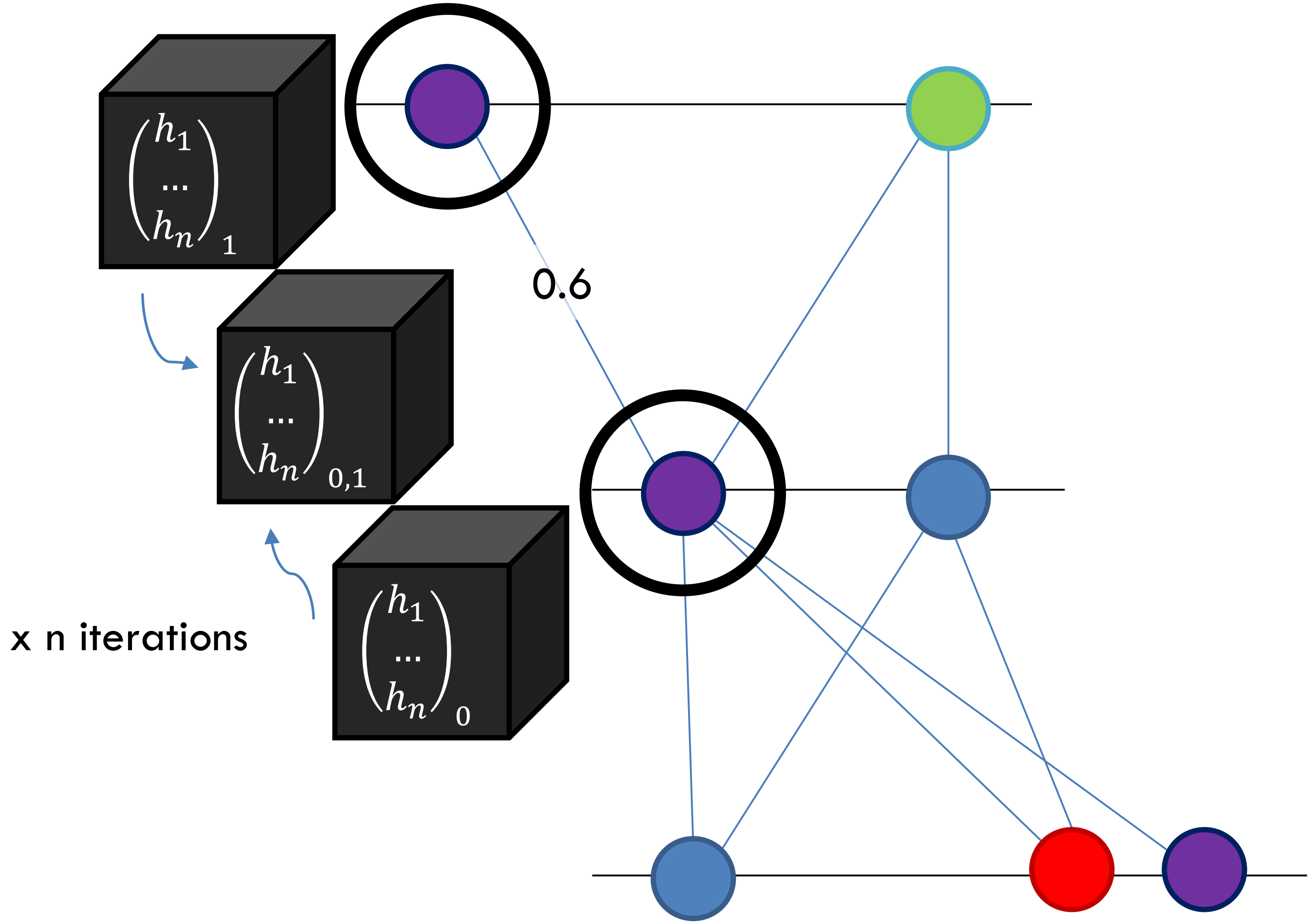
Edge prediction architecture

- Input node features
- Hidden node features
- Hidden edge features
- Edge score
- **Attention aggregation**
- **New hidden node features**
- New hidden edge features
- New edge score



Edge prediction architecture

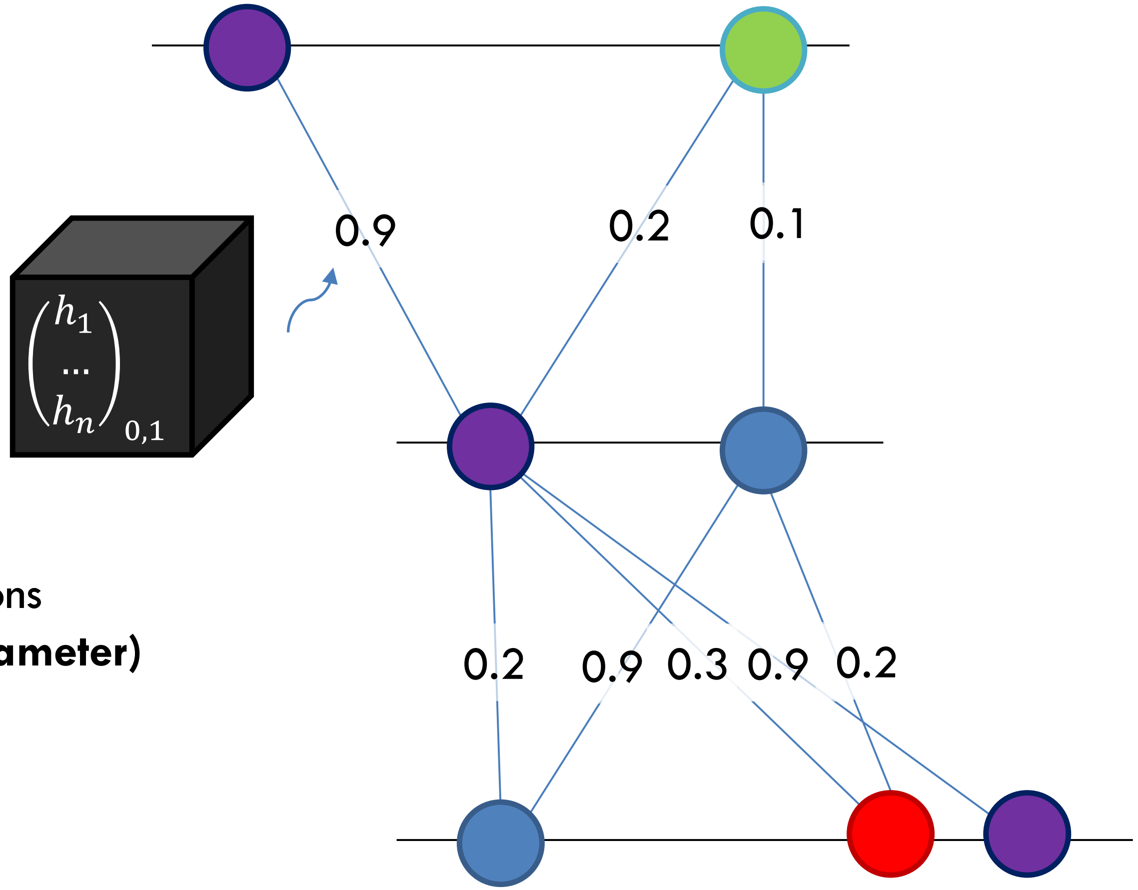
- Input node features
- Hidden node features
- Hidden edge features
- Edge score
- Attention aggregation
- New hidden node features
- **New hidden edge features**
- New edge score



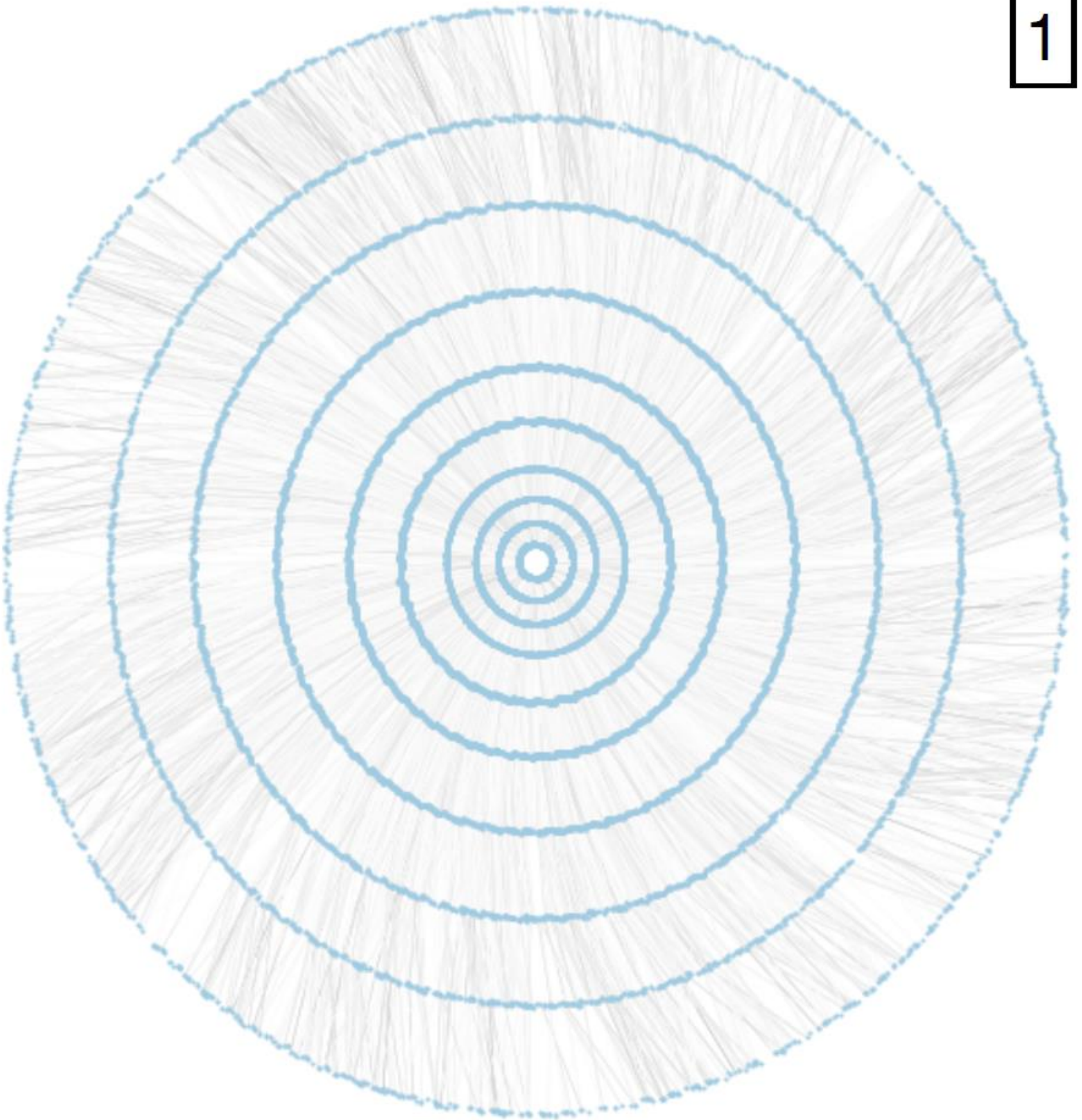
Edge prediction architecture

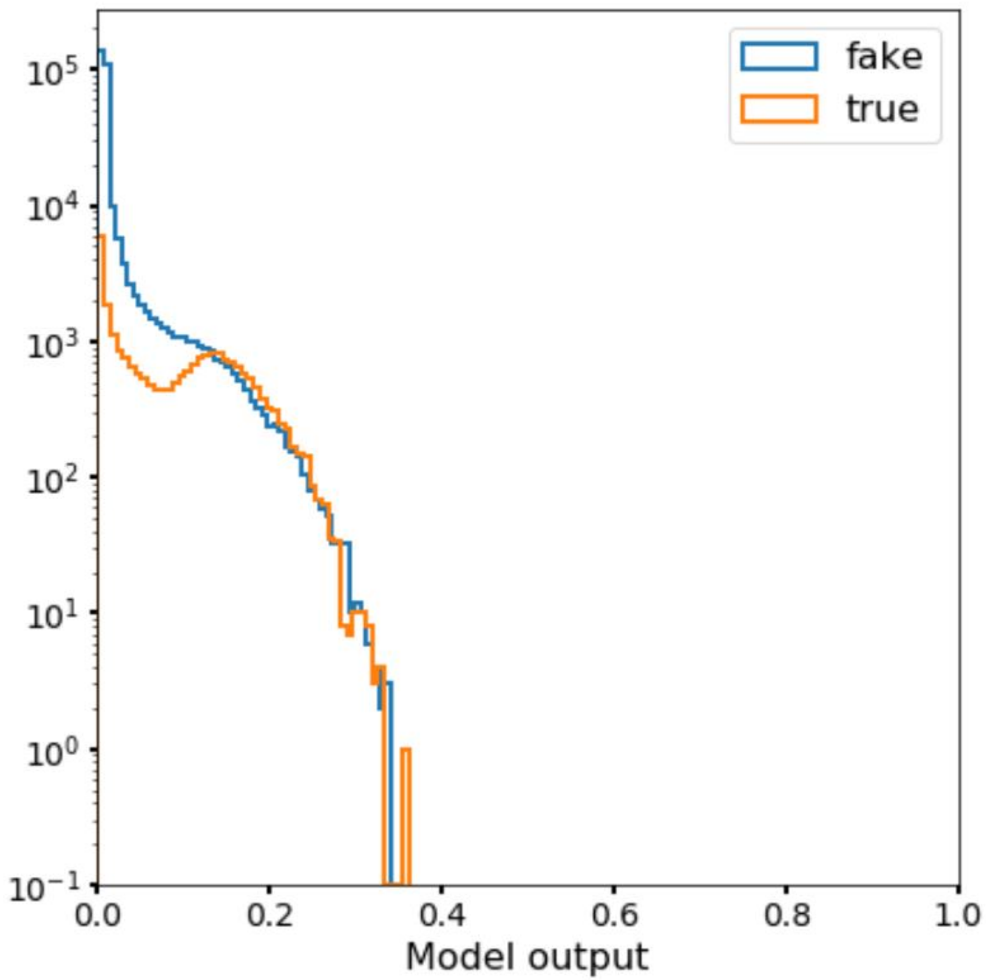
- Input node features
- Hidden node features
- Hidden edge features
- Edge score
- Attention aggregation
- New hidden node features
- New hidden edge features
- **New edge score**

x n iterations
(hyperparameter)



1

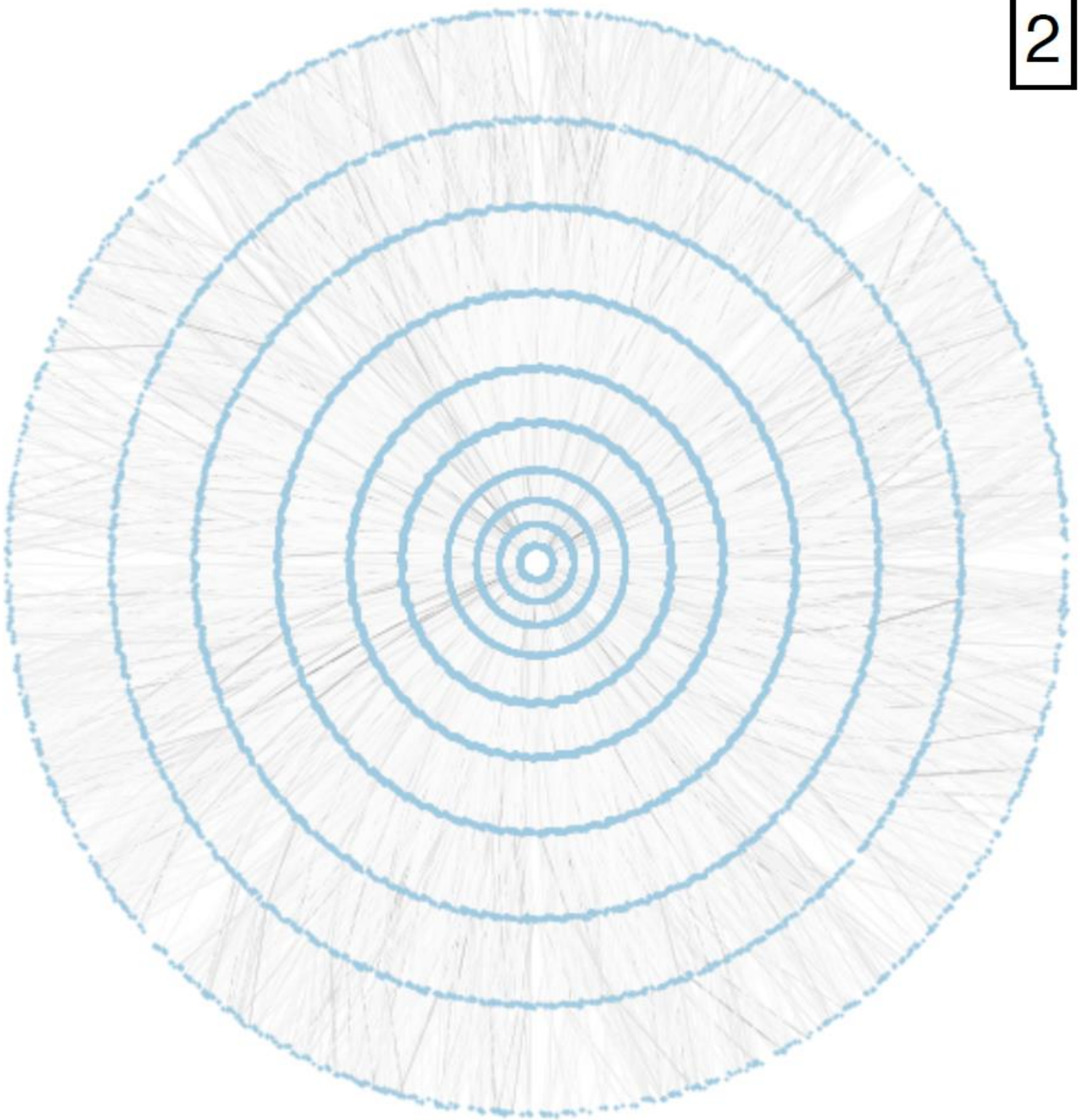


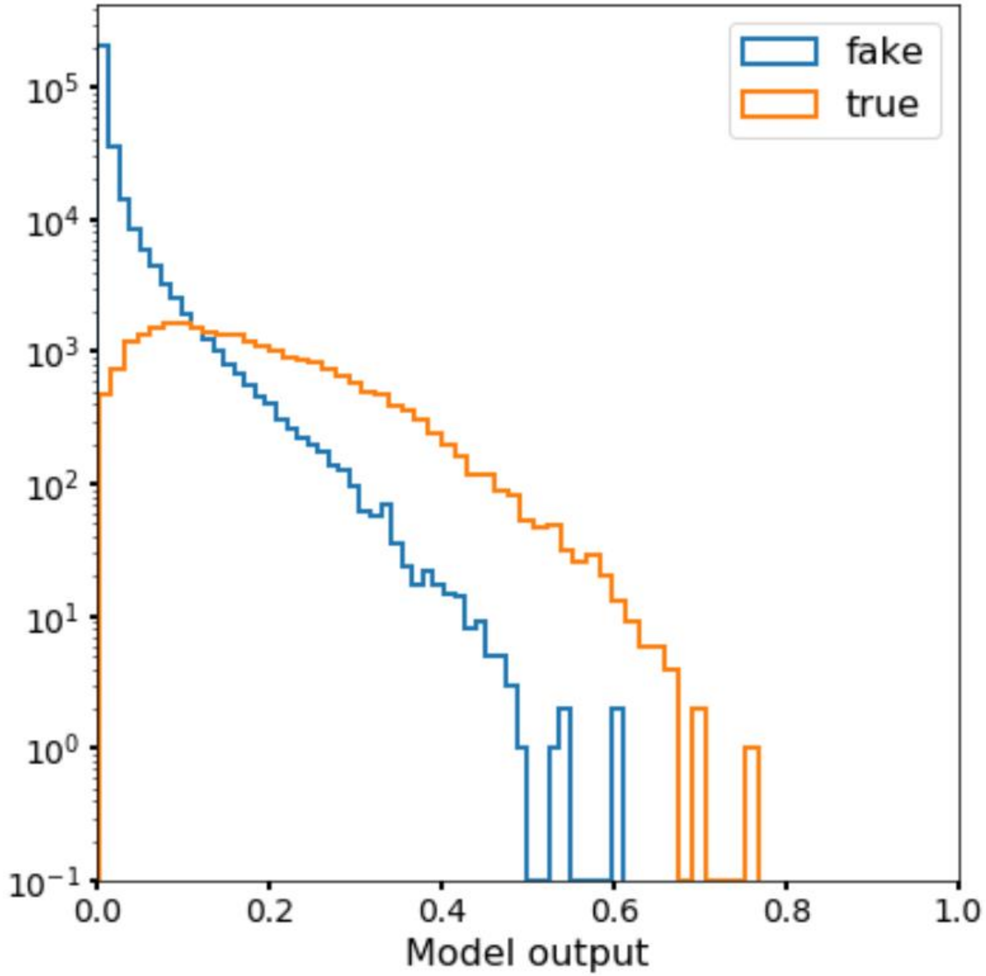


Edges with higher scores are darker than that with lower scores

Edges with scores < 0.01 are removed for visualization purpose.

2

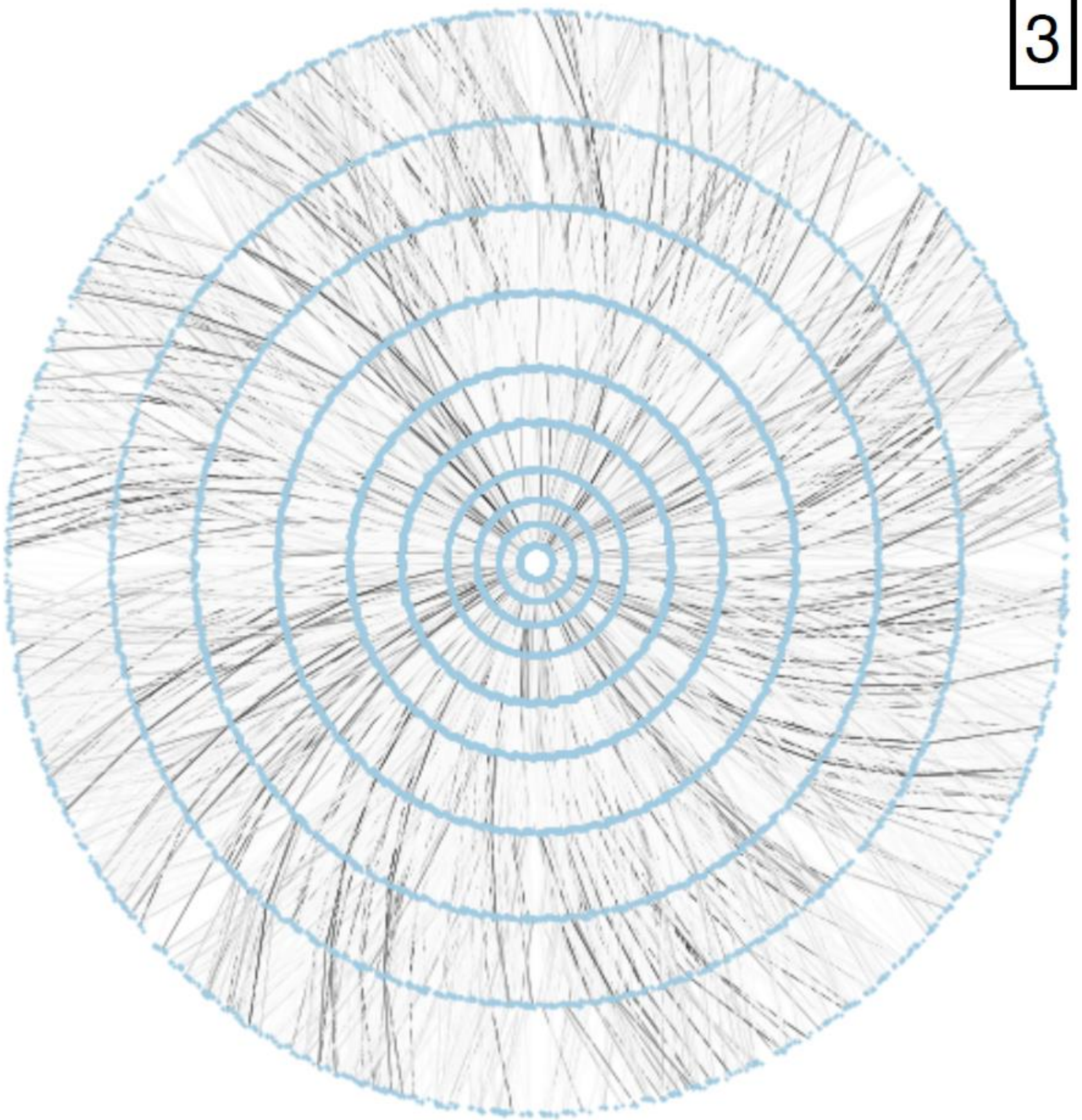


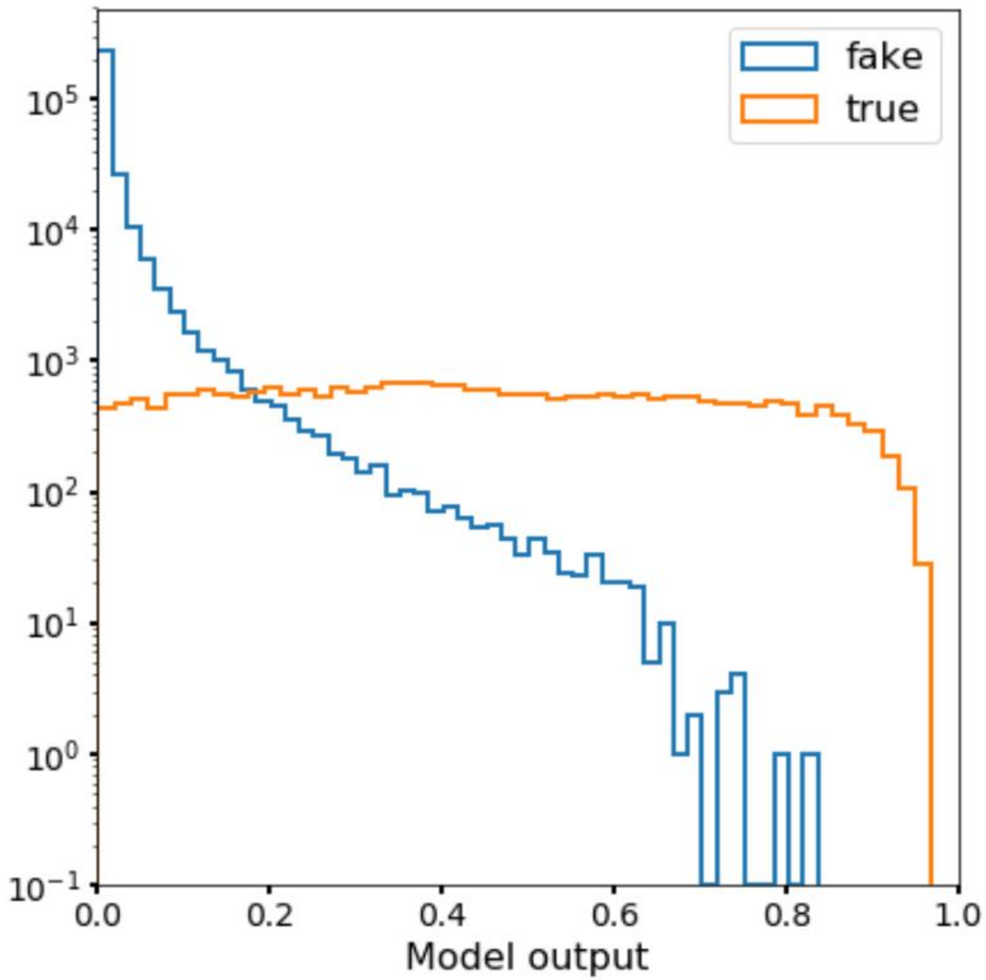


Edges with higher scores are darker than that with lower scores

Edges with scores < 0.01 are removed for visualization purpose.

3

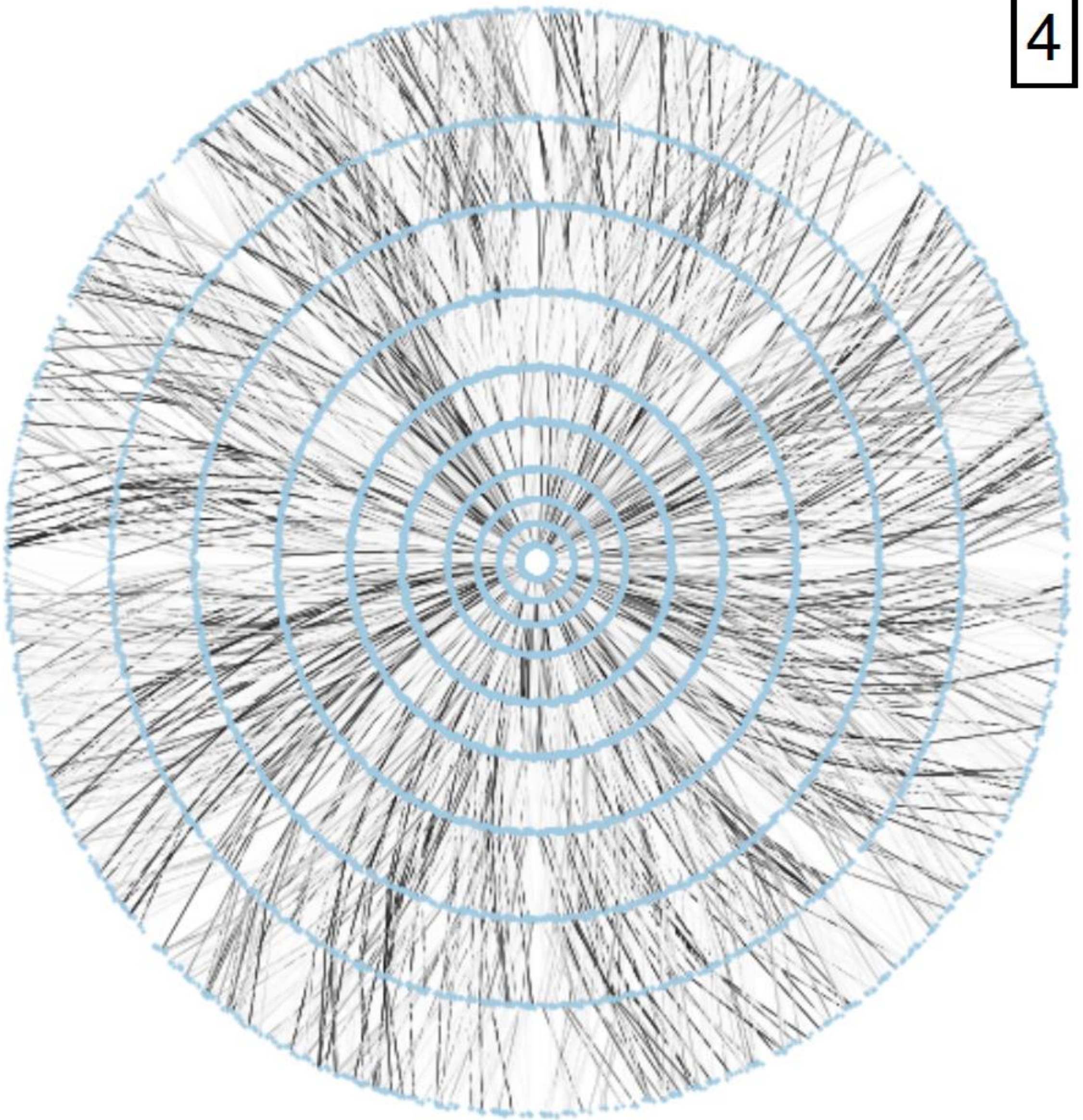


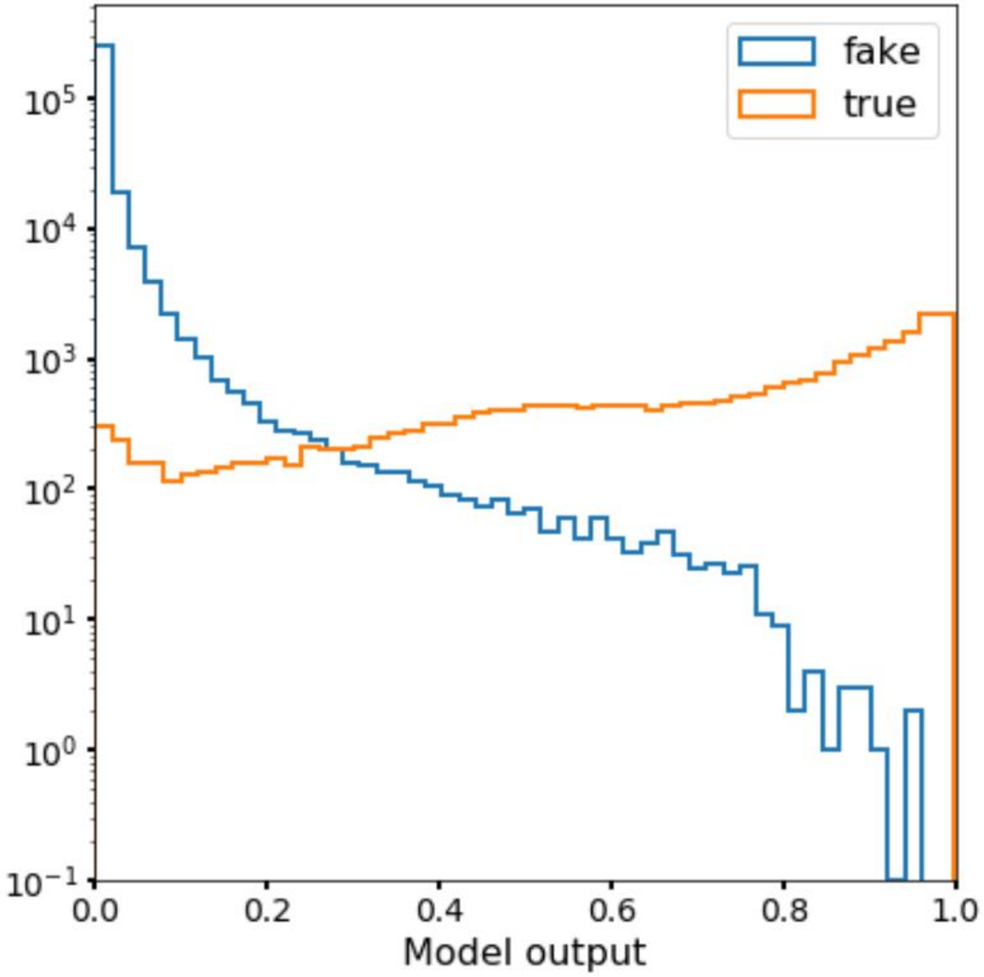


Edges with higher scores are darker than that with lower scores

Edges with scores < 0.01 are removed for visualization purpose.

4

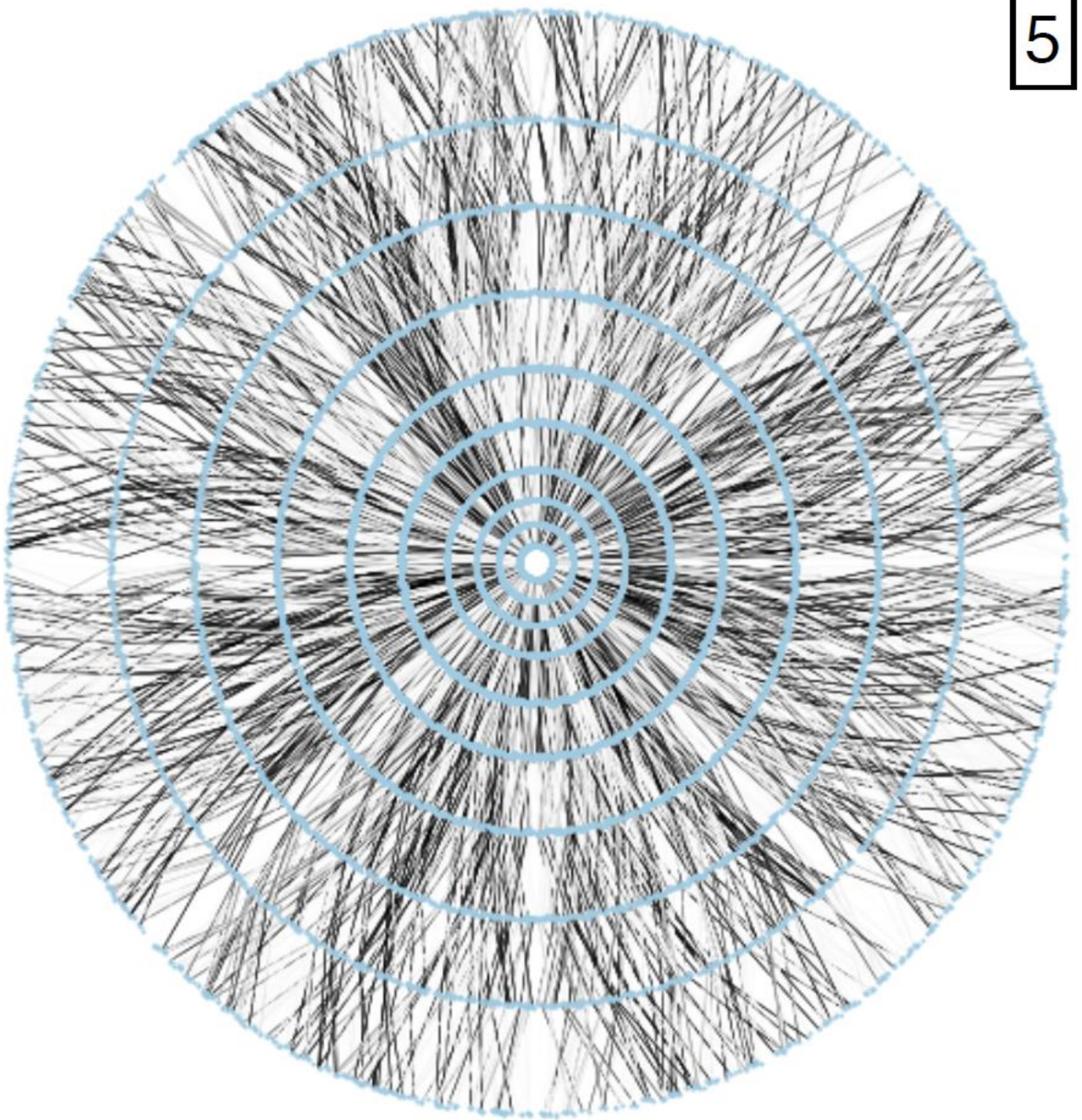


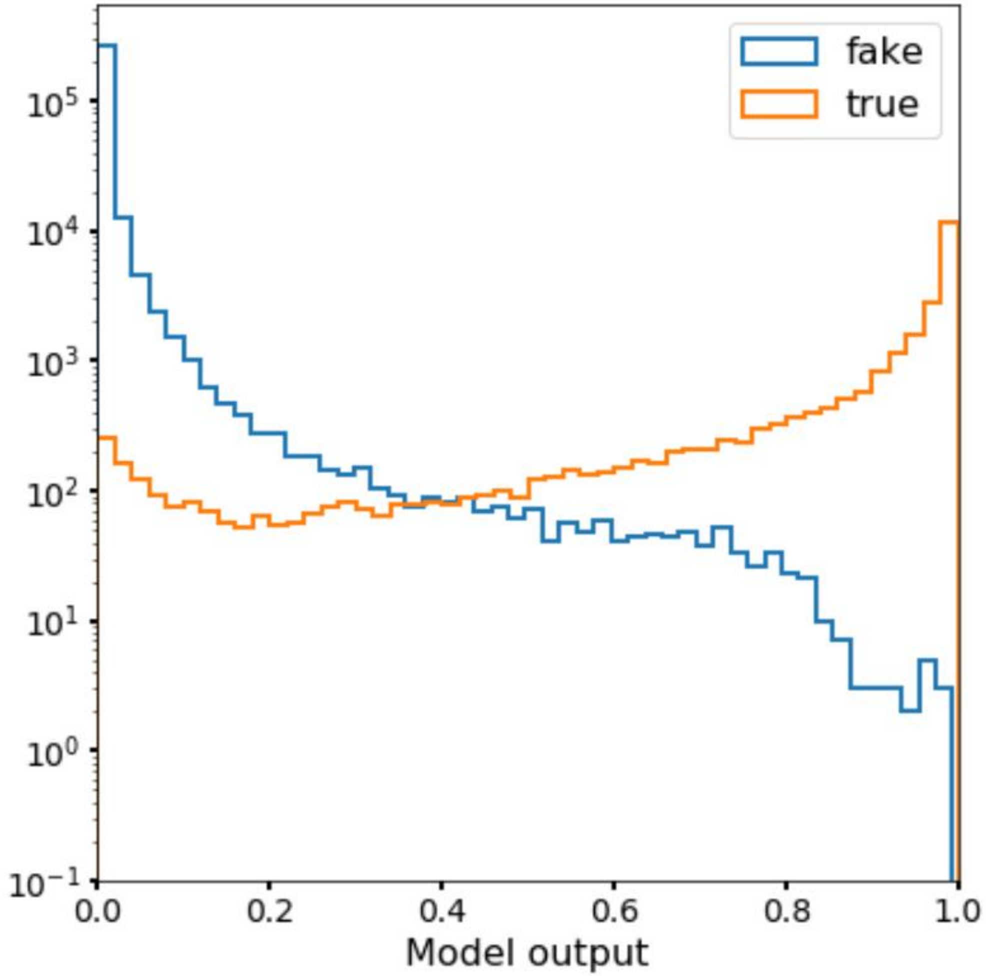


Edges with higher scores are darker than that with lower scores

Edges with scores < 0.01 are removed for visualization purpose.

5

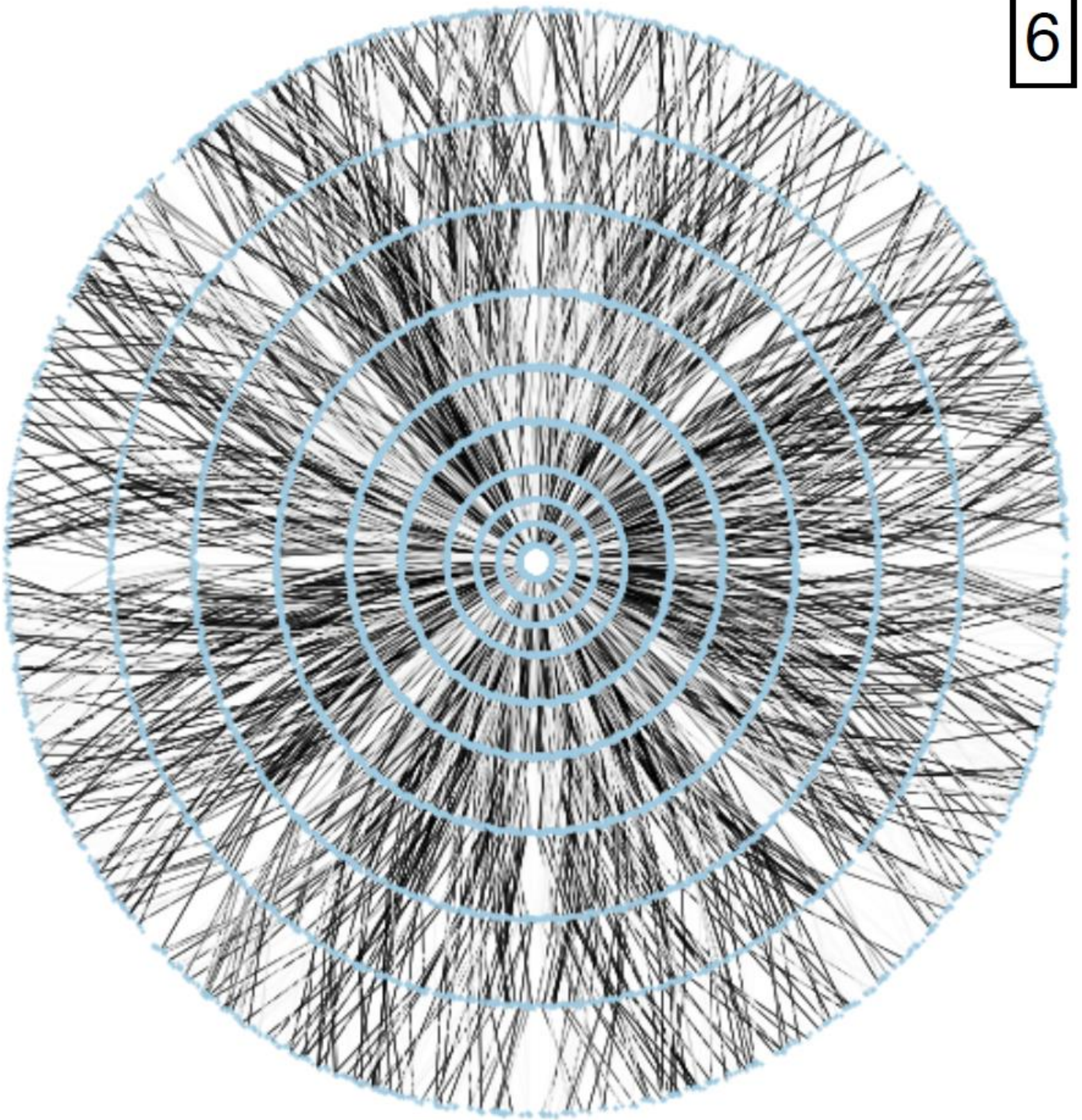


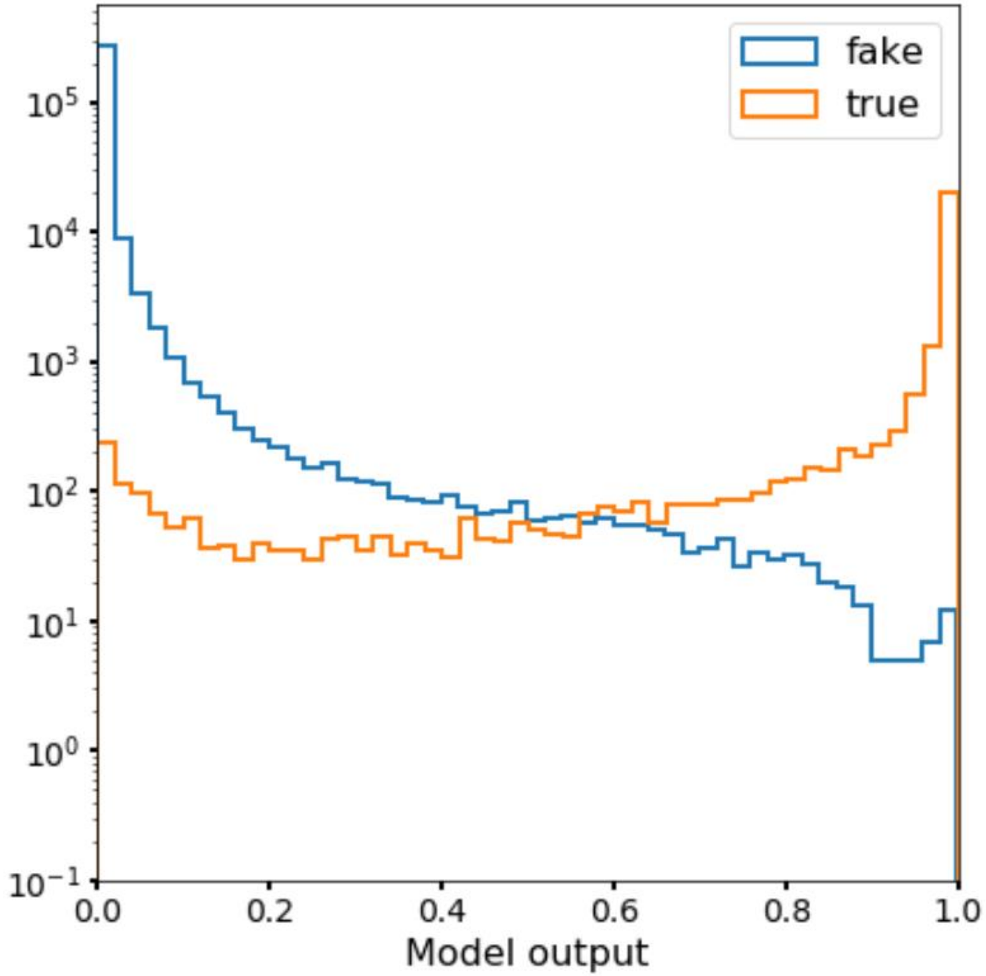


Edges with higher scores are darker than that with lower scores

Edges with scores < 0.01 are removed for visualization purpose.

6

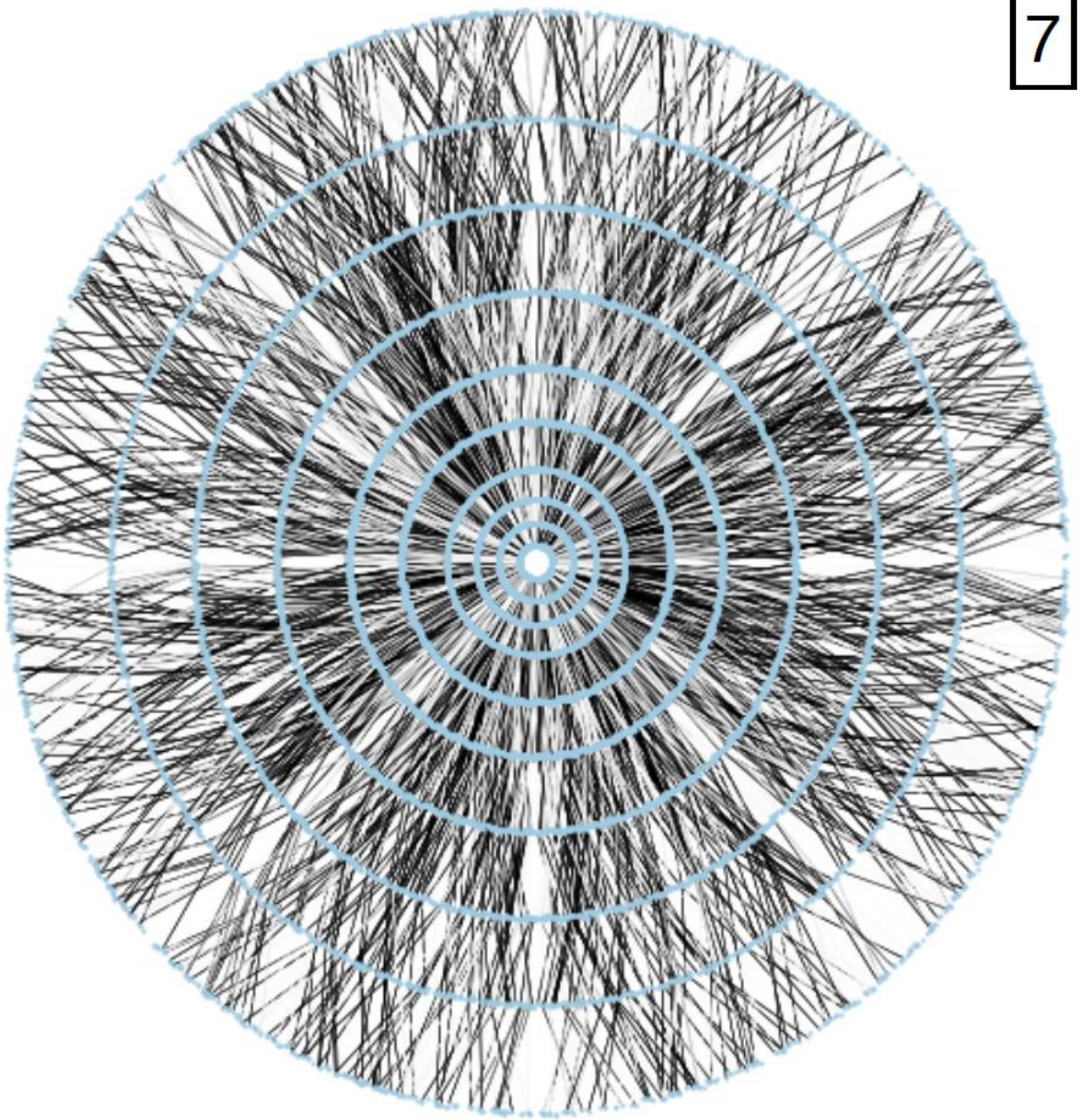


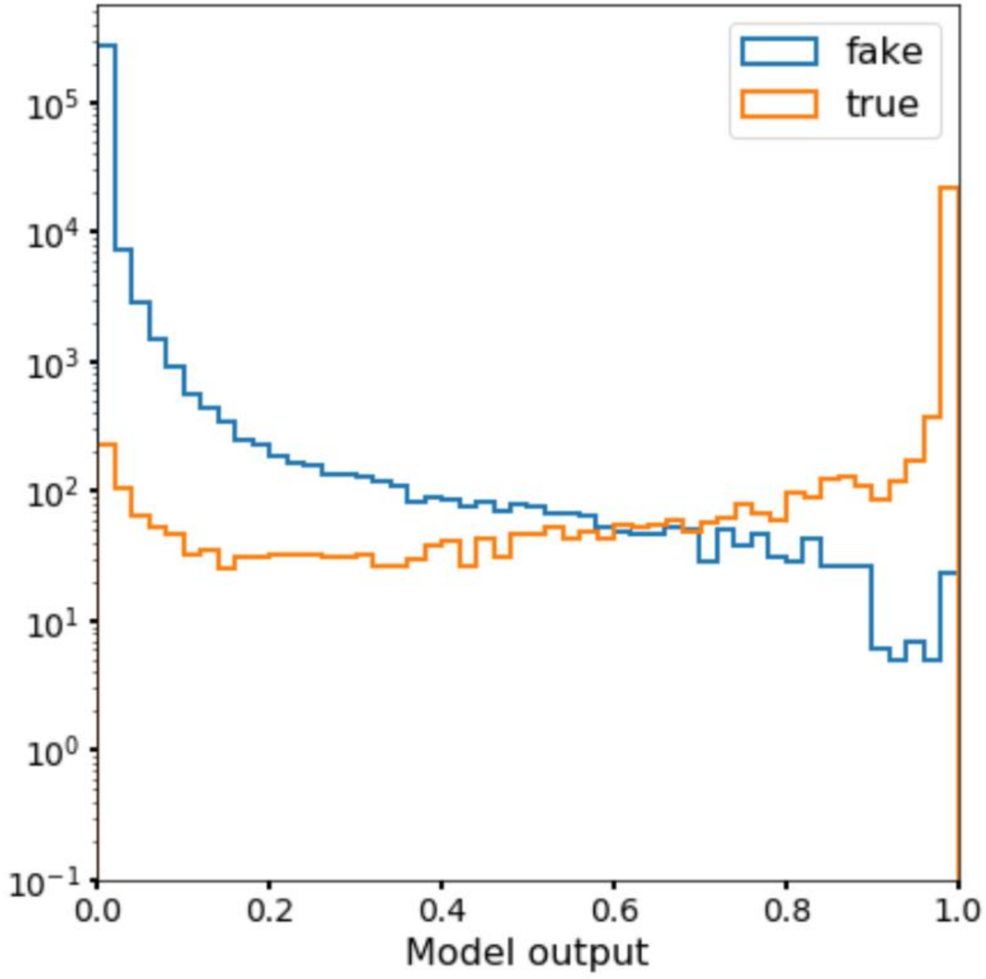


Edges with higher scores are darker than that with lower scores

Edges with scores < 0.01 are removed for visualization purpose.

7

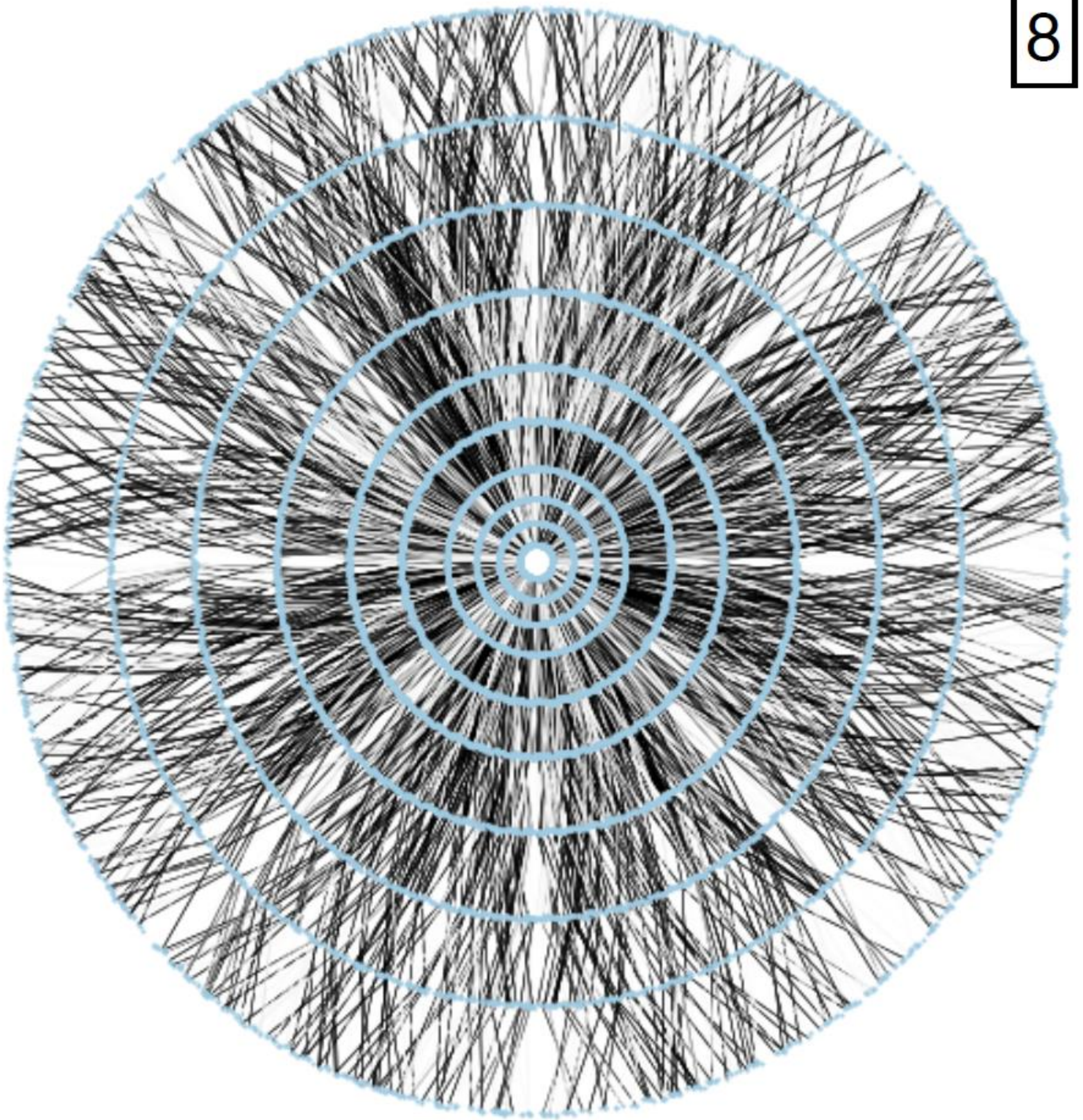


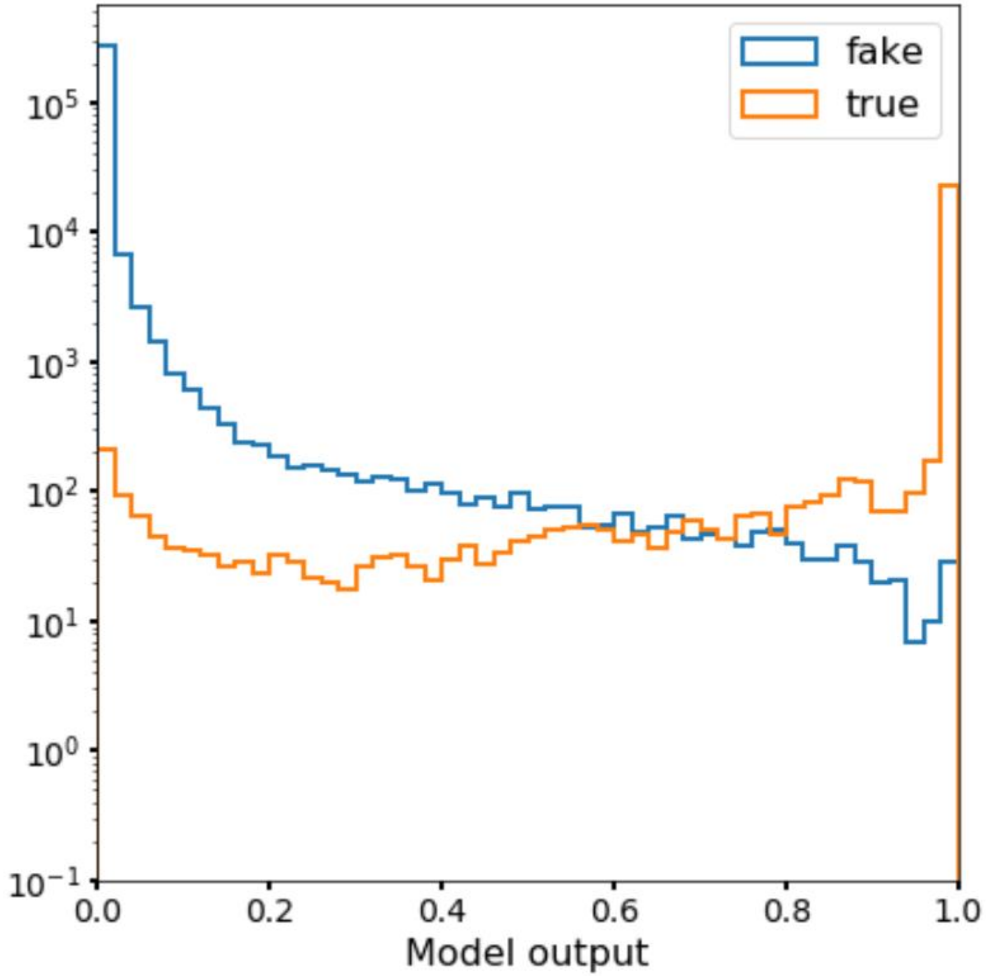


Edges with higher scores are darker than that with lower scores

Edges with scores < 0.01 are removed for visualization purpose.

8

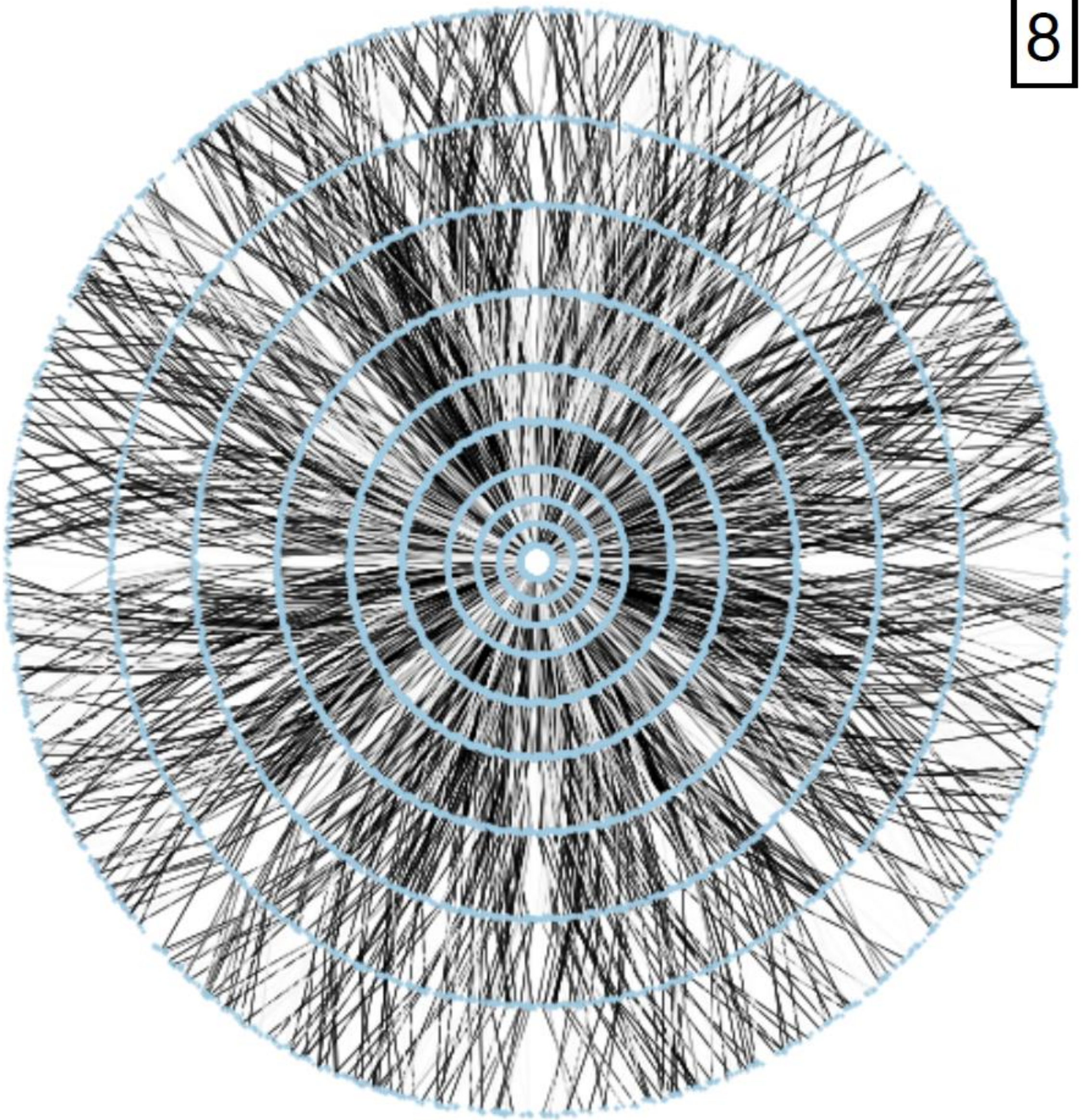


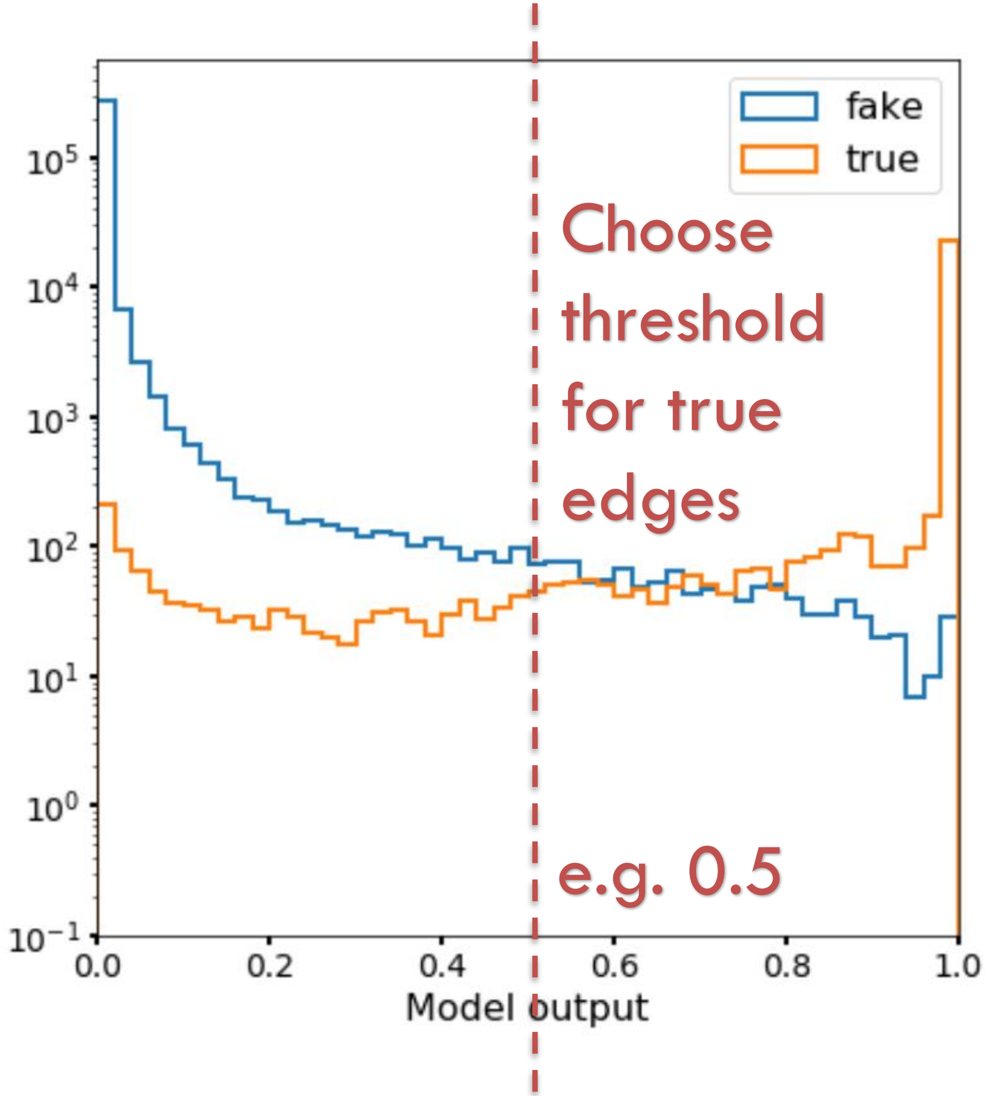


Edges with higher scores are darker than that with lower scores

Edges with scores < 0.01 are removed for visualization purpose.

8





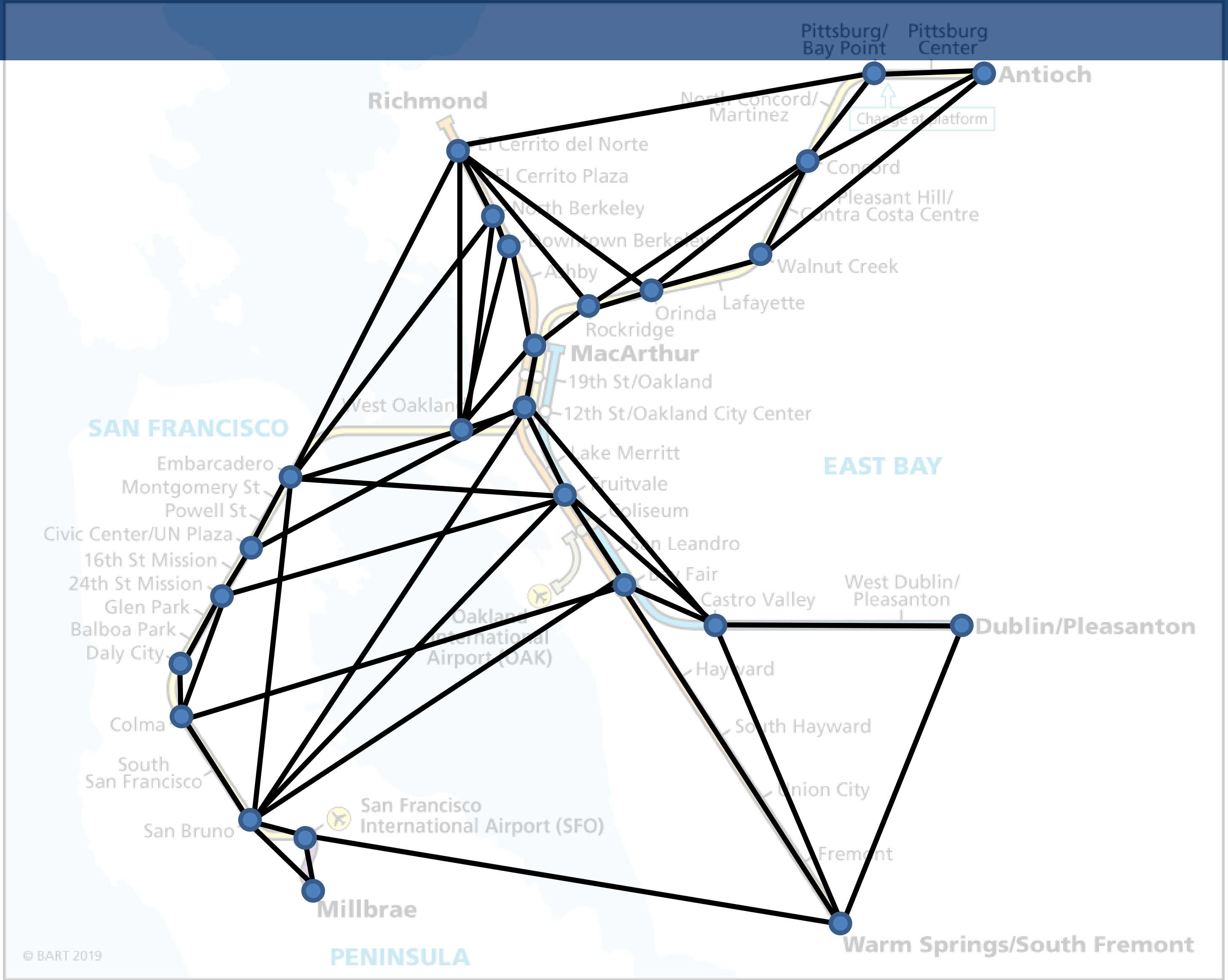
Edges with higher scores are darker than that with lower scores

Edges with scores < 0.01 are removed for visualization purpose.

Now we have classified doublets

Each edge has a score between [0,1]

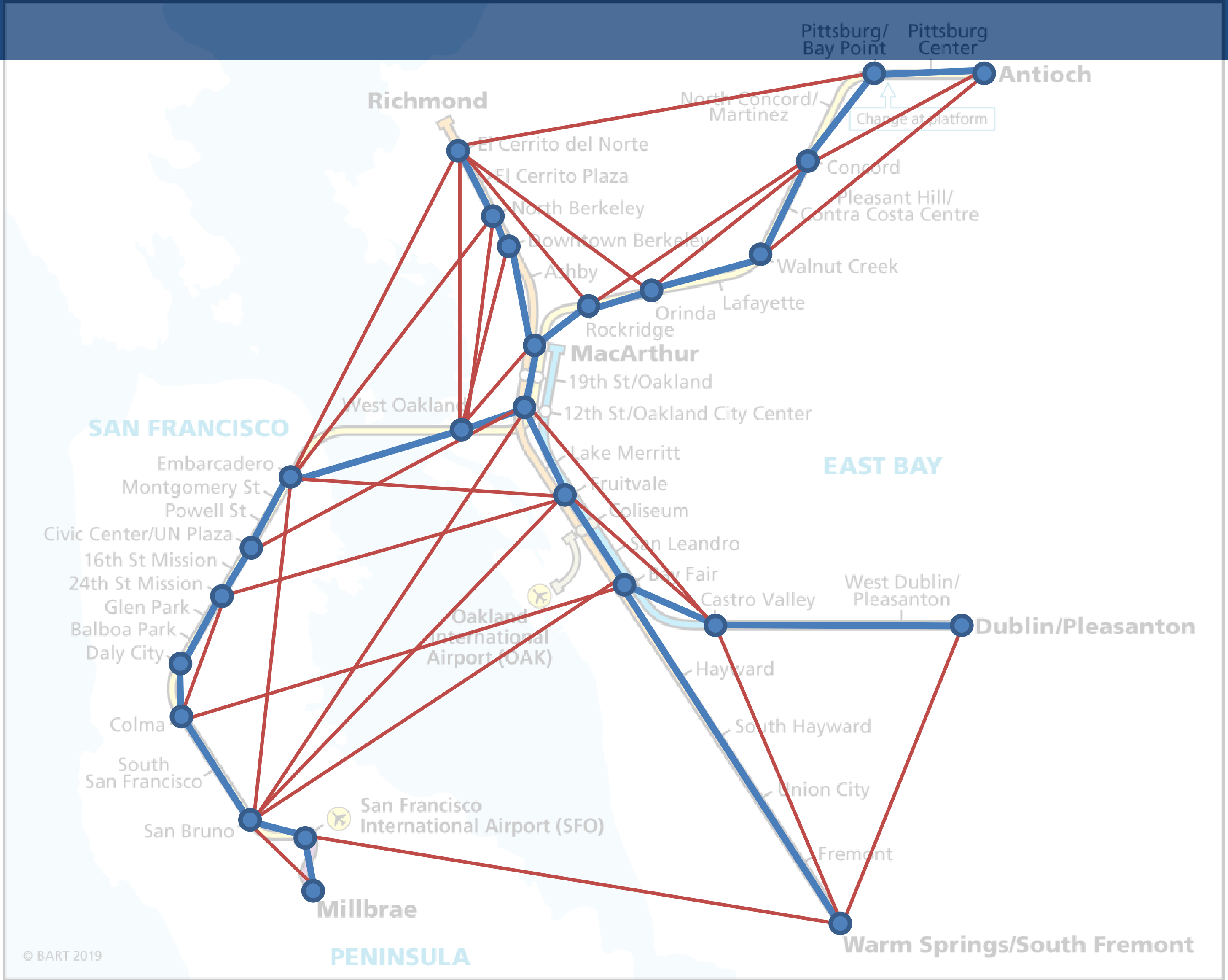
How do we make tracks…



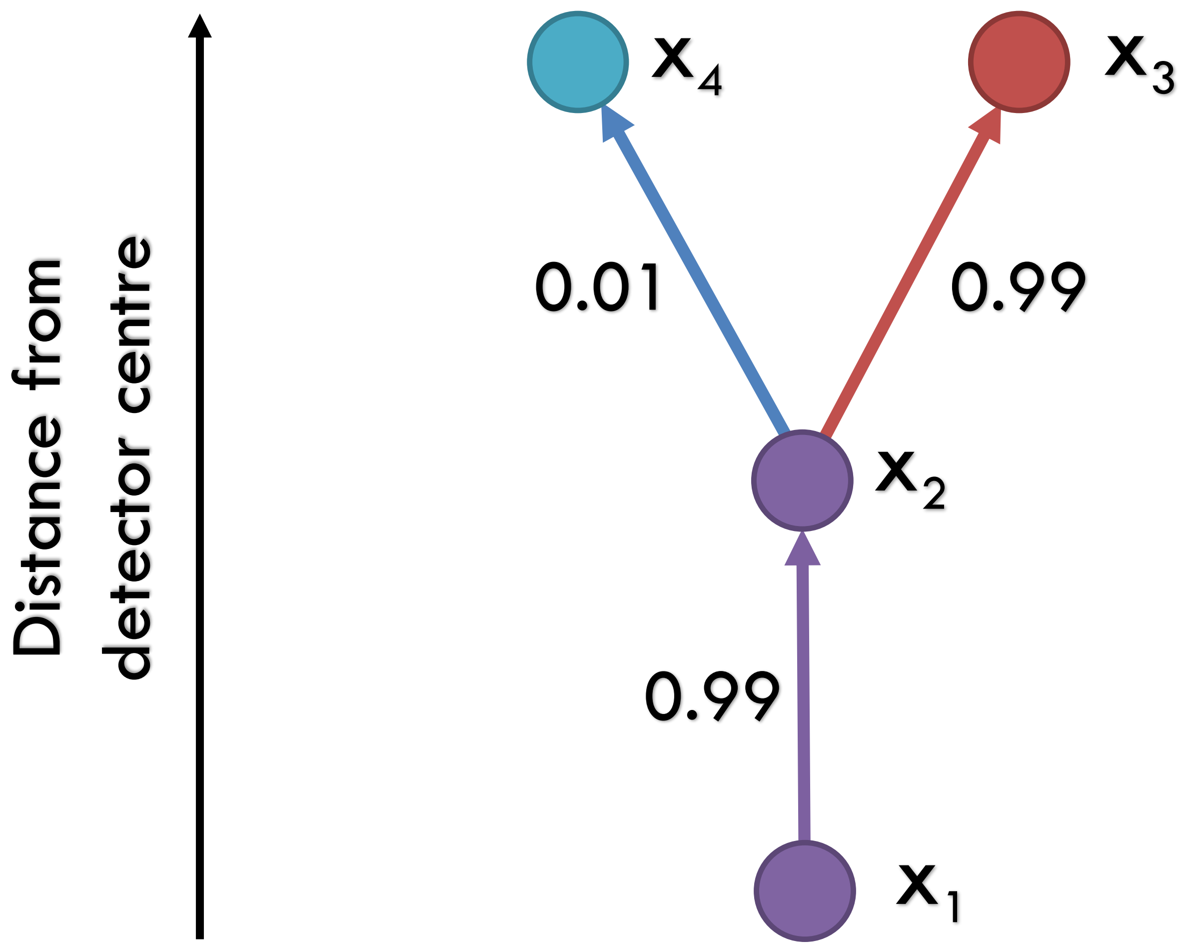
Now we have classified doublets

Each edge has a score between [0,1]

How do we make tracks…

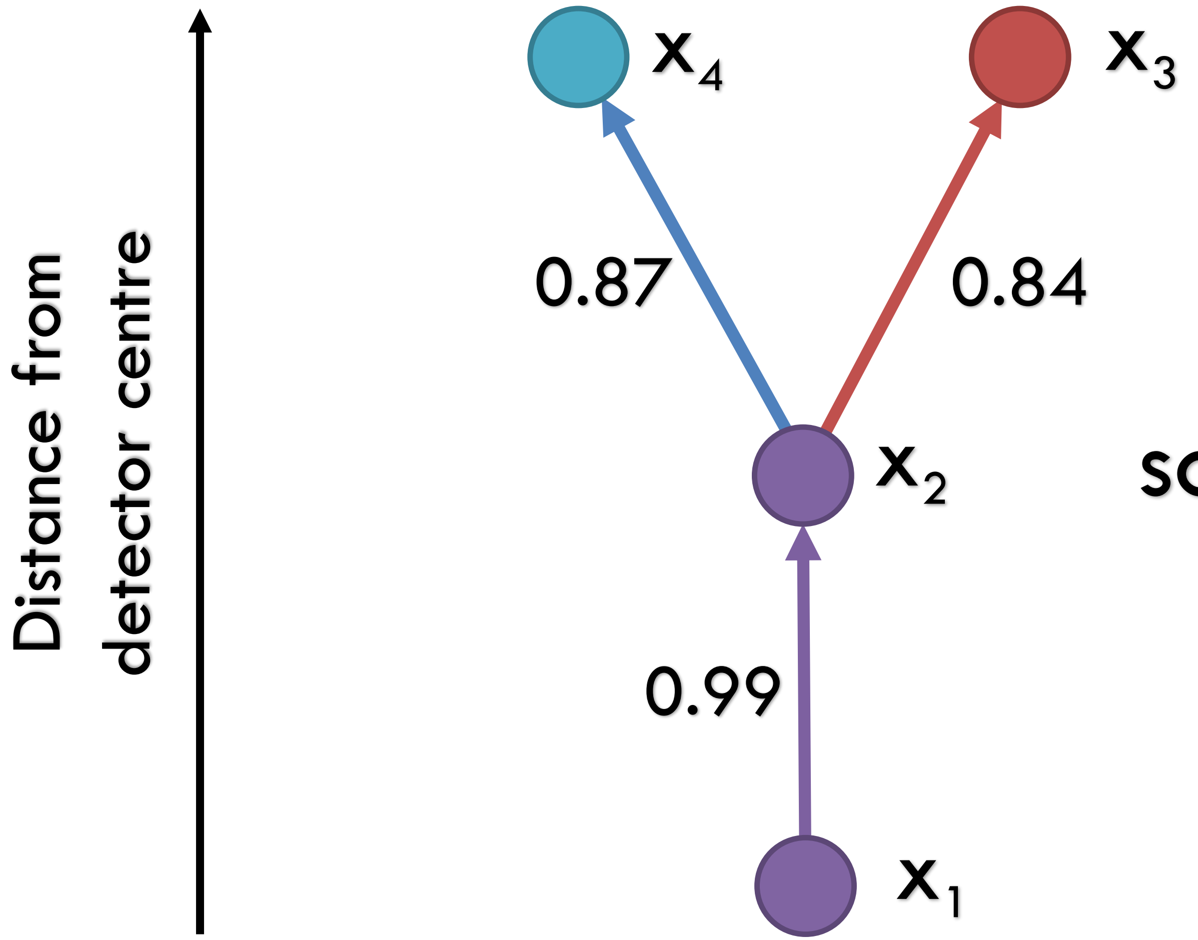


Why not simply join together our doublet predictions?



Pretty easy decision

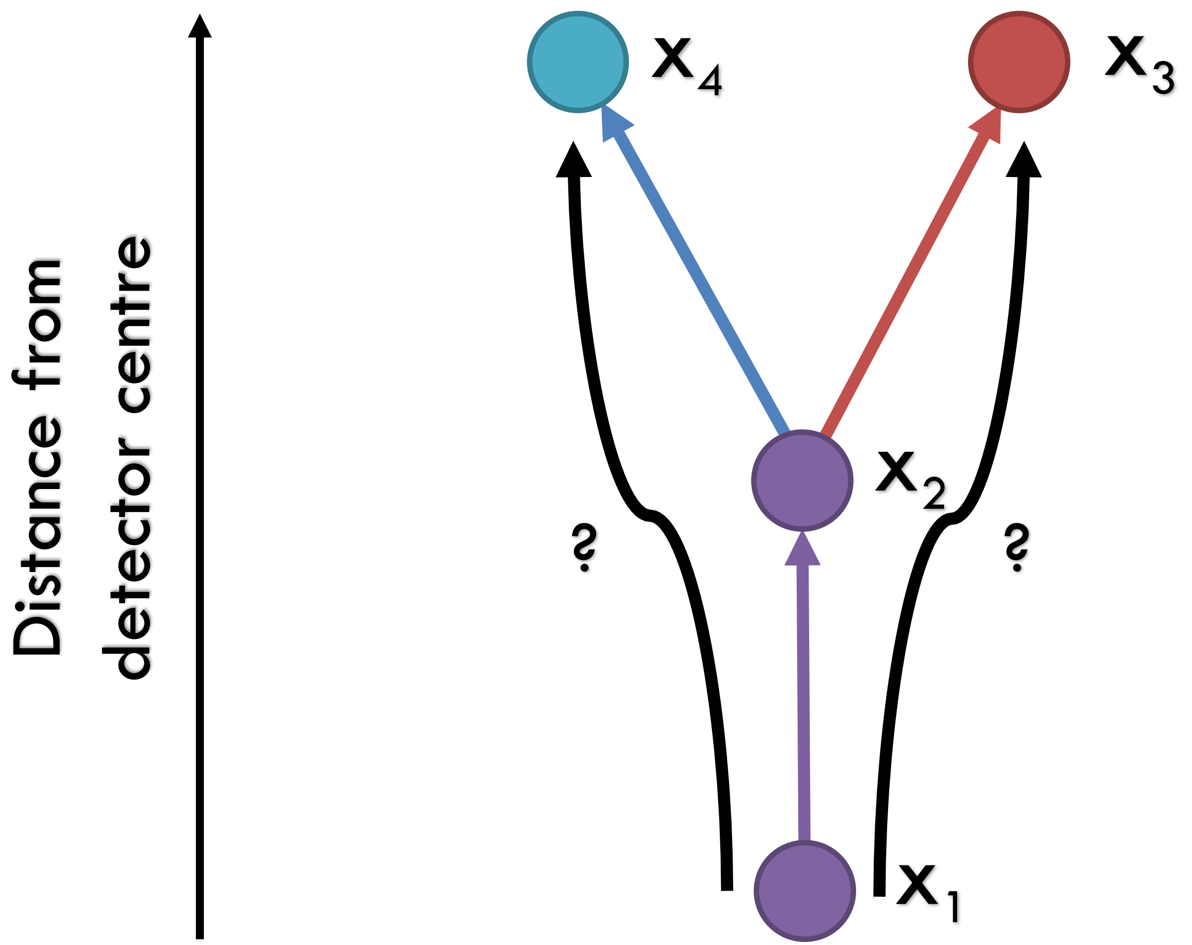
Doublet choice can be ambiguous



Not so easy…

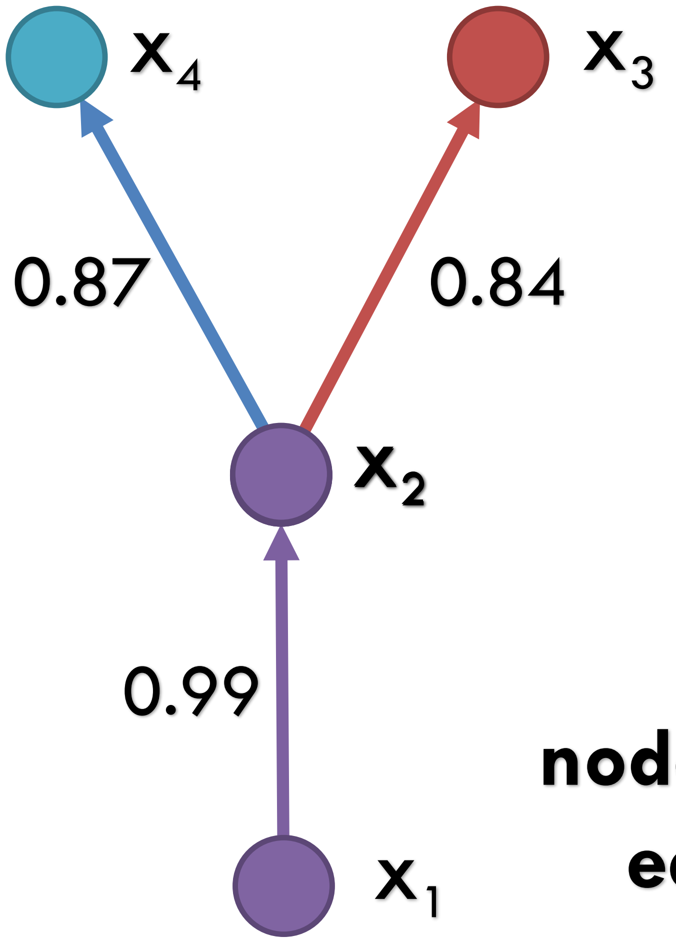
so teach the network how to combine

But a GNN doesn't know about "triplets"



A GNN only knows about nodes and edge

Moving to a “doublet graph” gives us back GNN power

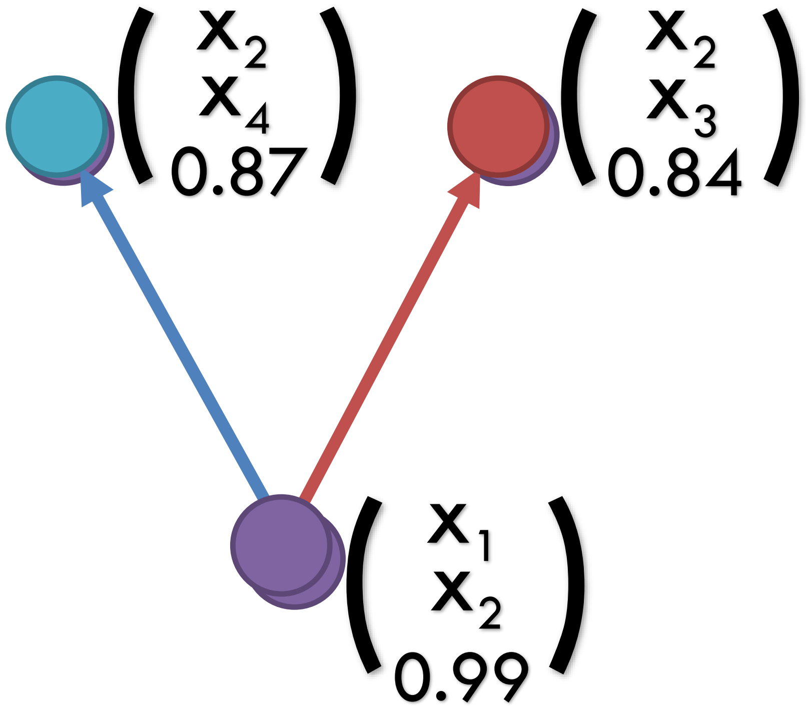


Now…

nodes represent **doublets**,

edges represent **triplets**

Moving to a "doublet graph" gives us back GNN power



Now…

nodes represent **doublets,**

edges represent **triplets**

The triplet classifier runs with all the benefits of the doublet classifier

- Aim is to beat all traditional methods of finding true triplets
- Can then either continue to 4, 5, …-plets in order to create and end-to-end GNN track builder…
- …or hand off the triplets as seeds to the traditional techniques, knowing we can be confident in their accuracy

Triplet GNN performs very well

Gold: Unambiguously correct triplet or quadruplet

Other colours: False positive/negative

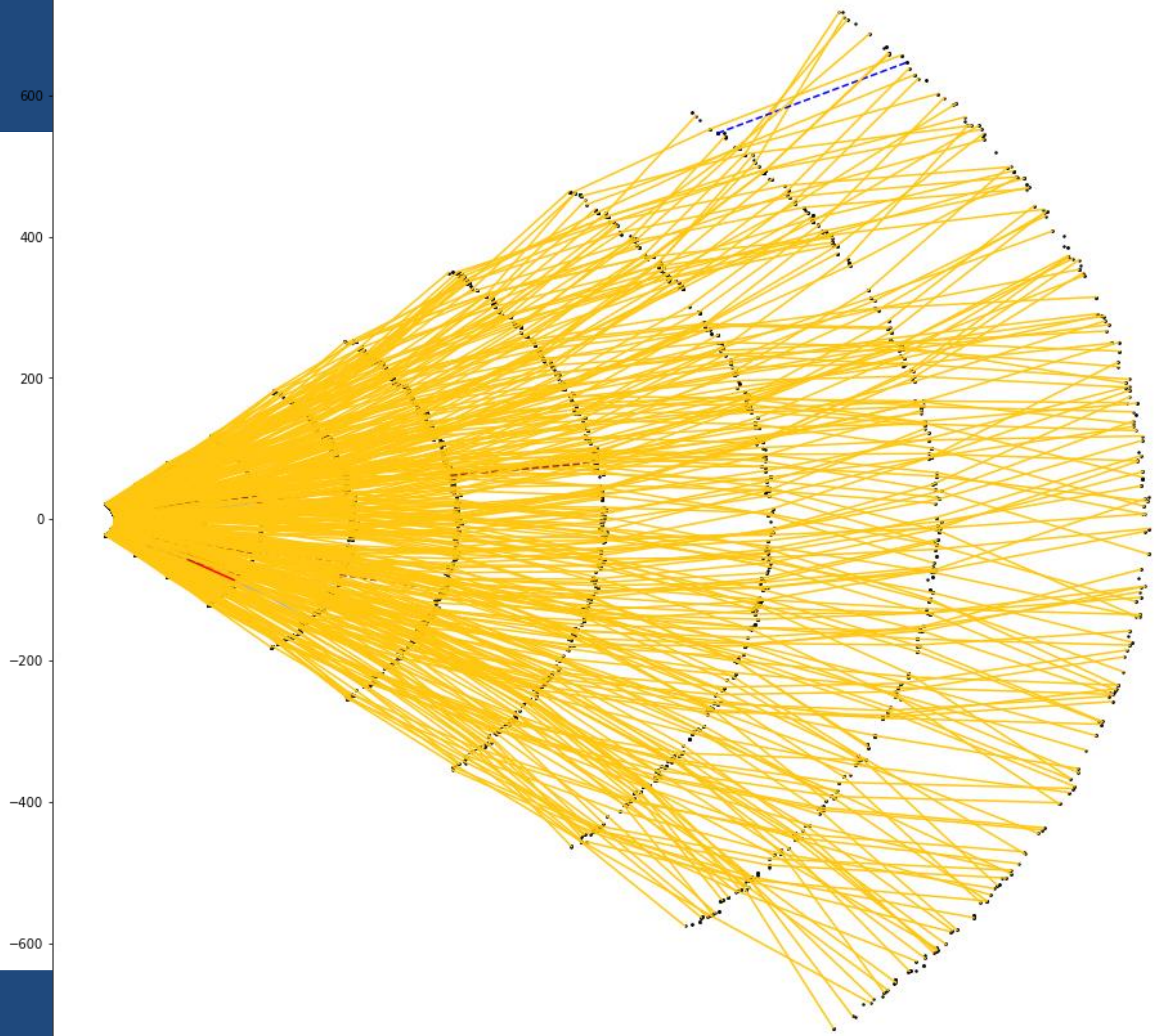
Key:

Silver: Ambiguously correct triplet or quadruplet (i.e. edge shared by correct triplet and false positive triplet)

Bronze dashed: Correct triplet, but missed quadruplet (i.e. edge shared by correct triplet and false negative triplet)

Red: Completely false positive triplet

Blue dashed: Completely false negative triplet



Triplet GNN performs very well

Gold: Unambiguously correct triplet or quadruplet

Other colours: False positive/negative

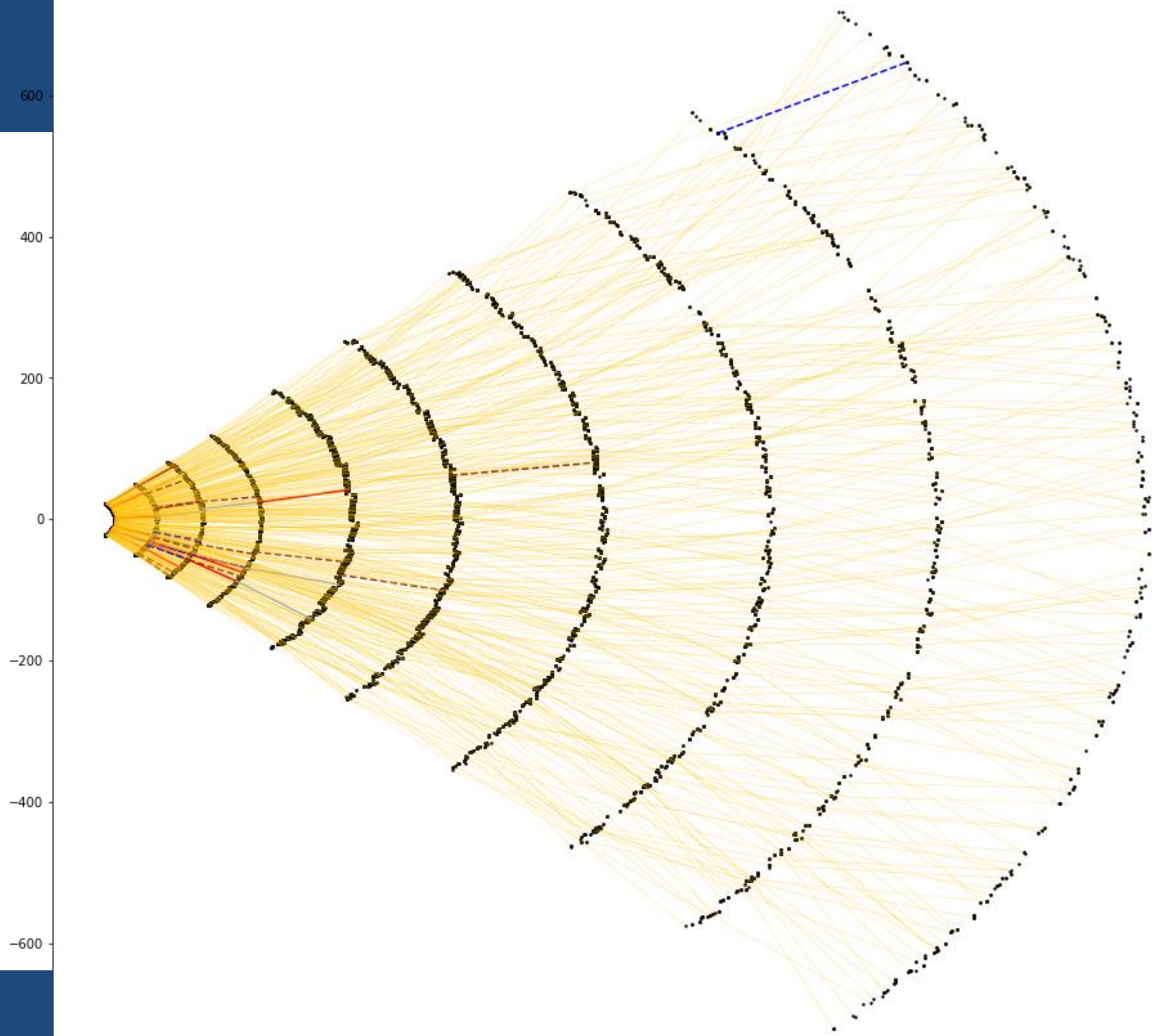
Key:

Silver: Ambiguously correct triplet or quadruplet (i.e. edge shared by correct triplet and false positive triplet)

Bronze dashed: Correct triplet, but missed quadruplet (i.e. edge shared by correct triplet and false negative triplet)

Red: Completely false positive triplet

Blue dashed: Completely false negative triplet



Triplet GNN improves doublet GNN results

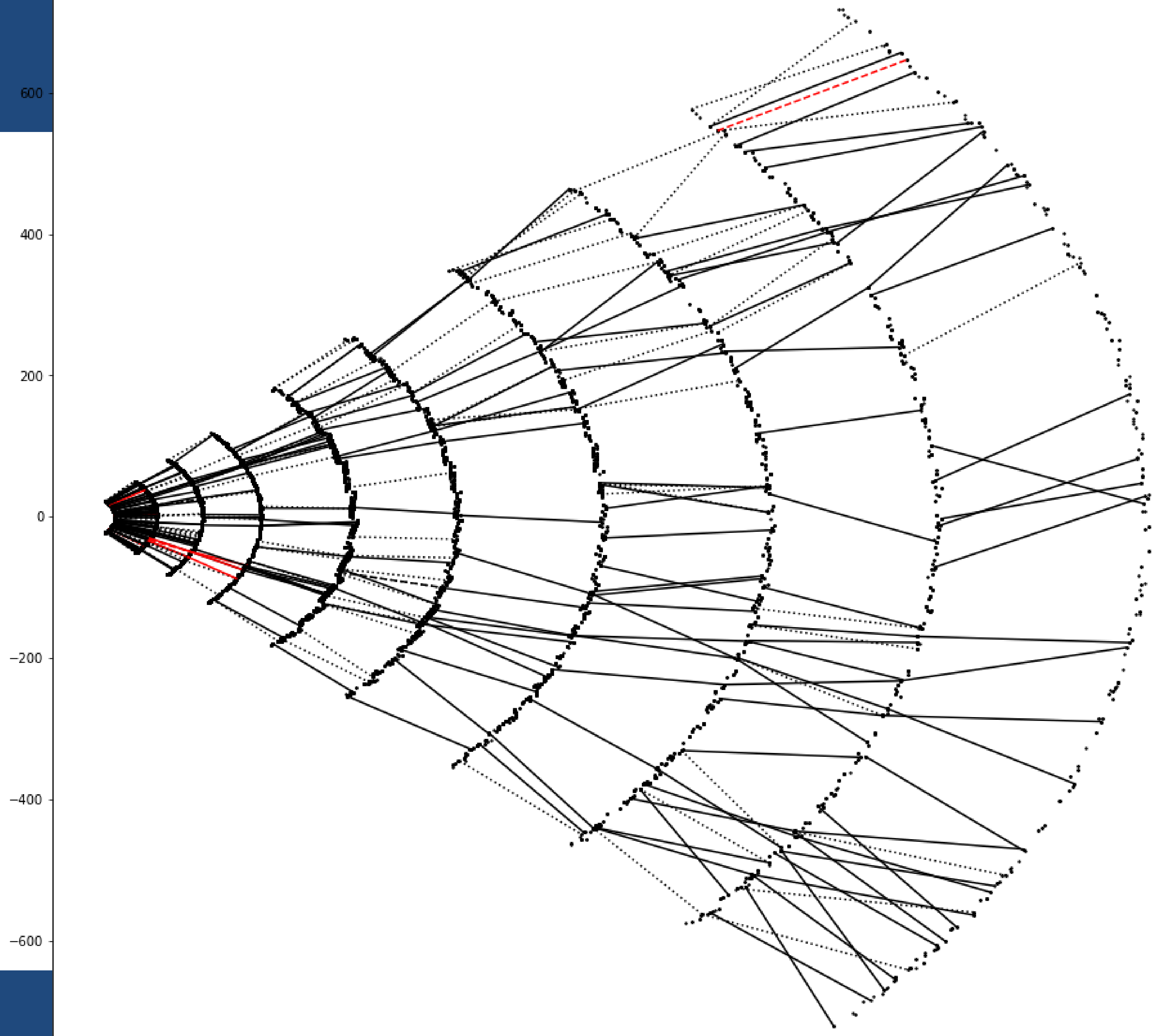
Black: Triplet classifier correctly labelled, doublet classifier mislabelled

Red: Doublet classifier correctly labelled, triplet classifier mislabelled

In this graph, triplet classifier

Fixes 389 edges

Worsens 10 edges



Score threshold gives a smarter triplet classifier

- **Excellent performance:**
  - 99.09% efficiency

But…

FYI

$$\text{Efficiency} = \frac{\text{\# triplets classified as true}}{\text{Total \# of true triplets}}$$

- **Problem:** combinatorically increasing graph size

  e.g. For TrackML data:
  - $O(1{,}000)$ tracks,
  - $O(6{,}000)$ hits,
  - $O(28{,}000)$ doublets,
  - $O(100{,}000)$ triplets

Score threshold gives a smarter triplet classifier

- **Excellent performance:**
  - 99.09% efficiency

FYI

$$\text{Efficiency} = \frac{\text{\# triplets classified as true}}{\text{Total \# of true triplets}}$$

- **Solution:** Cut doublet input before triplet construction
  - Doublet threshold of 0.04 retains 98% efficiency
  - Reduces doublets $O(28{,}000) \rightarrow O(6{,}000)$
  - We thus have a sustainable process to N-plet GNN

Howdy from the Exa.TrkX Collaboration!

- **Mission**

Optimization, performance and validation studies of ML approaches to the Exascale tracking problem, to enable production-level tracking on next-generation detector systems.

- **People**
  - *Caltech*: Joosep Pata, Maria Spiropulu, Jean-Roch Vlimant, Alexander Zlokapa
  - *Cincinnati*: Adam Aurisano, Jeremy Hewes
  - *FNAL*: Giuseppe Cerati, Lindsey Gray, Thomas Klijnsma, Jim Kowalkowski, Gabriel Perdue, Panagiotis Spentzouris
  - *LBNL*: Paolo Calafiura (PI), Nicholas Choma, Steve Farrell, Xiangyang Ju, Daniel Murnane, Prabhat
  - *ORNL*: Aristeidis Tsaris
  - *SLAC*: Kasuhiro Terao, Tracy Usher

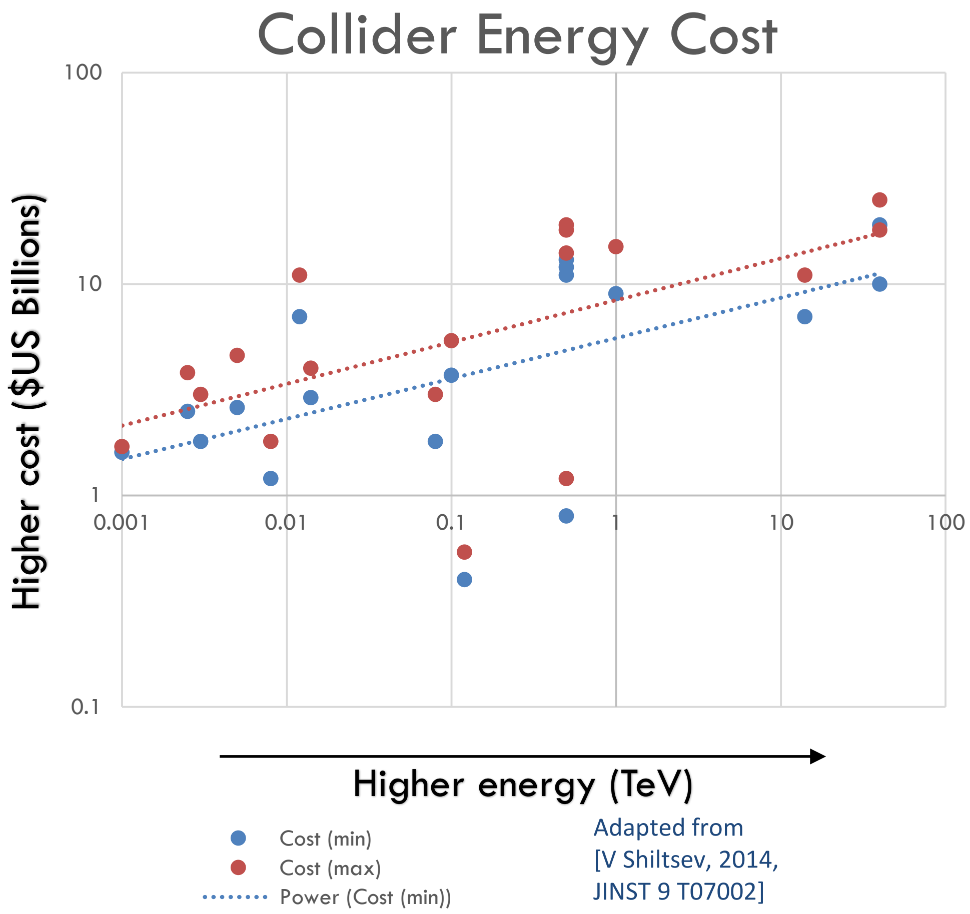
Next Steps

- Full pipeline of Embedded Graph Building → Doublet Classifier → Triplet Classifier
- Leverage multi-GPU/multi-node distributed training and inference
- Transfer as much data pre/post-processing to GPUs/multi-process as possible (e.g. RAPIDS, CuPy, Numba)
- Optimise model hyperparameters (e.g. Ray Tune, Weights & Biases)
- Explore other model architectures (e.g. ?)
- Investigate adding more features to classifiers (e.g. embedded space co-ordinates, detector pixel information)

BACKUP

Discovering new physics is getting harder and harder

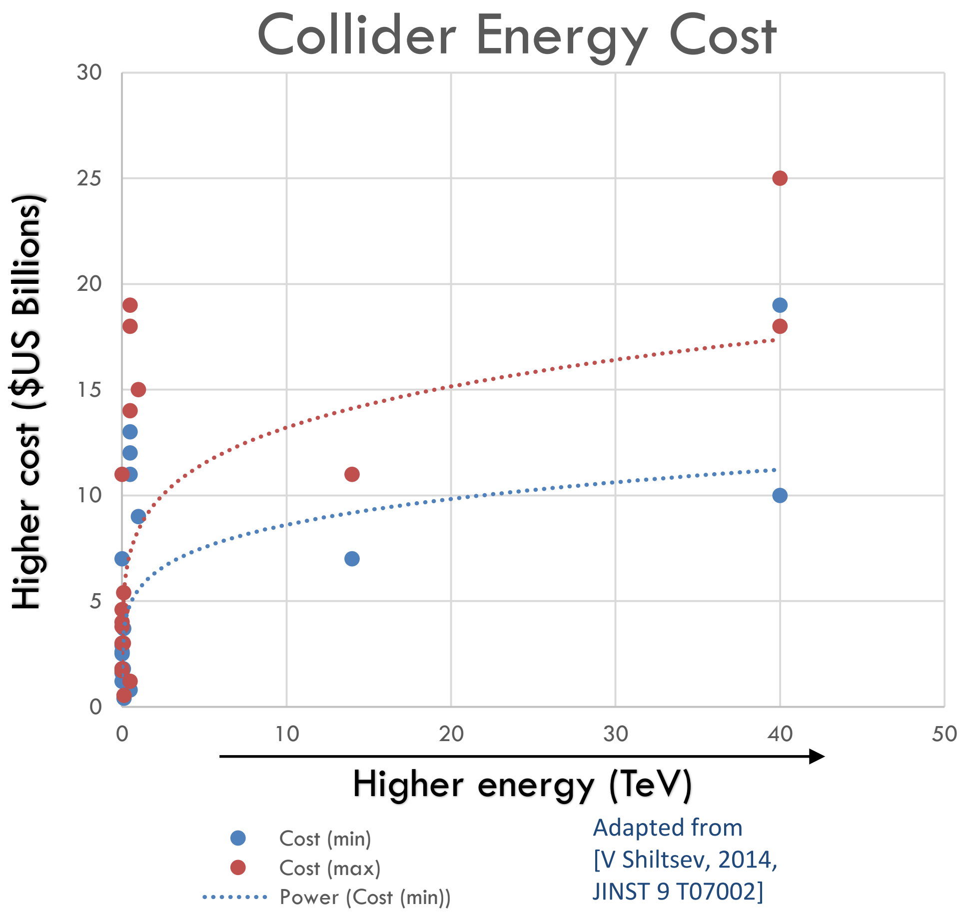
- New physics needs high energy
- Discovery cost is increasing with energy scale (LHC = $4.46 Billion)



Adapted from [V Shiltsev, 2014, JINST 9 T07002]

Discovering new physics is getting harder and harder

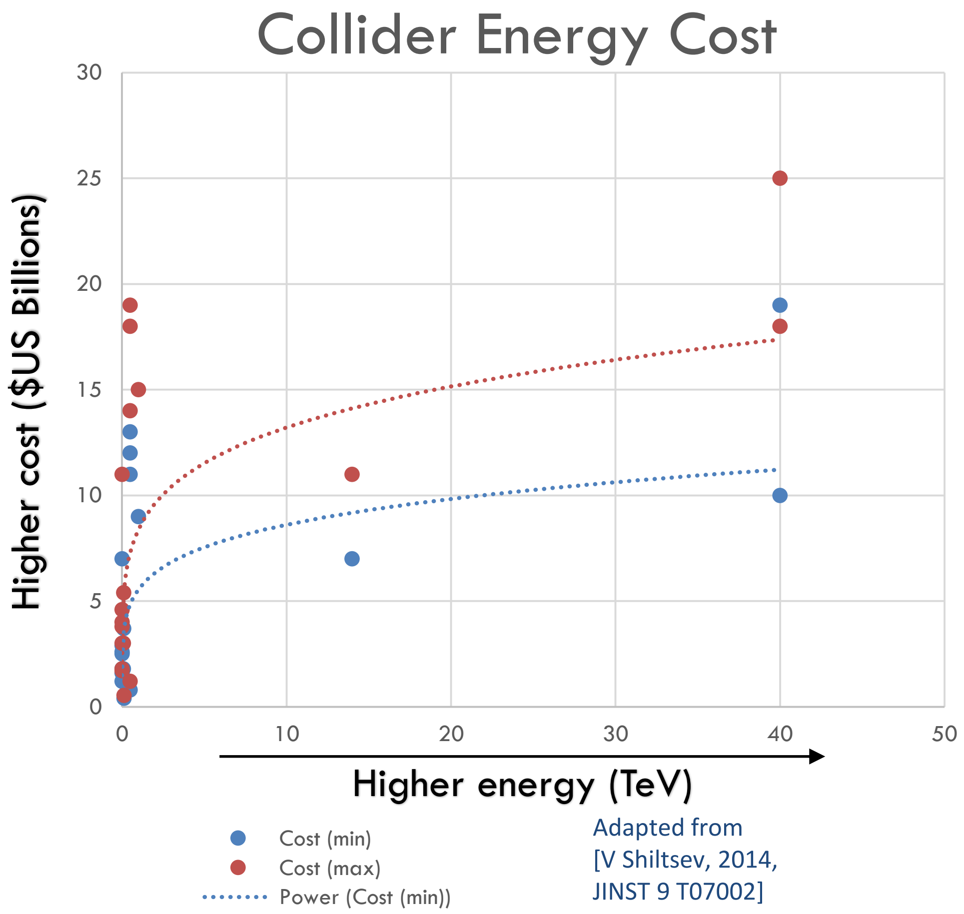
- New physics needs high energy
- Discovery cost is increasing with energy scale (LHC = $4.46 Billion)
- **Maybe there's hope: *Less than linear* increase!**



Adapted from [V Shiltsev, 2014, JINST 9 T07002]

Discovering new physics is getting harder and harder

- New physics needs high energy
- Discovery cost is increasing with energy scale   (LHC = $4.46 Billion)
- Maybe there's hope: *Less than linear* increase!
- **Nothing is that easy…**
- **With LHC run 3 (2021-25), more energy, more problems**

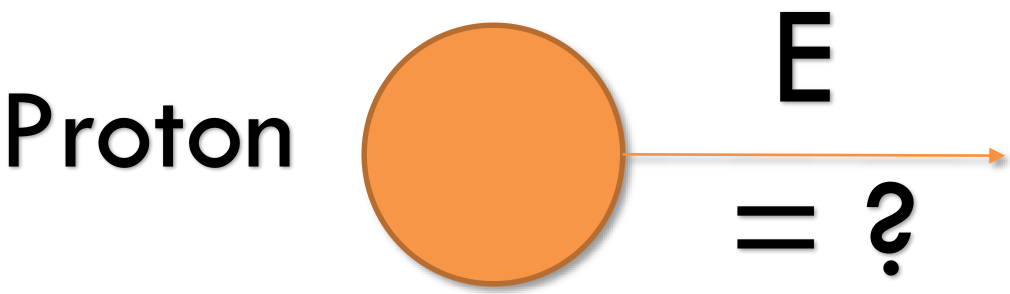


Adapted from [V Shiltsev, 2014, JINST 9 T07002]

But smashing things is not really the aim

- Dream is to produce a lone {Higgs boson, tau lepton, vector boson, …} to study its properties
- These are heavy (e.g. Higgs = 133 x Proton), so we need to introduce high energy to produce them

Ingredient 1:

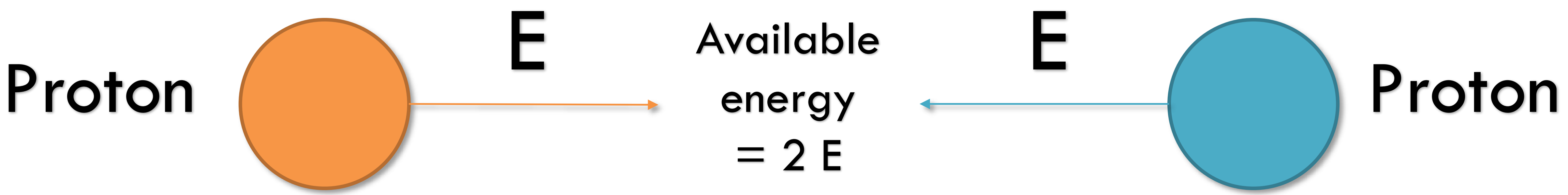


Relativity: It doesn't know its velocity/kinetic energy

But smashing things is not really the aim

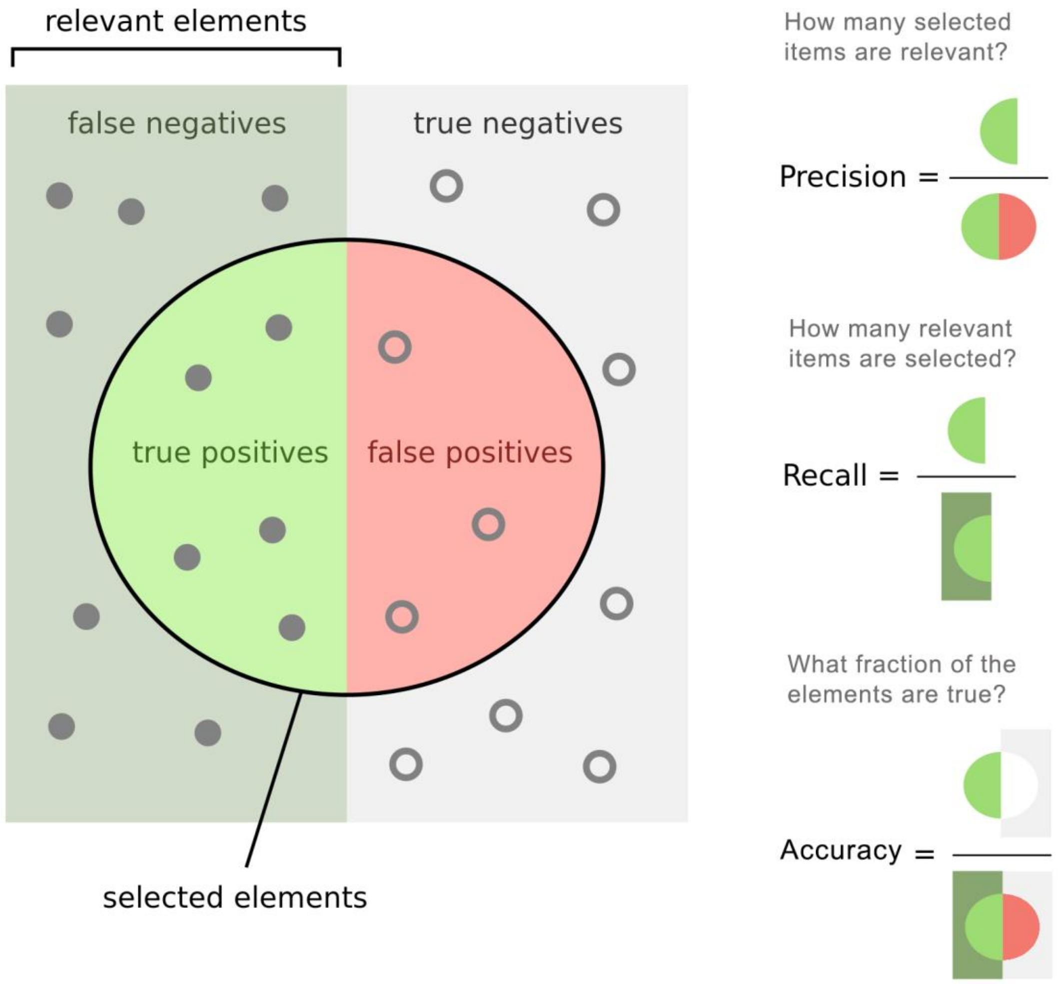
- Dream is to produce a lone {Higgs boson, tau lepton, vector boson, …} to study its properties
- These are heavy (e.g. Higgs = 133 x Proton), so we need to introduce high energy to produce them

Ingredient 1: Need to introduce…. Ingredient 2:



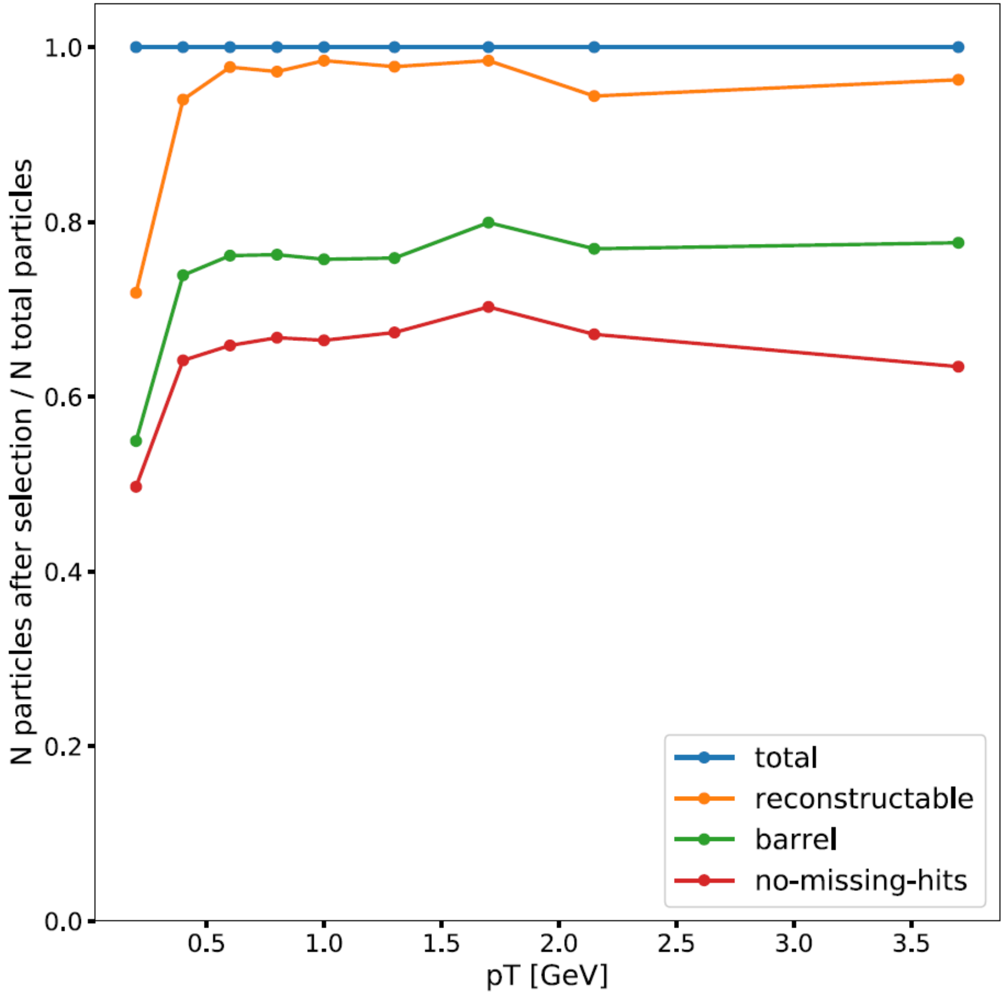
Aside: quick notation

- Recall ≡ Efficiency
- Precision ≡ Purity

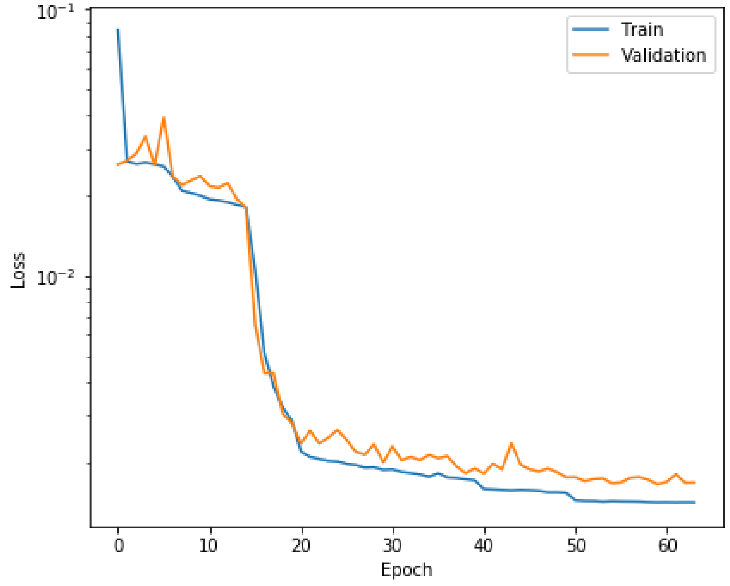


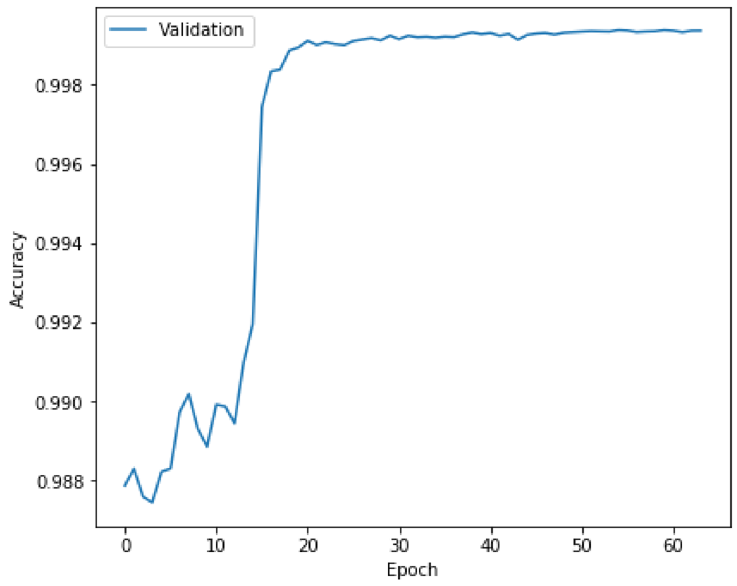
Graph construction

- Start with hits from the TrackML Challenge
- Detector geometry
- We select hits from barrel
- Other heuristical cuts: no doubles, implicit pT cut (from TrackML)



Triplet Classifier





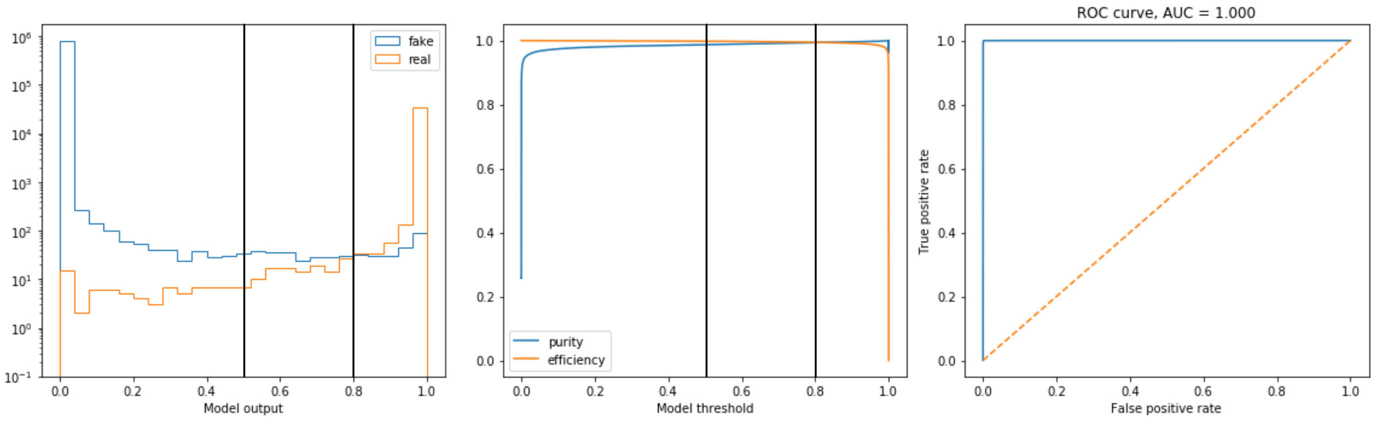
Architecture:

1. Input network,
2. (Edge network, Node network) x 3,
3. Edge network out

Triplet train set: 1,920
Doublet train set: 1,920

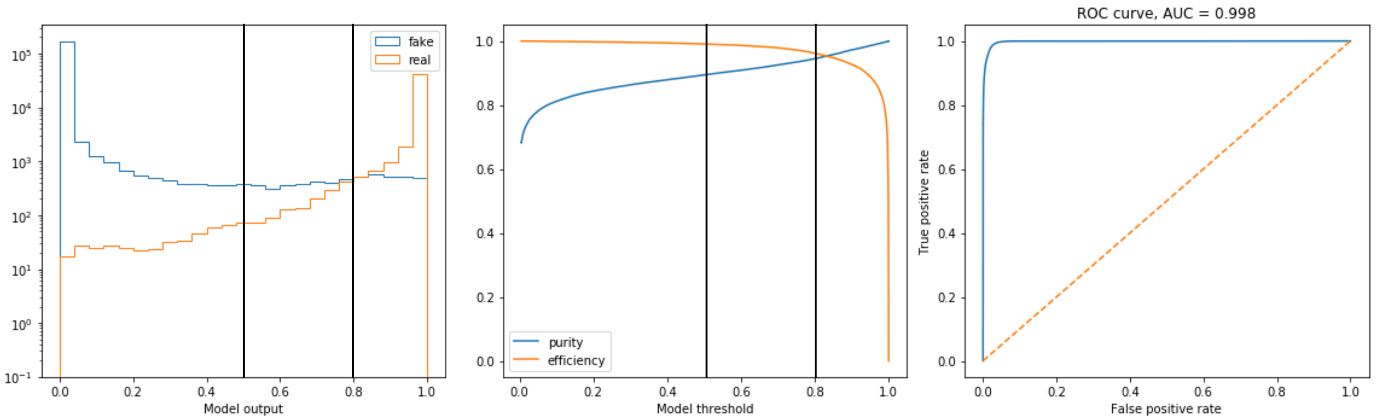
Triplet Classifier

(with $p_T > 0.5\ GeV$)



Threshold	0.5	0.8
Accuracy	0.9993	0.9998
Purity	0.9870	0.9935
Efficiency	0.9978	0.9945

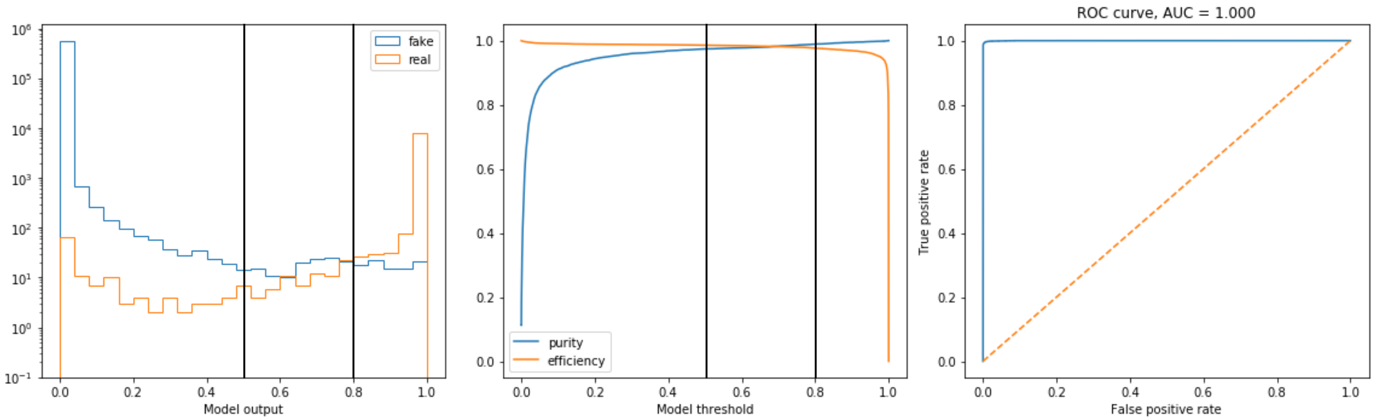
Compare with doublet classifier:



Threshold	0.5	0.8
Accuracy	0.9737	0.9804
Purity	0.8941	0.9449
Efficiency	0.9906	0.9615

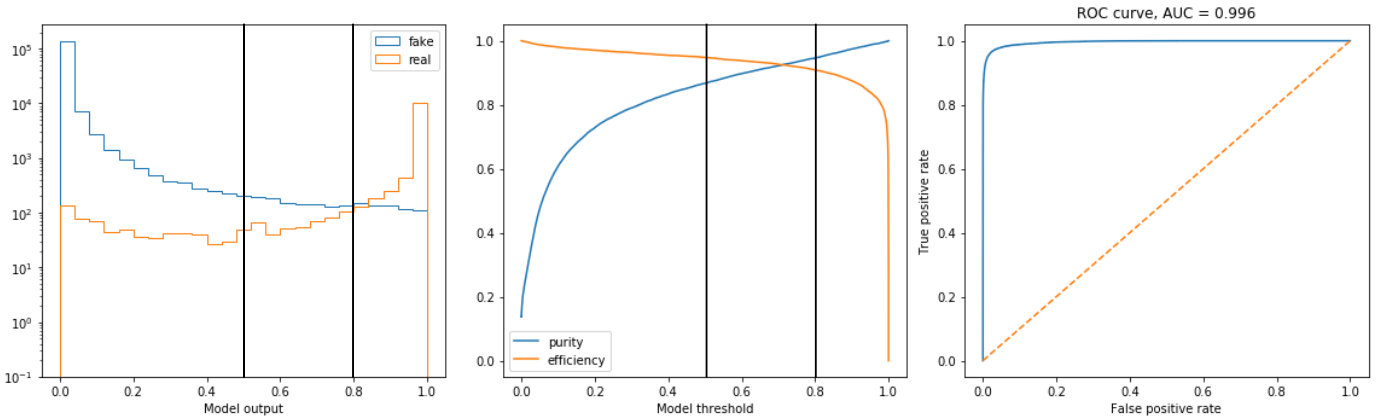
Triplet Classifier

(with no p_T cut)



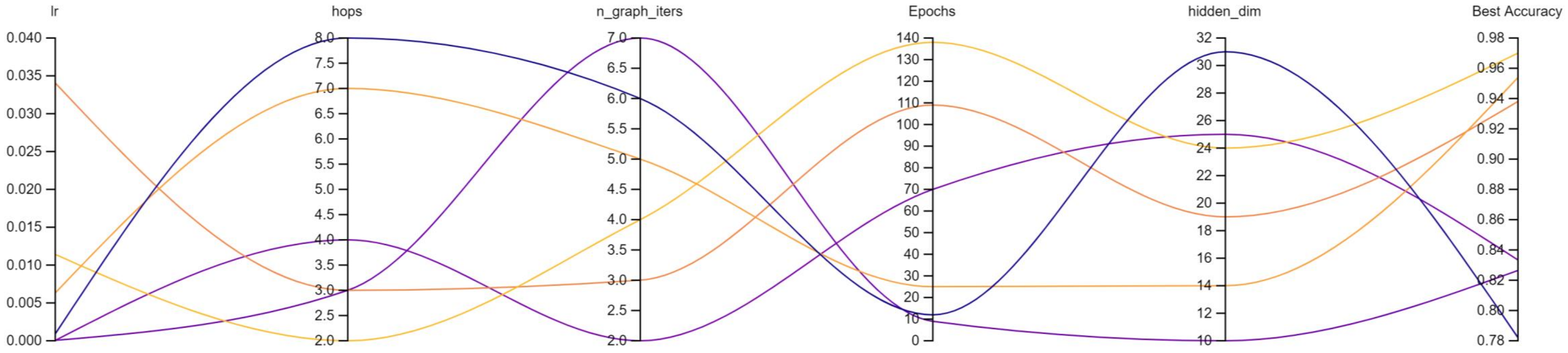
Threshold	0.5	0.8
Accuracy	0.9994	0.9995
Purity	0.9742	0.9890
Efficiency	0.9857	0.9768

Compare with doublet classifier:



Threshold	0.5	0.8
Accuracy	0.9848	0.9891
Purity	0.8668	0.9465
Efficiency	0.9478	0.9090

We can supercharge a particular GNN



1. Optimising hyperparameters with Weights & Biases [`wandb.ai`]
   (Hyperparameters are usually chosen by-hand to control the learning process)

2. Threshold on doublet scores:
   Order of magnitude improvement in triplet classifier

We could supercharge all GNNs

- We don’t know which GNN works best
- Every day sees more techniques being constructed – **flash list**
- Implementing by hand is time-consuming
- Would like a *GNN-generator*
- Call it ArchetrkX
- Represents an ML architecture (i.e. the collection of neural nets, convolutions, transformations, etc.) itself as a graph – a *computation graph*
- This graph can be randomly generated, run as a GNN on the data, and tested for accuracy
- We can optimise the best GNN architecture by applying a GNN to *this* computation graph
- Meta-GNN optimisation, all run in ArchetrkX

ArchetrkX performance

- Simple image showing the GNN architecture generation
- Preliminary results of a generated architecture
- Parallel co-ordinates example of results