HEP.TrkX
Charged Particle Tracking using Graph Neural Network

Xiangyang Ju for the HEP.TrkX project

Connecting The Dots / Intelligent Trackers 2019
Valencia, Spain

3 April, 2019

U.S. DEPARTMENT OF ENERGY

Fermilab

HEP.TrkX project

- Pilot project funded by DOE ASCR and COMP HEP
- Part of HEP CCE
- Mission: Explore deep learning techniques for track formation

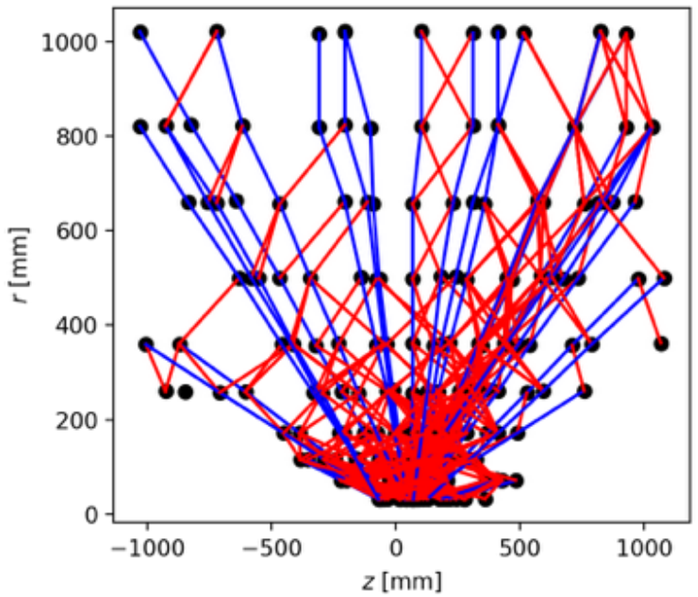
People:

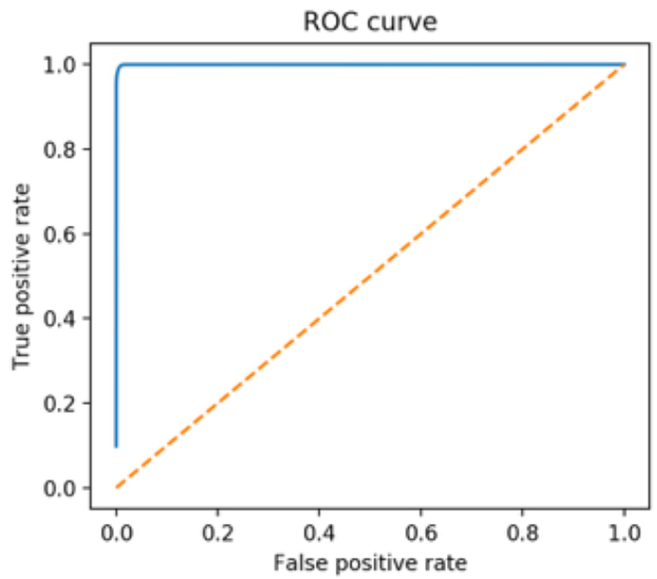
- LBNL: Paolo Calafiura, Steve Farrell, Xiangyang Ju, Prabhat,
- FNAL: Giuseppe Cerati, Lindsey Gray, Jim Kowalkowski, Panagiotis Spentzouris, Aristeidis Tsaris
- Caltech: Maria Spiropulu, Jean-Roch Vlimant, Alexander Zlokapa

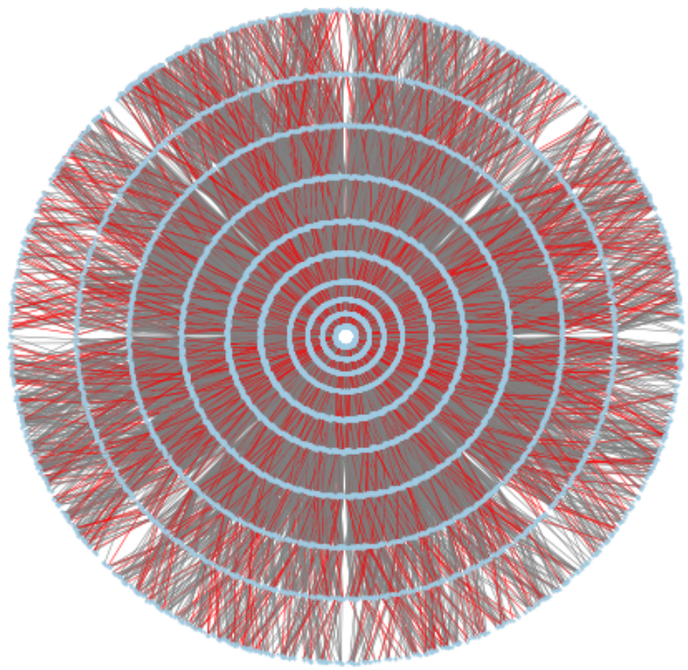
- Material available under https://heptrkx.github.io/

Introduction

In the CTD2018, Steve Farrell showed exciting performance of GNN on predicting edge scores. [link]







QCD data with μ = 10 [link]

```
Test set metrics
Accuracy:    0.9942
Purity:      0.9918
Efficiency: 0.9793
```

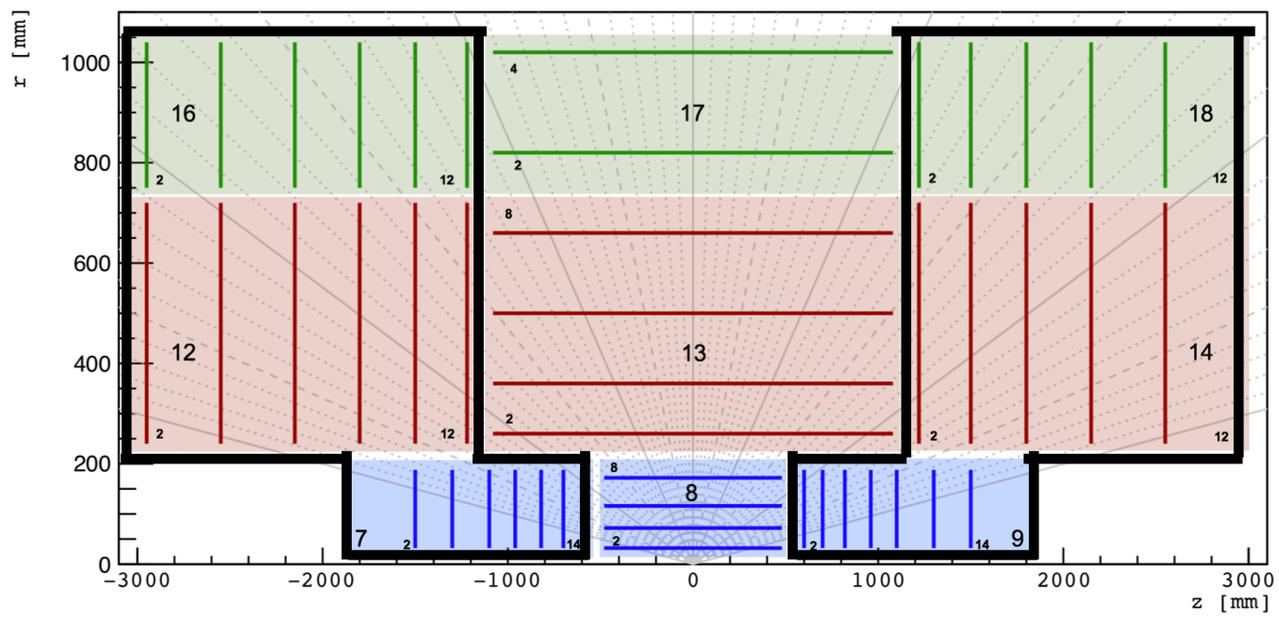
??????

Tracking ML challenge data

[link to the website]

Review of the Challenge will be given tomorrow by Andreas Salzburger [link]

The data provides the hit positions recorded in the inner detector with the geometry shown in right



Take one event as an example, there are 11700 particles, 86% are **reconstructable**, i.e. leaving hits in the detector

Our goal is to reconstruct tracks from these hits using GNN

The selected data is our starting point:

- Hits recorded in volume [8, 13, 17], basically barrel region
- Particles leaving fully connected tracks in the detector, i.e. no missing hits

Track reconstructions

Start with selected hits

↓

Graph Formation

↓

GNN

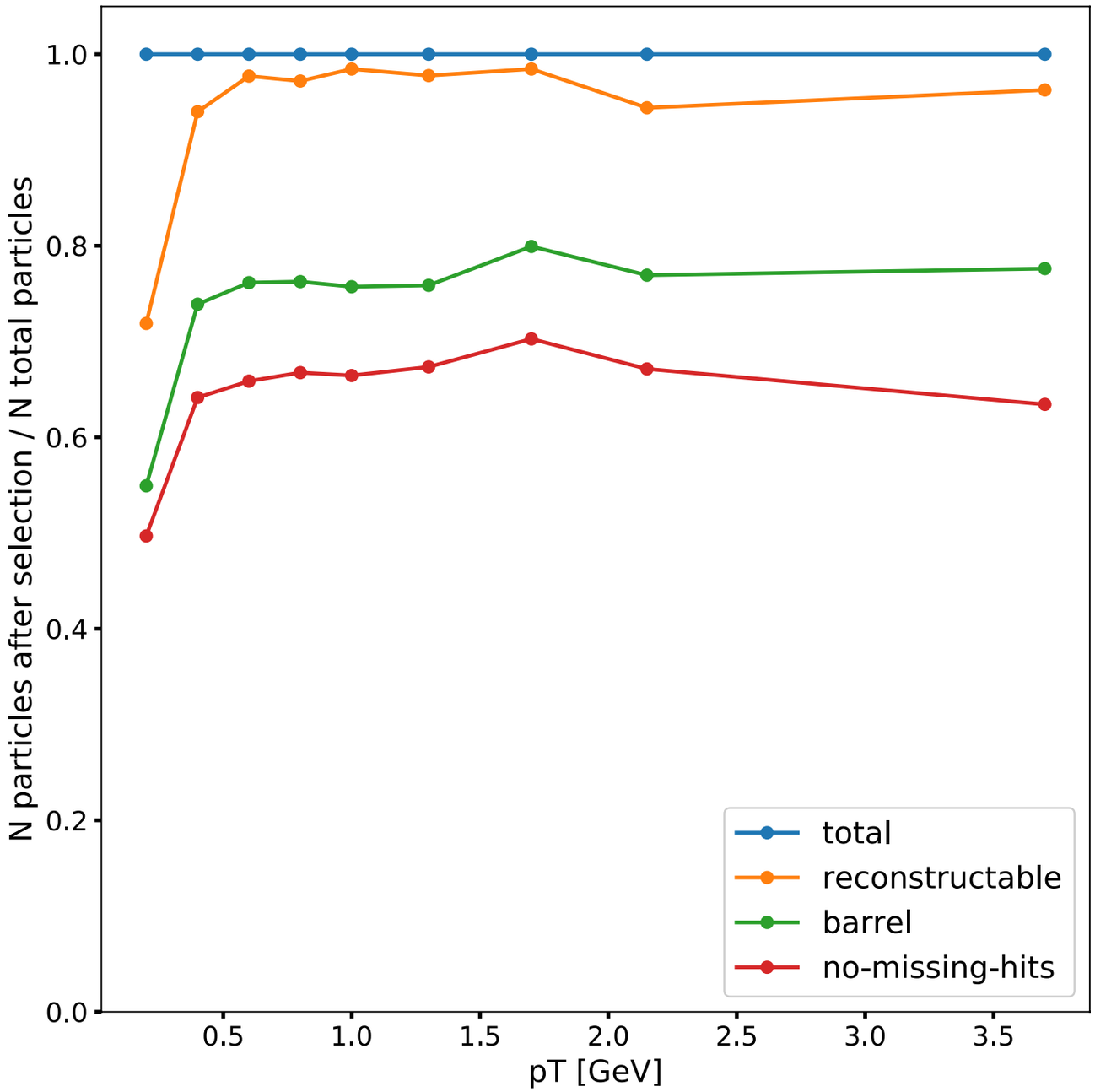
↓

CTD

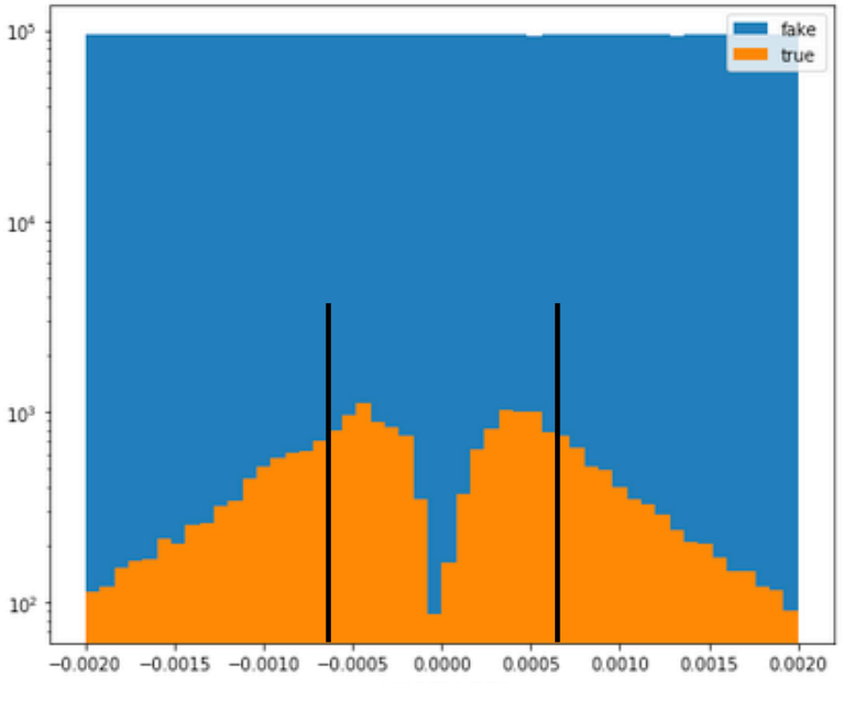
↓

Trk Refinement

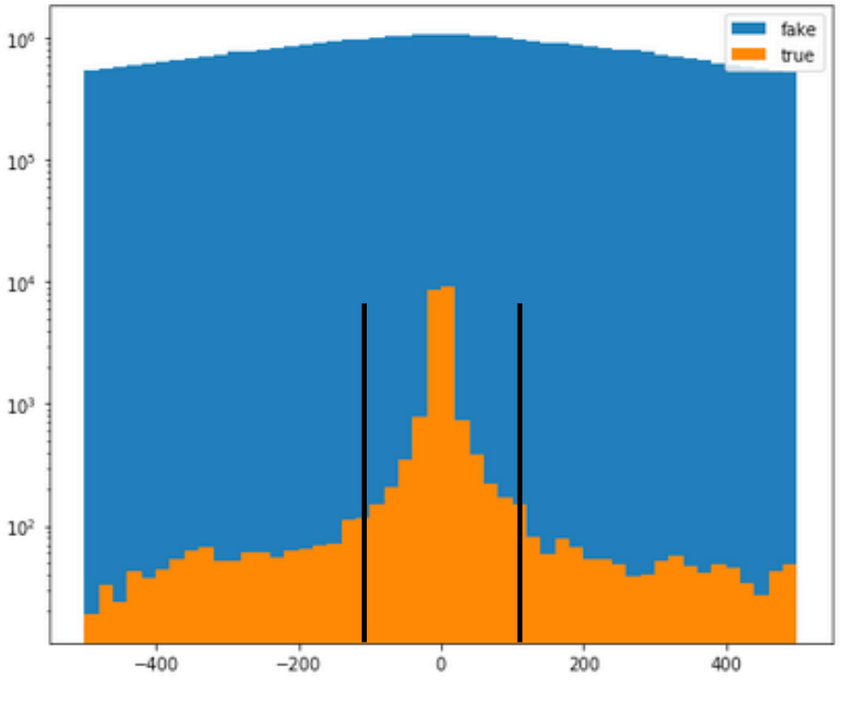
# of particles after each selection (accumulated) over the total # of particles in the event



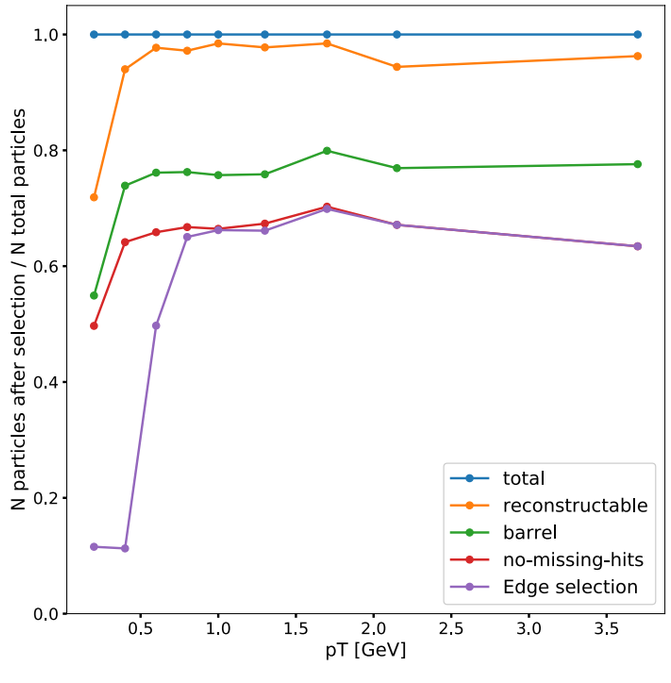
Graph Formation: edge selection



$\Delta\phi/\Delta r$ [rad/mm]

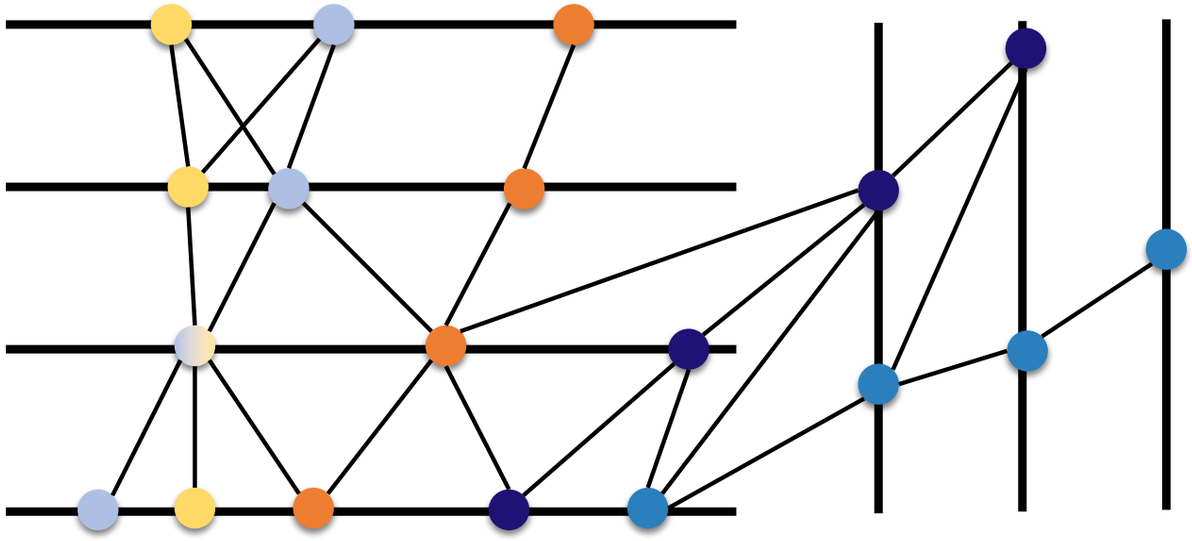


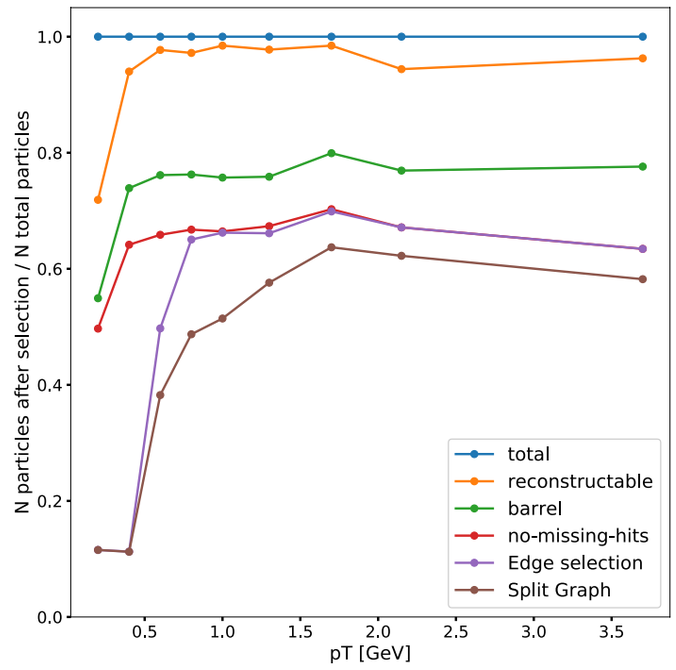
z_0 [mm]



- Make initial edges from hits in adjacent layers
- Use a simple selection to prune away fake edges
  - $\Delta\phi/\Delta r < 0.0006$,
  - z_0 (intercept of the line passing through the two hits) < 100 mm
- Tuned to be efficient for tracks with pT > 1 GeV

Graph Formation: construct graph from pairs

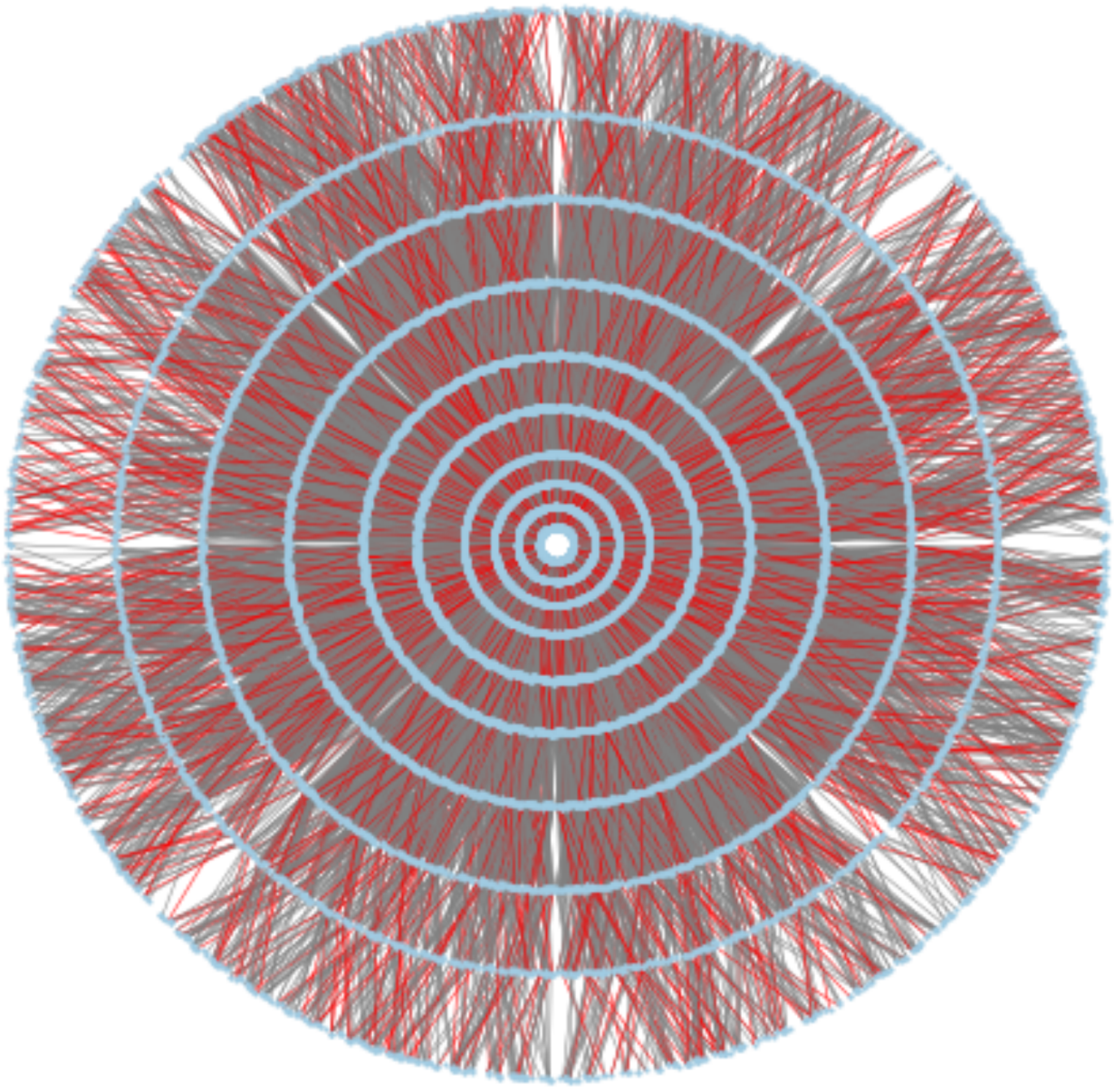


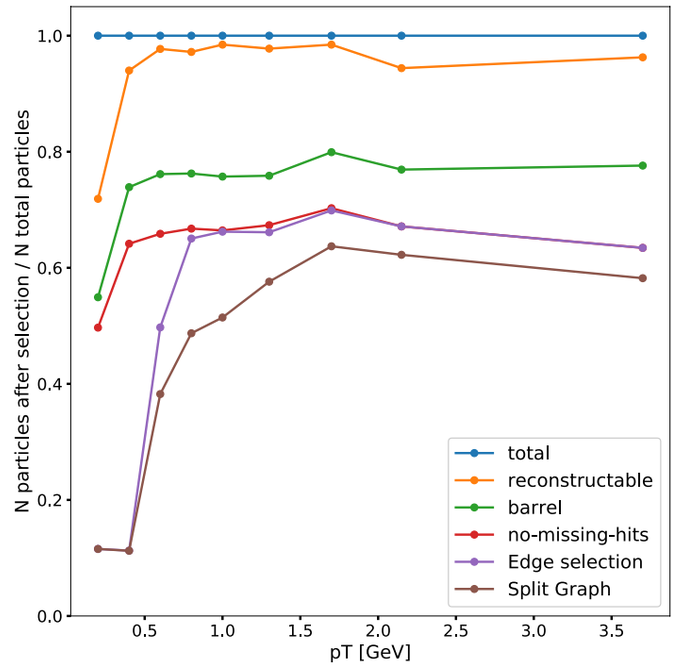


- Construct a directed graph, flowing inside-out
  - Split into 16 subgraphs, **8 ϕ bins and 2 η bins**
    - need smart algorithm to deal with hits in boundaries
  - **input node features**: [r, ϕ, z]
  - Edges belong to a track assigned with score of 1, 0 otherwise

Graph for one event

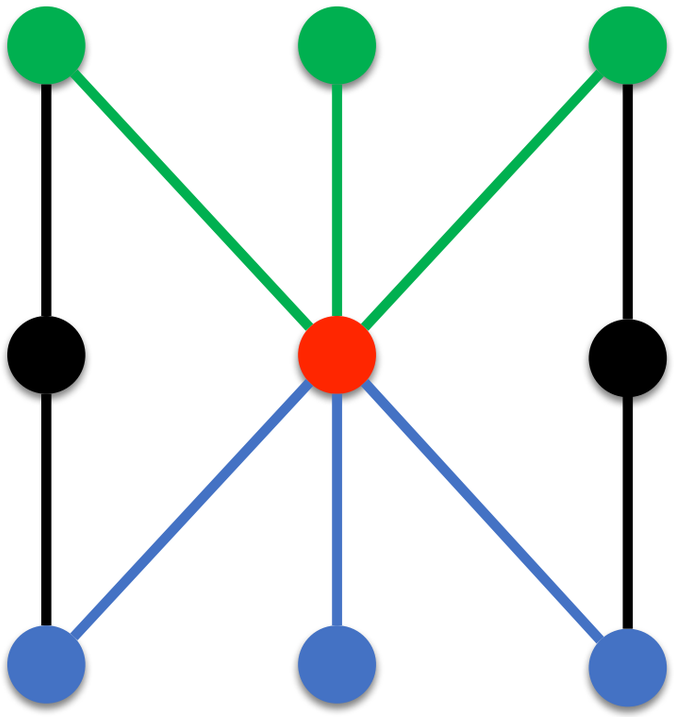
After all previous selections





- About 160k edges, 92% are fake (in gray)
- 8 gaps result from 8 sections in ϕ
- Can GNN find the 13k true edges out of the 147k fake ones?

GNN Architecture



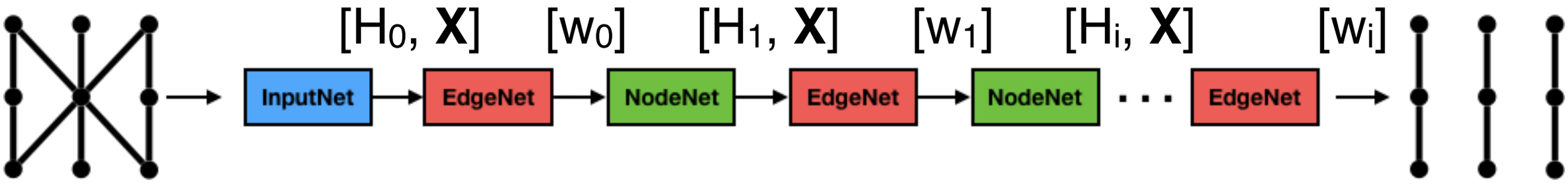
Three components operate on graph:

- **Input network** computes hidden node features
- **Edge network** computes **edge scores** from node features
- **Node network** computes hidden node features from aggregated **weighted incoming** and **outgoing** node features

Incoming and outgoing nodes with higher weights get more “attention”

Putting them together

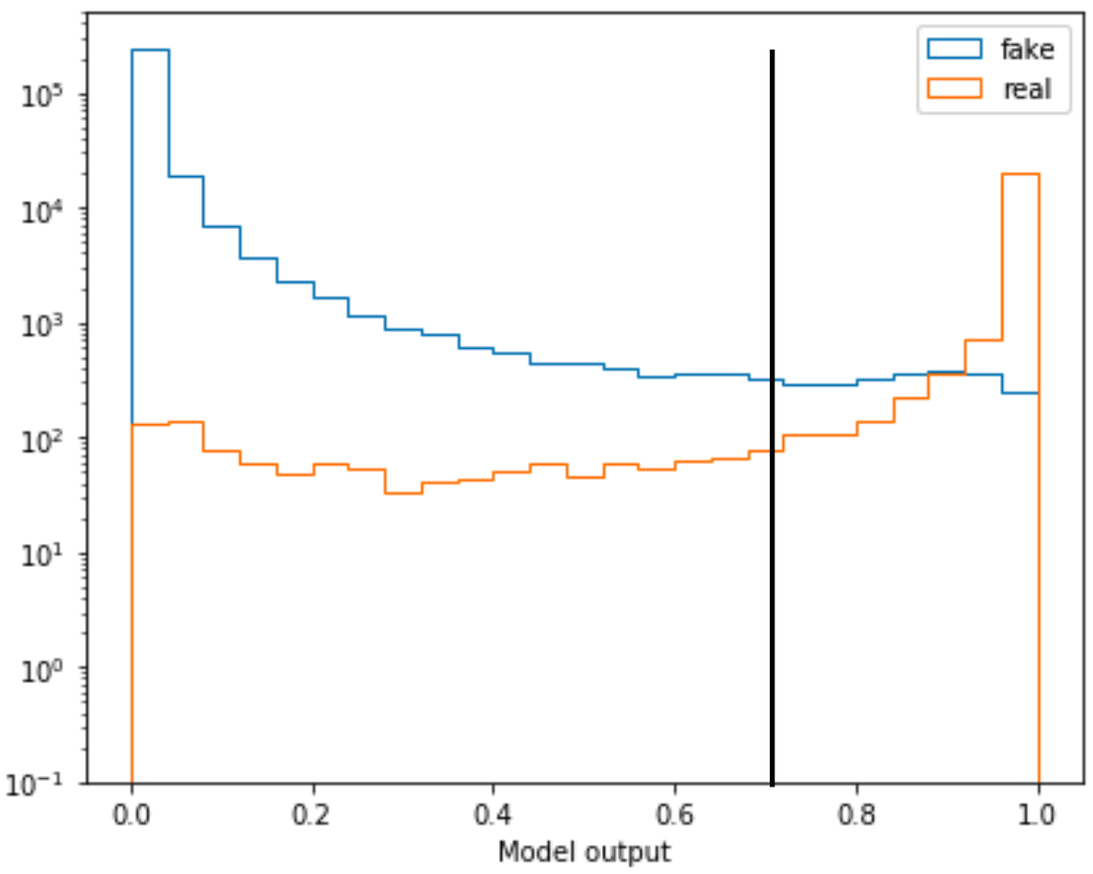
H: hidden states/features, X: input node features



- **EdgeNet and NodeNet iteratively applied 8 times, so 8 iterations**
- Message passed with the “attention mechanism”
- Hidden node features carry embedded track information for Edge Network to make predictions

GNN Output and performance

- ~43k tunable parameters in pytorch
- Trained on NVIDIA V100 'Volta' GPU for about 60 epochs
- Weighted loss function



With a threshold of 0.7:

Edge Efficiency: 95.2%

Edge Purity: 90.2%

$$\text{Efficiency} = \frac{\#\text{ of True Edges passed the threshold}}{\#\text{ of total True Edges}}$$

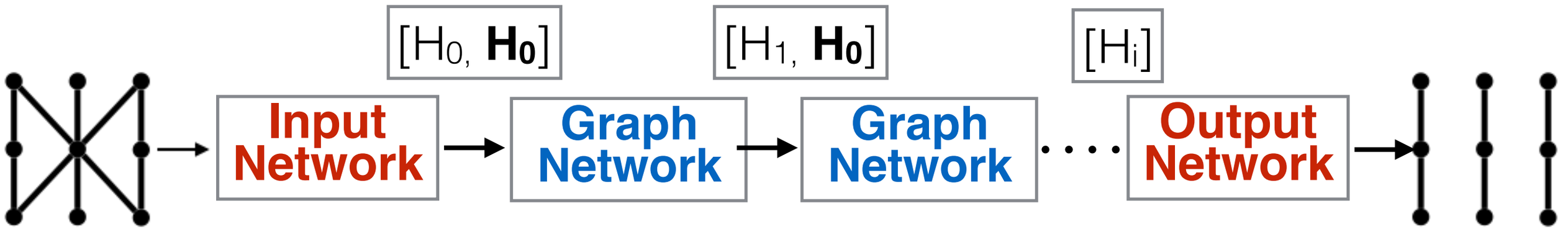
$$\text{Purity} = \frac{\#\text{ of True Edges passed the threshold}}{\#\text{ of total Edges passed the threshold}}$$

- One can use higher threshold to gain purity at the cost of less efficiency.

We are exploring other GNN architectures to push the performance further.

Following example uses graph_nets library from DeepMind and a model resembling the Interaction Networks [link]

Alternative implementation of GNN

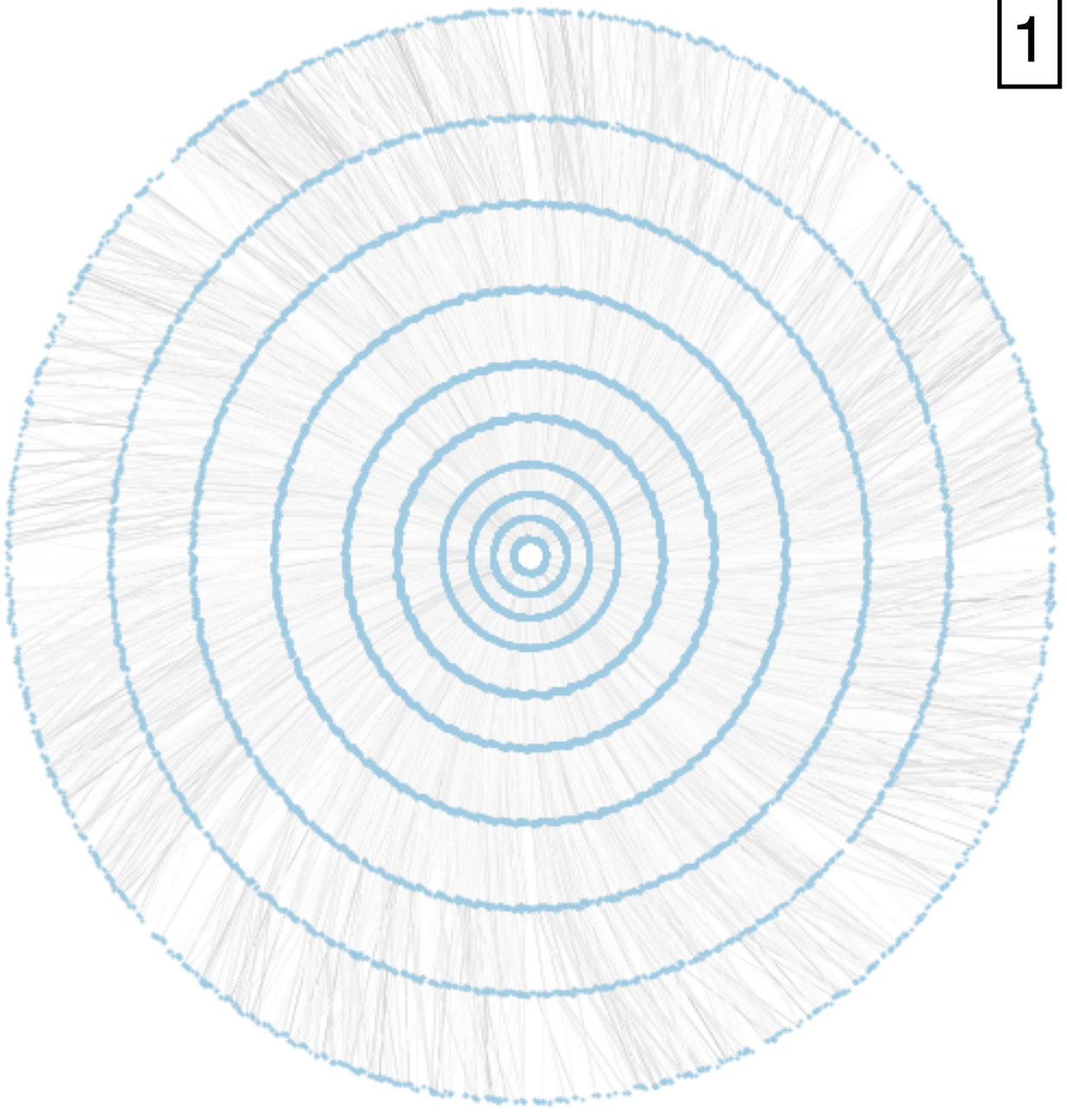


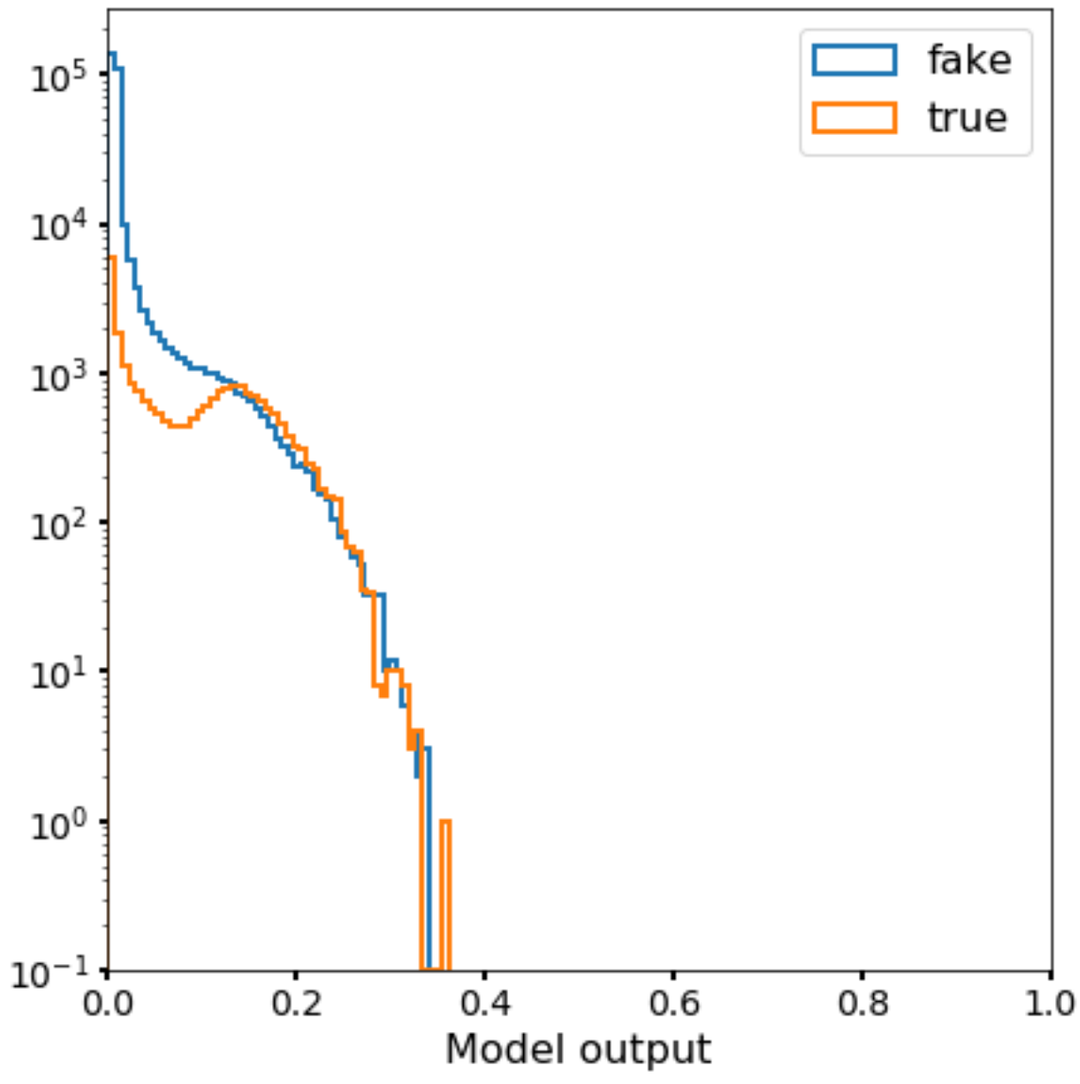
Differences:

- Edge features provided in the input
- Alternate message passing implementation:
  - No explicit attention mechanism
  - Edge features are computed from node features and then summed across all neighbors
- Output Network computes final edge scores
- Bigger, deeper model (266k parameters)

We can visualize the intermediate outputs of the model

Predicted score improves after each iteration

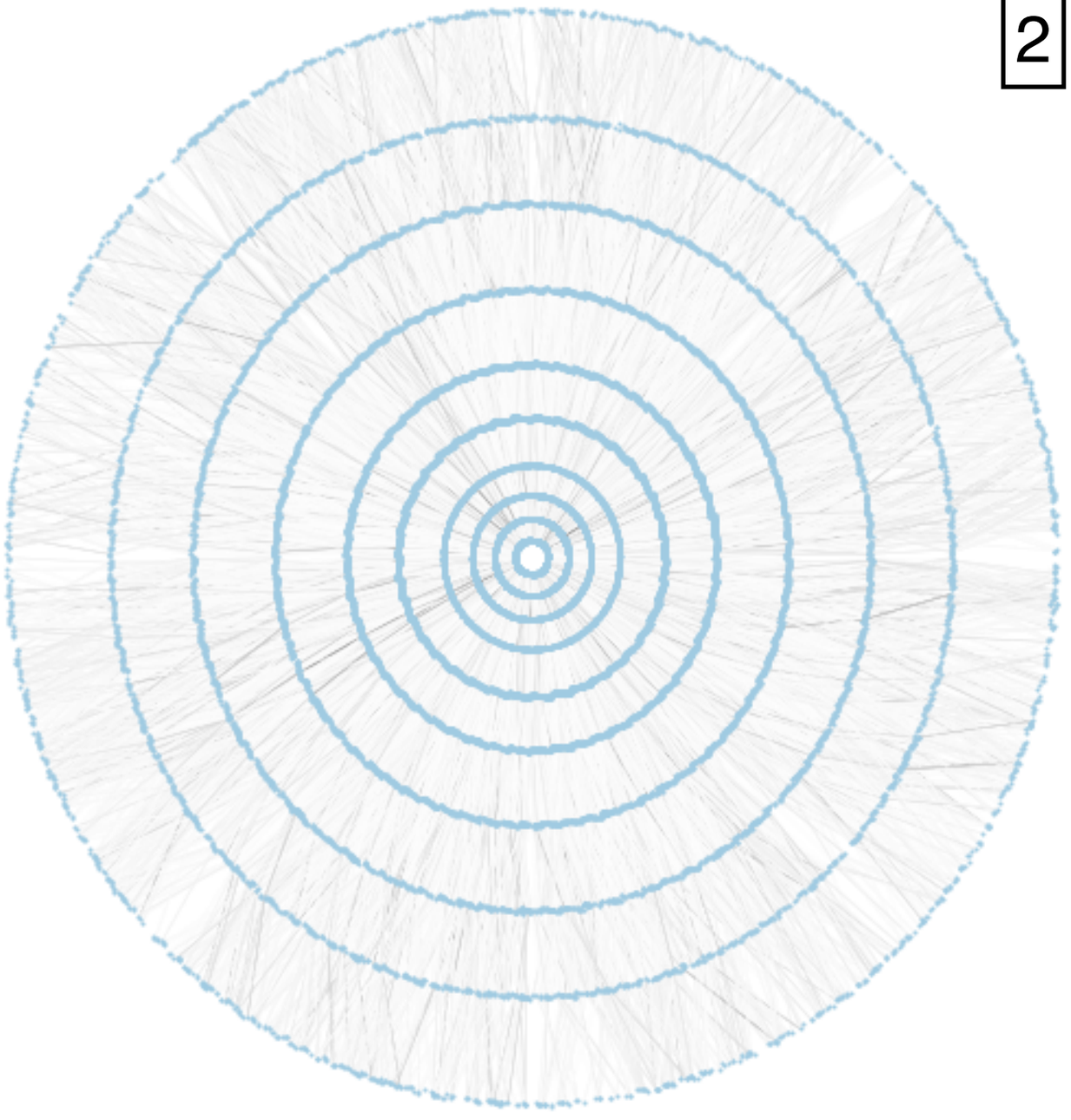


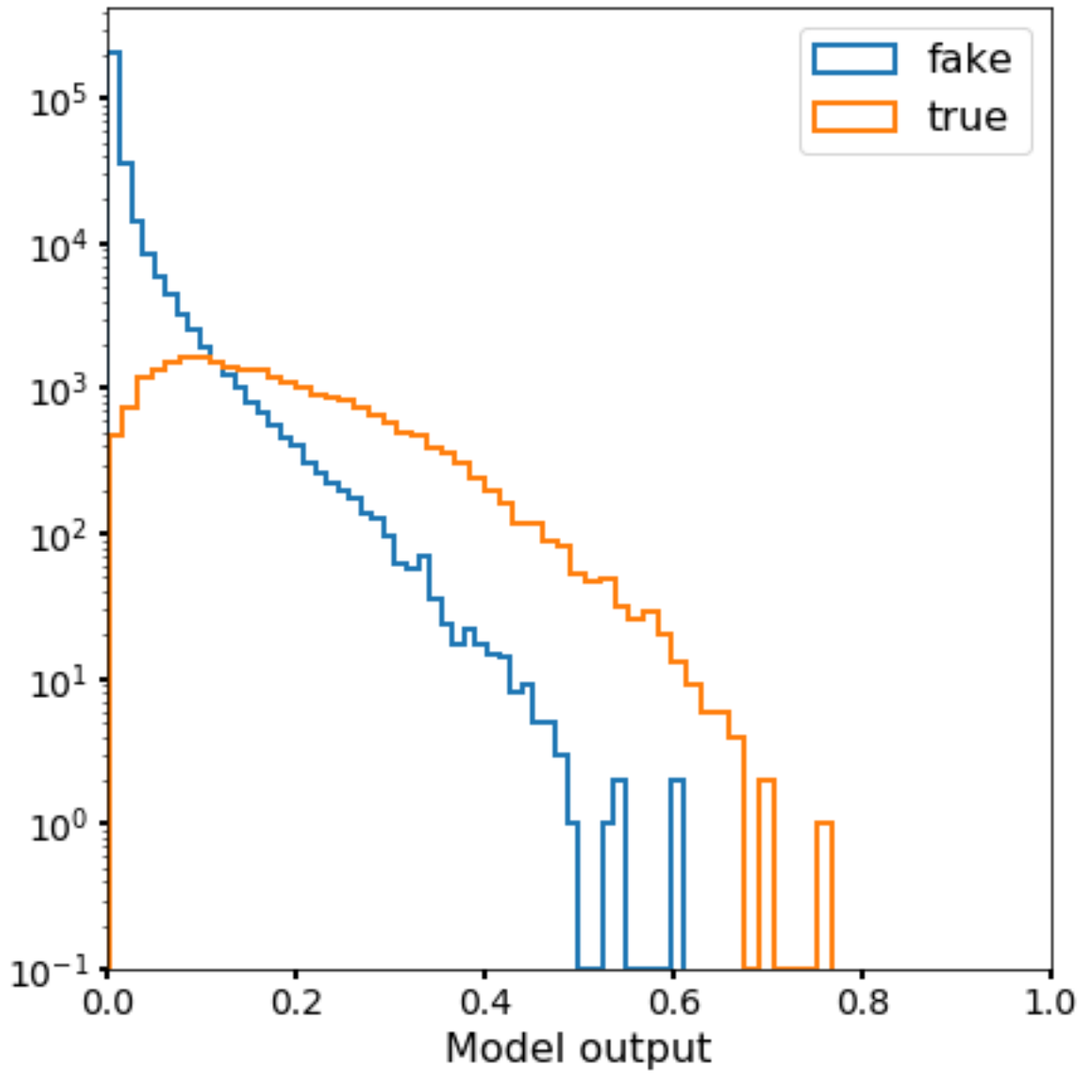


Edges with higher scores are darker than that with lower scores

Edges with scores < 0.01 are removed for visualization purpose.

Predicted score improves after each iteration

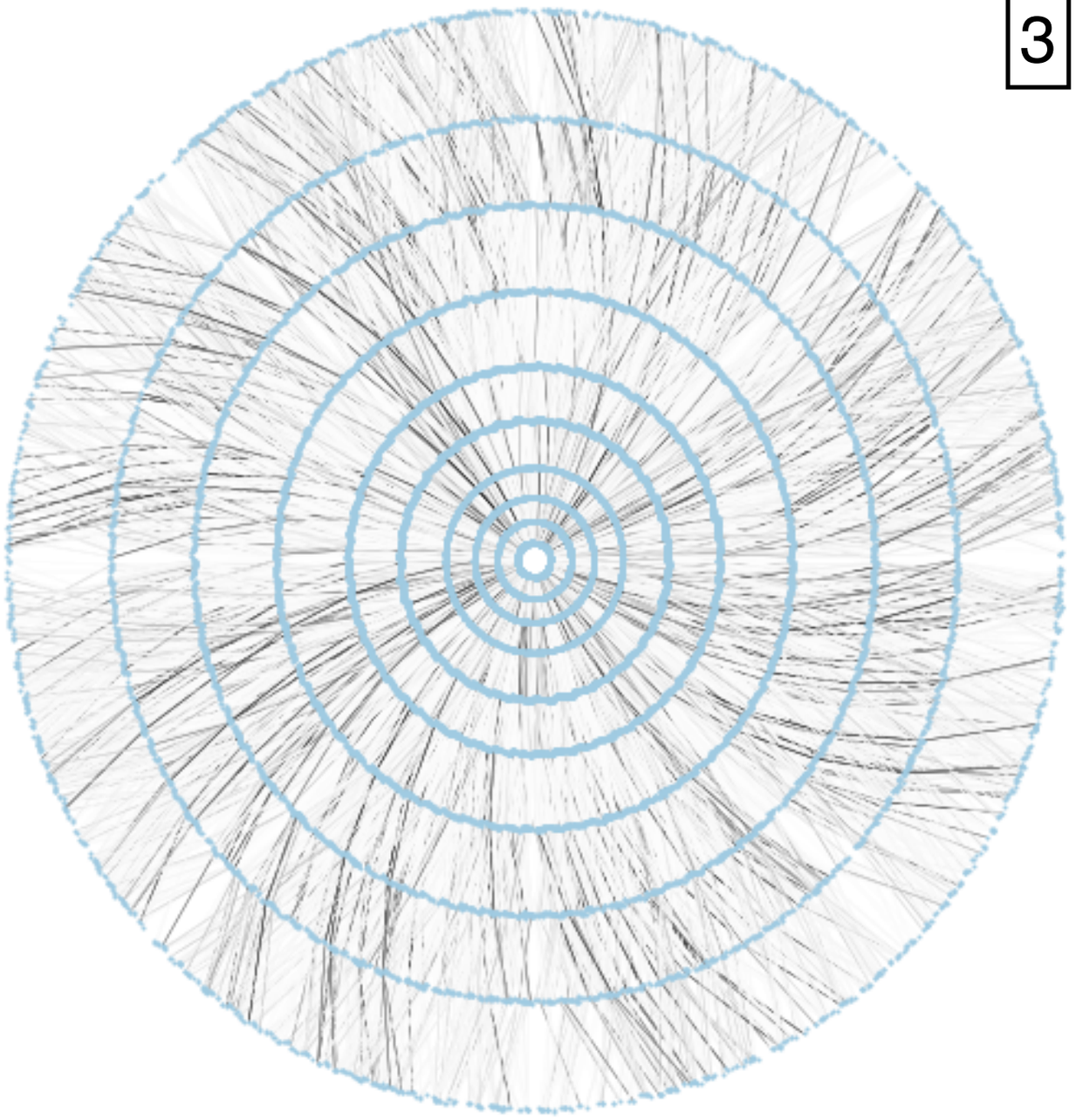


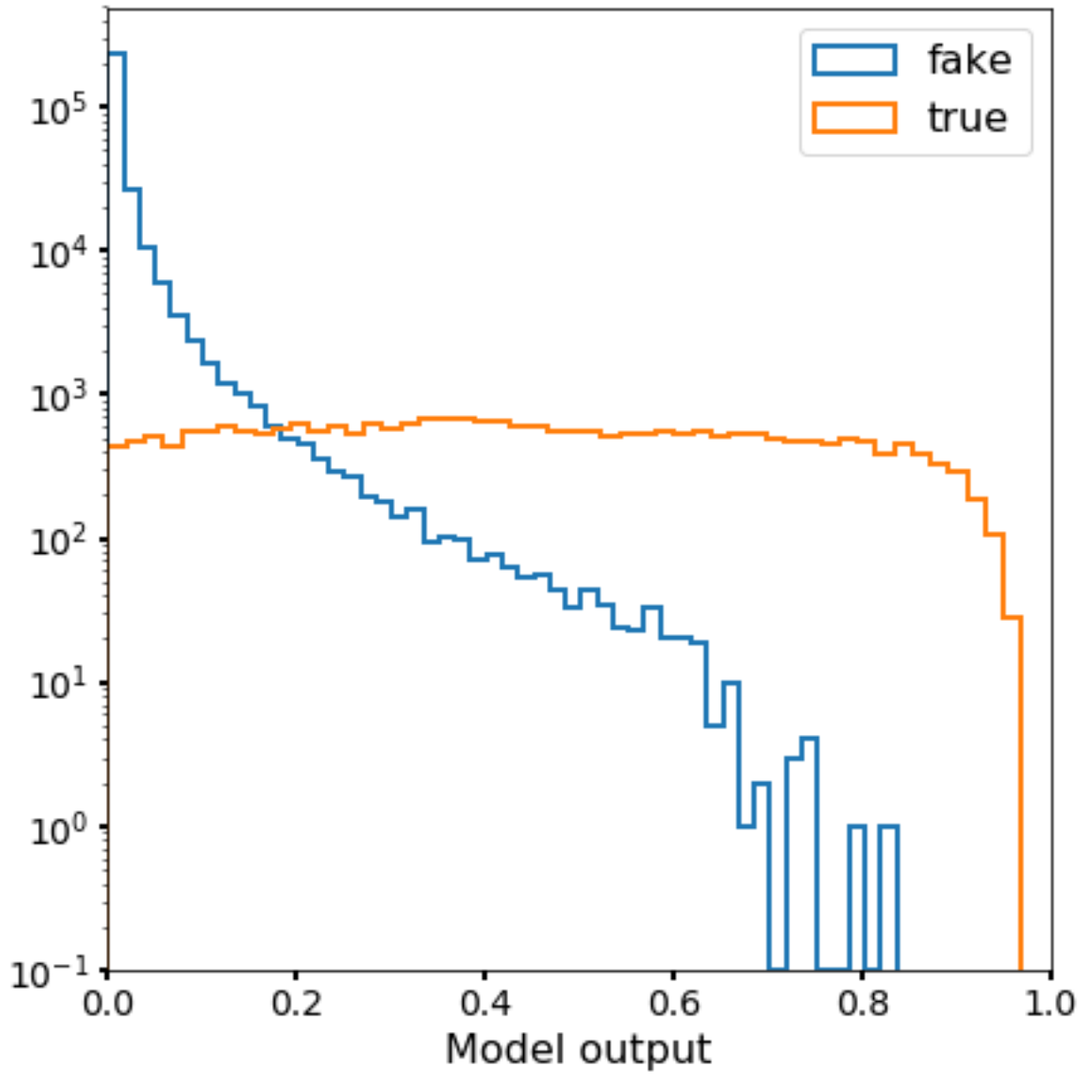


Edges with higher scores are darker than that with lower scores

Edges with scores < 0.01 are removed for visualization purpose.

Predicted score improves after each iteration

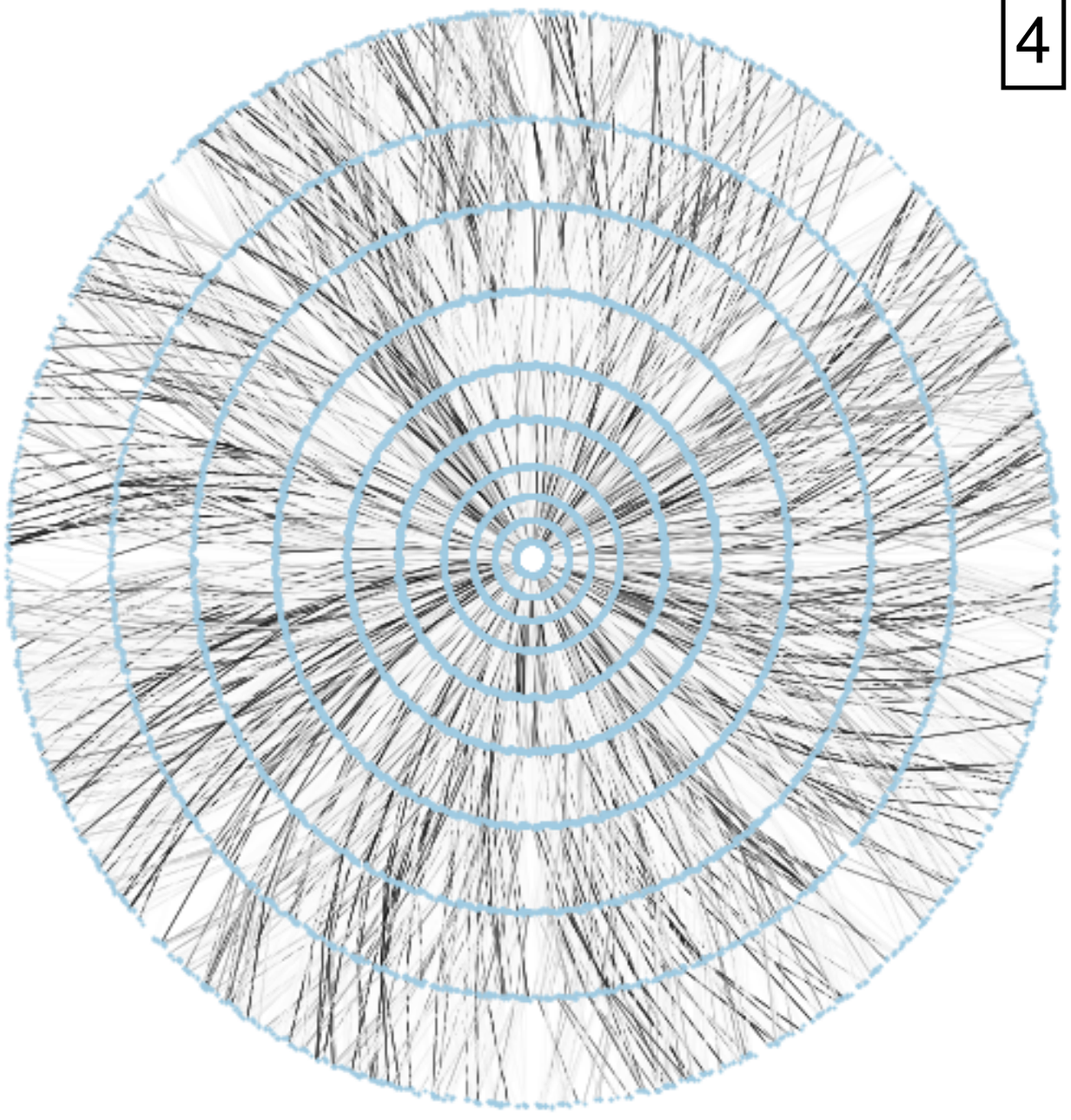


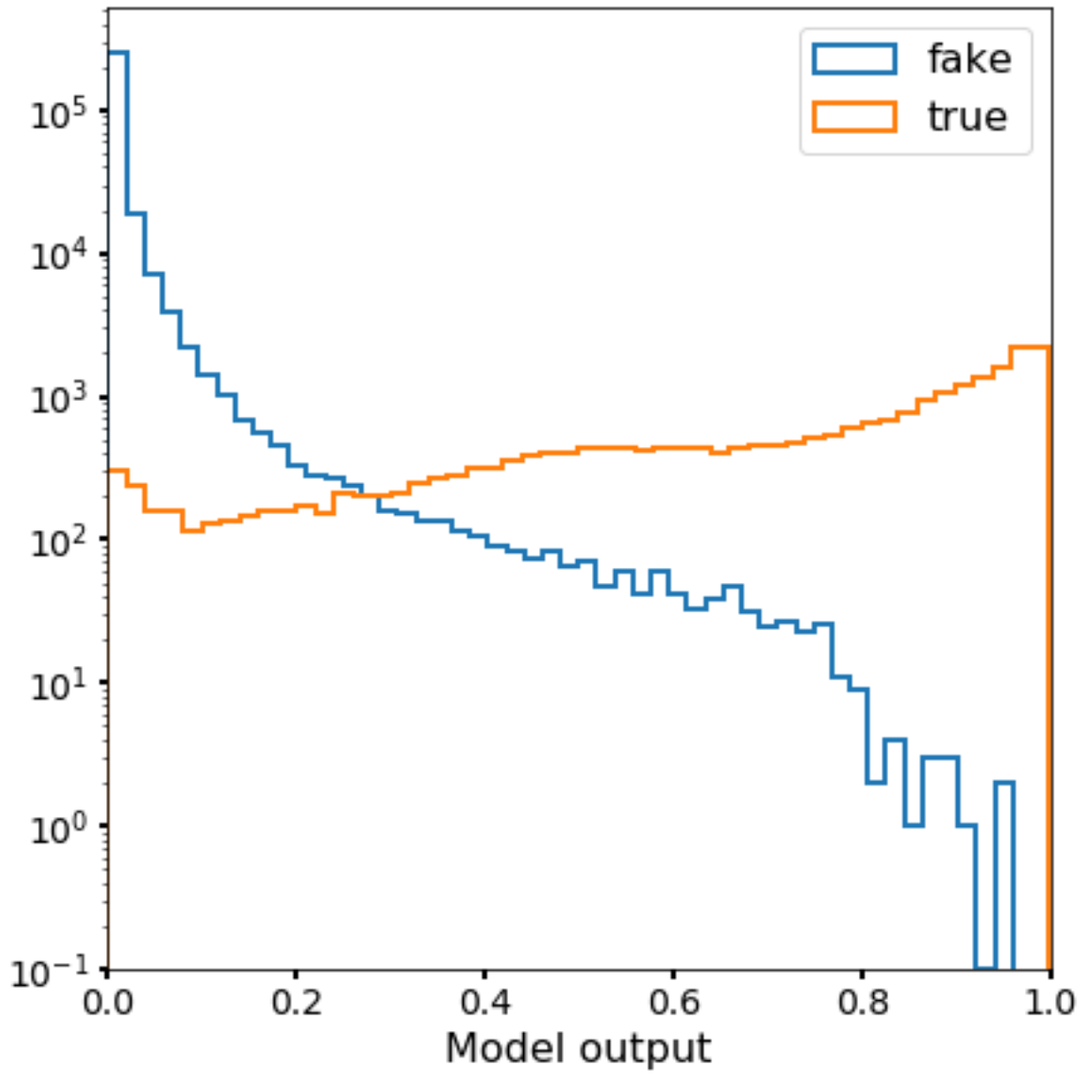


Edges with higher scores are darker than that with lower scores

Edges with scores < 0.01 are removed for visualization purpose.

Predicted score improves after each iteration

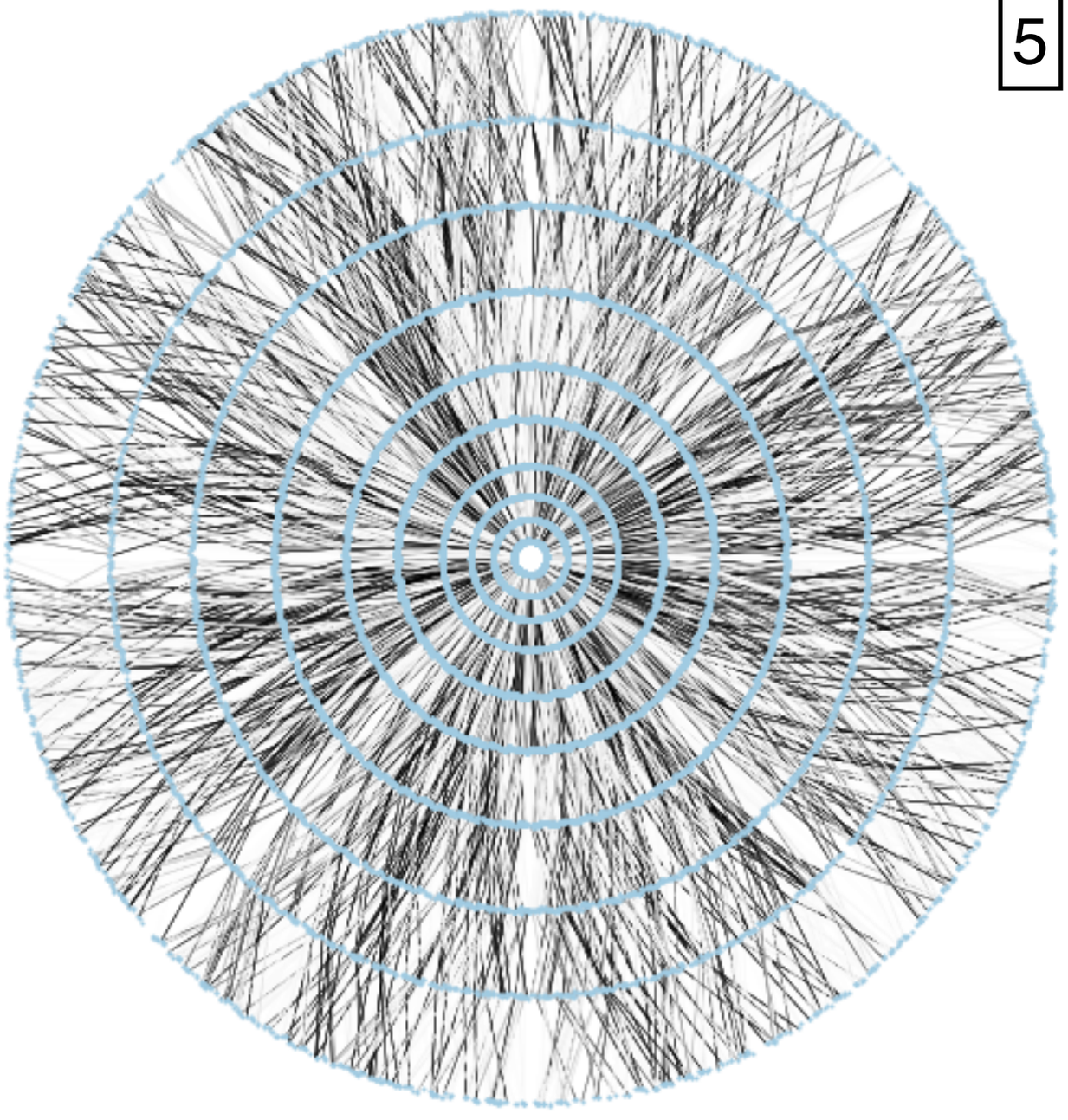


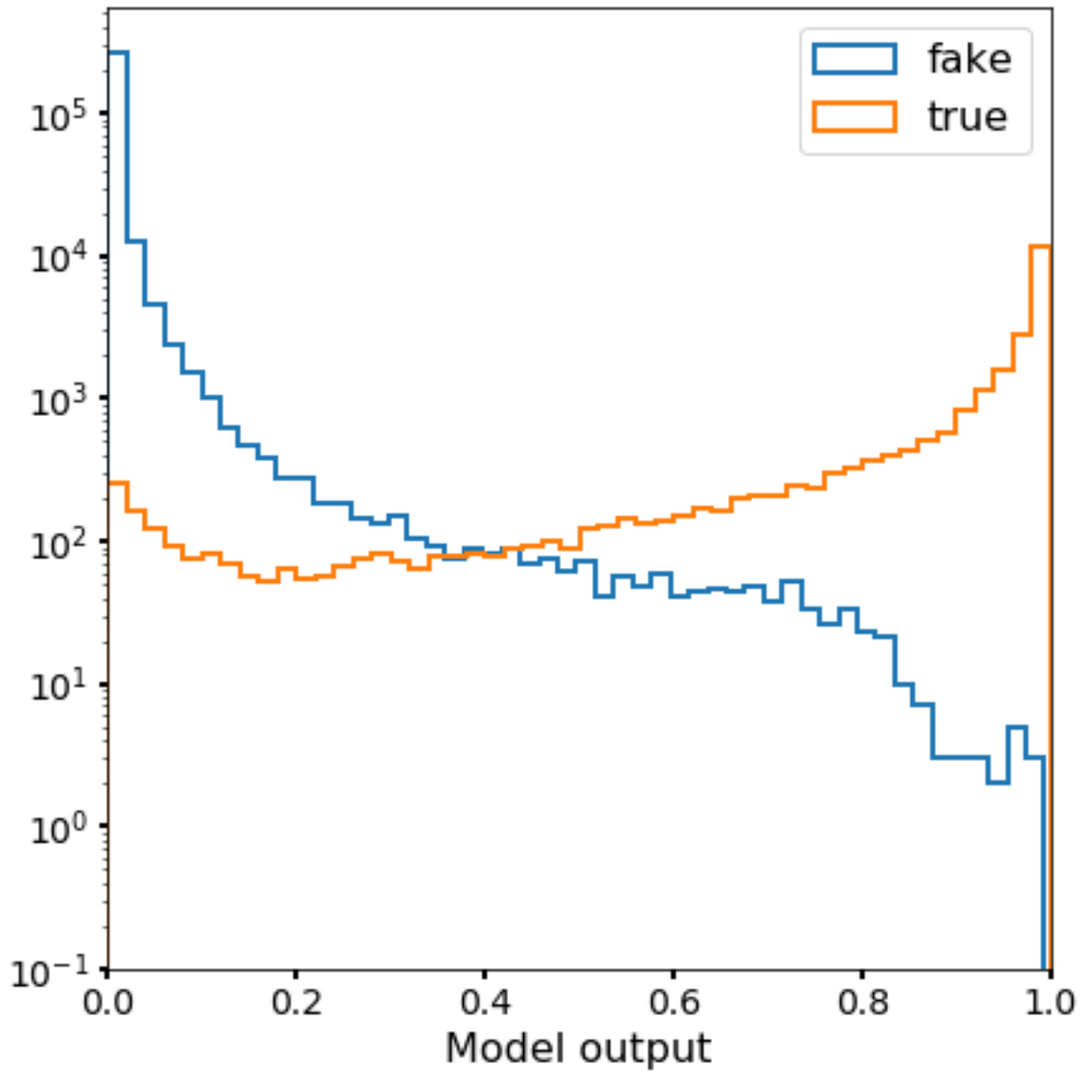


Edges with higher scores are darker than that with lower scores

Edges with scores < 0.01 are removed for visualization purpose.

Predicted score improves after each iteration

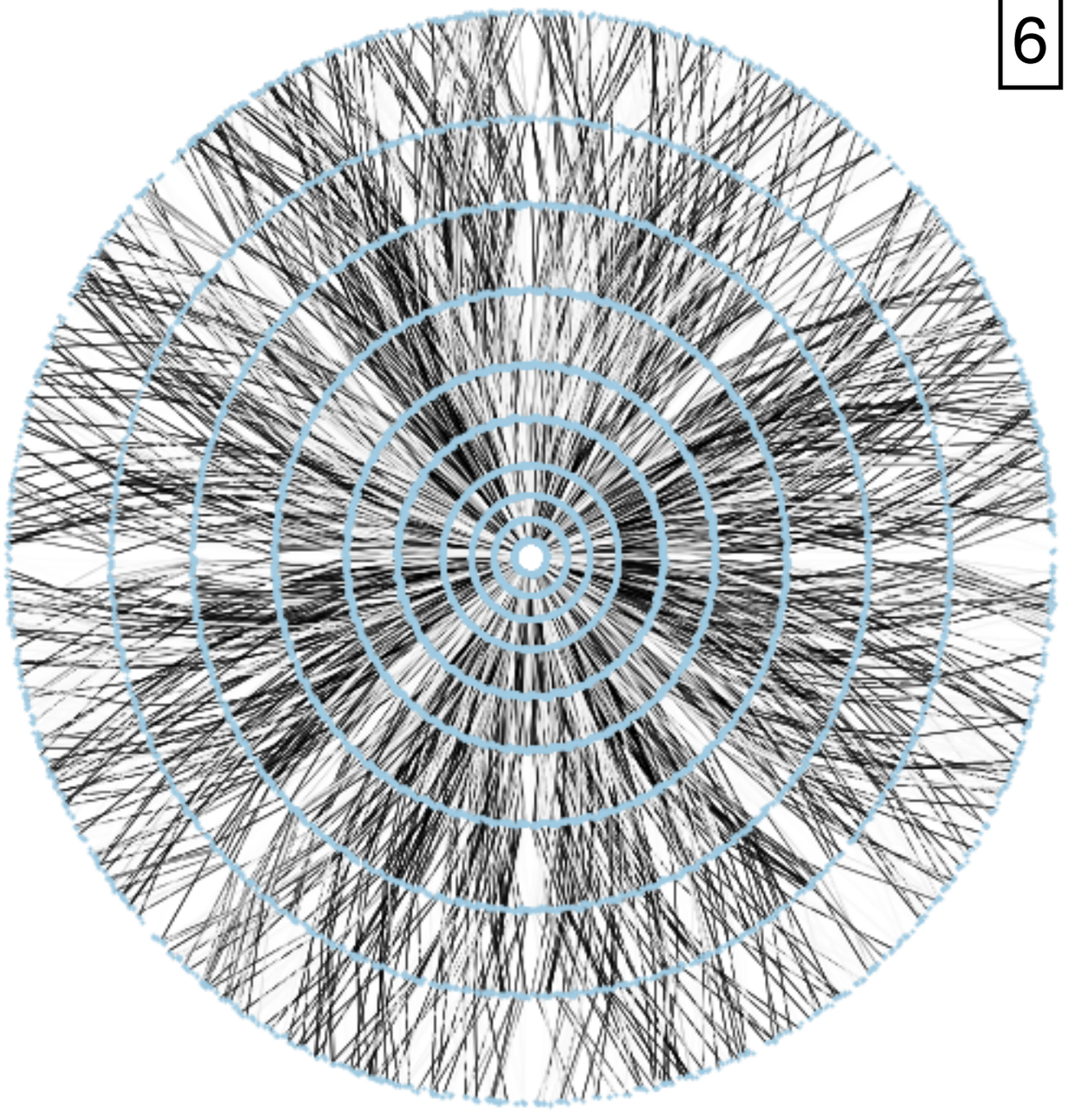


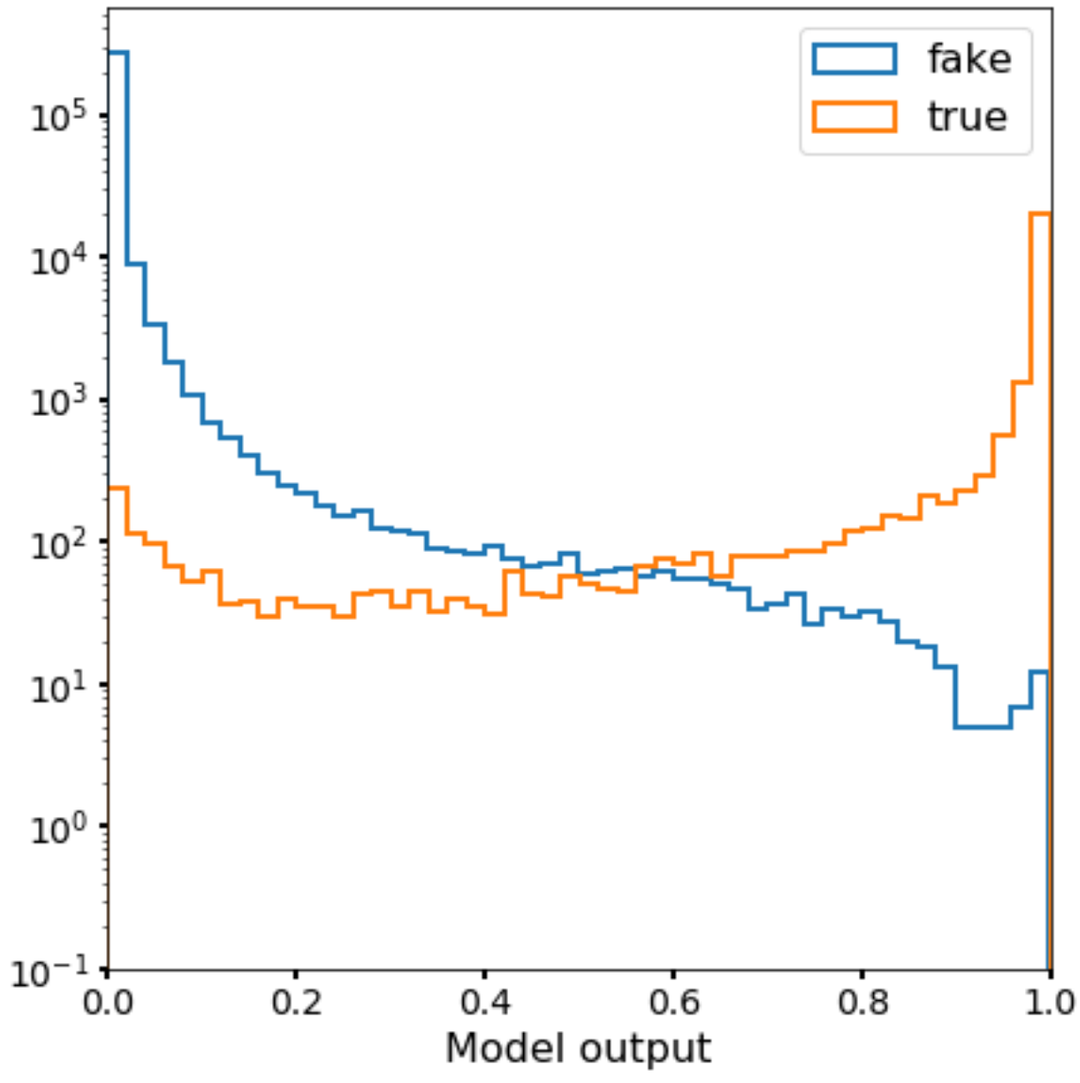


Edges with higher scores are darker than that with lower scores

Edges with scores < 0.01 are removed for visualization purpose.

Predicted score improves after each iteration

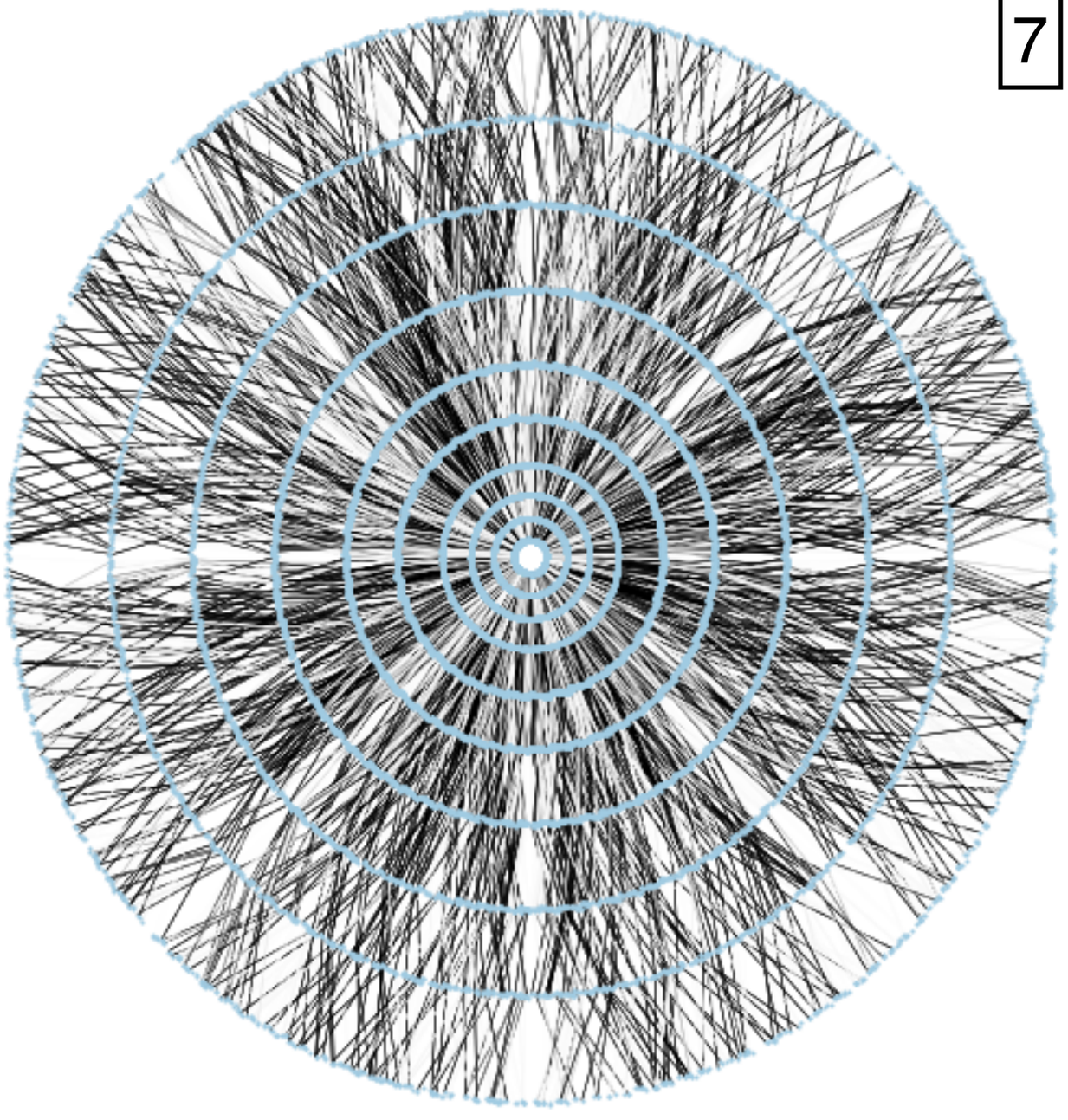


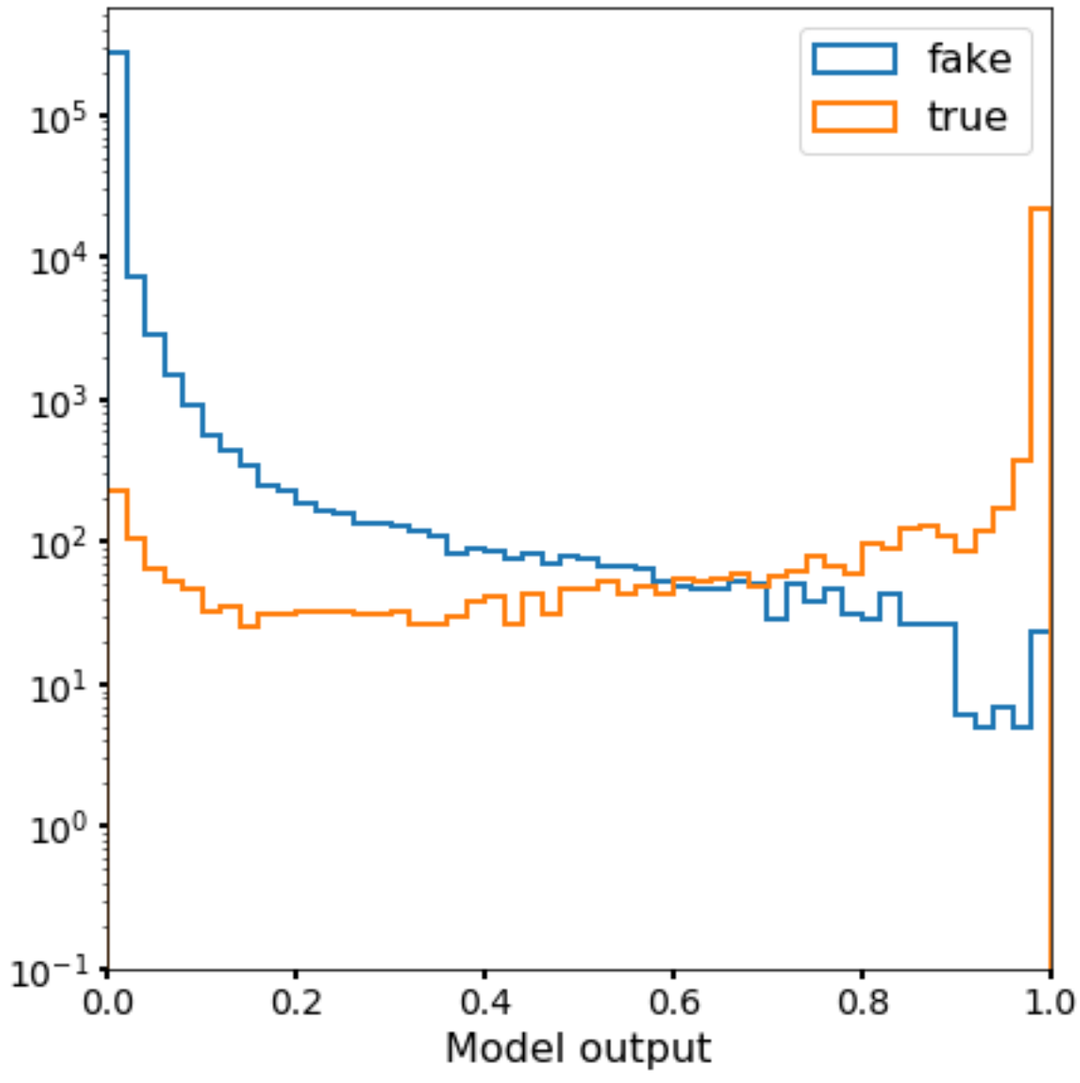


Edges with higher scores are darker than that with lower scores

Edges with scores < 0.01 are removed for visualization purpose.

Predicted score improves after each iteration

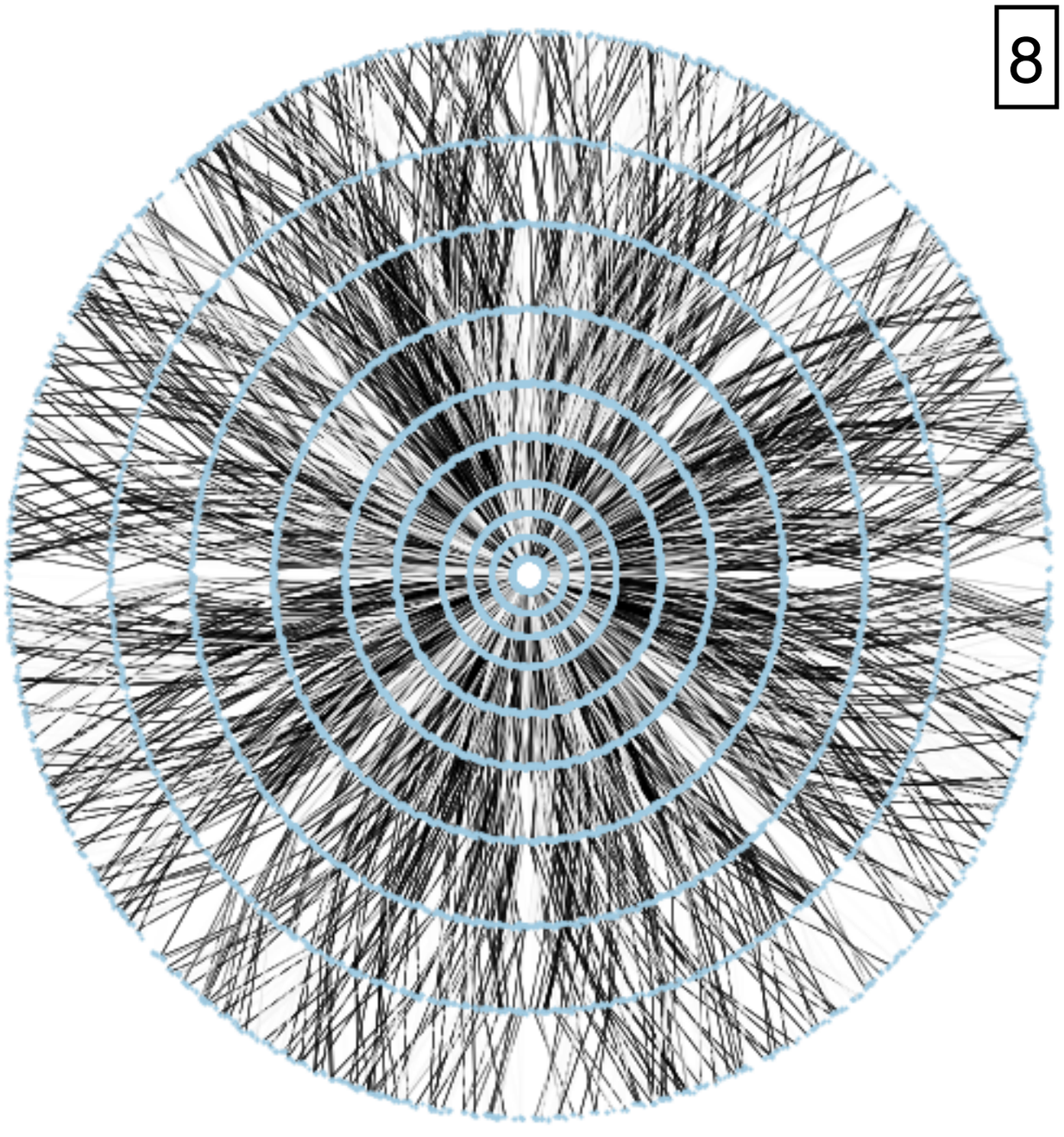


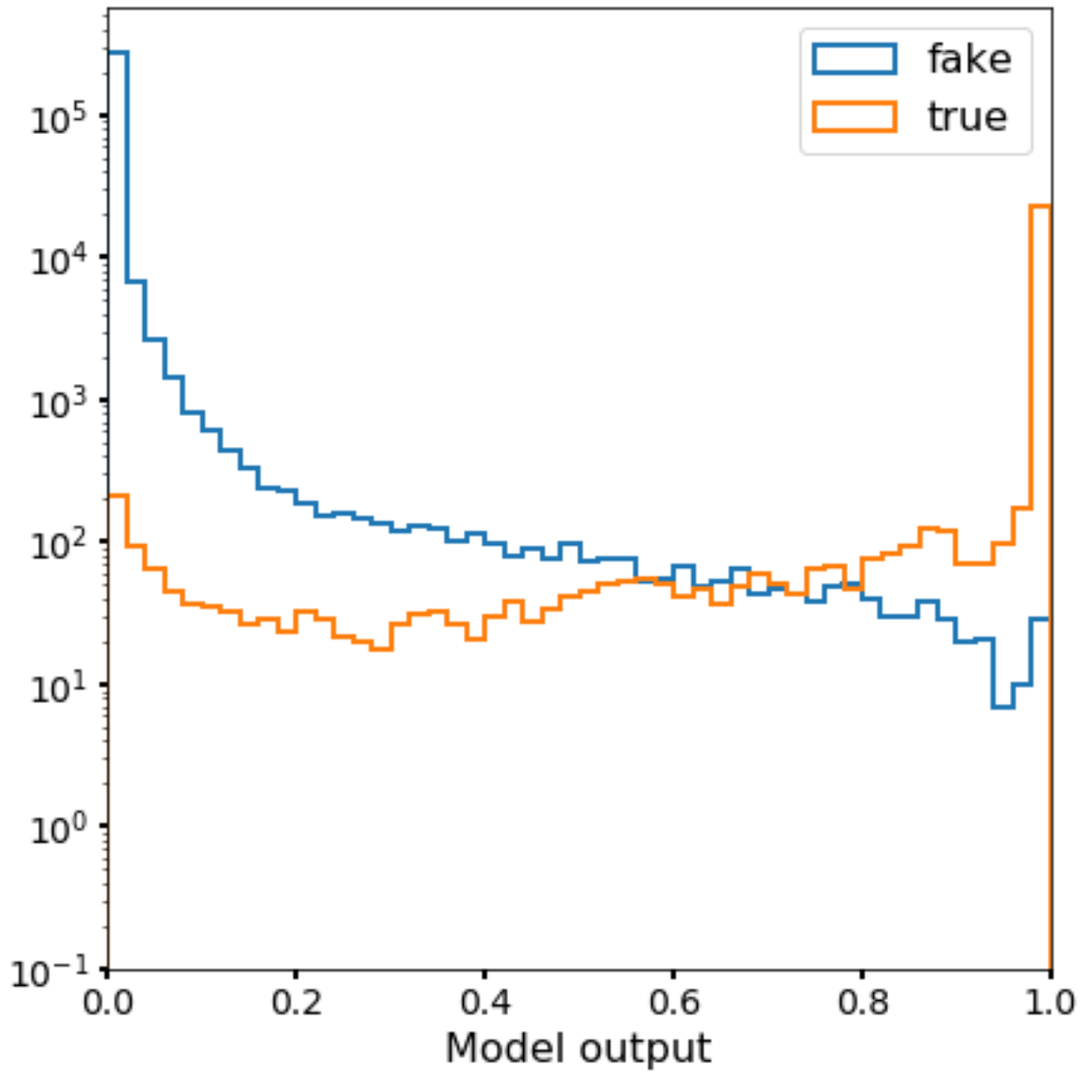


Edges with higher scores are darker than that with lower scores

Edges with scores < 0.01 are removed for visualization purpose.

Predicted score improves after each iteration



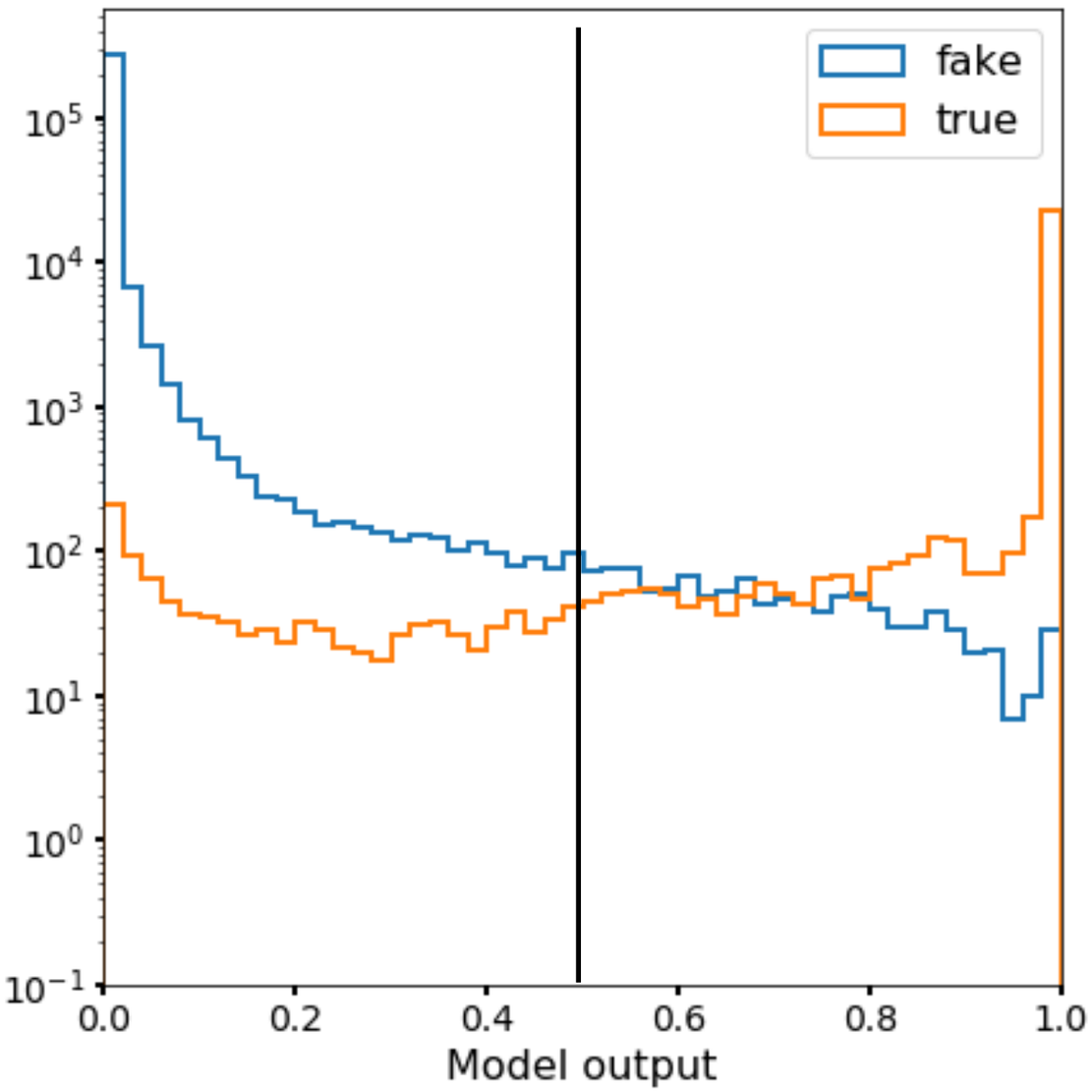


Edges with higher scores are darker than that with lower scores

Edges with scores < 0.01 are removed for visualization purpose.

Performances

- ~266k tunable parameters in TensorFlow
- Trained on a GPU for about 2 epochs
- Weighted loss function

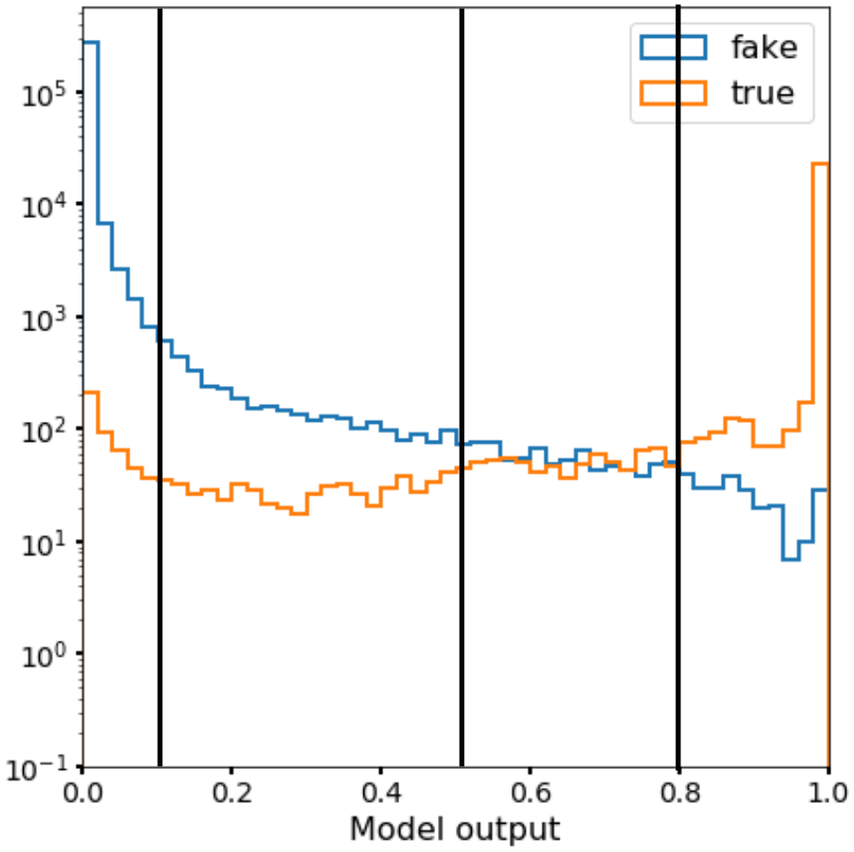


Threshold	0.1	0.5	0.8
Edge Efficiency	98.2%	95.9%	93.0%
Edge Purity	84.0%	95.7%	98.9%

$$\text{Efficiency} = \frac{\#\text{ of True Edges passed the threshold}}{\#\text{ of total True Edges}}$$

$$\text{Purity} = \frac{\#\text{ of True Edges passed the threshold}}{\#\text{ of total Edges passed the threshold}}$$

Connect The Dots, a simple algorithm



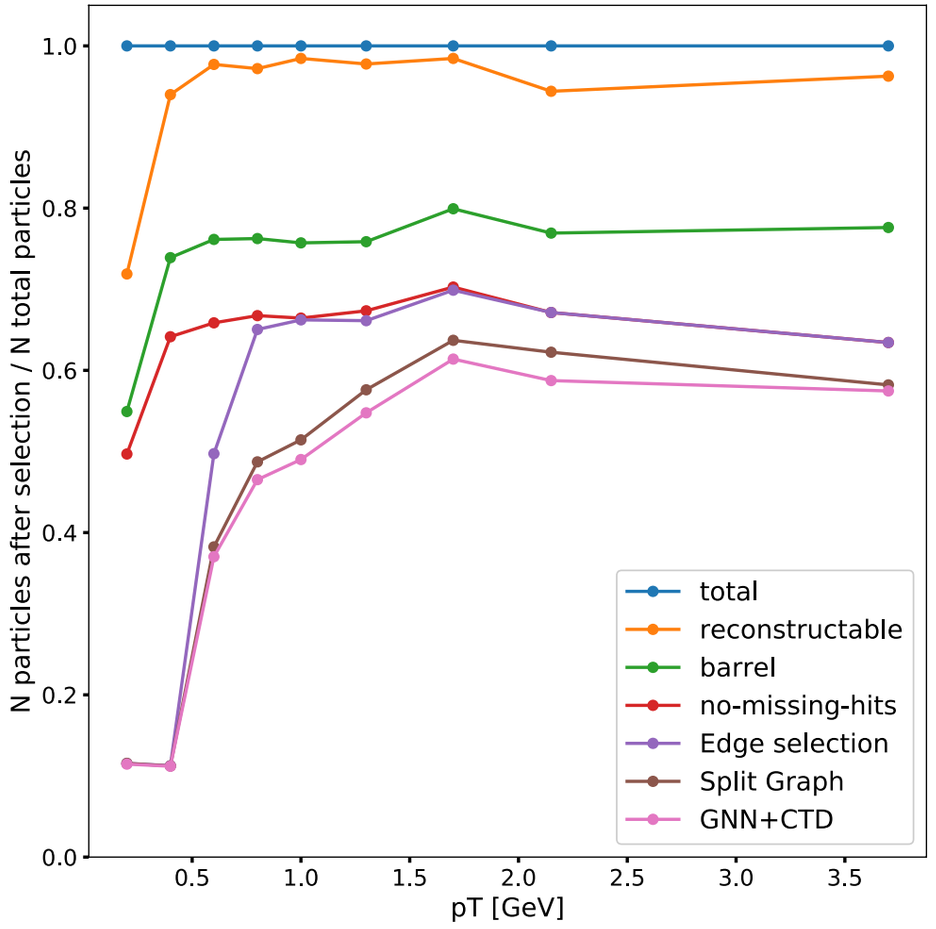
Guided by edge scores from GNN, we walk through the graph from inside to outside along edges with the maximum score that is **> 0.1,** as ones < 0.1 having high probability being fake

Add paths with scores **> 0.8** → having high probability being true

- Longest path is selected for the starting hit, then go to next not-used hit.
- Each hit is assigned to one track.
  - We will lift the constraint to gain efficiency and robustness, and then resolve ambiguities.

A summary

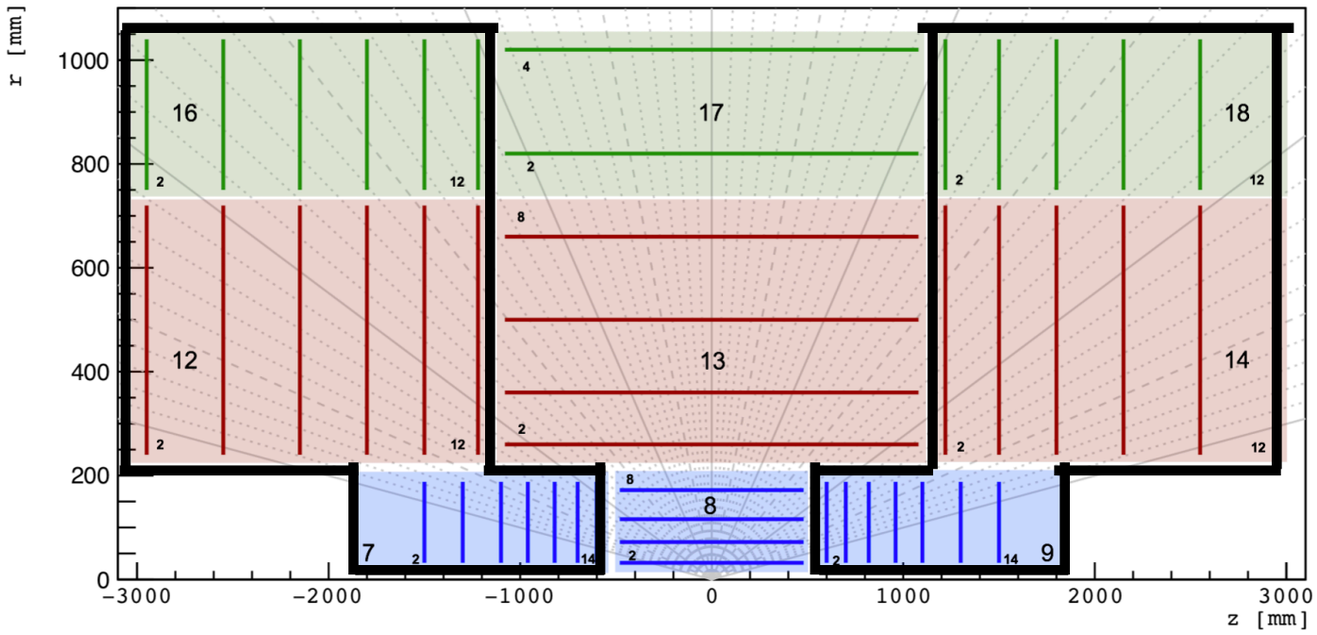
one-event	N-particles	ratio w.r.t Total	ratio w.r.t Reconstructable	relative ratio
Total	11170	100%		100%
Reconstructable	9635	86%	100%	86%
Barrel	7492	67%	78%	78%
No-missing hits	6600	59%	69%	88%
Edge selection	3114	28%	32%	**47%**
Split graph	2668	24%	28%	**86%**
GNN	2590	23%	27%	**97%**



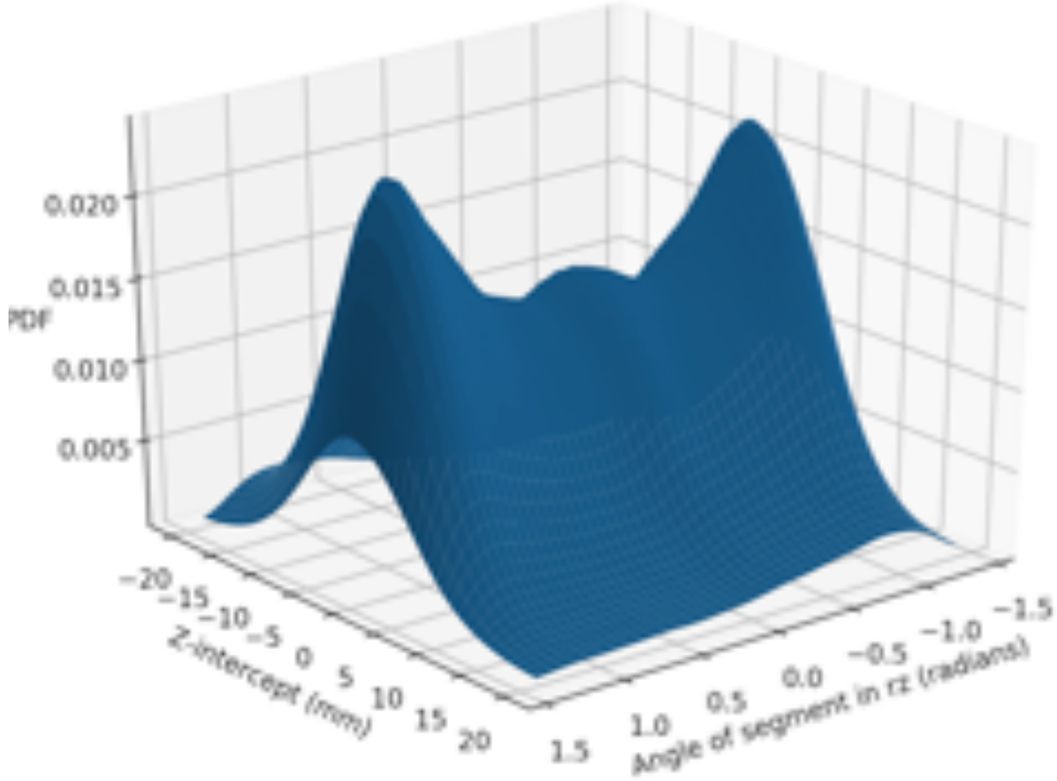
GNN edge classifier achieves over 95% efficiency across the pT range with a purity greater than 95%

Including Endcap and Noise hits

- Challenges to make promising pairs
  - Endcap has more complex geometry
  - Noise hits introduce more fake pairs



- Exploring Gaussian Kernel Density Estimation
  - inputs: angle in r-z, z-intercept, angle in r-ϕ, ϕ-intercept
  - output: probability of two hits being connected by a track
  - purity is about 1%.



Summary and Outlook

- **HEP.TrkX follow-up project (Exa.TrkX) funded for three years**
  - Will focus on using distributed training at HPC scale
  - Goal is to validate models for production in at least one experiment (ATLAS, CMS, DUNE)
- Preliminary results show promising performance of GNN in high density environments
  - It scales well from low-density data to high density one
  - **GNN achieves > 95% efficiency and purity in predicting edges**
  - On-going efforts in scrutinizing efficiency loss at each step
- We are actively working with computing experts to handle computing challenges
- We start to talk with tracking experts from HEP experiments to make the algorithm more practical