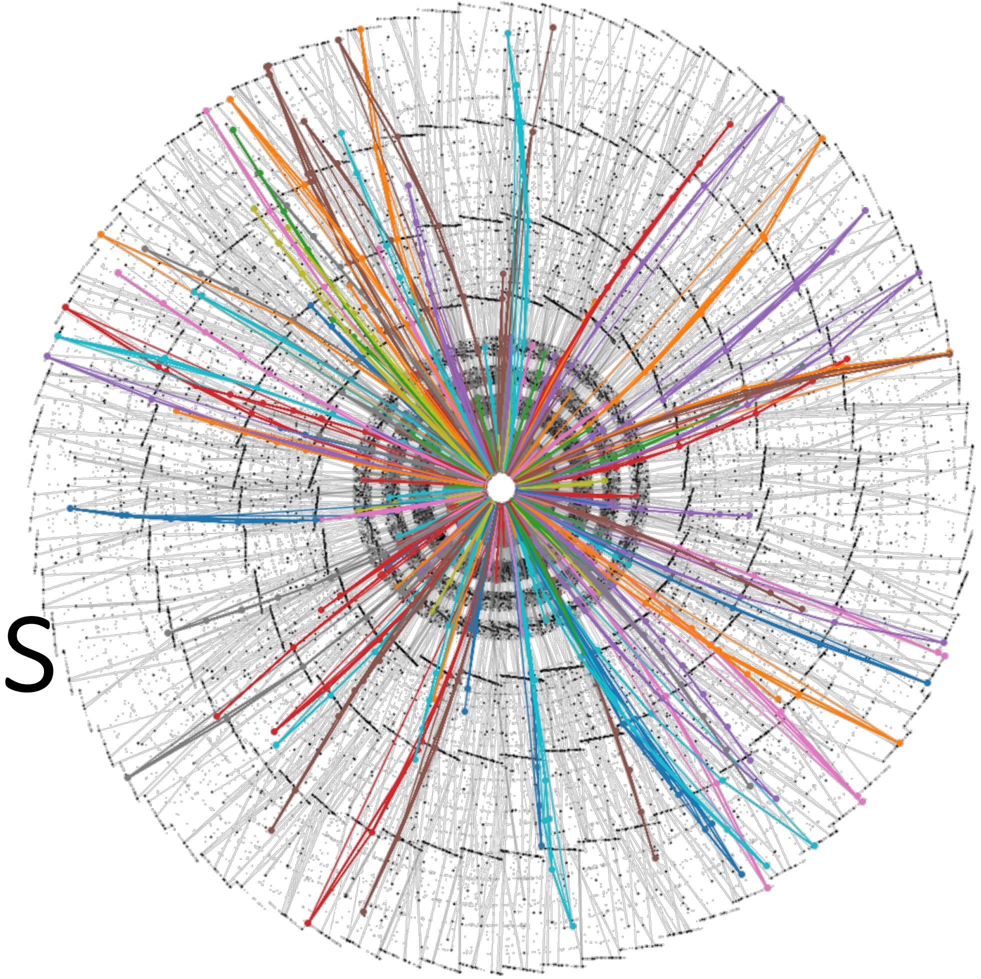
Tracking with Graph Neural Networks

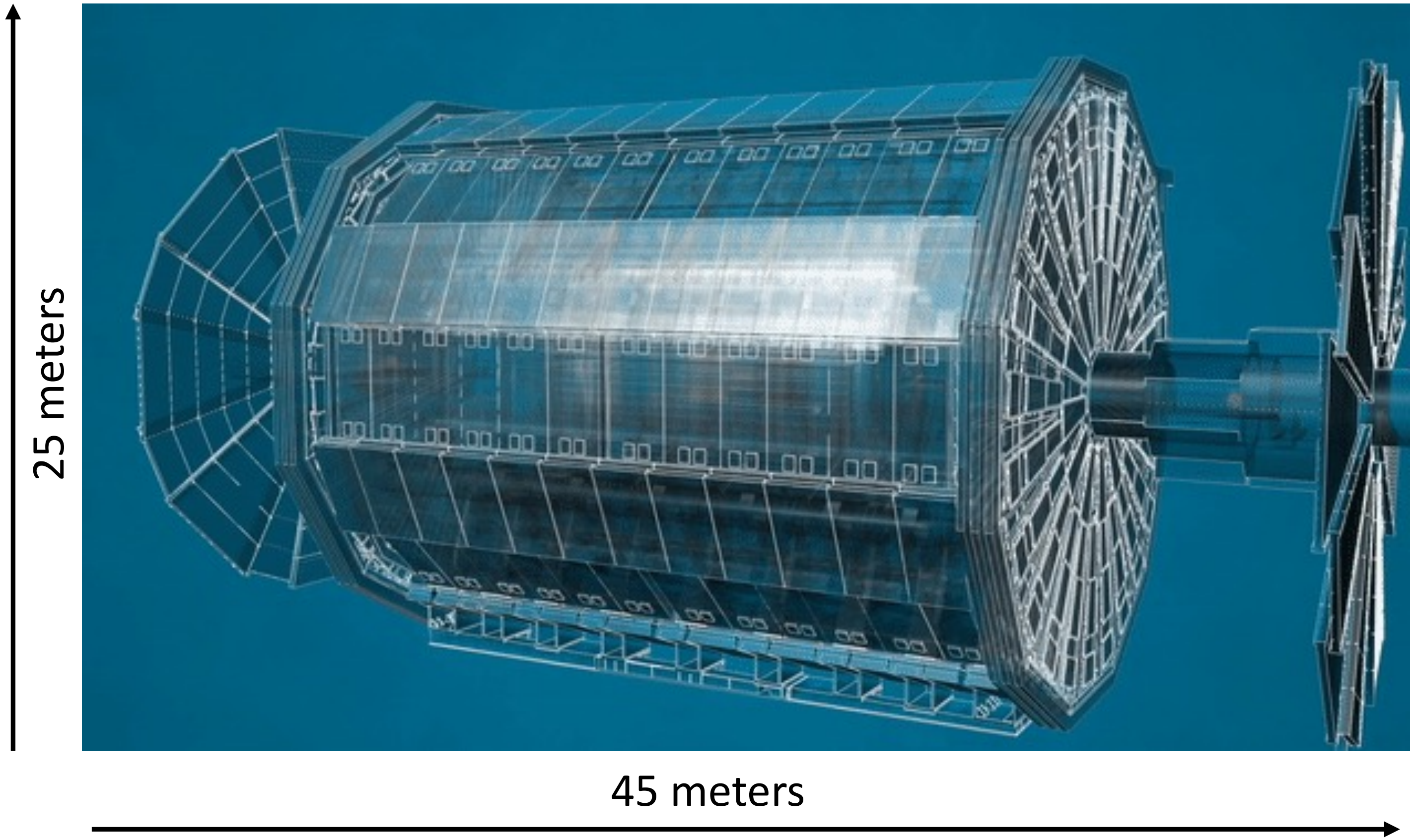
Brookhaven AIMS Series
14^{th} March 2023

Charline Rougier

Laboratoire des 2 Infinis - Toulouse



Event reconstruction

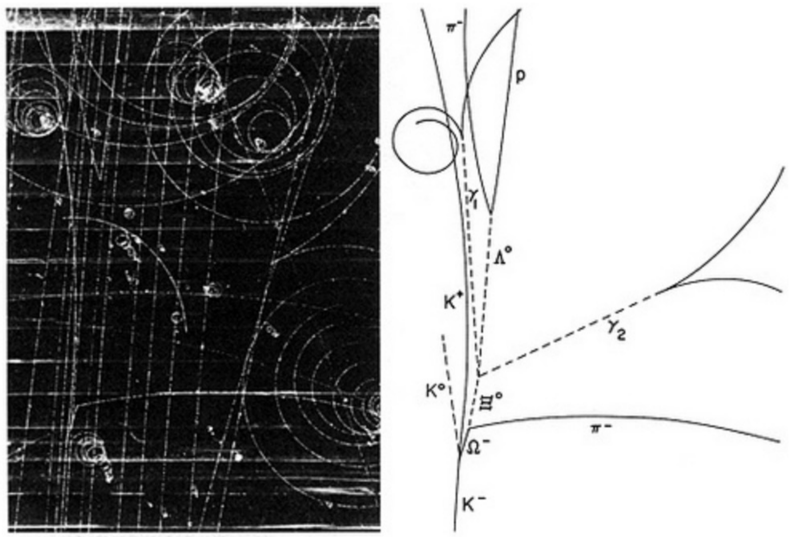


High energy detectors detectors work as massive cameras recording collisions:

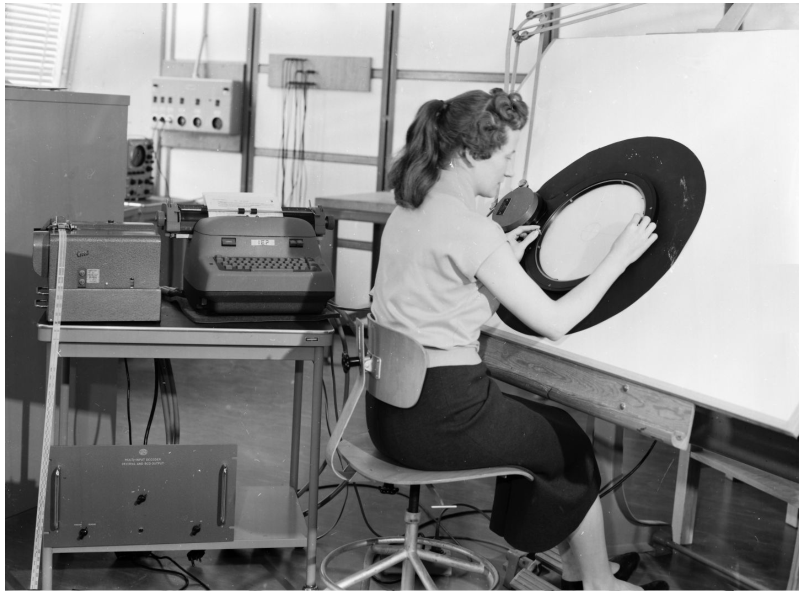
- Event reconstruction = interpretation of the picture to identify the particles produced.

Track reconstruction

In the 60s – 70s, bubble chambers take pictures of collisions.



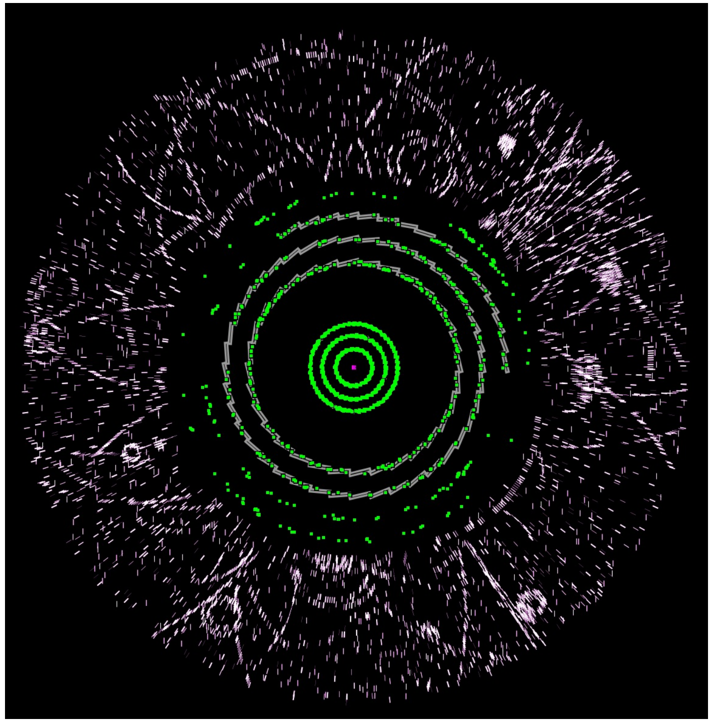
First observation of Ω^-, bubble chamber @ BNL in 1964.



At that time, the track parameter estimations were done by hand, from the photographs of the events seen in the bubble chamber.

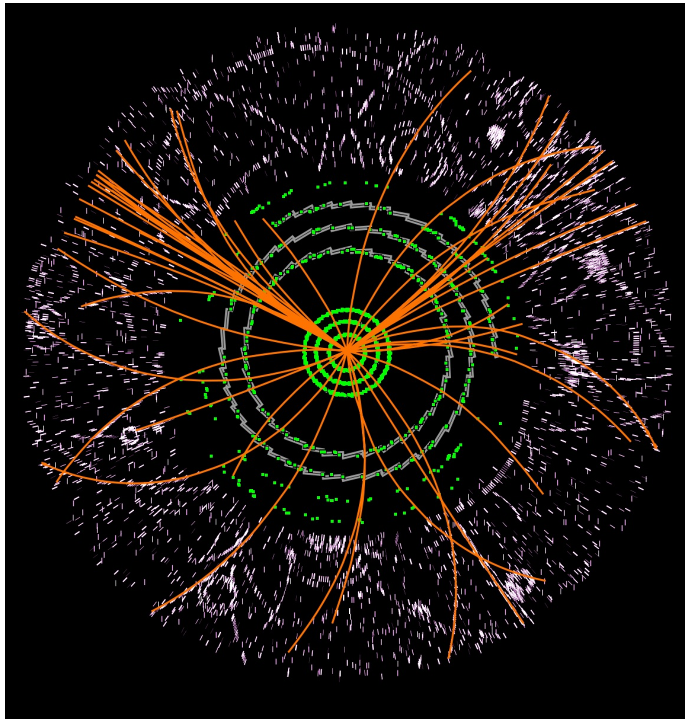
Track reconstruction

Nowadays, we use digital readout and algorithms.



Track reconstruction

Traditional methods:
sequential algorithms

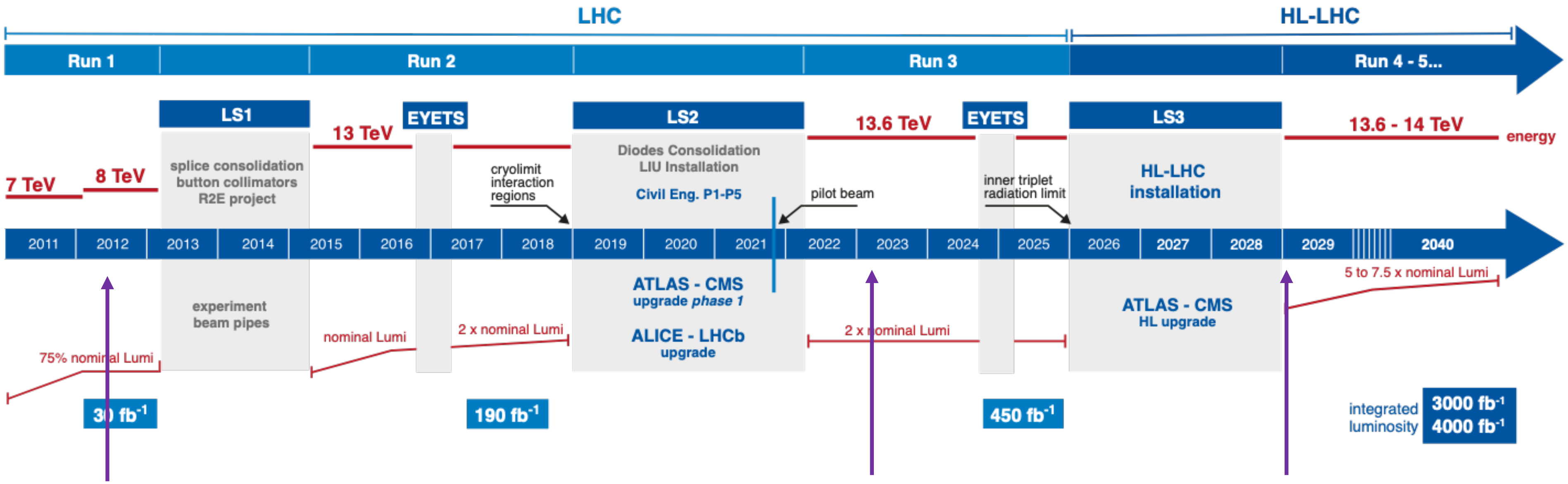


ATLAS current inner detector

ATLAS current inner detector

Picture from: [M. Elsing]

LHC timeline

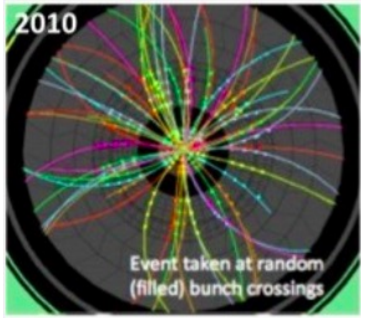


Discovery of the Higgs boson

Today

Start of the High Luminosity era

Start of data taking



12 pile-up

40 pile-up

High Luminosity phase

=

Increase of the pile-up

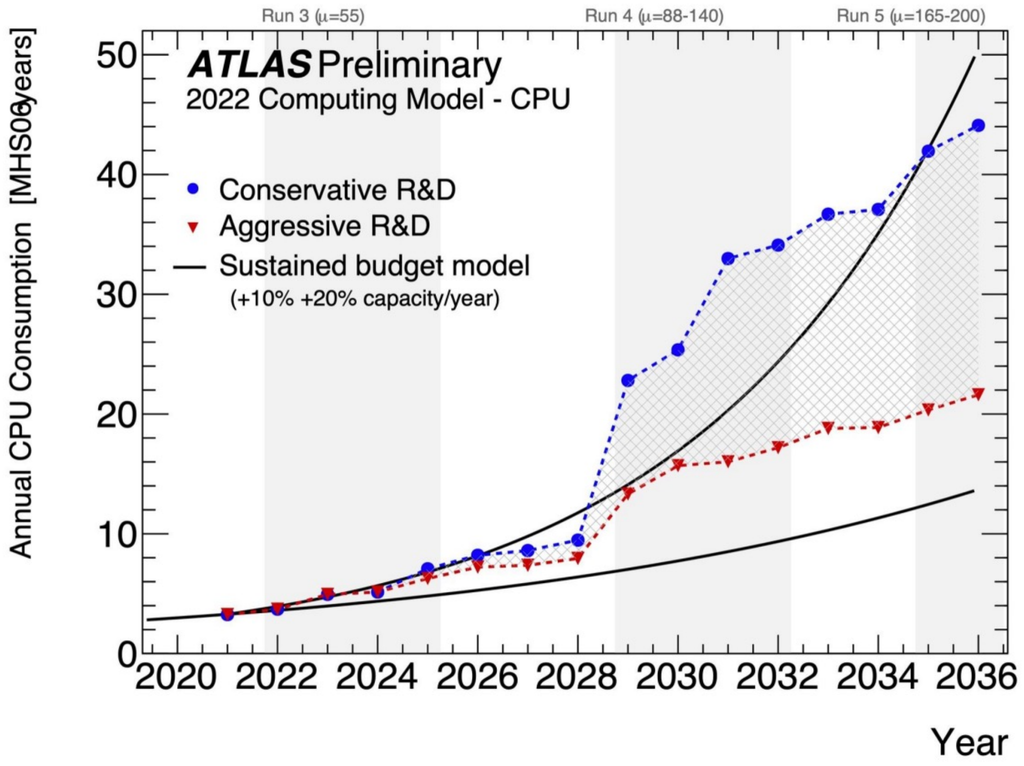
40 → 200

ATLAS EXPERIMENT

LHC High Luminosity upgrades

- **The LHC upgrade: HL-LHC era**
  - ➡ Physics run to start in 2029
  - ➡ Increase in event complexity: $\langle \mu \rangle \approx 200$
  - ➡ Increase in data taking rate
  - ➡ ATLAS detector upgrades: new **I**nner **T**rac**k**ing detector **ITk** included

Brings unprecedented **challenges** for software and computing.

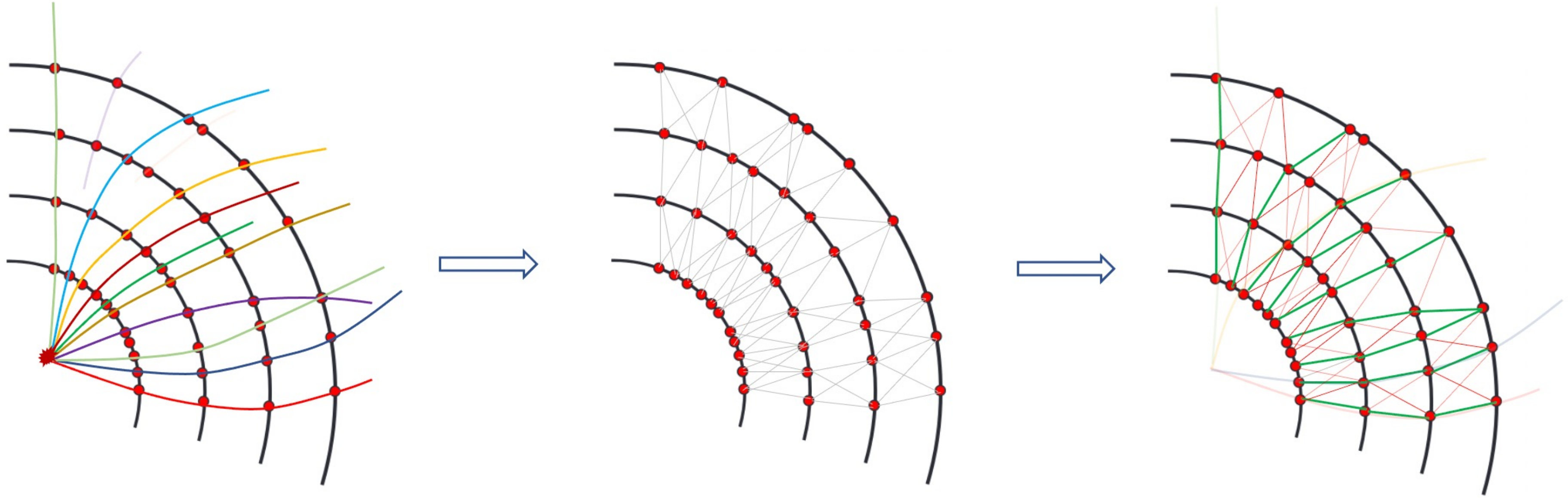


From *ATLAS HL-LHC Computing Conceptual Design Report*

Machine learning applied to tracking

- **Track reconstruction = CPU-intensive stage**
  - ➡ ML techniques running on GPUs ? Raw data from tracking detectors are **sparse** data

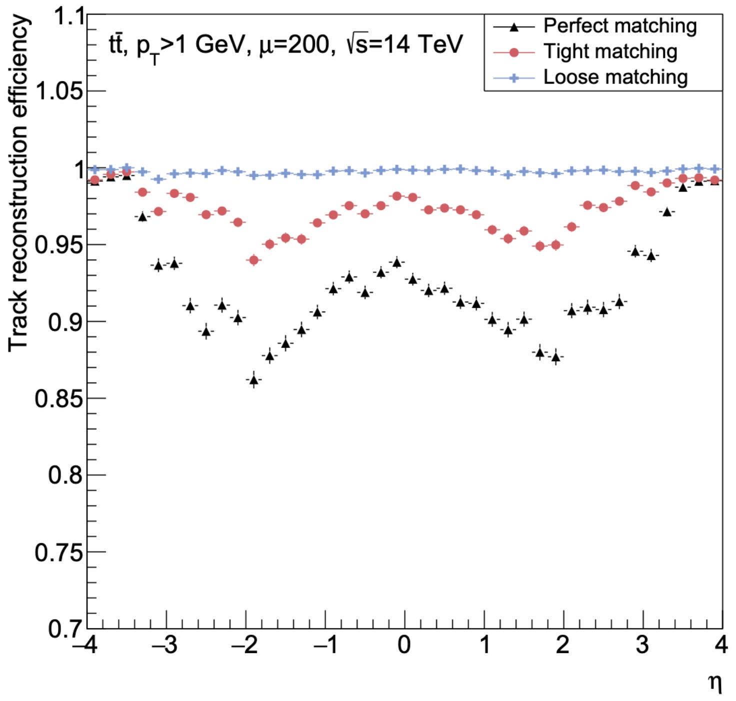
- **Graph Neural Networks (GNNs):** proof of principle by Exa.TrkX project



Machine learning applied to tracking

- **Track reconstruction = CPU-intensive stage**
  - ➡ ML techniques ? Raw data from collisions are **sparse** data

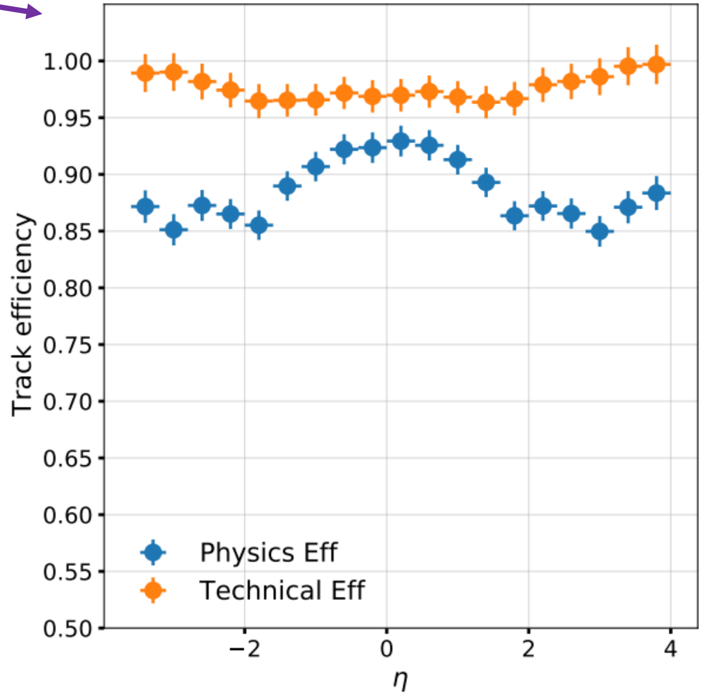
- **Graph Neural Networks (GNNs):** proof of principle by Exa.TrkX project
  - ➡ Method applied to TrackML data by L2IT and Exa.TrkX projects



These were proofs of principle that GNN tracking is a promising solution.

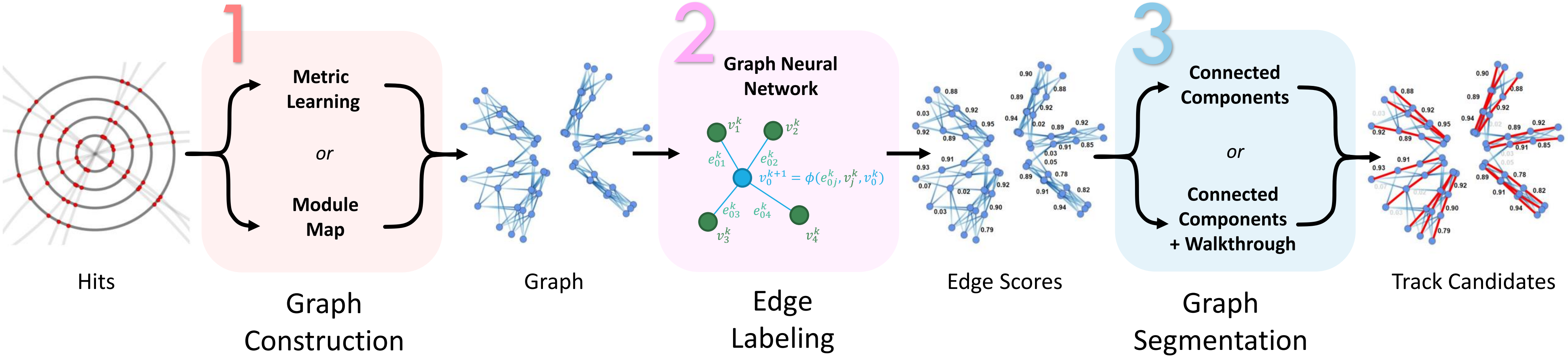
TrackML sample does not have:

- **Secondary particles (δ rays, nuclear interactions, …)**
- **Realistic simulation of strip detectors**



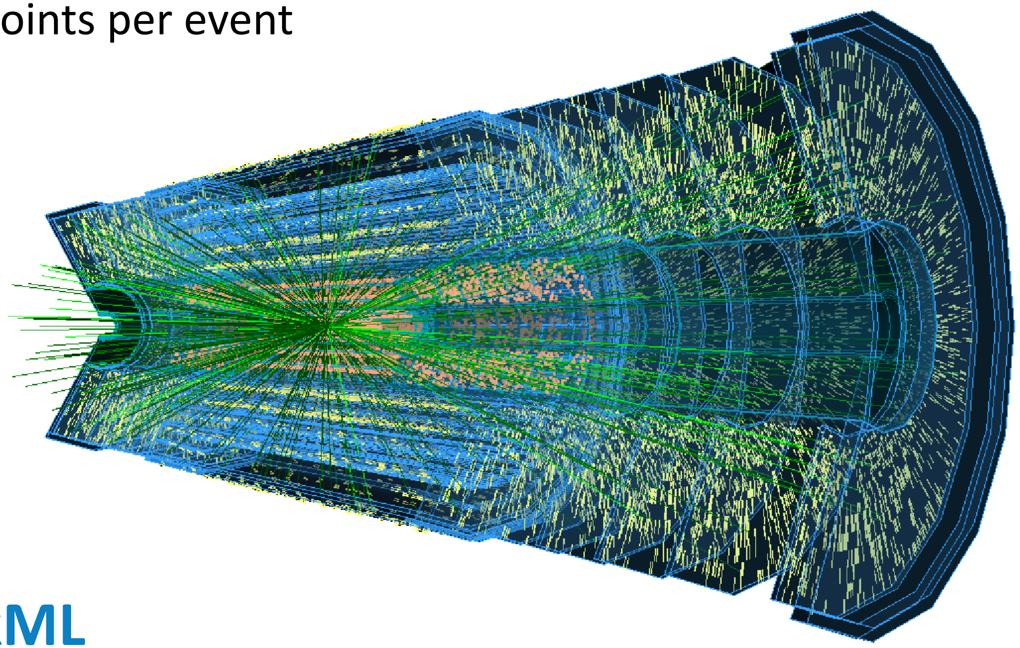
Machine learning applied to tracking

- **Time to apply it to ATLAS simulated samples**



Simulated sample

- **ATLAS simulated sample: $t\bar{t}$ with $\langle \mu \rangle = 200$ at $\sqrt{s} = 14\ TeV$**
  - About 100k events available
  - About 10k particles per event
  - About 300k space-points per event

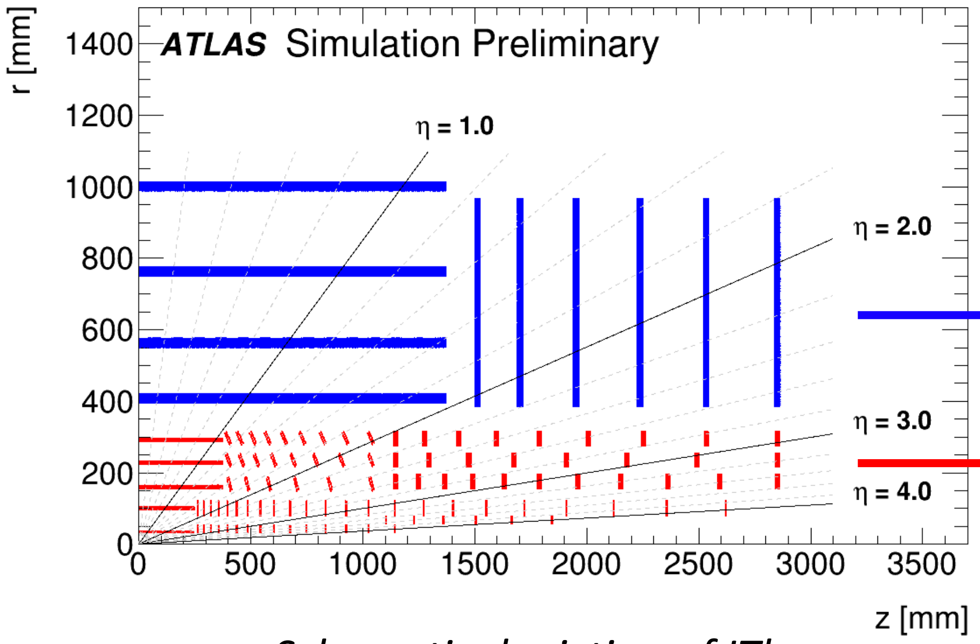


- **Compared to TrackML**
  - Number of space-points multiplied by ~ 3
  - No more parametrized simulation of trajectory (Geant 4)
  - Size of the luminous region : ~ 2 cm => ~ 20 cm

Simulated sample

- **Define target particles**
  - $p_T > 1\ \mathrm{GeV}$
  - No secondaries
  - No electrons
  - At least 3 space-points in the detector

Dominated by soft interactions



Strip subdetector: 1 space-point = 2 clusters

Pixel subdetector: 1 space-point = 1 cluster

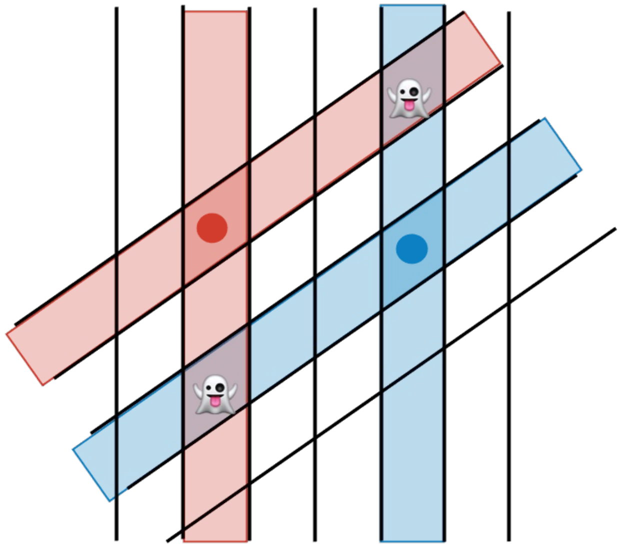
Schematic depiction of ITk

Simulated sample

- **Define target particles**
  - $p_T > 1$ GeV
  - No secondaries
  - No electrons
  - At least 3 space-points in the detector

Dominated by soft interactions

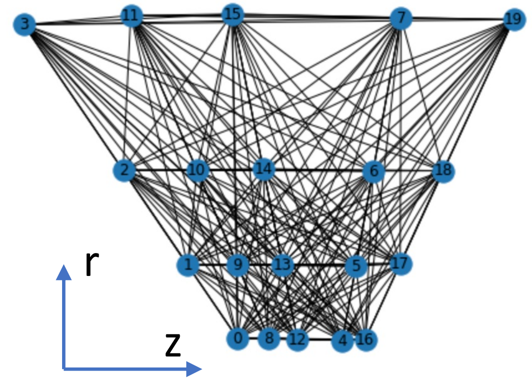
Strip subdetector: 1 space-point = 2 clusters



Space-points from two different particles

👻 Ghost space-point: accidental combination of strip clusters

Graph representation of tracking data



Example with 19 hits in the (z,r) plane

Node = 1 space-point

Edge = connection between two nodes.

- Existence of edge = the 2 nodes could potentially represent 2 **successive** space-points on the same track.

Graph construction is one of the most important parts of a GNN tracking pipeline:

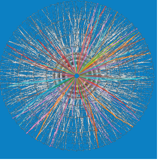
- High efficiency is mandatory: so far lost edges means incorrect track reconstruction
- Graph topology has a huge influence on the performance of a GNN model

$\mathrm{O}(300\mathrm{k})$ space-points in an event => fully connected graph $O(10^{10})$ edges

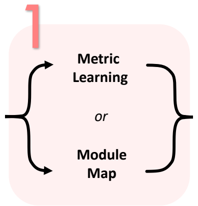
- Comprises unphysical connections

Key question of graph construction:

- **How de we choose the connections between nodes ?**



Graph creation: learning the connections



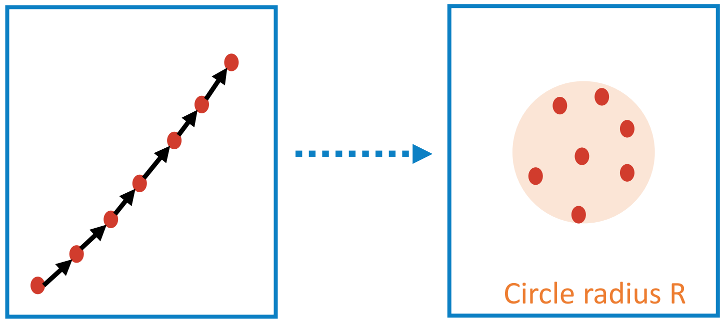
Graph Construction

Metric Learning

All space-points belonging to the same target particles are **learned** by a Multi-Layer Perceptron (MLP) to be embedded into a **space** where they are close.

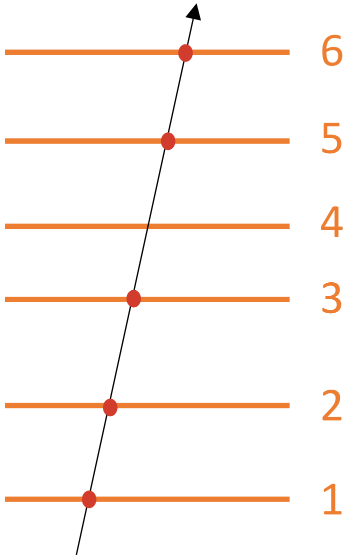
Euclidian space

N-dimensional space learned by the MLP



Module Map

The path of a target particle is followed inside ITk to record all possible **connections** between triplet of silicon **modules**.

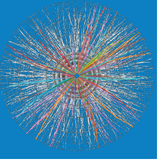


Connections record :

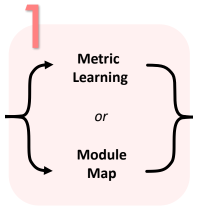
1 → 2 → 3
2 → 3 → 5
3 → 5 → 6

The Module Map is built using 90 000 events. It comprises **1 242 665** connections.

ATLAS EXPERIMENT



Graph creation: learning the connections



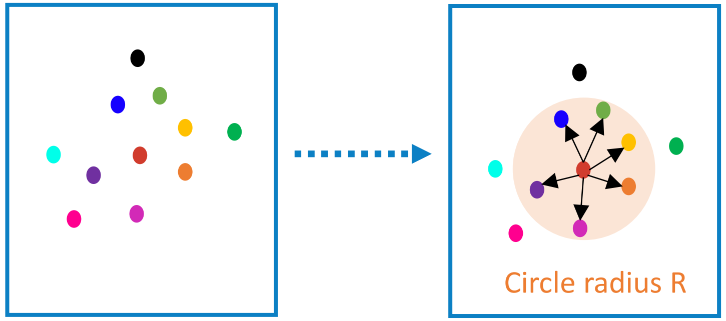
Graph Construction

Metric Learning

Given a source node, edges between this node and all nodes within a radius R from the source are created.

N-dimensional space learned by the MLP

Edges created



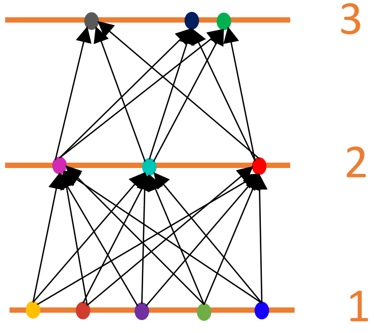
No particular meaning of direction.

O(3 million) edges

Module Map

Edges are created following the connections of the Module Map.

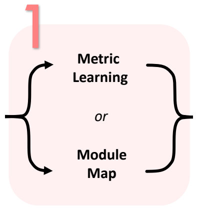
1 → 2 → 3
2 → 3 → 5
3 → 5 → 6



Direction “***inside-out***” are given to edges.

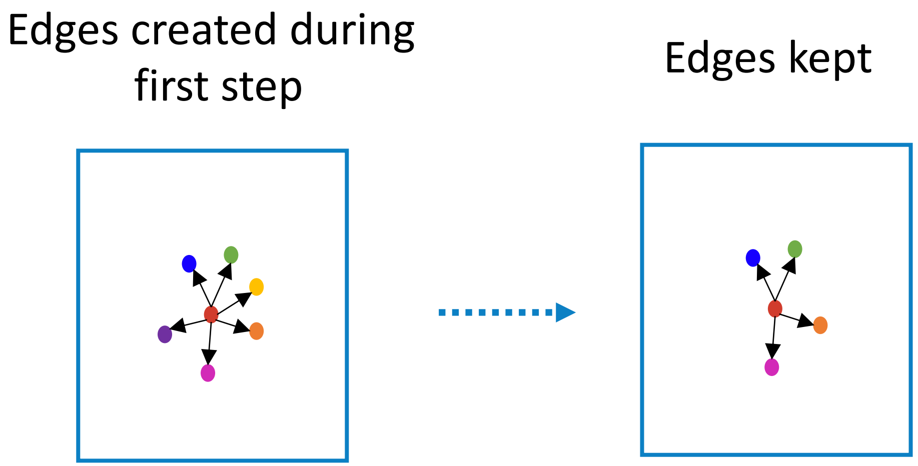
O(billions) edges

Graph creation: pruning the connections



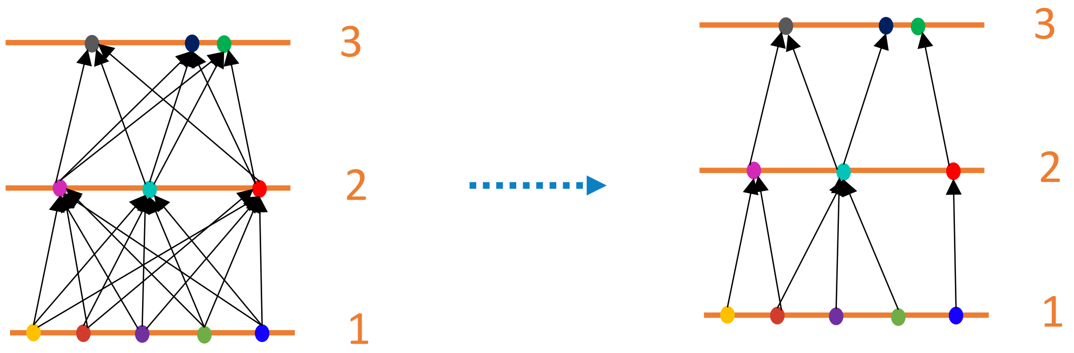
Metric Learning

Additional filtering is done using another MLP.



Module Map

Additional filtering is done with geometric cuts.

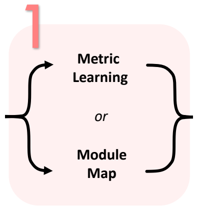


- **At pair level:**
  - $z_0 = z_{h1} - r_{h1} \times \left(\frac{\Delta z}{\Delta r}\right)$
  - $\phi_{slope} = \frac{\Delta\phi}{\Delta r}$
  - $\Delta\phi = \phi_{h2} - \phi_{h1}$
  - $\Delta\eta = \eta_{h2} - \eta_{h1}$
- **At triplet level:**
  - $\Delta\frac{\Delta y}{\Delta x} = \frac{\Delta y_{12}}{\Delta x_{12}} - \frac{\Delta y_{23}}{\Delta x_{23}}$
  - $\Delta\frac{\Delta z}{\Delta r} = \frac{\Delta z_{12}}{\Delta r_{12}} - \frac{\Delta z_{23}}{\Delta r_{23}}$

($h_{1,2}$ being the hits connected by the edges)

Geometrical cuts automatically adjusted for each module triplet.

Graph creation



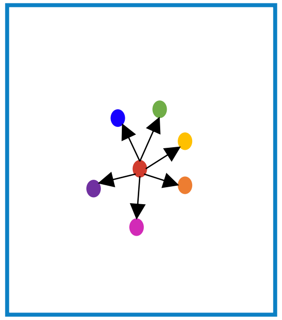
Graph Construction

Metric Learning

Additional filtering is done using another MLP.

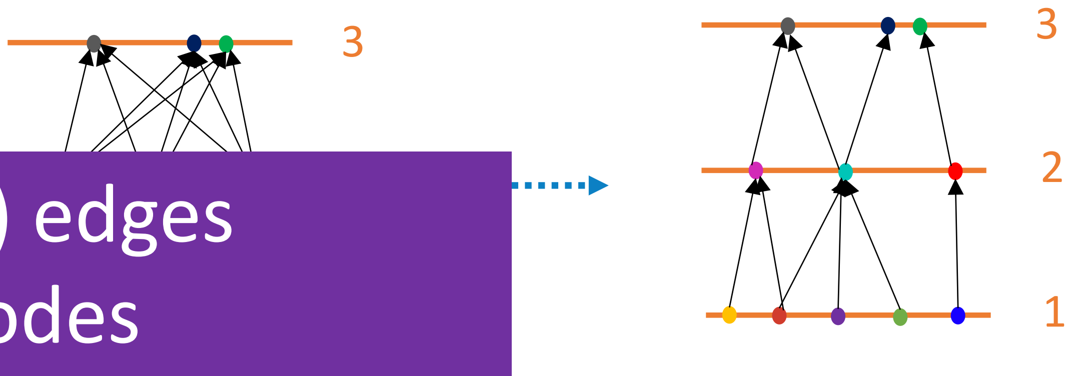
Edges created during first step

Edges kept



Module Map

Additional filtering is done with geometric cuts.



O(1.3 million) edges
O(300k) nodes
~15k true edges

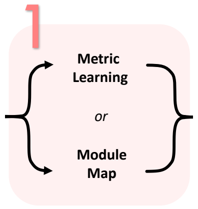
- $z_0 = z_{h1} - r_{h1} \times \left(\frac{\Delta z}{\Delta r}\right)$
- $\phi_{slope} = \frac{\Delta\phi}{\Delta r}$
- $\Delta\phi = \phi_{h2} - \phi_{h1}$
- $\Delta\eta = \eta_{h2} - \eta_{h1}$

- At triplet level:
  - $\Delta\frac{\Delta y}{\Delta x} = \frac{\Delta y_{12}}{\Delta x_{12}} - \frac{\Delta y_{23}}{\Delta x_{23}}$
  - $\Delta\frac{\Delta z}{\Delta r} = \frac{\Delta z_{12}}{\Delta r_{12}} - \frac{\Delta z_{23}}{\Delta r_{23}}$

($h_{1,2}$ being the hits connected by the edges)

Geometrical cuts automatically adjusted for each module triplet.

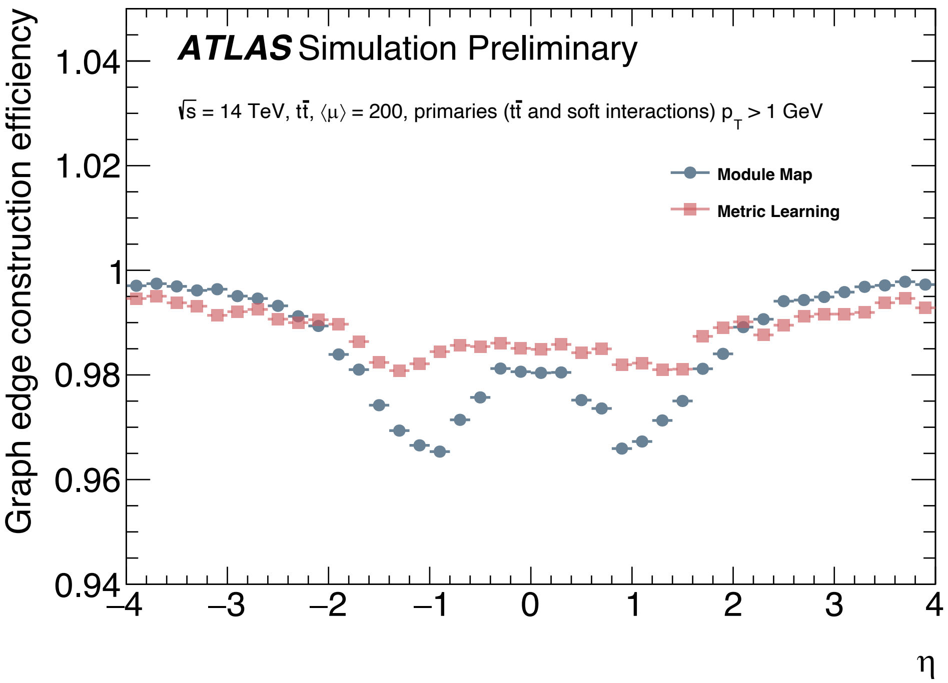
Graph edge construction efficiency

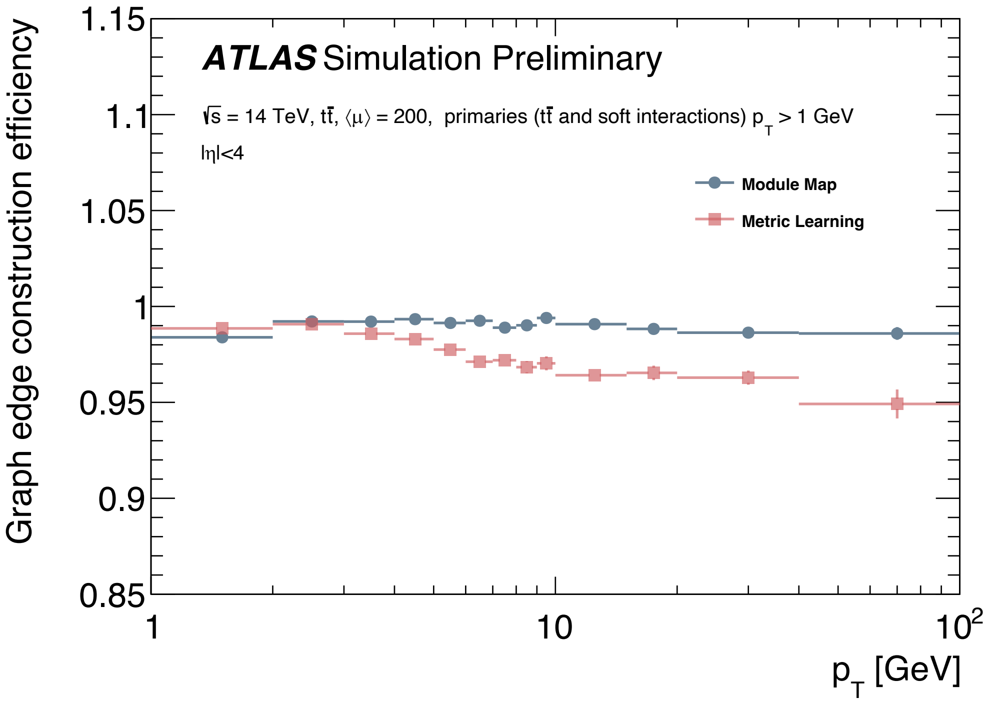


Graph Construction

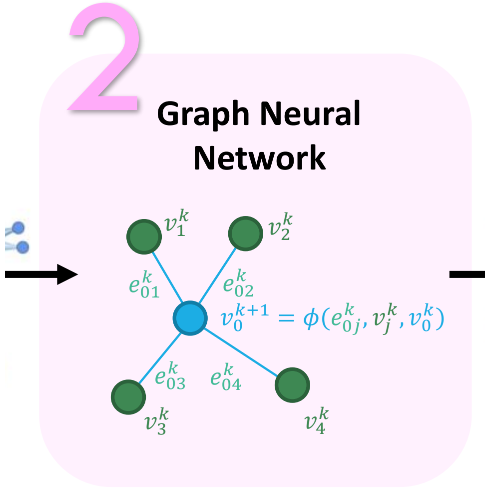
- **Graph edge construction efficiency**

High efficiency is a necessity: an edge lost during the graph construction can't be recovered later.



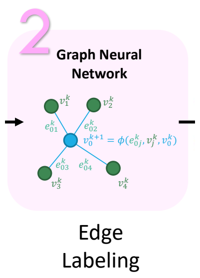


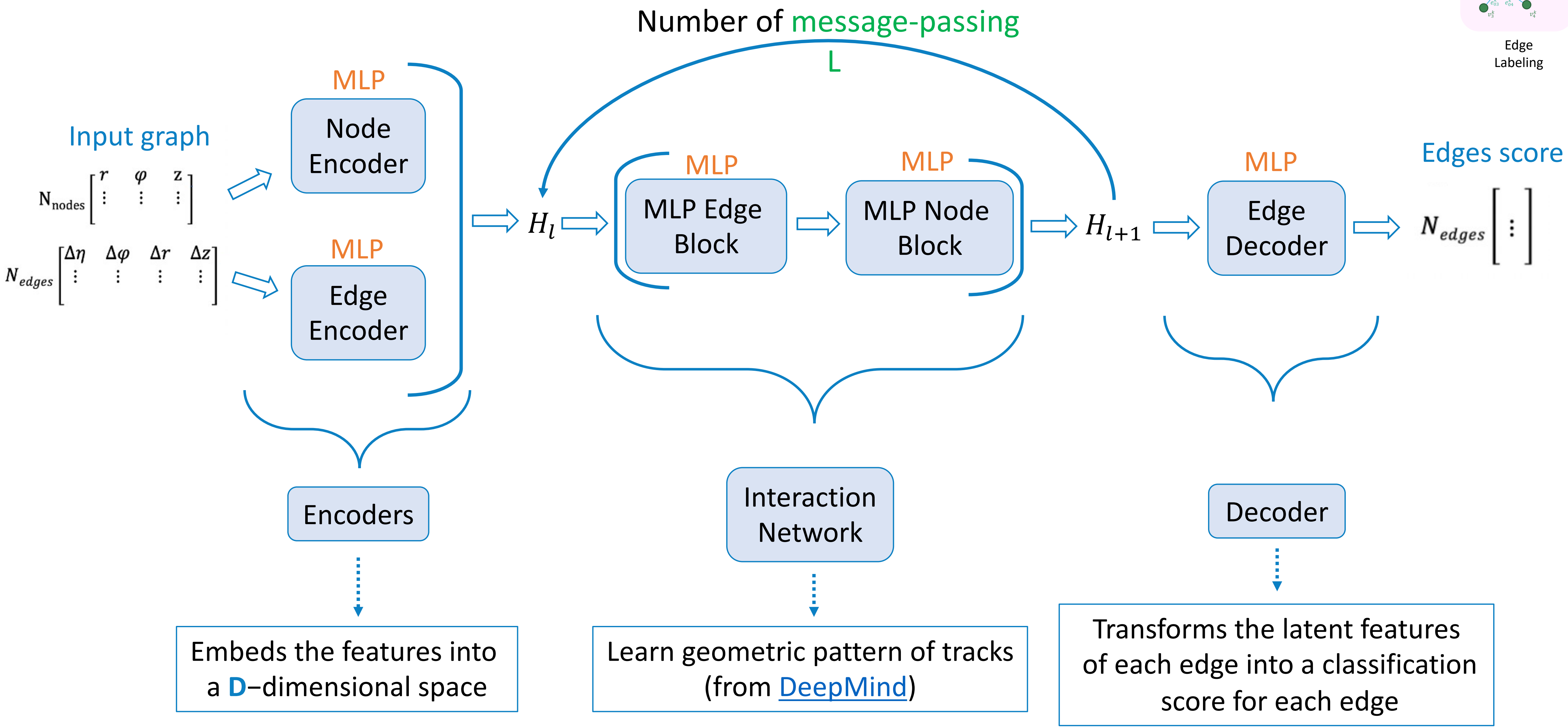
In the following the module map is the method used to build graphs.
The graphs built have 100% efficiency (events have been used during the module map creation).

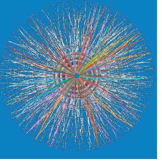


Edge
Labeling

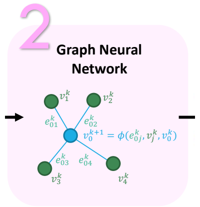
Graph Neural Network model







Memory consumption during training



Edge
Labeling

Memory consumption in ML framework

- Large number of parameters: use of automatic differentiation in reverse mode
- Advantage: fast
- Cost: large storage $\propto$ number of operations

With this GNN model

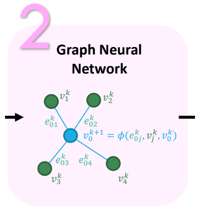
- Our GNN acts on **large** graphs with features embedded in a large-dimensional space
- Use of generic function available in ML framework

Exceed the memory of a GPU

- Around **300 GB** of memory needed to train the model
- Largest GPU available on market have 80 GB of memory

How to train the GNN ?

Memory consumption during training



Edge Labeling

Memory management

- Reduce precision from double to single (FP32)
- TensorFlow Large Model Support (TFLMS) allows to temporarily swap tensors to the GPU host memory when they are not needed

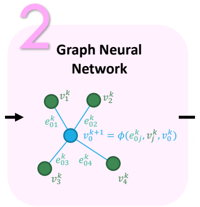
➡ GPU memory no longer a limitation

Training timing dominated by reading / writing -> trade-off between architecture complexity and training speed.

In the future

- TFMLS library no longer maintained ☹
- Checkpointing method: don’t keep the gradient in memory but recompute them when needed.
- On a larger time scale: use of dedicated kernel operation.

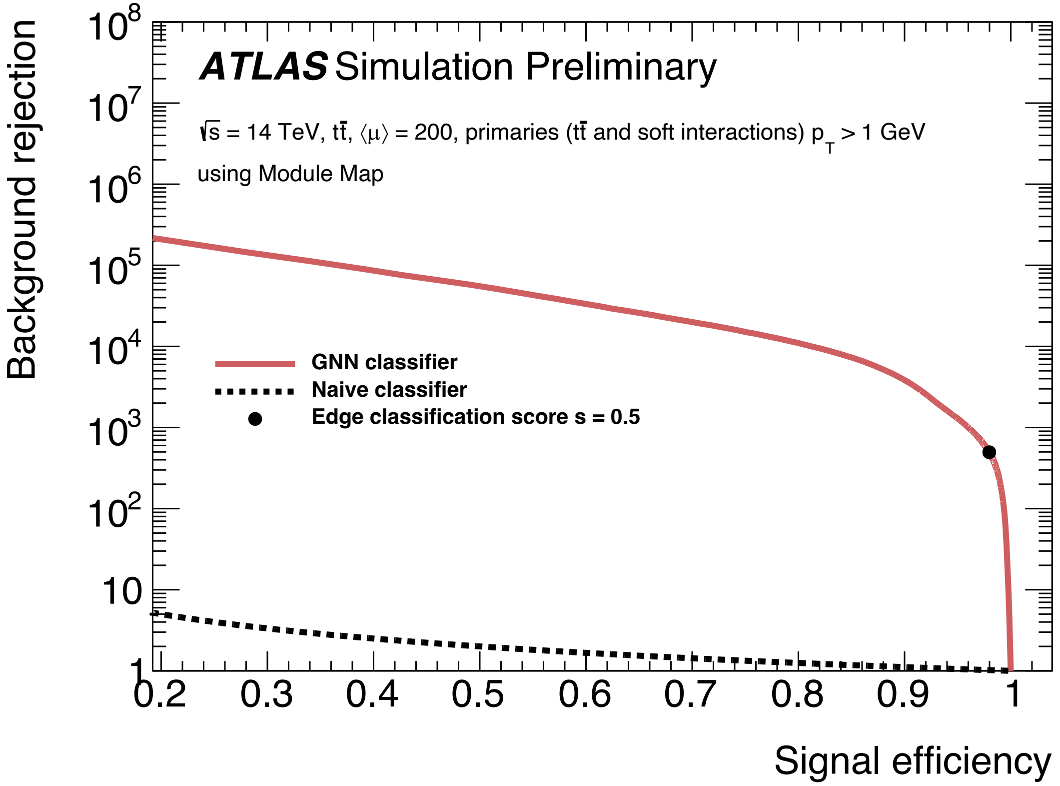
Training the GNN



Edge Labeling

- **Configuration of the GNN architecture**
  - 2 layers in each MLP
  - 128–dimensional space parameters
  - 8 message-passing iterations

- **Training the GNN**
  - 400 graphs for training, 20 for validation
  - Amsgrad optimizer (variant of Adam)
  - Weighted binary Cross Entropy loss:
    - 1.0 for true edges
    - 0.1 for fake edges
    - 0.0 for edges coming from non-target particles



Cut at s = 0.5 on the edge classification score for illustration

Computing resources during inference

Prediction sample

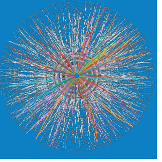
- 100 events from TrackML
- Graphs have similar size as those obtained with ITK

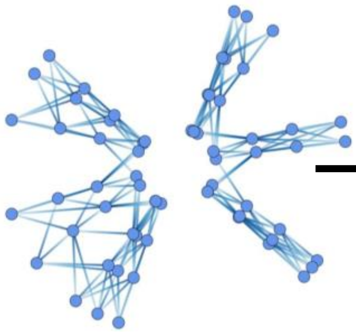
	Quadro RTX 8000	GeForce RTX 2080 Ti Gaming GPU
GPU memory capacity (GB)	48	11
Runtime mixed precision (16/32)	350 ms / event	
Memory peak consumption	5.4 GB	

Study on TrackML sample, without optimization.

No memory issue during prediction

Will also benefit from dedicated kernel => large factor of improvement expected from dedicated CUDA kernel

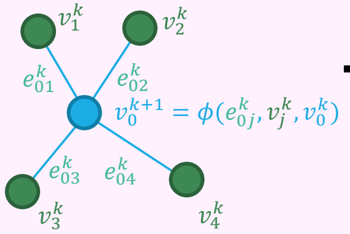




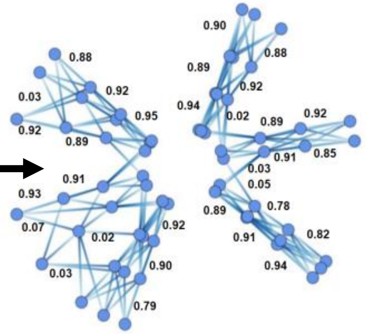
Graph

2

Graph Neural Network

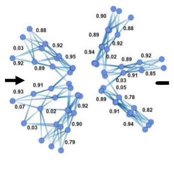


Edge Labeling



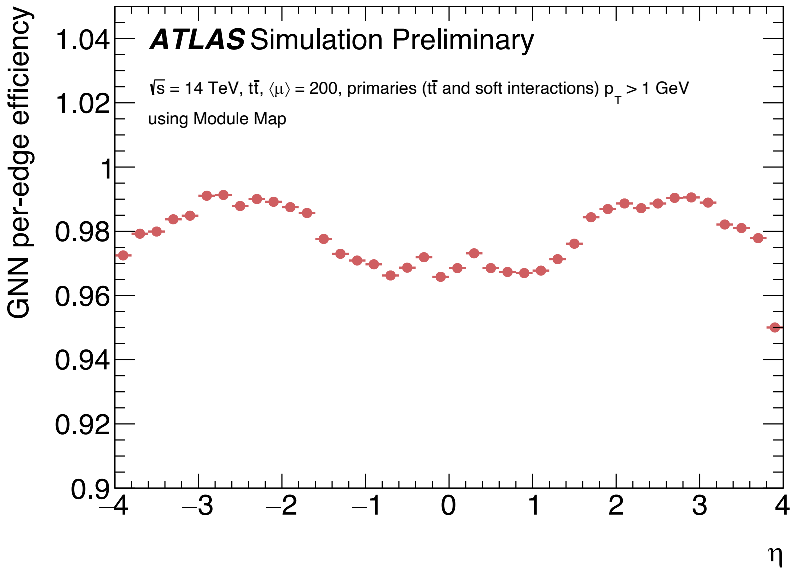
Edge Scores

GNN edge-level performance

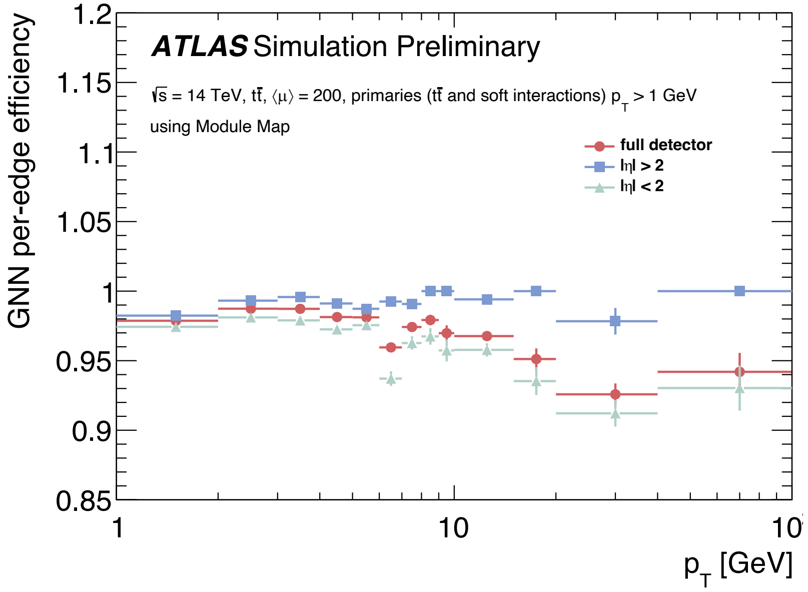

Edge Scores

Cut at s = 0.5 on the edge classification score

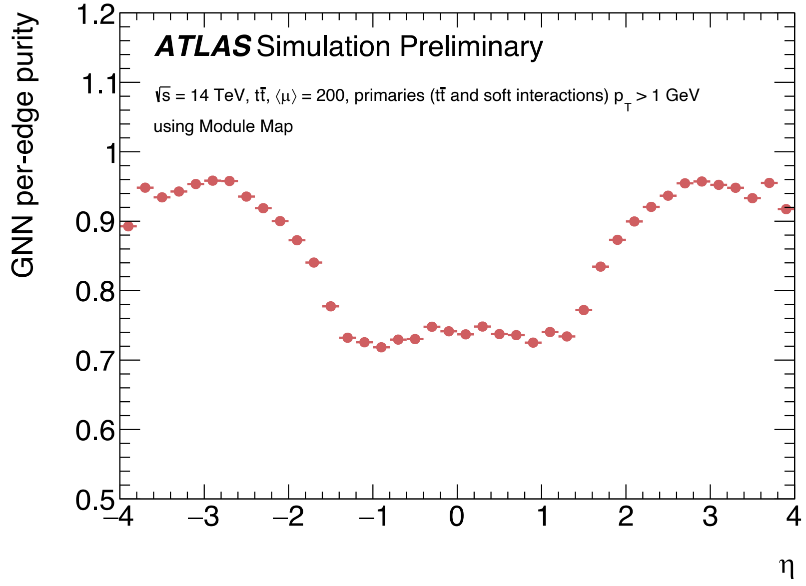
Efficiency vs. η



Efficiency vs. p_T



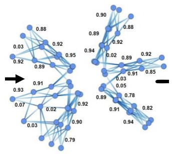
Purity vs. η



Efficiency and purity degradation in the central region.

➡ **What is the source of this inefficiency ?**

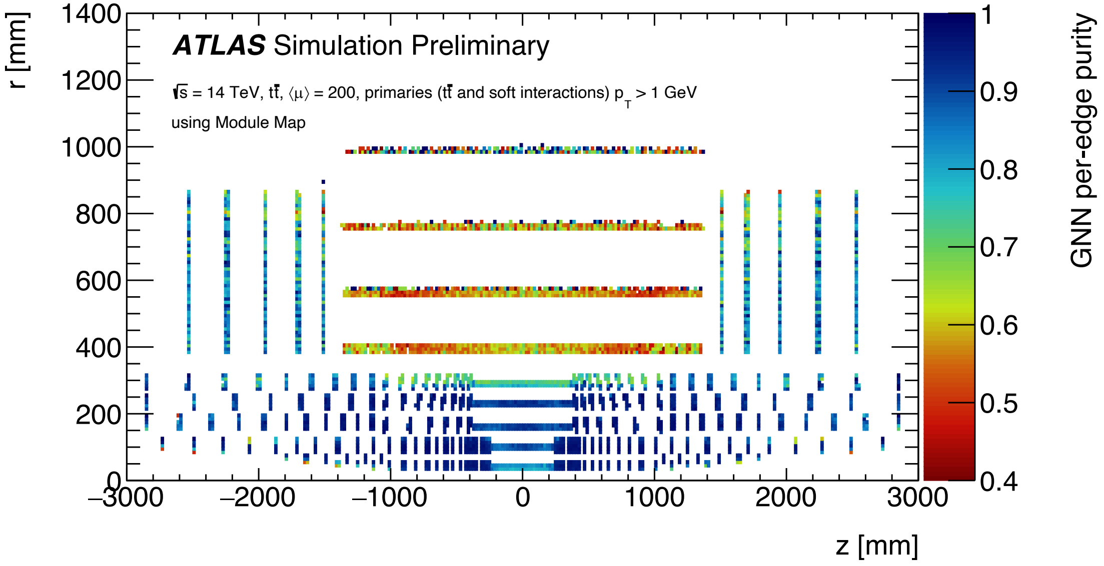
Investigation of the GNN edge-level performance



Edge Scores

- **Misclassification in the central region**

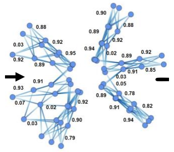
Before building the tracks, the GNN classification must be good. We applied a cut at $s = 0.5$ on the GNN edge classification score.



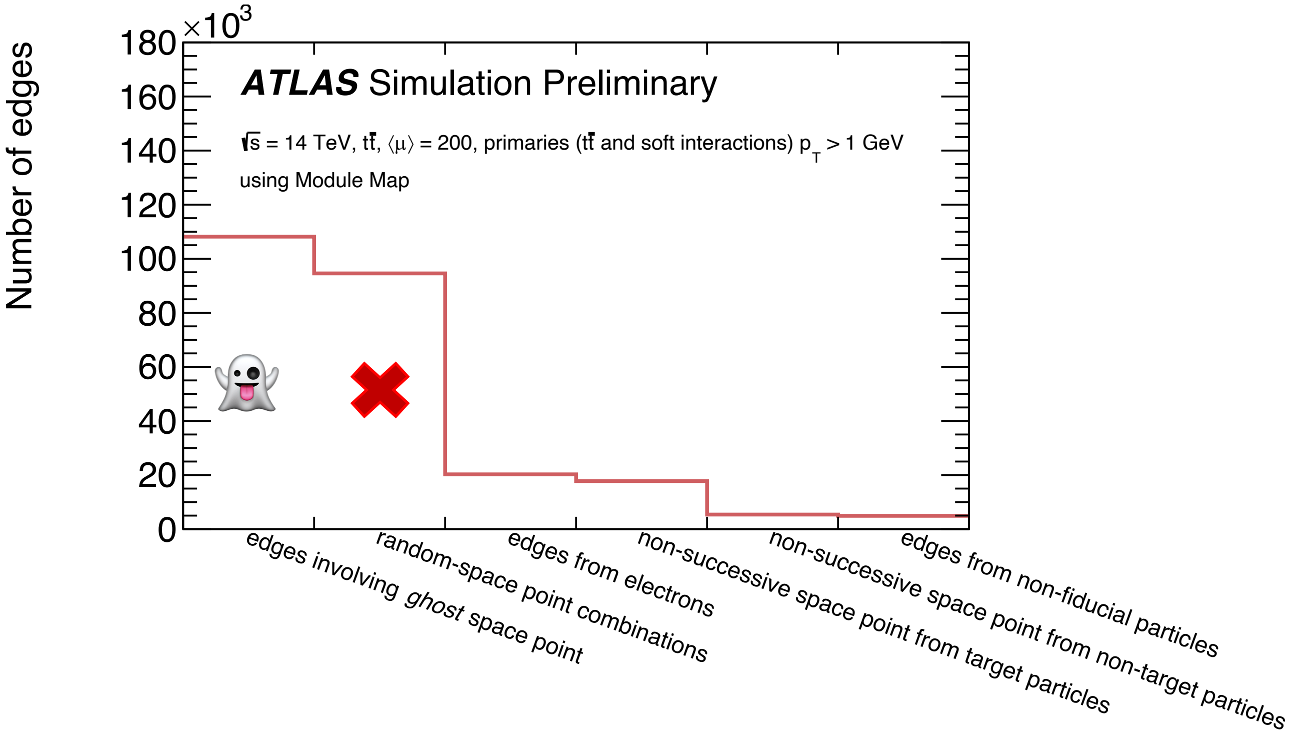
The largest misclassification arises in the barrel of the strip detector:

- Lower spatial space-point resolution in this region,
- Presence of ghost (👻) space-points.

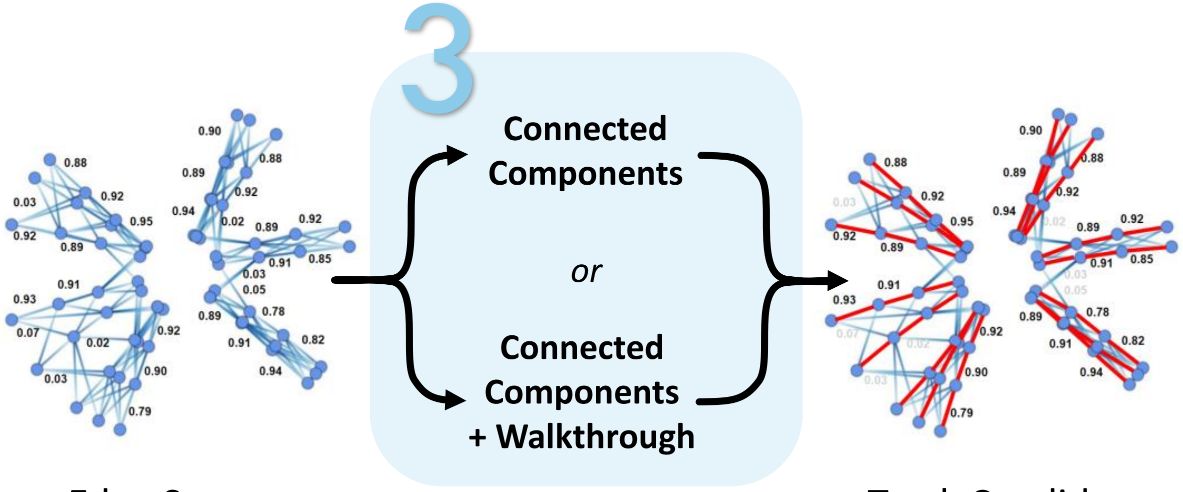
Investigation of the GNN edge-performance



- **Origin of misclassified edges**



Non-fiducial particles = particles with $|\eta| > 4$ or r > 26 cm



Edge Scores

Track Candidates

Graph Segmentation

Building track candidates

Legend

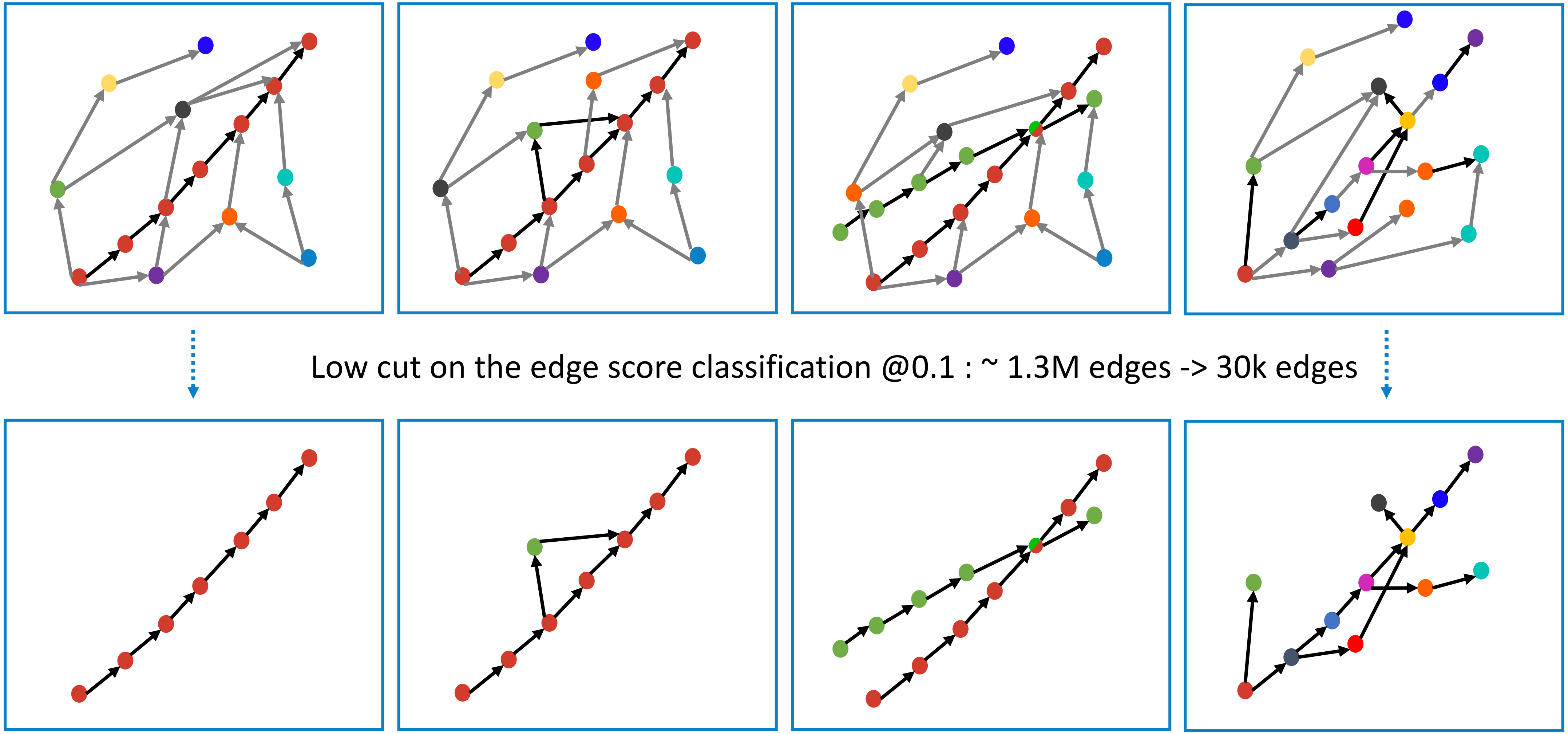
Edge below threshold

Edge above threshold

Nodes same color = Nodes same particles

Zoom on part of a graph

Connected component



Low cut on the edge score classification @0.1 : ~ 1.3M edges -> 30k edges

Building track candidates

Legend

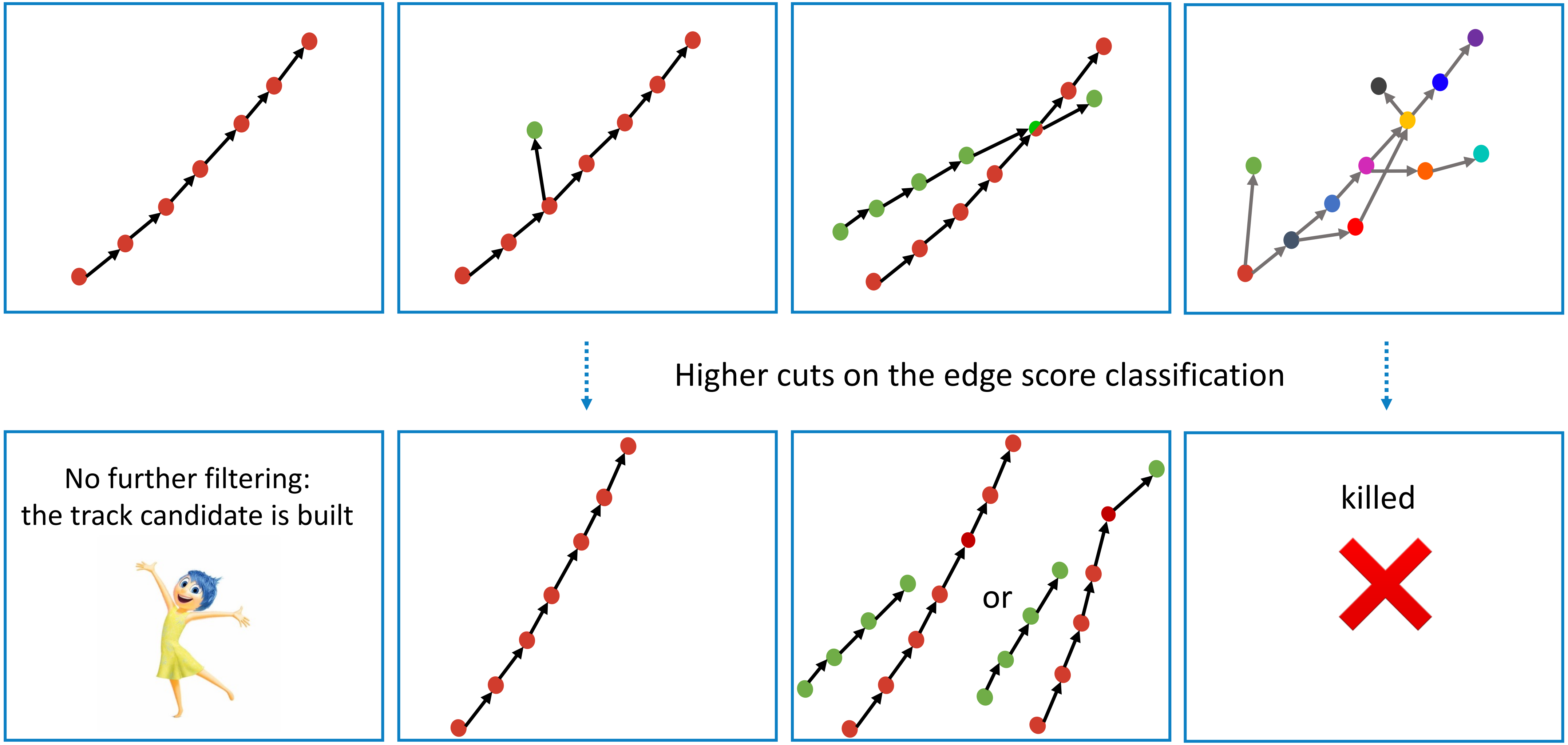
Edge below threshold

Edge above threshold

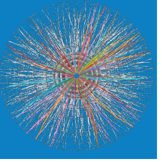
Nodes same color = Nodes same particles

Connected component

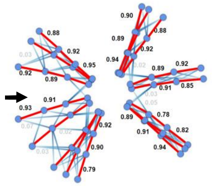
Walk-through algorithm
(from TrackML)



+ 2 possibilities



Track reconstruction matching criteria



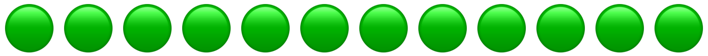
• Evaluation of the track candidates

No track fit is applied.

Evaluation done on $t\bar{t}$ + PU.

• Matching criteria

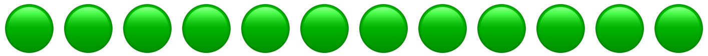
Particle



Track candidate

Standard matching

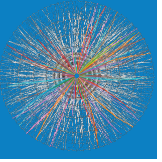
$P_{\mathrm{match}} > 0.5$



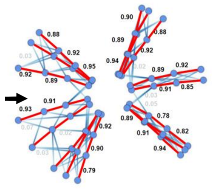
Strict Matching

$P_{\mathrm{match}} = 1$

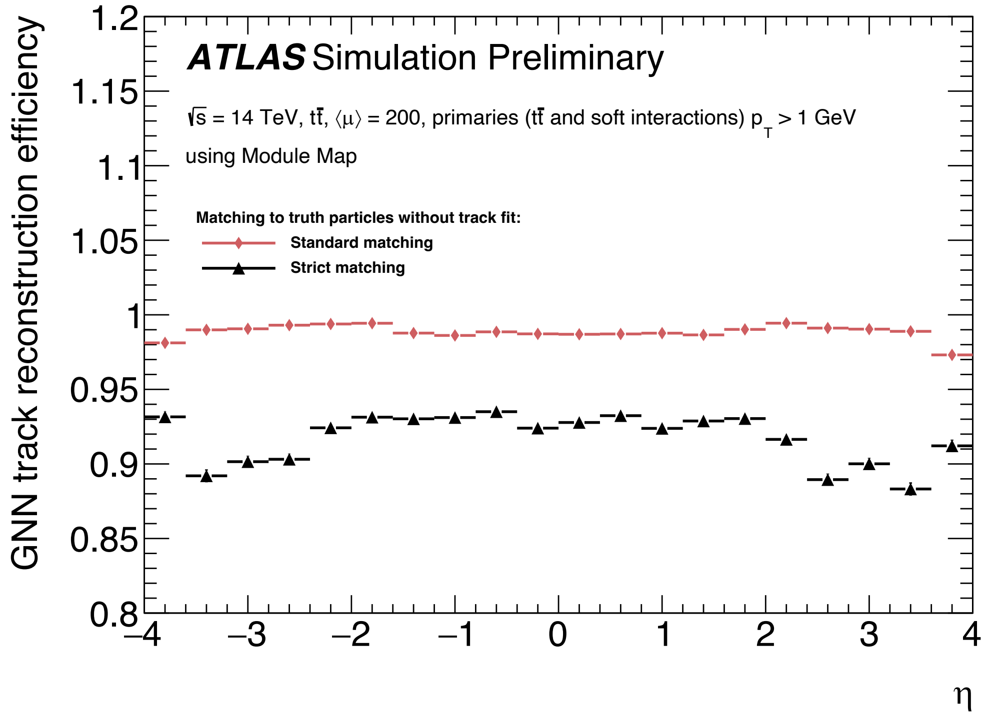
Track purity = 1

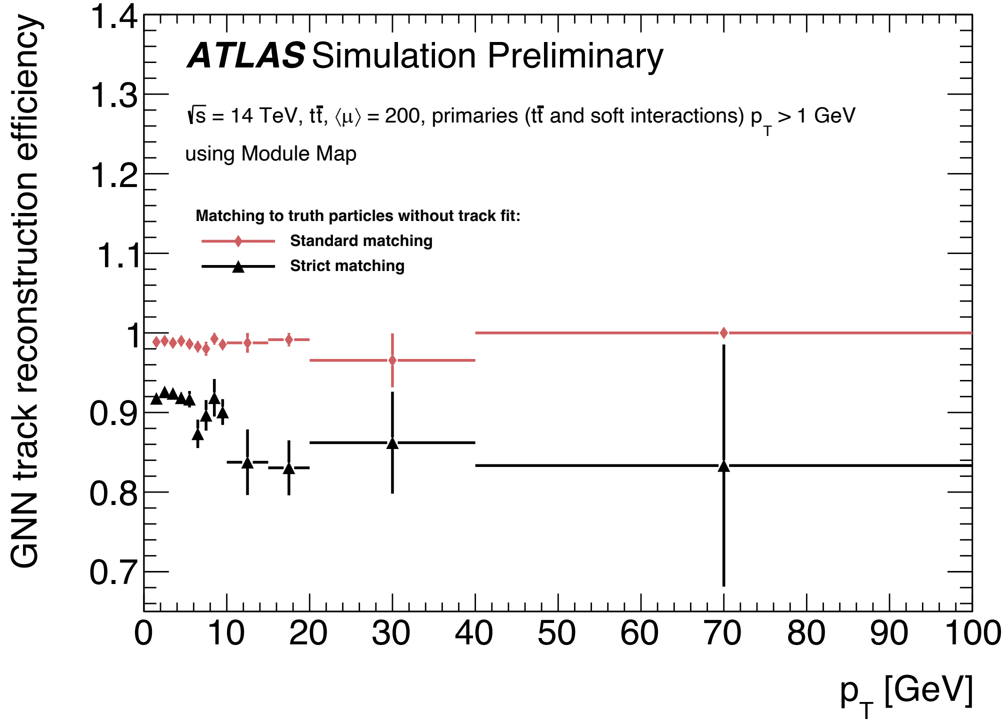


GNN track reconstruction efficiency



Track Candidates





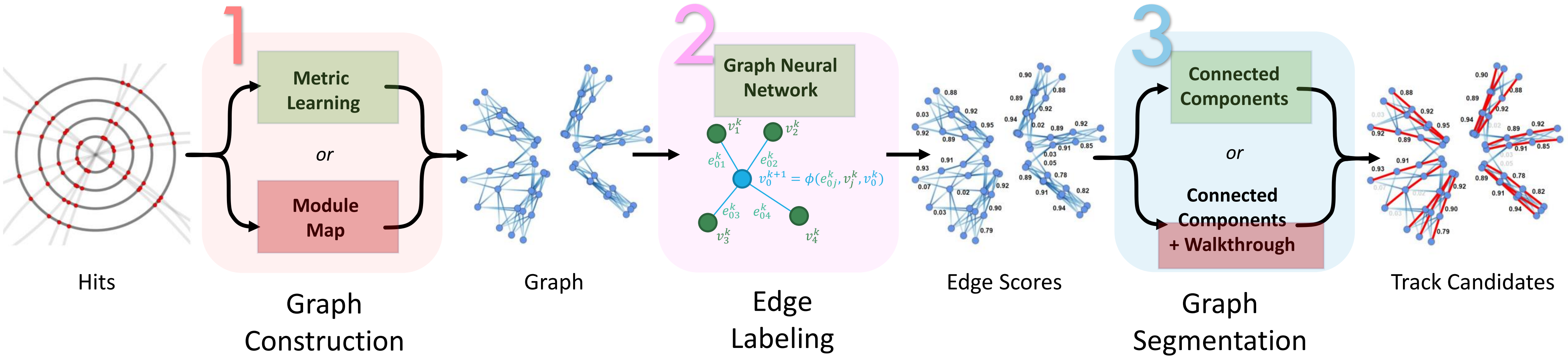
Track candidate not matched to any particle = fake track

found to be $O(10^{-3})$

GNN for tracking

GNN tracking achieved very promising results

We are very exciting to compare against CKF.



Available in ACTS

Not yet available, in progress

➡ **Part of the pipeline already available on ACTS ☺**

See tutorial available here

GNN for tracking and timing

Timing consideration

- The target is to run the full pipeline in **< 1 second.**
- Need to be fully run on GPU.

TrackML timing (Similar graph size as for ITk)

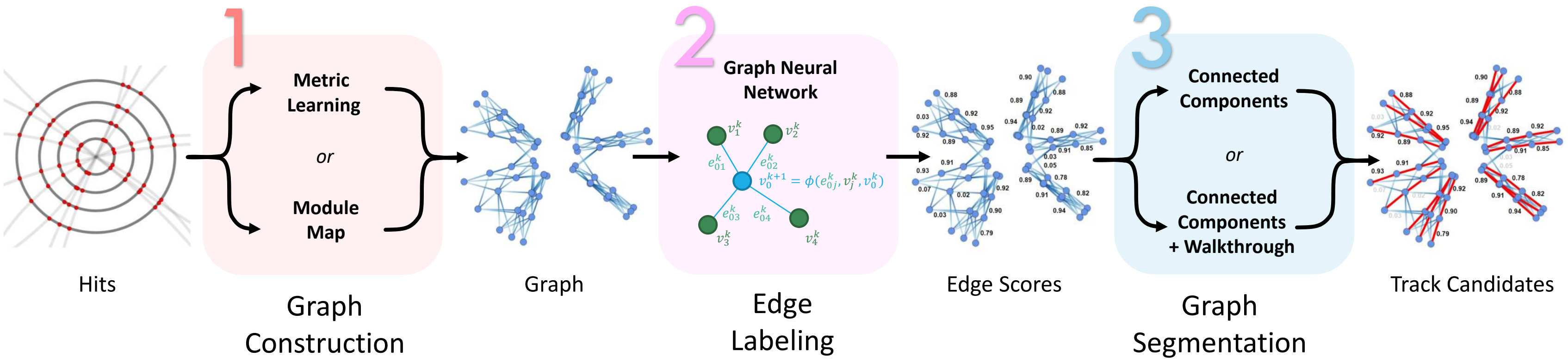
Pipeline step	V100 GPU
Graph construction (metric learning)	~ 500 ms
Graph construction (module map)	In progress target ~ 100 ms
GNN	~170 ms
Connecting component	~100 ms

(See this paper)

How to improve: GPU kernels have dedicated operation for NN. But the GNN model is much complex with its 8 message-passing operations and the way the memory is therefore handle.
Using dedicated GNN kernels could only improve the timing, the memory consumption and the energy cost.

GNN tracking framework

Public pipeline



We plan to made the code accessible in a few weeks.

To run the pipeline, simulated sample could be obtain from ACTS using the OpenDataDetector, or using official sample from experiment.

➡ **You are invited to contribute**

Contribution possibilities include:

- ➡ Creation of dedicated GPU kernel for GNN
- ➡ Faster graph construction
- ➡ Track building stage
- ➡ Heterogeneous GNN architectures (strip detector)
- ➡ Any part you fancy of GNN tracking

Many kinds of projects requiring different types of expertise.

Conclusion and prospects

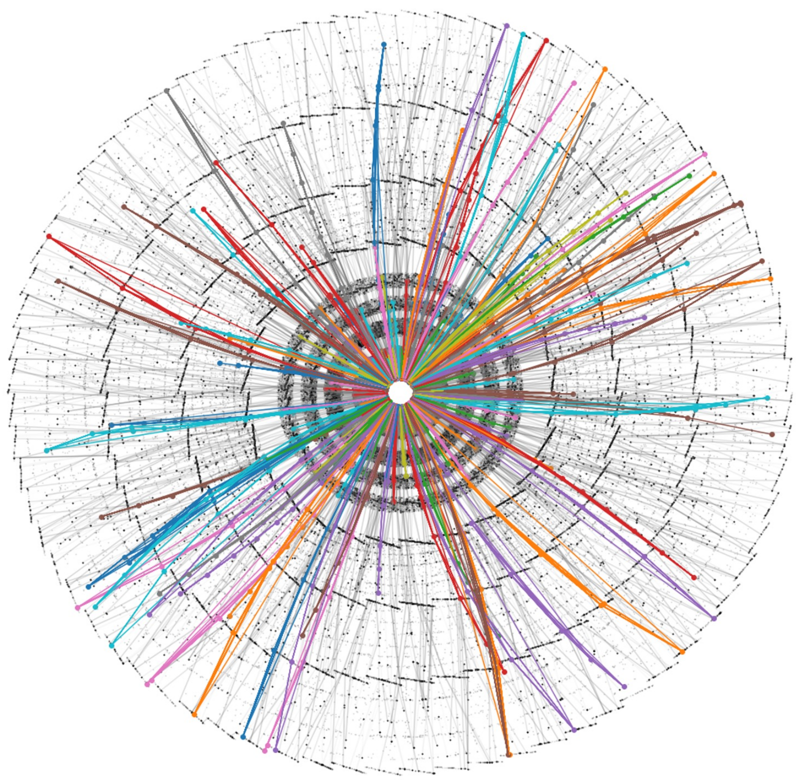
- **Conclusion**

First results using a GNN-based track reconstruction with ITk simulated data are promising and realistic.

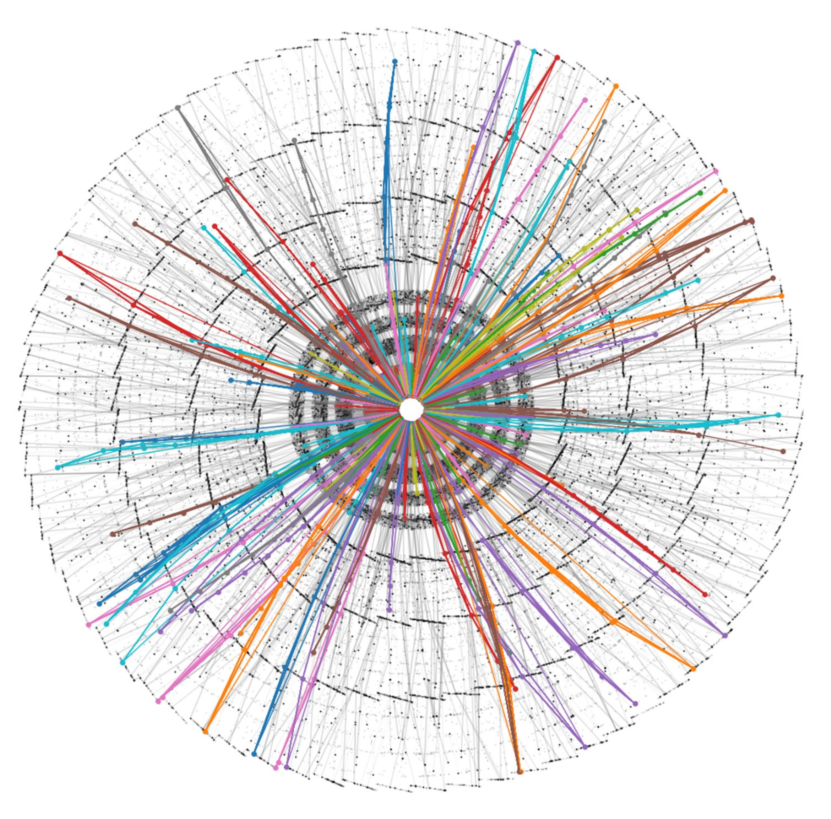
- **Prospects**

Several studies are ongoing:

- Finish integration ACTS and Athena
- Fair comparison (timing and reconstruction performances) with Athena and ACTS Combinatorial Kalman Filter
- New GNN architectures to fix the degradation of efficiency in the strips
- New track building stage, able to run on GPU



Thanks for your attention ☺



BACK-UP

Graph construction in GNN tracking

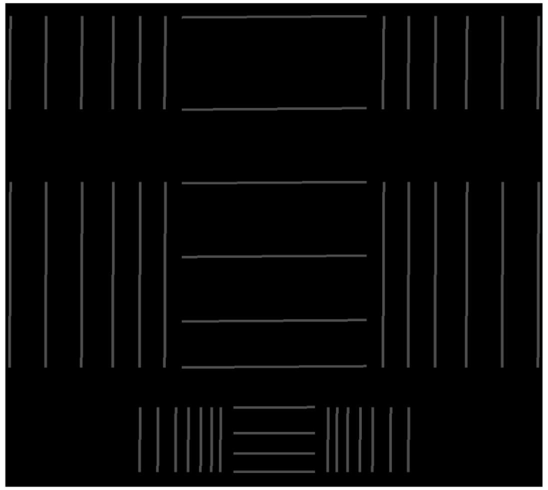
Graph construction in the GNN tracking community

Lot of development is made on graph construction, mostly using machine learning technics.

Reinforcement learning
environment TrackML

True hits

RL
predictions



See presentation by Liv Helen Vage @CTD

Per sector breakdown

Cut detector into multiple regions to simplify the graph construction:

- Very handy to start development
- Not really a realistic solution
- Can be found in proof of principle by Exa.TrkX projects