BERKELEY LAB

Bringing Science Solutions to the World

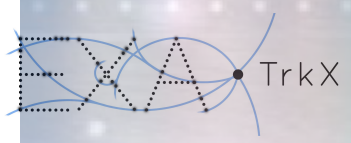
Performance of GNN-based tracking for ATLAS ITk

Xiangyang Ju

On behalf of the ATLAS Collaboration

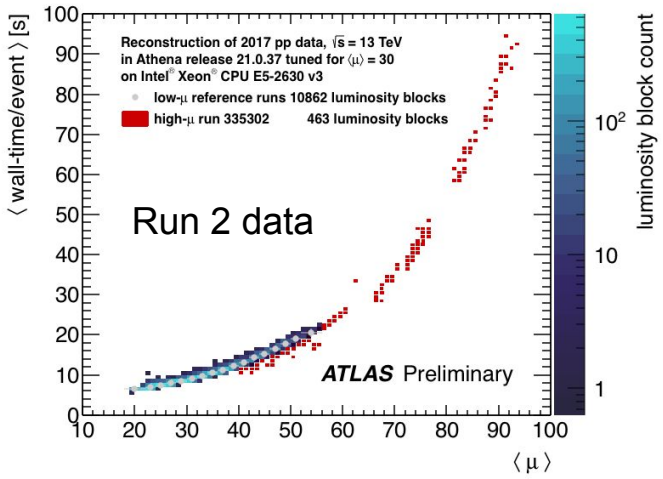
CHEP 2023, Norfolk Virginia

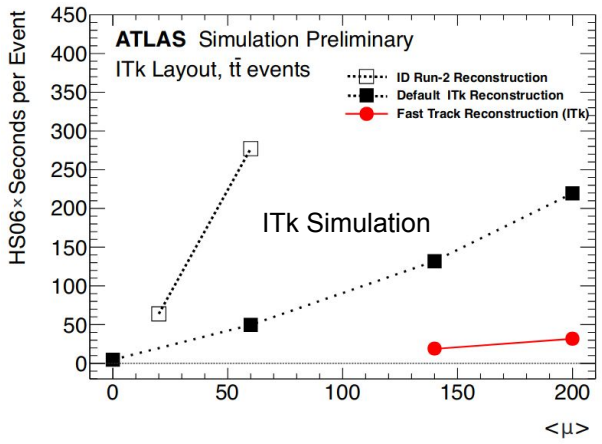
8 May 2023

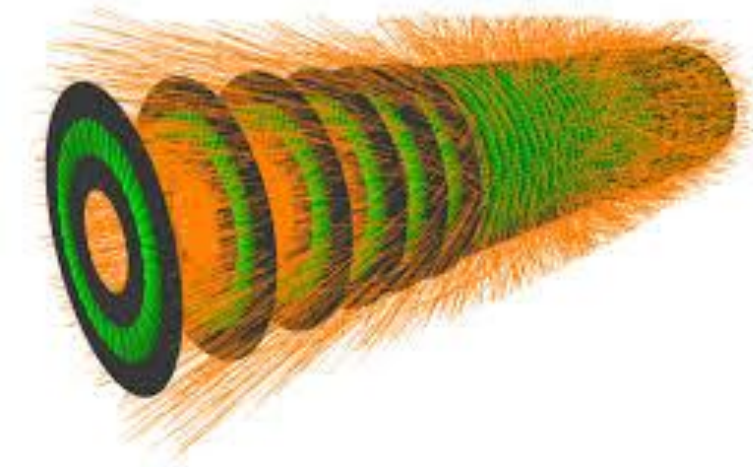


Particle tracking challenges

It’s computationally expensive





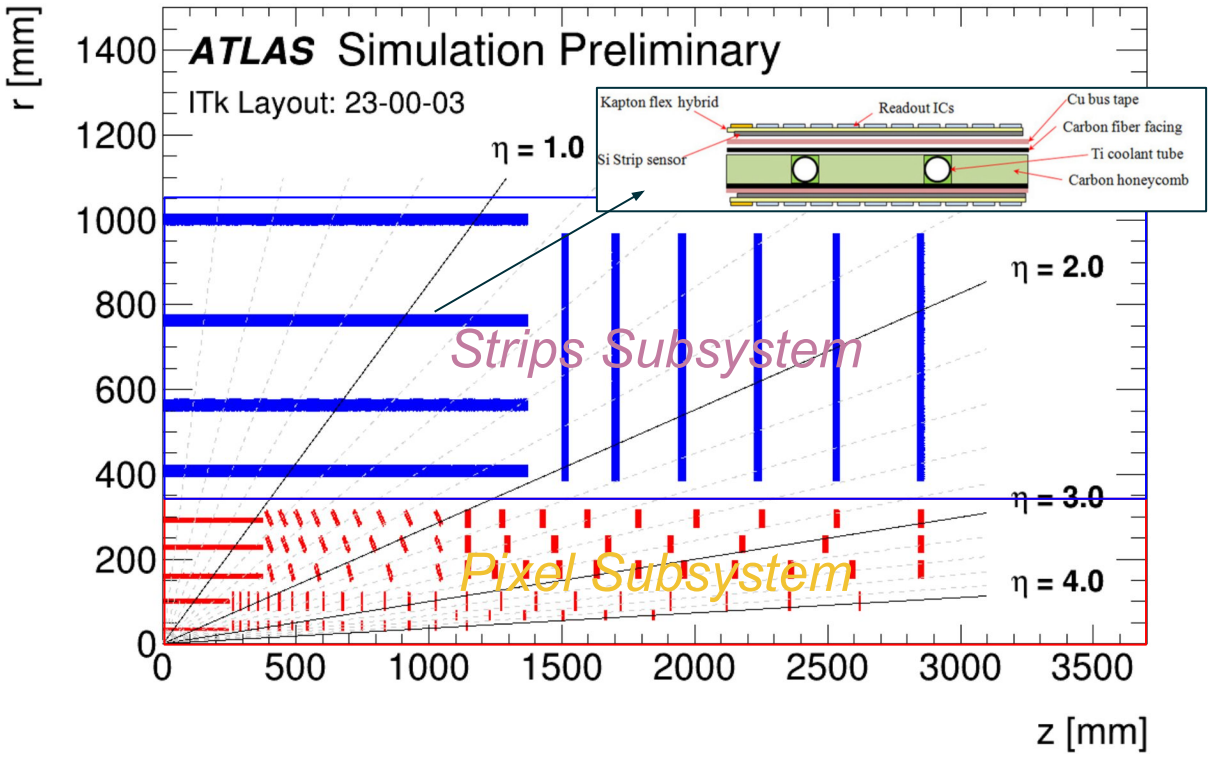


- Tracking takes ~40% of total reco time in ATLAS
- Existing track finding algorithm (e.g. CKF) does not scale well, and cannot be easily ported to GPUs
- Ongoing efforts to reduce the tracking reco time

- Pileup <μ> increased from ~50 to ~200
- Very dense environment: O(10 k) particles

ATLAS Inner Tracker (ITk) Upgrade

Goal: the ITk should have the same or better performance as the current detector for HL-LHC



S. Diez, Silicon strip staves and petals for the ATLAS Upgrade tracker of the HL-LHC

Pixel subsystem:

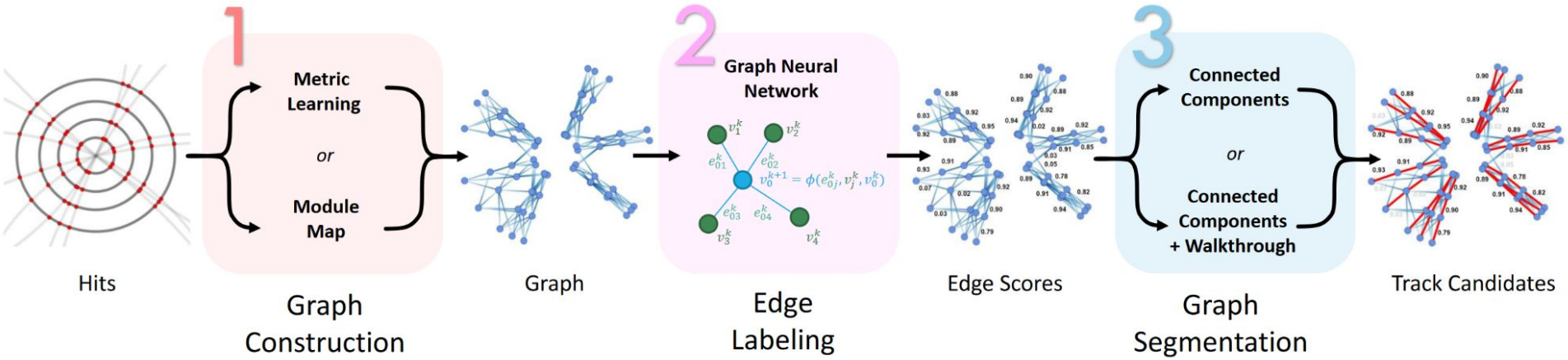
- Near beam, finer segmentation needed to separate tracks → drives impact parameter resolution
- Pixel pitch is 50x50 μm^2 (some 25x100 μm^2)
- **One spacepoint ←→ One cluster**

Strip subsystem:

- Covers large area, further from the beam → drives momentum and η resolution
- Sensitive silicon sensor elements long and skinny (75.5 μm x 24.1 or 48.2 mm)
- Double-sided sensors with a stereo angle
- **One spacepoint ←→ Two clusters**

The Graph Neural Network-based pipeline

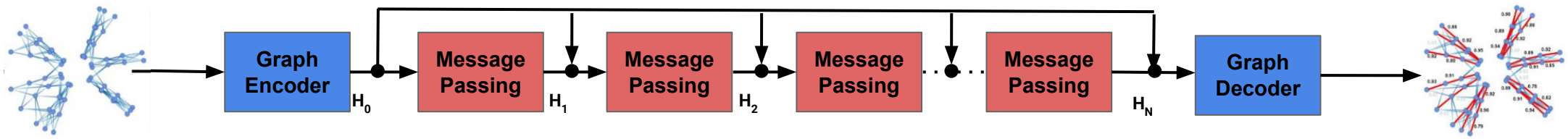
The pipeline presented in the CTD 2022, See C. Rougier's talk.



- A tracking graph: nodes are spacepoints and edges are possible connections between nodes. *True edges* are connections of nodes from the same particle of interest
  - We are targeting primary particles with p_T > 1 GeV, no electrons, no secondaries
- Graph Neural Network is an **Edge Classifier** that assigns scores to edges

CTD 2022 Graph Neural Network

It contains three major components.



Graph Encoder

- Map node inputs into a latent space

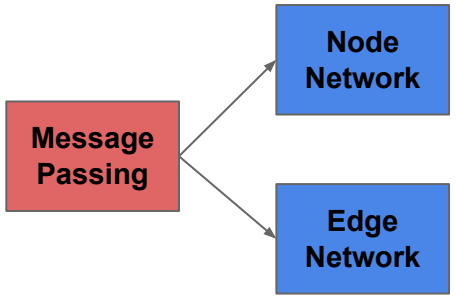
$$v_i' \leftarrow \phi^v(v_i)$$

- Map edge inputs into a latent space

$$e_k' \leftarrow \phi^e(e_k)$$

Message Passing Module

→ We use **the same** Message Passing modules for each message passing step



$$v_i' \leftarrow f^v(v_i^0, v_i, \bar{e}_i^r, \bar{e}_i^s)$$

$$\text{where } \bar{e}_i^{r/s} \leftarrow \sum_{(r/s)_k=i}(e_k, e_k^0)$$

$$e_k' \leftarrow f^e(e_k^0, e_k, v_{r_k}', v_{s_k}')$$

Graph Decoder

→ We apply a simple MLP to edges to get edge scores

Call for GNN improvements

GNN did not perform well in the Strip barrel region

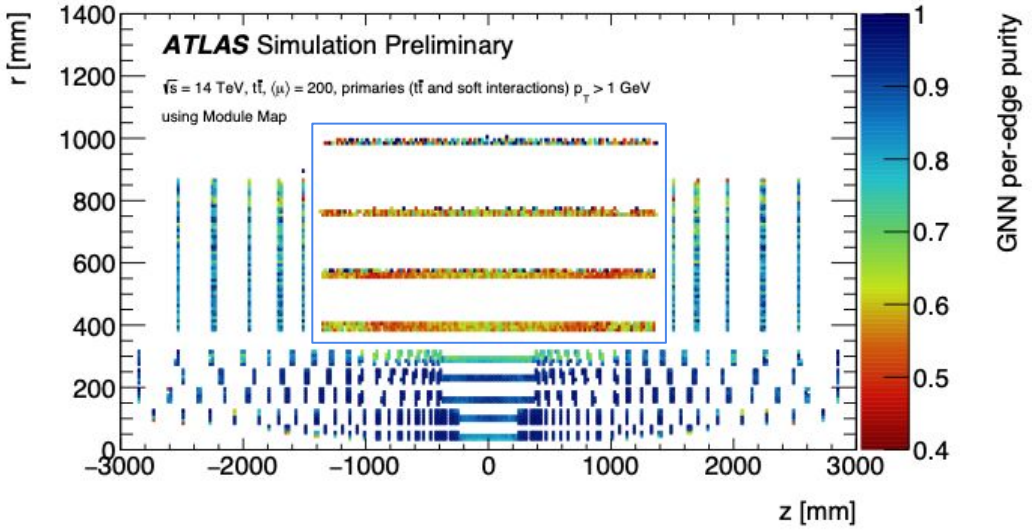
While attaining good per-edge efficiency (>98%), the per-edge purity is only about 50% for the strip barrel region

Per-edge efficiency: true edges passing the threshold / total true edges

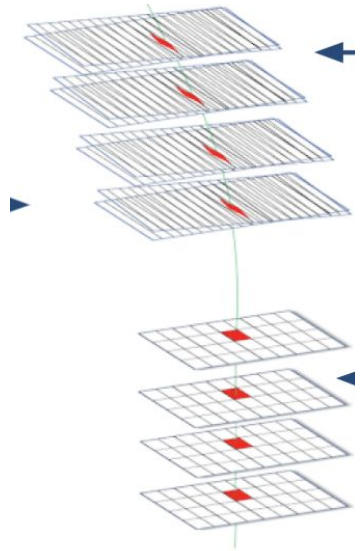
Per-edge purity: true edges passing the threshold / total edges passing the threshold

We explored the **heterogeneity** in the GNN-based track finding.

GNN Per-Edge purity evaluated for different detector regions with a score threshold of 0.5



Heterogeneity in GNN tracking



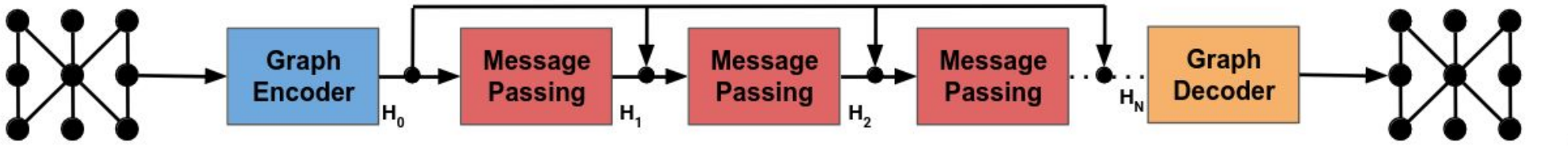
Heterogeneous data:

- Pixel detector: one spacepoint = one cluster, [r, φ, z]
- Strip detector: one spacepoint = two clusters, [r, φ, z] + cluster one + cluster two

In CTD 2022 results, the two cluster information for the strip SP **was not used.**

Heterogeneous GNN:

- In the Graph Encoder, use different MLPs to encode Strip and Pixel spacepoints differently
- Or / And in the message passing, encode messages differently for Pixel and Strip spacepoints



Explore heterogenous data

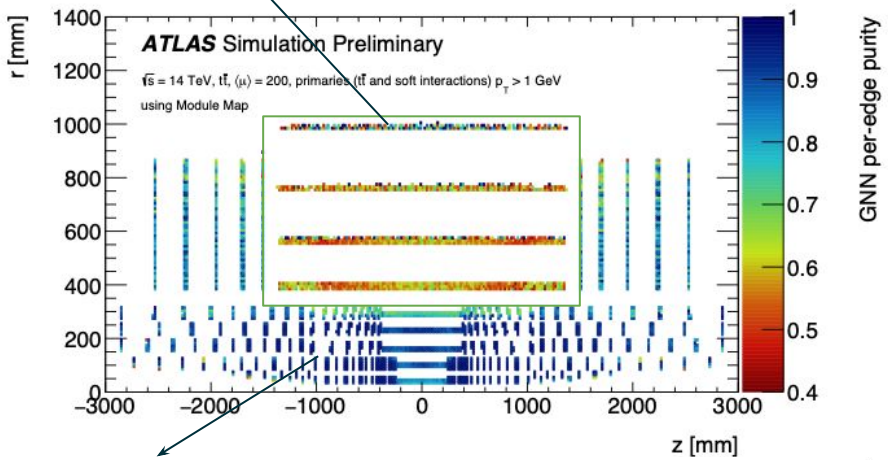
Key idea: Add cluster features to spacepoint features

- For Strip spacepoints in barrel region, add the two associated cluster information
- For Pixel spacepoints and Strip spacepoints in endcap region, repeat its features to reach the same length

The GNN model is re-trained with the "extended node features". We call the trained model as the "*Extended GNN*"

In the Extended GNN, we use different Message Passing Modules for each message passing step

$(r_{cluster1}, \varphi_{cluster1}, z_{cluster1}, \eta_{cluster1}, r_{reco}, \varphi_{reco}, z_{reco}, \eta_{reco}, r_{cluster2}, \varphi_{cluster2}, z_{cluster2}, \eta_{cluster2})$



$(r_{reco}, \varphi_{reco}, z_{reco}, \eta_{reco}, r_{reco}, \varphi_{reco}, z_{reco}, \eta_{reco}, r_{reco}, \varphi_{reco}, z_{reco}, \eta_{reco})$

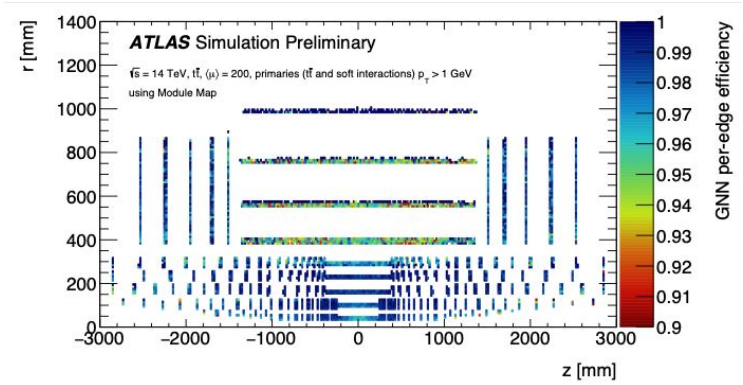
CTD 2022 GNN vs Extended GNN

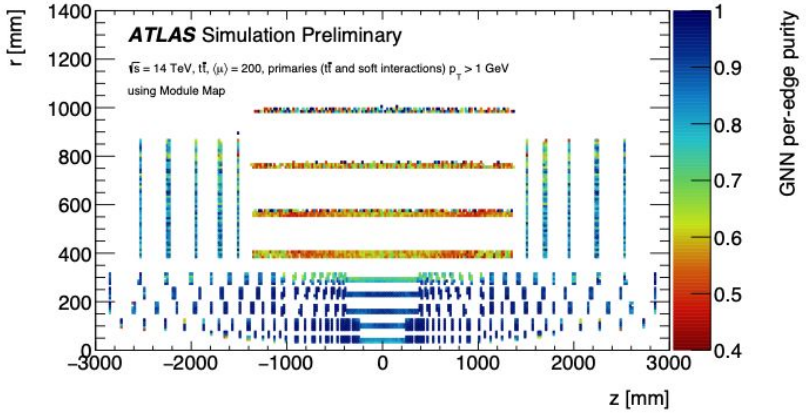
They use the same inputs graphs constructed from Module Map

Extended GNN results in similar Per-Edge **efficiencies**

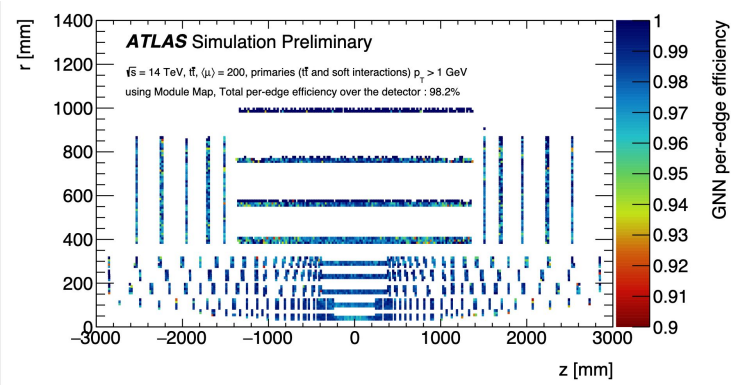
Extended GNN visibly improves the Per-Edge **purities** in Strip barral region

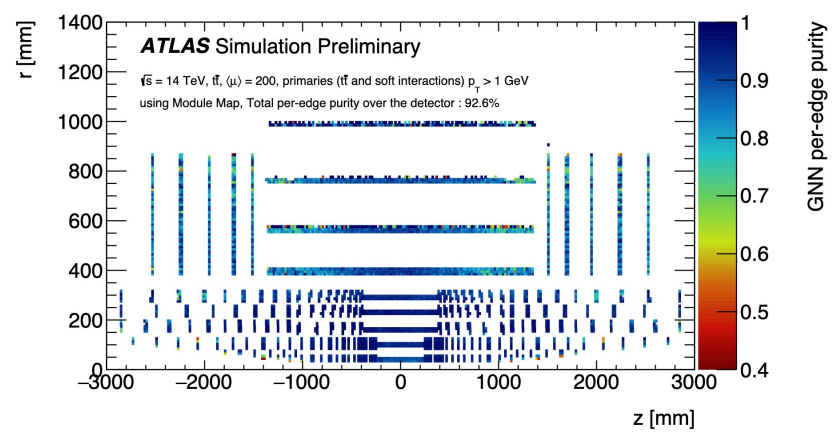
CTD 2022 GNN





Extended GNN

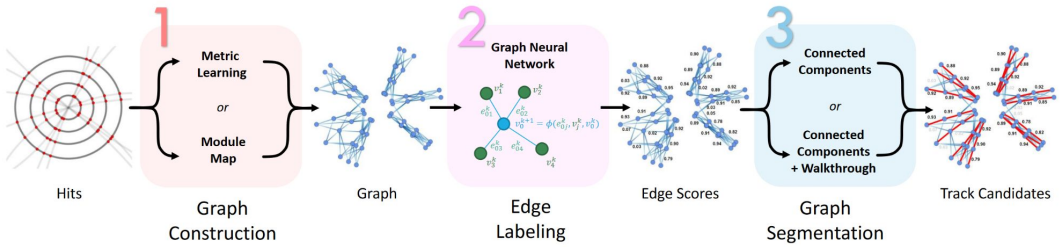


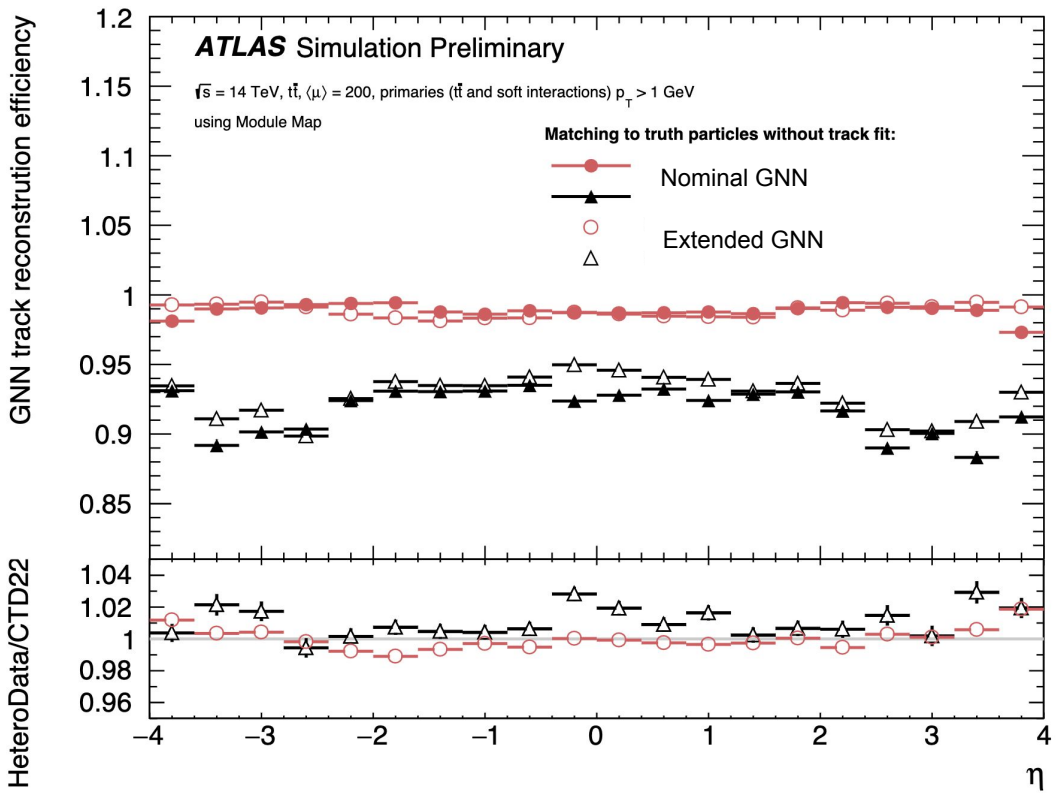


Track reconstruction efficiency

Use the Extended GNN for graph segmentation

- Thanks to the much higher Per-edge purity, the portion of tracks reconstructed with the Connected Component is increased
  - *Connected Components can be executed in GPUs via the cuGraph*



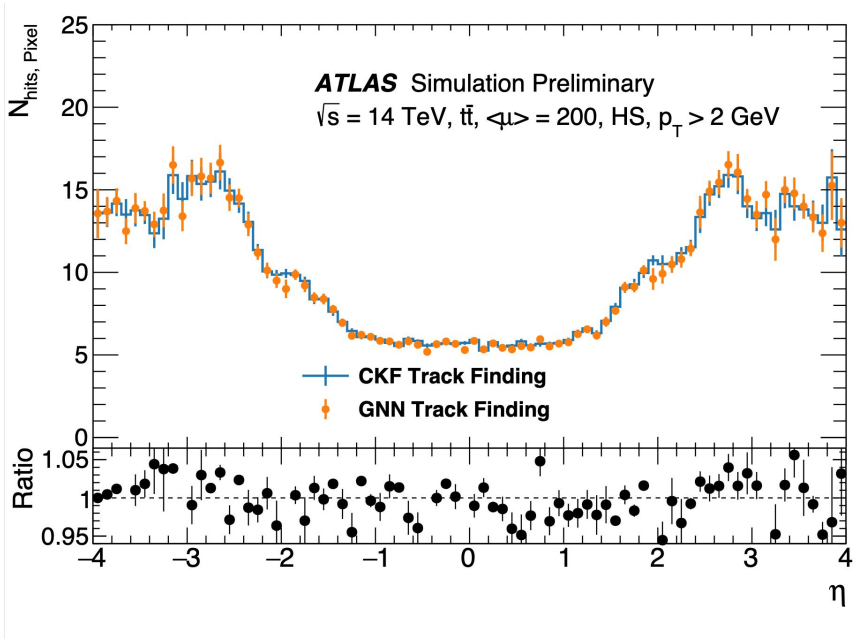


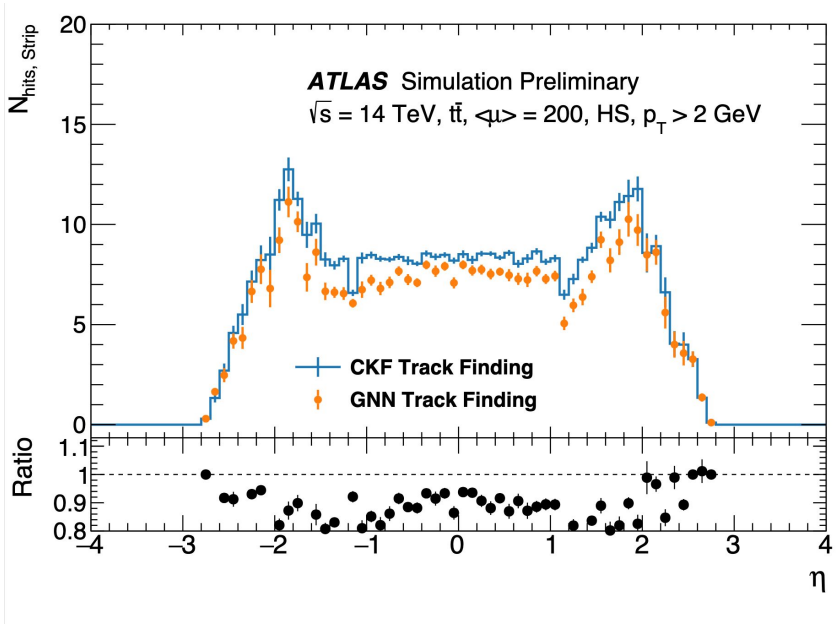
GNN track reconstruction efficiency calculated with two matching schemes:

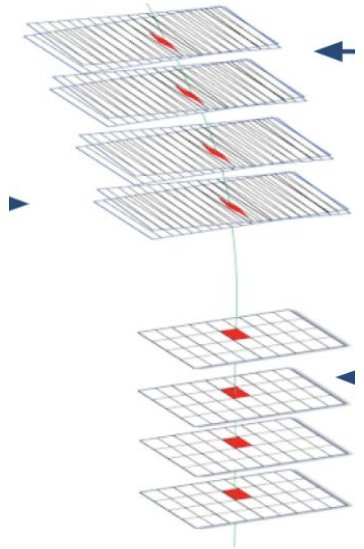
- Red circle (“standard matching”) : > 50% of the spacepoints in the reconstructed track are matched to a true track
- Black triangle (“strict matching”): 100% of the spacepoints in the reconstructed track are matched to a true track

Track content comparison

We use the default ITk reconstruction in ATLAS as a reference





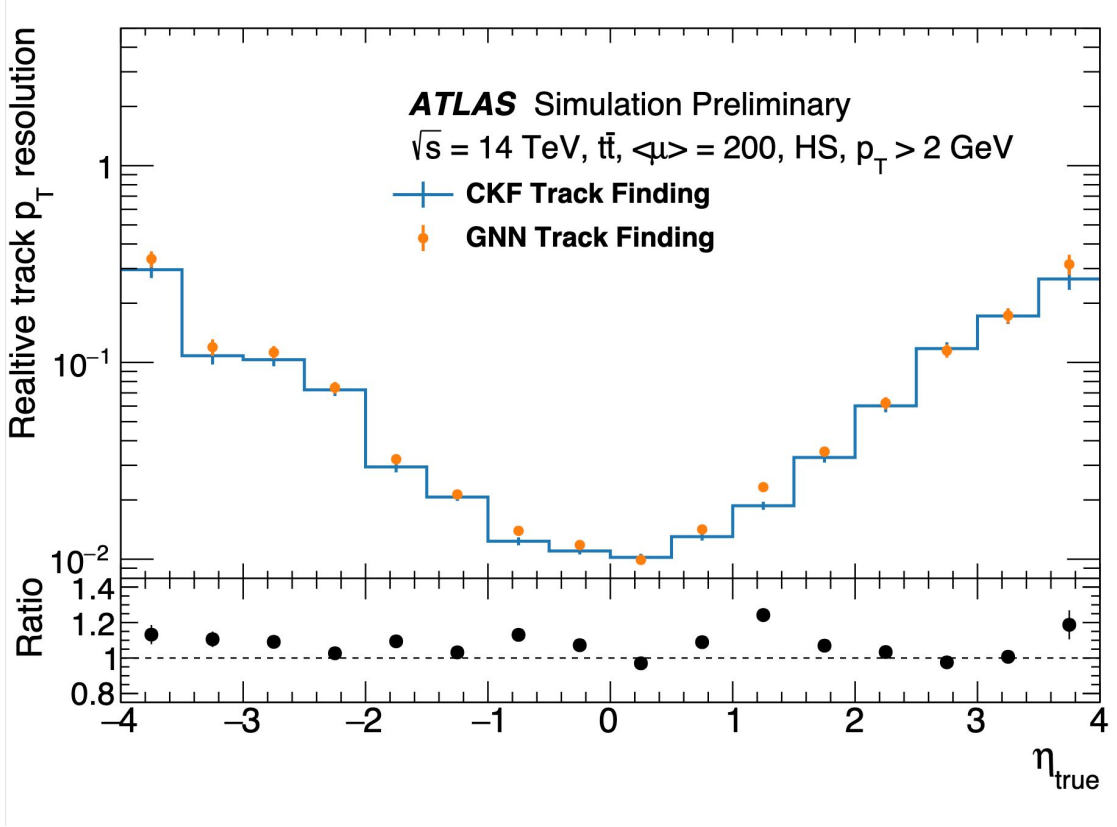


- GNN tracks have the same number of Pixel hits per track as the default ITk reconstruction
- GNN tracks have less Strip hits per track, possibly due to
  - missing clusters (those not forming a spacepoint will never enter GNN track candidates)
  - wrongly assigned clusters

Track parameter resolution comparison

We use the default ITk reconstruction in ATLAS as a reference

- GNN track candidates are fitted by the standard global χ2 fitting algorithm [ATL-PHYS-PUB-2019-014] implemented in the Athena framework. The same algorithm used for the default ITk construction
- GNN track finding yields similar track parameter resolution as the default ITk reconstruction algorithm
- Fitted GNN tracks are readily usable for downstream tasks



Relative track p_T resolution is measured as the multiplication of p_T^{true} and the RMS of the pull distribution of $(q/p_T^{reco} - q/p_T^{true}) / q/p_T^{true}$.

Explore Heterogeneous GNN

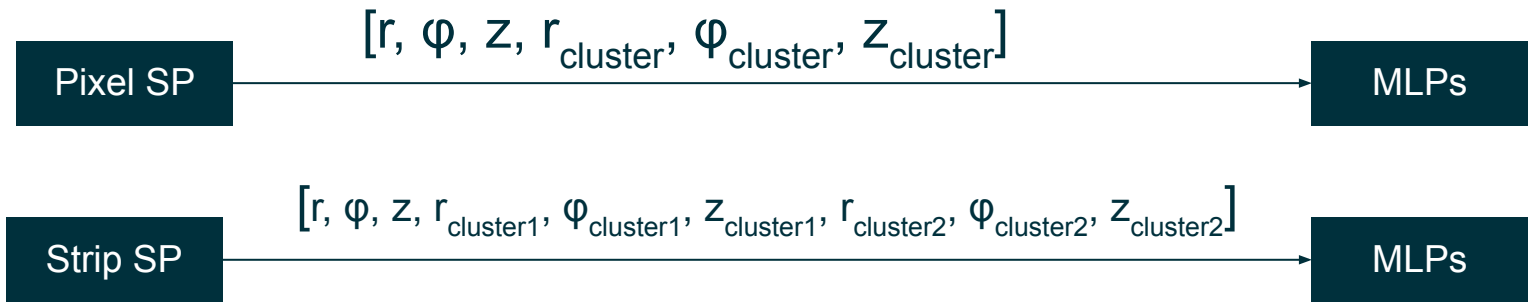
We compare the extended GNN with the heterogeneous GNN

Experimental setup

- Graphs constructed from the metric learning
- Use the "*extended spacepoint features*" without eta
- But ***do not*** pad pixel spacepoints with its features to reach the same length

Heterogeneous GNN

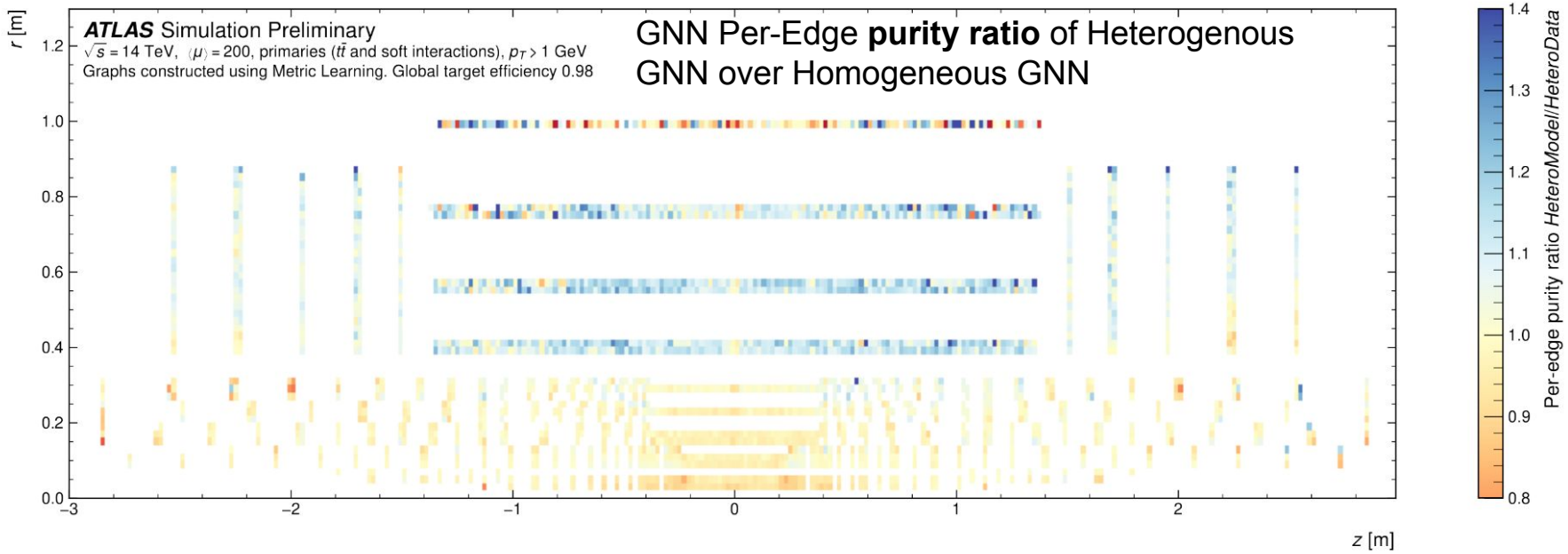
- Use a heterogenous Graph encoder



Heterogeneous GNN vs Extended GNN

While keeping the same per-edge efficiency (98%), we compare their per-edge purities

- The average total purity is 94% for both models
- Adding model heterogeneity results in up to 11% improvement in GNN per-edge purity in the Strip barrel region, with ~1% loss in the Pixel subsystem
- Room for improvement e.g. to try heterogenous message passing



Conclusion

- The GNN-based pipeline provides not only competitive track efficiency but also high quality track parameter resolutions.
- The GNN-based track finding is integrated into ACTS and ATLAS tracking framework, enabling us to use the existing tools to perform track fitting and evaluate tracking performance
- Significant improvement are achieved for the Graph Neural Network
  - While keeping the same efficiency, the extended GNN improved the edge-level purity by more than 30%. The high per-edge purity simplifies the graph segmentation
- With the above improvements, the GNN-based particle tracking steps steadily towards the production-level quality
- We are investigating further different heterogeneous GNN models and their impacts
- We developed a *CommonFramework* for GNN tracking
  https://github.com/gnn4itkteam/commonframework