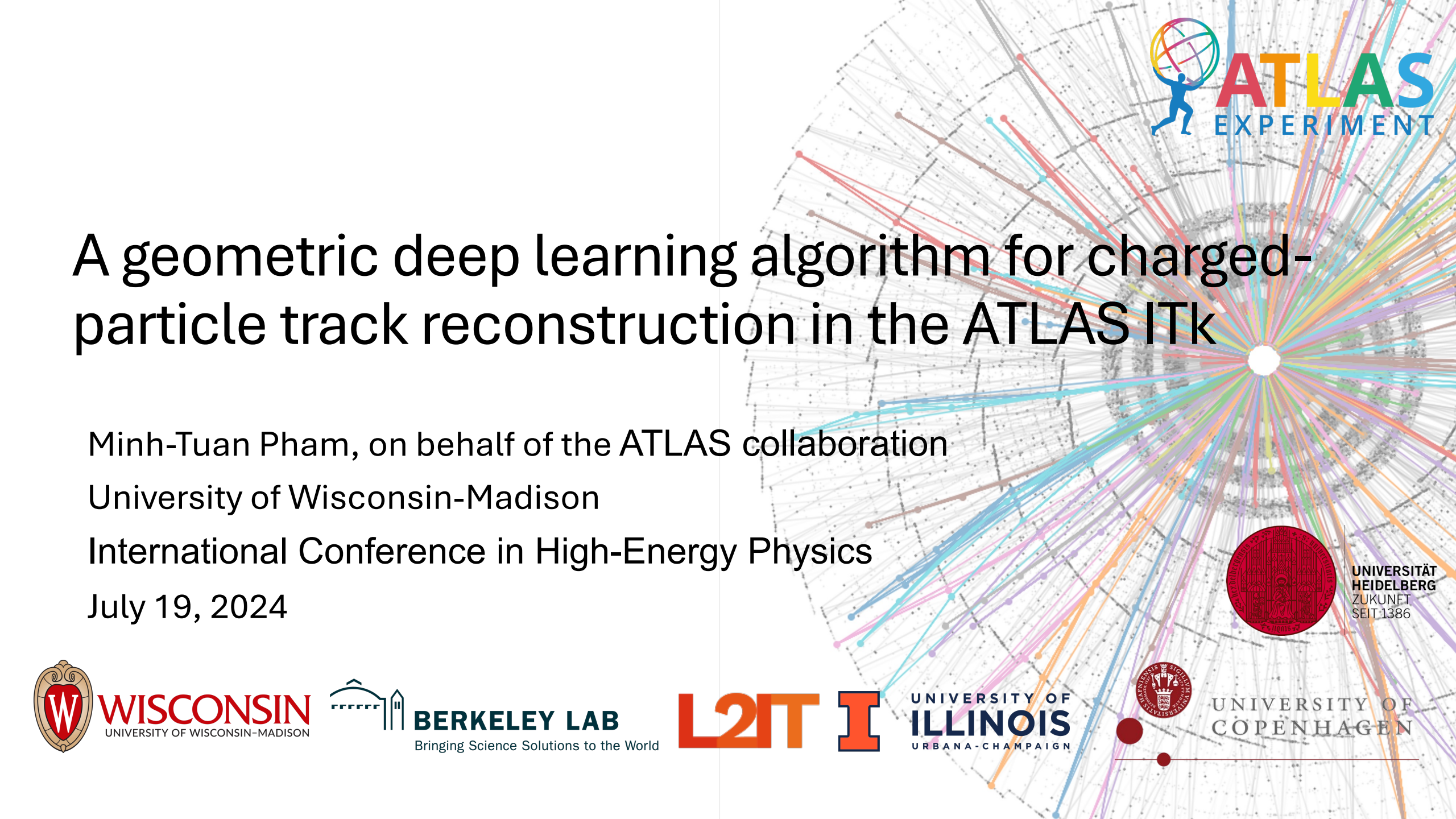


A geometric deep learning algorithm for charged-particle track reconstruction in the ATLAS ITk

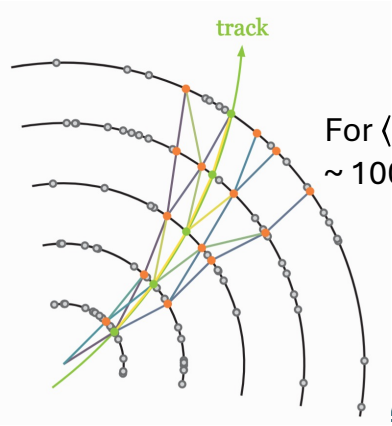
Minh-Tuan Pham, on behalf of the ATLAS collaboration

University of Wisconsin-Madison

International Conference in High-Energy Physics

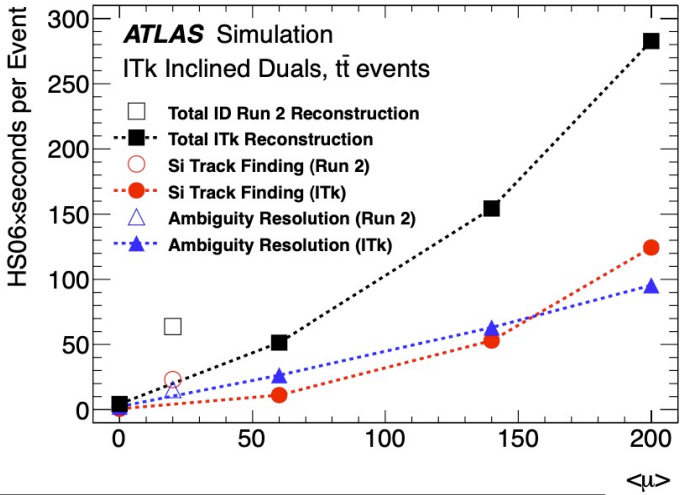
July 19, 2024

The tracking problem

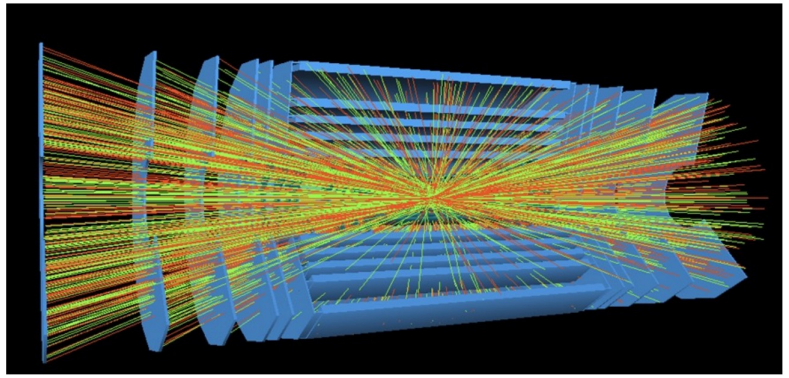


For $\langle\mu\rangle = 200$, ~ 400k hits,
~ 1000 Interesting particles

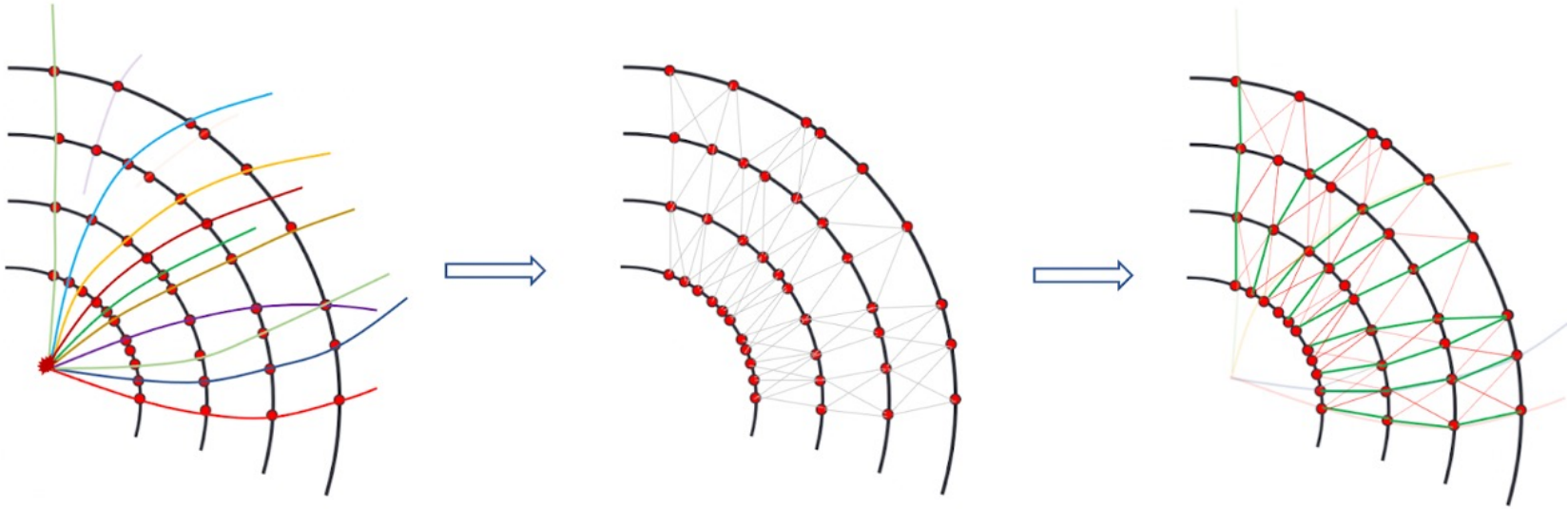
ATLAS-TDR-030



- In a collision event, charged particles leave energy deposits in the detector. Track reconstruction recreates particle trajectories from these deposits (hits).
- At $\langle\mu\rangle = 200$, need to reconstruct O(1000) target particles from 400k hits/BX. Huge combinatorics, most expensive process in offline event reco.
- HEP community seeks to develop hardware-accelerated, ML-based tracking algorithms.[1]
- We build a machine learning pipeline based on Graph Neural Network (GNN) for track finding under HL-LHC condition ($\langle\mu\rangle = 200$).
- In this presentation, we discuss
  - An overview of our machine learning pipeline,
  - A comparison in tracking performance with the state-of-the-art,
  - Throughput optimizations.



A graph-based approach to tracking



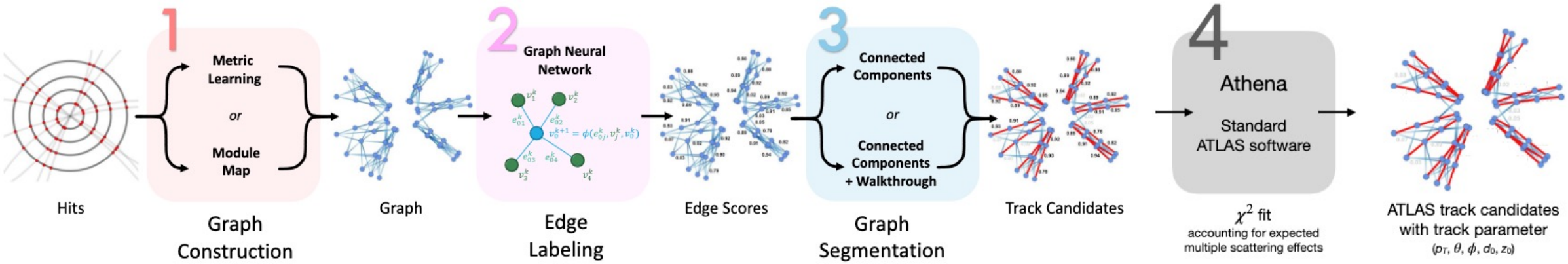
Detector hits | Graph | Graph with edges classified

Hits: Clusters of energy deposits in detector

Graph: A collection of nodes and connecting edges.

- Represent each collision event as a graph, each hit as a node.
- Nodes are connected by edges, representing the possibility of being consecutive hits on the same track.
- Classify edges with a pattern recognition algorithm.

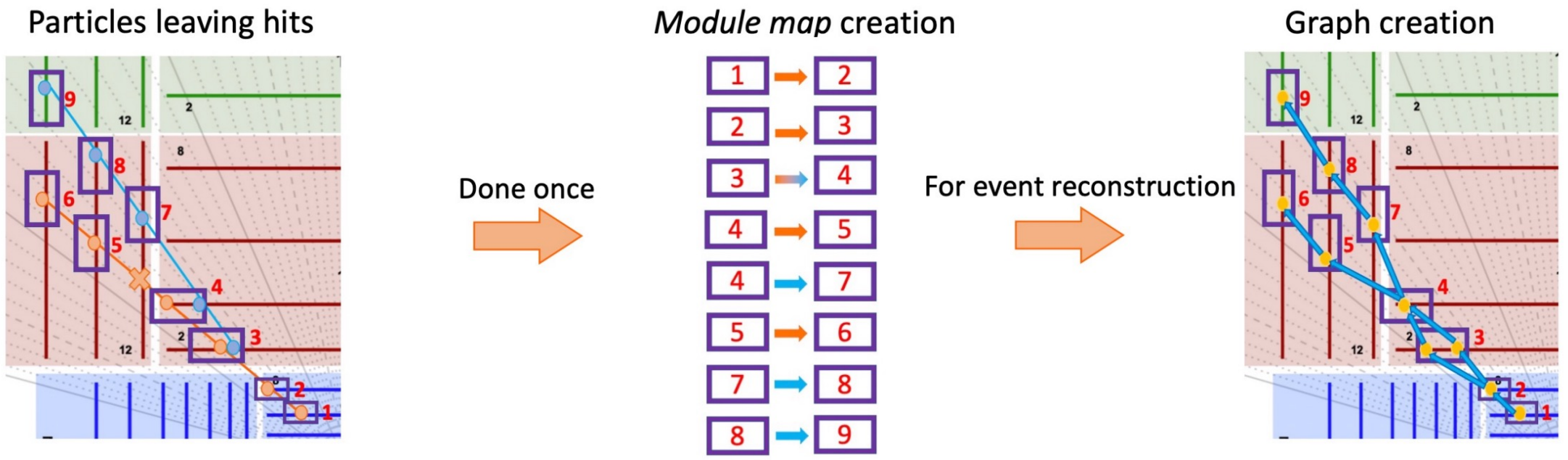
The GNN4ITk reconstruction chain



- Construct a graph from hits.
- Classify edges with a Graph Neural Network (GNN).
- Segment the graph to build track candidates.
- Fit track candidates and evaluate track reconstruction performance.
- Git repo, documentation.

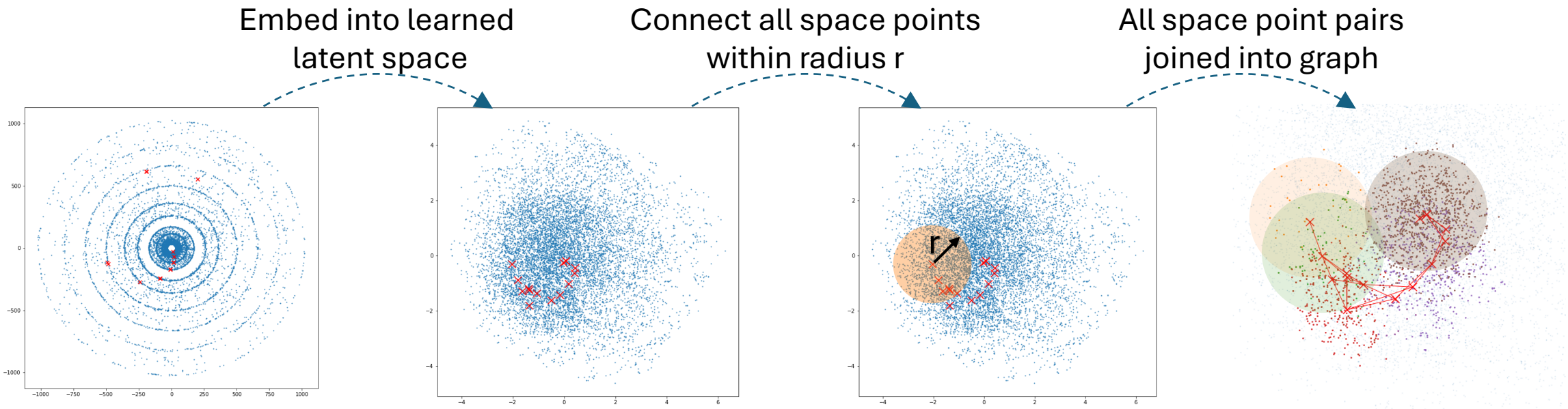
Graph construction: Module Map

- Data driven approach: Build a list of connections between detector modules from 90k simulated events.
- Inference: Connect modules that are simultaneously hit if there is a connection between them in the module map.



Graph construction: Metric learning

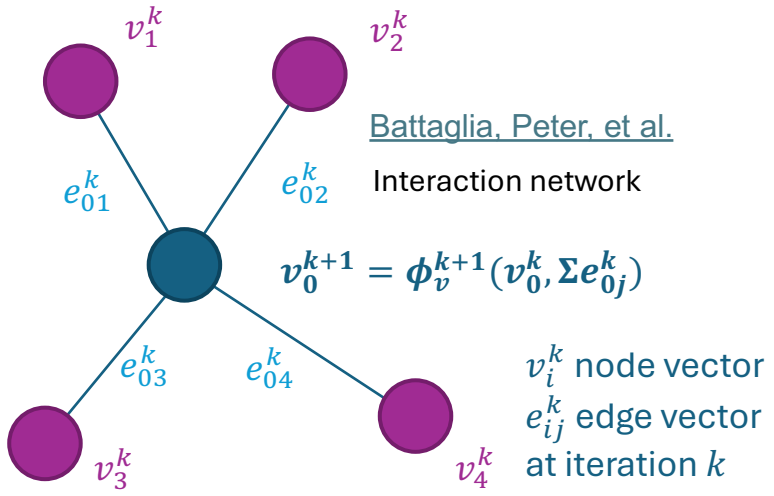
- **The idea:** Embed hit features to a high-dimensional space; consecutive hits from a track are **near each other** (in Euclidean distance d), otherwise, far away.
- Inference: Each hit connected to all other hits in a hypersphere centered on the hit with a radius **r**.
- Both methods create graphs containing >99% of all true edges.



GNN edge classification

Graphs contain many fake edges (see figure). Eliminate fakes while preserving true edges with a GNN

1. Encode nodes and edge features.
2. **Aggregate edge vectors, acting as messages between nodes.**
3. **Update node features with aggregated message. Update edge features using updated node features**.
4. Repeat ***n*** times steps **2** and 3.
5. Compute an edge score representing the probability of being a true edge.

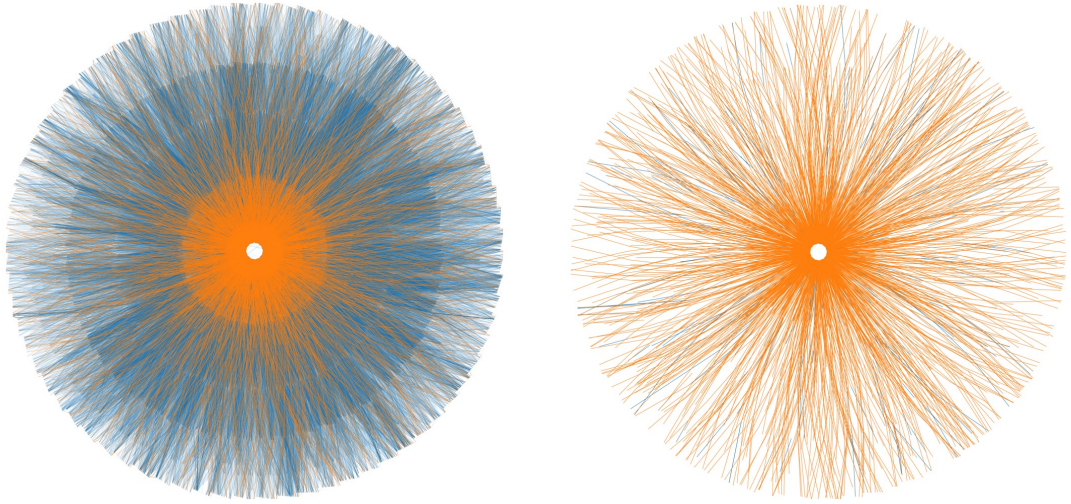


Battaglia, Peter, et al.

Interaction network

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

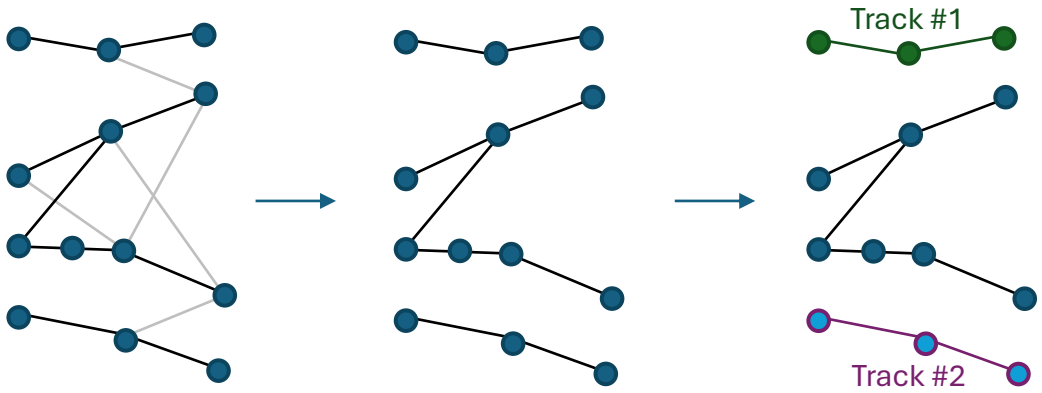
v_i^k node vector
e_{ij}^k edge vector
at iteration k



Input graph (left) and classified graph (right). **Fake** = **blue**. **True** = **orange**

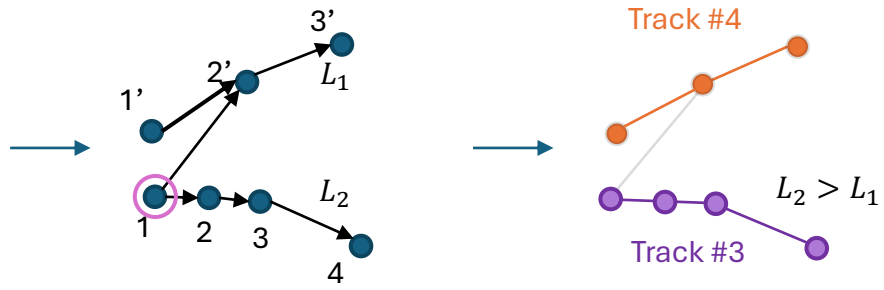
Graph segmentation

Connected Components



Classified edges | Loose score cut | Label simple candidates

Walkthrough



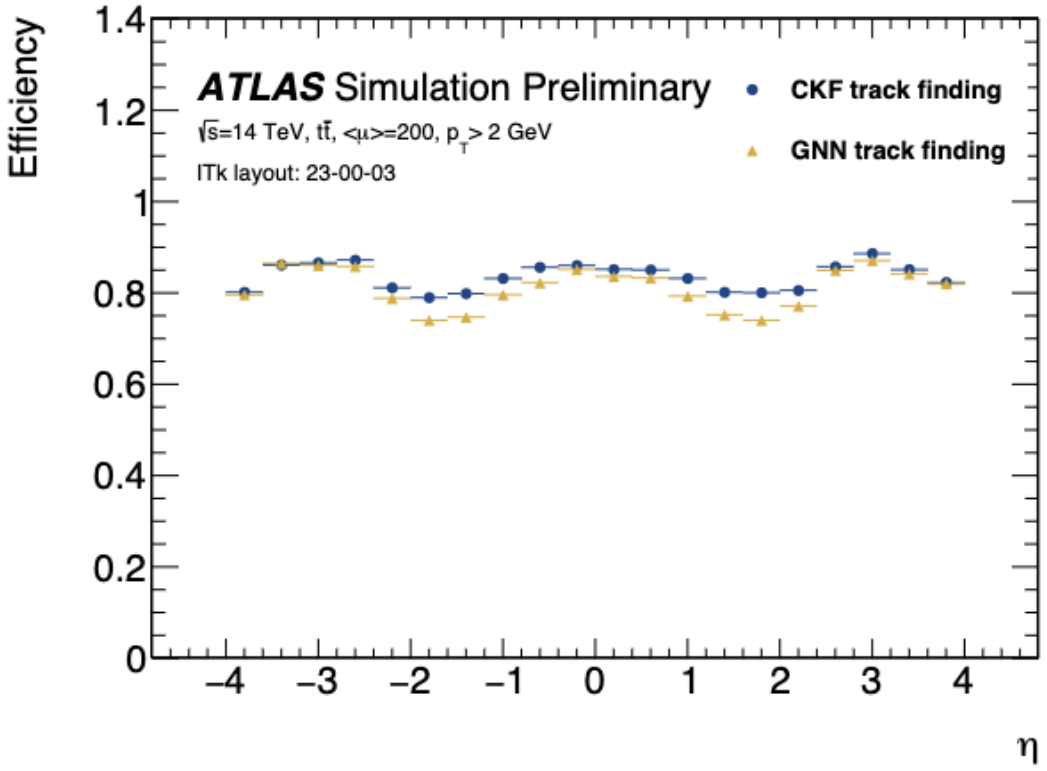
Walk through paths from starting node, count length L | Assign longest path as candidate

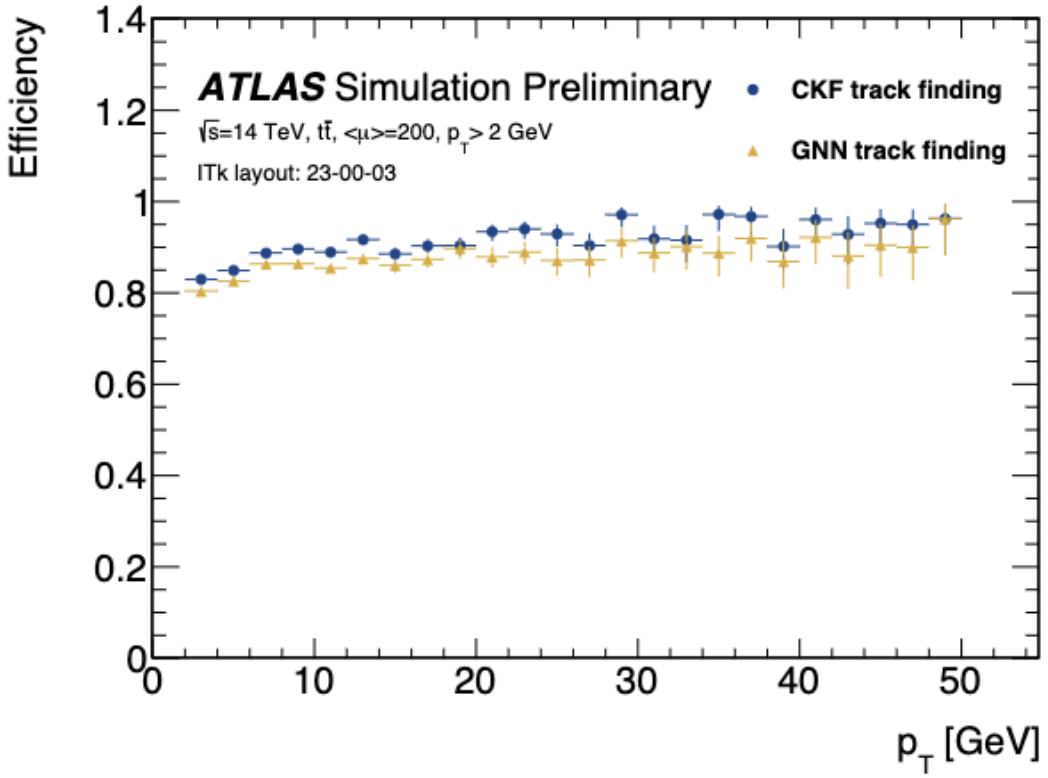
- 2-step sequence: Connected components and walkthrough:
  1. Use CC to isolate subgraphs with no branching.
  2. On subgraphs with branching, use walkthrough to separate track candidates (optional).
- Each track candidate is a list of hits => extract track parameters by a track fit and match to truth particles for physics performance evaluation

Tracking performance

Track reconstruction efficiency

ATL-SOFT-PROC-2023-047

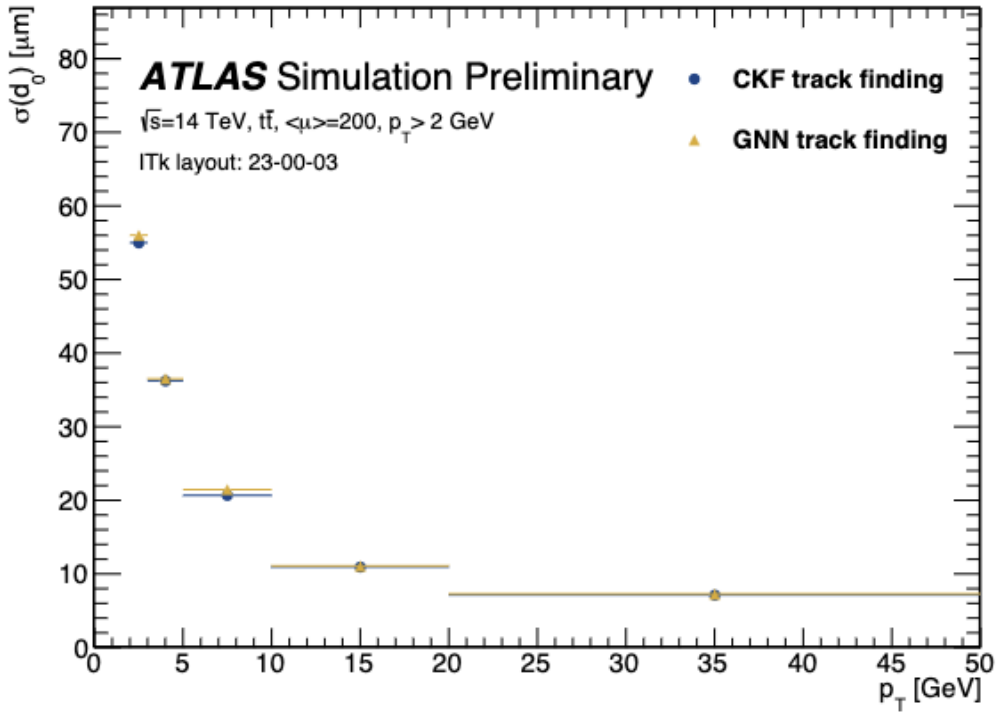


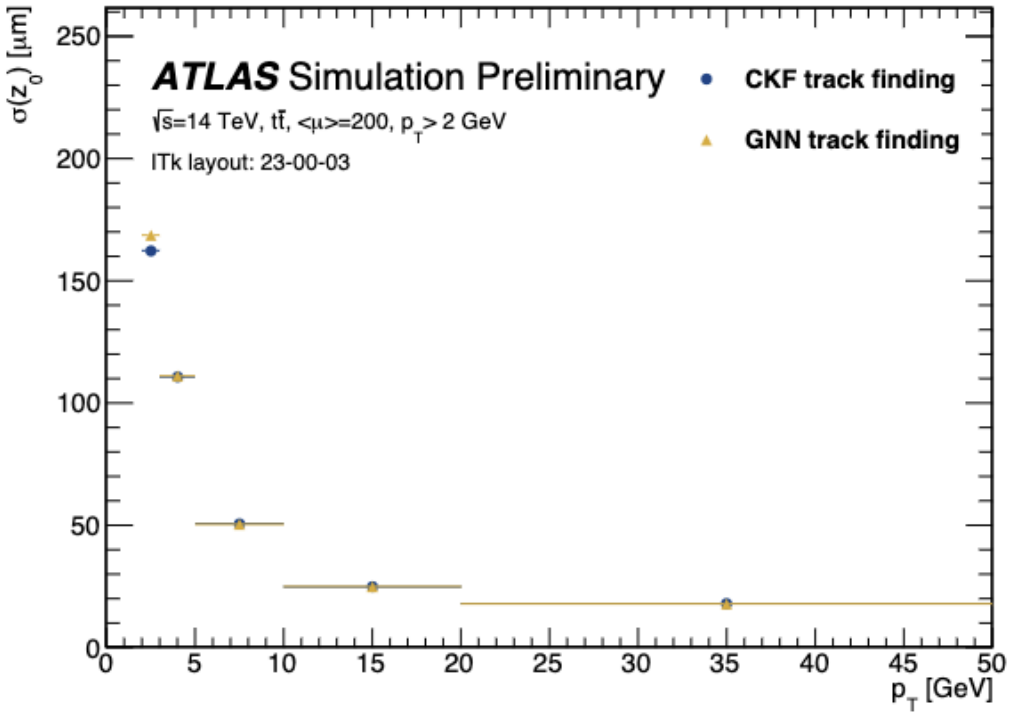


The GNN gives competitive performance compared to the Combinatorial Kalman Filter (CKF). Similar efficiency in the central and forward regions, worse around $|\eta| = 2$. The efficiency correlates with particle η, suggesting the transition region is particularly difficult for the GNN.

Impact parameter resolution

ATL-SOFT-PROC-2023-047





Parameter resolution reflects how well a track represents the particle characteristics. (Left) transverse (d0) and longitudinal (z0) impact parameter resolution vs pT. Overall good agreement.

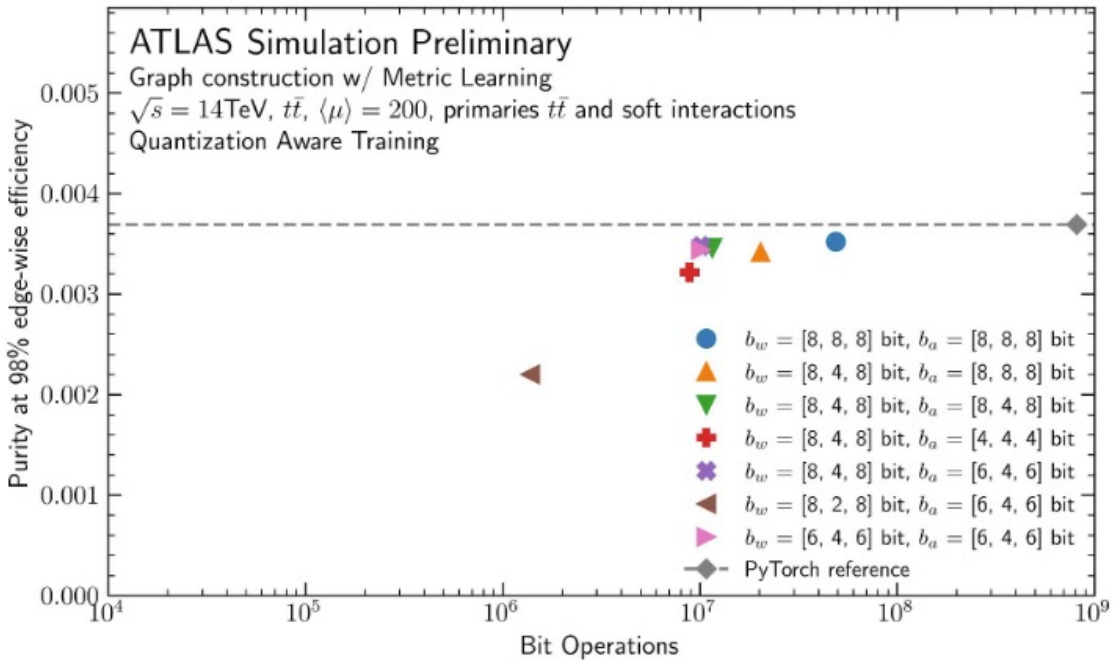
More performance plots in back-up

Extension to Event Filter

Optimization with quantization

- **Use case**: Event Filter on **FPGAs** → Compress models to arbitrary precision, e.g. 2-, 4-, 8-bit: quantization
- Post-training quantization works poorly → account for accuracy loss while training.
- With quantization-aware training, effect of precision reduction included in the loss function.
  - Forward pass in low and specified precision.
  - Loss computation and model update in full precision.
- Achieve **similar performance** as default 32-bit precision with up to O(100) fewer bit operations.

S. Dittmeier - ATL-DAQ-SLIDE-2024-027

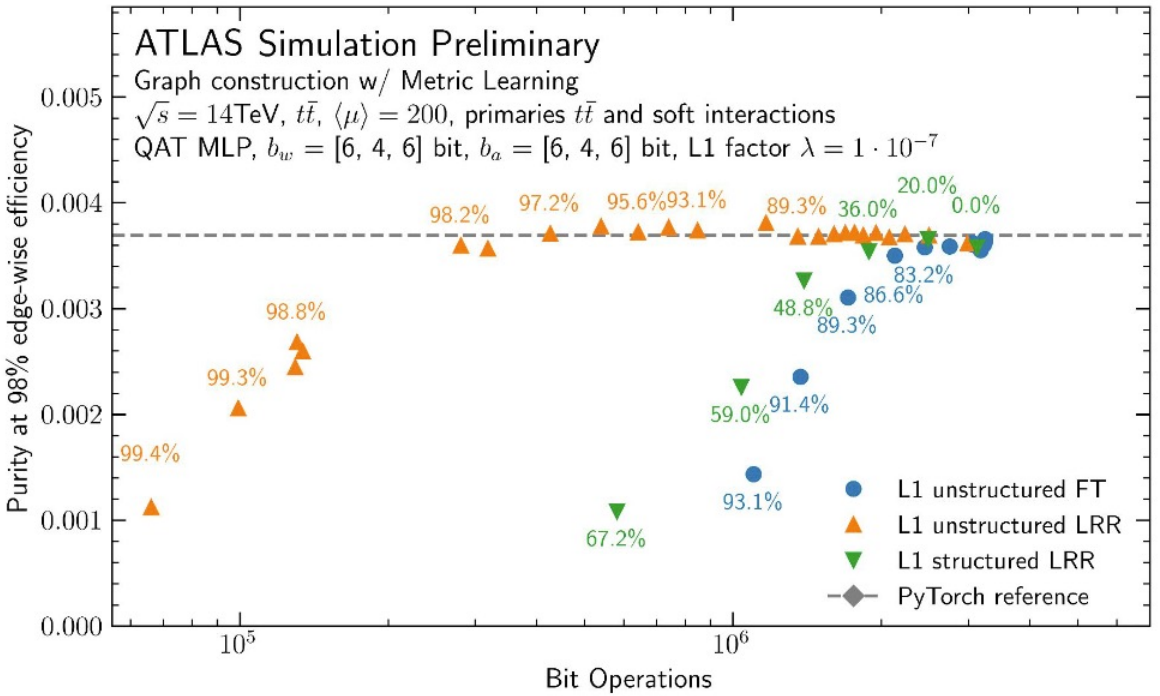


b_w: bit width on the first layer, hidden layers, and the last layer of weight matrix
b_a: bit width on the first, hidden, and the last activation function

Optimization with Iterative Pruning

- **The idea**: encourage model weights to take small numerical values, remove those under a threshold.
- Control model weights with L1 regularization. Prune iteratively:
  1. Train a network to a certain performance,
  2. Remove some neurons/channels of the network if the weight falls below a threshold,
  3. Fine-tune (FT) or retrain model by rewinding the learning rate (LRR).
- Can prune model to 1/56 (98% sparsity) the size and maintain the same level of performance with **unstructured** pruning and **LRR**.

S. Dittmeier - ATL-DAQ-SLIDE-2024-027



FT: fine-tuning
LRR: learning-rate rewind
Unstructured pruning: Set small weights to 0 in weight matrix.
Structured pruning: Remove a neuron of the network.

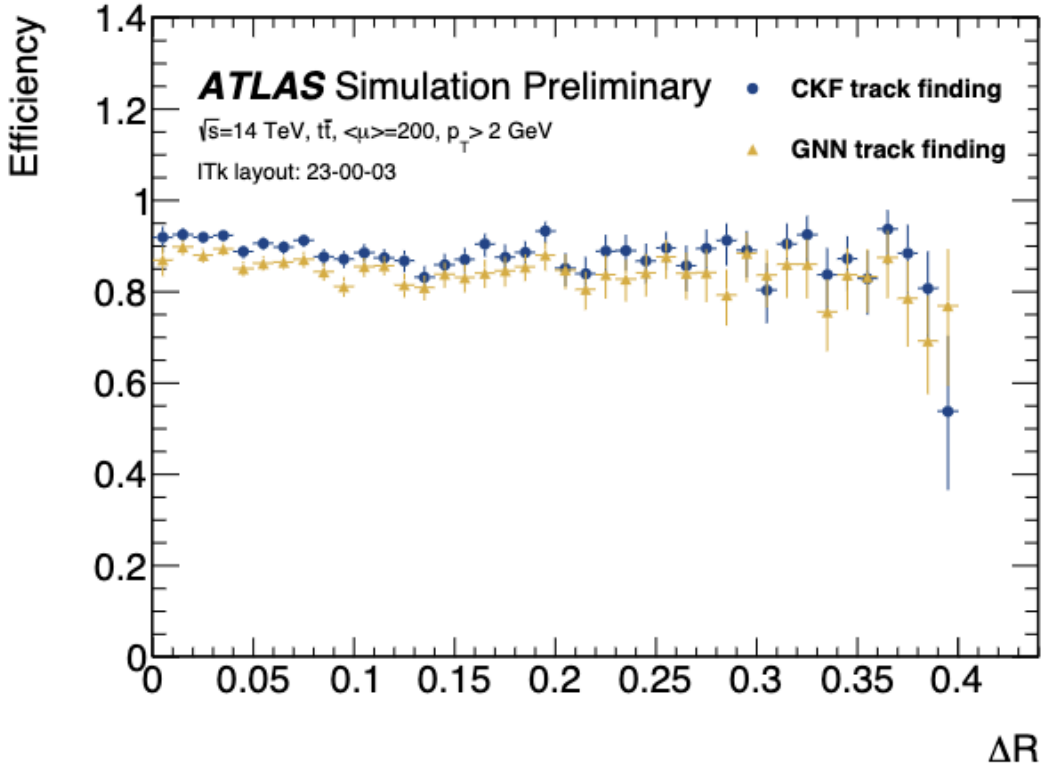
Summary and prospects

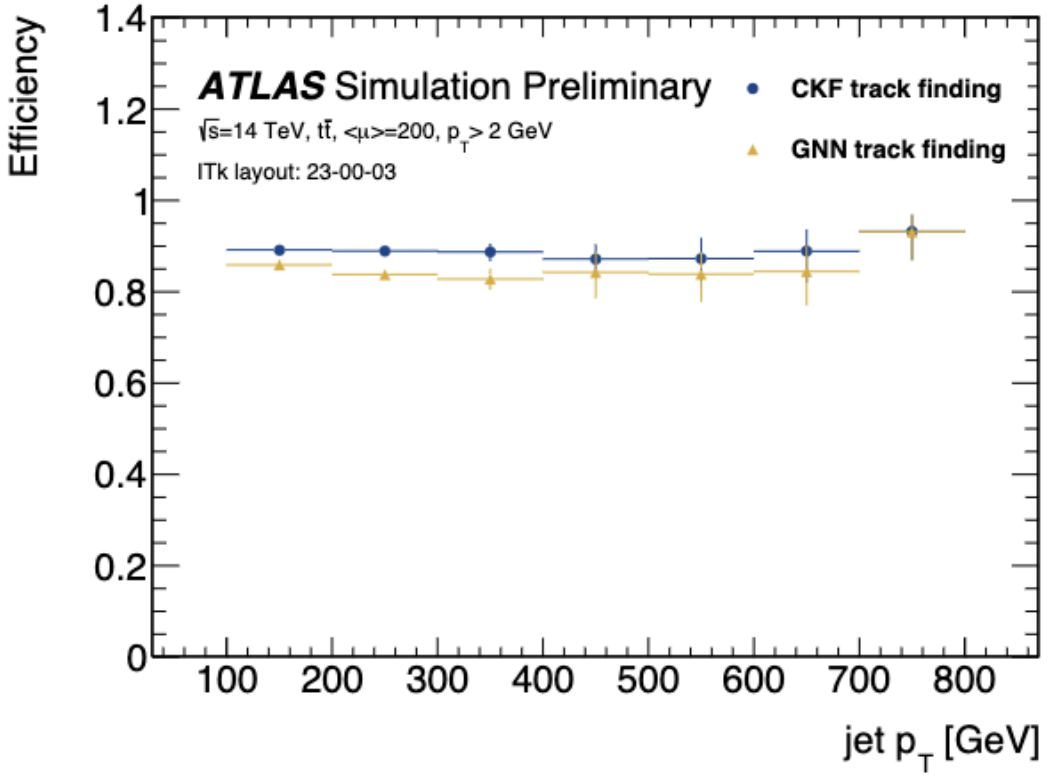
- Overview of a complete machine learning pipeline for track reconstruction.
- An apples-to-apples comparison of tracking performance to the default Combinatorial Kalman Filter showing good performance.
- Promising computational optimization with quantization and pruning for FPGAs and accelerated GPU inference.
- Next steps:
  - Full-chain inference in Athena.
  - Refine pipeline models to improve performance.
  - Optimize the throughput of the entire pipeline by quantization, pruning, and compilation.
  - Study robustness against misalignment and dead modules.
  - Generalization to other processes: single particles, Z’⟶jets, long-lived particles, etc.

Back-ups

Efficiency in dense environment

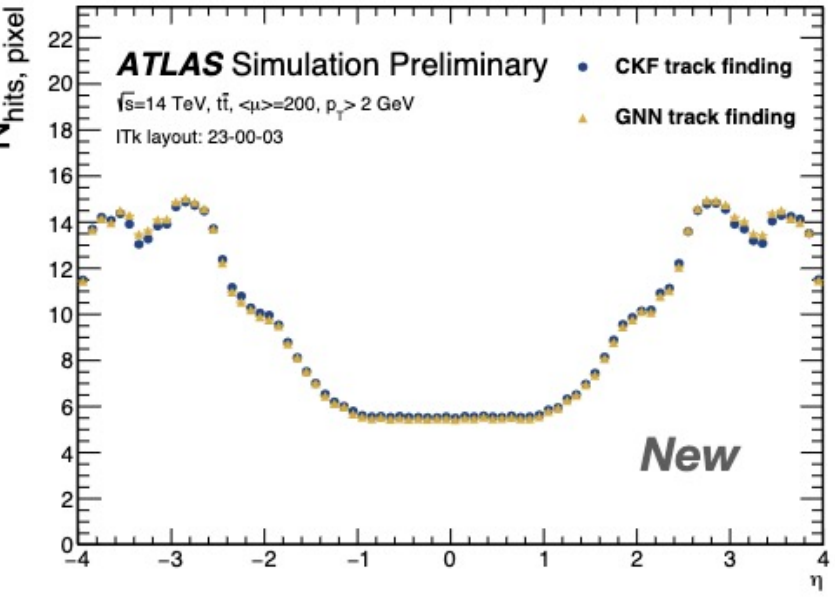
ATL-SOFT-PROC-2023-047

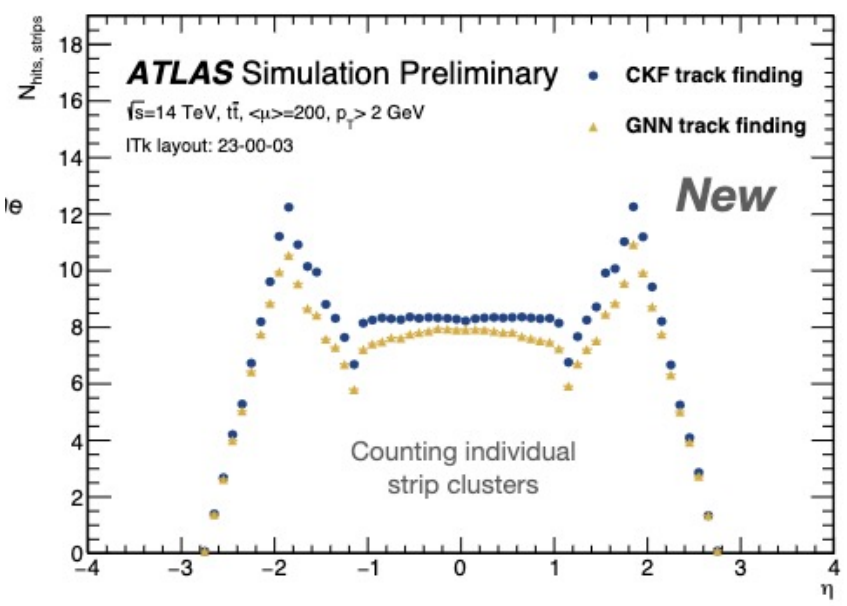


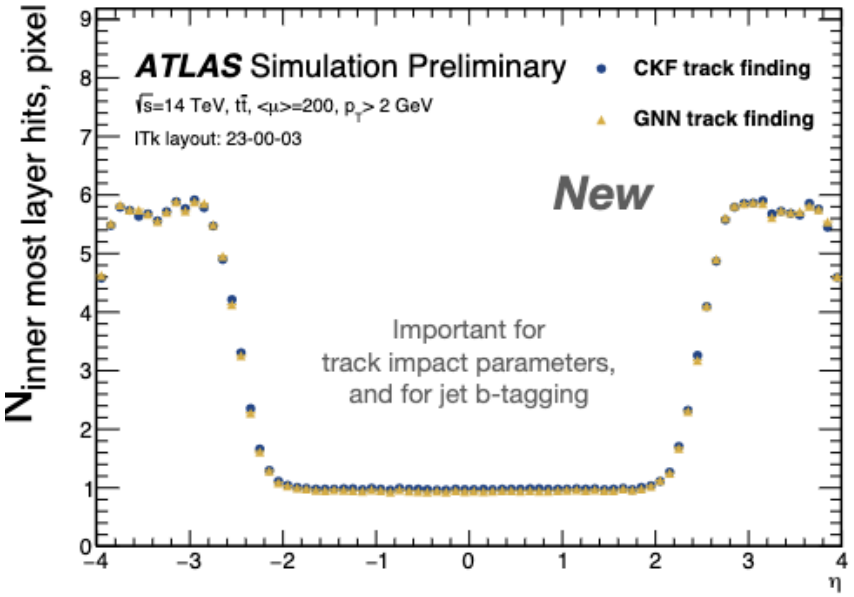


Tracking efficiency inside jets as function of (left) the angular separation of the track from jet axis (ΔR) and (right) jet p_T. The GNN performance is very close to the CKF and remains constant wrt both p_T and ΔR. No degradation is observed with increasing track density (towards the jet core and with higher jet p_T).

Hit content comparison



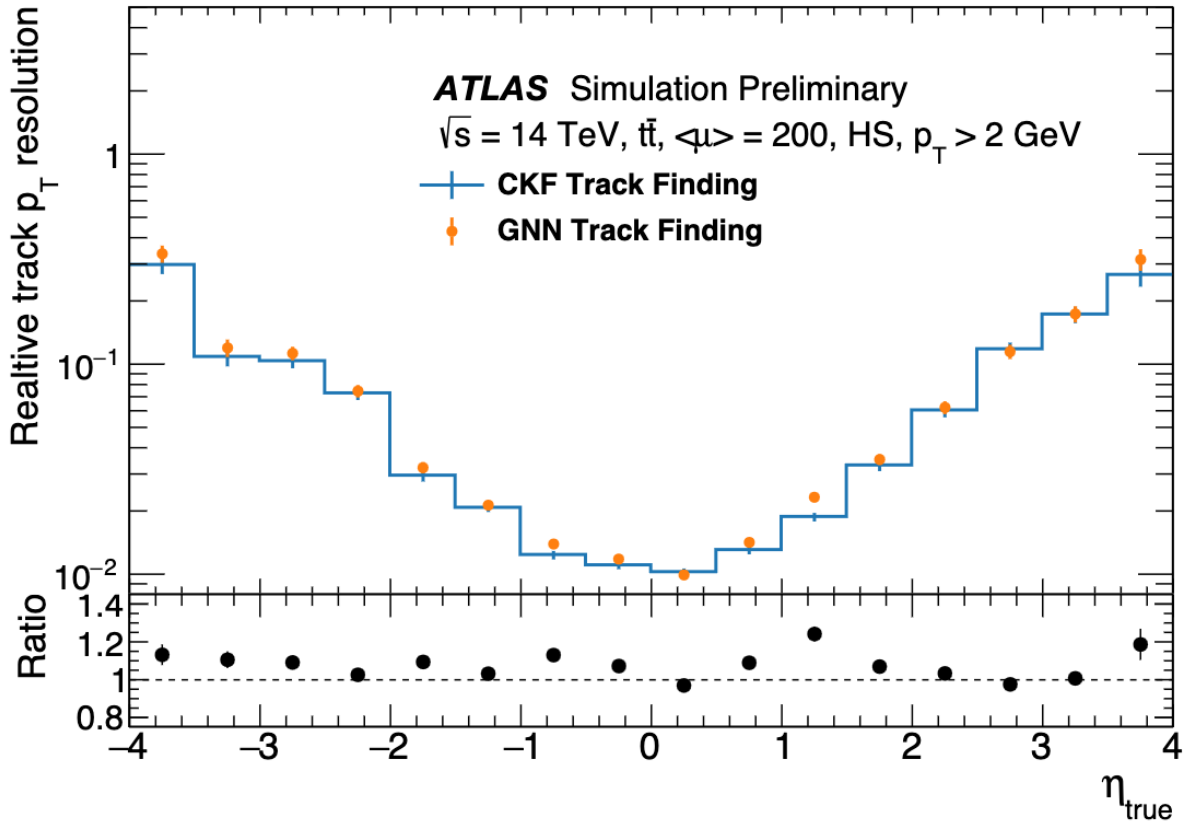




The number of (left) pixel hits, (center) SCT hits, and (right) innermost pixel hits as a function of track η. The number of (innermost) pixel hits shows very good agreement between CKF and GNN tracks. Innermost pixel hits are important in constraining impact parameters (d_0, z_0), expect good impact parameter resolution. The difference in SCT is well-understood: Because the GNN builds tracks from space points, it ignores single-cluster hits in the SCT by design.

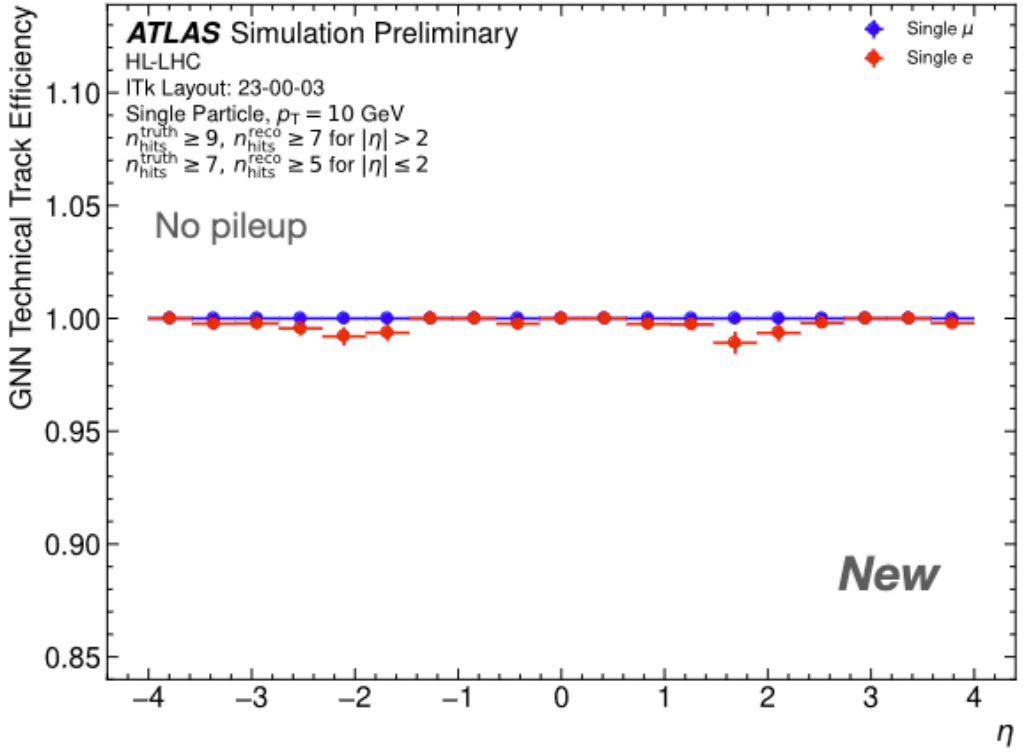
Track pT resolution

Hits in the SCT are effective in constraining track pT.
GNN tracks have lower SCT hit counts than CKF tracks, hence lower pT resolution.



Generalization to single particle samples

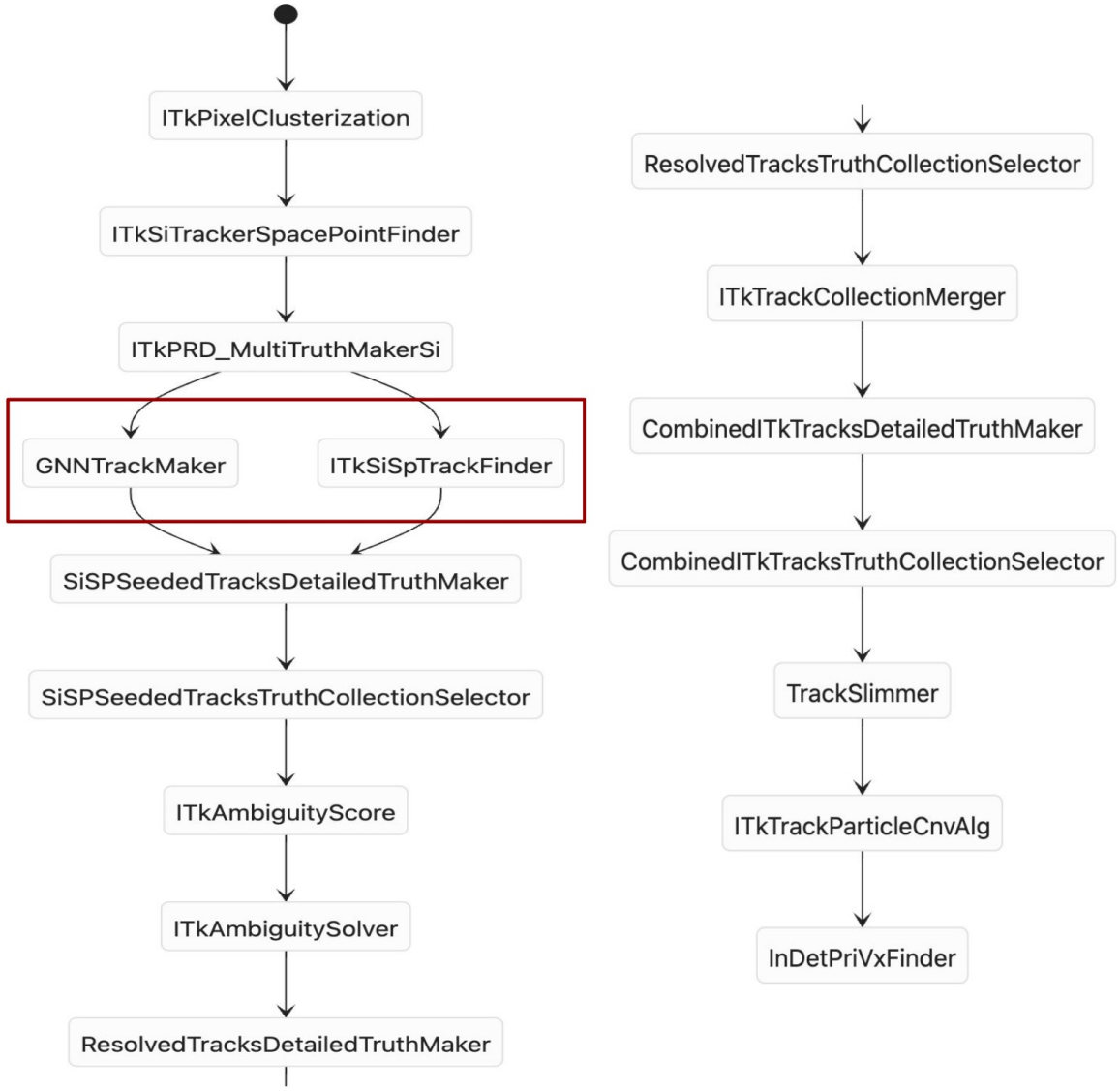
Tracking efficiency of muon and electron at $p_T = 10\ GeV$.
Muons do not significantly undergo Bremsstrahlung and hadronic interaction, should have close to 100% efficiency, which is observed. Electrons are significantly affected by Bremsstrahlung and multiple scattering, but still attain good efficiency.
Note: GNN models are not train on electron tracks, but still able to reconstruct them at satisfactory efficiency levels.



Integration into ATLAS analysis software

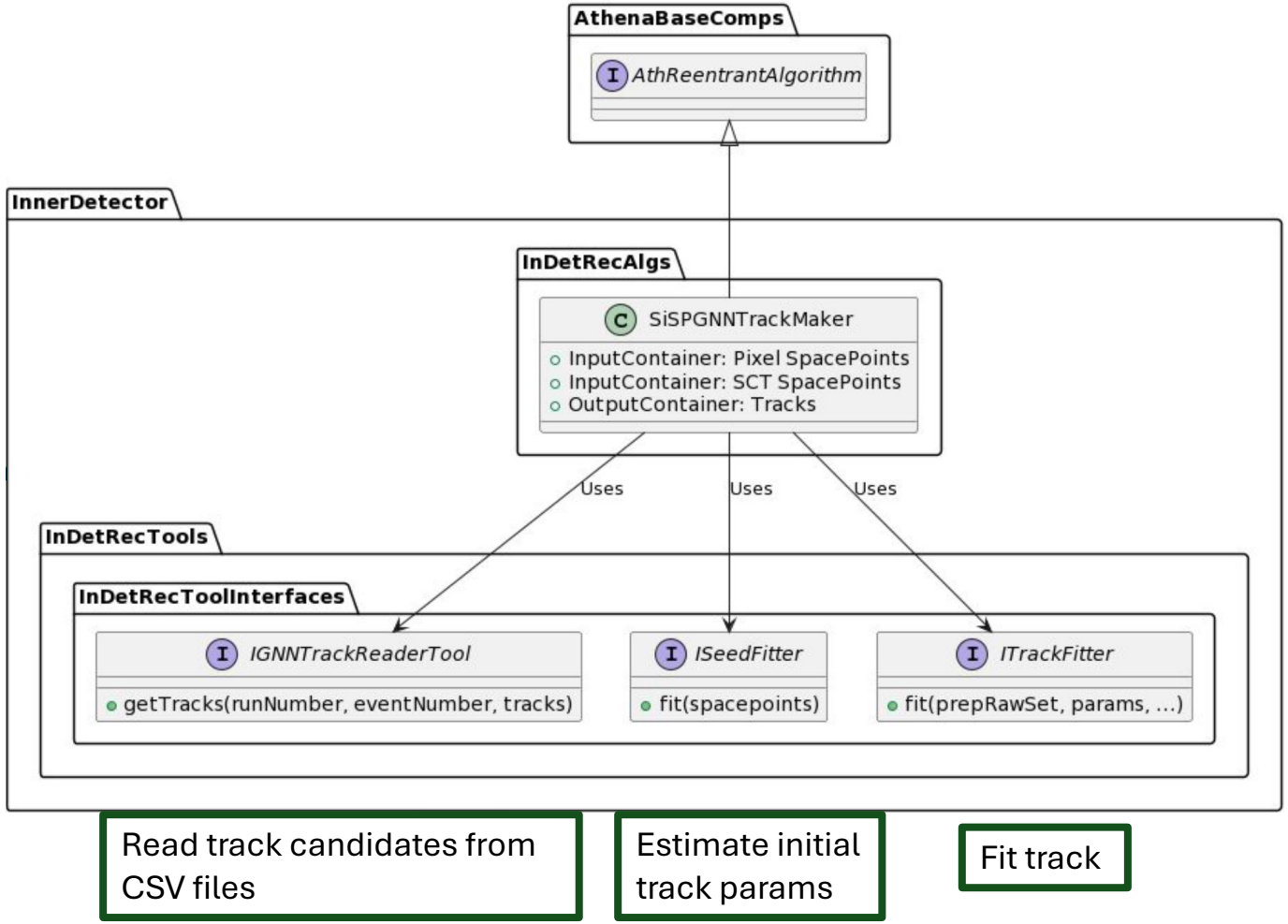
GNNTrackMaker

- Currently, Athena implements the CKF in the **ITkSiSPTrackFinder** class.
- Integrate the GNN chain via a **GNNTrackMaker** class.
- **Goal**: Input all space points from an event,
  1. Build track candidates
  2. Process track candidates.
- Implemented in Athena Rel24, code here.

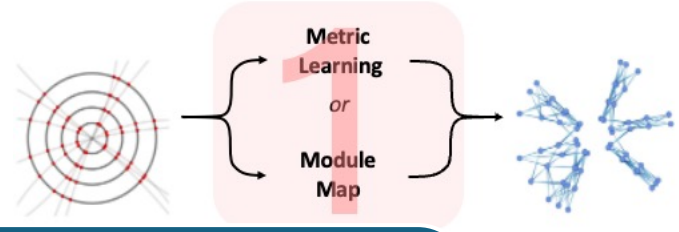


The GNNTrackMaker

- IGNNTrackReaderTool only used to test **ITrackFitter**.
- Test track processing with CKF track candidates
  1. Find tracks with CKF and save to CSV
  2. Read tracks from CSV and process, compare to CKF performance
- Next step: GNNTrackFinder. Convert trained models to ONNX runtime. (WIP)
  1. Build graph from space points
  2. Classify edges
  3. Build tracks by segmenting graphs with fake edges removed



Data driven approach: Module map

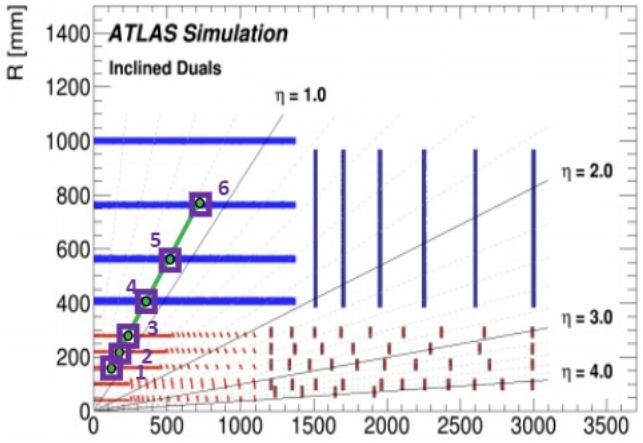


The idea: Build a list of detector module triplets from truth information: a connection A⟶B⟶C is added if a particle passes sequentially through them.

Construction: From 100k events, build all combinations of sequential triplets.

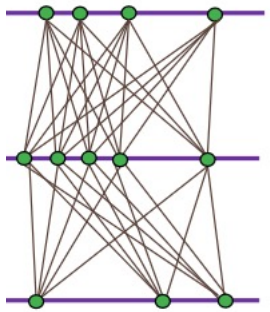
Register/update geometric cuts.

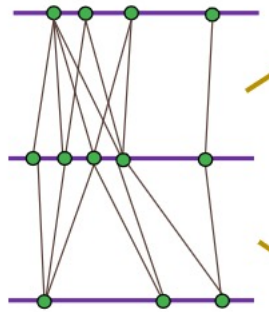
Inference: Connect a hit triplet **a⟶b⟶c** if **abc** contained in MM. Apply these geometric cuts to reduce possible fakes.



Connections added:

1 -> 2 -> 3
2 -> 3 -> 4
3 -> 4 -> 5
4 -> 5 -> 6





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

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

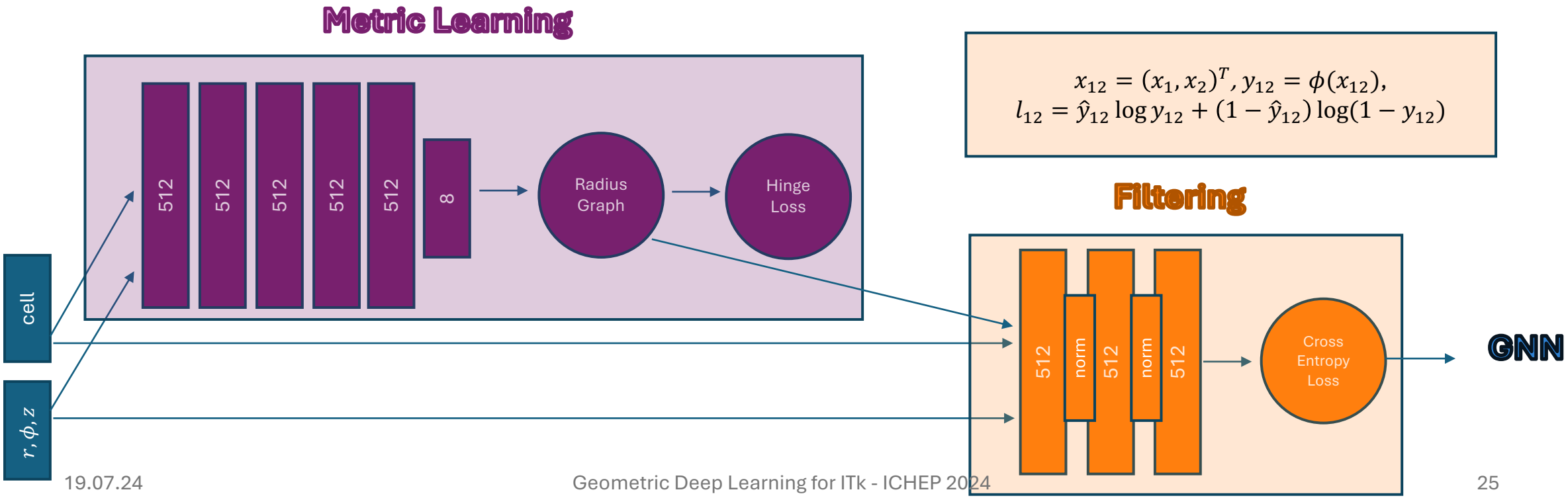
- $\Delta\frac{\Delta y}{\Delta x} = \frac{\Delta y_{12}}{\Delta x_{12}} - \frac{\Delta y_{23}}{\Delta x_{23}}$
- $\Delta\frac{\Delta z}{\Delta r} = \frac{\Delta z_{12}}{\Delta r_{12}} - \frac{\Delta z_{23}}{\Delta r_{23}}$

Resulting graphs have $\langle N_e \rangle = \mathbf{1.9{\times}10^6}$, contains **99.5%** all true edges.

Machine learning approach: Metric learning

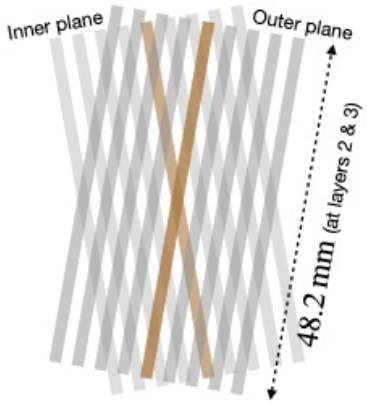
- ML graphs are too large for later steps => Train a simple NN to classify edges from node features.
- Result in graph of $\langle N_e \rangle = \mathbf{1.0{\times}10^6}$, contains **99.3%** true edges.

	$\langle N_e \rangle$	True edges
Module map	$1.9{\times}10^6$	**99.5%**
Metric learning	$\mathbf{1.0{\times}10^6}$	99.3%



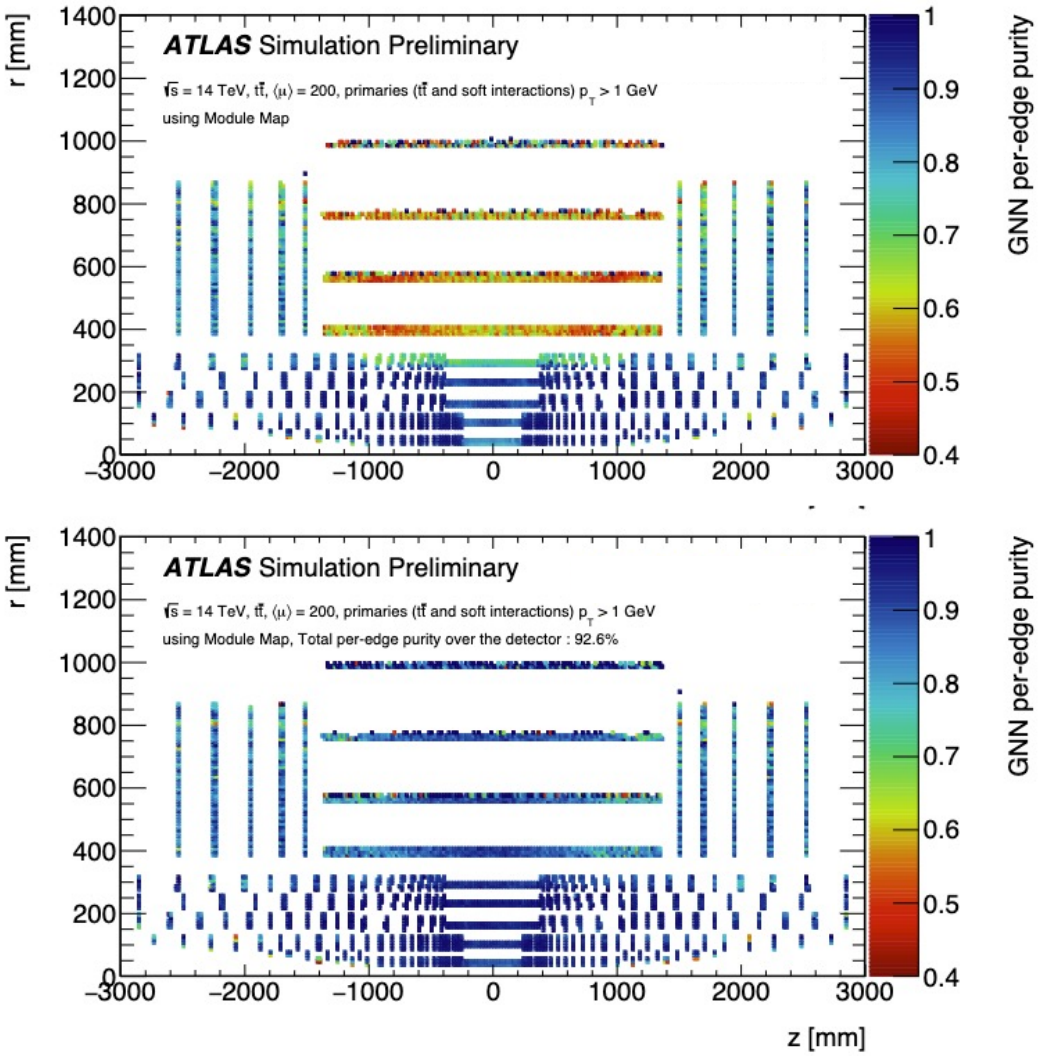
Heterogeneous data

- Strip space points have low resolution compared to pixel.
- Address by passing individual cluster information to the GNN
  - Strip: $(\vec{r}_{sp}, \vec{r}_{cl1}, \vec{x}_{cl1}, \vec{r}_{cl2}, \vec{x}_{cl2})$, x: cluster info
  - Pixel: $(\vec{r}_{sp}, \vec{r}_{cl}, \vec{x}_{cl}, \vec{r}_{cl}, \vec{x}_{cl})$
- Significant improvement in purity in the strip barrel at the same efficiency.

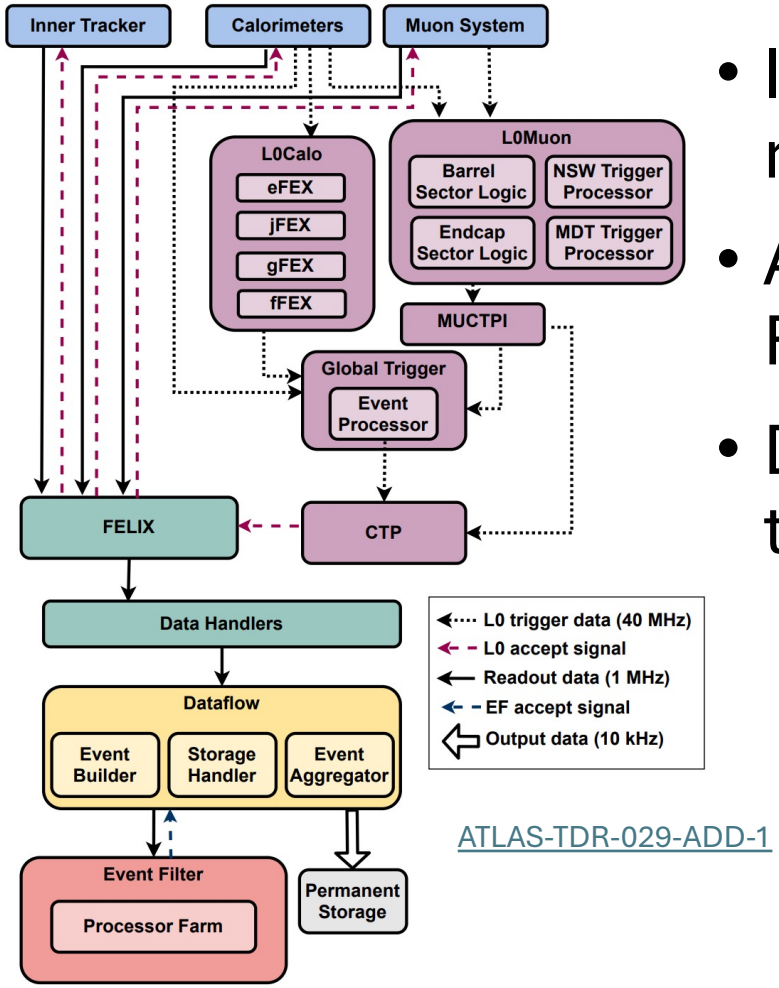


Double strip sensor planes in barrel module
Two strips fired by a particle in brown
Where did the particle hit the inner plane?

Default hits:
Poor σ_z ~ 1-3 cm
(was limiting GNN performance)



Event filter in HL-LHC



ATLAS-TDR-029-ADD-1

- In HL-LHC, Event Filter receives 10 times the L0 readout rate compared to Run 3 condition.
- ATLAS has demonstrators for CPU-, GPU-, and FPGA-based EF tracking.
- Demonstrate the viability of GNN-based tracking on FPGA.

	LHC Run 3	**HL-LHC**
L0 trigger accept	100 kHz	**1 MHz**
Event Filter accept	1 kHz	**10 kHz**
Event size	1.5 MB	**4.6 MB**