Track Reconstruction for the ATLAS Detector's Phase II Trigger and Data Acquisition System using Graph Neural Networks

Sudha Ahuja on behalf of the ATLAS Collaboration, CERN

AI @ Royal Holloway

ROYAL HOLLOWAY UNIVERSITY OF LONDON

High Luminosity Large Hadron Collider (HL-LHC)

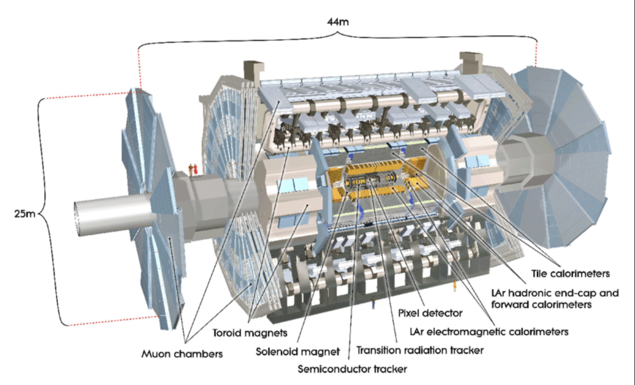
HL-LHC will deliver an order-of-magnitude increase in integrated luminosity and amount of data produced per event.

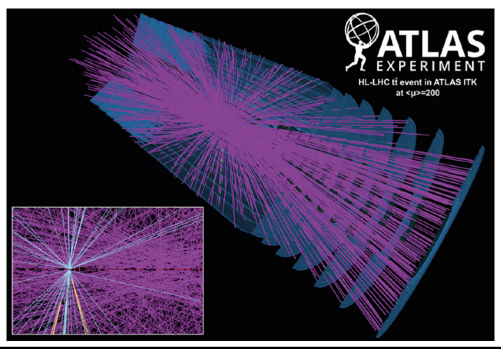
Challenging environment with high background pileup noise & particle production rate.

Major upgrades are foreseen for the ATLAS detector and the Trigger and Data Acquisition System (TDAQ).

The TDAQ system consists of a Level-0 hardware trigger, a data acquisition and an Event Filter system.

The Event Filter system is exploring a heterogeneous computing farm comprising CPUs, and potentially GPUs and/or FPGAs.


44m
25m
Tile calorimeters
LAr hadronic end-cap and forward calorimeters
Pixel detector
LAr electromagnetic calorimeters
Toroid magnets
Solenoid magnet
Transition radiation tracker
Muon chambers
Semiconductor tracker



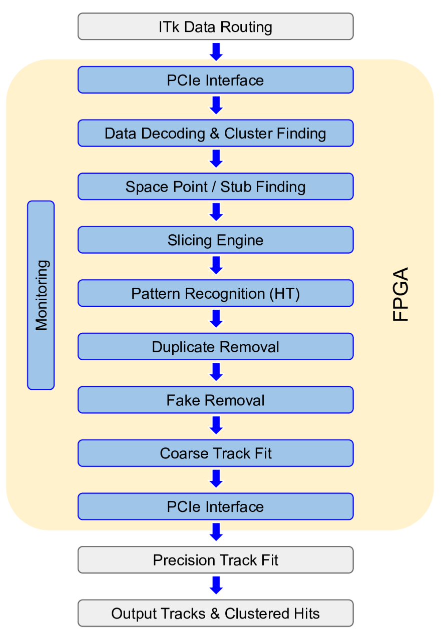
Event Filter Tracking

Track reconstruction is the most computationally intensive process, due to combinatorics, within the ***Event Filter*** system.

Tracking predominantly influences the Event Filter system requirements.

Exploring novel track reconstruction algorithms and modern machine learning techniques with heterogeneous computing architecture (CPUs + GPUs +FPGAs)

- ***Pattern Recognition*** (from the pipeline) being studied as a geometric deep learning problem. Achieved by representing Inner Tracker data as a graph and training the model to classify its connections.



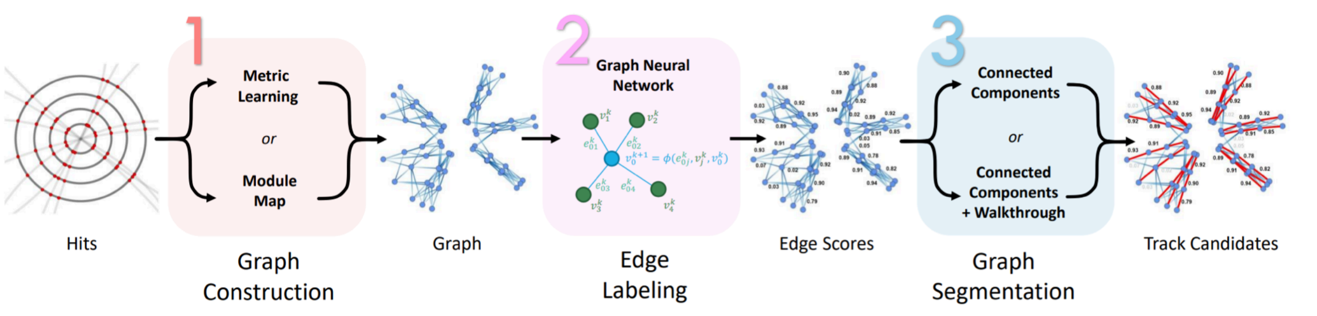
Event Filter Tracking Pipeline

Track Reconstruction using Graph Neural Networks (GNN) & Implementation on FPGA's

Particle Track reconstruction involves precisely measuring the trajectory and the properties of charged particles (reconstruction) in the Tracking detector by identifying which of the isolated measurements (hits) belong to the same particle. ***A GNN approach to charge particle tracking has been proposed [1][2]***, while studying Deep Neural Networks for Track reconstruction. In this approach, each graph node represents a sparse measurement (hit), with edges connecting pairs of hits based on geometrically plausible relationships.

GNN based pipeline:

1. Graph reconstruction - aims to maximize efficiency of true edges while limiting the amount of false edges. Can be done in two ways:
   - Metric Learning: Construct a graph using available information (r, phi, z), then use a Multi-Layer Perceptron (MLP) to map hits into a latent space. Create edges between hits lying within a circle of radius r. Network learns to push same-track hits within this circle. Trade-off between efficiency and purity by adjusting the radius threshold.
   - Module Map: Uses geometric observables for construction.
2. Edge Labeling: Implements interaction network to model the relationship between graph objects. GNN assigns an edge classification score to each edge to recognise edges originating from true track particles.
3. Graph Segmentation: Threshold applied to the edge score to remove false edges from the graph. Track candidates are then formed by segmenting the graph into connected components, using a walkthrough algorithm to resolve multiple paths for specific nodes. These track candidates inputted into track-fitting algorithms.



Performance Metrics for Graph Construction (step 1)

$$\text{Efficiency} = \frac{\text{True edges in the graph}}{\text{Total true edges}}$$

$$\text{purity} = \frac{\text{True edges in the graph}}{\text{Total edges in the graph}}$$

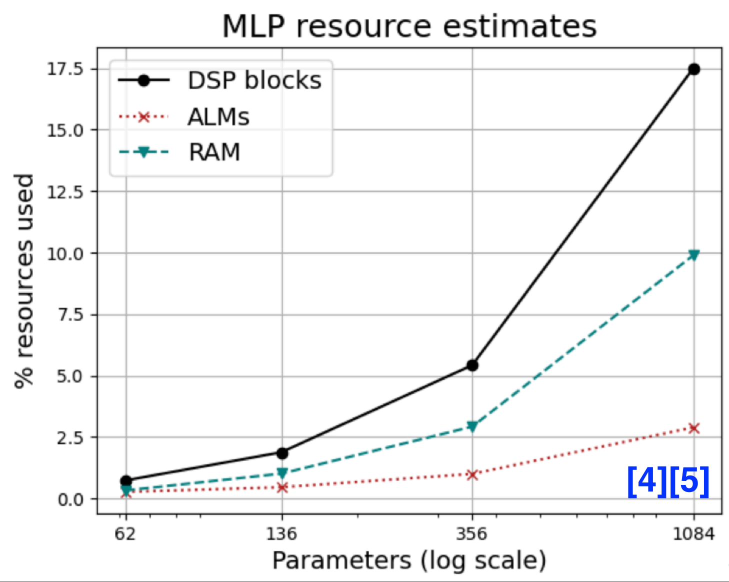
Resource Optimisation for the Graph Reconstruction Step

Resource Estimation on FPGAs

GNN Tracking on FPGAs [3] can simultaneously provide parallelization with low energy consumption (compared to CPUs and GPUs).

For FPGA deployment, converted the algorithm to Hardware Descriptive language (HDL). Used HLS4ML for high-level synthesis of the ML component.

- **Using ITk Dataset [5]**: Trained the algorithm with PyTorch and converted it to ONNX. Used HLS4ML to generate HDL code and Quartus RTL compilation to estimate resources on the Intel S10 GX device.



Resource Estimates for MLP used in the Graph Construction step (No Pruning)

Results show that the Digital Signal Processing blocks are predominantly used

Model Compression

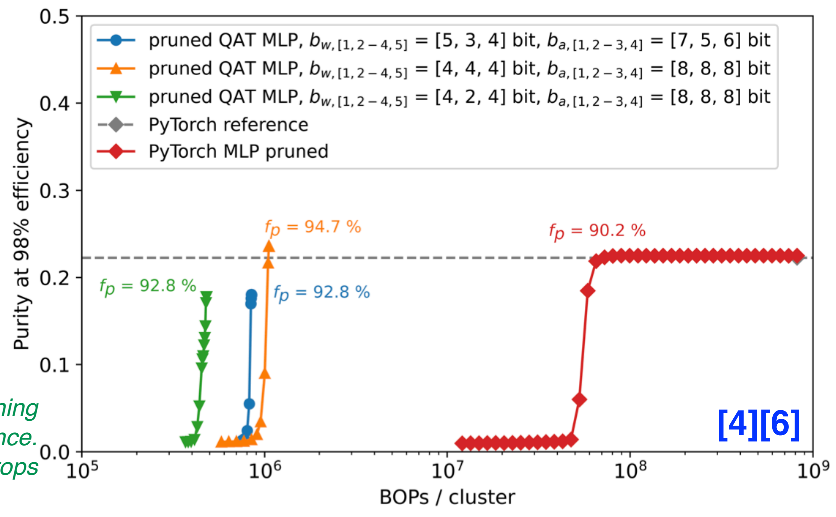
The Tracker is expected to generate hundreds of thousands of parameters, posing a challenge to reduce resource usage while maintaining efficiency and purity. Various ML model compression techniques have been studied:

1. Quantization Aware Training (QAT) - Trained parameter representation using fixed-point instead of floating-point. Defined bit-widths for weights and activation before training. Implemented QAT with BREVITAS [7] integrated into PyTorch.
2. Pruning - Iteratively removed non-contributing weights from the network, leveraging sparse matrices to speed up computation.

Results (Trained & tested with the TrackML Dataset [6])

Applied QAT and model pruning to the MLP during the graph construction stage, using heterogeneous bit widths for weights and activations. Integrated Batchnorm for FPGA-specific requirements. Pruned 10% of weights every 180 epochs.

Model performance measured by computing purity at fixed (98%) efficiency vs model size given in Bit Operations (BOPs) per hit cluster. Three QAT models are chosen and pruned iteratively.



The three QAT models shown here achieved pruning factors f_p above 92% without drops in performance. While the pruned PyTorch reference model drops only at pruning factors larger than 90% .

Summary and Outlook

Application of Graph Neural Networks on FPGAs is under investigation for the Event Filter Tracking system for the HL-LHC. Model resource optimization studies are ongoing, as the pipeline is primarily occupied by the Machine Learning components. Initial model compression using QAT and pruning showed promising resource reduction while maintaining performance. Further studies are ongoing in various aspects (using the realistic ITk simulation samples, comparing MLP and Module Map techniques, segmenting the Graph into detector regions, etc.) and the final goal is to have a fully running GNN-based track reconstruction pipeline on an FPGA.

References: Graph Neural Networks for Particle Reconstruction in High Energy Physics detectors (2003.11603(2020)) [1]
Novel deep learning methods for track reconstruction, arXiv 1810.06111 2018 [2]
Graph Neural Networks for Charged Particle Tracking on FPGAs. doi: 10.3389/fdata.2022.828666 [3]
Track reconstruction for the ATLAS Phase-II Event Filter using GNNs on FPGAs (ATL-DAQ-PROC-2023-006/ ATL-DAQ-PROC-2023-014) [4]
ITk (Inner Tracker) dataset - https://gitlab.cern.ch/gnn4itkteam/acorn [5],Track ML DataSet - https://www.kaggle.com/c/trackml-particle-identification [6]
https://github.com/Xilinx/brevitas [7]