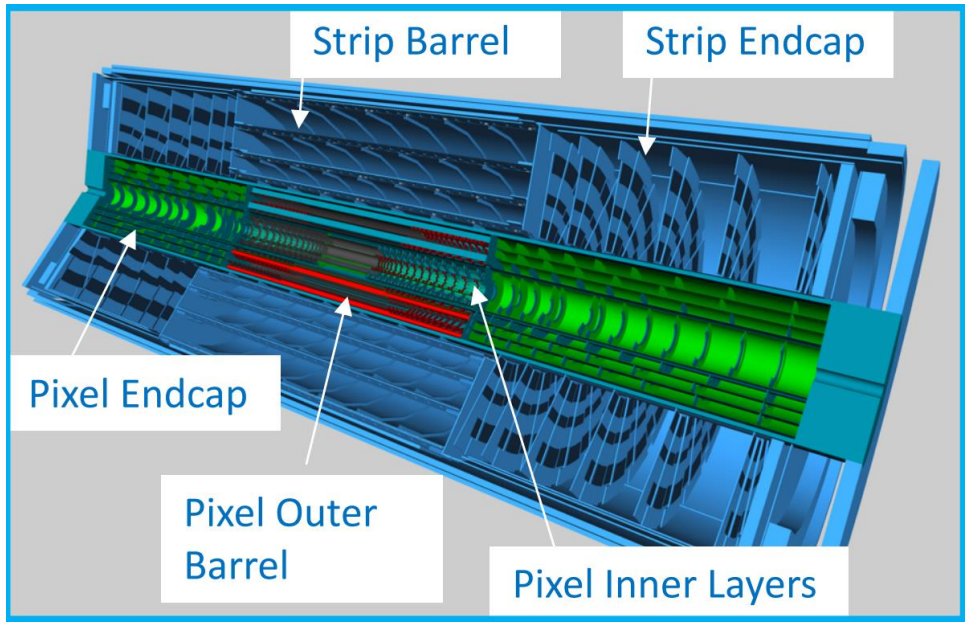
Track reconstruction for the ATLAS Phase-II High-Level Trigger using Graph Neural Networks on FPGA

On behalf of the ATLAS TDAQ collaboration
Mini-workshop on Real time tracking
CTD, October 13, 2023, Toulouse, France

Sachin Gupta (Physikalisches Institut, Heidelberg University)

ATLAS upgrade for HL-LHC

- From 2029, LHC Luminosity will be 7.5 times the current one.
- Average number of inelastic p-p collision per bunch will be increased to 200 (Currently 40).
- Shifting towards a new all silicon Inner Tracker(ITk)
- Upgrade in the trigger and data acquisition system (TDAQ) is also required



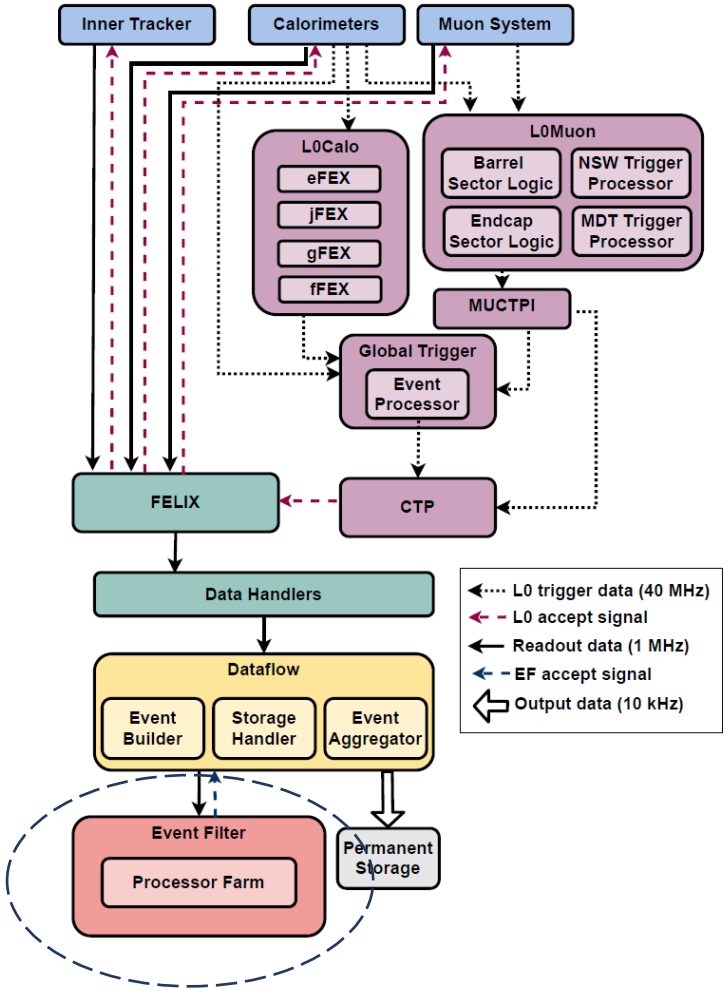
Ref : ATL-ITK-SLIDE-2018-073

ATLAS Event Filter for HL-LHC

- Track reconstruction for Inner Tracker
- Computationally intensive task for high particle density

- For real time tracking :
  - Precise algorithm
  - Fast computing resources
  - Accept trade off in precision for high throughput

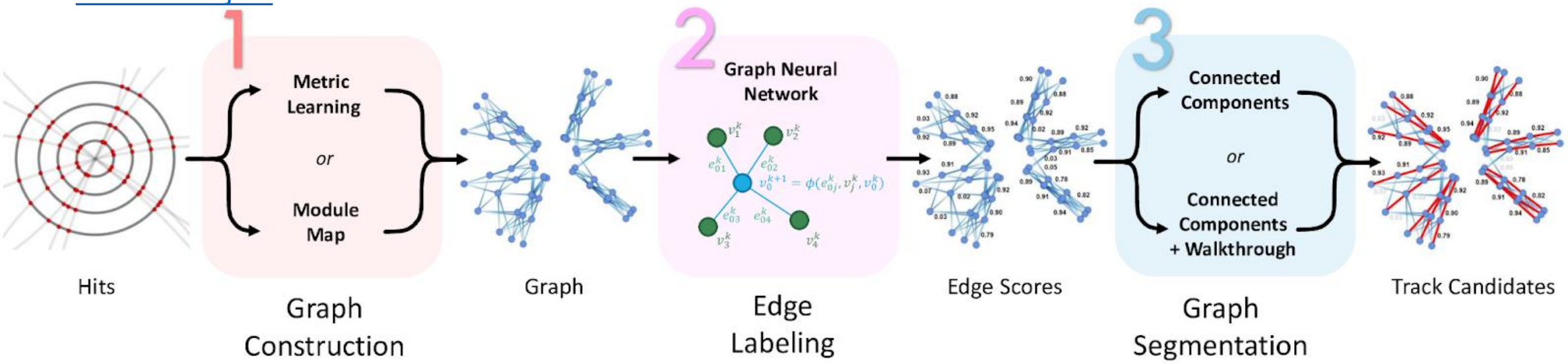
- Exploration of modern machine learning techniques with heterogeneous computing architecture (CPUs + GPUs +FPGAs)



Ref : ATLAS-TDR-029-ADD-1

Track reconstruction with GNN

[ExaTrkX Project]



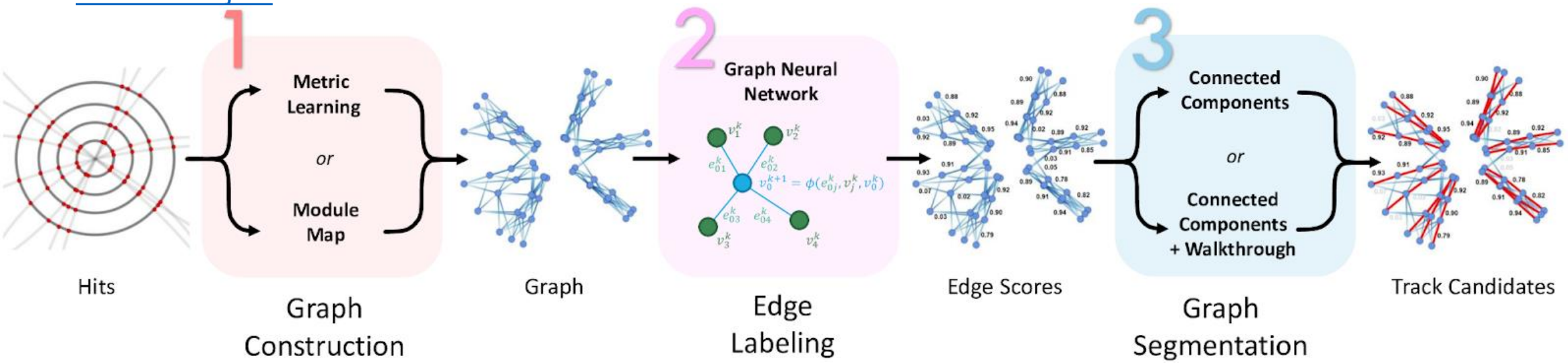
- **Metric Learning** : uses deep neural network
- **Module Map** : uses geometric observables for construction

- Implements interaction network to model the relationship between graph objects

- Creates track candidates as a sequence of hits

Track reconstruction with GNN

ExaTrkX Project



ML Based

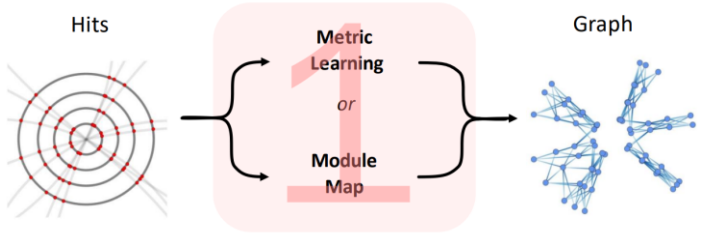
- **Metric Learning** : uses deep neural network
- **Module Map** : uses geometric observables for construction

- Implements interaction network to model the relationship between graph objects

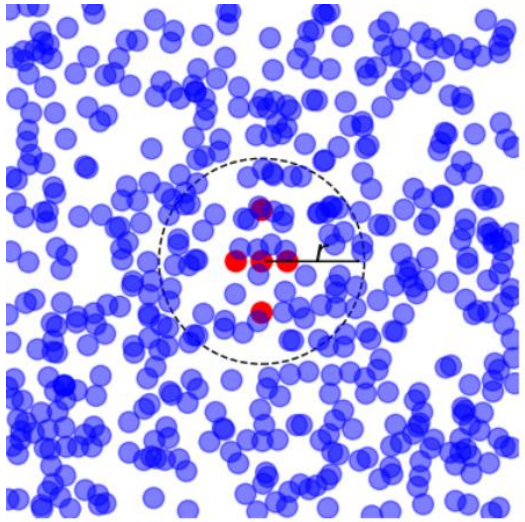
- Creates track candidates as a sequence of hits

Graph construction using MLP

- MLP is trained to construct graph with the available info : hit position (r, ϕ, z)



- Graph : Nodes(hits) , Edge (Connection between two hits)
- All hits are mapped to some latent space and Edge is drawn among the hits lying within the circle of radius r
- Network learns to push hits belonging to the same track within the circle : Red hits in the figure

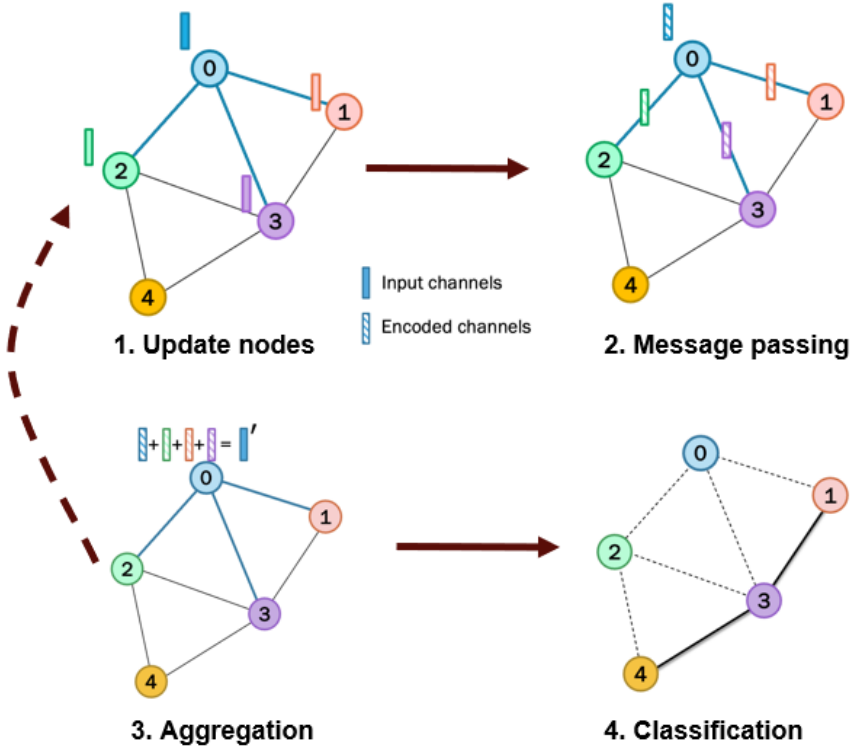


Hits in the latent space

Edge classification (GNN)

- Goal: classify edges as “True” or “False”
  - could they belong to a track?

- GNNs rely on Message Passing:
  - Node vectors (properties of node) are updated through an update function (MLP)
  - Information is passed along edges to neighboring nodes
  - All messages are aggregated at each node
  - After one or more message passing steps, use classifier to make edge-level predictions



Resource Estimation for Intel FPGAs* with HLS4ML

*** AMD FPGAs : Under exploration**

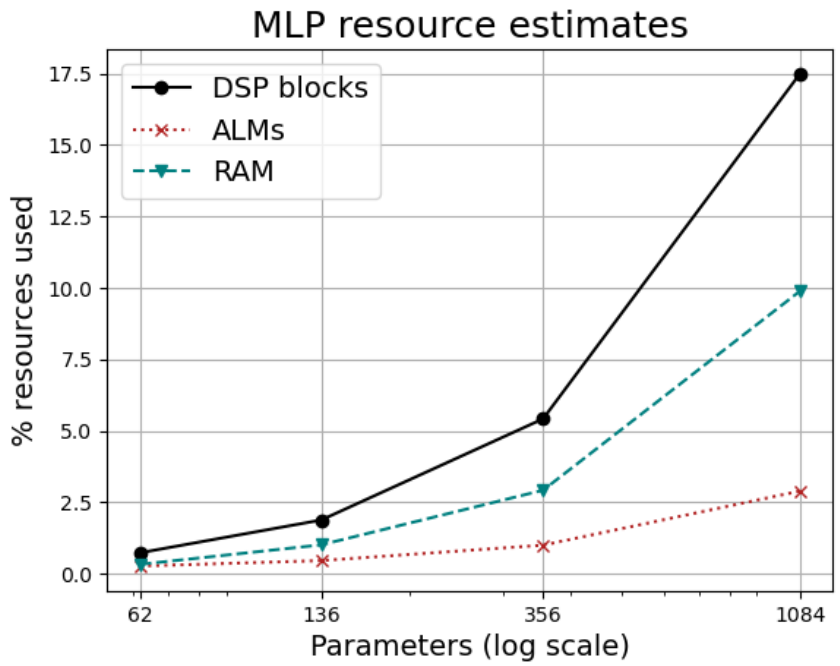
FPGA deployment

- For FPGA translation, ML model must be converted to HLS
- Frameworks HLS4ML and Xilinx/FINN are used for conversion
- For **resource estimation :**
  - PyTorch model is translated to **ONNX,** then HLS4ML generates HDL code
  - Using **Quartus** RTL compilation resource were estimated on target device (**Intel S10 GX**)

HLS4ML documentation

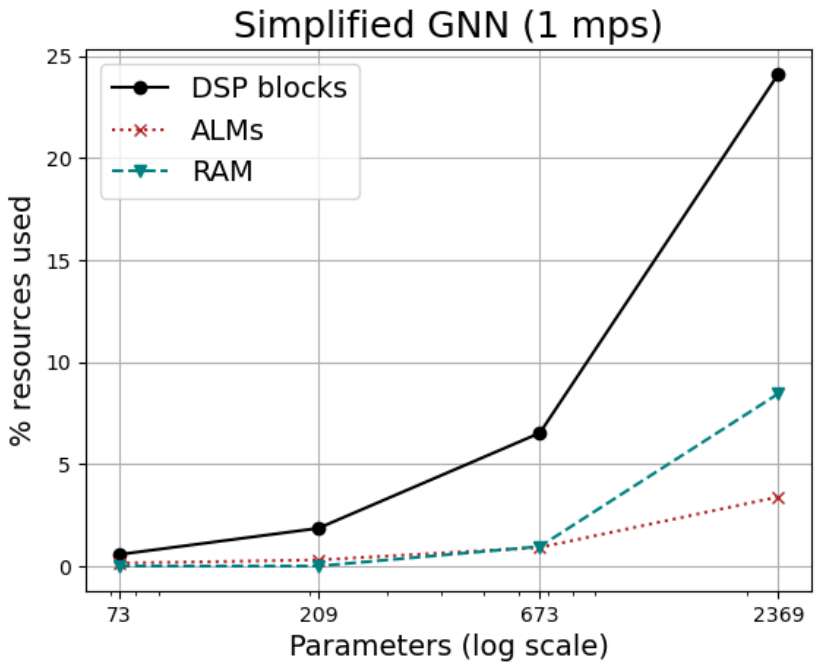
FINN documentation

Resource Estimates : Intel S10 GX



- Entire model was unrolled onto FPGAs
- Only small models were considered because of the reuse factor 1

Ref :M.Sc. thesis Sara Schjødt Kjaer (public link to be added)



- Aggregation at nodes and Index matrix not supported in HLS4ML
- Simplified GNN with 1 message passing steps: more resources for full GNN

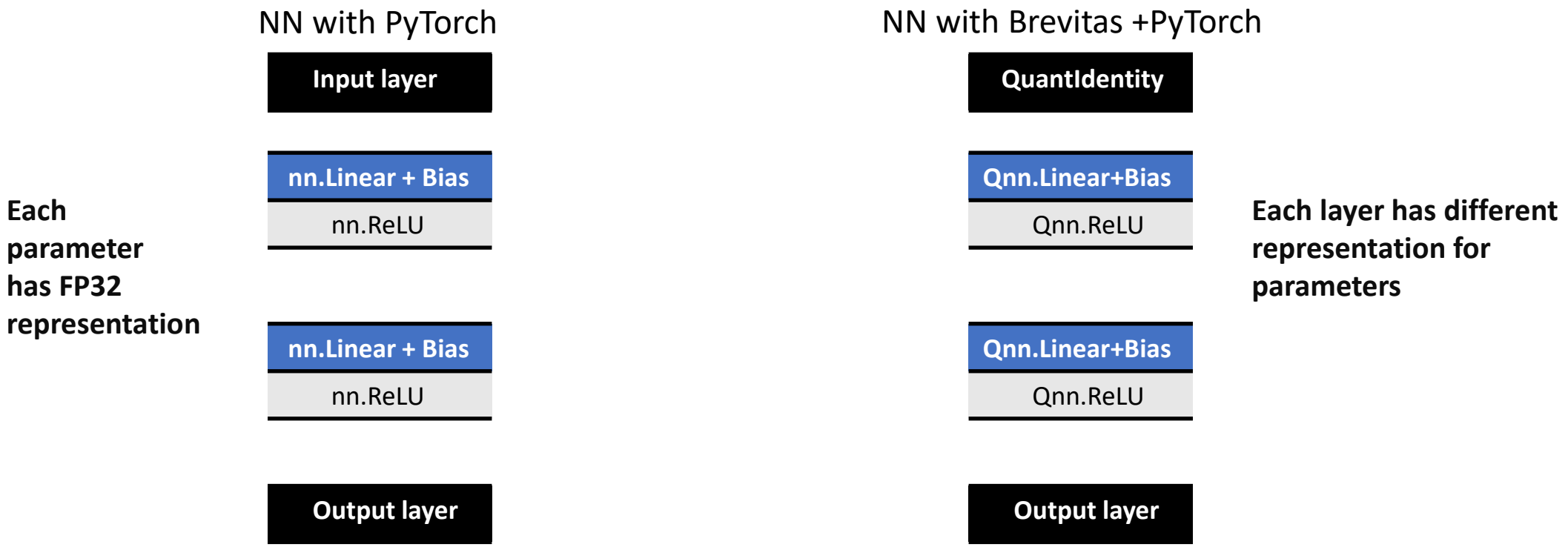
Takeaway

- To completely unroll a model on FPGAs, DSP blocks are predominantly used
- Can we compress the model while keeping the pipeline precise ?
- Yes, by applying ML compression techniques :
  - Quantization Aware Training
  - Pruning

Model Compression study in step 1

Quantization Aware Training (QAT)

- Parameter's representation are trained for arbitrary precision instead of FP32
- Bit width of **weights, activation and bias** is predefined, and training is done afterwards

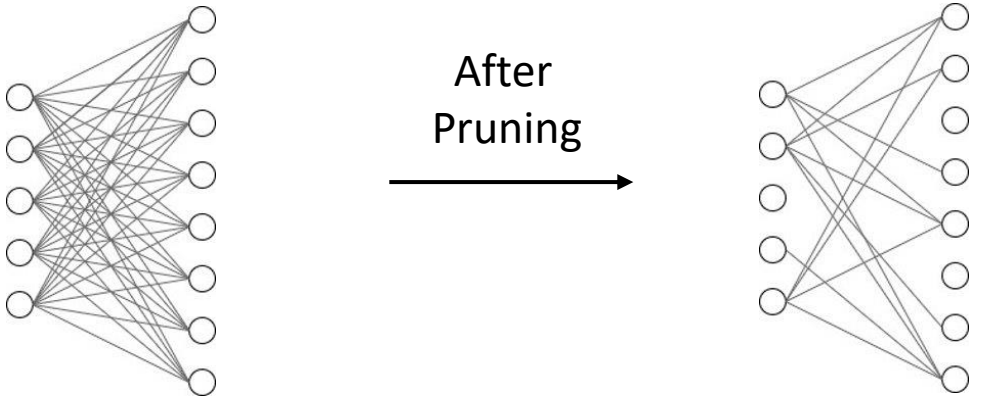


Pruning

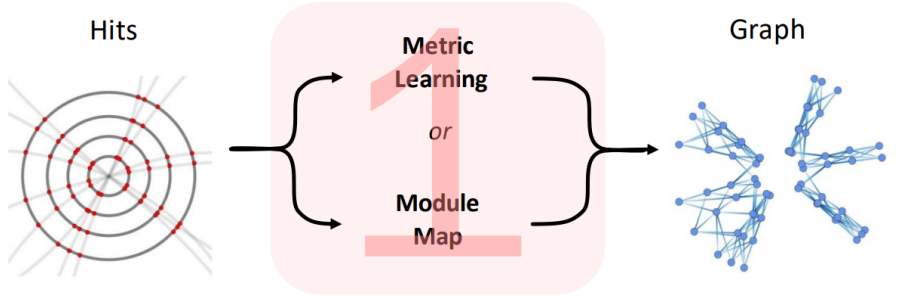
- Idea : Sparse matrix speed up computation
- Pruning is performed iteratively, after certain epochs weights can be removed either with or without structure
- Weights are penalized with L1 loss i.e., making some of them closer to zero and pruning them after some epochs

$$\begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} & w_{15} & w_{16} & w_{17} & w_{18} \\ w_{21} & w_{22} & w_{23} & w_{24} & w_{25} & w_{26} & w_{27} & w_{28} \\ w_{31} & w_{32} & w_{33} & w_{34} & w_{35} & w_{36} & w_{37} & w_{38} \\ w_{41} & w_{42} & w_{43} & w_{44} & w_{45} & w_{46} & w_{47} & w_{48} \\ w_{51} & w_{52} & w_{53} & w_{54} & w_{55} & w_{56} & w_{57} & w_{58} \end{bmatrix}$$

$$\begin{bmatrix} w_{11} & & & w_{14} & w_{15} & & & w_{18} \\ w_{21} & w_{22} & & & w_{25} & & w_{27} & w_{28} \\ & & & & & & & \\ & & & & & & & w_{48} \\ w_{51} & w_{52} & & & w_{55} & & & w_{58} \end{bmatrix}$$

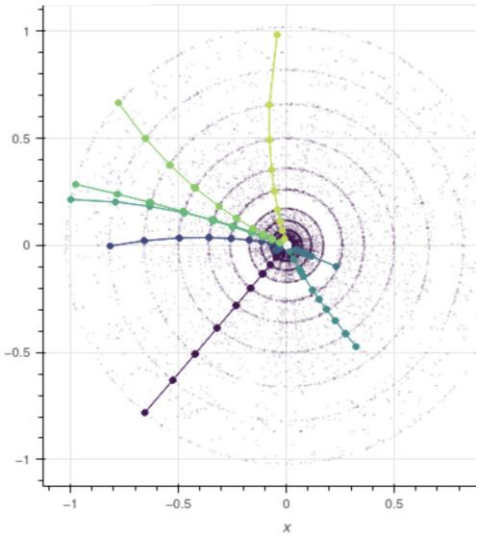


Performance metric for Step 1



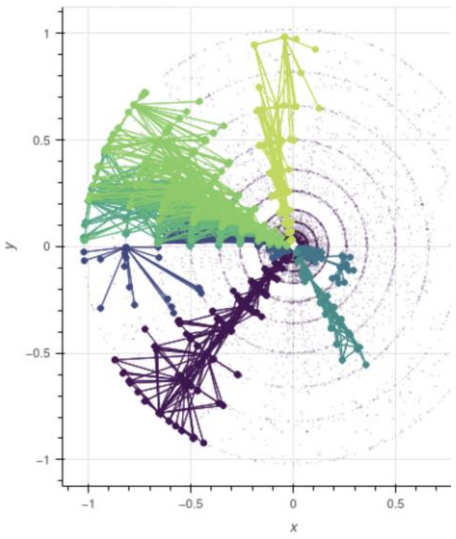
- Aim : To have all true edges with small fraction of false edges
- Performance quantified by Efficiency and Purity
- $eff = \frac{\#\ true\ edges\ in\ \ the\ graph}{\#Total\ true\ edges}$
- $purity = \frac{\#\ true\ edges\ in\ the\ graph}{\#total\ edges\ in\ the\ graph}$

Truth graph



Eff: 100 %
Purity : 100%

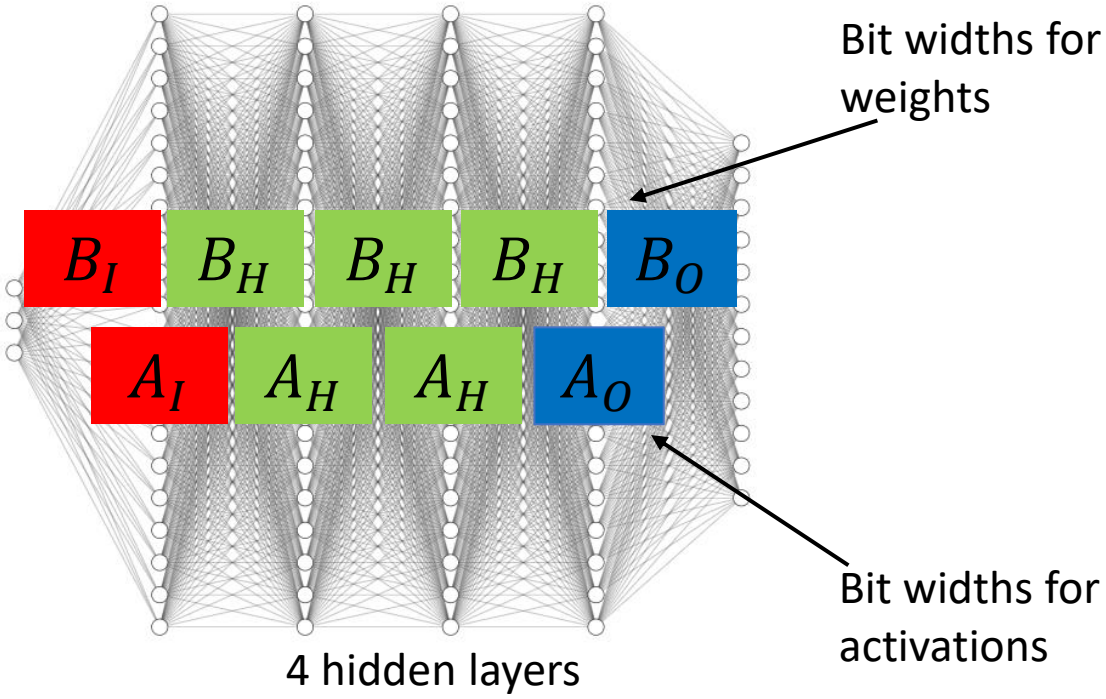
Predicted graph



Eff: 100 %
Purity < 100 %

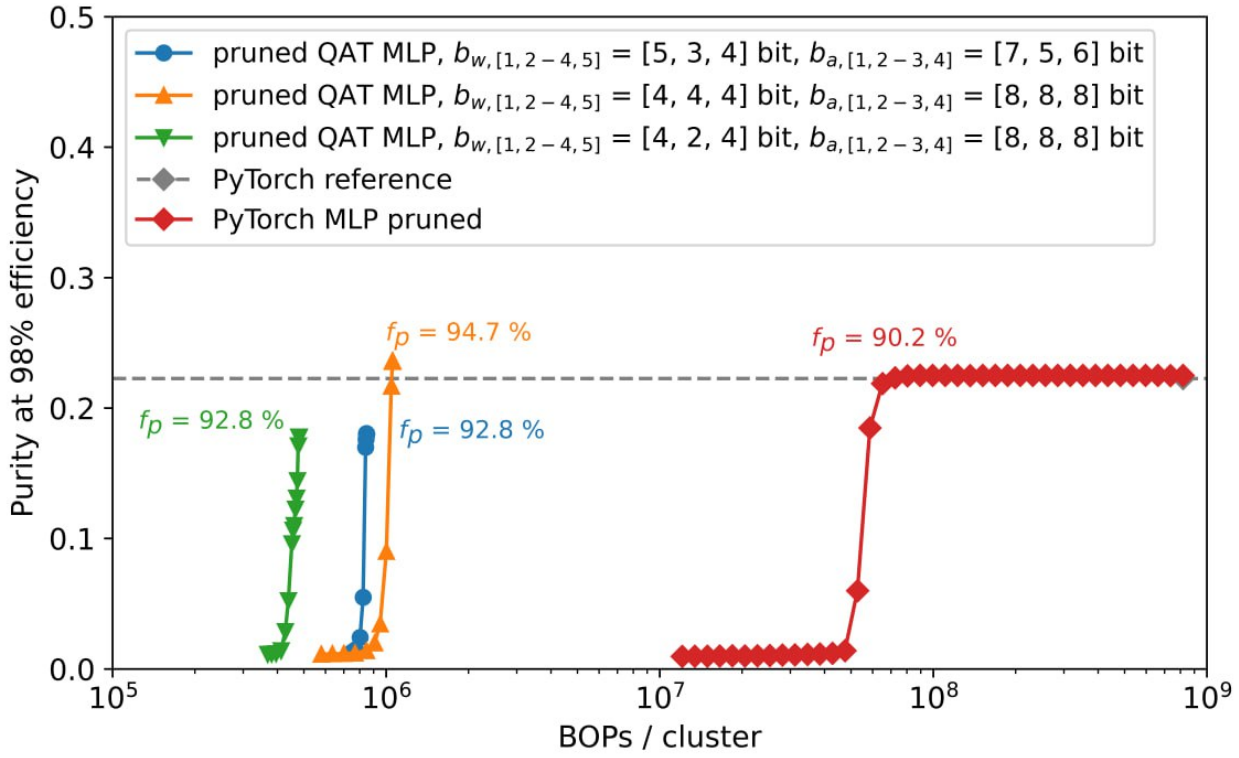
Setting up QAT

- Parameter's bit width are different for different regions
- For our study we have three variable for each parameter:
  - $b_{w[1,2-4,5]} = [B_I, B_H, B_O]$
  - $b_{A[1,2-3,4]} = [A_I, A_H, A_O]$
- Architecture adapted for FPGA target :
  - Batchnorm was used
  - No bias after linear layers



Results with TrackML dataset

- Pruning : 10 % with frequency - 180 epochs
- Performance is retained even by order of 3 reduction in BOPs for QAT+pruned
- Smaller model (less BOPs) can be selected, but the graph size will be larger in comparison



Ref : https://indico.jlab.org/event/459/contributions/11375/

Summary and Outlook

- Translation of GNN based pipeline on FPGAs with HLS4ML and resource estimation
- Resource estimation suggests to compress ML model
- Model compression techniques studied for Step 1 (Graph Construction)
- For non-ML methods manual translation of the code to hardware (VHDL)

- Outlook:
  - Resource estimate for QAT+Pruned MLP
  - Model compression study for the GNN and resource estimation
  - Study with AMD FPGAs via FINN translation
  - Graph segmentation to reduce the graph size
  - Performance with ITk data

Thank you for your attention

Questions ?

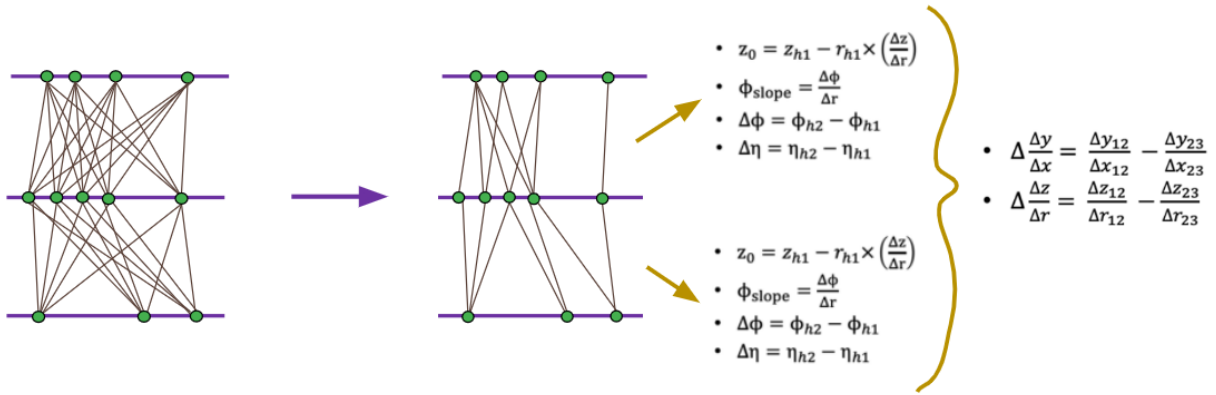
Backup Slides

Translation of non-ML Steps : Module Map

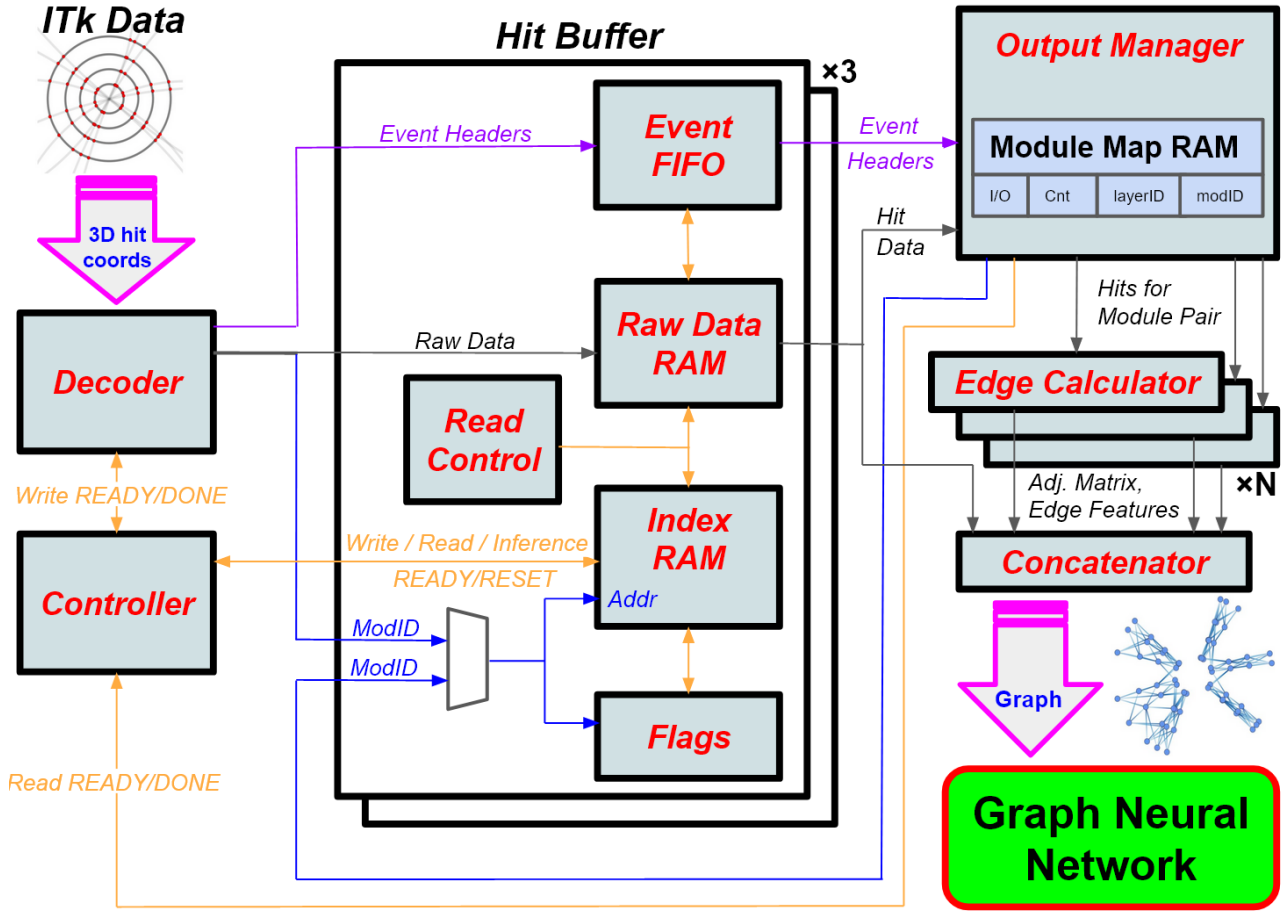
Module Map

Goal of graph construction:
Limit amount of false edges while keeping true edges

- For each module pair, find max and min of geometric values
- Apply geometric cuts for each pair
- Construct map of possible connections/edges
- Store in permutation invariant matrix

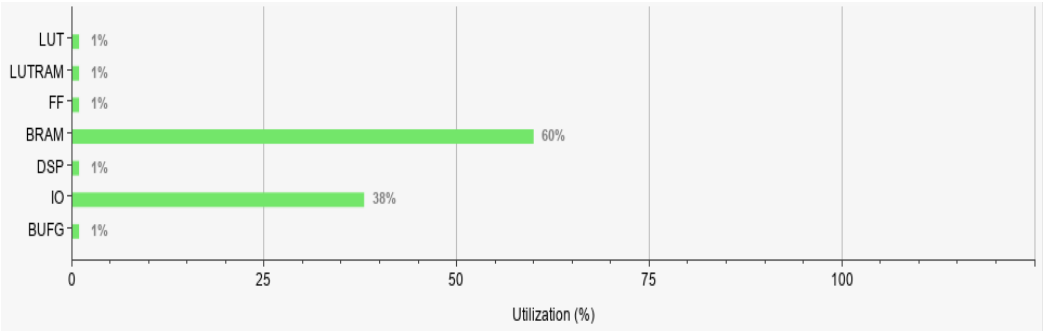


VHDL implementation of Module Map



Module Map: VHDL implementation

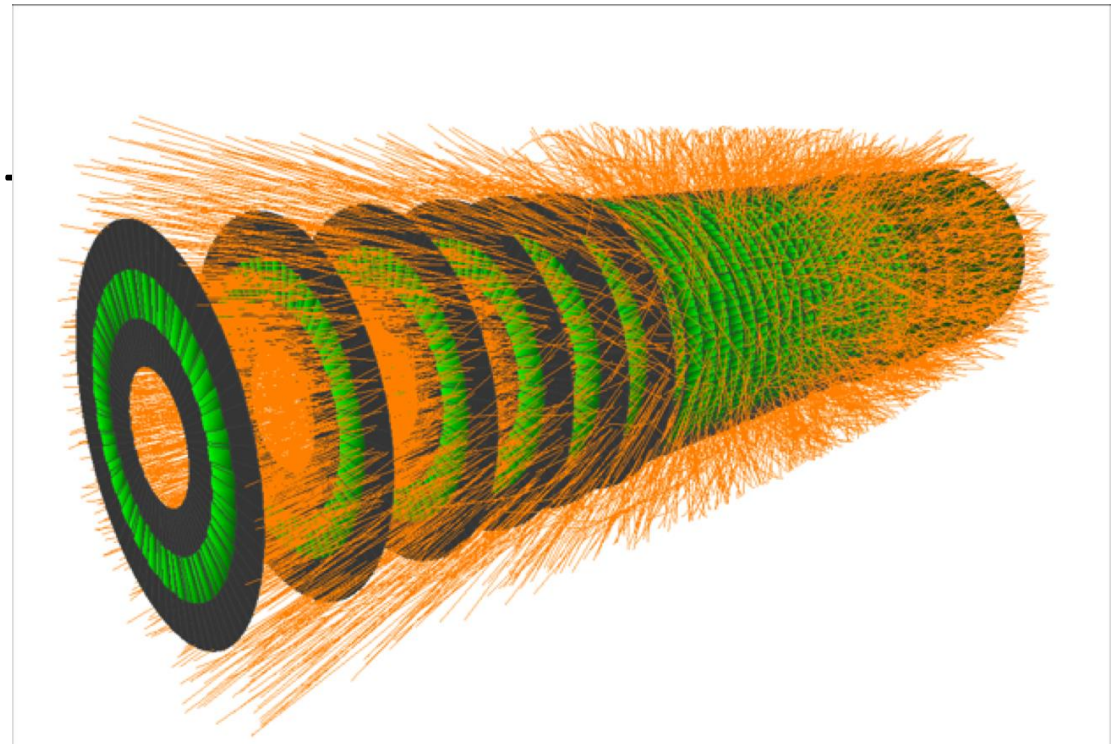
- Preliminary design implemented targeting realistic ITk conditions
- End-to-end simulation working
- (Very) preliminary resource utilization estimates on AMD xcvu37p-fsvh2892-2L-e

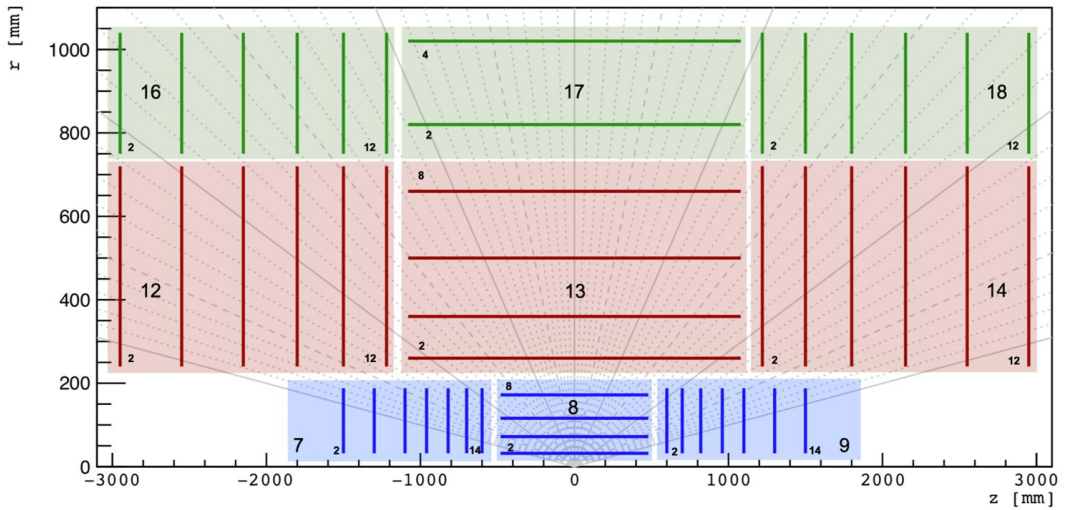


Name	CLB LUTs (1303680)	CLB Registers (2607360)	CLB (162960)	Block RAM Tile (2016)	DSPs (9024)
graph_constructor_top	6784	4997	2572	1214	5
module_mapper_0 (module_mapper)	1739	984	919	474	0
concatenator_0 (concatenator)	0	90	30	52	0
GEN_EDGE_CALC[1].edge_calc_gen (edge_calc__xdcDup__1)	68	124	64	0	0
GEN_EDGE_CALC[0].edge_calc_gen (edge_calc)	4978	3799	1637	688	5

What is the TrackML Dataset

- Simplified detector geometry, adapted from early ATLAS ITk designs
- Pile-up 200 conditions like @ HL-LHC
- We are starting with a pre-processed dataset of particles $p_T > 1$ GeV





Translating & obtaining resource estimates

- **Metric learning**
  - MLP extracted from graph construction
  - Converted to ONNX before translation

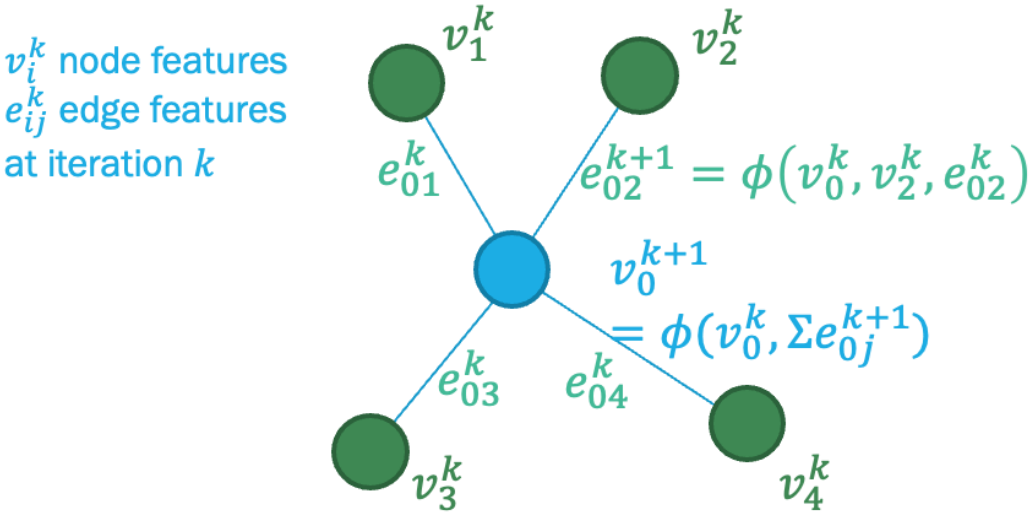
- **Graph Neural Network**
  - Some GNN operations are currently not supported in HLS4ML
  - Constructed a simplified GNN based on GNN4ITk architecture, removing two unsupported model features:
    - Aggregation at nodes (message passing)
    - Indexing operation connecting node and edge indices

- **Track reconstruction**
  - VHDL implementation of walkthrough method developed and tested in simulation

Edge classification (GNN)

Use type of GNN called **“Interaction Network”**

- Interaction Network adds an extra step to the message passing algorithm
- “Edge network” updates edge features, allowing two nodes to form unique relationships
- Improves quality of edge-level predictions



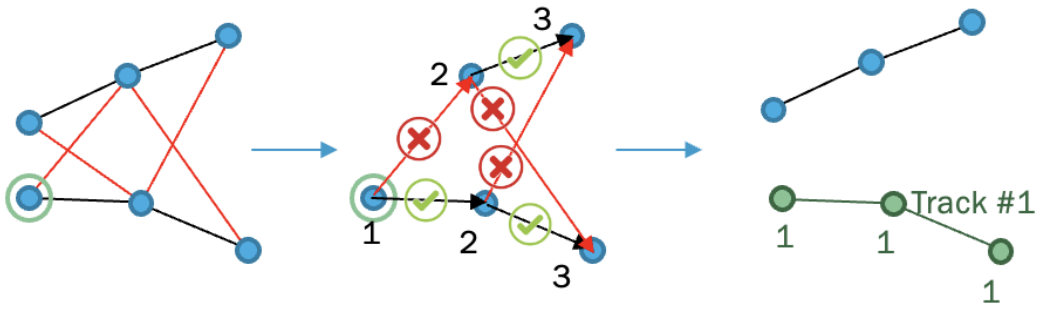
Finally, edge scores are assigned to all edges in an event

Use threshold value (e.g. ~0.5) to discard false edges

Track reconstruction methods

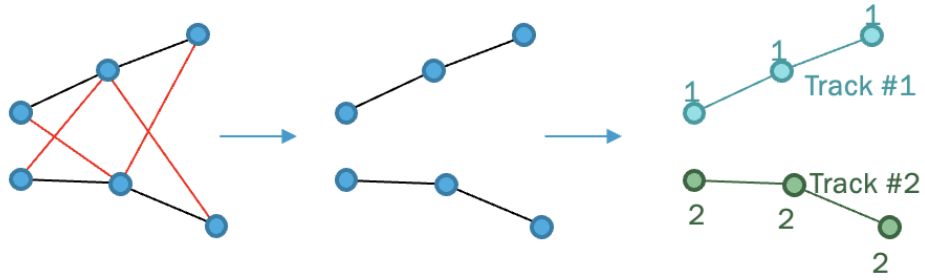
Method 1: Walkthrough

- Identify starting node
- Traverse edges with high scores
- Longest path found
  → track candidate



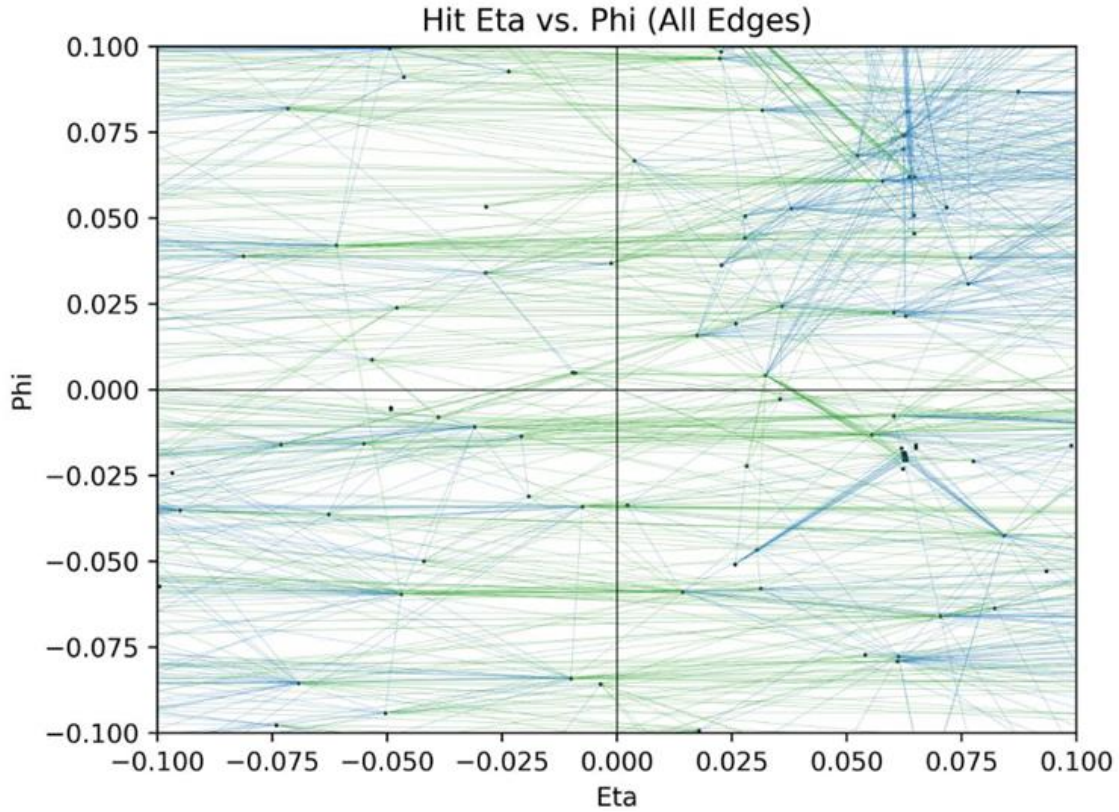
Method 2: Connected components

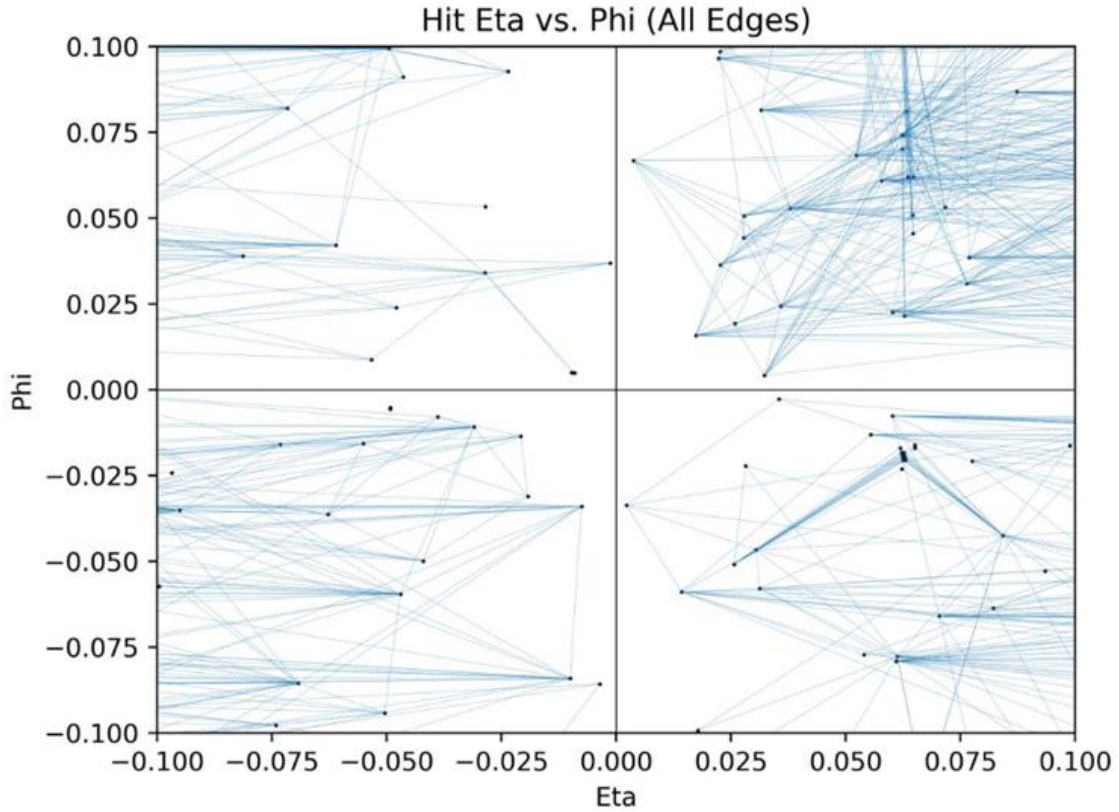
- For edges above threshold score, identify connected paths
- Assign component index to nodes



Method 3: Connected components followed by walkthrough

Segmentation :Subgraph Edge Visualization (all edges)





Black points: hits = nodes
Blue lines: edges preserved in subgraph (nodes in same segments)
Green lines: edges NOT preserved in subgraph (nodes in different segments)