Trigger Level Tracking With Neural Networks on Heterogeneous Computing Systems

Speaker: Alex Gekow (Ohio State University)

On Behalf of the **ATLAS** Collaboration

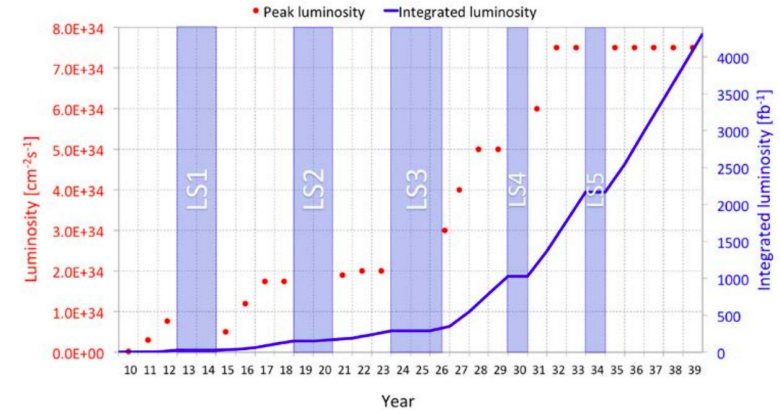
Connecting The Dots 2023

Outline

- ATLAS Track Trigger Requirements
- Problem Statement
  - Why consider ML for tracking?
- Algorithmic Overview
  - How we using ML to approach tracking
- Towards a Heterogenous Implementation
  - Current efforts focused on FPGA (GPU also possible)

ATLAS Phase-II Upgrade

LHC will be undergoing “High Luminosity” Upgrade by increasing the bunch density of pp collisions (amongst other things)



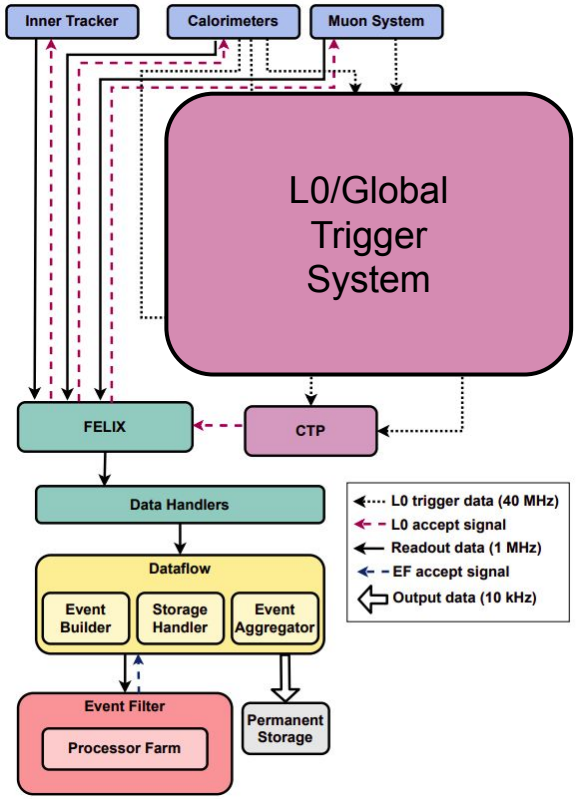
<u>Original ATLAS Track Trigger plan (prior to 2021)</u>

- Use associative memory ASICS to accommodate 1-4 MHz readout rate in high lumi conditions
- Decision to not move forward with ASICS,

  How to move forward with the ATLAS track trigger?

  - CPU & Software only?
  - Heterogeneous computing?

Event Filter



ATLAS is being re-designed to cope with $\langle\mu\rangle$ = 200 conditions

Hit combinatorics explode! Tracking becomes **increasingly difficult**

Relevant **tracking** and **TDAQ** upgrades include:

- Hardware:
  - Phase-II Inner Tracker (ITk)
  - Event Filter (EF) CPU/Accelerator farm
    - Potentially using commodity accelerators (GPU & FPGA)
    - Pursue accelerators to potentially mitigate risks related to power and cost
- Software/Firmware
  - Tracking algorithms running on EF computing farm

Goal of **150 kHz** “full scan” tracking

Problem Statement

Tracking is the most time consuming step of event reconstruction in ATLAS making it difficult to implement in the trigger

The ***Event Filter Tracking Group*** within ATLAS is developing methods for tracking in the EF farm that utilize commodity hardware accelerators

Many proposed solutions (several involving ML) to tackle this complex pattern recognition problem:

- Hough Transform
- Kalman Filter on GPU
- Graph Neural Networks
- Deep Metric Learning
- *Machine Learning Hit Search* (this talk)

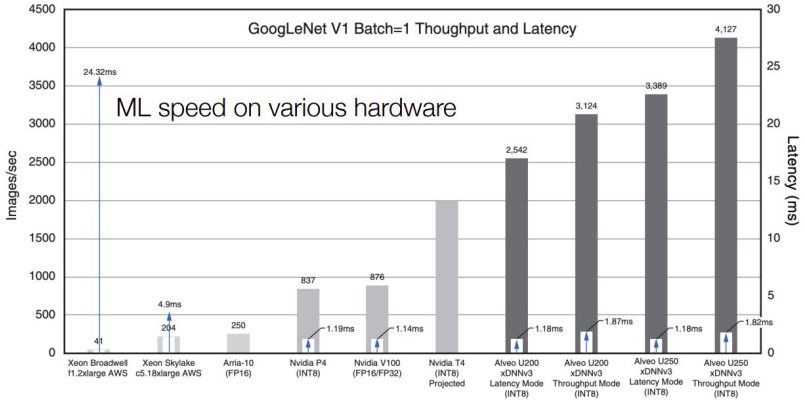
Many with the intent of leveraging heterogeneous computing for **low** latency **high** throughput execution

Much work has been documented in the EF Tracking TDR:
https://cds.cern.ch/record/2802799/files/ATLAS-TDR-029-ADD-1.pdf

Why Machine Learning?

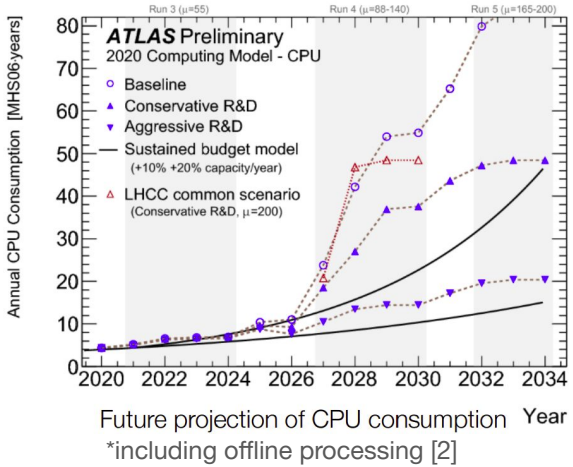
& Why leverage heterogeneous computing?

Machine learning algorithms and the hardware & software required to deploy them are a rapidly expanding domain (e.g Tesla FSD, Apple neural engine, Google tensor, Vitis AI…)



- ML is *at its core* is a sequence of large matrix multiplications
- Reducing complicated tasks to batched matrix multiplication can allow for massive parallelization
- Hardware such as GPU/FPGA are capable of highly efficient and fast execution of ML operations

[1]https://www.xilinx.com/support/documentation/white_papers/wp504-accel-dnns.pdf



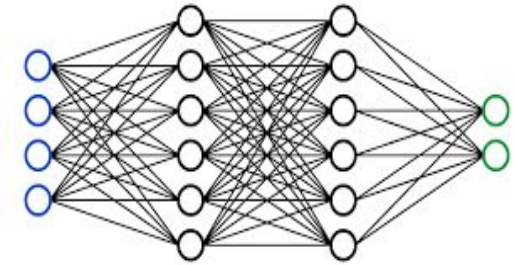
- The overall CPU demands of ATLAS are growing steadily
- Finding areas to offload CPU work to other devices can save time, money, and energy

[2] https://cds.cern.ch/record/2729668/files/LHCC-G-178.pdf

ML Tracking - Pattern Recognition

- Most ML approaches view tracking as a *pattern recognition* problem rather than a pure physics problem
  - Precision fits already more than capable of determining track parameters - no need for ML!
- Pattern recognition: Simply…Connect The Dots
  - Full Combinatorial Kalman Filter (CKF)
    - Precise, runs traditionally on CPU
  - Use ML to find collections of compatible hits and perform KF after
    - Well suited to run on CPU, GPU, FPGA

Hit level information



Track Candidates

ATLAS Primary Tracking

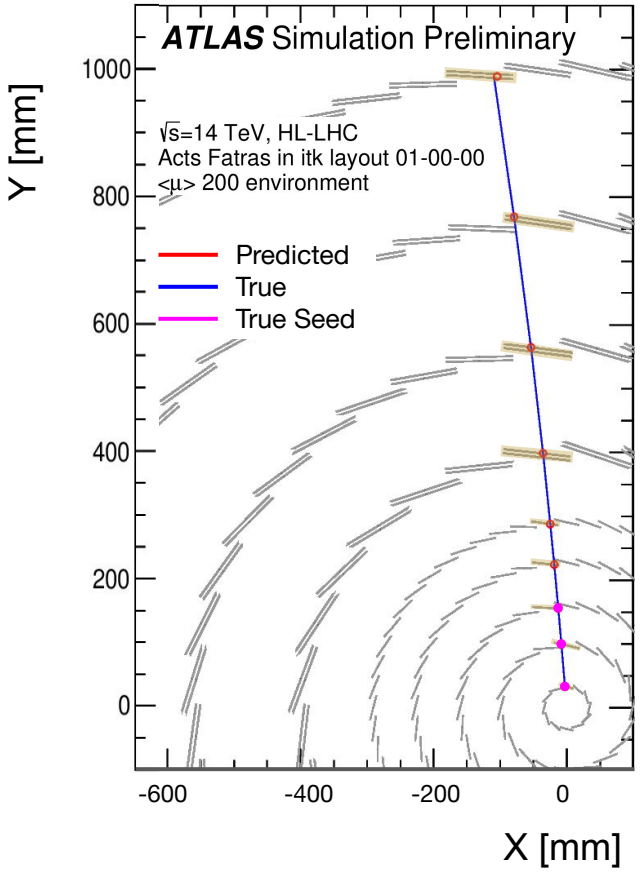
Focus on these steps

Overview of Algorithms

ML Hit Search

Track Finding (Inside out track extension)

1. Predict a hit coordinate for each track seed (3 hits from the innermost pixel layers)
2. Open a search window for compatible hits around predicted coordinate
3. Create “track” candidates from hits within window
4. Repeat

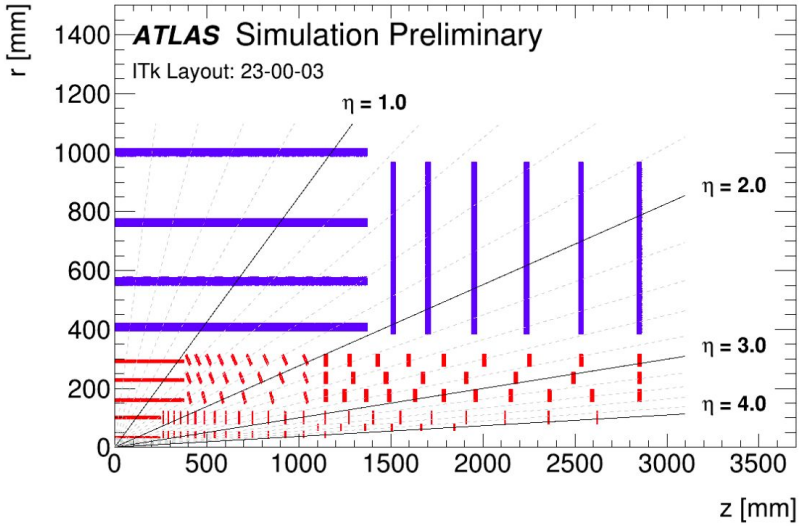


(Right) A true track (blue) is shown alongside the coordinates predicted by a neural network (red) in a simulated ITk geometry ttbar event with <μ>=200 environment [1]. The grey rectangles depict the detector modules.Modules that the particle passes through are highlighted in yellow.

[1] Technical Design Report for the ATLAS Inner Tracker Strip Detector, ATLAS Collaboration - ATLAS-TDR-025

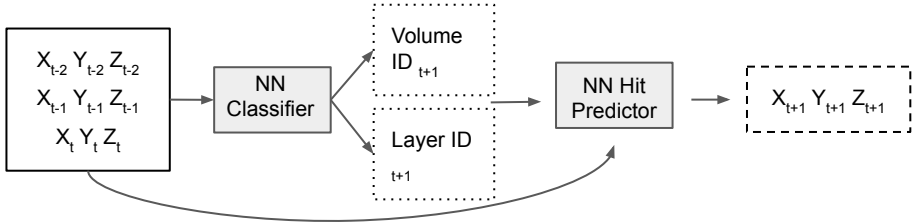
Overview of Algorithms

The irregular orientation of detector layers of ITk make straightforward coordinate prediction difficult for a NN



Expected tracking and related performance with the updated ATLAS Inner Tracker layout at the High-Luminosity LHC ATL-PHYS-PUB-2021-024.pdf

If spacepoints **and** the detector layer are given as input, the NN learns to associate discrete sets of coordinates to each detector layer

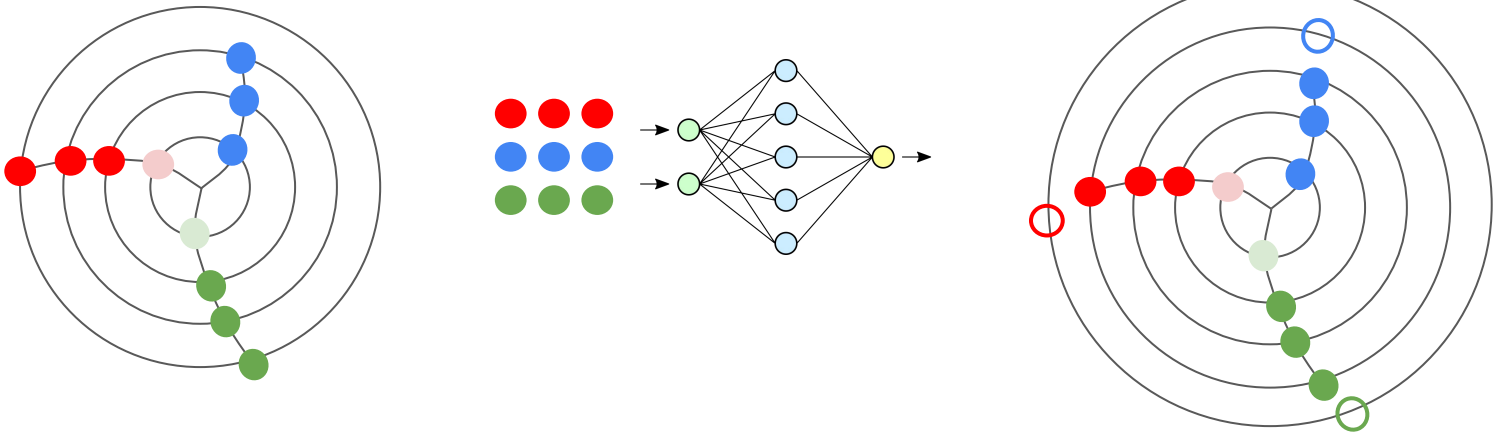


(Above) Hit coordinate prediction neural network Architecture. Available hit coordinates are used as input to predict the expected detector layer/volume of the next hit. The predicted volume/layer as well as the available hit coordinates are used to predict the expected next hit coordinate

Parallelized Track Finding

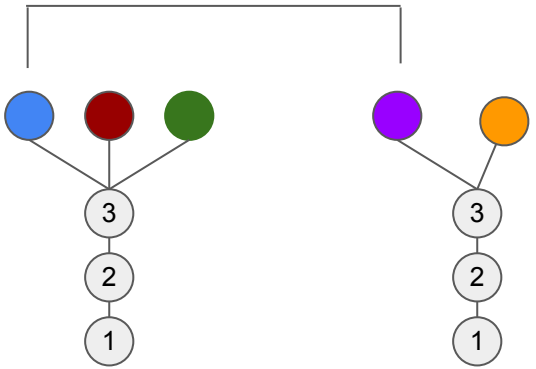
Predicting hit coordinates with a NN enables *parallelized* predictions for faster execution

Available proto-tracks can be batched into a matrix of size (proto-tracks × N input features) which is input to the NN, which outputs a matrix of size (proto-tracks × N output features)



Parallelized Track Finding

- In practice we may find more than one hit per proto-track
  - Grow “Multitrajectories” from track seeds
  - Tree structure containing all track candidates for each seed
- We cannot run on arbitrarily sized matrices due to I/O and memory constraints
  - Optimally batch proto-tracks into configurable sized matrices



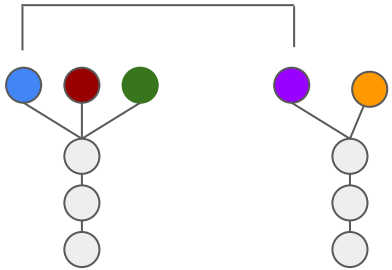
Input Matrices to prediction NN

$$\begin{bmatrix} X_1\ Y_1,\ Z_1\ X_2\ Y_x\ Z_z\ X_3\ Y_3\ Z_3 \\ X_1\ Y_1,\ Z_1\ X_2\ Y_x\ Z_z\ X_3\ Y_3\ Z_3 \\ X_1\ Y_1,\ Z_1\ X_2\ Y_x\ Z_z\ X_3\ Y_3\ Z_3 \\ X_1\ Y_1,\ Z_1\ X_2\ Y_x\ Z_z\ X_3\ Y_3\ Z_3 \end{bmatrix} \quad \begin{bmatrix} X_1\ Y_1,\ Z_1\ X_2\ Y_x\ Z_z\ X_3\ Y_3\ Z_3 \end{bmatrix}$$

Seed 1
Three compatible hits found

Seed 2
Two compatible hits found

Parallelized Track Finding



Seed 1
Three compatible hits found

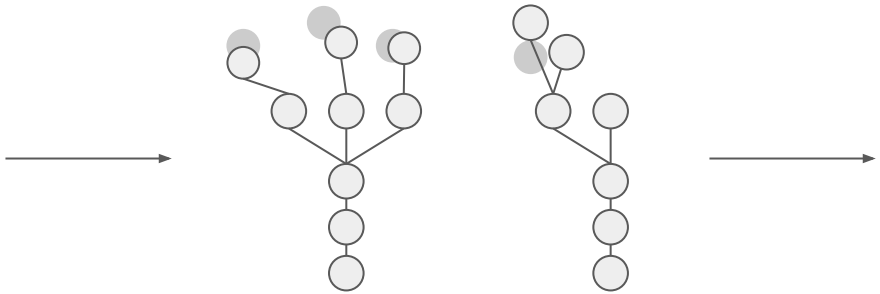
Seed 2
Two compatible hits found

Input Matrix to prediction NN

$$\begin{bmatrix} X_1\ Y_1\ Z_1\ X_2\ Y_x\ Z_z\ X_3\ Y_3\ Z_3 \\ X_1\ Y_1\ Z_1\ X_2\ Y_x\ Z_z\ X_3\ Y_3\ Z_3 \\ X_1\ Y_1\ Z_1\ X_2\ Y_x\ Z_z\ X_3\ Y_3\ Z_3 \\ X_1\ Y_1\ Z_1\ X_2\ Y_x\ Z_z\ X_3\ Y_3\ Z_3 \end{bmatrix}$$

Predictions

$$\begin{bmatrix} X_4\ Y_4\ Z_4 \\ X_4\ Y_4\ Z_4 \\ X_4\ Y_4\ Z_4 \\ X_4\ Y_4\ Z_4 \end{bmatrix}$$



Process is continued for all active track candidates and terminates when no track candidates remain

Prediction resolution weaker than detector resolution leads to many fake/duplicate tracks. Filter them with a second stage NN

Overview of Algorithms

Each operating on spacepoint representation of tracks

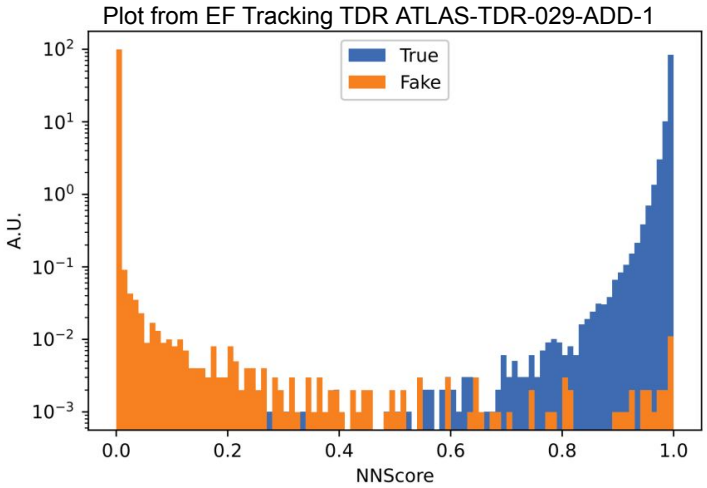
ATLAS EXPERIMENT

THE OHIO STATE UNIVERSITY

Ambiguity Resolution

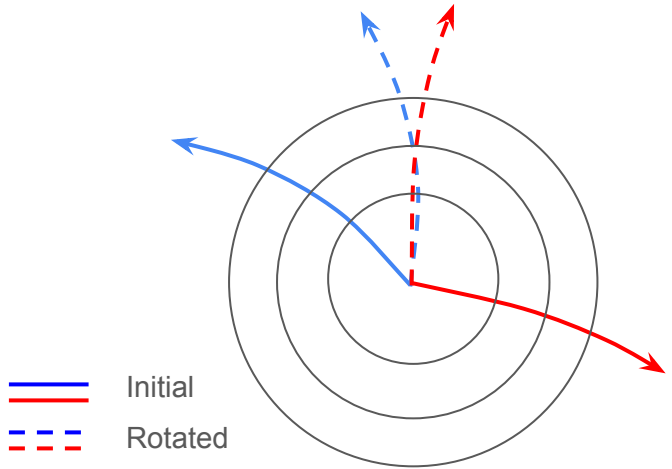
1. Score each track candidate with a NN
2. Compare tracks with more than M shared hits
3. Keep only the track with the highest NN score

Can be trained on the output tracks from any pattern recognition step involving spacepoints

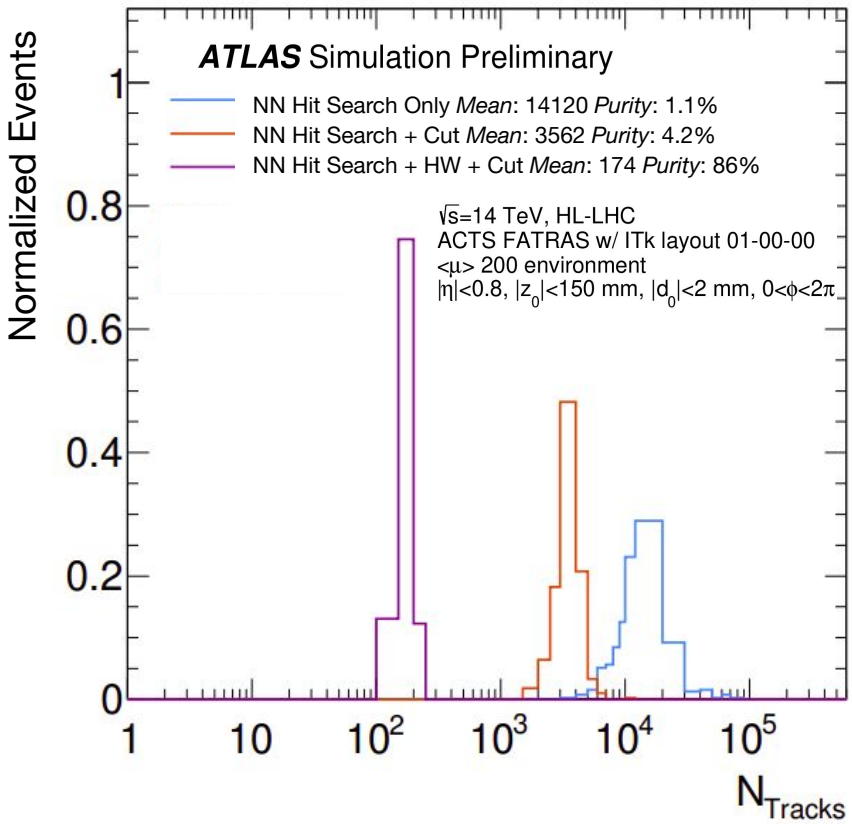


All of the spacepoints along a track are enough for a track to be evaluated as Real or Fake

The approximate azimuthal symmetry of ATLAS can be exploited for better performance by rotating tracks to lie in the same initial direction

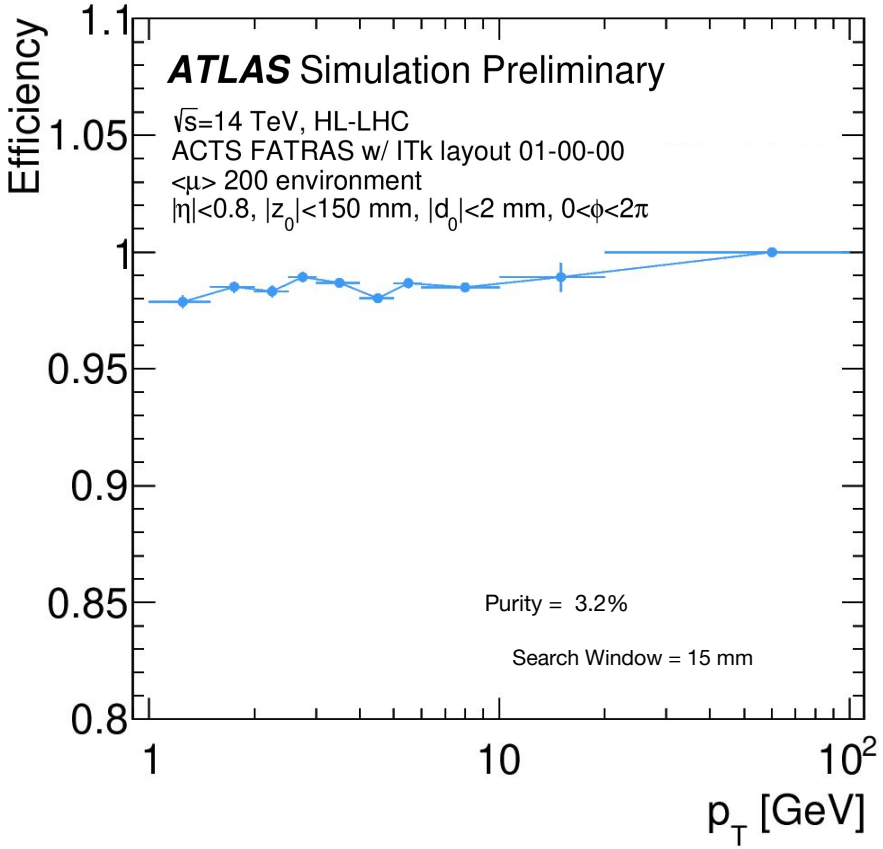


Tracking Performance



- The number of track candidates that are found by the machine learning (ML) - hit search algorithm in a $<\mu>$=200 event, using a search window of 20 mm at each prediction point (blue).
- The number of candidate tracks after filtering with a NN are shown in orange
- Any candidate tracks sharing hits are compared, and only that with the highest NN score is kept (HW in legend)
- Tracking efficiency deviates at most 1% at each stage as seen in performance tests with various other detector geometries

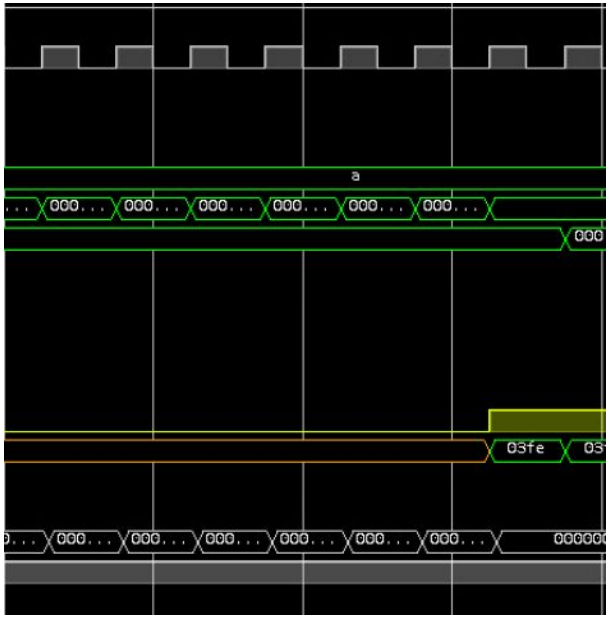
Tracking Performance



- The track finding efficiency in the ITk barrel after the initial hit search is performed using a search window of 15 mm.
- Efficiency is defined as the fraction of true tracks found with respect to the number of true tracks in the event in the barrel region of the detector as described on the figure.
- Tracking efficiency deviates at most 1% at each stage as seen in performance tests with various other detector geometries
- ACTS FATRAS (FAst TRAcking Simulation) does not include hadronic simulation

Tracking & FPGA Performance

FPGA Vivado Simulation



Clock

Input

Output

Perfectly pipelined execution of Ambiguity Resolution NN requires 10+B clockcycles where B is the number of “tracks” to evaluate

1 clockcycle = 10ns

Xilinx Alveo U250 FPGA resource usage estimates for ambiguity resolution:

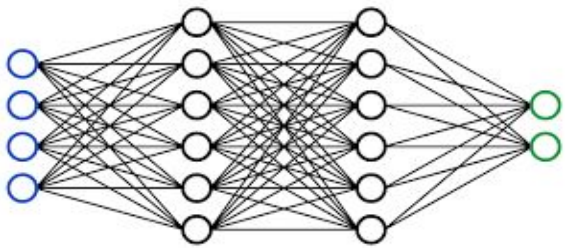
* rough estimates as NN architecture may change

	Latency (ns)	LUT (%)	FF (%)	BRAM/URAM (%)	DSP (%)
Ambiguity Resolution	50	18	1	<0.01	31
Path Finder	50	7	0.5	<0.01	21

Future Work

Potential Algorithmic Improvements

- Metric learning for detector region classification
- KNN hit search
- Dynamic uncertainty estimation
- Synchronous track filtering
- Alternate seeding algorithms as input
- Strategies to account for missing hits



Goal: More precise predictions -> Fewer fakes as output & Faster execution

Demonstrator System

Algorithm being implemented in tracking framework *ACTS* and ATLAS software to be compatible with offline tracking tools

- FPGA kernels being written (NN, I/O, etc.)
- Quantizing/Pruning/Optimizing for latency and efficiency

Goal: end-to-end fully functional implementation

Conclusion

Online tracking with <μ>=200 presents a new set of challenges with many exciting solutions to explore!

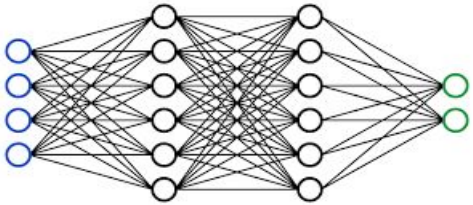
ML on heterogeneous computing systems is a subset of many promising solutions. Factoring the problem into steps similar to offline tracking enables *fast* evaluation of dense tracking environments

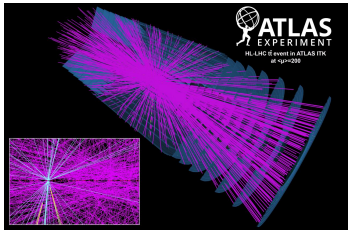
ML algorithm for track extension that is suitable for fast inference on FPGAs

- High Tracking efficiency, fast execution on FPGA/GPU
- Minimal bussing of data between cpu & heterogenous device (No external detector geo or magnetic field info required at run time)

ML Ambiguity resolution/Fake & Overlap Removal Algorithm

- Strong discriminating power between True/Fake tracks
- Also enables fast execution and implementation on FPGA/GPU





https://twiki.cern.ch/twiki/bin/view/AtlasPublic/UpgradeEventDisplays

THE OHIO STATE UNIVERSITY

Thank You