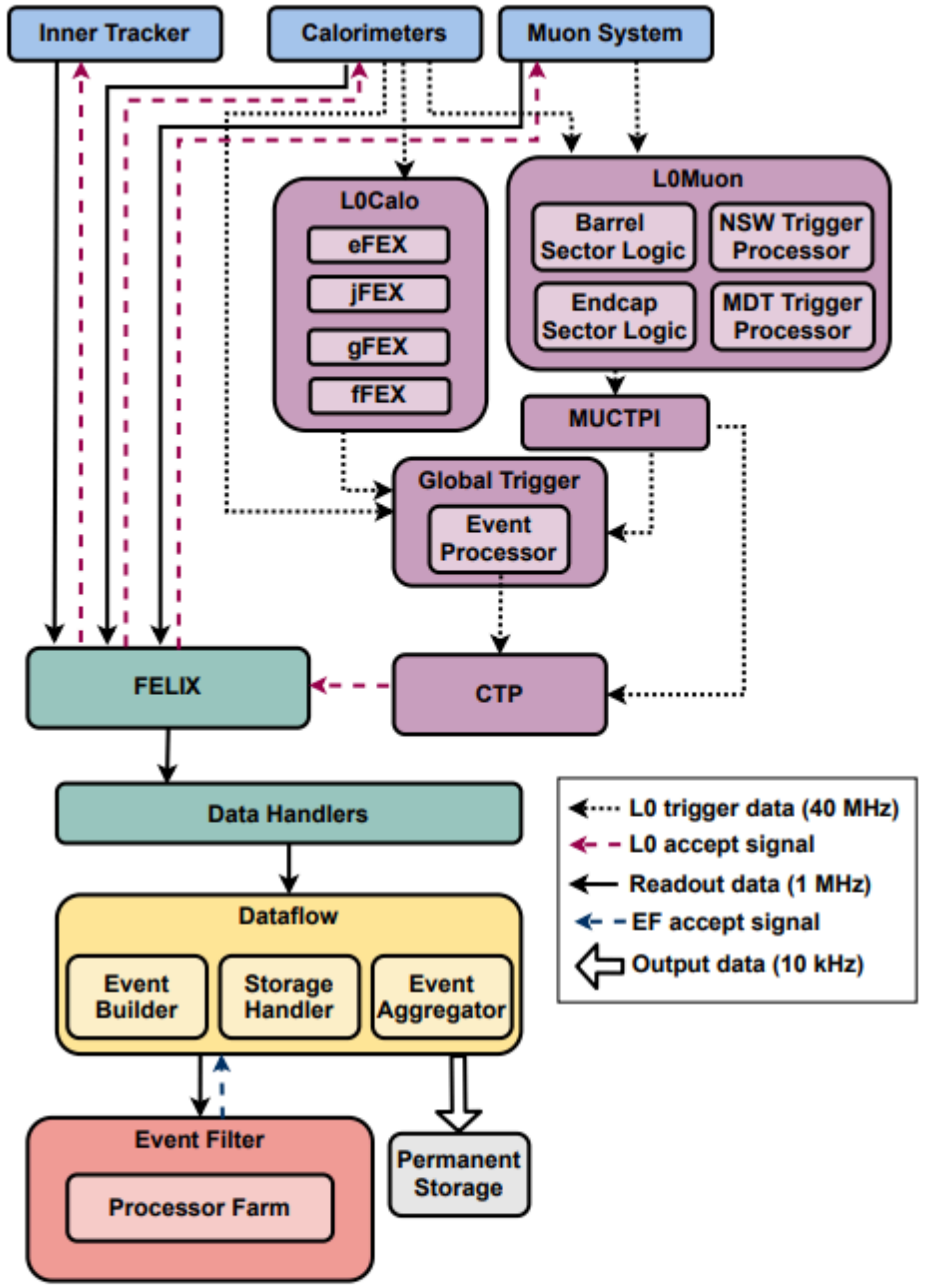
AI/ML in EF Tracking

Haider Abidi (for EF Tracking)
AI/ML Coordinator
Feb 25th, 2024

Introduction

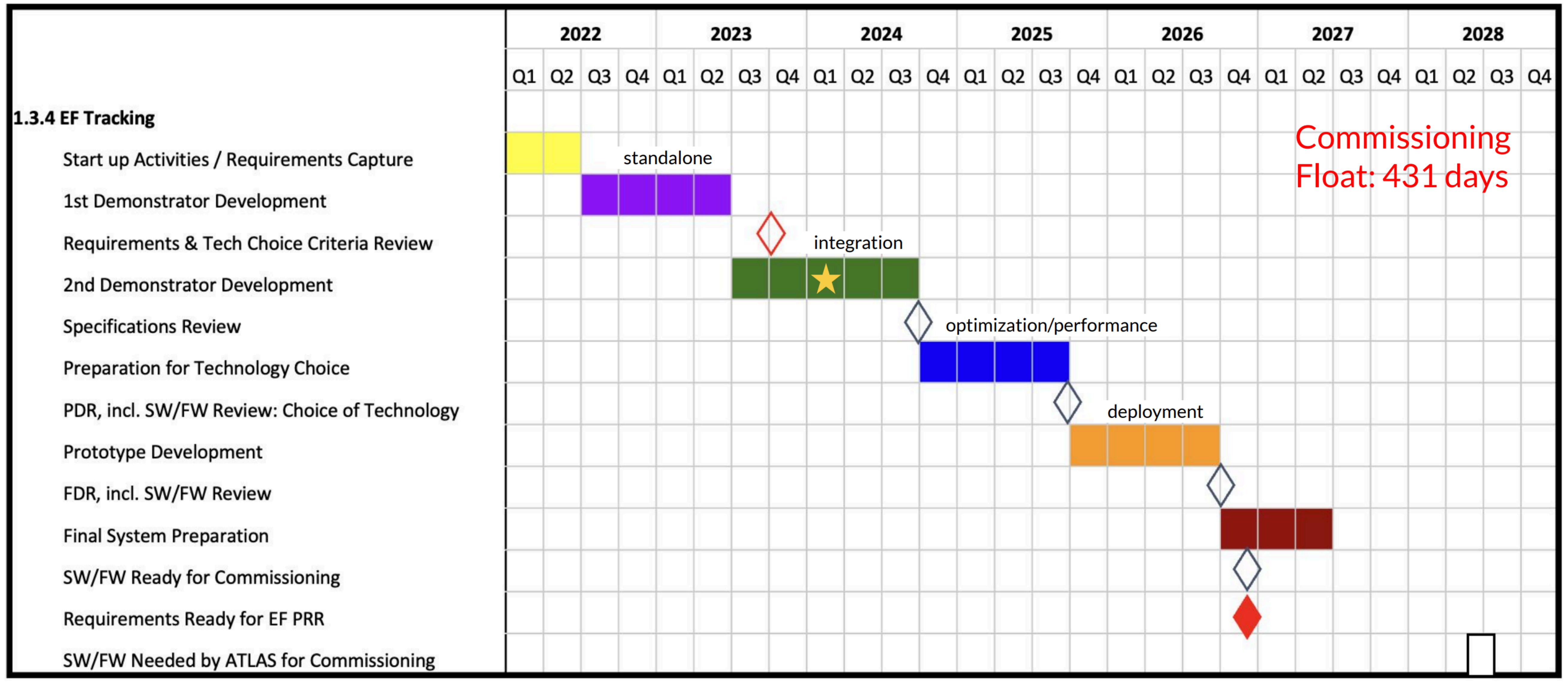
- EF Tracking Review Committee: Recommendation that **"ATLAS commit to a commercial solution for EF Tracking at HL-LHC,"**
  - Heterogeneous devices (e.g., GPUs and FPGAs) allow the CPU to offload specialized tasks, and may provide **power saving** and/or **throughput increase**
- Within this context, use of AI/ML doesn't just stop at pushing performance
  - Computing & throughput requirements are just as critical and necessary
- Use of AI/ML stands within the context of the criteria/constraints of the full project
- For CPUs/GPUs, a lot of work is performed and hosted in other ATLAS groups
  - Slight bias towards FPGA based solution in this talk



EF Tracking: Philosophy & Schedule

- 1st Demonstrator: (Completed) Encouraged “bubbling up” of standalone algorithms
  - Various ideas (including AI/ML) were tested to understand the efficacy and base performance
  - Highly synergistic collaboration across various communities within ATLAS
- 2nd Demonstrator: (Current) Integrate algorithms into tracking pipelines on each technology
  - Promising ideas have been picked, including AI/ML ones
- AI/ML solutions in this presentation are those being integrated for the 2nd demonstrator cycle

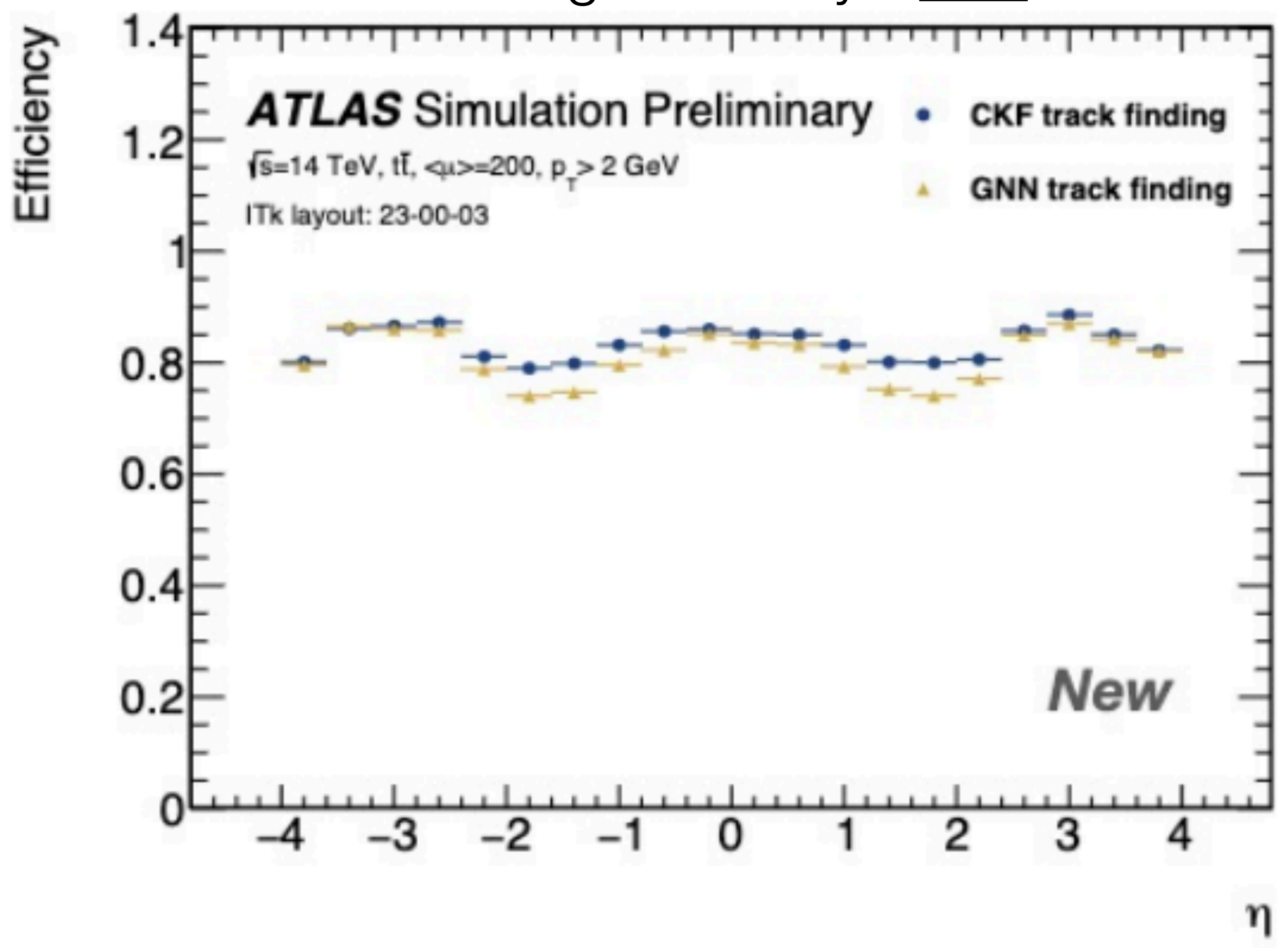
EF Tracking Schedule



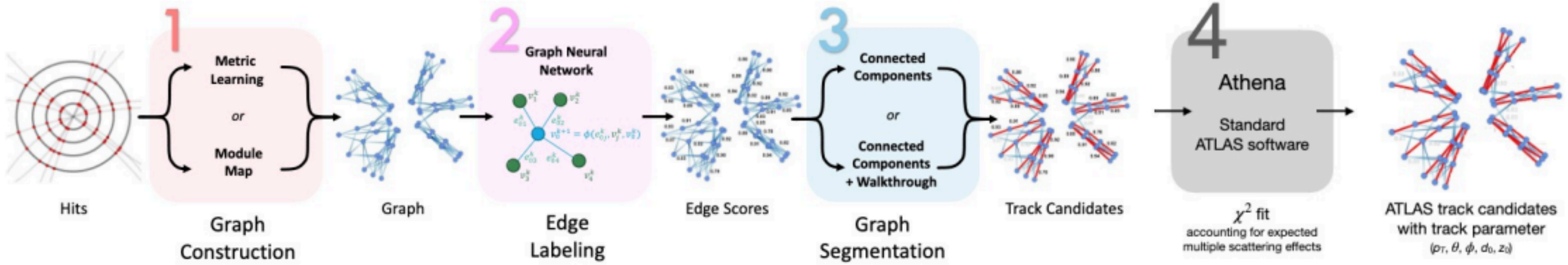
AI/ML Idea: Graph Neural Networks

- Close collaboration with offline - Significant overlap of personnel
- Ongoing studies to update to latest geometry
  - Comparable performance to offline results
- Sparse random data access for messaging passing step is a challenging on accelerators
  - Optimization being performed to reduce inference time and leverage GPU technology to increase throughput
  - Dedicated effort for implementing GNNs on FPGAs

Tracking efficiency - Link

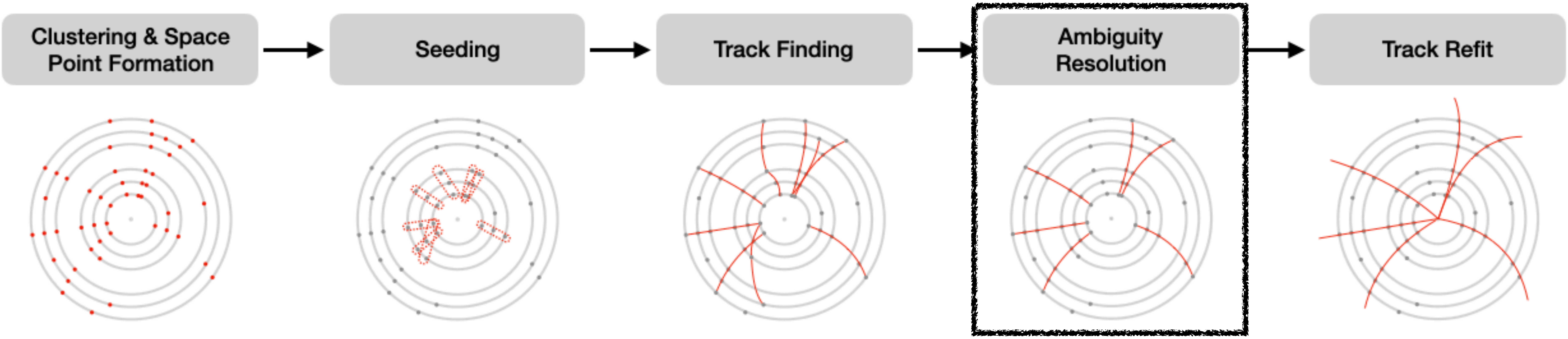


GNN pipeline



ML Enabling Classical Algorithms on Accelerators

Typical Tracking pipeline

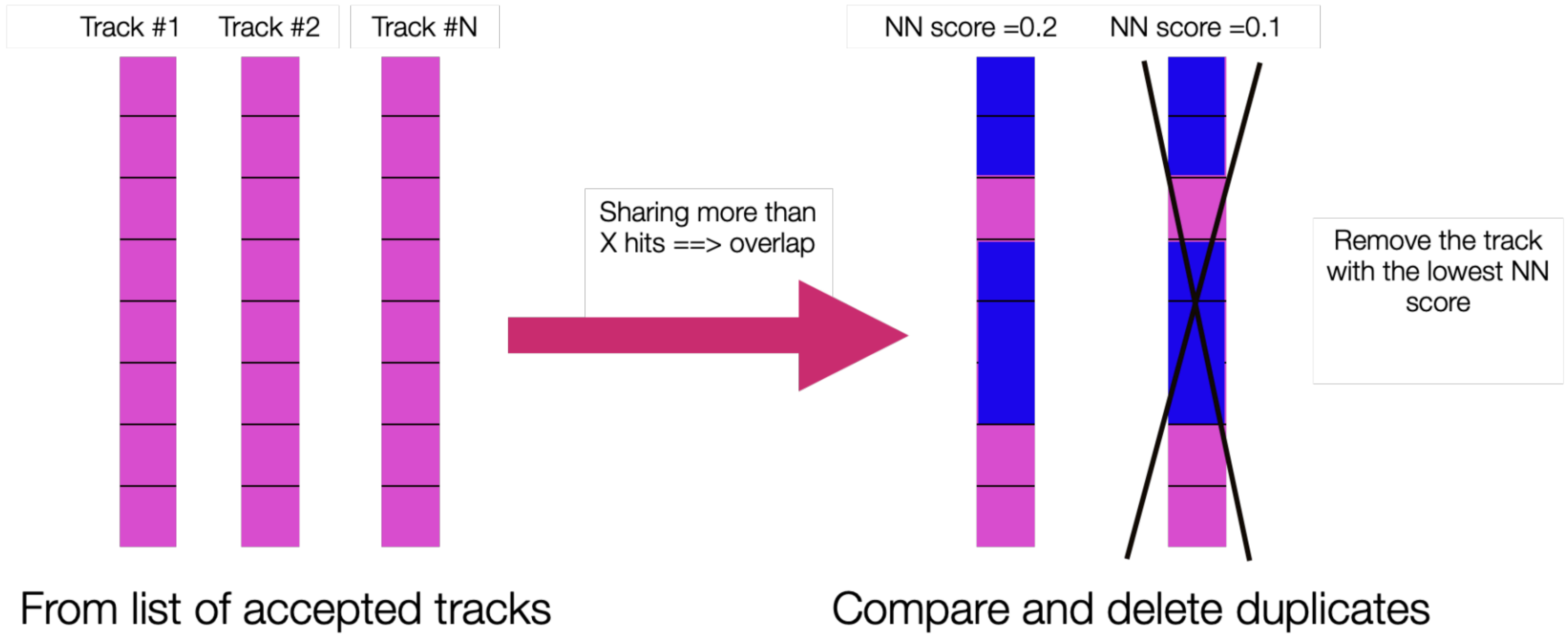


- Conformal (Hough) transform is a relative simple/cheap pattern recognition algorithm
  - Fixed data access pattern is significantly more efficient
  - Cost: Lot of fake hit combinations & No figure of merit on fit quality
- Need to preform a preliminary “Ambiguity resolution” without using the time consuming fit for each track
  - Leverage the performance of ML to predict this figure of merit?
- Classify a vector of x/y/z position coordinates as coming from a 'true or fake track’
  - Established during TDR addendum process - Link
  - Being incorporated into ACTS for seed filtering - Link

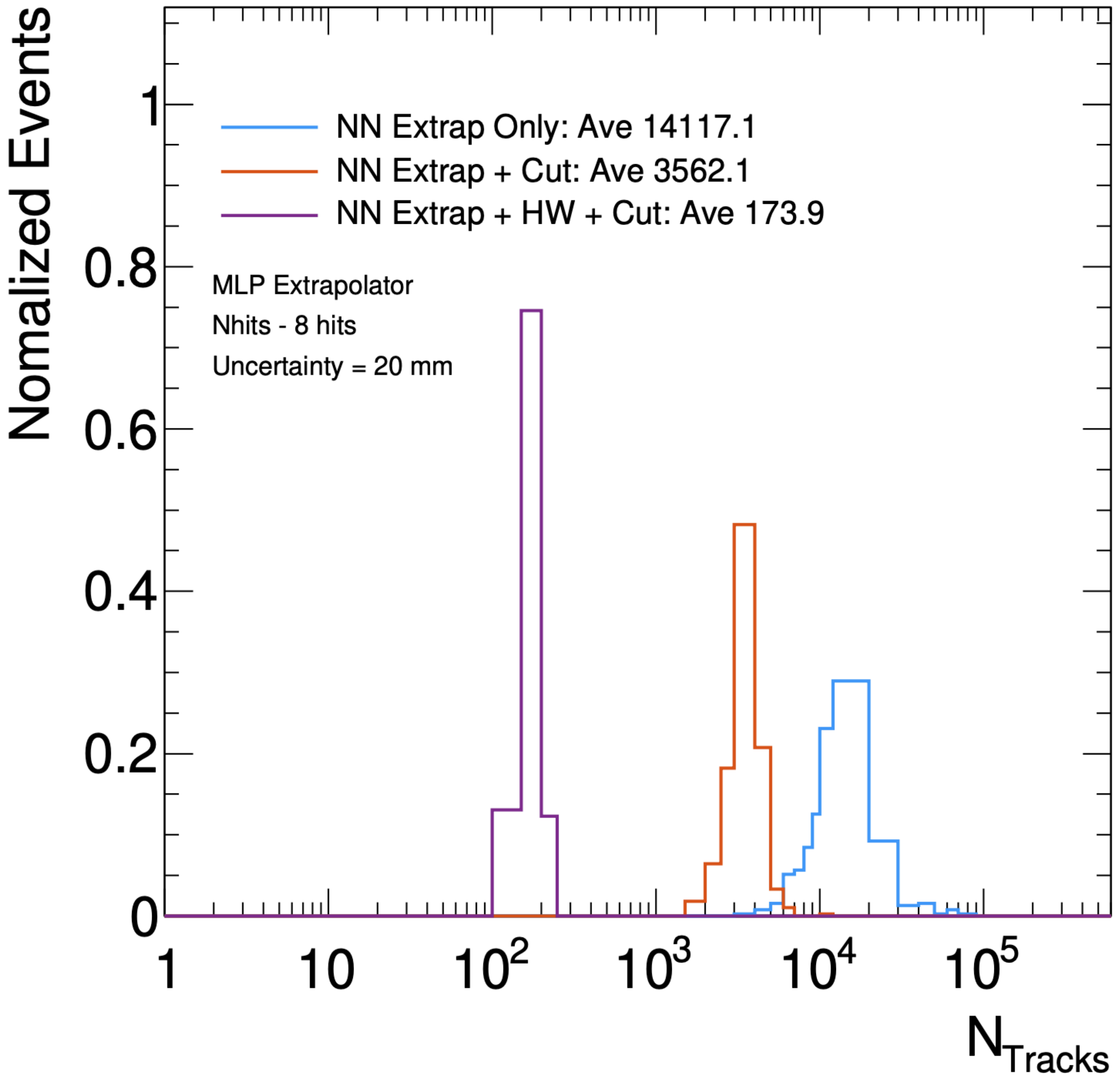
AI/ML Idea: Fake Removal

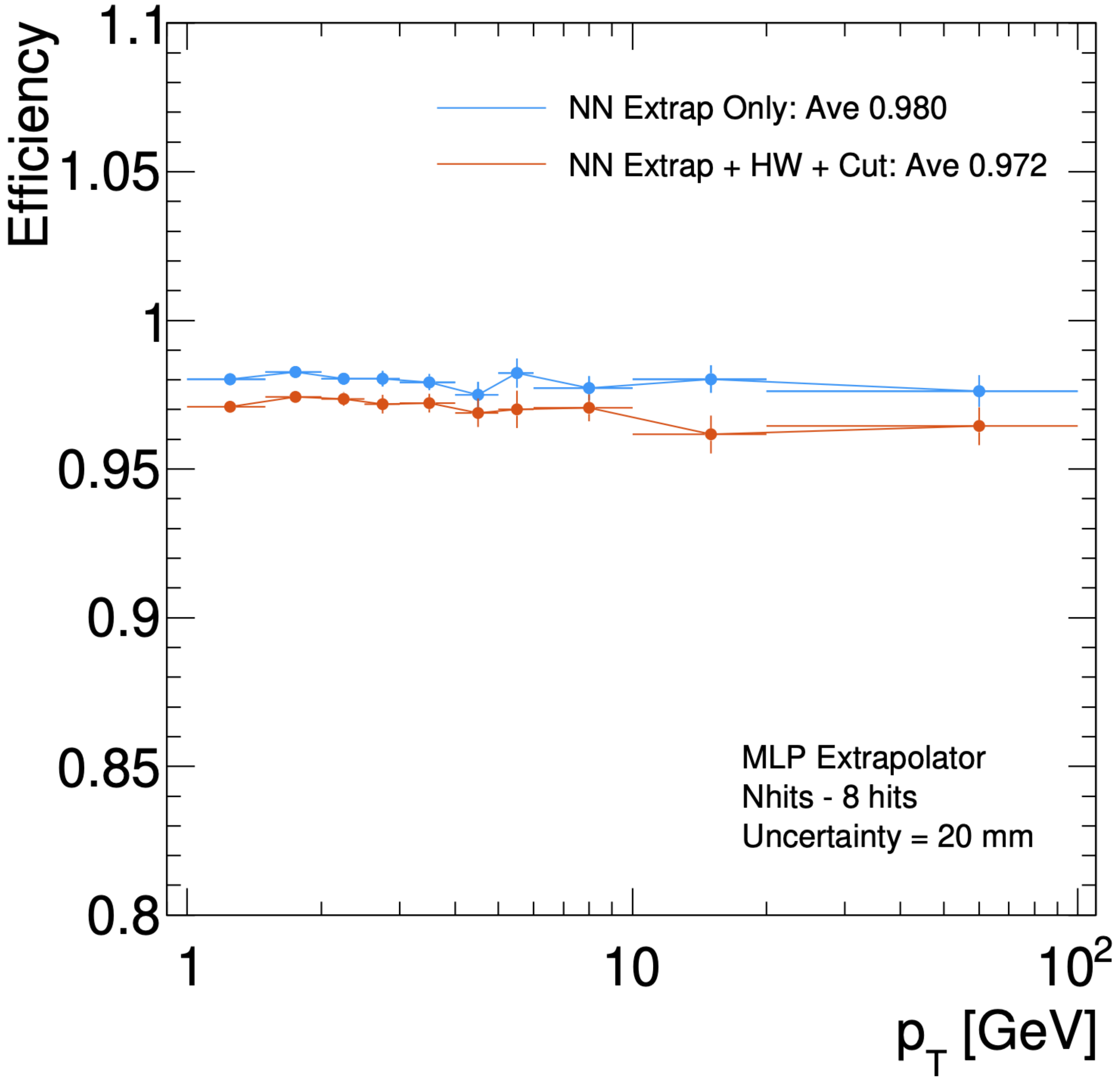
- NN can be used to calculate the figure of merit (FOM)
  - Tracks sharing hits are rejected based this on FOM
- Large reduction in the fake tracks with relatively high efficiency!
  - This AI/ML idea allowed track candidates to fit within the data bandwidth requirements and enable this pipeline for FPGAs

Fake rejection algorithm



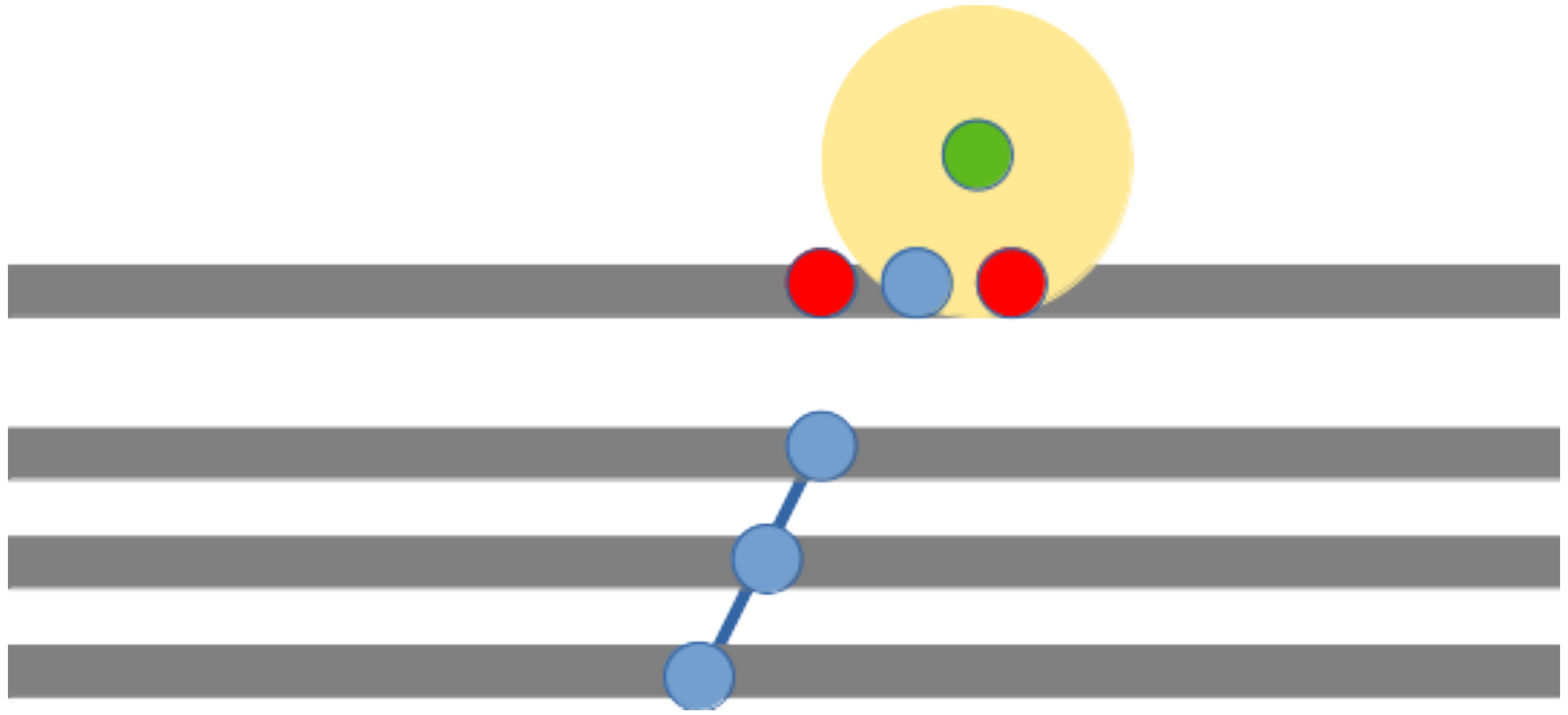
Large reduction in duplicate/fake track for a very small loss of tracking efficiency

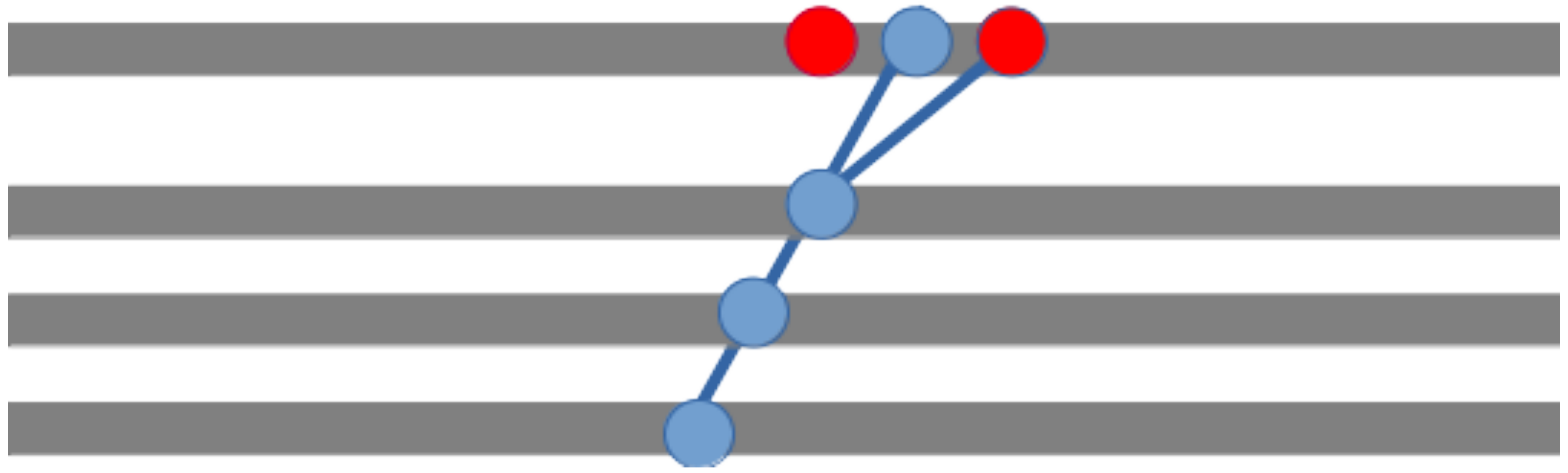




AI/ML Idea: Path Finder

- Kalman filter/extrapolation algorithm is the standard for reconstructing/fitting the track
  - Precision algorithm that requires magnetic field and detector description
- Train a NN to encode this information and predict the trajectory
  - Given a tracklet, can the NN predict the next hit?
- Reduce expensive computation to matrix multiplication that can be accelerated on FPGA/GPUs



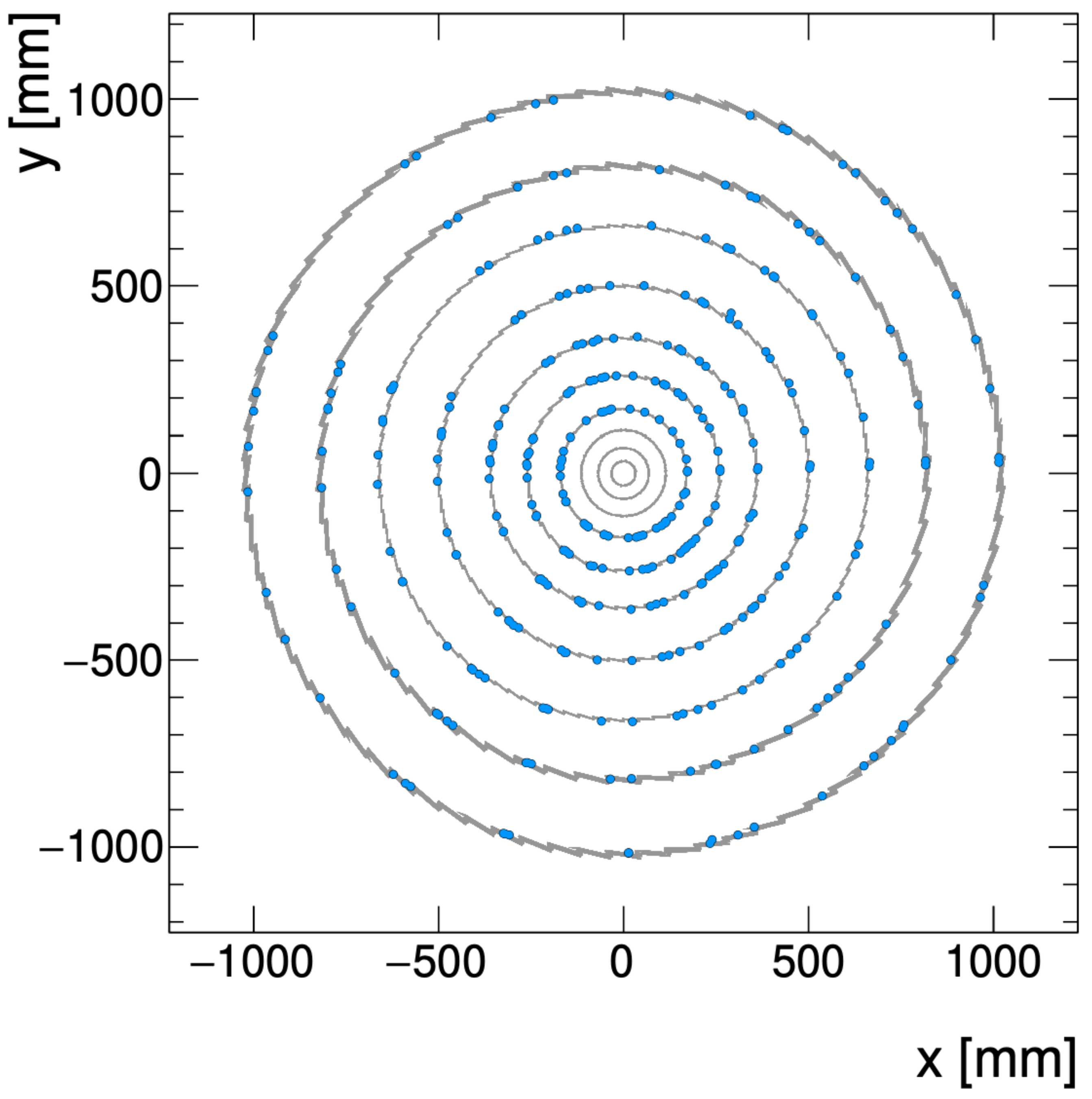


Assume Seeds of three hits are available

1. Input 3 hits into the NN
2. Predict (extrapolate) the location of the 4th hit
3. Look for hits in the detector nearby the predicted location
4. Append all compatible hits to the track seed
5. Repeat until the edge of the detector is reached or no compatible hits are found

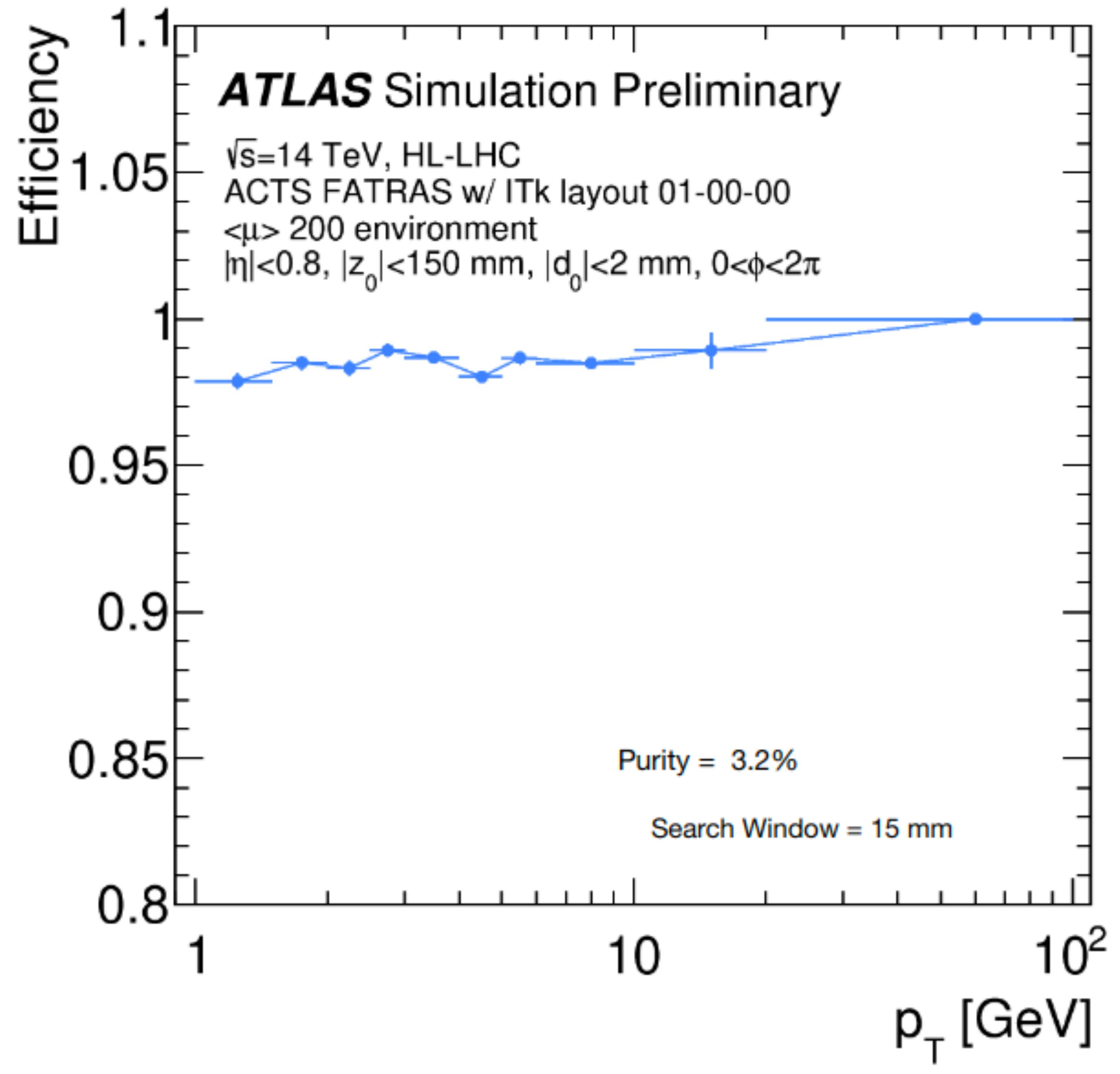
Full Track Reconstruction

- NN is able to **learn the trajectory** of the particle in our complex detector/magnetic field
- Implement the NN in a standalone **track finding algorithm**
  - Comparably high reconstruction efficiency
- Being implemented in Athena with ACTS for large scale testing



NN learns the detector geometry

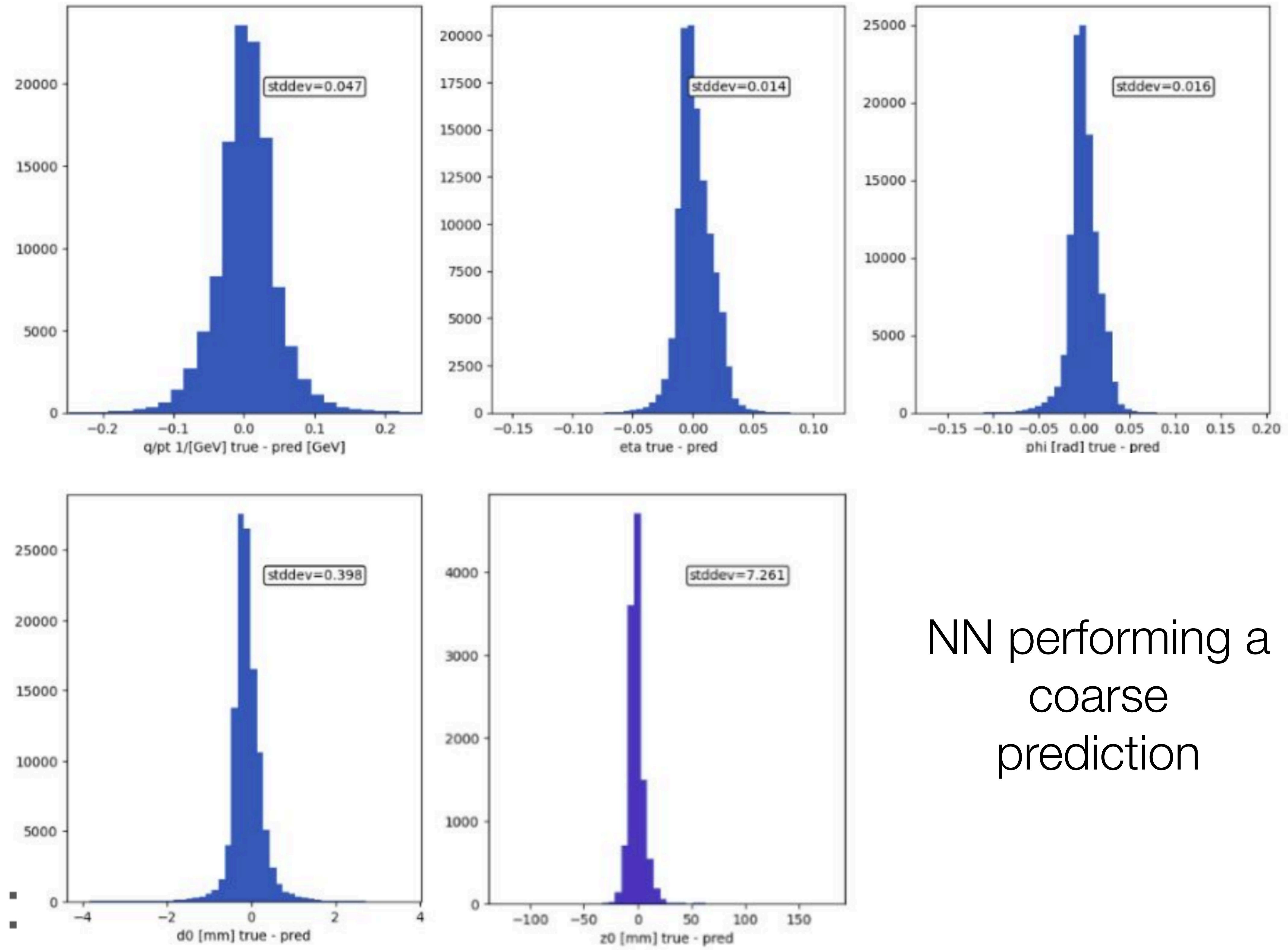
Tracking efficiency in **mu = 200**



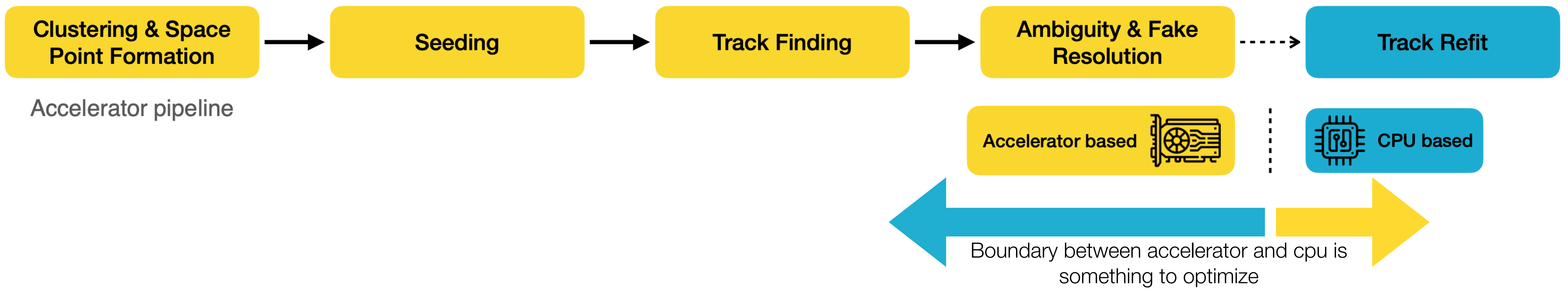
Tracking efficiency in a HL-LHC environment

AI/ML Idea: Coarse Parameter Prediction

- All accelerator based algorithms (including GNNs) envision a CPU based track fit using the **ACTS Kalman Fitter**
  - Identified the need for a simple for a coarse parameter estimation
- Use a vector of x/y/z position coordinates to predict pT/eta/phi/d0/z0 as starting guesses for Kalman Fitter
  - Being integrated into ACTS/Athena for larger scale testing

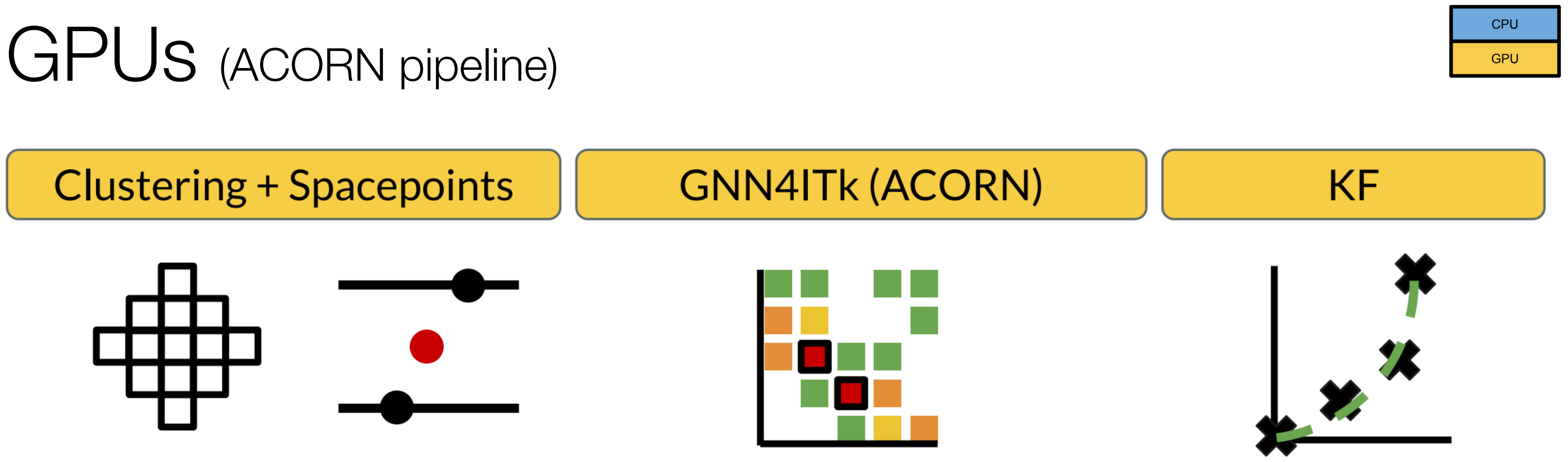


NN performing a coarse prediction

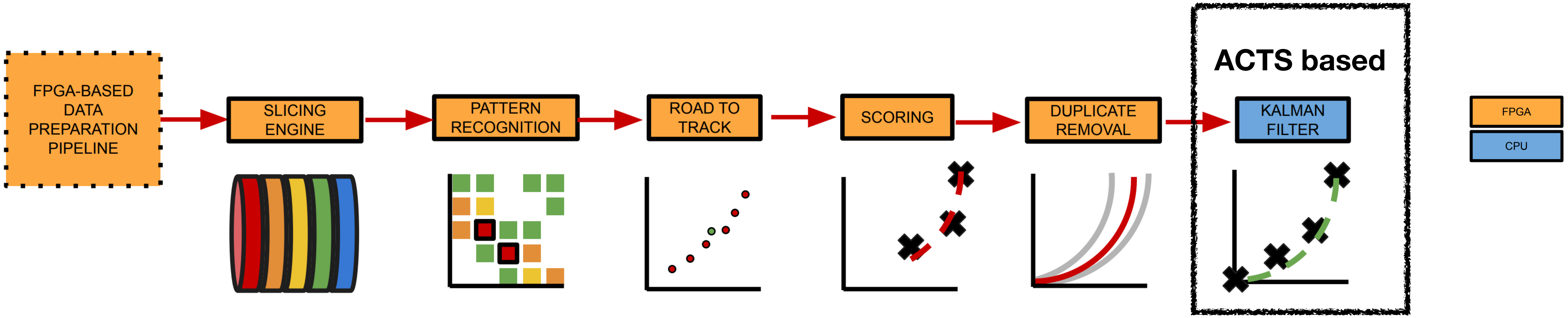


AI/ML Pipelines

- Requires conversion of AI/ML (python based) to C++ based algorithms
  - Significant effort into developing the tools on incorporate AI/ML into pipelines
  - Successfully demonstrated over the two development cycles



FPGAs



ML algorithms for major tracking sub-algorithms

Pathfinder GNNs

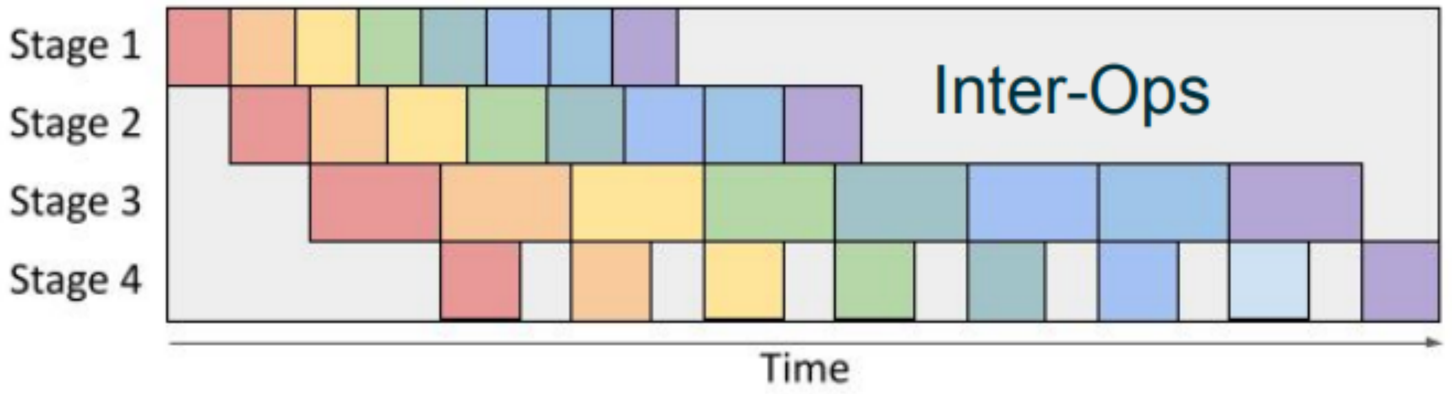
Overlap Resolution Parameter Prediction

ONNX & HLS4ML

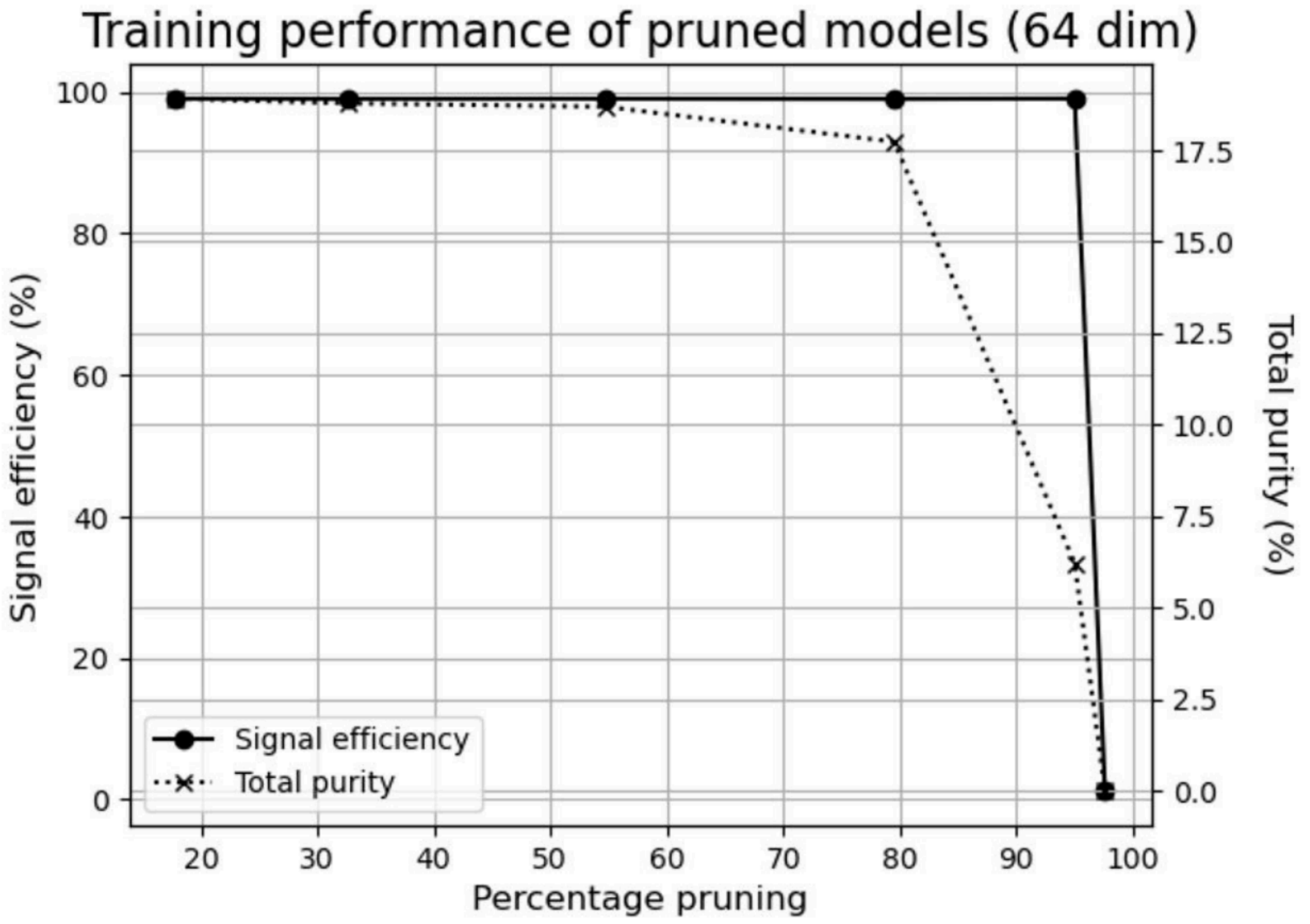
- For CPUs/GPUs, **ONNX** is the goto standard
  - **Leveraging Core software development** for FTAG/Calo/JetEtMiss NN inference
- Established **HLS4ML** as **goto tool** for ATLAS
  - Work performed in first cycle lead to a **common understanding**
  - Understanding limitation, creating implementation, validating inference calls
  - Establishing pruning/quantization strategies for efficient implementation

Link - Athena interface for GPU NN inference

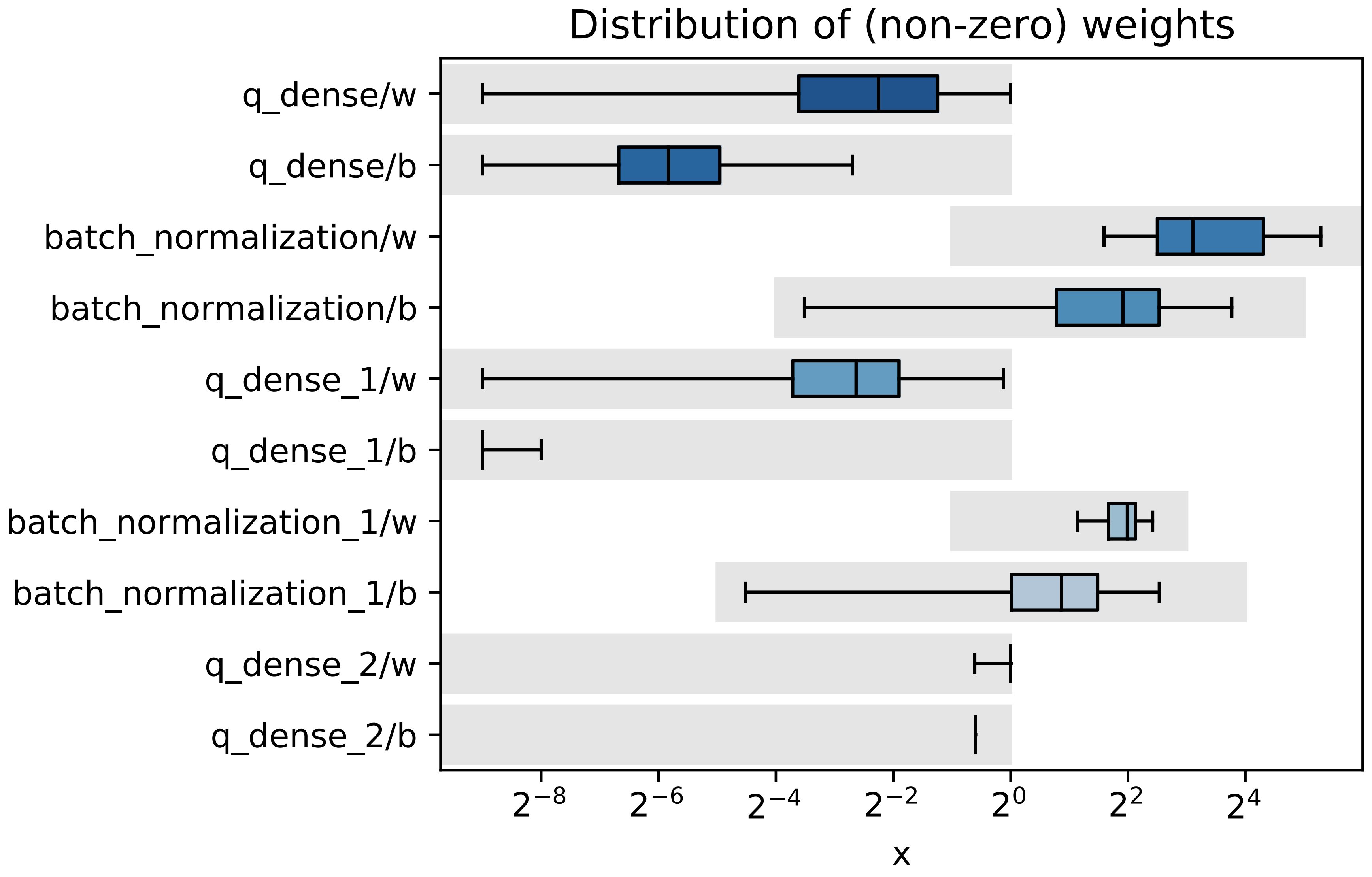
```
def AthExOnnxRuntimeExampleCfg(flags, name="AthOnnxExample", **kwargs):
    acc = ComponentAccumulator()

    model_fname = "/cvmfs/atlas.cern.ch/repo/sw/database/GroupData/dev/MLTest/2020-03-02/MNIST_testModel.onnx"
    kwargs.setdefault("OutputLevel", Constants.DEBUG)

    from AthOnnxConfig.OnnxRuntimeSessionConfig import OnnxRuntimeSessionToolCfg
    kwargs.setdefault("OnnxRuntimeSessionTool", acc.popToolsAndMerge(
        OnnxRuntimeSessionToolCfg(flags,
                                  model_fname,
#                                  execution_provider="CPU",  # optionally override flags.AthOnnx.ExecutionProv
                                  **kwargs)
    ))

    kwargs.setdefault("BatchSize", 3)
    acc.addEventAlgo(CompFactory.AthOnnx.EvaluateModel(name, **kwargs))

    return acc
```



Impact of pruning for GNNs



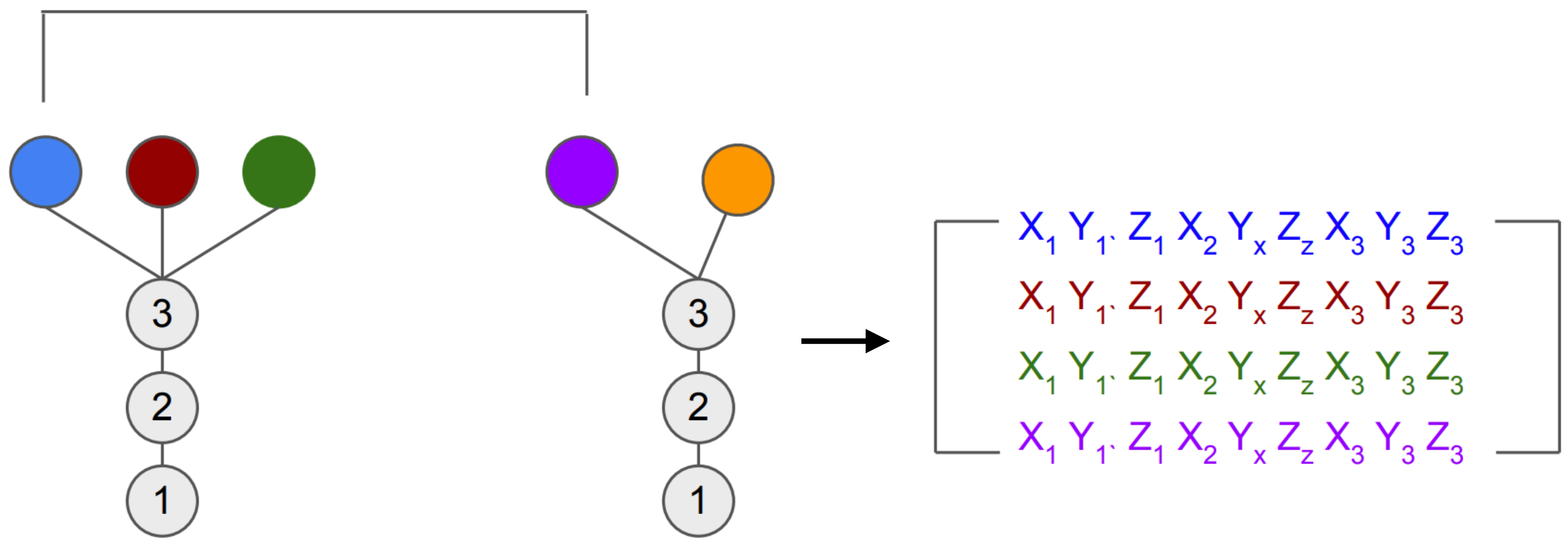
Tools for customizing quantization



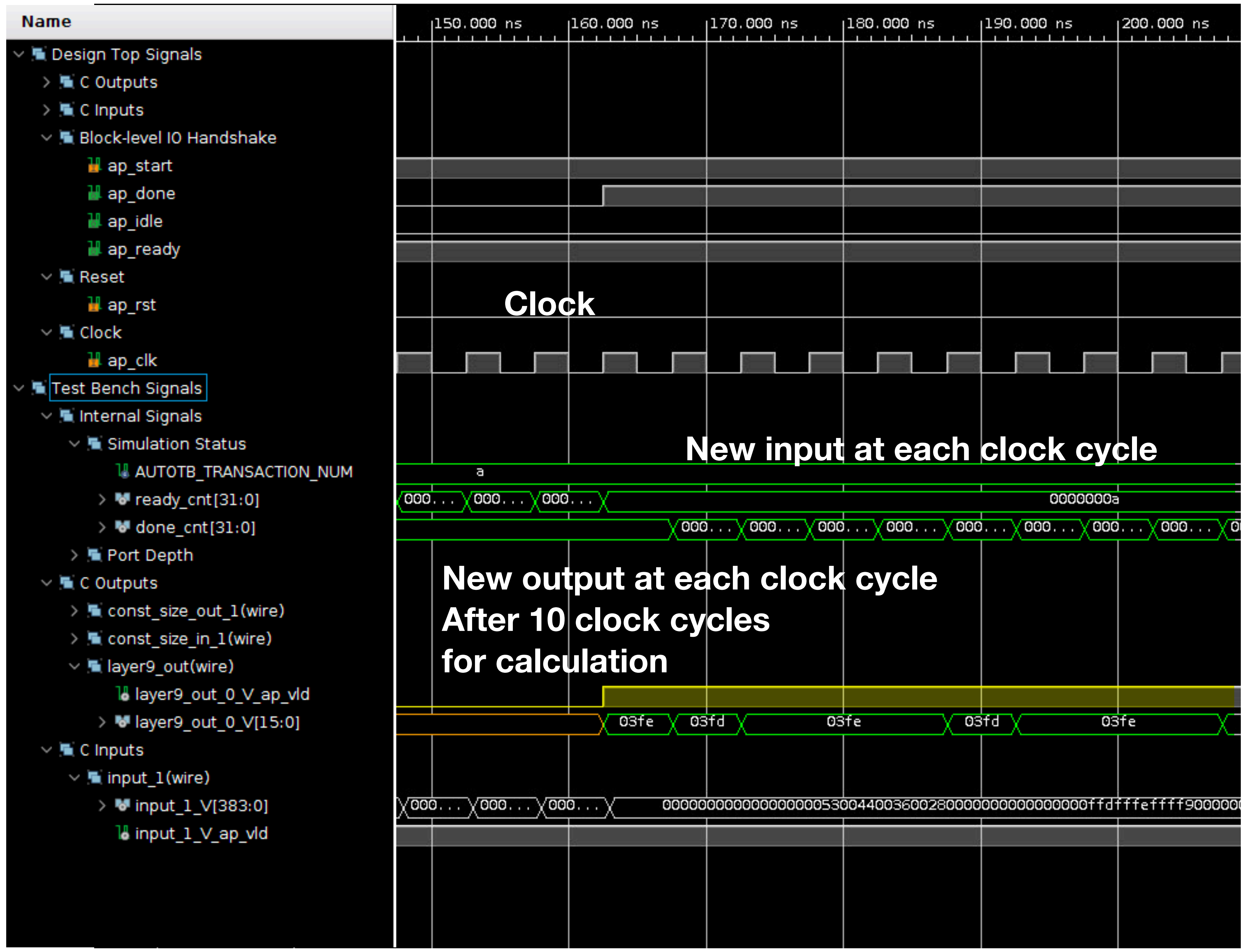
AI/ML on accelerators

- For CPUs/GPUs, single inference calls are very inefficient
  - Studies ongoing to **batch inference calls** around NN
- For FPGAs, the **Vitis kernel flow has been established & validated** for NNs
  - Matrix multiplication is perfectly pipelined
  - For N evaluations, total latency is **O(N + constant), not O(constant*N)**

Optimization of batch evaluation for NN on GPUs/FPGAs



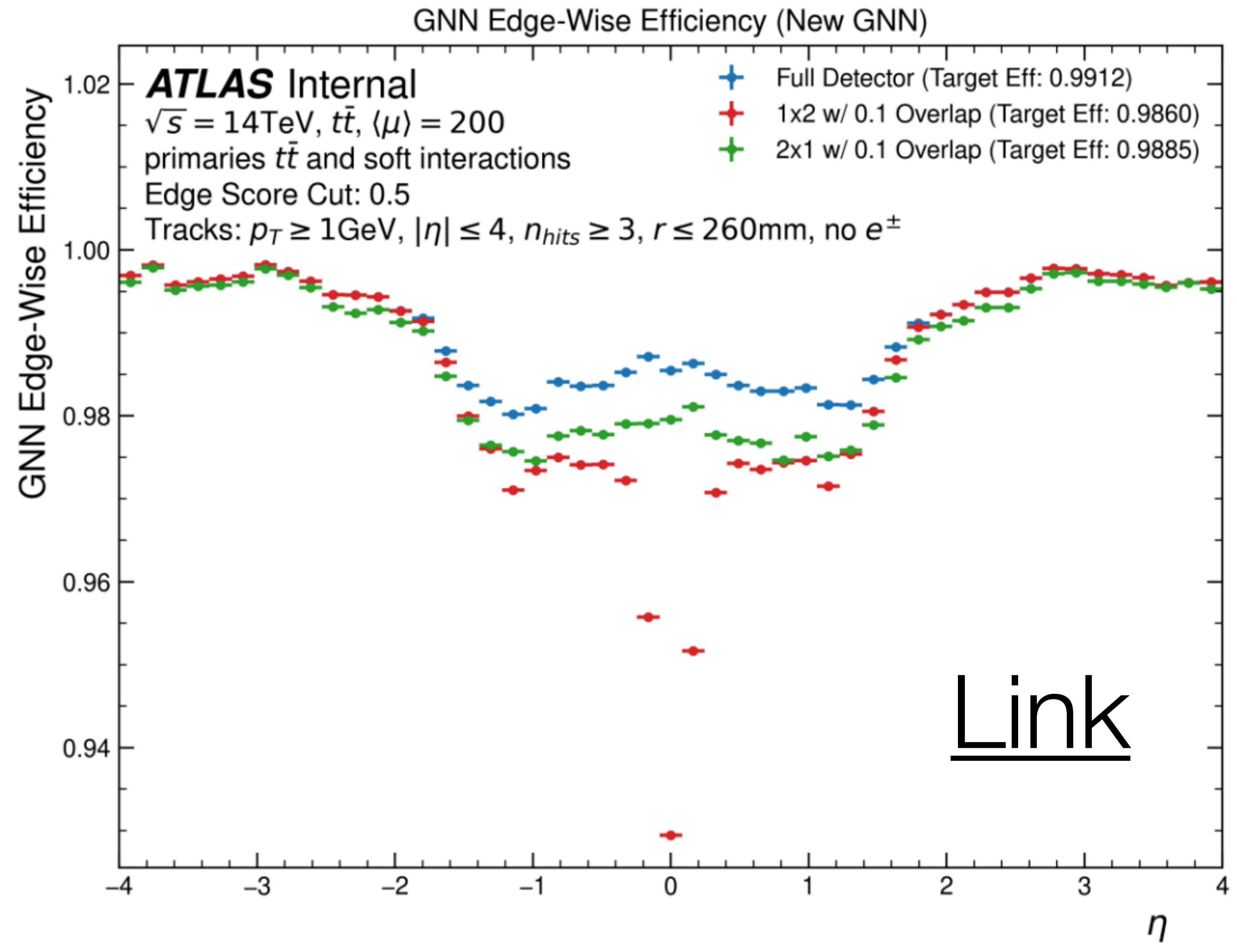
Validation of NN on FPGA



Scaling restriction

- Reasonably priced accelerator cards tend to have small memory, with FPGAs having comparatively smaller compared to GPUs
  - AI/ML algorithms are not immune to the combinatorial growth of tracking
- GNN Graph building requires huge memory
  - Work ongoing to fit this algorithms on FPGA though segmenting the detector in eta/phi slices
  - Some degradation in performance, retraining recovers the losses

GNNs Edge efficiency



Impact of detector segmentation on GNNs memory requirement

Regions in Phi-Eta	1x1 (Full Detector)	1x2 w/ 0.1 Overlap	2x1 w/ 0.1 Overlap	2x2 w/ 0.1x0.1 Overlap	4x4 w/ 0.1x0.1 Overlap	8x8 w/ 0.1x0.1 Overlap
Num Graphs	1000	2000	2000	4000	16000	64000
Avg Nodes	~310k	~156k	~158k	~80k	~22k	~7k
Avg Edges	~1.9m	~929k	~951k	~475k	~129k	~35k
Avg Size (MB)	~108 MB	~54 MB	~55 MB	~28 MB	~8 MB	~2 MB

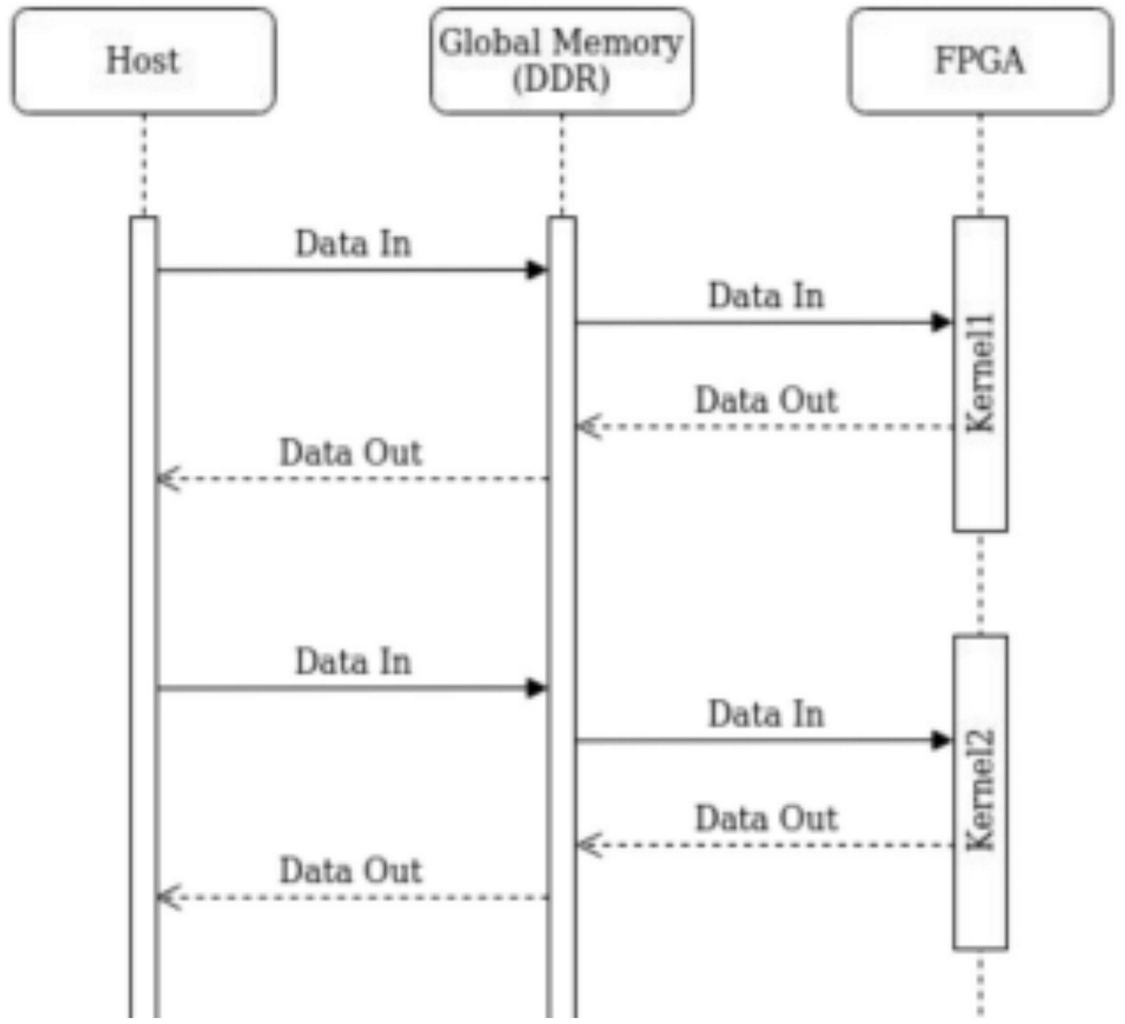
Integration in ACTS/Athena

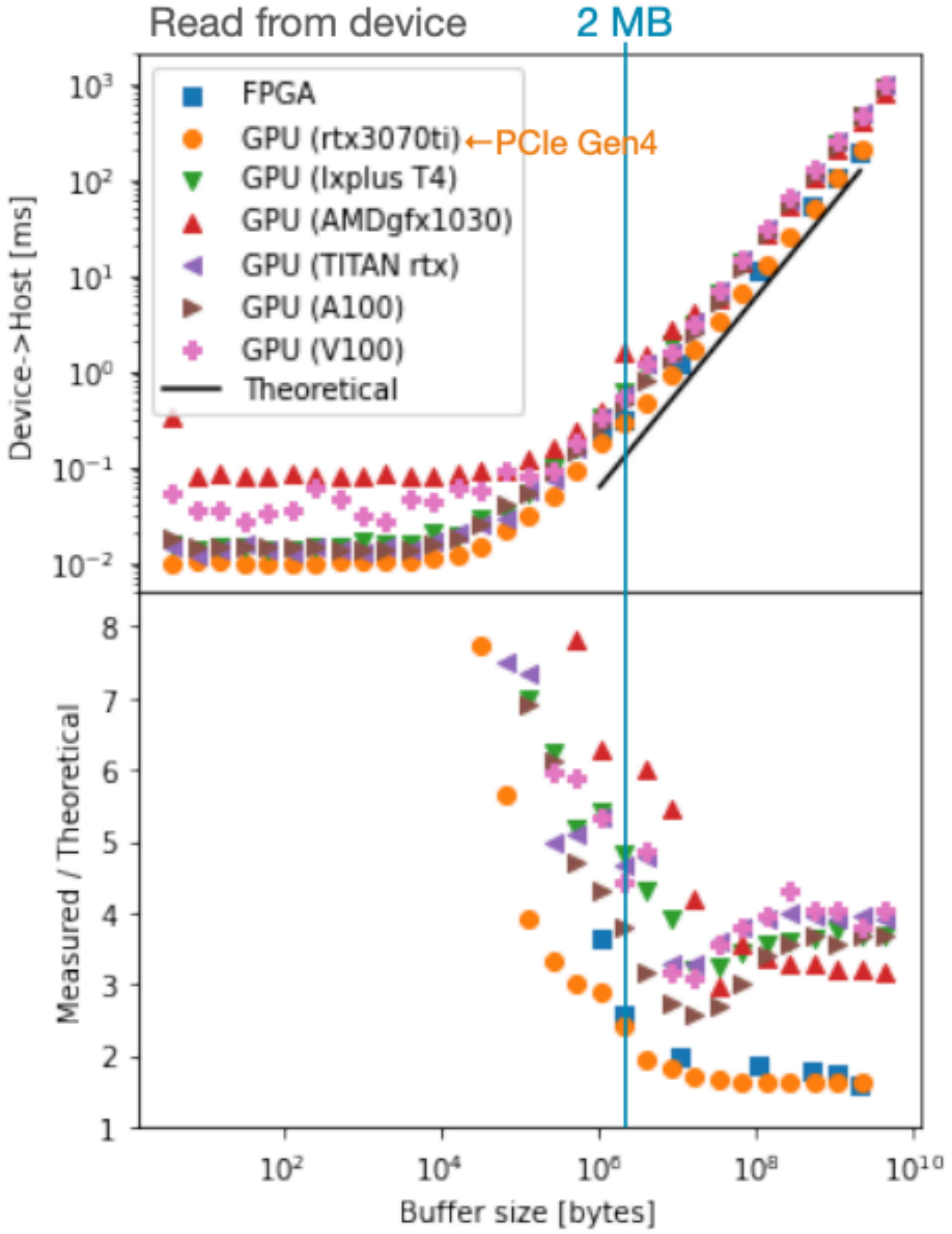
- Pipelines are required to start/end in Athena
  - All algorithms (including AI/ML) need 'plumbing' to make this happen
- Accelerators require complex data transfer and EDM management
  - GPUs/FPGAs communication have been established and validated in Athena
  - Spacepoint FPGA kernel has successfully been integrated with Athena
  - Working to establish quantization requirements - required feedback for NN training
- Interface to ACTS based KF fitter has established and merged into Athena
  - Developed this interface in collaboration with Tracking CP/ACTS developers
  - Provides the conversion to xAOD Track EDM objects for analysis and connection to common monitoring tools

KF interface
for EF pipelines

```
/// @brief small non-persistent data class to wrap the output
/// of the EF-tracking development pattern finding placeholder
You, 6 days ago | 1 author (You)
namespace ActsTrk{
  You, 6 days ago | 1 author (You)
  struct ActsEFProtoTrack{
    /// set of measurements assigned to the same pattern
    std::vector<ActsTrk::ATLASUncalibSourceLink> measurements = {};
    /// estimate of initial track parameters for this pattern
    std::unique_ptr<Acts::BoundTrackParameters> parameters = nullptr;
  };
};
```

Data Transfer pattern and latency





Conclusions

- AI/ML algorithms are integral components of the EF tracking pipelines
  - Many pioneering efforts, but also collaborating with and adapting other ongoing efforts
  - Both novel new solution and enabling of classical algorithms for accelerators
- First development cycle identified various promising AI/ML algorithms
  - Focus to integrate these into the large scale simulation and hardware chains
  - NNs are never the full chain and require plumbing
  - Once connections are defined, well defined opportunities for newer AI/ML ideas to be incorporated