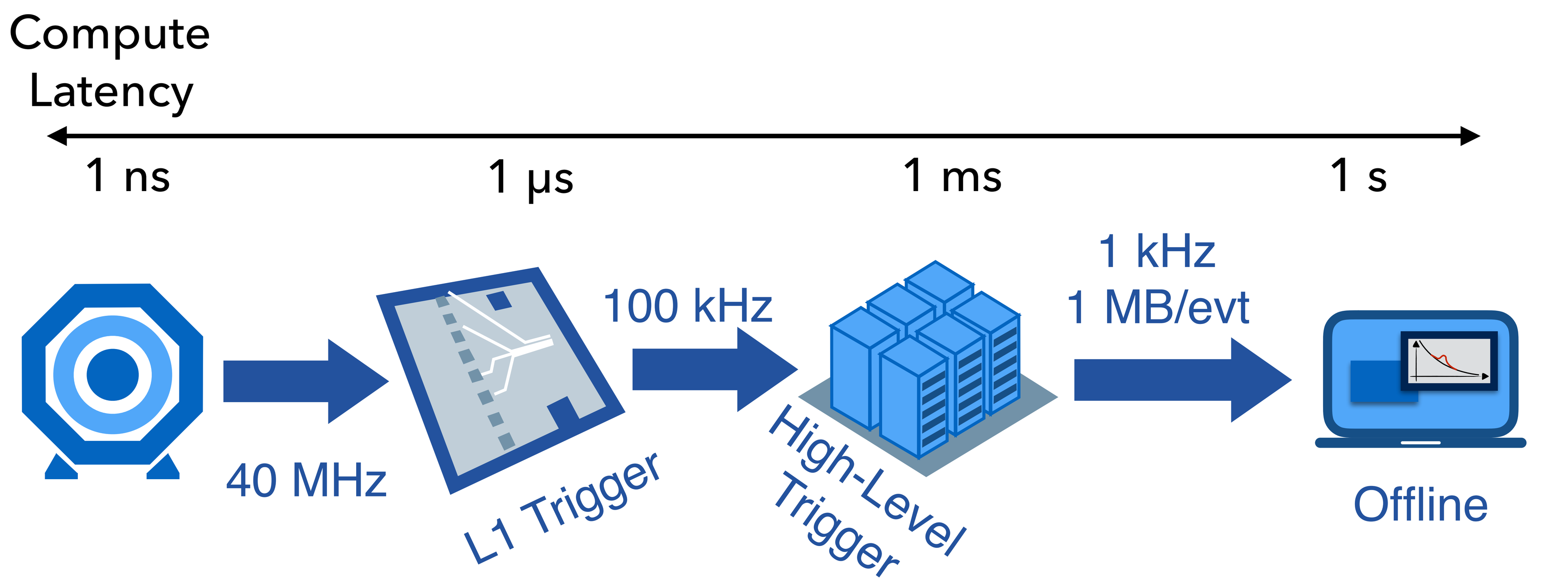
Nanosecond inference for a graph neural network based $\tau \to 3\mu$ detection at CMS

Hyeon-Seo Yun[1], Mia Liu[1], hls4ml project [2]

[1]Purdue University [2]https://github.com/fastmachinelearning/hls4ml

Need for low-latency in L1 Trigger



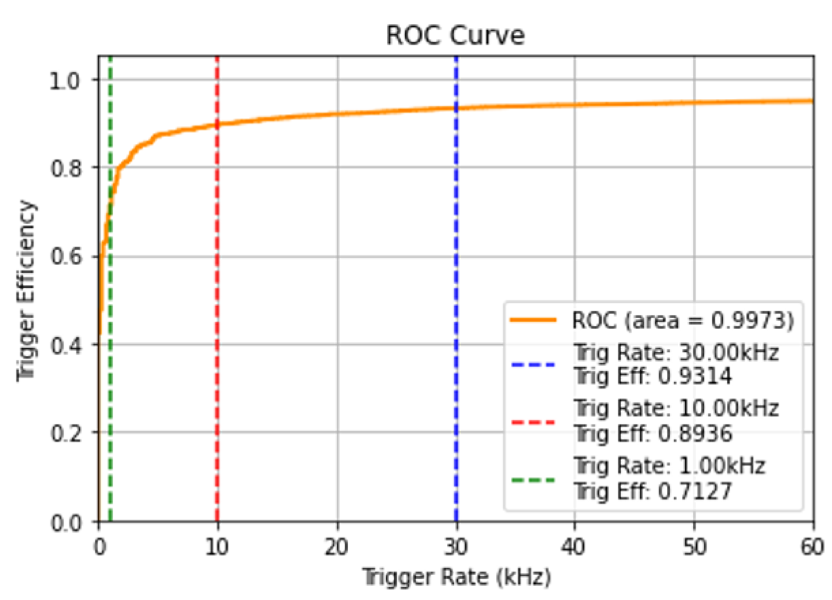
- Role of Trigger: Detect events that contains interesting data (decays of τs into 3 μs in our case).
- We keep those events, and discard the uninteresting ones.
- Machine Learning (ML) use case in particle physics: L1 Trigger, the first stage of real-time data processing and filtering
- Due to low latency requirements, field programmable gate arrays (FPGAs) are used in Triggers
- CERN CMS L1 Trigger requirements: high input data rates ($\approx$ **40** TB/s) into smaller output data rates ($\approx$ **0.75** TB/s), with fixed algorithm latency in the order of tens of nanoseconds.
- High Level Synthesis (HLS) Compiler named `hls4ml` is used to rapidly prototype ML models in FPGAs

Quest for Observing $\tau \to 3\mu$

- Decays of τ into 3 μ is predicted to be extremely rare in the Standard Model of particles physics.
- New physical phenomena could enhance the probability significantly.
- Observing these decays at the LHC could be a sign of the existence of physics beyond the standard model

Usage of Graph Neural Networks (GNN) in Filtering Task

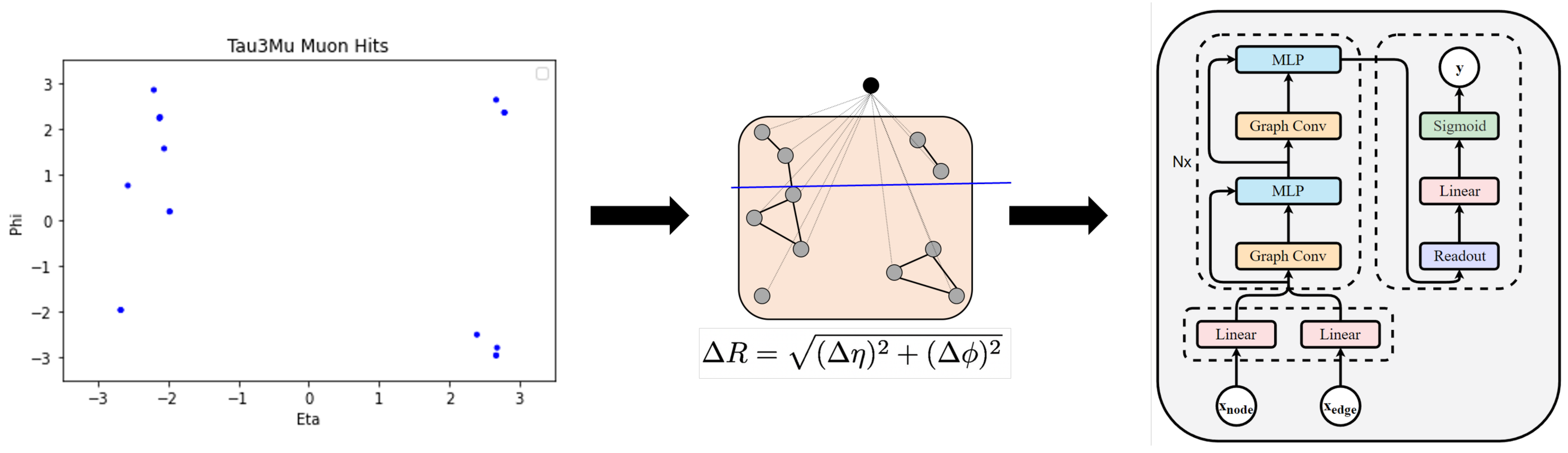
- Proton-proton collisions give data that can easily be formed as a graph.
- Graphs can easily connect the signals particles leave while traversing the CMS detector.
- Ideal for GNN, a Neural Network (NN) type that is optimized for graph based data (permutation invariant data).
- Good for μ detection from τ decay, as they have low transverse momentum (p_t) signature that makes it difficult for conventional ML Trigger algorithms to perform well.
- Task: Classification of events as Signal or Background in real-time using GNN.
- Performance quantified in a receiver operating characteristic (ROC) curve of signal efficiency versus Trigger Rate (kHz).



Summary

- `hls4ml`: compiler based on HLS for porting fully-connected NNs to an FPGA from conventional training frameworks such as `Keras` and `PyTorch`
- Focus on real-time event reconstruction and filtering at the LHC in FPGAs, with many other applications to real-time detector systems in the physical sciences
- Implemented a GCN model in HLS code, but too much resources
- In the progress of implementing QAT to optimize GNN for low bit, low latency inference

GNN Design



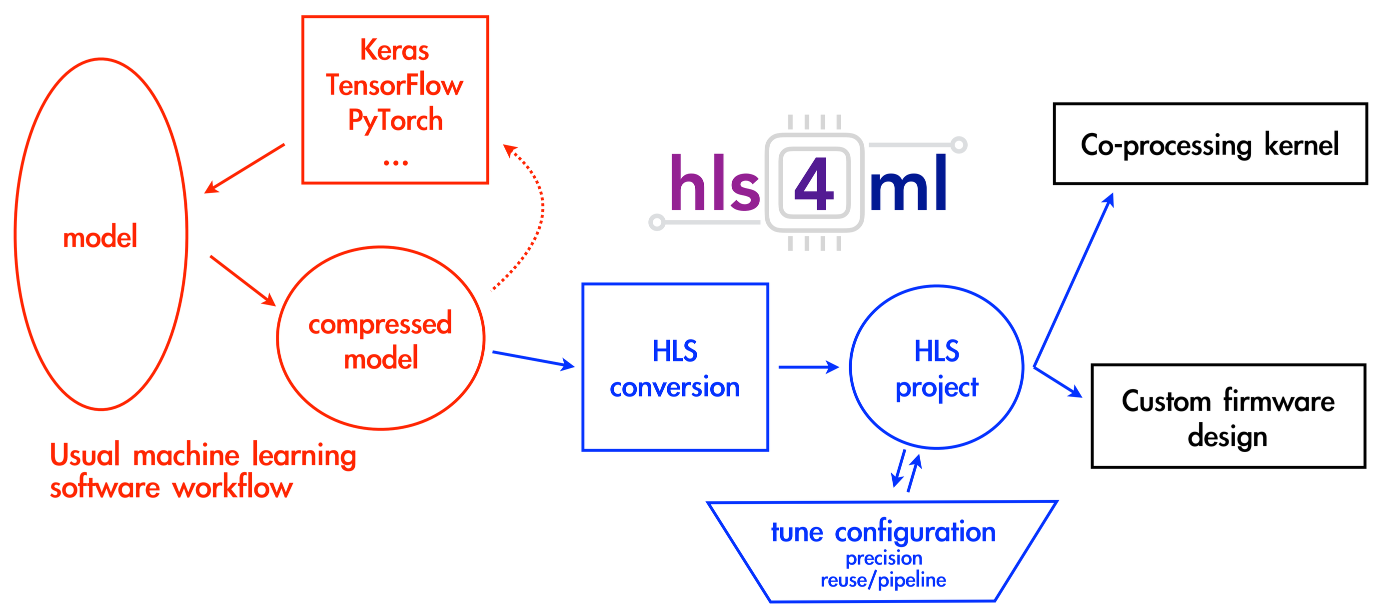
Graph Building

- **Nodes Designation:** hits from Station 1 endcap detectors from CMS
- **Edge Designation:** edge between two nodes if $dR = \sqrt{\Delta\eta^2 + \Delta\phi^2} < 1$
- **Node features:** z and η coordinate, and bend angle
- **Edge features:** Δz, $\Delta\eta$, $\Delta\phi$ and bend angle difference.
- Bend Angle: Angular difference of particles between hit signatures as they enter and exit the detector.

GNN Architecture

- **Type:** 8 Graph Convolution (GCN) Layers with residual connections in-between the layers.
- **MLP block:** 128 $\to$ 256 $\to$ 128 node fully-connected layers

FPGA Implementation via hls4ml



- `hls4ml` is a pythonic compiler that translates `Keras` or `PyTorch` NN models into High Level Synthesis (HLS) code.
- HLS ports said NN models to FPGAs for L1 Trigger.
- `hls4ml` supports normal fully-connected NNs, but little GNN support.
- We have custom hard-coded our GNN model into `hls4ml`, but FPGA resources overloaded. Our GNN model needs to be more lightweight

Quantization Aware Training (QAT)

Quantization of GNN is necessary to meet the FPGA latency constraints

- Typical NNs have their trainable parameters represented in a 32-bit system.
- Quantization decreases the 32-bit representation to a more manageable number, typically 4 or 8 bits.
- `hls4ml` uses arbituary precision fixed (ap-fixed) quantization method for faster inference.
- Simply quantizing our pre-trained GNN greatly decreases performance.
- QAT is a specialized method of training NNs that maintains performance with quantization.
- Using `Brevitas`, we demonstrated QAT on fully-connected NN with non $\tau \to 3\mu$ data, for benchmarking.

To demonstrate QAT, we use Higgs Kaggle data as a benchmark (https://www.kaggle.com/competitions/higgs-boson/data)