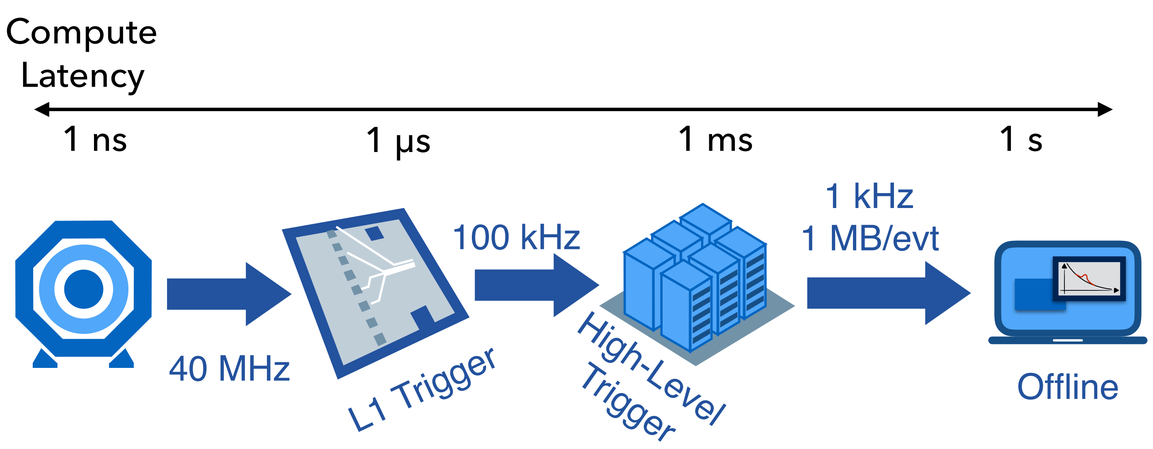
Low-latency L1 Trigger Inference using GNN for $\tau \longrightarrow 3\mu$

Hyeon-Seo Yun[1], Mia Liu[1]

[1]Purdue University [2]Fermilab [3]MIT [4]CERN [5]UW [6]UIC [7]HawkEye360

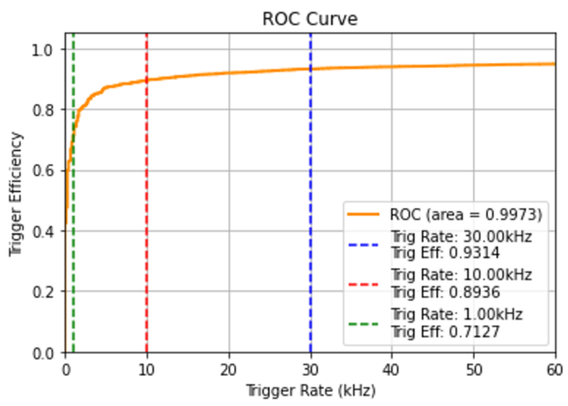
Need for low-latency in L1 Trigger



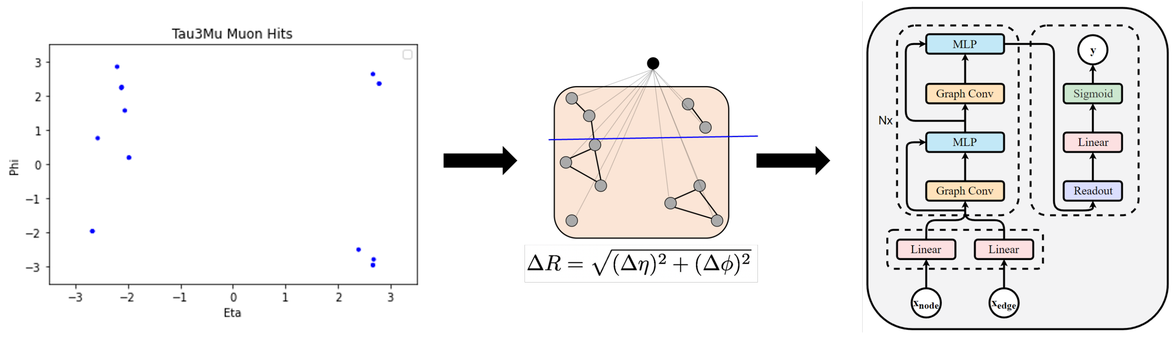
- Machine learning (ML) use case in particle physics: L1 Trigger, thge first stage of real-time data processing and filtering, using field programmable gate arrays (FPGAs)
- CERN CMS L1 Trigger requirements: high input data rates $>$ **100** TB/s, $<$ **100**ns fixed algorithm latency, constrained FPGA resources
- Compiler based on high-level synthesis (HLS) called `hls4ml` to rapidly prototype ML models in FPGAs

Usage of Graph Neural Networks (GNN) in Filtering Task

- Task: differentiate showers (or *jets*) produced in decays of heavy standard model particles (W and Z bosons and top quarks), from backgrounds consisting mainly of light quark- (u, d, c, s, b) and gluon-initiated jets
- GNN
- Performance quantified in a receiver operating characteristic (ROC) curve of signal efficiency versus misidentification rate for quark, gluon, W boson, Z boson, and top quark jets



Design



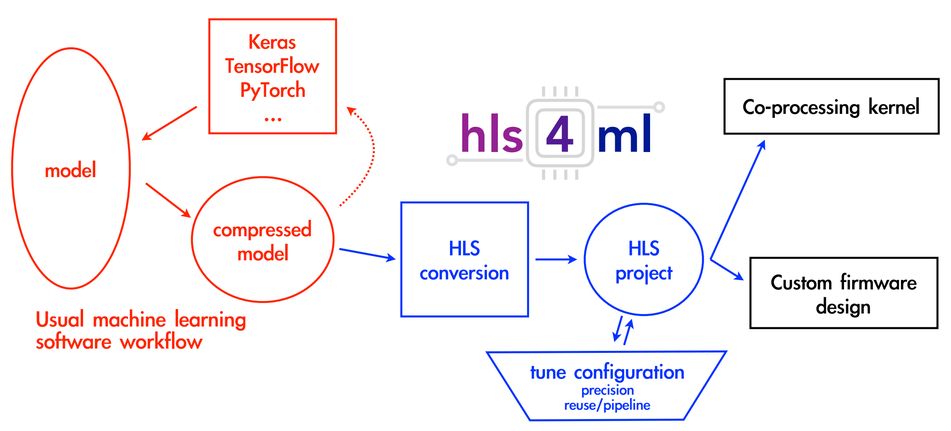
Explore the FPGA design space through

- **compression**, the three-hidden-layer model with 70% of the parameters removed using iterative retraining with L_1 regularization and magnitude-based pruning
- **quantization**, the precision of the inputs, weights, and biases
- **parallelization**, the number of times a given multiplier is used for a layer computation, quantified by a *reuse factor*

With these handles, monitor

- **resources**: digital signal processors (DSPs), block random access memory (BRAM), flip-flops (FFs), and lookup tables (LUTs)
- **latency**: time it takes to compute the full network
- **initiation interval (II)**: time before a new set of inputs can be accepted

FPGA Implementation via hls4ml



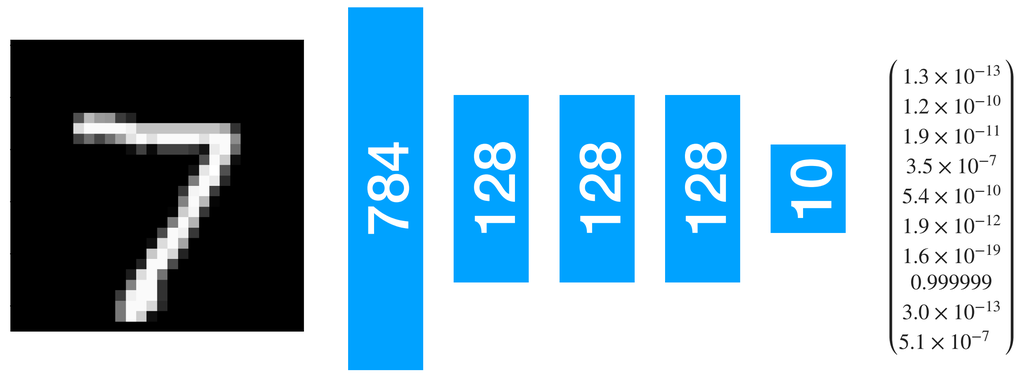
- First evaluate NN with fixed point precision: `<16,6>` fixed-point precision reproduces the ROC curve performance
- DSP usage in the compressed 3-hidden-layer model increases as a function of the network precision and decreases for larger reuse factors
- Latency increases from 10 to 35 clock cycles (50 to 175 ns) for larger reuse factors
- Results based on Xilinx Kintex Ultrascale FPGA part number `xcku115-flvb2104-2-i`, 200 MHz clock frequency, Vivado HLS 2017.2

Quantization Aware Training

To enhance the flexibility of `hls4ml`, several new developments include

- extension to allow for significantly larger dense networks in terms of the number of neurons per layer
- inclusion of zero-suppression for weights stored in on-chip memory or BRAM, reducing the use of on-chip logic registers
- addition of binary and ternary matrix multiplication

To demonstrate the new developments, several versions of a large dense network to classify handwritten MNIST digits are benchmarked



Model	II	Accuracy	Latency	DSP	BRAM	FF	LUT
MNIST dense	128	0.97	2.6 μs	21%	45%	12%	33%
MNIST binary dense	128	0.93	2.6 μs	0%	33%	7%	39%
MNIST ternary dense	128	0.95	2.6 μs	0%	33%	7%	40%
MNIST dense, 95% pruned	128	0.96	2.8 μs	1%	34%	13%	164%
MNIST dense	4096	0.97	68.1 μs	1%	66%	27%	83%
MNIST dense, 95% pruned	4096	0.96	82.1 μs	0%	34%	9%	25%

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 dense 3-hidden-layer NN in a Xilinx Kintex Ultrascale using roughly 10% of the available DSPs and latency of approximately 75–150 ns with a clock frequency of 200 MHz
- Extend the capabilities and flexibility of `hls4ml` to allow larger NN architectures for applications with latency constraints of approximately 1–100 μs