End-to-end codesign of Hessian-aware Quantized neural networks for FPGAs

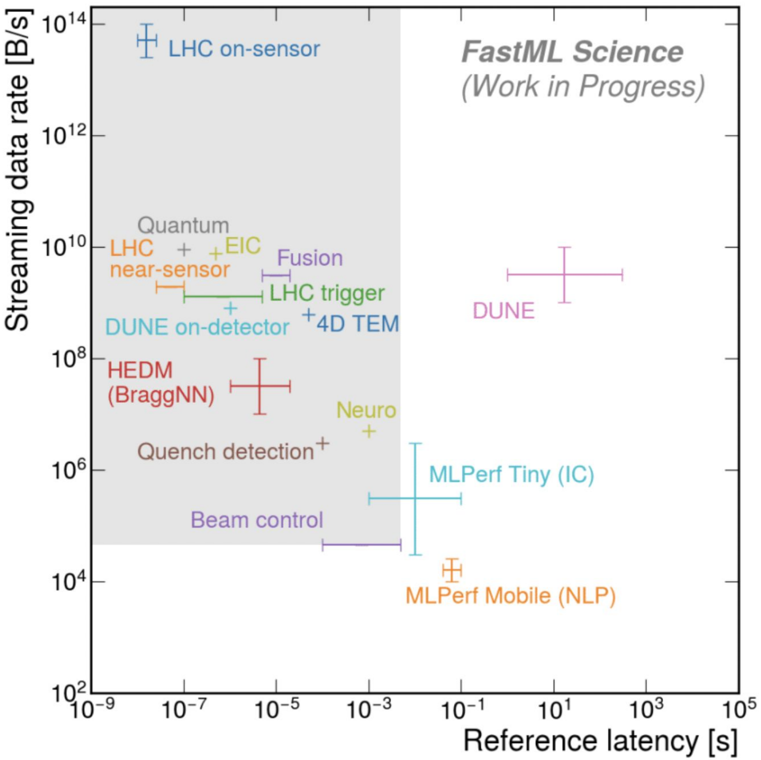
Javier Campos, Zhen Dong, Javier Duarte, Amir Gholami, Michael W. Mahoney, Jovan Mitrevski, Nhan Tran, Vladimir Loncar

24th IEEE Real-Time Conference

24 April 2024

Scientific Challenges

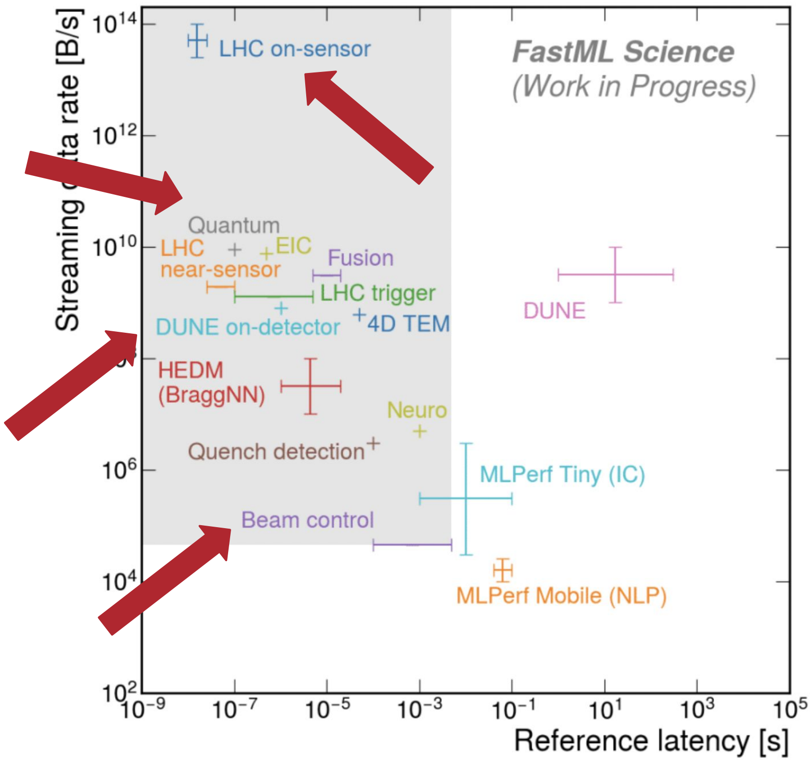
- Multiple scientific applications using ML/DL
- Timing and throughput constraints are heavily dependent on scientific domain, experiment, and computational resources
- Scientific applications:
  - HEP
  - Quantum Computing
  - Nuclear
  - Material Science
- Industry applications:
  - IoT
  - Manufacturing



Fermilab

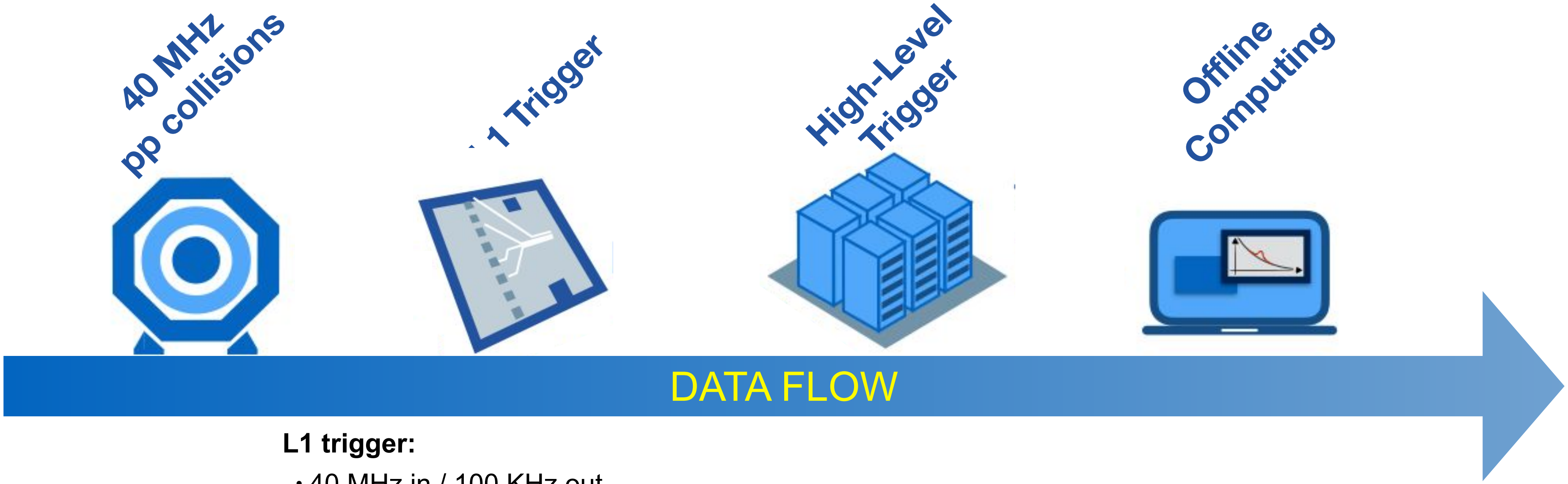
Scientific Challenges: @FNAL

- Multiple scientific applications using ML/DL
- Timing and throughput constraints are heavily dependent on scientific domain, experiment, and computational resources
- Scientific applications:
  - HEP
  - Quantum Computing
  - Nuclear
  - Material Science
- Industry applications:
  - IoT
  - Manufacturing



Fermilab

LHC Experiment Data Flow

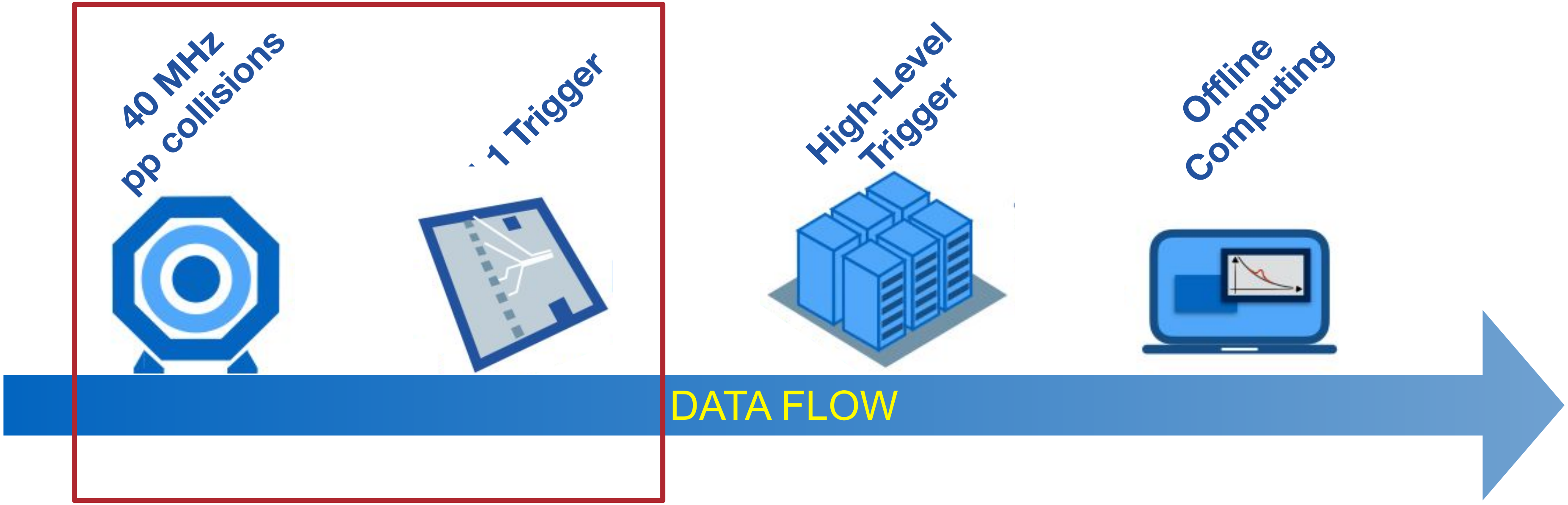


L1 trigger:

- 40 MHz in / 100 KHz out
- Process 100s TB/s
- Trigger decision to be made in ≈ **10 μs**
- Coarse local reconstruction
- FPGAs / Hardware implemented

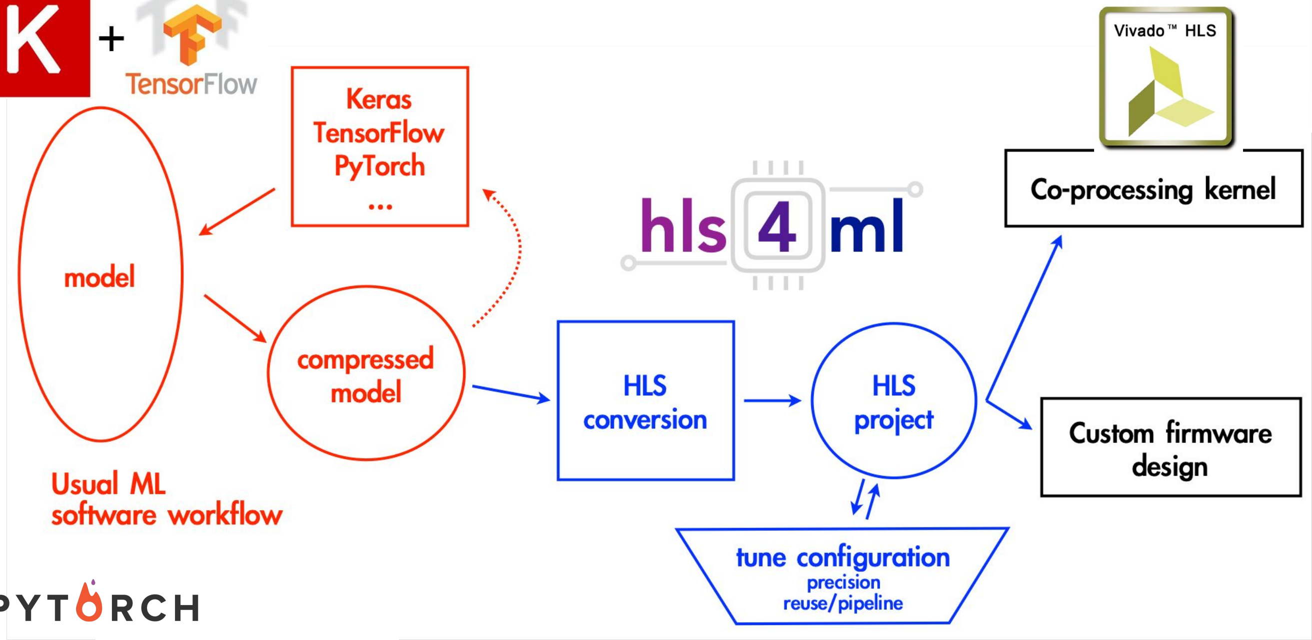
Fermilab

LHC Experiment Data Flow



ML in trigger and sensor applications must be implemented in FPGAs or custom ASICs! Must be robust to noise and radiation and meet high throughput low latency requirements.

high level synthesis for machine learning

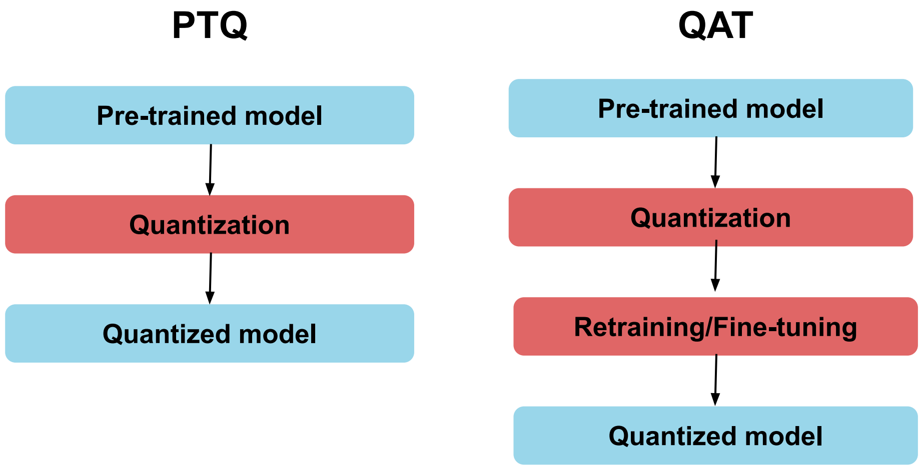


Mentor® Catapult
Coming Soon

https://fastmachinelearning.org/hls4ml/

Quantized Neural Networks

- Quantization is one of the most effective techniques to reduce latency, hardware area, and energy consumption in NNs
  Weights are represented in lower precision, most commonly as fixed-point
- Typically comes in two flavors: PTQ & QAT



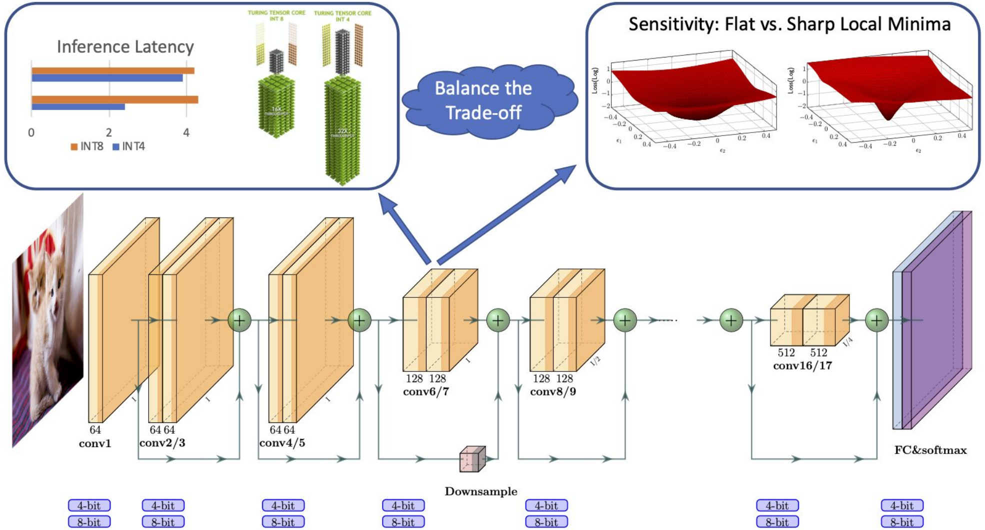
In PTQ (left), pre-trained model is quantized, then calibrated to determine scaling factors and clipping range.

In QAT (right), pre-trained model is quantized then weights are fine tuned using training data to recover accuracy.

Hessian-AWare Quantization (HAWQ)

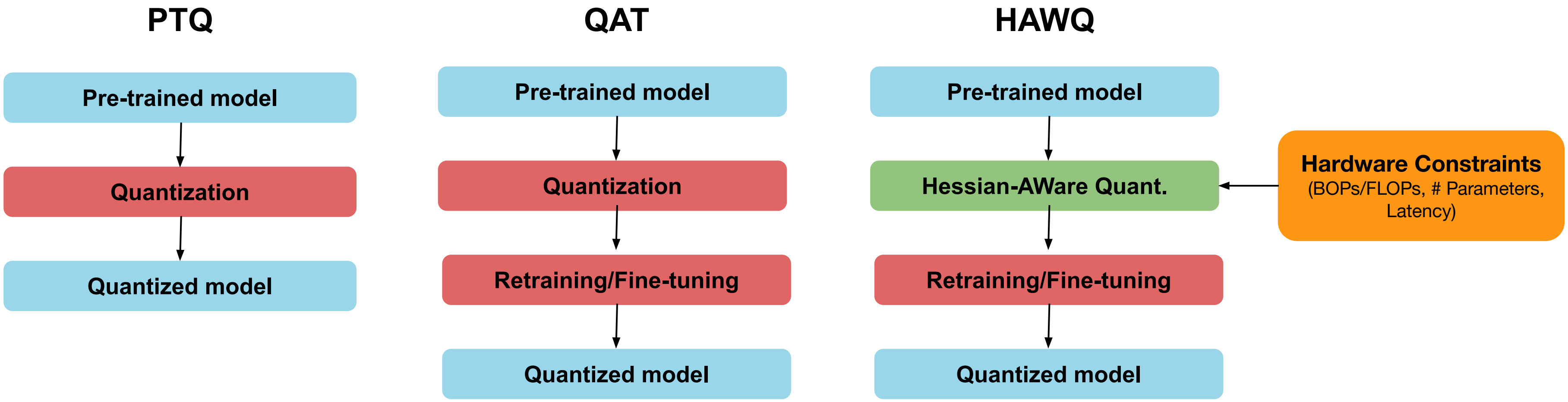
arXiv:2011.10680

- Accuracy degradation is significant for ultra-low precision
- Mixed-precision quantization addresses this
  Sensitive layers are kept at higher precision than less sensitive layers
- **Problem**: search space is exponential to number of layers in the model



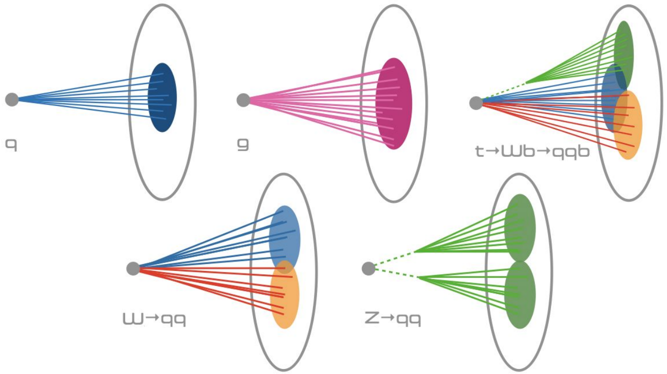
Hessian-AWare Quantization (HAWQ)

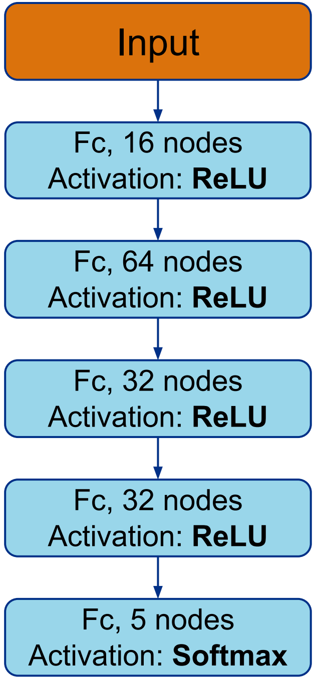
- HAWQ: An advanced quantization library written for PyTorch
- Optimize hardware constraints (latency, bitwise operations, size limit, etc) with precision
- Features:
  - Enables low-precision (down to binary)
  - Mixed-precision quantization
  - Integer-only computational graph



Case Study: Jet substructure @LHC

- Jets are collimated showers of particles that result from the decay and hadronization of quarks and gluons
- Jets contain 100s particles whose properties and correlations may be exploited to identify physics signals





~4.7k parameters

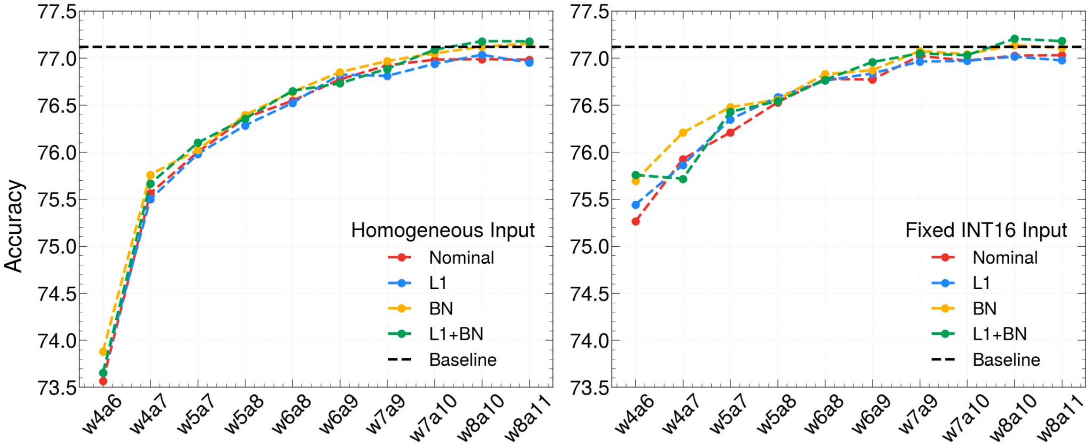
Uniform Bitwidth Quantization

- Utilizing a uniform bitwidth setting across all parameters
- Performance degradation observed when transition below INT8 weights
- Minimal to negligible influence observed from L1 Regularization and Batch Normalization on performance

Precision		Baseline [%]	L_1 [%]	BN [%]	L_1+BN [%]
Weights	Inputs				
INT12	INT12	76.916	72.105	77.180	76.458
INT8	INT8	76.605	76.448	76.899	76.879
INT6	INT6	73.55	73.666	74.468	74.415
INT4	INT4	62.513	63.167	63.548	63.431
FP-32	FP-32	76.461	76.826	76.853	76.813

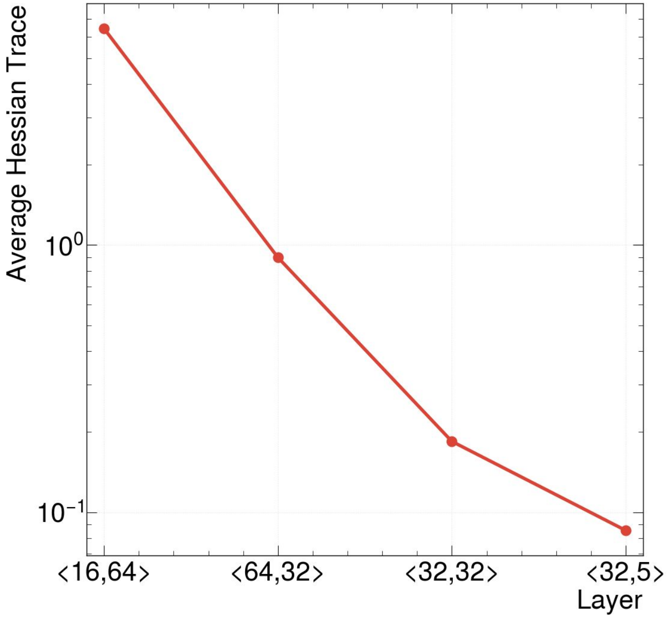
Progression Towards Mixed-Precision

- Utilizing a uniform bitwidth setting across all parameters
  With different activation bitwidths



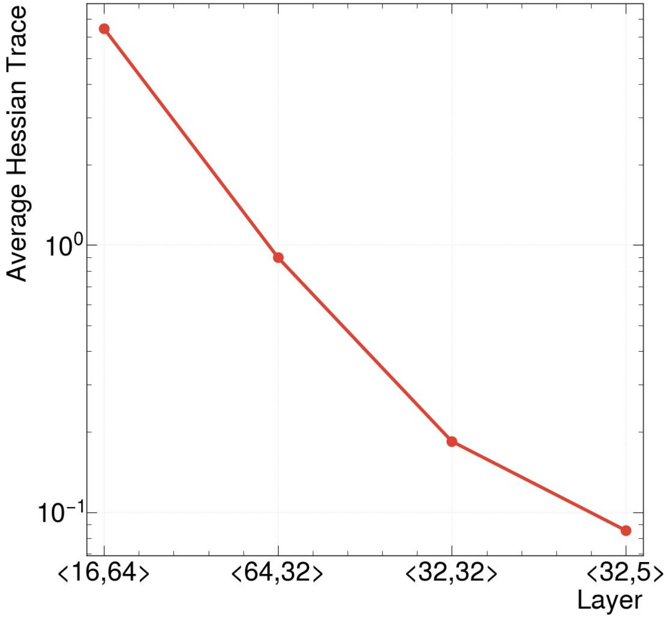
Layer Sensitivity and Hessian Analysis

- Certain layers are more sensitive to quantization than others
- Mixed precision strategy: aggressively quantize less sensitive layers to lower bitwidths
- NNs generalize better with locally flat minima-determined by the Hessian
- Use Hessian as sensitivity metric for quantization
  Layers ranked by Hessian trace



Layer Sensitivity and Hessian Analysis

- Certain layers are more sensitive to quantization than others
- Mixed precision strategy: aggressively quantize less sensitive layers to lower bitwidths
- NNs generalize better with locally flat minima-determined by the Hessian
- Use Hessian as sensitivity metric for quantization
  Layers ranked by Hessian trace



$$\Omega = \sum_{i=1}^{L} \Omega_i = \sum_{i=1}^{L} \boxed{\overline{Tr}(H_i)} * \boxed{\left\|Q(W_i) - W_i\right\|_2^2}$$

Hessian Trace **L2 norm of quantization perturbation**

Compute and sort each layer by Ω and select the bitwidth with the minimal Ω

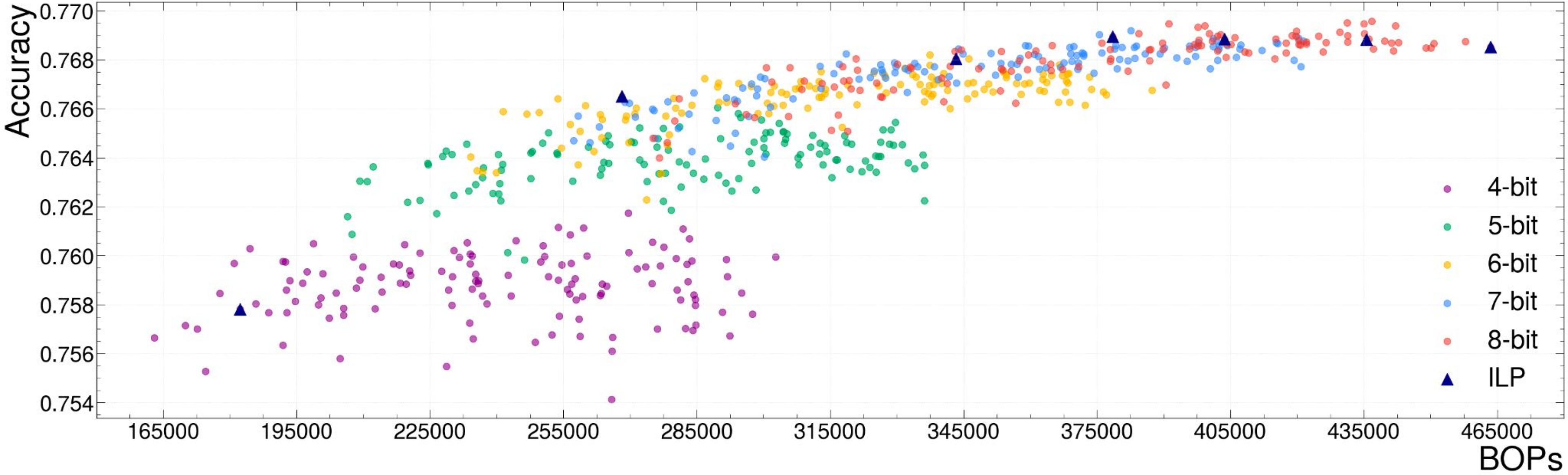
Hardware Constraints Optimization

- Bit Operations (BOPs) are computed to estimate model complexity
  Number of operations per inference

$$BOPs \approx mn\big(\big(1 - f_p\big)b_a b_w + b_a + b_w + \log_2(n)\big)$$

- Other constraints: Measured or est. latency, model size (number of parameters or memory size)

Mixed Precision Pareto Front



Data color coded based on the first bitwidth of the first fully-connected layer. It's importance in quantization coincides with the observed clusters, with higher performing points using larger bitwidths for input layer

PyTorch

hls 4 ml

HAWQ

Usual ML workflow:
neural architecture
search, compression,
...

ONNX

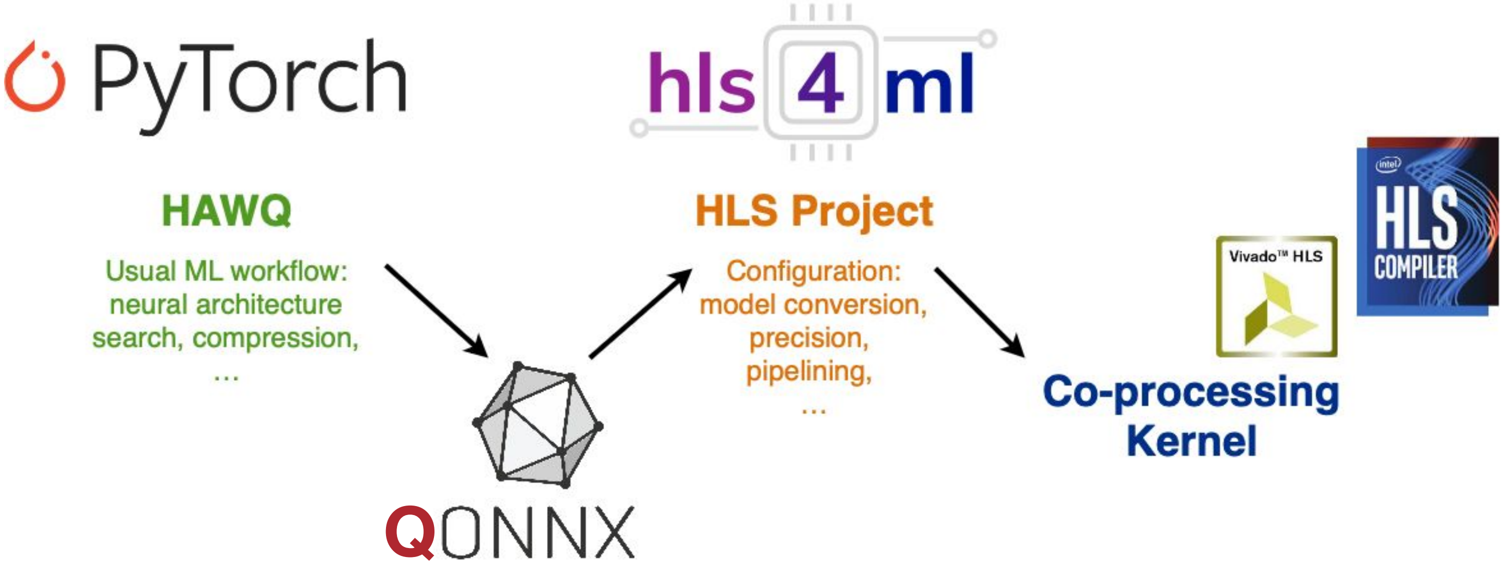
HLS Project

Configuration:
model conversion,
precision,
pipelining,
...

Vivado™ HLS

intel HLS COMPILER

Co-processing Kernel



Quantized ONNX extends ONNX open-source format for representing ML algorithms.

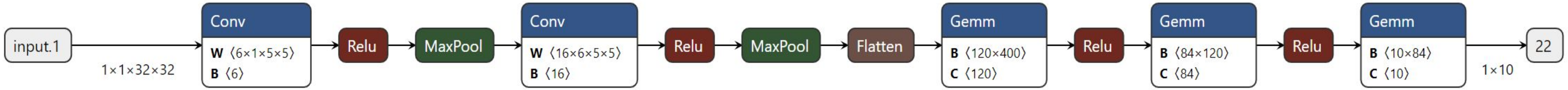
- **Low-bitwidth representation**
- **Mixed and arbitrary precision**
- **Intermediate representation abstraction**

```python
def __init__(self):
    super(Net, self).__init__()
    # 1 input image channel, 6 output channels, 5x5 square convolution
    # kernel
    self.conv1 = nn.Conv2d(1, 6, 5)
    self.conv2 = nn.Conv2d(6, 16, 5)
    # an affine operation: y = Wx + b
    self.fc1 = nn.Linear(16 * 5 * 5, 120)  # 5*5 from image dimension
    self.fc2 = nn.Linear(120, 84)
    self.fc3 = nn.Linear(84, 10)
```

PyTorch JIT COMPILER

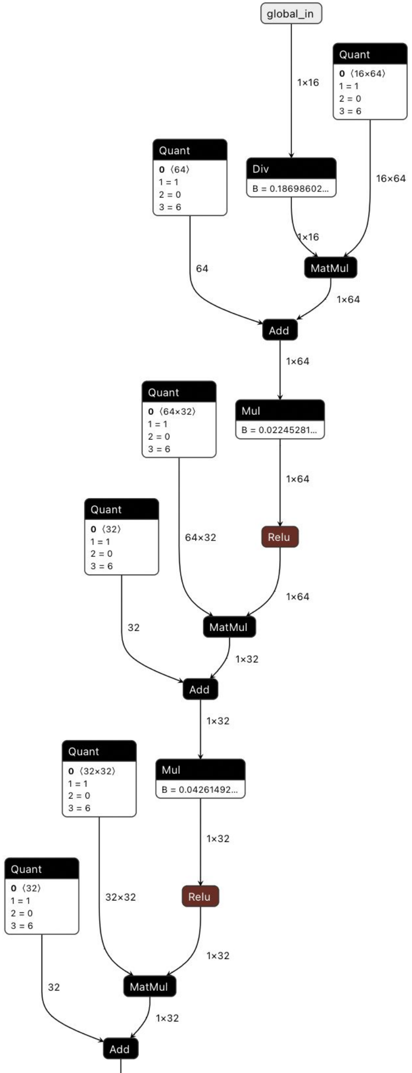
Torch IR Graph

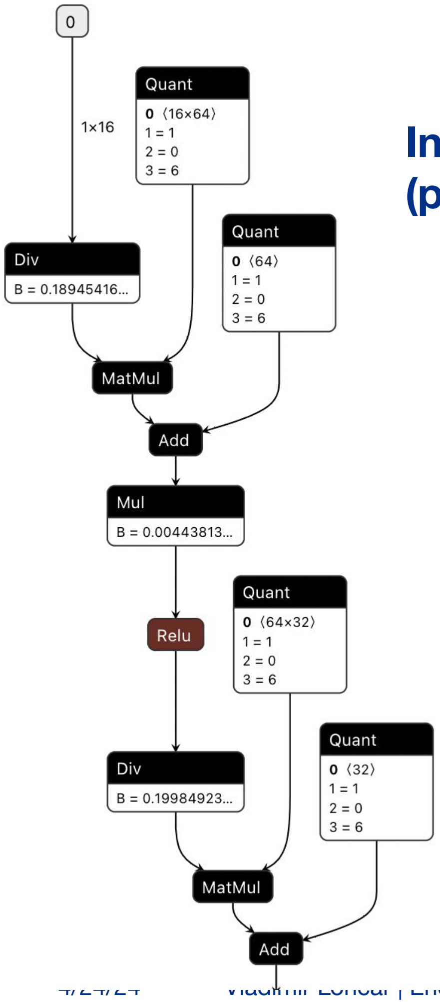
Torch IR Graph to ONNX Graph Translator



HAWQ to QONNX

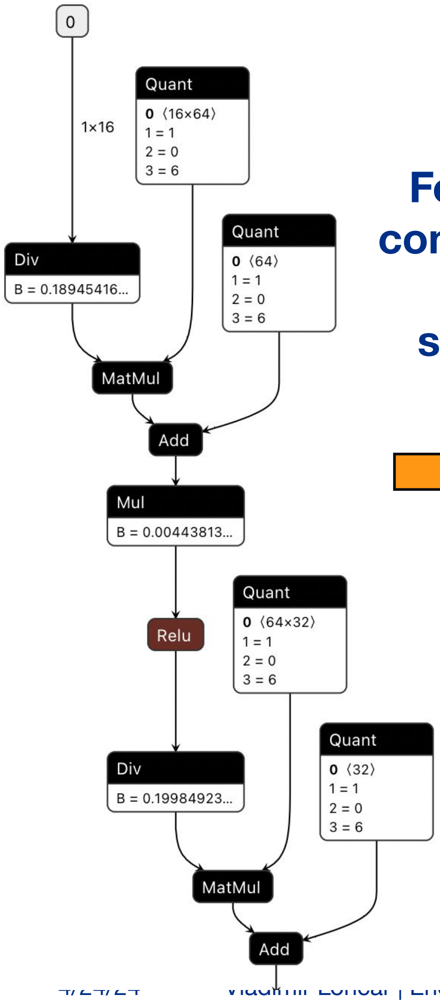
- QONNX introduces new operators to represent uniform quantization and abstracts implementation details



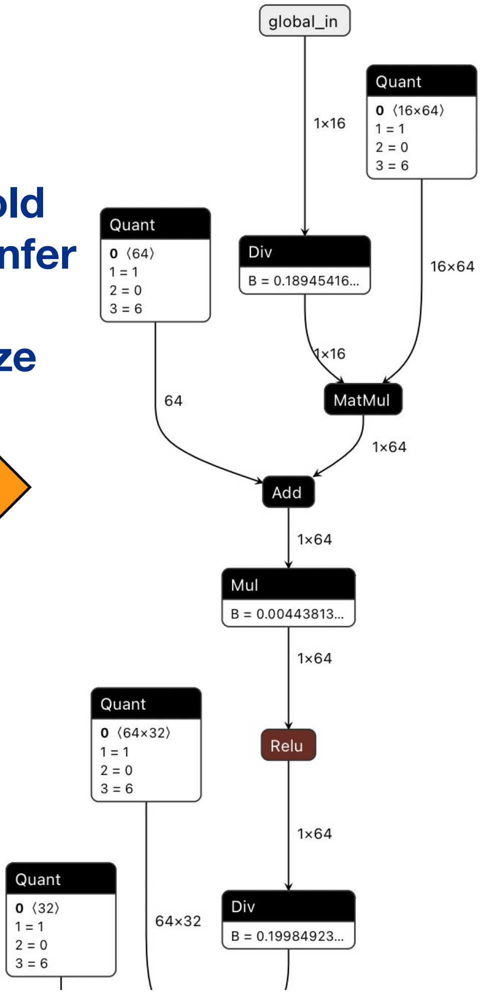


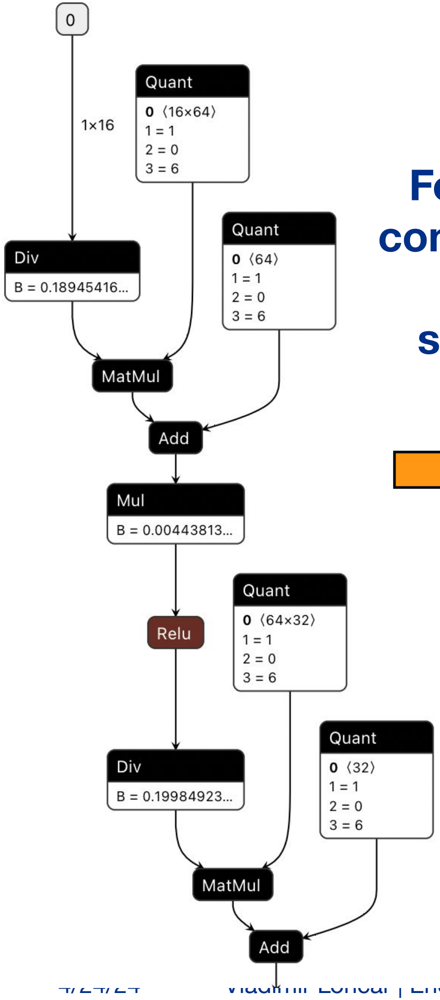
Initial format (post-export)

Fermilab

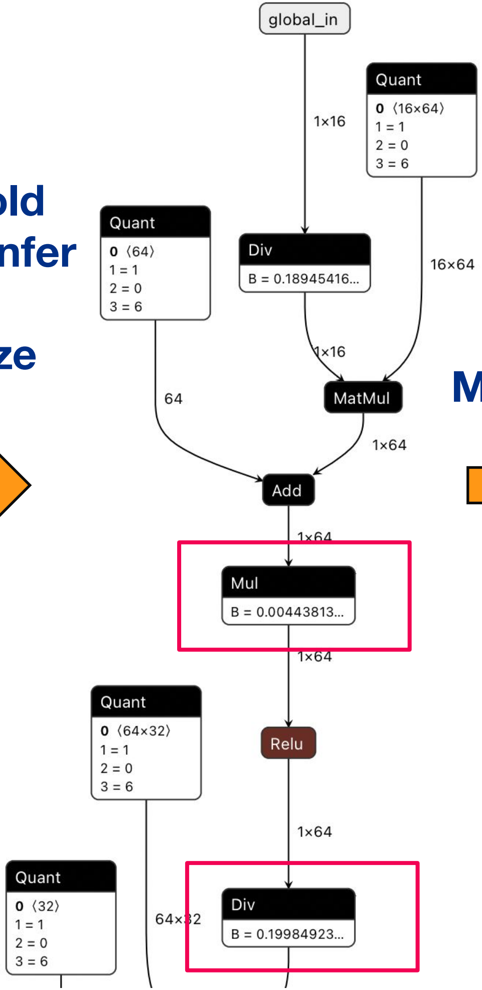


Format: Fold constants, infer shapes, standardize names

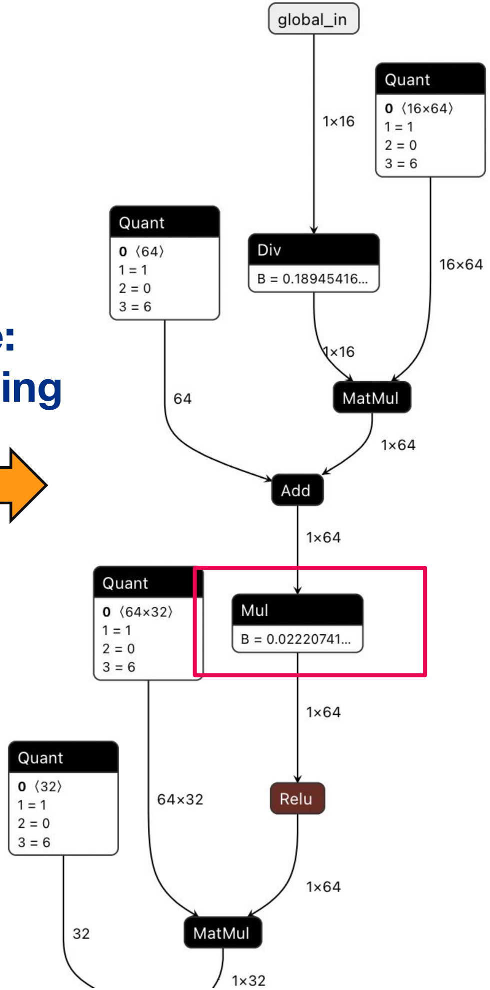


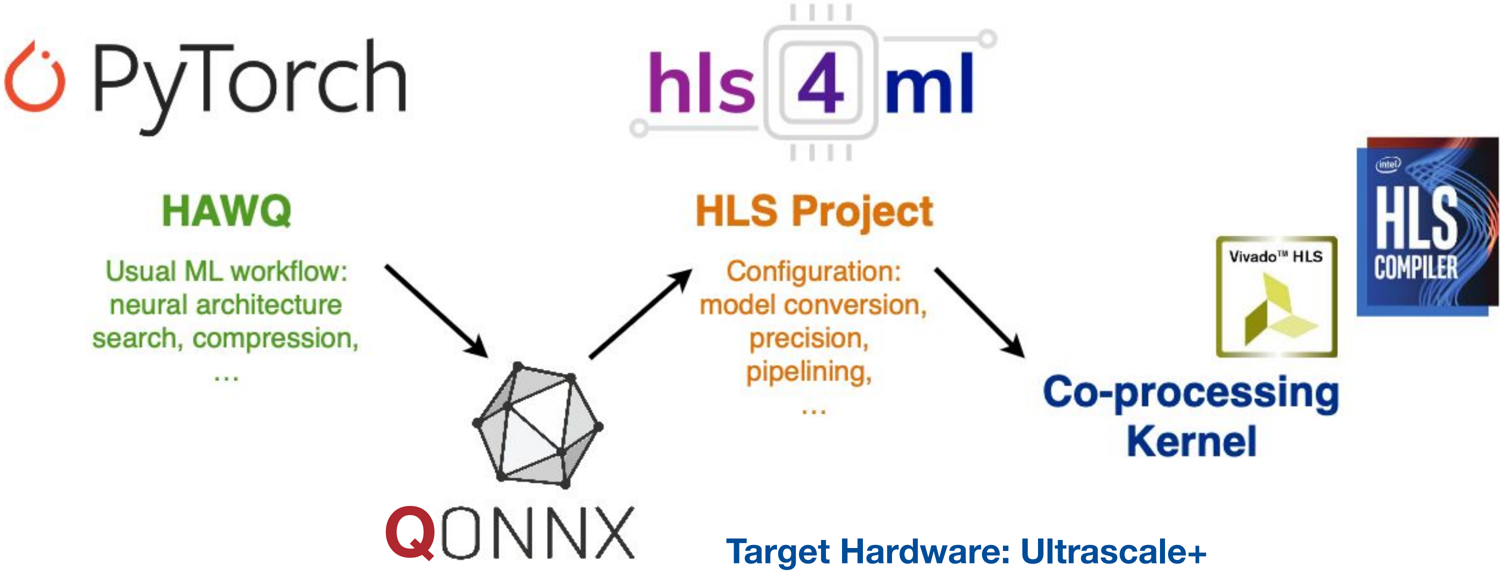


Format: Fold constants, infer shapes, standardize names



Optimize: Merge scaling factors

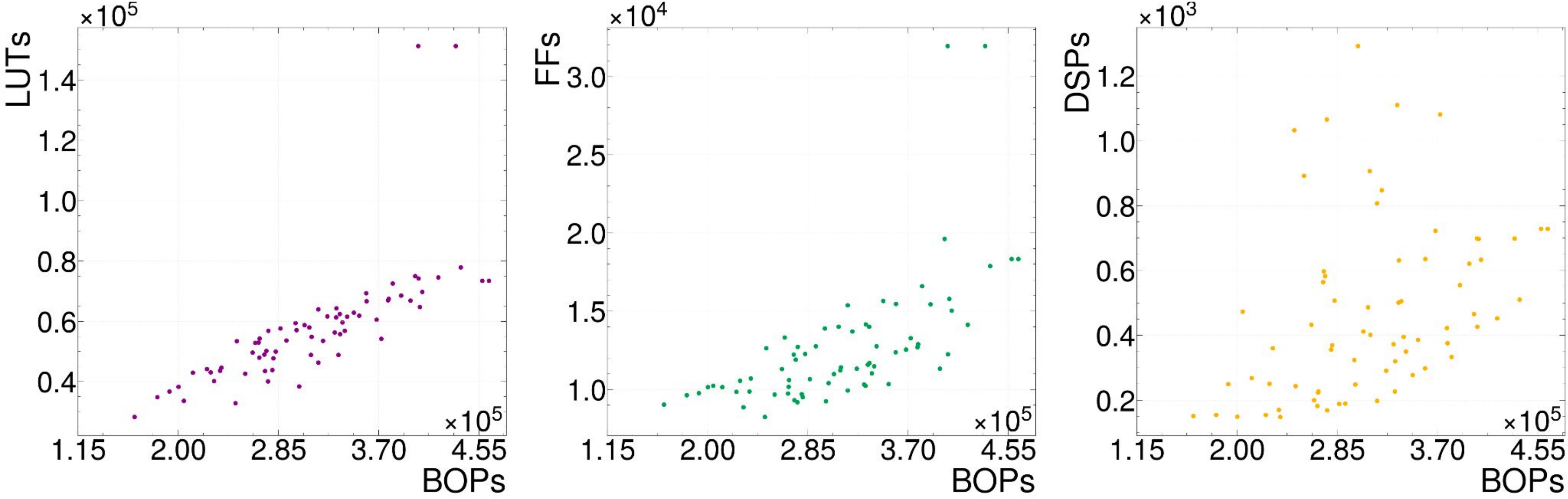




Target Hardware: Ultrascale+
Maximally Parallelized
Target Clock: 200MHz
*****Only NN is synthesized*****

ONNX ingestion in HLS4ML is experimental!

Firmware Results on Pareto Front



Firmware Results

- Hessian-aware solution significantly reduces all resource metrics
  Using 95.7%, 42.2%, and 36.3% fewer DSPs, LUTs, and FFs respectively
- Solution '*QB*' from AutoQkeras minimizes total bits in model
  Using binary and ternary operators at the cost of accuracy
- Unlike AutoQkeras, Hessian-aware quantization is done only once, then fine tuned after quantization

Model	Acc. [%]	Latency [ns]	Resources			Sparsity [%]	BOPs
			LUTs	FFs	DSPs		
Basline	76.85	65	60,272	15,116	3,602	0	4,652,832
INT8	76.45	95	54,888	14,210	671	30	281,277
Hessian	75.78	90	34,842	9,622	154	33	182,260
QB	72.79	60	16,144	4,172	5	23	122,680

Summary

- Hessian-aware solutions to mixed precision quantization schemes provide reliable solutions
- Exporting HAWQ to QONNX intermediate Representation is now possible
  - Standard ONNX is also supported!
    - Other ONNX accelerators can be targeted as well
- Models successfully translated from HAWQ to a firmware implementation
  - Bit operations serve as an early predictor of resource usage in LUTs, DSPs are less reliable

Questions