End-to-end codesign of Hessian-aware quantized NNs for FPGAs and ASICs

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

Fermilab

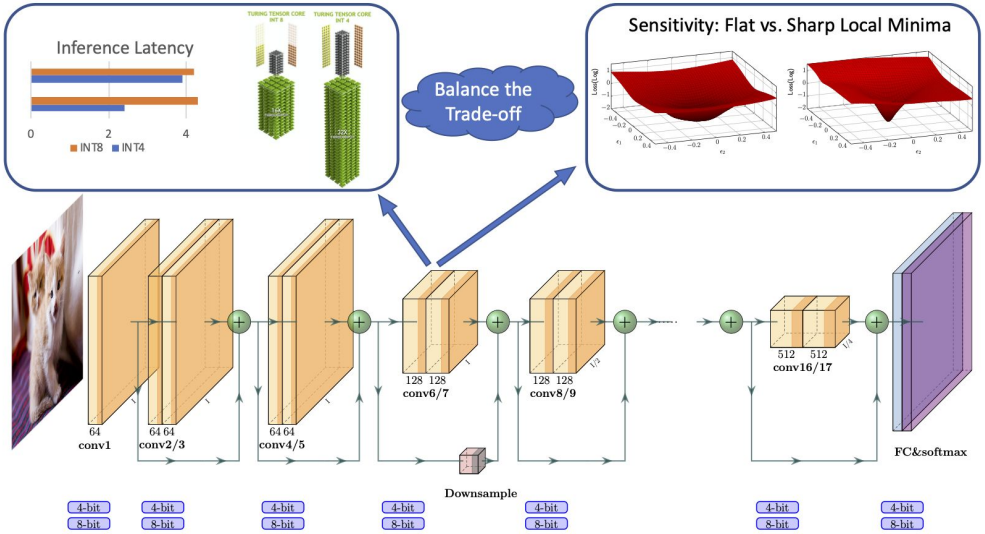
Background

Hessian-AWare Quantization (HAWQ)

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 the number of layers in the model



https://arxiv.org/pdf/2011.10680.pdf

Background cont.

Hessian-AWare Quantization (HAWQ)

HAWQ: An advanced quantization library written for PyTorch

Introduce hardware constraints (latency, bitwise operations, size limit, …) with precision

Features:

- Enables low-precision
- Mixed-precision quantization
- Integer-only computation graph

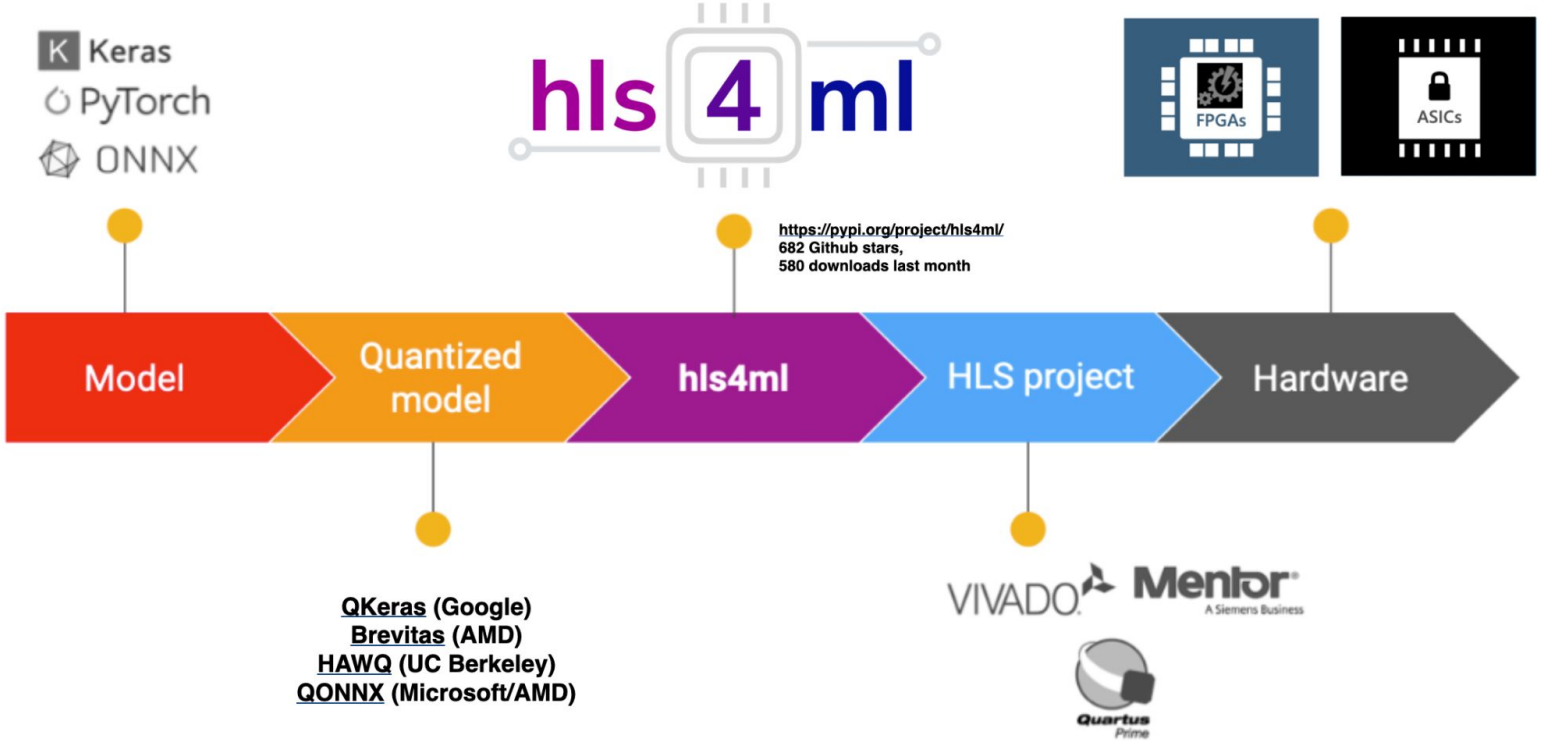
HAWQV3: Dyadic Neural Network Quantization

Zhewei Yao, Zhen Dong, Zhangcheng Zheng, Amir Gholami, Jiali Yu, Eric Tan, Leyuan Wang, Qijing Huang, Yida Wang, Michael W. Mahoney, Kurt Keutzer

Current low-precision quantization algorithms often have the hidden cost of conversion back and forth from floating point to quantized integer values. This hidden cost limits the latency improvement realized by quantizing Neural Networks. To address this, we present HAWQV3, a novel mixed-precision integer-only quantization framework. The contributions of HAWQV3 are the following: (i) An integer-only inference where the entire computational graph is performed only with integer multiplication, addition, and bit shifting, without any floating point operations or even integer division; (ii) A novel hardware-aware mixed-precision quantization method where the bit-precision is calculated by solving an integer linear programming problem that balances the trade-off between model perturbation and other constraints, e.g., memory footprint and latency; (iii) Direct hardware deployment and open source contribution for 4-bit uniform/mixed-precision quantization in TVM, achieving an average speed up of 1.45× for uniform 4-bit, as compared to uniform 8-bit for ResNet50 on T4 GPUs; and (iv) extensive evaluation of the proposed methods on ResNet18/50 and InceptionV3, for various model compression levels with/without mixed precision. For ResNet50, our INT8 quantization achieves an accuracy of 77.58%, which is 2.68% higher than prior integer-only work, and our mixed-precision INT4/8 quantization can reduce INT8 latency by 23% and still achieve 76.73% accuracy. Our framework and the TVM implementation have been open sourced.

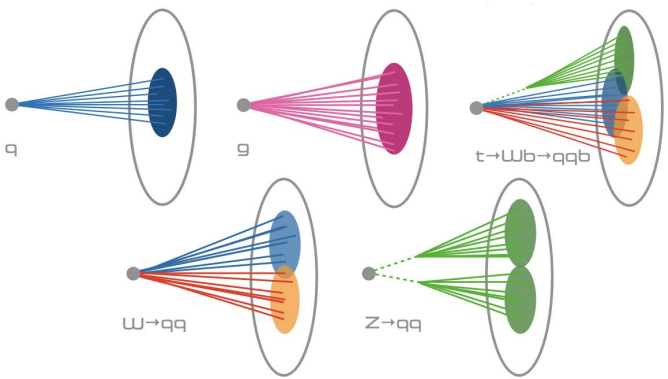
https://arxiv.org/pdf/2011.10680.pdf

https://fastmachinelearning.org/hls4ml/



Goal: Develop an end-to-end codesign workflow with HAWQ and hls4ml

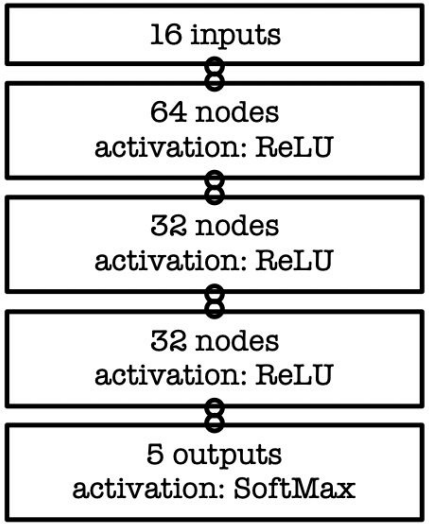
Case Study: Jet Tagging



Dataset of high-p_T jets from simulations of LHC proton-proton collisions

Multi-Layer Perceptron classifies jets into 5 classes (light flavor quarks, gluons, W and Z bosons, and top quarks)

Timing Constraint: 1μs

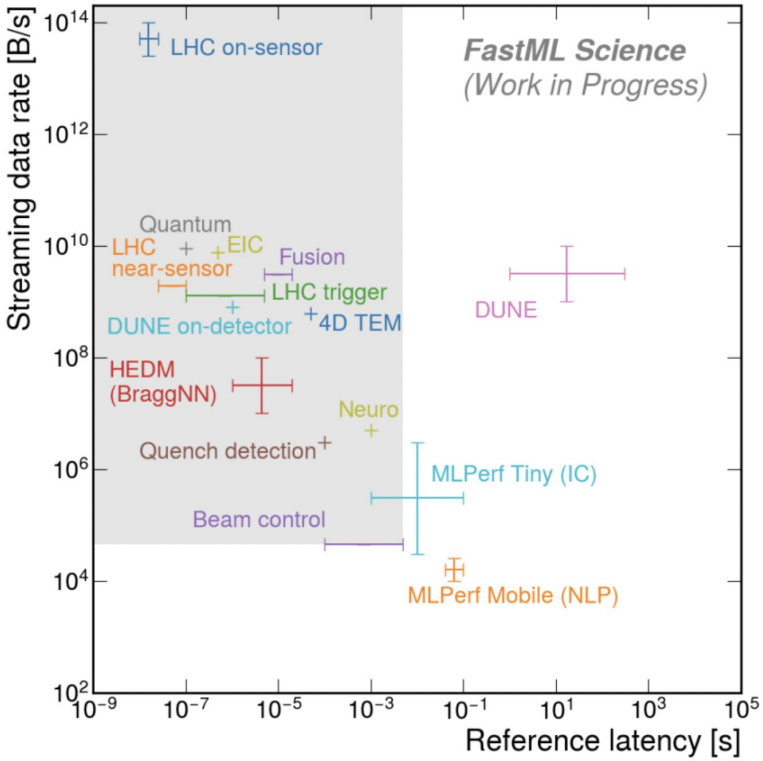


Scientific Challenges

Experiments: LHC and DUNE

Science: Quantum, Magnet development, Fusion, Neuroscience, Nuclear, Material sciences, etc.

Industry: Internet-of-Things, manufacturing



Homogeneous Quantization

Applying single bit width setting for all parameters

Performance begins to drop below INT8 weights

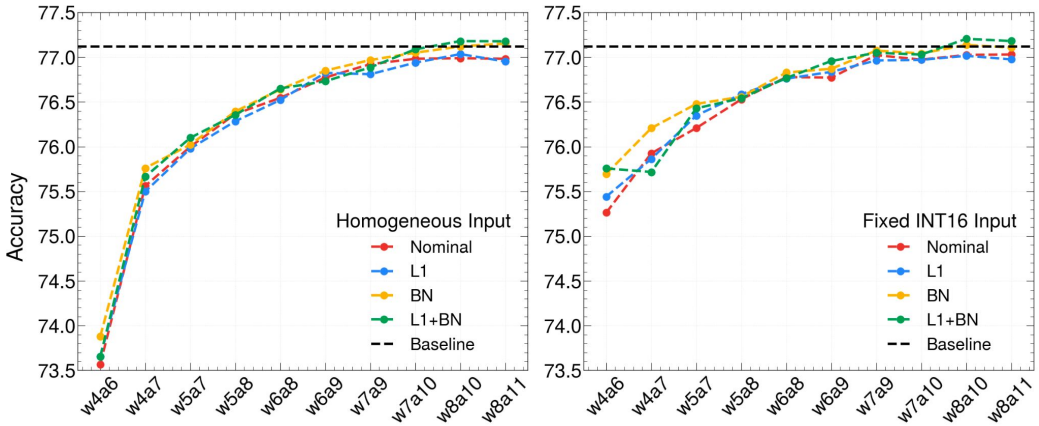
L1 Reg. and Batch normalization has little to no impact

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

Mixed Precision

A step towards mixed-precision

- All weights are assigned a single bit width
- All weights are assigned the same but different bit width
- Increase precision of inputs to INT16



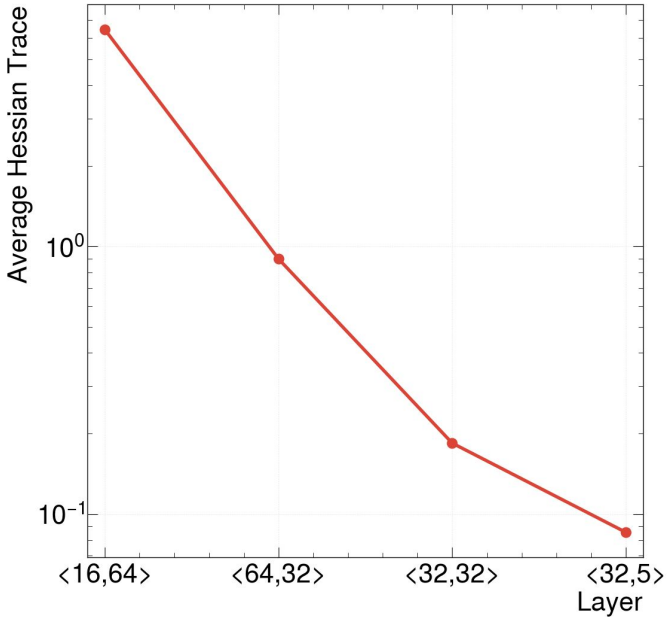
Hessian-Aware Quantization

Certain layers are more *sensitive* to quantization than others

Mixed precision: aggressively quantize less sensitive layers to lower bit widths

NNs generalize better with locally flat minima - determined by the Hessian

Use the Hessian as a sensitivity metric to quantization, where layers are ranked based on their trace



Hessian-Aware Quantization cont.

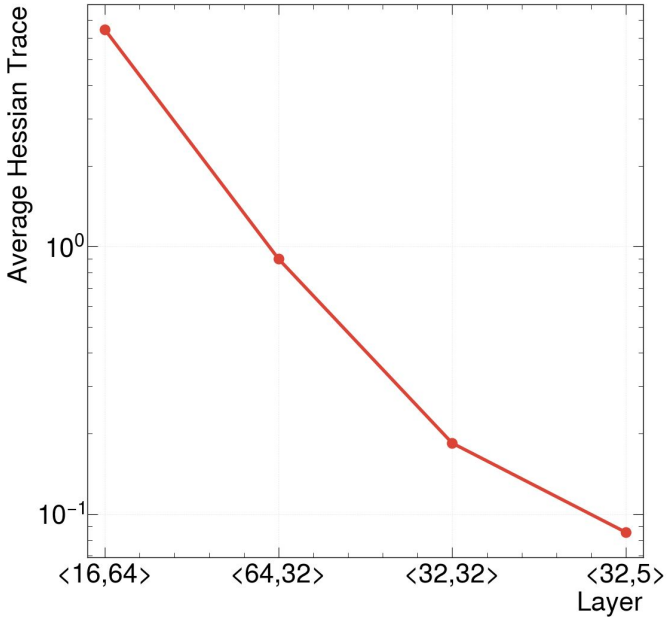
Integer Linear Programming (ILP): Optimization problem where variables are integers values, objection function and equations are linear

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

Hessian Trace

L2 norm of quantization perturbation

Compute and sort each layer by Ω, and select the bit setting with the minimal Ω



Adding Constraints

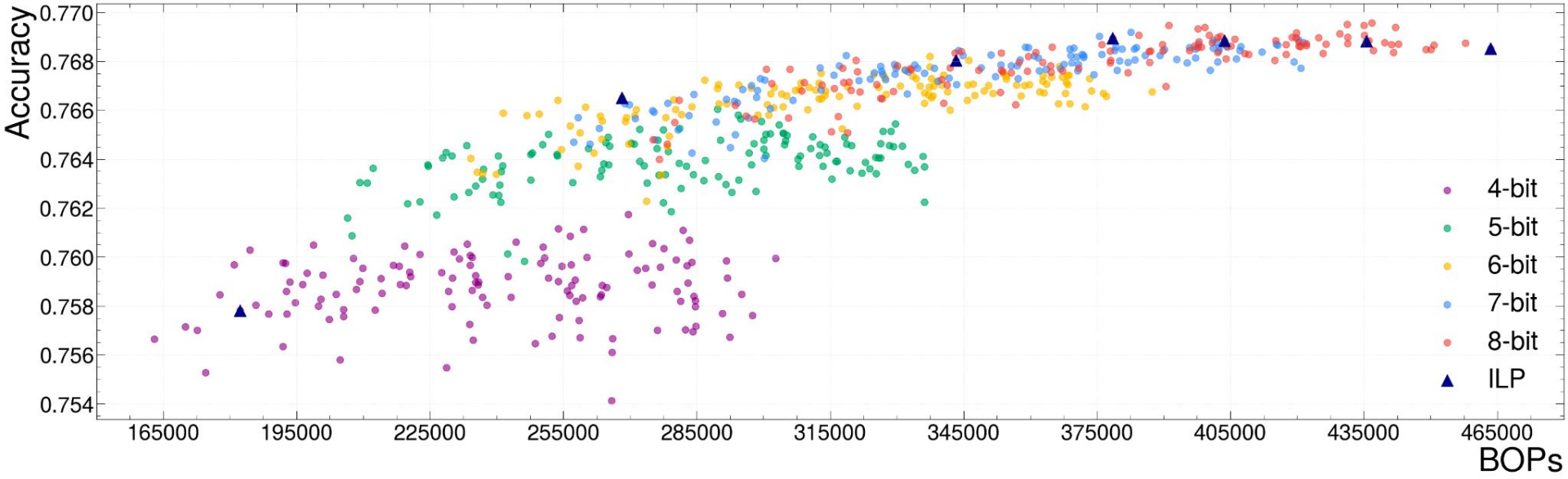
Bit Operations (BOPs) are computed to estimate model complexity and the number of operations per inference

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

Under constraints: ILP solves for the bit width of each layer, choosing the lowest Ω while remaining under the BOPs limit. *250 k to 550 k in steps of 50 k.*

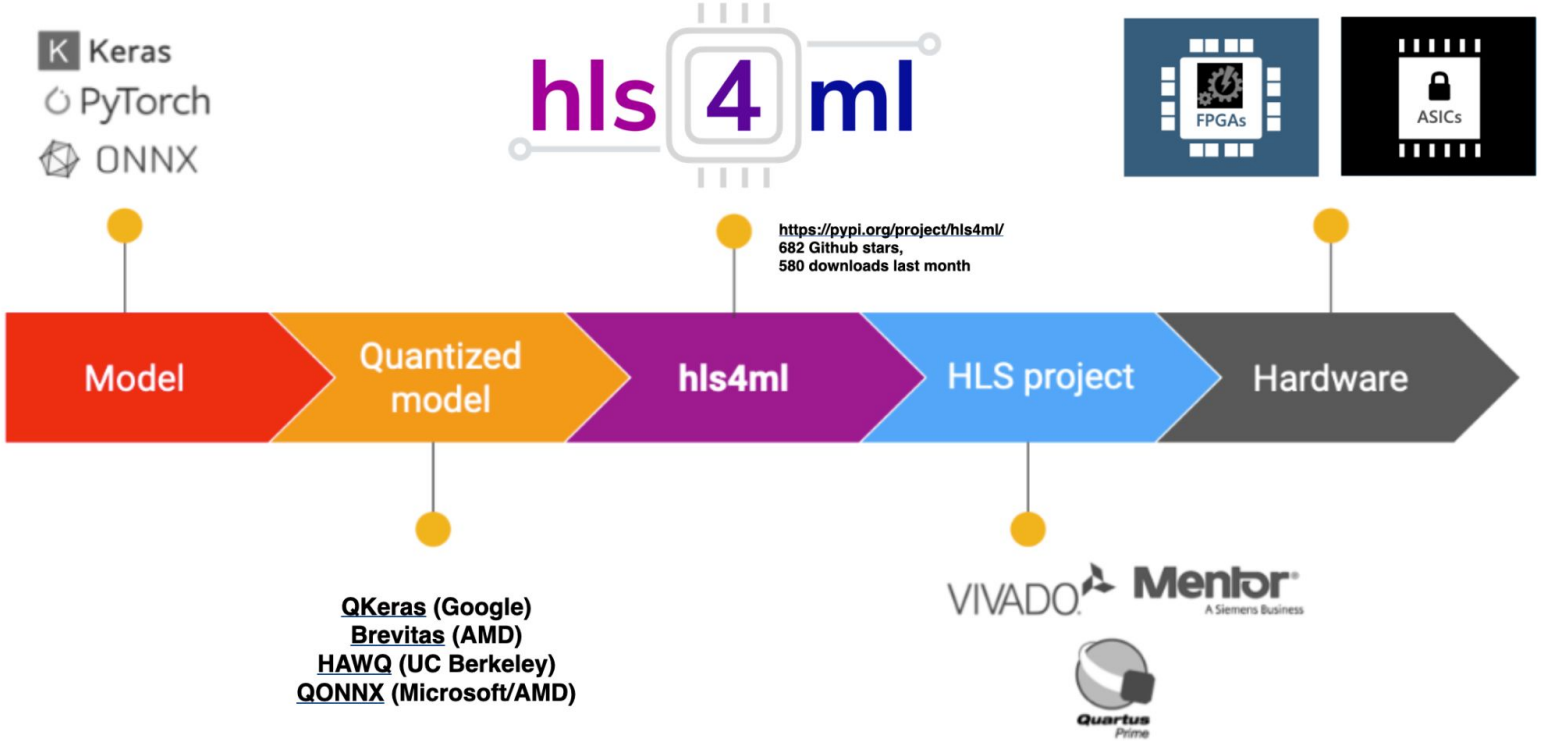
Other constraints: Latency (estimated or measured), model size (number of parameters or memory size)

Automatic Bit Selection cont.



- *Each data point is color-coded based on the bit width of the first fully-connected layer.*
- *It's importance in quantization coincides with the observed clusters, with higher performing models using larger bit widths.*

https://fastmachinelearning.org/hls4ml/



Goal: Develop an end-to-end codesign workflow with HAWQ and hls4ml

QONNX

Quantized ONNX (QONNX) provides an extension to ONNX

- Low-bitwidth quantization
- Mixed and arbitrary precision
- IR Abstraction

ONNX

hls4ml

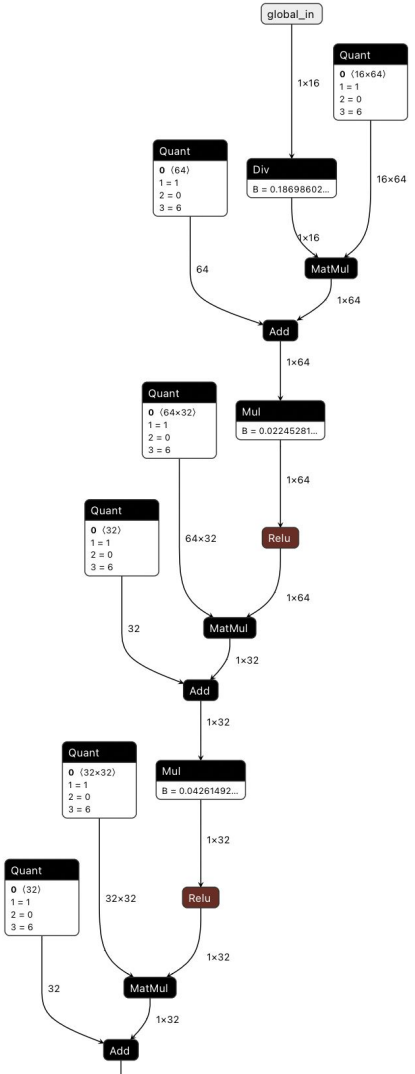
QONNX/ONNX support in hls4ml scheduled to be released in v0.8.0 (Summer 2023)

HAWQ to QONNX

Quantized ONNX introduces new operators to represent uniform quantization and abstracts implementation details

1. Conversion of HAWQ models leverages the PyTorch Just-In-Time compiler to trace computational graph and translate standard ONNX and custom QONNX operators (Quant, BipolarQuant, Trunc)
2. Post conversion, QONNX graphs are optimized and formatted for hls4ml ingestion

We can evaluate its classification performance to ensure we haven't lost accuracy using the [qonnx python package]



HAWQ to QONNX cont.

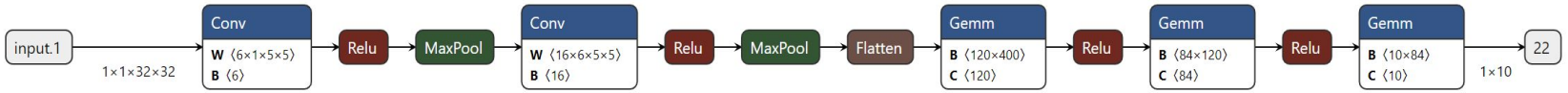
Model Translation (High Level View)

```
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



https://github.com/fastmachinelearning/qonnx.git
https://arxiv.org/abs/2206.07527

HAWQ to QONNX

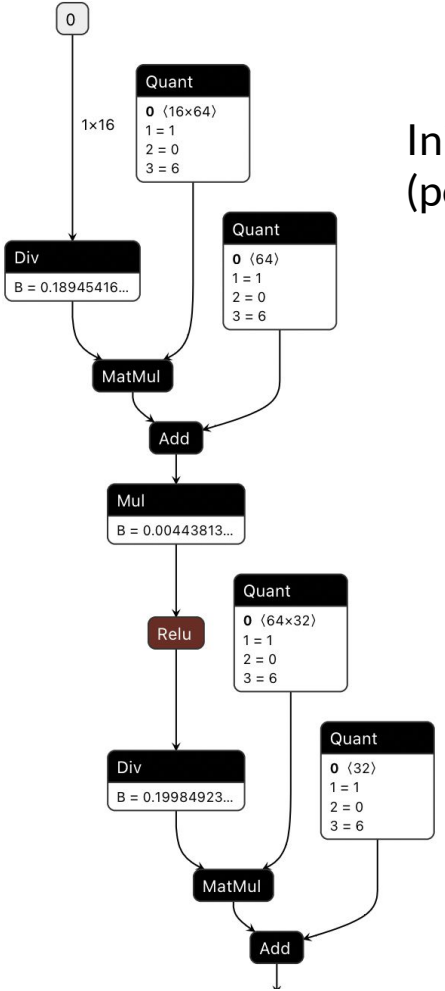
Exporting

Options:

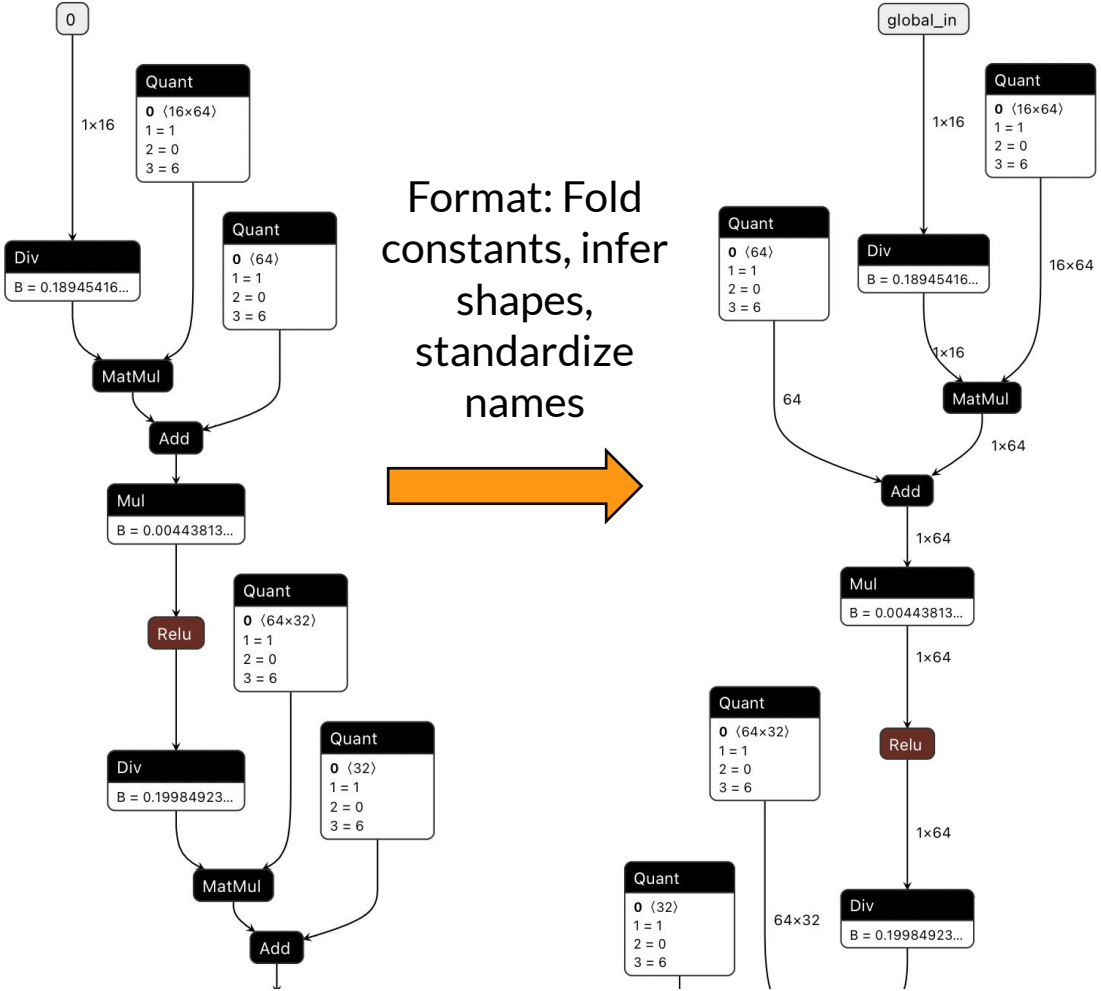
- Export with QONNX nodes
- Export with only standard ONNX nodes

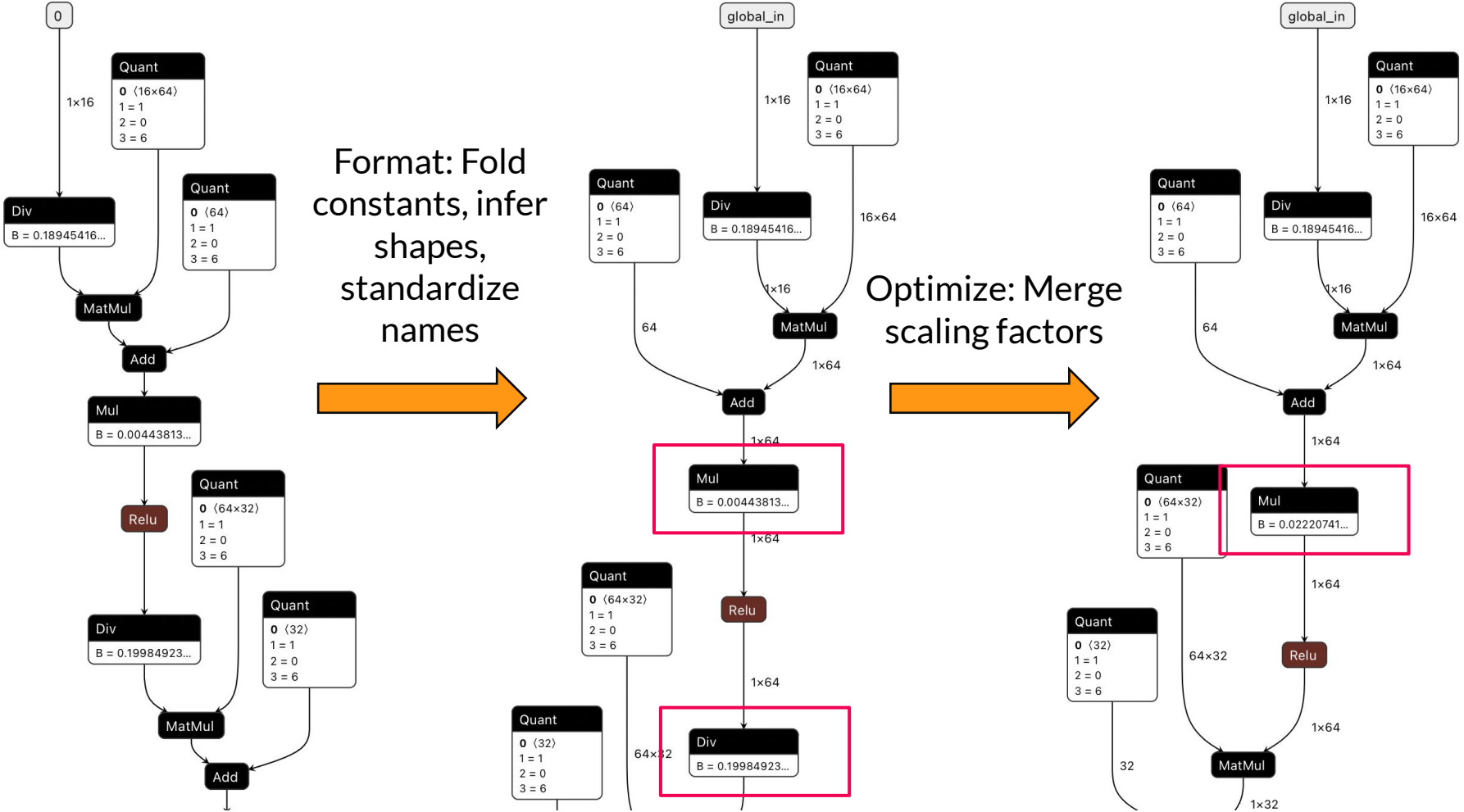
QONNX software utilities allows conversion from QONNX to Quantized Clip DeQuantize (QCDQ) and vice versa

```
from hawq.utils.export import ExportManager

...

manager = ExportManager(hawq_model)
manager.export(
    torch.randn([1, 16]),  # input for tracing
    "hawq2qonnx_model.onnx"
)
```



Initial format (post-export)


0
1×16
Quant
0 ⟨16×64⟩
1 = 1
2 = 0
3 = 6
Div
B = 0.18945416...
Quant
0 ⟨64⟩
1 = 1
2 = 0
3 = 6
MatMul
Add
Mul
B = 0.00443813...
Relu
Quant
0 ⟨64×32⟩
1 = 1
2 = 0
3 = 6
Div
B = 0.19984923...
Quant
0 ⟨32⟩
1 = 1
2 = 0
3 = 6
MatMul
Add
Format: Fold constants, infer shapes, standardize names
global_in
1×16
Quant
0 ⟨16×64⟩
1 = 1
2 = 0
3 = 6
Quant
0 ⟨64⟩
1 = 1
2 = 0
3 = 6
Div
B = 0.18945416...
16×64
1×16
64
MatMul
1×64
Add
1×64
Mul
B = 0.00443813...
1×64
Quant
0 ⟨64×32⟩
1 = 1
2 = 0
3 = 6
Relu
1×64
Quant
0 ⟨32⟩
1 = 1
2 = 0
3 = 6
64×32
Div
B = 0.19984923...


0
1×16
Quant
0 ⟨16×64⟩
1 = 1
2 = 0
3 = 6
Div
B = 0.18945416...
Quant
0 ⟨64⟩
1 = 1
2 = 0
3 = 6
MatMul
Add
Mul
B = 0.00443813...
Relu
Quant
0 ⟨64×32⟩
1 = 1
2 = 0
3 = 6
Div
B = 0.19984923...
Quant
0 ⟨32⟩
1 = 1
2 = 0
3 = 6
MatMul
Add
Format: Fold constants, infer shapes, standardize names
global_in
1×16
Quant
0 ⟨16×64⟩
1 = 1
2 = 0
3 = 6
Quant
0 ⟨64⟩
1 = 1
2 = 0
3 = 6
Div
B = 0.18945416...
16×64
1×16
64
MatMul
1×64
Add
1×64
Mul
B = 0.00443813...
1×64
Quant
0 ⟨64×32⟩
1 = 1
2 = 0
3 = 6
Relu
1×64
Quant
0 ⟨32⟩
1 = 1
2 = 0
3 = 6
64×32
Div
B = 0.19984923...
Optimize: Merge scaling factors
global_in
1×16
Quant
0 ⟨16×64⟩
1 = 1
2 = 0
3 = 6
Quant
0 ⟨64⟩
1 = 1
2 = 0
3 = 6
Div
B = 0.18945416...
16×64
1×16
64
MatMul
1×64
Add
1×64
Quant
0 ⟨64×32⟩
1 = 1
2 = 0
3 = 6
Mul
B = 0.02220741...
1×64
Quant
0 ⟨32⟩
1 = 1
2 = 0
3 = 6
64×32
Relu
1×64
32
MatMul
1×32

QONNX to hls4ml

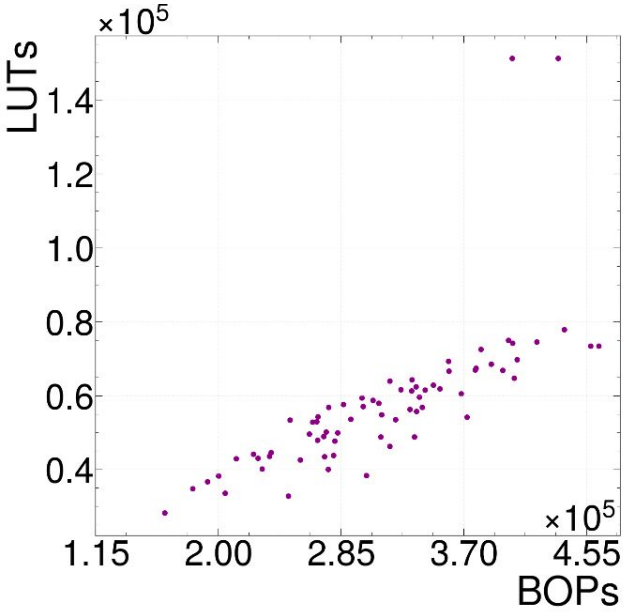
```
Backend: Vivado [2020.1]
Part: xcu250-figd2104-2L-e
ClockPeriod: 5
IOType: io_parallel
HLSConfig:
  Model:
    ReuseFactor: 1
    Strategy: Resource
```

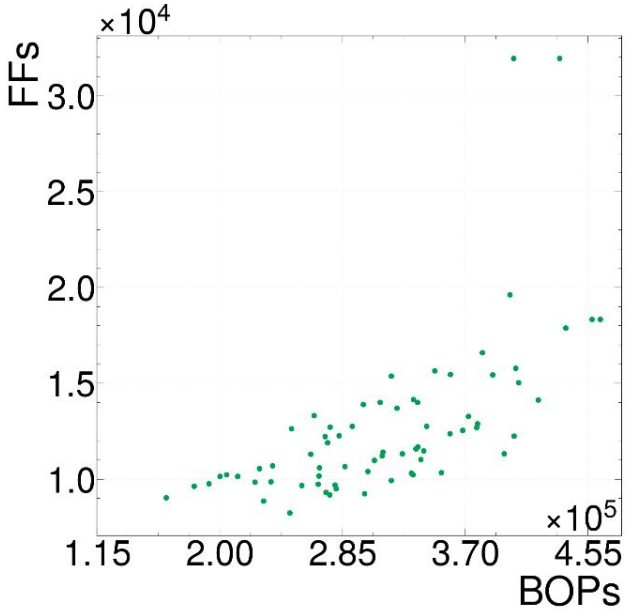
Vivado HLS (vitis support released in v0.7)

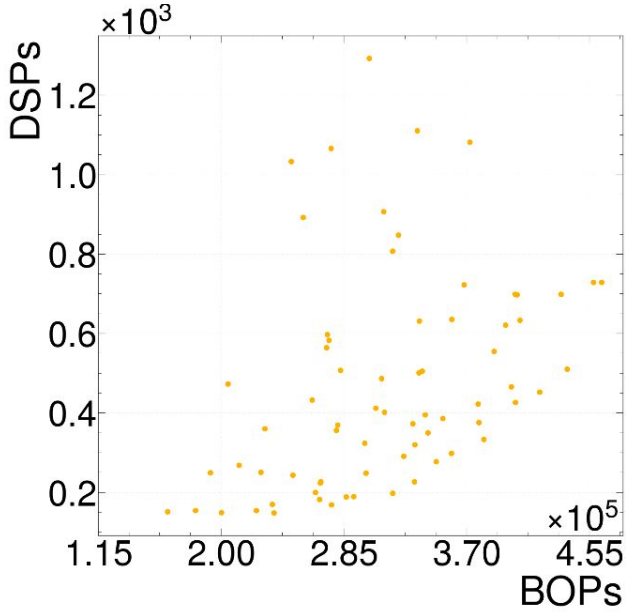
Memory mapped IO

Fully-unrolled loops (maximally parallelized design)

Firmware Synthesis







Firmware Synthesis cont.

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

- The Hessian-aware solution significantly reduces all resource metrics compared to baseline (LUTs, FFs, and DSPs)
  - 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

Summary

- Showed that the Hessian-aware solution to a mixed precision quantization scheme provides a reliable solution
- Used our exporter in HAWQ to translate multiple MLPs optimized with various bit settings to their QONNX IR
- Models were successfully translated from HAWQ to a firmware implementation, and we've observed the resource usage compared to the total BOPs and accuracy
- Compared the Hessian-aware model with a homogeneous bit configuration and baseline.
- The Hessian-aware solution significantly reduced all resource metrics (LUTs, FFs, and DSPs), with the most significant improvements in DSPs and LUTs, using 95.7% and 42.2% fewer DSPs and LUTs compared to baseline, respectively

Links

HAWQ library

HAWQ Papers [v1][v2][v3]

QONNX [paper][software]

HLS4ML

End-to-end codesign of Hessian-aware quantized neural networks for FPGAs and ASICs

JAVIER CAMPOS, JOVAN MITREVSKI, and NHAN TRAN, Fermi National Accelerator Laboratory, USA
ZHEN DONG, AMIR GHOLAMI*, and MICHAEL W. MAHONEY†, University of California Berkeley, USA
JAVIER DUARTE, University of California San Diego, USA

We develop an end-to-end workflow for the training and implementation of co-designed neural networks (NNs) for efficient field-programmable gate array (FPGA) and application-specific integrated circuit (ASIC) hardware. Our approach leverages Hessian-aware quantization (HAWQ) of NNs, the Quantized Open Neural Network Exchange (QONNX) intermediate representation, and the hls4ml tool flow for transpiling NNs into FPGA and ASIC firmware. This makes efficient NN implementations in hardware accessible to nonexperts, in a single open-sourced workflow that can be deployed for real-time machine-learning applications in a wide range of scientific and industrial settings. We demonstrate the workflow in a particle physics application involving trigger decisions that must operate at the 40 MHz collision rate of the CERN Large Hadron Collider (LHC). Given the high collision rate, all data processing must be implemented on custom ASIC and FPGA hardware within the strict area and latency requirements. Based on these constraints, we implement an optimized mixed-precision NN classifier for high-momentum particle jets in simulated LHC proton-proton collisions.

Additional Key Words and Phrases: neural networks, field programmable gate arrays, firmware, high-level synthesis

https://arxiv.org/abs/2304.06745