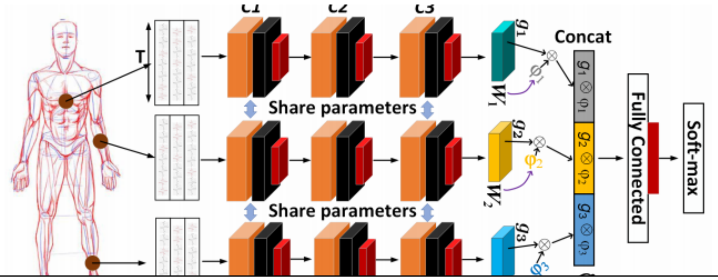
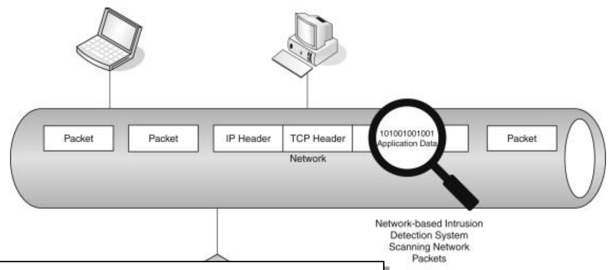
QONNX: A proposal for representing arbitrary-precision quantized NNs in ONNX

Alessandro Pappalardo, Yaman Umuroglu

with FastML/hls4ml collaborators:
Hendrik Borras, Nhan Tran, Javier Duarte, Jovan Mitrevski, Sioni Summers, Vladimir Loncar, et al.

Few-bit Quantized DNNs Work Well for Many Tasks





Object Detectio

[Liu et al, arXiv:20

n (2/3-bit)

2004.03021]

When & where do few-bit QNNs make sense?

- **Need highly optimized deployment**
- **Throughput, latency, power constraints**

Not covered here due to time constraints:

- **How to train accurate few-bit QNNs with quantization-aware training**
- **Basics of uniform quantization**

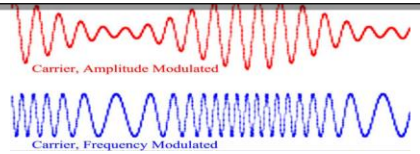


Image Classification (1- to 4-bit)

[Zhang et al, arXiv:1807.10029]

Modulation Classification (2-bit)

[Tridgell et al, FPT'19]

Uniform affine quantization

- Define the uniform affine quantization function as:

$$y = quantize(x) = clamp(round\left(\frac{x}{s} + z\right), y_{min}, y_{max})$$

- Define the uniform affine dequantization function as:

$$u = dequantize(y) = (y - z) * s$$

- Where s is the scale/resolution, z the zero-point, y_{min}, y_{max} depend on the bit-width and signedness.
- Then the fake quantization function is:

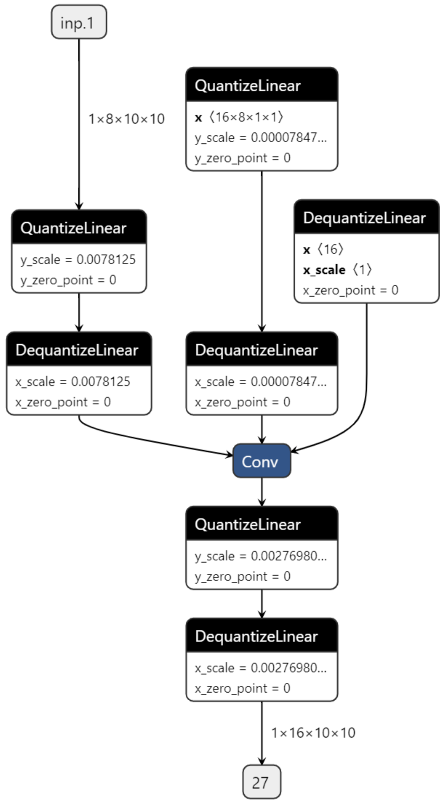
$$u = dequantize\ (quantize(x))$$

- Maps floating-point value x to a floating point approximation u.
- At inference time can be mapped to integer operations.

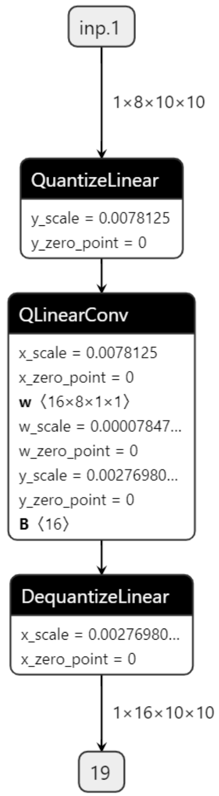
Current quantization formats in ONNX

- QDQ for fake-quantization.
- QLinear operators for fused quantized ops.
- QLinear can be output of QDQ + fusion transforms.
- Limited to 8b quantization and 8b/32b dequantization.

QDQ



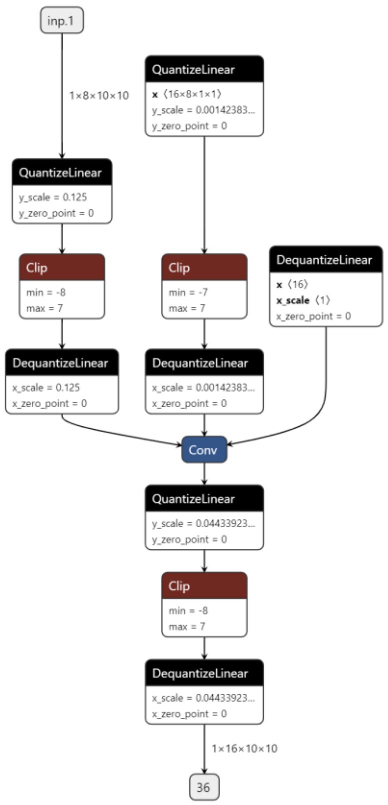
QLinear ops



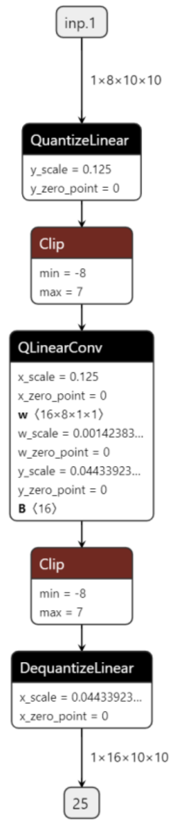
Extending ONNX quantization formats with Clip

- QCDQ for modelling <= 8b fake-quantization.
- Clip integer output of QuantizeLinear.
- Clip integer range implies lower bit width.
- Runs correctly on existing toolchains/backends.
- Updated backends can take advantage of extra acceleration < 8b.
- Qlinear w/ Clip can be output of QCDQ + fusion transforms.
- Limited to a precision <= 8b .
- Limited to a layer-wise precision.
- Support in the next
  release of the Brevitas PyTorch quantization lib
  https://github.com/Xilinx/brevitas

4b QCDQ

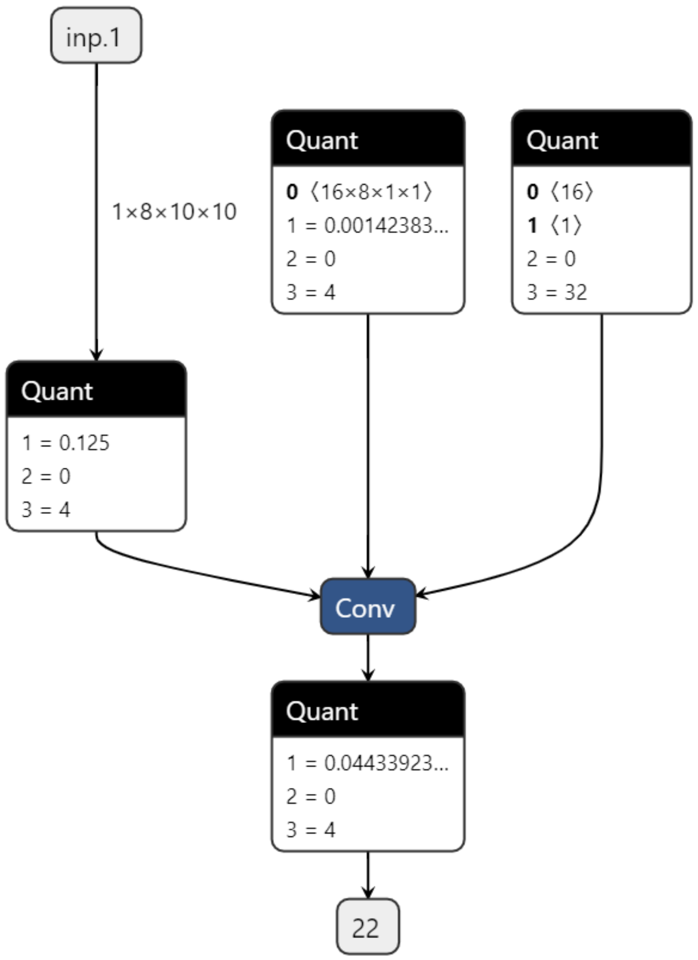


4b QLinear w/ Clip



QONNX – generalized fake-quantization dialect

- Set of custom ONNX ops performing arbitrary precision fake quantization.
- Quant node - general uniform affine quantization
  - Takes as inputs the value to quantize, scale, zero-point, bit-width
  - Supports different types of rounding
- BipolarQuant node - bipolar (-1,+1) binary quantization
- Supported as export format in the Brevitas quantization library https://github.com/Xilinx/brevitas



Open-Source QONNX Repositories Available Today

Python Toolkit

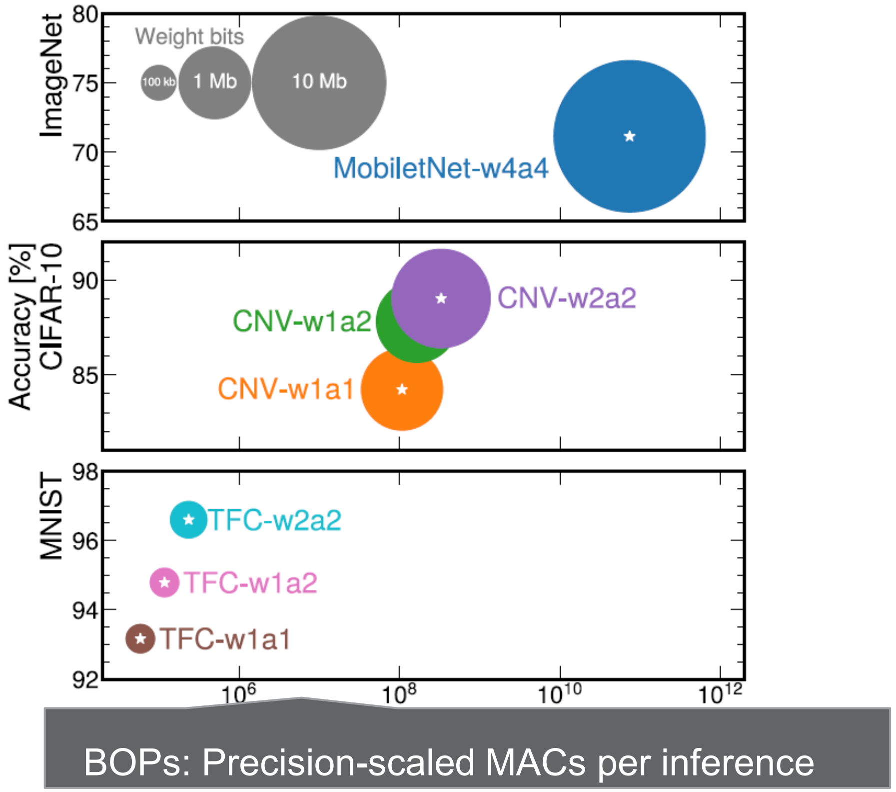
Utilities for executing, optimizing and transforming QONNX models

- execution of custom QONNX nodes
- shape inference
- data layout conversion (NCHW -> NHWC)
- general-purpose optimizers
  - const folding, batchnorm-to-affine, reordering...
- framework for defining Python-based optimizers

fastmachinelearning/qonnx

Model Zoo

Pretrained few-bit QONNX models



fastmachinelearning/QONNX_model_zoo

Conclusion

- Presented two new styles for expressing sub-8-bit quantization in ONNX:
  - QCDQ (standard ops, verbose, limited to <=8-bit) and QONNX (custom ops, compact, arbitrary precision)
  - See HiPEAC'22 AccML paper http://arxiv.org/abs/2206.07527 for more details
- QONNX already adopted as output in the Brevitas PyTorch quantization library, QCDQ in the next release
- QONNX Already adopted as common input format by two major FPGA NN inference frameworks
  - FINN and hls4ml - adopted by 100s of users and customers
- QONNX being adopted by the HAWQ quantization library, QKeras->QONNX converter under construction
- Open-source Python toolkit and model zoo available
- Open questions
  - Are the QONNX operators of interest for the broader community?
  - Should they become part of the standard ONNX ops?

Interested?
Please get in touch!

Alessandro Pappalardo
alessandro.pappalardo@amd.com

Yaman Umuroglu
yaman.umuroglu@amd.com

AMD

Why are sub-8-bit quantized NNs important?