FAST ML WORKSHOP:

Automatic heterogeneous quantization of DNNs for ultra low-area, low-latency inference at particle colliders

arXiv:2006.10159

Thea Aarrestad, Vladimir Loncar, Jennifer Ngadiuba, Maurizio Pierini, Adrian Pol, Sioni Summers (CERN)
Claudionor N. Coelho Jr. (Palo Alto Networks)
Aki Kuusela, Shan Li, Hao Zhuang (Google LLC)

01.12.20
Southern Methodist University

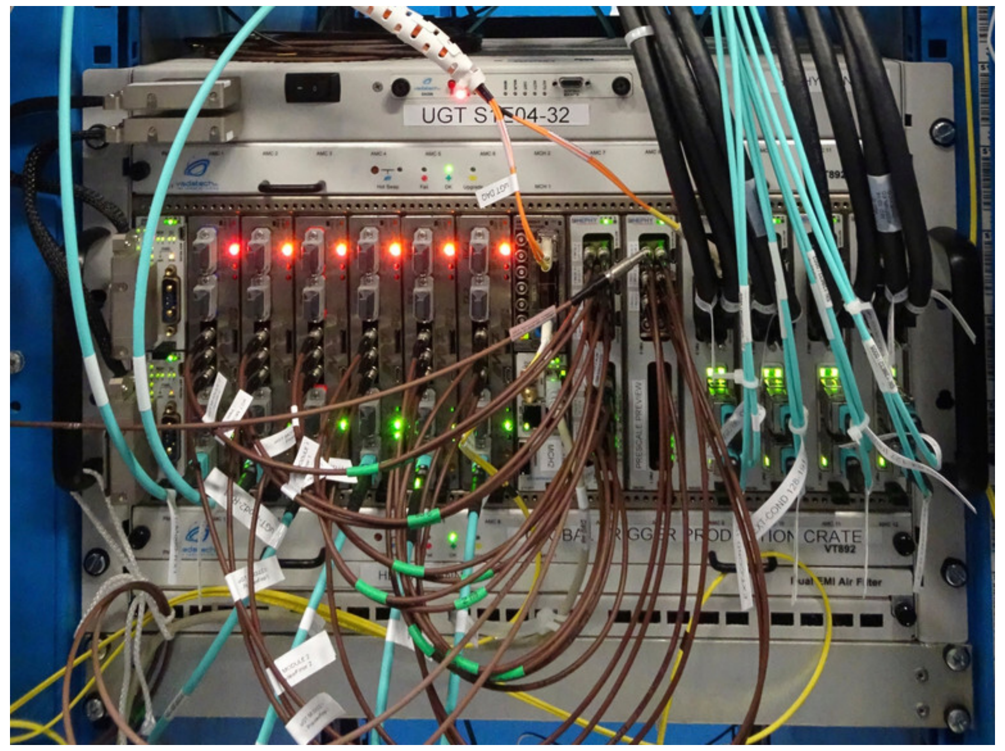
Edge inference at LHC

See Jennifer's talk

Level-1 hardware trigger

- **12.5 μs** to make decision
- Input data bandwidth **63 Tb/s**
- **1024** algos in parallel on **12 FPGAs**

40 MHz



750 kHz

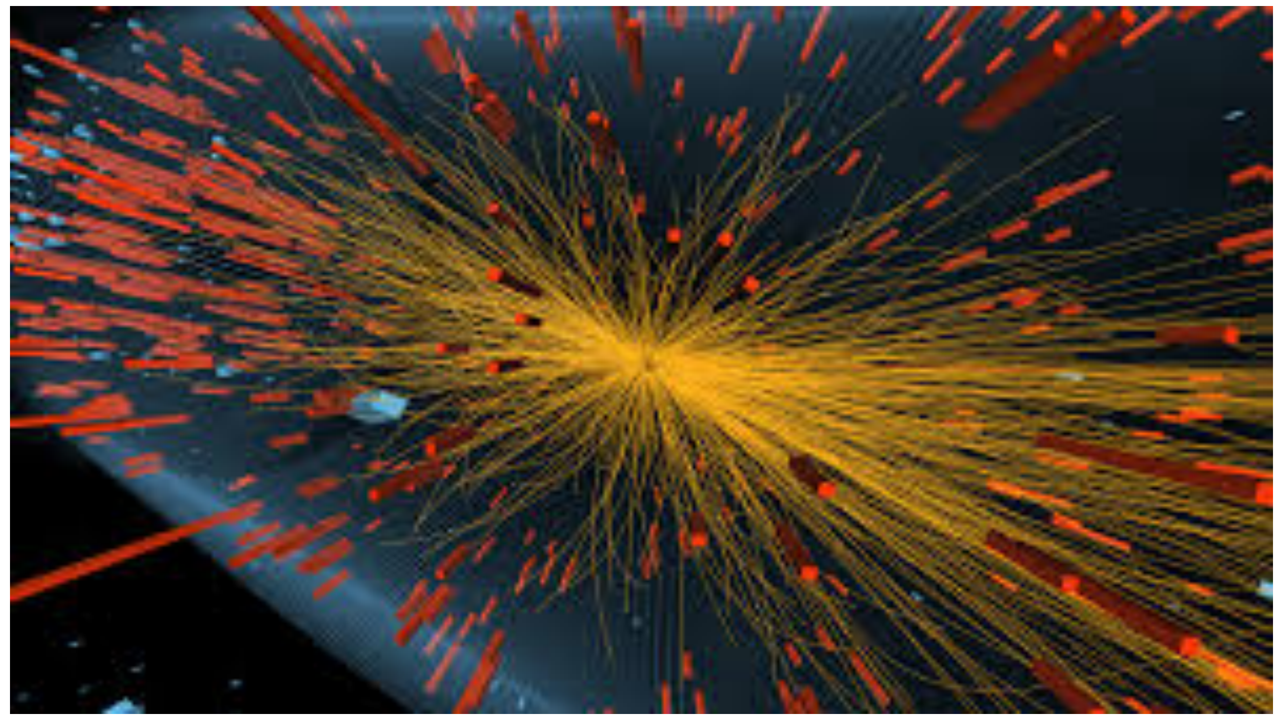
High Level Trigger CPU farm

1 kHz

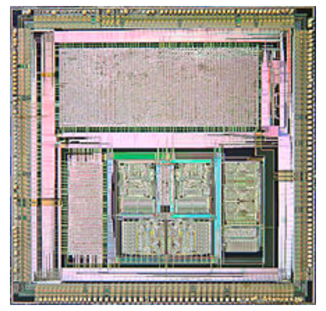
Offline reconstruction and storage

Detector

- Collisions every 25 ns
- Detector front–end **ASICs**



See Vladimir's and Claire's talks



See Giuseppe's talk

Extremely limited area and high competition over resources → need ultra-compressed DNNs (and tools for obtaining them easily)

See Michaela's and Amir's talks

Quantization

Fixed point post-training quantization

- Floating point 32 arithmetic use **x3-5** more resources, **x2** higher latency than fixed-point → convert to fixed-point

Decimal: 3.25

During training: $-1^{S} \cdot 2^{E-127} \cdot (1.M)$

01000000010100000000000000000000

S Exponent Mantissa

On hardware: ap_fixed (W,I)

⟨00011.01⟩

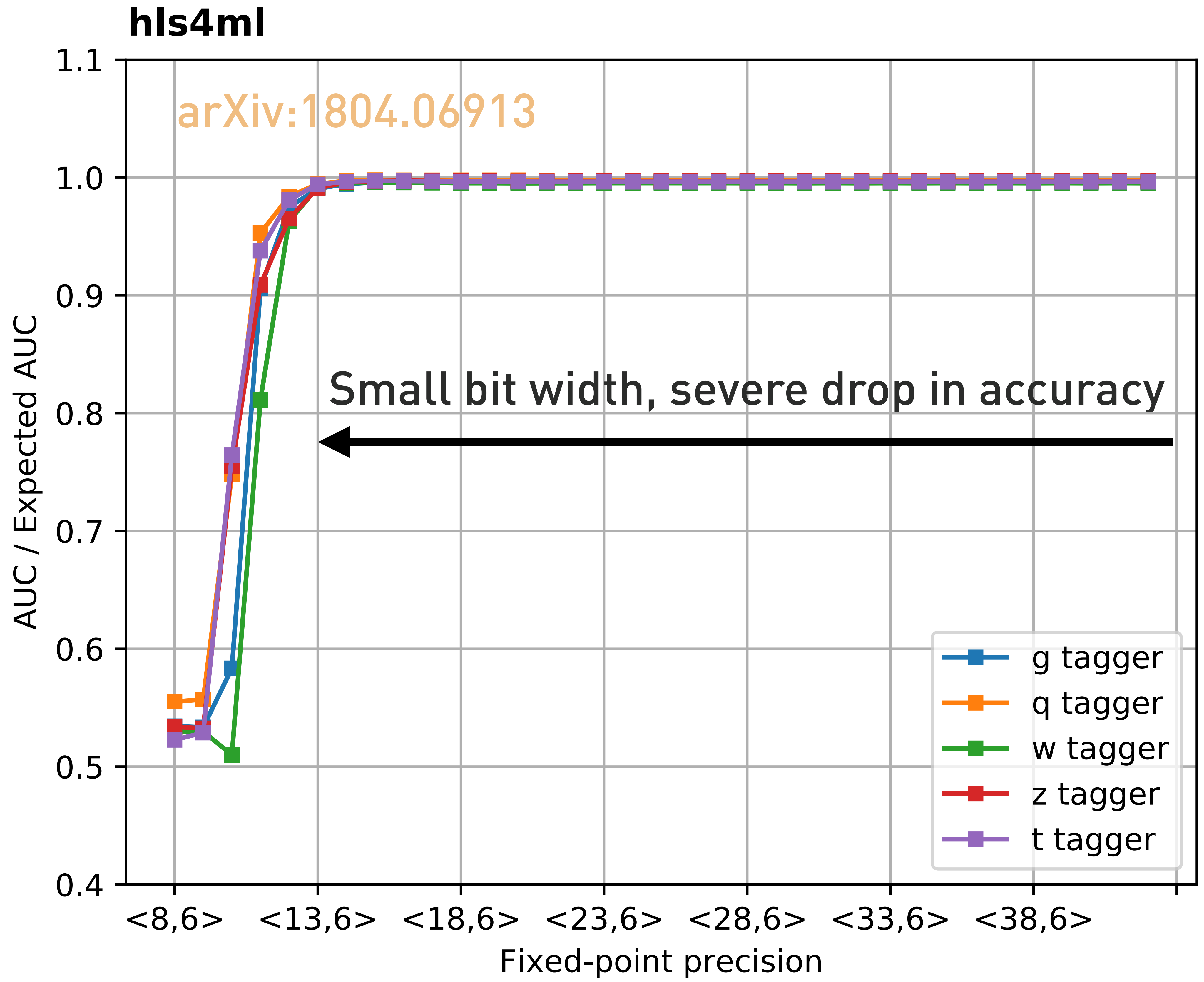
I

W

By definition lossy, precision must be tuned carefully (weights usually don't need large dynamic range. But, worse 'resolution')

Can we do better? Yes!

- Quantization-aware training (QAT): Rounded/clipped quantized weights in forward pass, fp32 in back-prop (straight-through estimator)
- Binary/ternary quantized networks already supported in hls4ml



QKeras+hls4ml

QKeras: Library for training quantization-aware Keras models

- Simple drop-in replacement of Keras layers
- Heterogenous quantization (per layer, parameter type)

Several quantizers available

- Exponent quantization, e.g ‘quantized_po2'
- Mantissa quantization, e.g 'quantized_bits’
- Both above, eg. ‘ternary' and 'binary'

Full support for QKeras models in hls4ml

- Easy for users to design and deploy quantized, low-latency DNNs on chip!

```
from tensorflow.keras.layers import Input, Activation
from qkeras import quantized_bits
from qkeras import QDense, QActivation
from qkeras import QBatchNormalization

x = Input((16))
x = QDense(64,
    kernel_quantizer = quantized_bits(6,0,alpha=1),
    bias_quantizer   = quantized_bits(6,0,alpha=1))(x)
x = QBatchNormalization()(x)
x = QActivation('quantized_relu(6,0)')(x)
x = QDense(32,
    kernel_quantizer = quantized_bits(6,0,alpha=1),
    bias_quantizer   = quantized_bits(6,0,alpha=1))(x)
x = QBatchNormalization()(x)
x = QActivation('quantized_relu(6,0)')(x)
x = QDense(32,
    kernel_quantizer = quantized_bits(6,0,alpha=1),
    bias_quantizer   = quantized_bits(6,0,alpha=1))(x)
x = QBatchNormalization()(x)
x = QActivation('quantized_relu(6,0)')(x)
x = QDense(5,
    kernel_quantizer = quantized_bits(6,0,alpha=1),
    bias_quantizer   = quantized_bits(6,0,alpha=1))(x)
x = Activation('softmax')(x)
```

QKeras+hls4ml

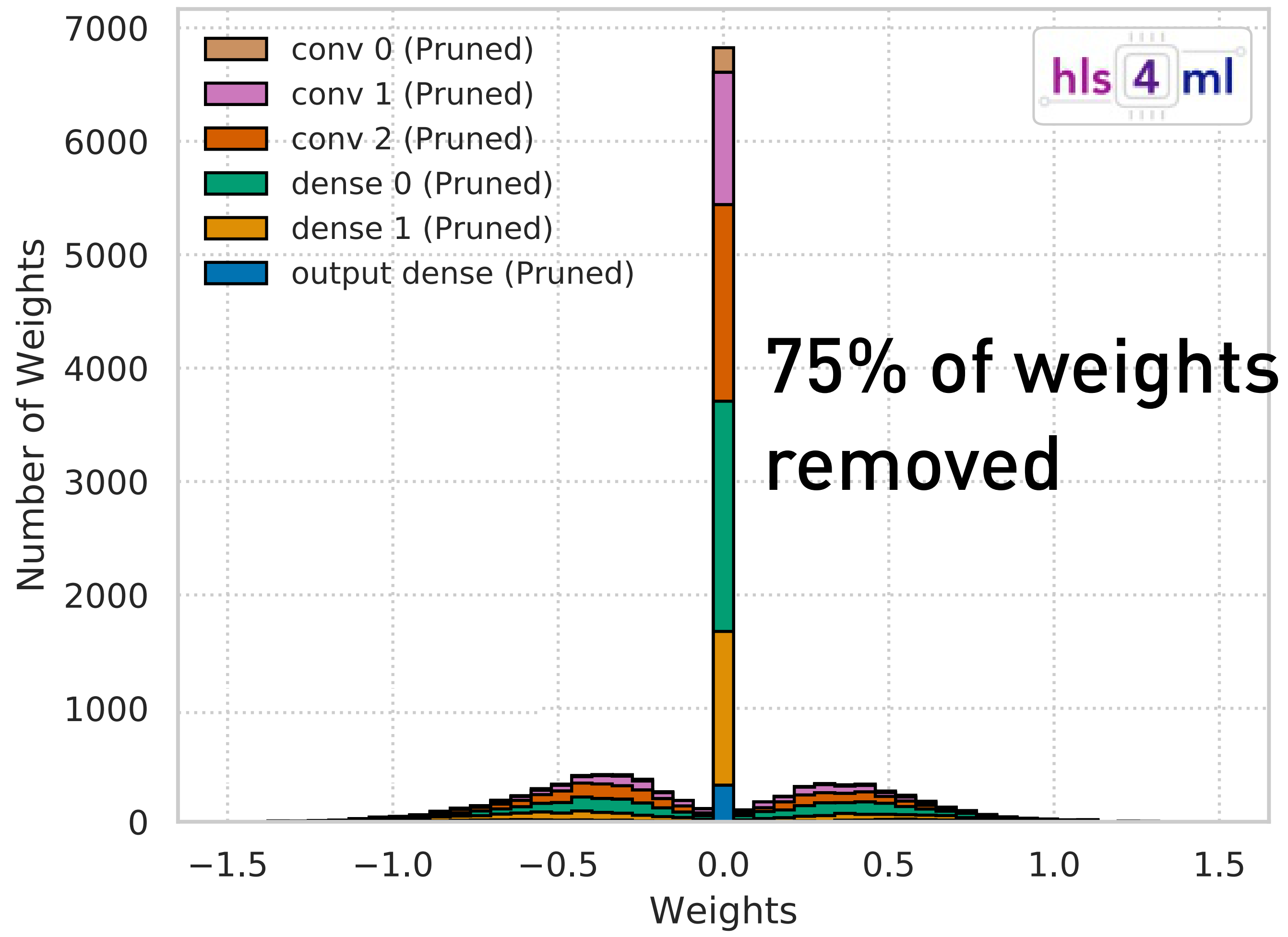
QKeras: **Library for training quantization-aware Keras models**
- Simple drop-in replacement of Keras layers
- Heterogenous quantization (per layer, parameter type)

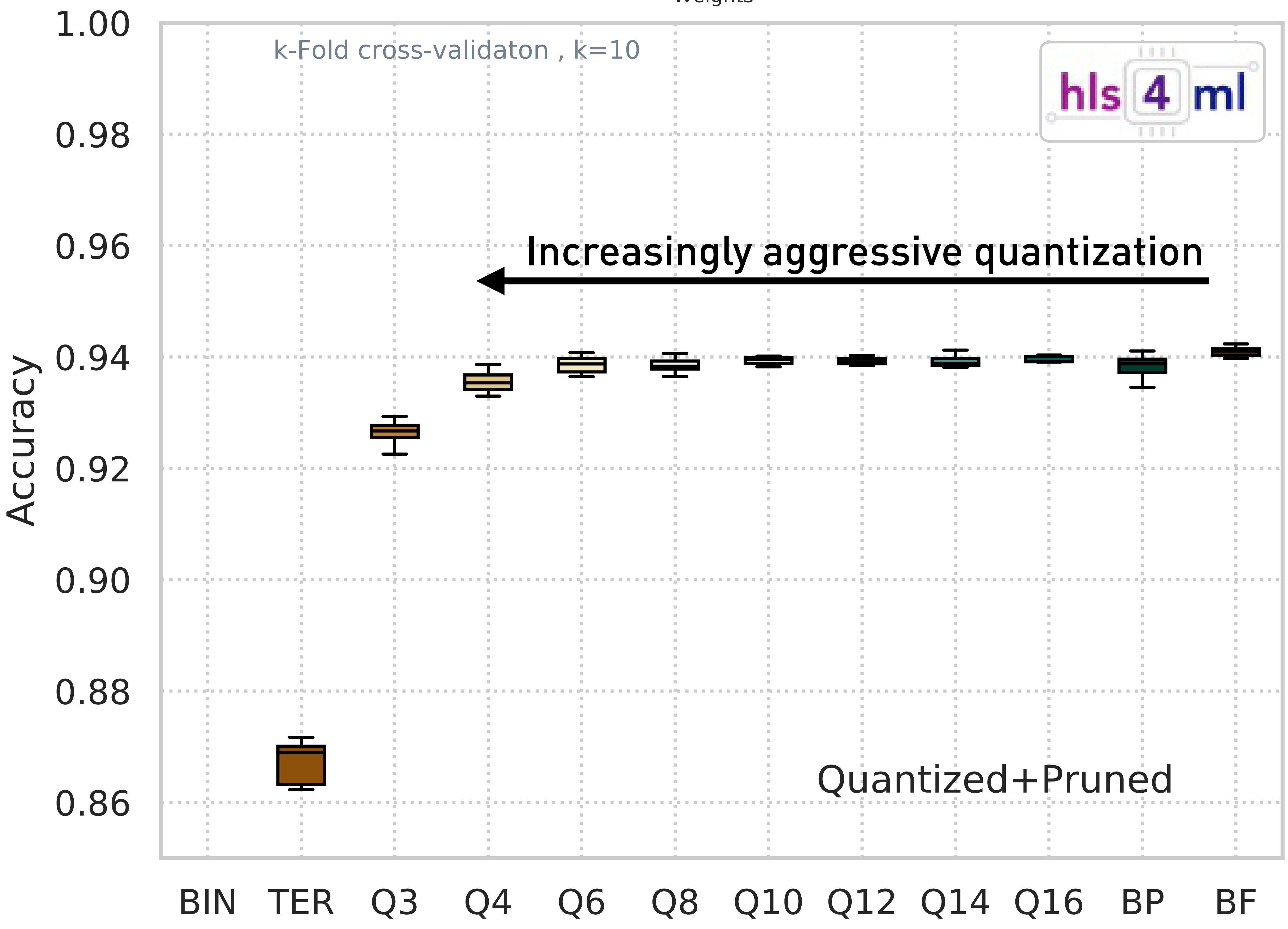
Several quantizers available
- Exponent quantization, e.g 'quantized_po2'
- Mantissa quantization, e.g 'quantized_bits'
- Both above, eg. 'ternary' and 'binary'

Full support for QKeras models in hls4ml
- Easy for users to design and deploy quantized, low-latency DNNs on chip!

Demonstrated in Vladimir's talk yesterday

arxiv: 2006.10159





QTools energy estimate

Some layers more accommodating for aggressive quantization, others require expensive arithmetic

- heterogeneous quantization (see Amir's talk)

For edge inference, need best possible quantization configuration for

- Highest accuracy ↑...
- ... and lowest resource consumption ↓

→ hyper-parameter scan over quantizers which considers energy and accuracy simultaneously

QTools energy estimate

Some layers more accommodating for aggressive quantization, others require expensive arithmetic

- heterogeneous quantization (see Amir's talk)

For edge inference, need best possible quantization configuration for

- Highest accuracy ↑...
- ... and lowest resource consumption ↓

→ hyper-parameter scan over quantizers which considers energy and accuracy simultaneously

QTools: Estimate QKeras model bit and energy consumption, assuming 45 nm Horowitz process

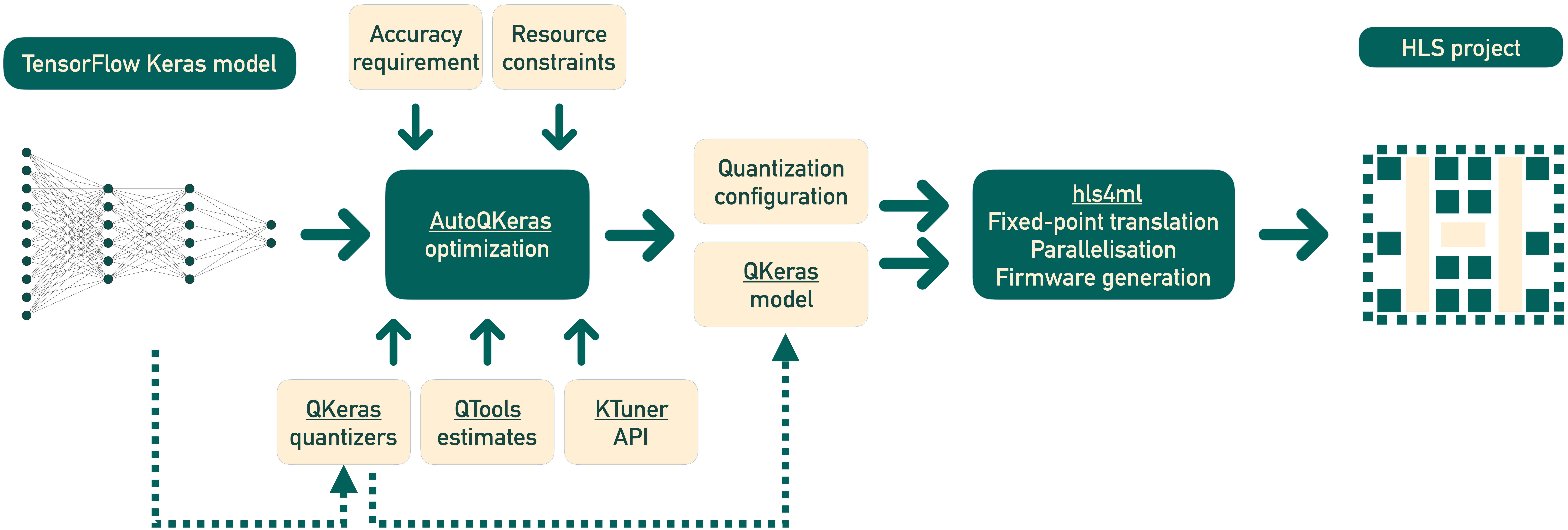
- Model size in bits
- Energy consumption in Watts

Model	Accuracy [%]	Per-layer energy consumption [pJ]								Total energy [μJ]	Total bits
		Dense	ReLU	Dense	ReLU	Dense	ReLU	Dense	Softmax		
BF	74.4	1735	53	3240	27	1630	27	281	11	0.00700	61446
Q6	74.8	794	23	1120	11	562	11	99	11	0.00263	26334

$$\text{Forgiving Factor} = 1 + \Delta_{accuracy} \times \log_{rate}\left(S \times \frac{Cost_{ref}}{Cost_{trial}}\right)$$

Maximize accuracy + minimizing cost in hyper parameter scan over quantizers: AutoQKeras

Workflow

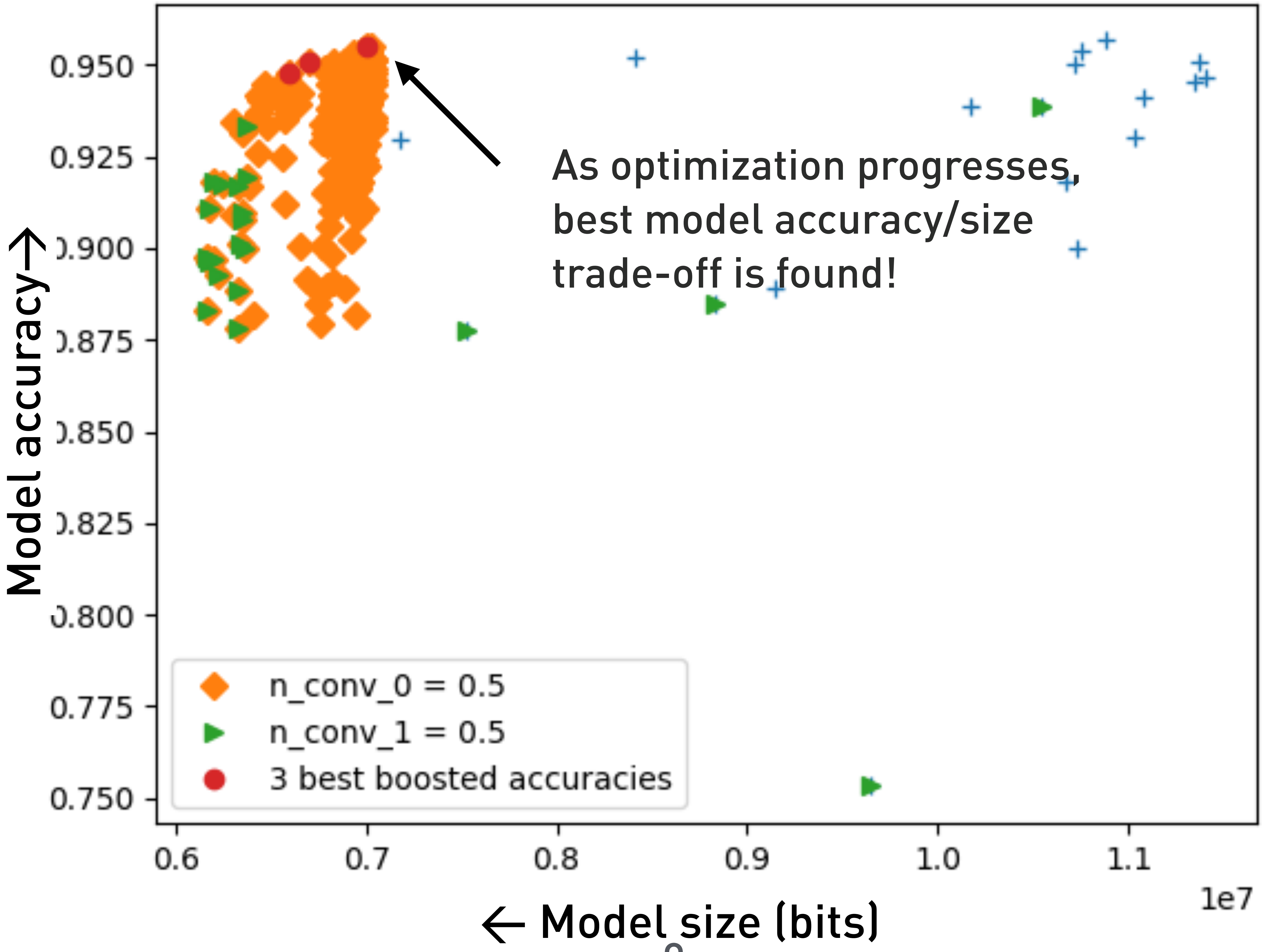

TensorFlow Keras model
Accuracy requirement
Resource constraints
AutoQKeras optimization
Quantization configuration
QKeras model
hls4ml
Fixed-point translation
Parallelisation
Firmware generation
HLS project
QKeras quantizers
QTools estimates
KTuner API

H. Linander, C. Petersson (Zenseact)

AutoQKeras

AutoQ Bayesian optimization at work!

- Simultaneously scan over quantizers and N filters(often less/more filters needed when quantizing)

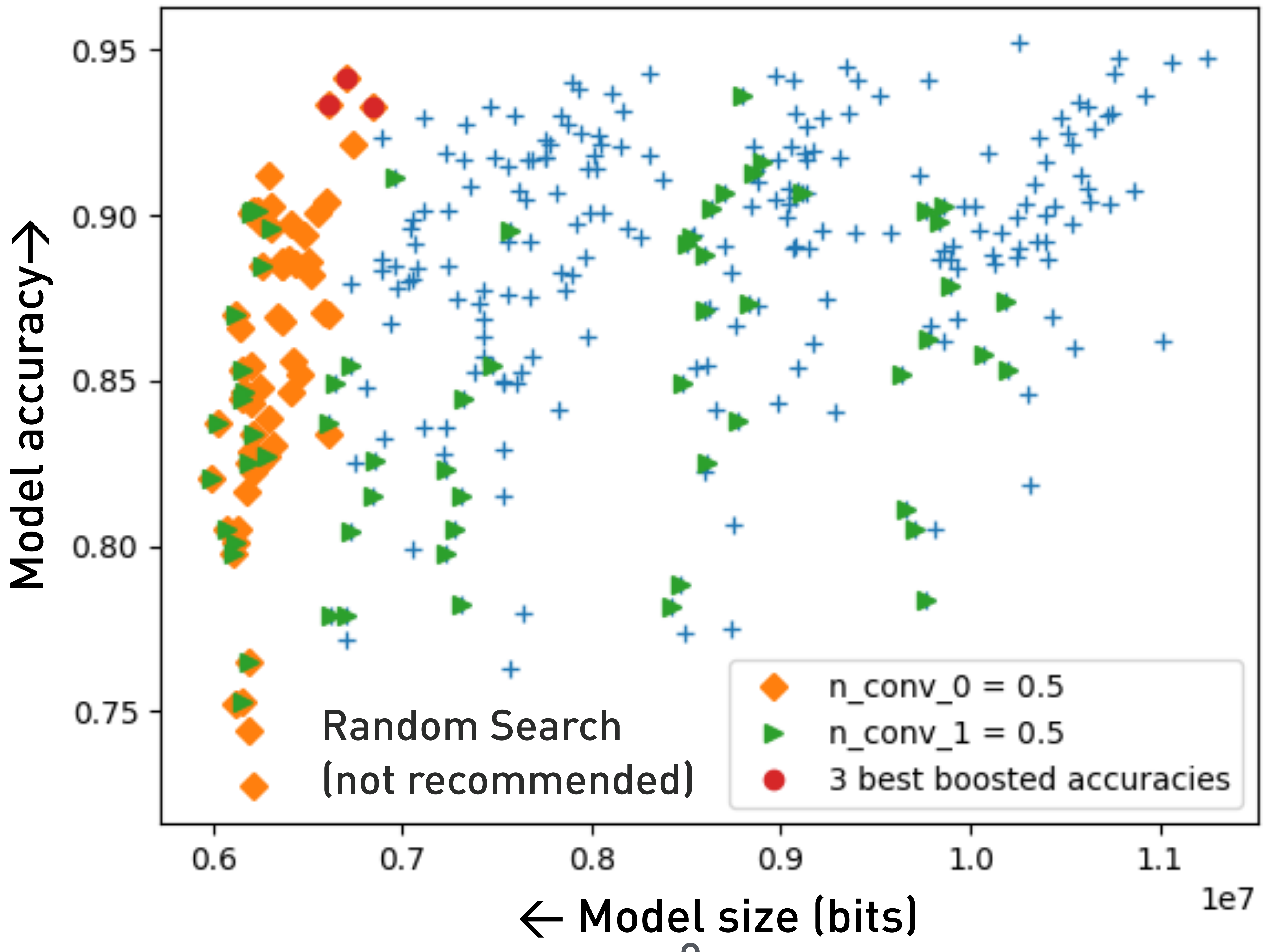


AutoQKeras

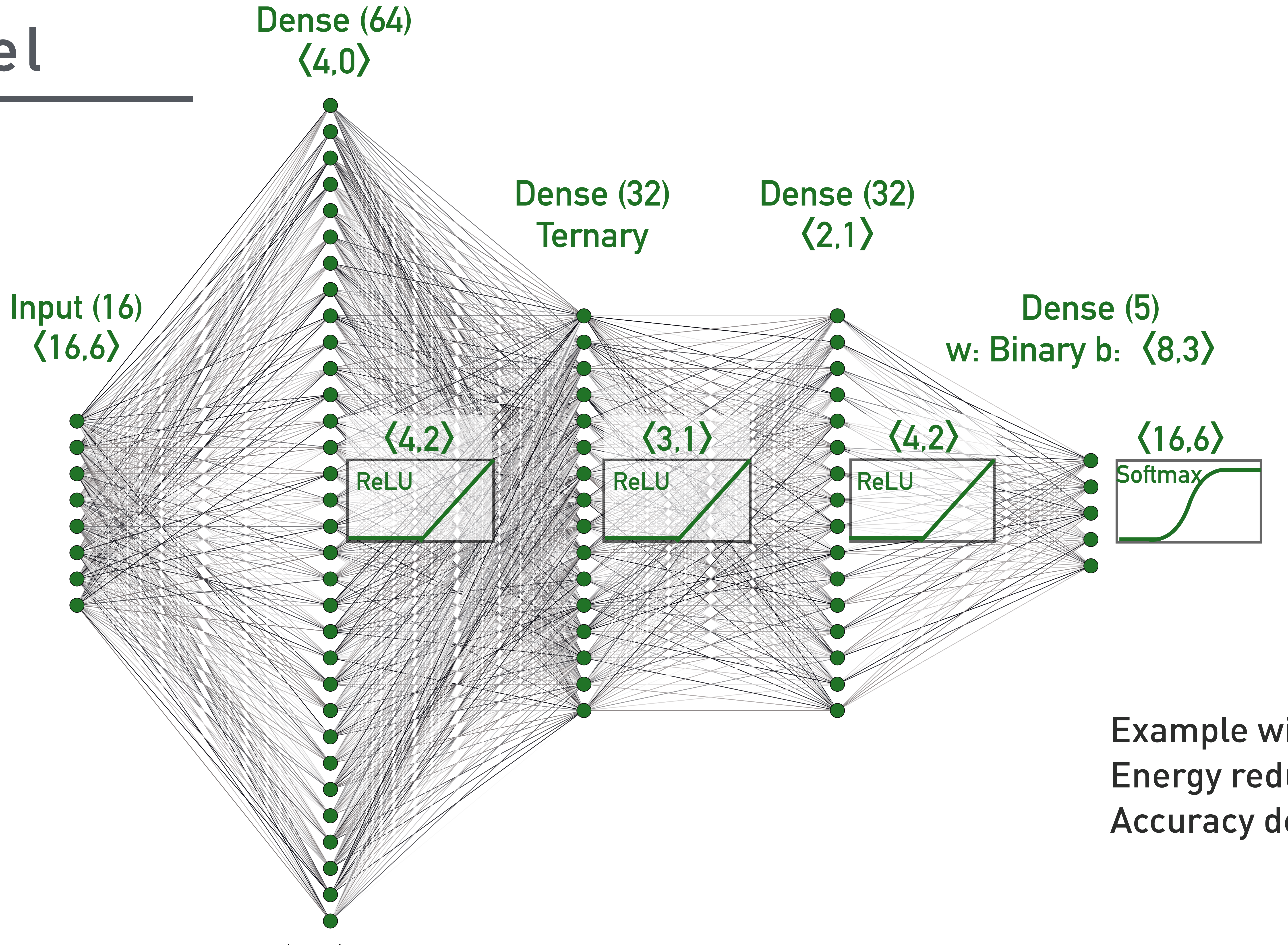
H. Linander, C. Petersson (Zenseact)

AutoQ Bayesian optimization at work!

- Simultaneously scan over quantizers and N filters(often less/more filters needed when quantizing)



AutoQ model

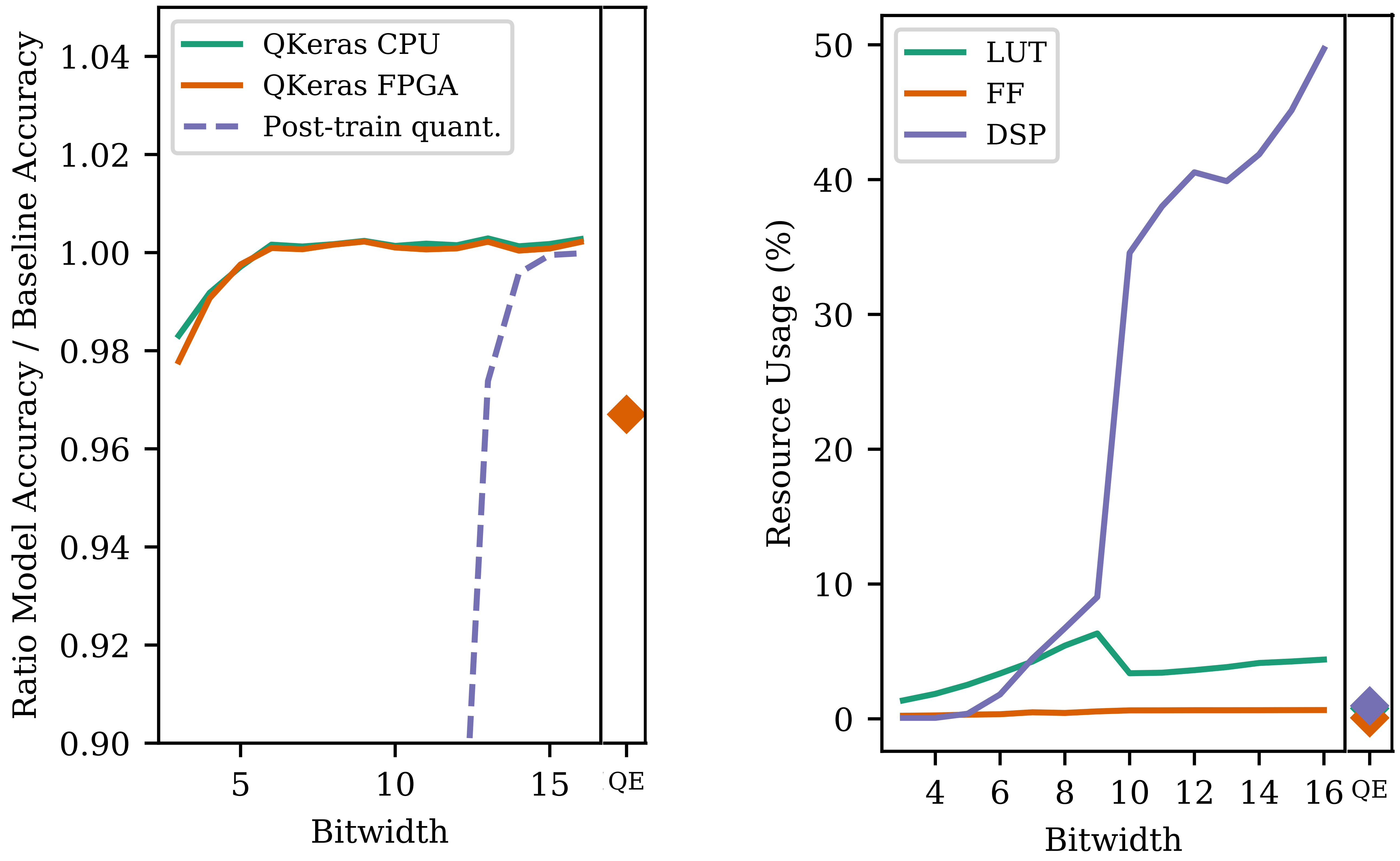


Example with target:
Energy reduction x4
Accuracy degradation max 5%

Model	Acc. [%]	Precision								Tot. energy [μJ]	Tot. bits
		Dense	ReLU	Dense	ReLU	Dense	ReLU	Dense	Softmax		
QE	72.3	⟨4, 0⟩	⟨4, 2⟩	Ternary	⟨3, 1⟩	⟨2, 1⟩	⟨4, 2⟩	w: Stoc. Bin. b: ⟨8, 3⟩	⟨16, 6⟩	0.00095	4728

FPGA performance

Multiplications move to LUTs at bit width <10.
Good, usually $O(10^3)$ more LUTs than DSPs



Model	Accuracy [%]	Latency [ns]	Latency [clock cycles]	DSP [%]	LUT [%]	FF [%]
BF	74.4	45	9	56.0 (1826)	5.2 (48321)	0.8 (20132)
Q6	74.8	55	11	1.8 (124)	3.4 (39782)	0.3 (8128)
QE	72.3	55	11	**1.0 (66)**	**0.8 (9149)**	0.1 (1781)

Summary

From TensorFlow Keras model to ultra-compressed, low-latency firmware in two steps with QKeras and hls4ml

- Nanosecond inference, x50 reduction in resources with little loss in model accuracy

Quantization-aware training and pruning (Vladimir's talk, TF Pruning API) are measures every application developer can take to simplify on-chip deployment

- Extremely useful for those designing edge inference engines (like DNN applications for HL-LHC)

pip install hls4ml
pip install git+https://github.com/google/qkeras.git@master
And join the hls4ml tutorial by Sioni on Thursday for hands-on QKeras+hls4ml experience!

thea.aarrestad@cern.ch