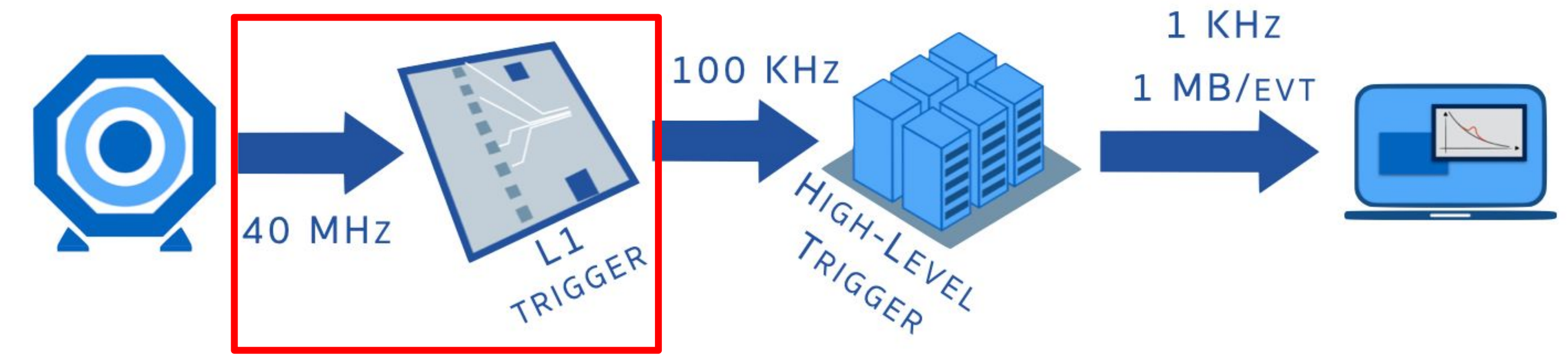
Machine Learning at 40 MHz with hls4ml

Adrian Alan Pol[1], Dylan Sheldon Rankin[2], Javier Mauricio Duarte[3], Jennifer Ngadiuba[4], Katya Govorkova[1], Maksymilian Graczyk[5], Maurizio Pierini[1], Nhan Tran[4], Nicolo Ghielmetti[1], Philip Coleman Harris[2], Sioni Paris Summers[1], Thea Aarrestad[1], Vladimir Loncar[1], Zhenbin Wu[6]
[1] CERN, [2] MIT, [3] UC San Diego, [4] Fermilab, [5] Imperial College, [6] University of Florida

Motivation

CMS experiment trigger pipeline



- Deploy ML algorithms very early
- Challenge: strict latency constraints!

Features

On-chip weights (registers or block RAM)

- Much faster access times ➞ lower latency

User controllable trade-off between resource usage and latency/throughput

- Tuned via “reuse factor”

Quantization with QKeras

- Binary/Ternary precision support

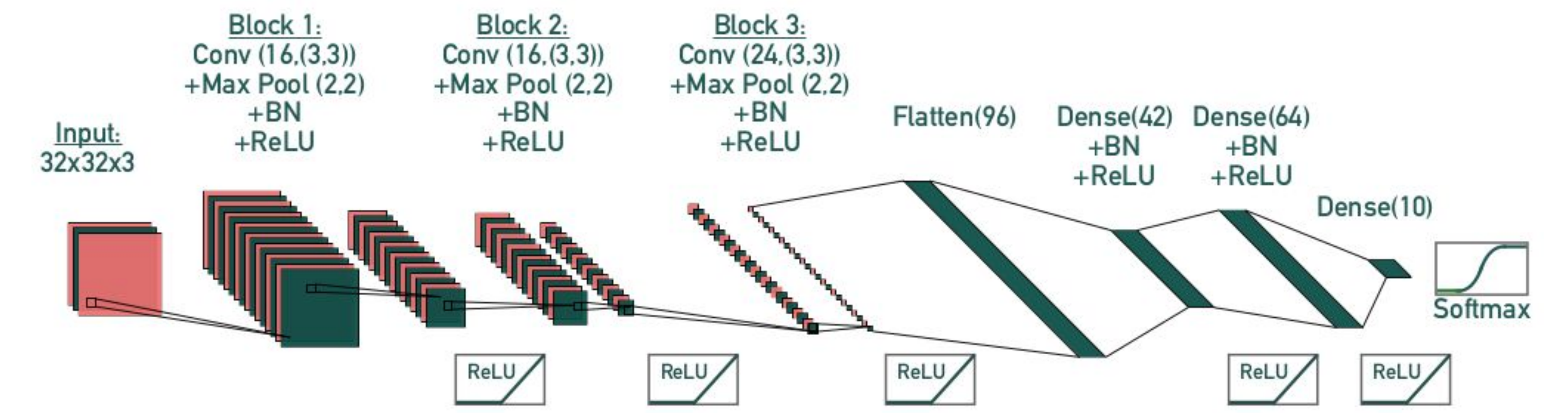
Supported network architectures:

- DNNs, CNNs, RNNs, GNNs

FPGA porting

Example: *digit classification with CNNs*

- Street-view house numbers dataset (SVHN)
- Model architecture (~13k weights):



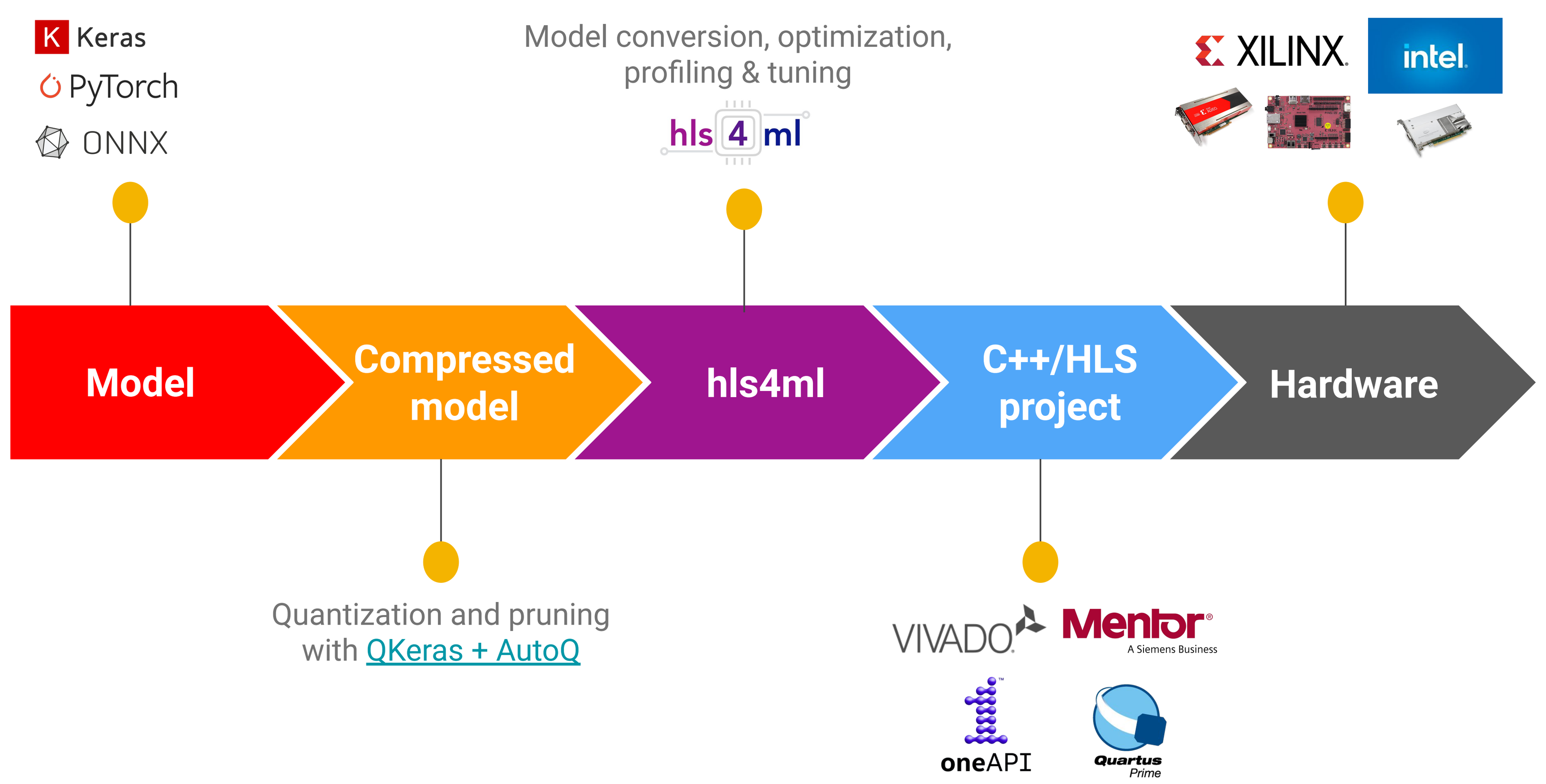
- Various configurations: Baseline Floating-point (BF), Baseline Pruned (BP), QKeras (Q) and QKeras Pruned (QP) models, the heterogeneously quantized models with AutoQ (AQ) and AutoQ Pruned (AQP)

Accuracy, resource consumption and latency for the Baseline Floating-point (BF) and Baseline Pruned (BP) models quantized to a bit width of 14, the QKeras (Q) and QKeras Pruned (QP) models quantized to a bit width of 7 and the heterogeneously quantized AutoQ (AQ) and AutoQ Pruned (AQP) models. The numbers in parentheses correspond to the total amount of resources used.

FPGA: Xilinx Virtex UltraScale+ VU9P

Model	Accuracy	DSP	LUT	FF	BRAM	Latency [cc]
BF 14-bit	0.87	93.2%	19.4%	3.4%	3%	5.2μs
BP 14-bit	0.93	48.9%	12.3%	2.8%	3%	5.2μs
Q 7-bit	0.94	2.5%	12.8%	1.5%	3%	5.2μs
QP 7-bit	0.94	2.5%	9.4%	1.4%	3%	5.2μs
AQ	0.88	1.0%	4.0%	0.6%	2%	5.3μs
AQP	0.88	1.0%	3.3%	0.6%	1%	5.3μs

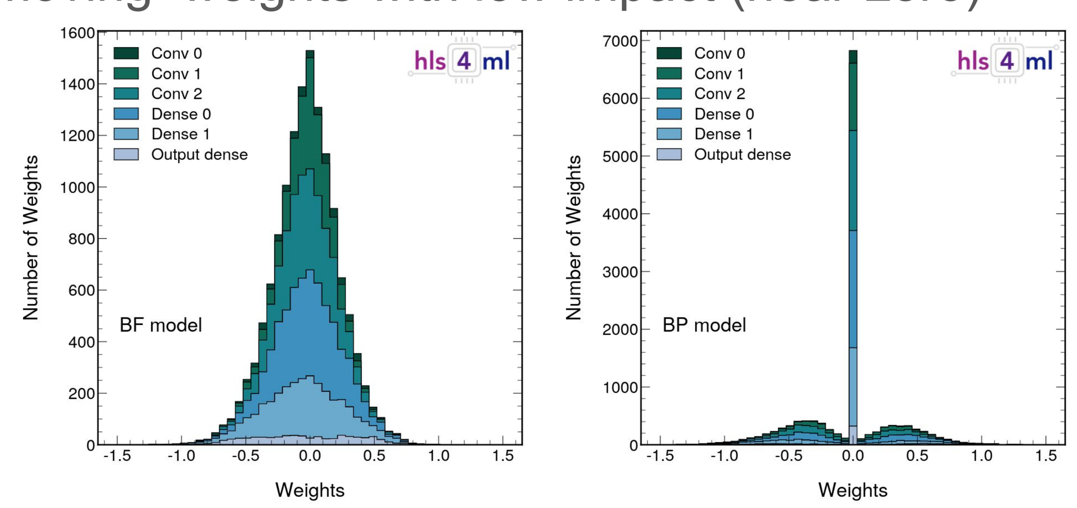
hls4ml pipeline



Model compression

- Pruning

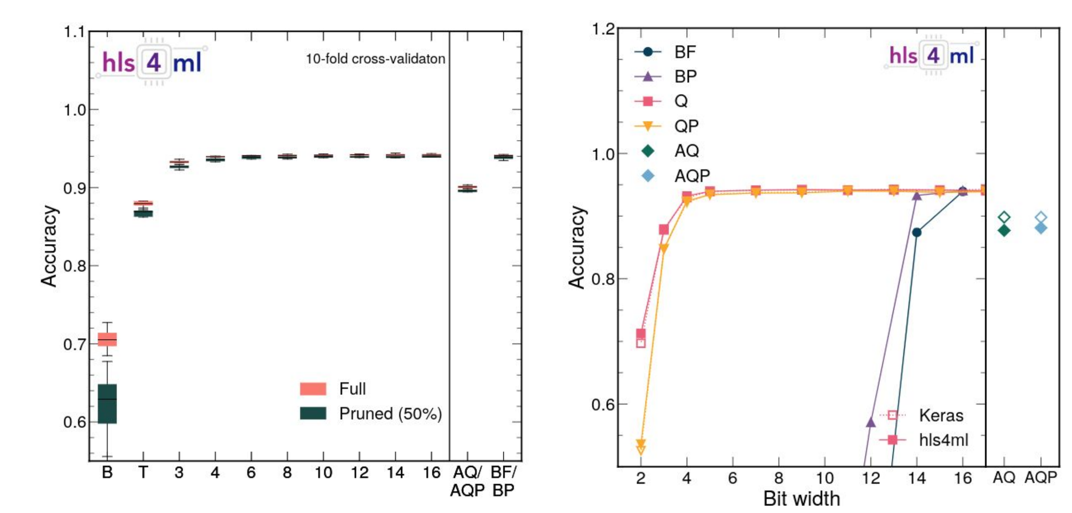
  Removing weights with low impact (near-zero)



- Quantization

  Reduce the numerical precision of the model weights

  Post-training or quantization-aware training



More information

- Website: https://fastmachinelearning.org/hls4ml
- Code: https://github.com/fastmachinelearning/hls4ml
- Tutorial: https://github.com/fastmachinelearning/hls4ml-tutorial
- Papers: https://fastmachinelearning.org/projects.html

Try it yourself

Installation: `pip install hls4ml`

Usage via Python API:

```
import hls4ml
my_model = … # build the model in Keras
my_config = hls4ml.utils.config_from_keras_model(my_model)
hls_model = hls4ml.converters.convert_from_keras_model(my_model,
    output_dir='my_hls_prj', hls_config=my_config)
report = hls_model.build()
```

Usage via CLI:

```
hls4ml config  --model my_model.onnx --dir my_hls_prj --output my_config.yml
hls4ml convert --config my_config.yml
hls4ml build   --project my_hls_prj --all
```