The Abdus Salam International Centre for Theoretical Physics

UNIVERSITÀ DEGLI STUDI DI TRIESTE

IAEA

Joint ICTP-IAEA School on Systems-on-Chip based on FPGA for Scientific Instrumentation and Reconfigurable Computing

FPGA for accelerating machine learning algorithms

Romina Soledad Molina

MLAB - STI

Trieste - Italy
2023

E-mail: rmolina@ictp.it

Outline

- Introduction.
- Machine learning (ML).
- Remarks from SOTA.
- ML and model compression techniques
- An end-to-end workflow to compress and deploy DNN on FPGA.
  - DNN training and compression.
  - Integration with a hardware synthesis tool for ML.
  - Hardware assessment framework.
- Applications.
  - Gamma/Neutron discrimination.
  - Image classification based on CNN.
  - Pulse shape discriminator for cosmic rays studies

Introduction

Introduction

SoC-based FPGA

Introduction

Low latency

SoC-based FPGA

Introduction

Low latency

Low power consumption

SoC-based FPGA

Introduction

Low latency

Low power consumption

High parallelism

SoC-based FPGA

Introduction

Low latency

Low power consumption

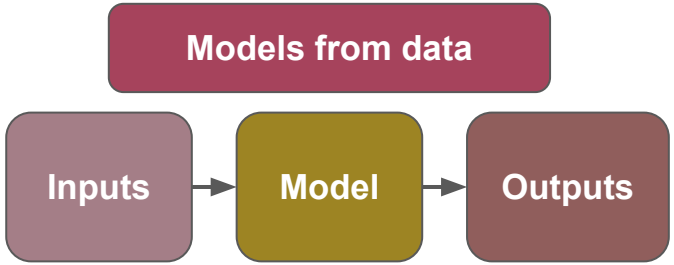
High parallelism

SoC-based FPGA

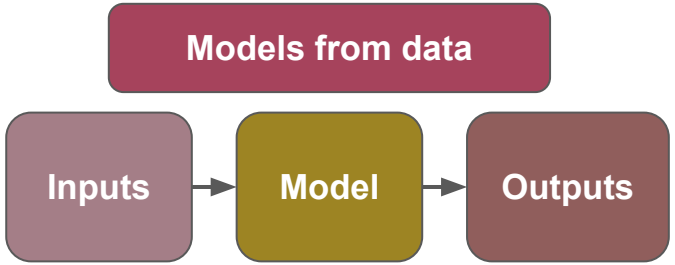
Resource-constrained devices

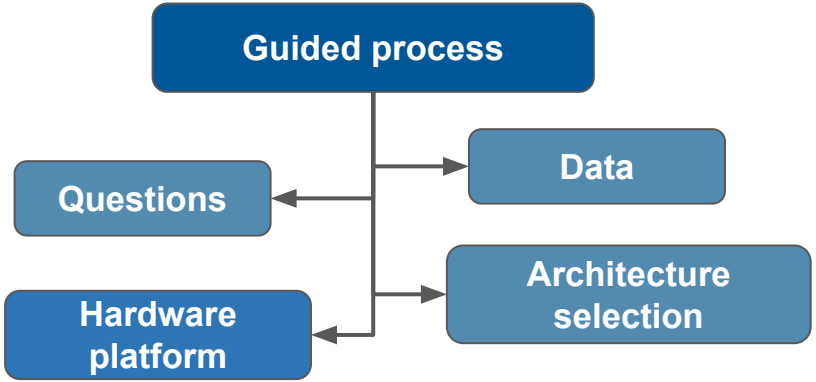
Machine learning

Machine learning

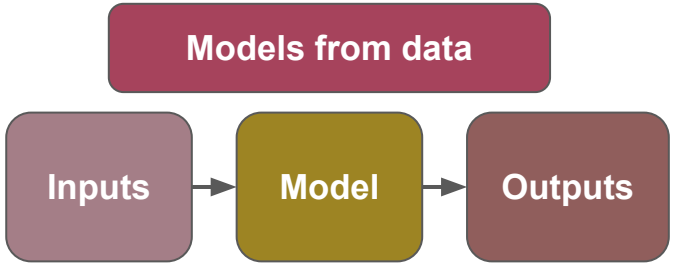


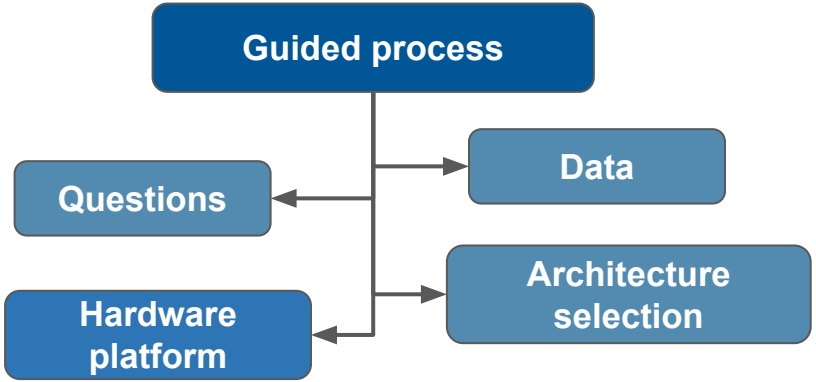
Machine learning

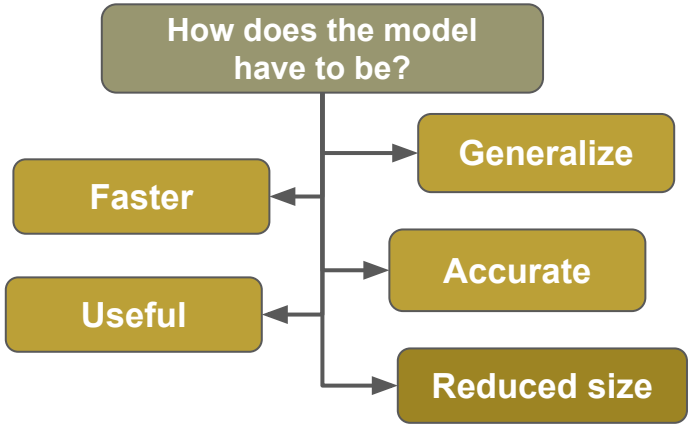




Machine learning







Remarks from the SOTA

Remarks from the SOTA

. Towards ML-based models implemented on resource-constrained devices.

Remarks from the SOTA

. Towards ML-based models implemented on resource-constrained devices.

. SOTA models: VGG16, MobileNet V2, BERT, U-Net, YOLO.

Remarks from the SOTA

. Towards ML-based models implemented on resource-constrained devices.

. SOTA models: VGG16, MobileNet V2, BERT, U-Net, YOLO.

. Compression: focused on pruning and quantization.

Remarks from the SOTA

. Towards ML-based models implemented on resource-constrained devices.

. SOTA models: VGG16, MobileNet V2, BERT, U-Net, YOLO.

. Compression: focused on pruning and quantization.

. Workflows addressing some parts of the development cycle.

Remarks from the SOTA

. Towards ML-based models implemented on resource-constrained devices.

. SOTA models: VGG16, MobileNet V2, BERT, U-Net, YOLO.

. Compression: focused on pruning and quantization.

. Workflows addressing some parts of the development cycle.

. Off-chip memory transactions.

Remarks from the SOTA

. Towards ML-based models implemented on resource-constrained devices.

. SOTA models: VGG16, MobileNet V2, BERT, U-Net, YOLO.

. Compression: focused on pruning and quantization.

. Workflows addressing some parts of the development cycle.

. Off-chip memory transactions.

Memory footprint and latency

Remarks from the SOTA

. Towards ML-based models implemented on resource-constrained devices.

. SOTA models: VGG16, MobileNet V2, BERT, U-Net, YOLO.

. Compression: focused on pruning and quantization.

. Workflows addressing some parts of the development cycle.

. Off-chip memory transactions.

Memory footprint and latency

Ensemble of compression techniques

Remarks from the SOTA

. Towards ML-based models implemented on resource-constrained devices.

. SOTA models: VGG16, MobileNet V2, BERT, U-Net, YOLO.

. Compression: focused on pruning and quantization.

. Workflows addressing some parts of the development cycle.

. Off-chip memory transactions.

Memory footprint and latency

Ensemble of compression techniques

On-chip memory deployment

Remarks from the SOTA

. Towards ML-based models implemented on resource-constrained devices.

. SOTA models: VGG16, MobileNet V2, BERT, U-Net, YOLO.

. Compression: focused on pruning and quantization.

. Workflows addressing some parts of the development cycle.

. Off-chip memory transactions.

Memory footprint and latency

Ensemble of compression techniques

On-chip memory deployment

Productivity

End-to-end workflow

ML and model compression techniques

ML and model compression techniques
for reconfigurable hardware accelerators

Ensemble of compression techniques - Exploration of the interplay between:

ML and model compression techniques

for reconfigurable hardware accelerators

Ensemble of compression techniques - Exploration of the interplay between:

Pruning	Quantization	Knowledge distillation

ML and model compression techniques

for reconfigurable hardware accelerators

Ensemble of compression techniques - Exploration of the interplay between:

Pruning

Remove neurons and connections.

Quantization

Knowledge distillation

ML and model compression techniques

for reconfigurable hardware accelerators

Ensemble of compression techniques - Exploration of the interplay between:

Pruning

Remove neurons and connections.

Quantization

Selection of the number of bits to represent the weights and bias.

Knowledge distillation

ML and model compression techniques

for reconfigurable hardware accelerators

Ensemble of compression techniques - Exploration of the interplay between:

Pruning	Quantization	Knowledge distillation
Remove neurons and connections.	**Selection of the number of bits to represent the weights and bias.**	**Transfers the knowledge from a teacher network to a smaller and faster target network.**

ML and model compression techniques

for reconfigurable hardware accelerators

Ensemble of compression techniques - Exploration of the interplay between:

Pruning	Quantization	Knowledge distillation
Remove neurons and connections.	Selection of the number of bits to represent the weights and bias.	Transfers the knowledge from a teacher network to a smaller and faster target network.

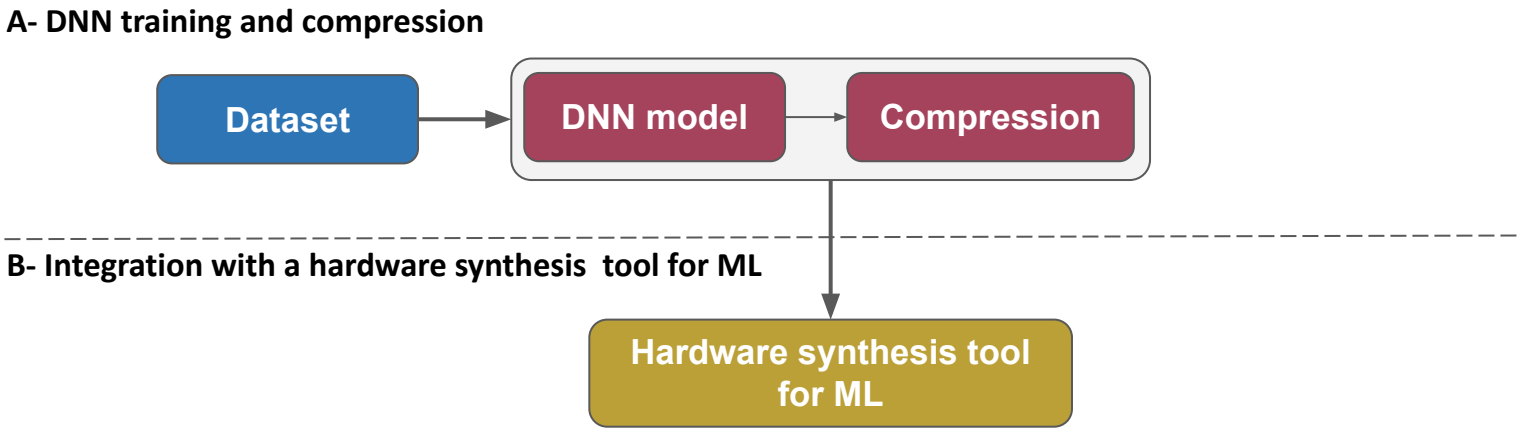
Fully on-chip deployment

An end-to-end workflow to efficiently compress and deploy DNN on SoC/FPGA

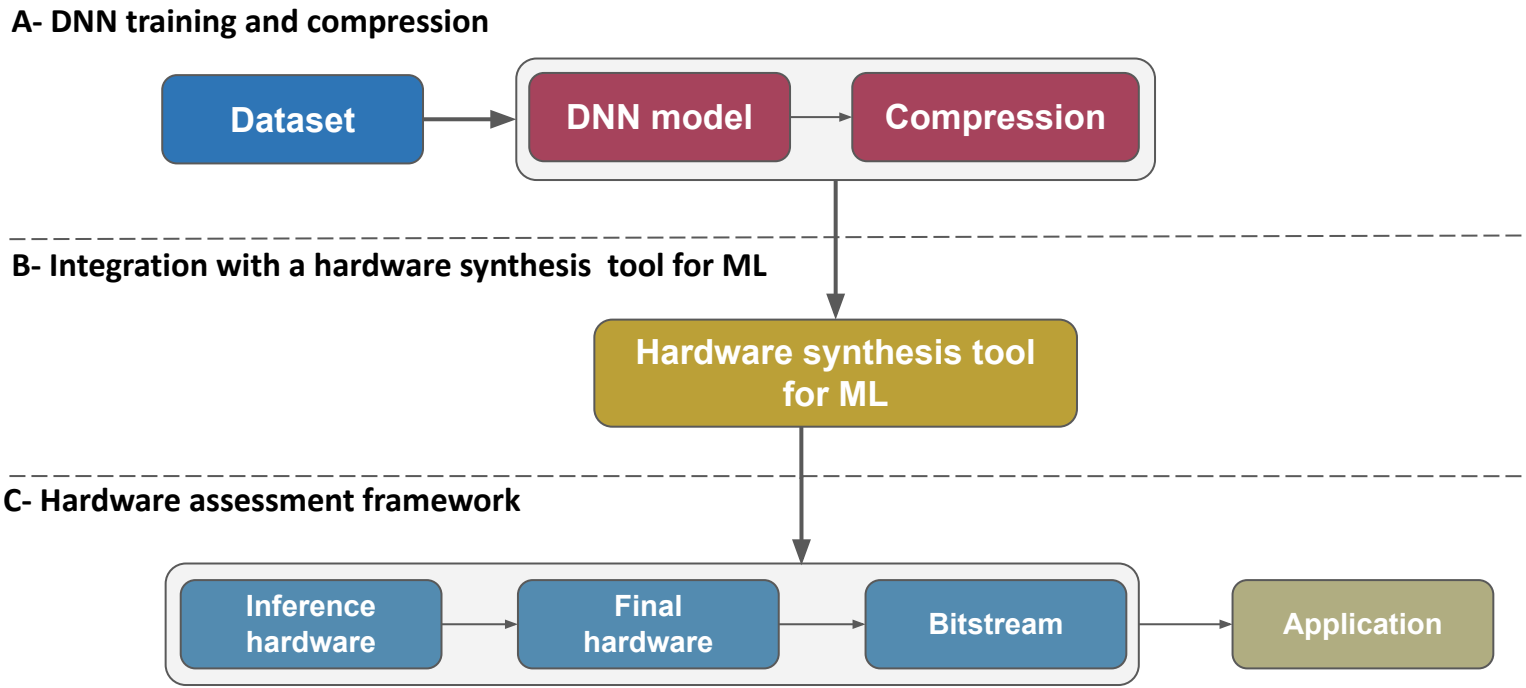
End-to-end workflow

A- DNN training and compression

End-to-end workflow



End-to-end workflow

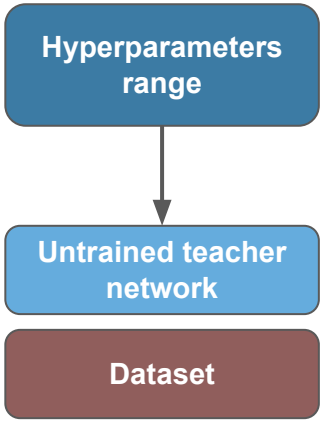


Available at https://github.com/RomiSolMolina/workflowCompressionML

A. DNN training and compression

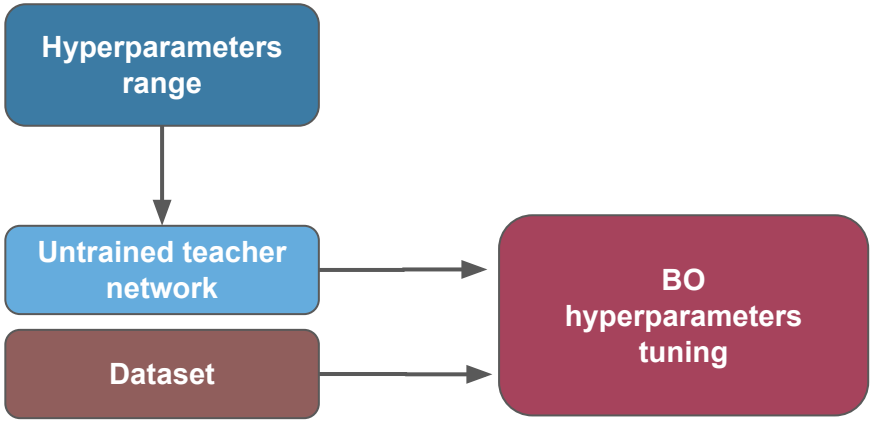
DNN training and compression

Stage 1 - Teacher training



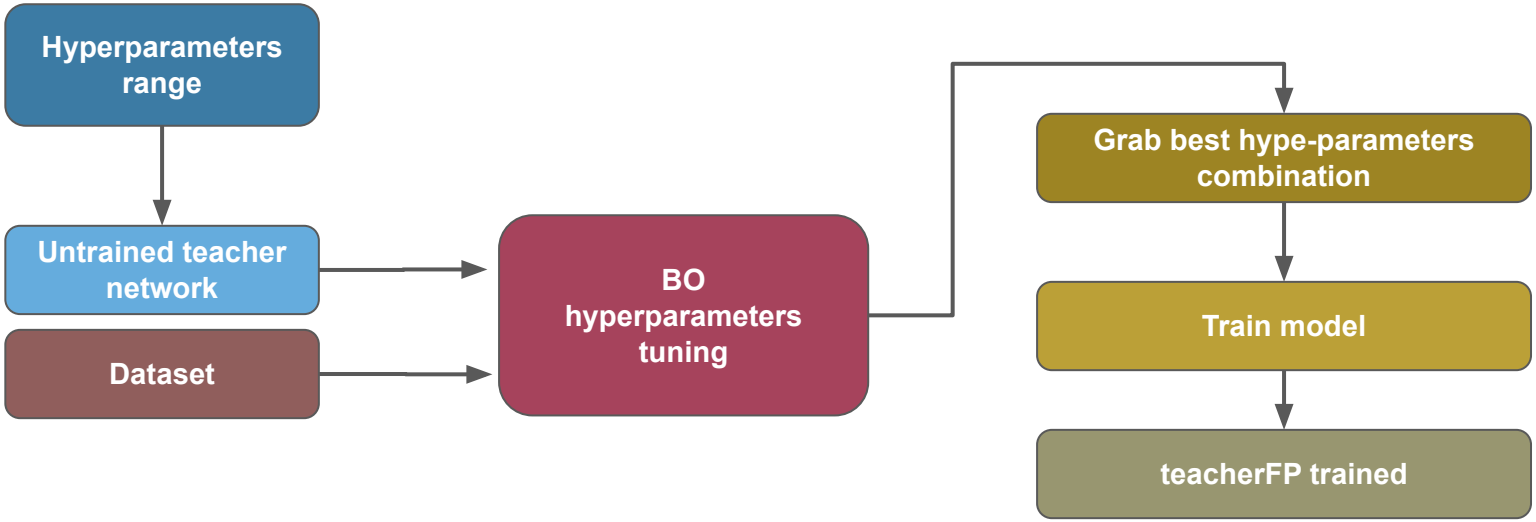
DNN training and compression

Stage 1 - Teacher training



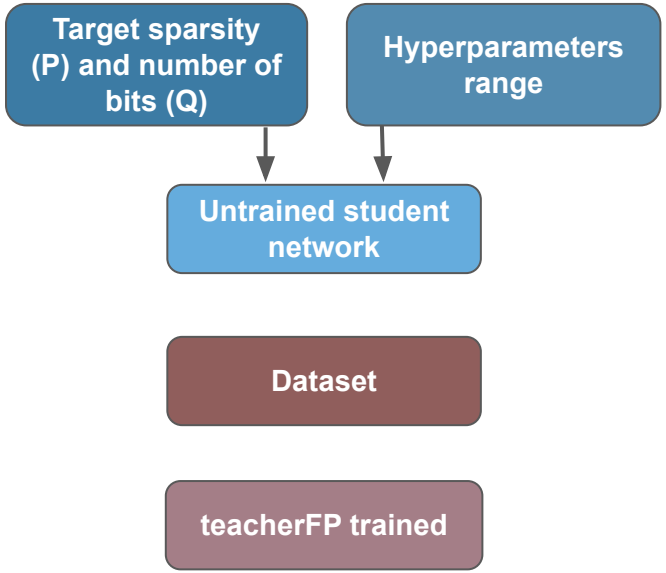
DNN training and compression

Stage 1 - Teacher training



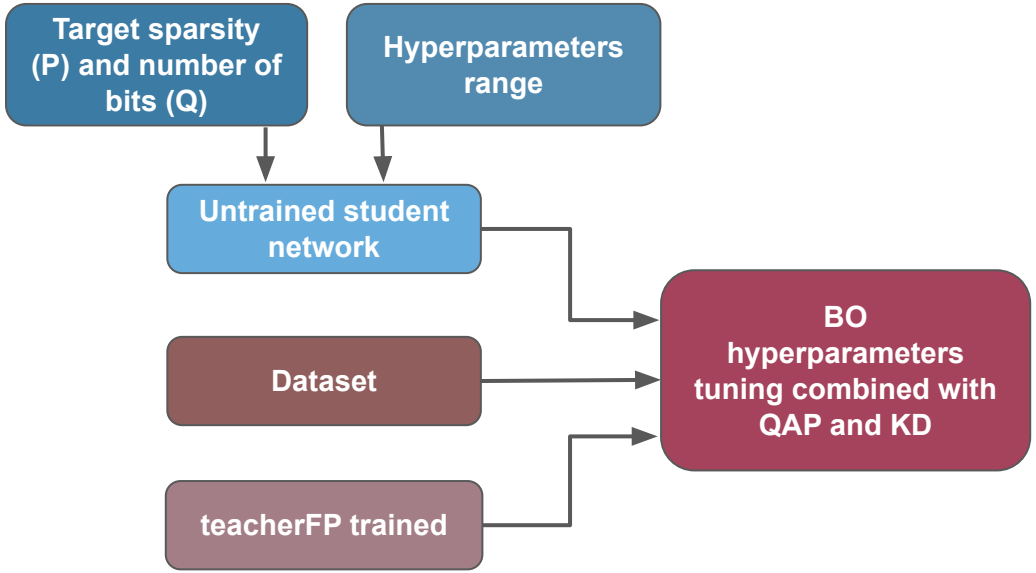
DNN training and compression

Stage 2 - Student training



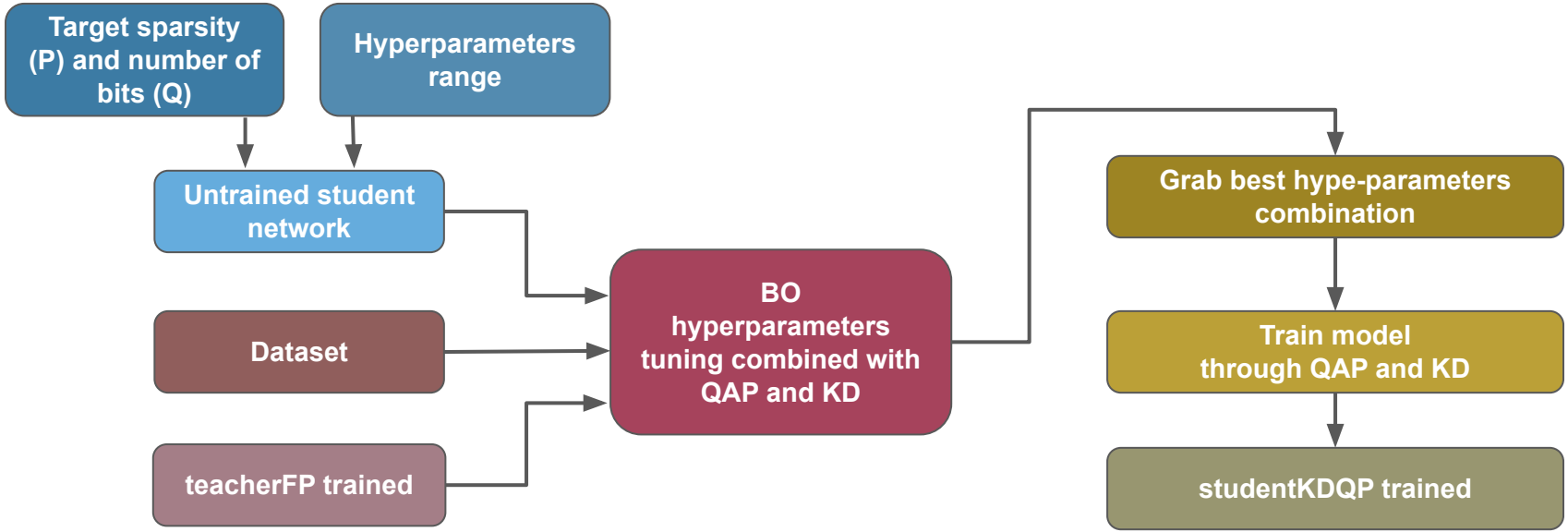
DNN training and compression

Stage 2 - Student training



DNN training and compression

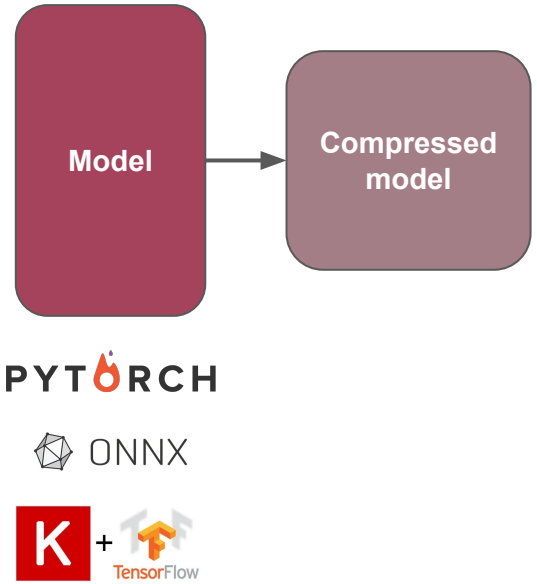
Stage 2 - Student training



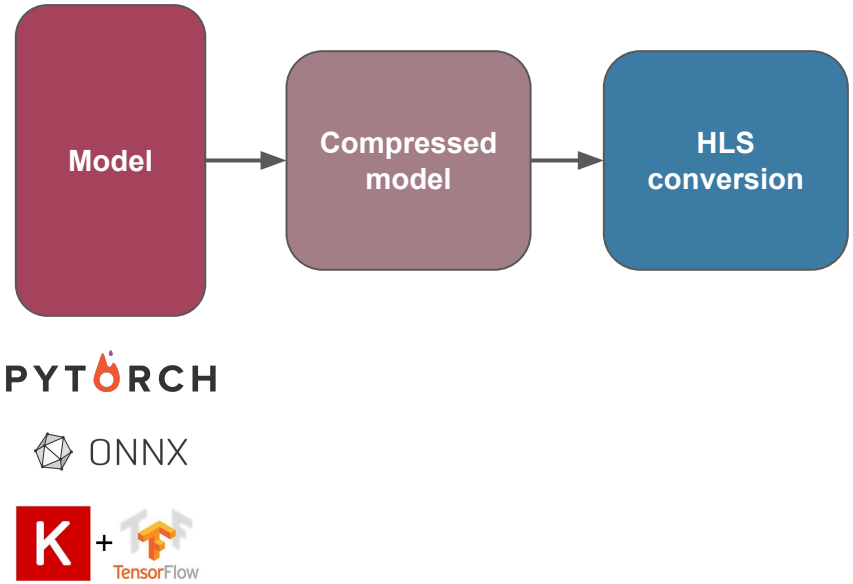
B. Integration with a hardware synthesis tool for ML

Integration with a hardware synthesis tool for ML

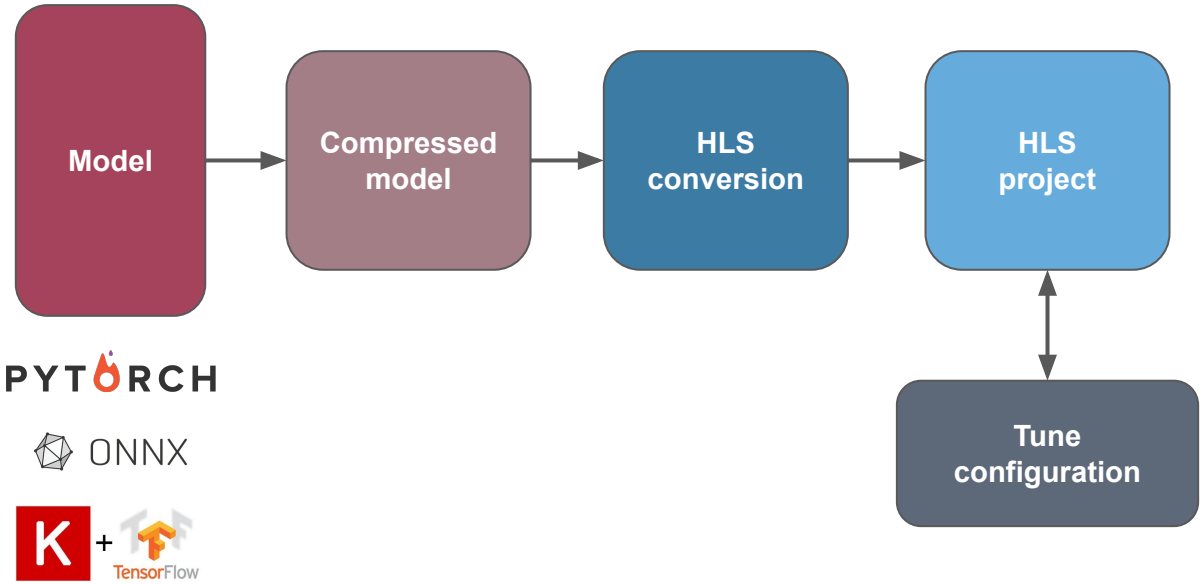
Integration with a hardware synthesis tool for ML



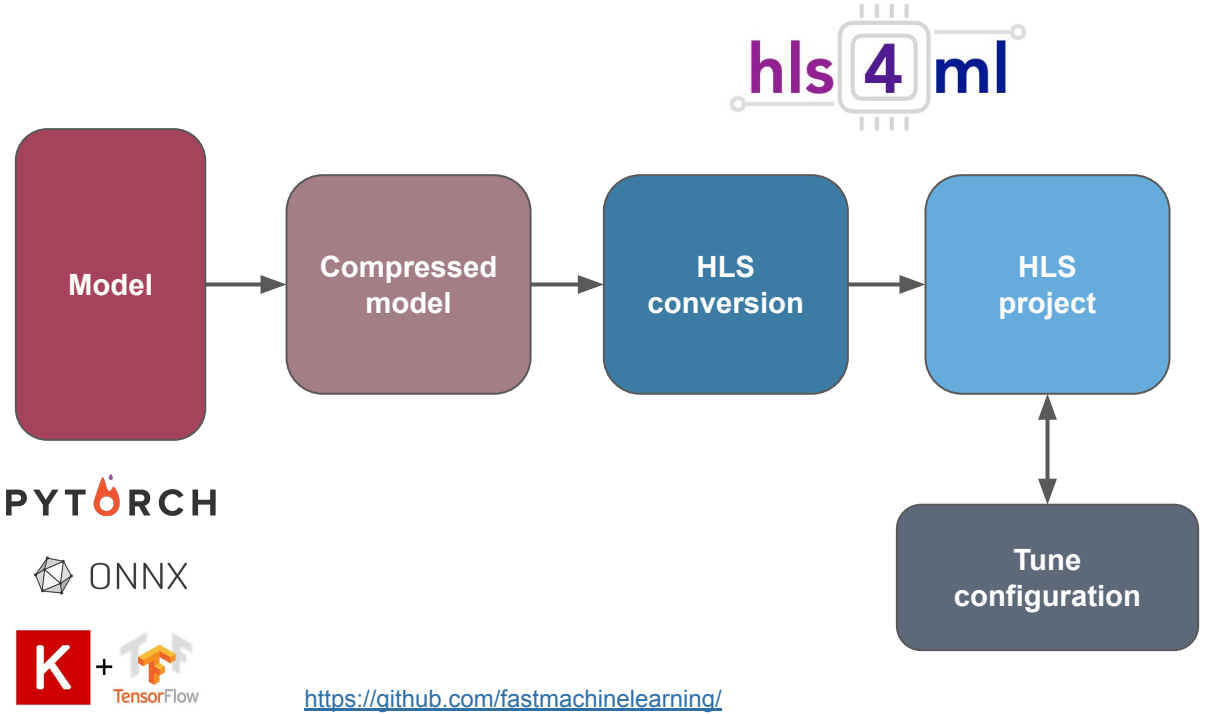
Integration with a hardware synthesis tool for ML



Integration with a hardware synthesis tool for ML

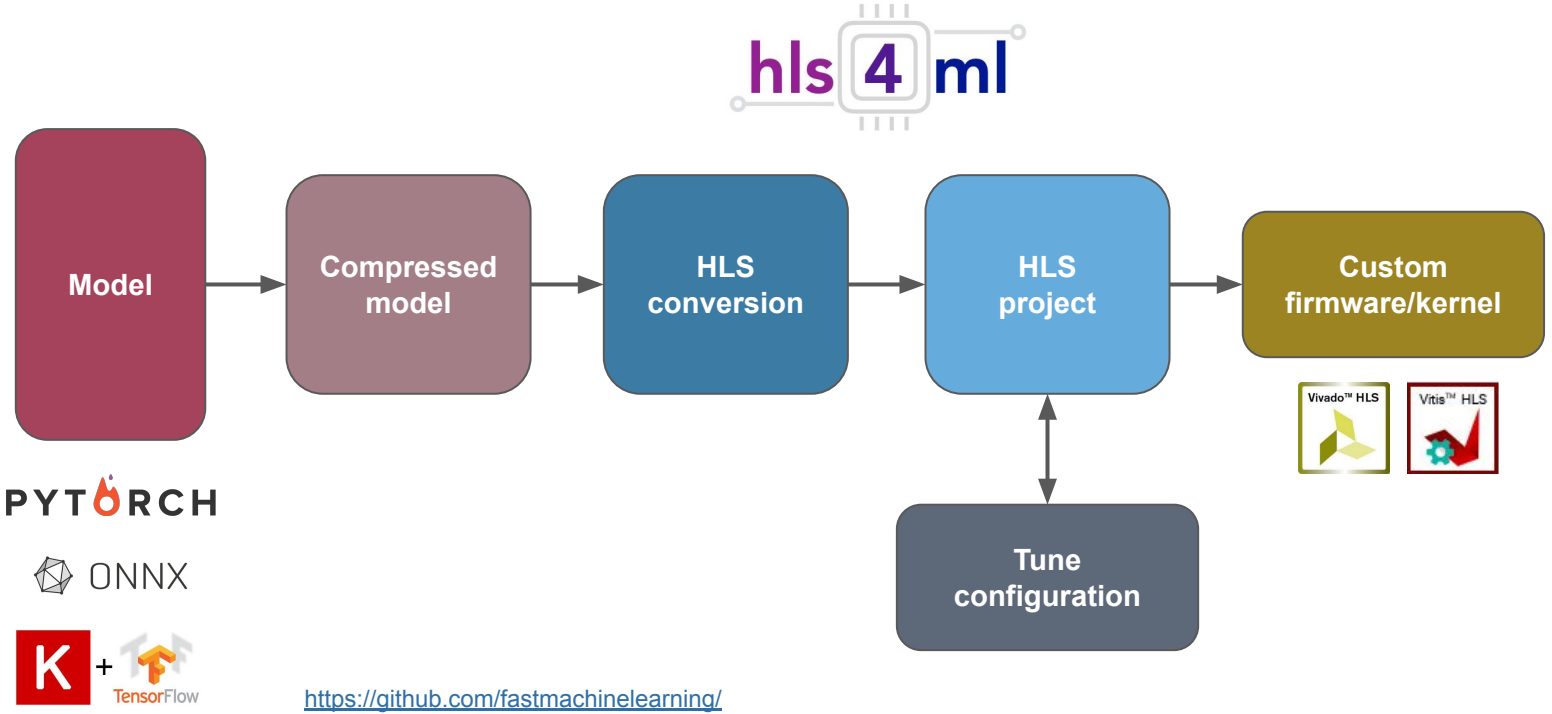


Integration with a hardware synthesis tool for ML



https://github.com/fastmachinelearning/

Integration with a hardware synthesis tool for ML



https://github.com/fastmachinelearning/

Integration with a hardware synthesis tool for ML

ML framework support:

- (Q)Keras
- PyTorch (limited)
- (Q)ONNX (in development)

https://fastmachinelearning.org/hls4ml/

Integration with a hardware synthesis tool for ML

ML framework support:

- (Q)Keras
- PyTorch (limited)
- (Q)ONNX (in development)

Neural networks architectures:

- Fully Connected NN
- Convolutional NN
- Recurrent NN
- Graph NN

https://fastmachinelearning.org/hls4ml/

Integration with a hardware synthesis tool for ML

ML framework support:

- (Q)Keras
- PyTorch (limited)
- (Q)ONNX (in development)

Neural networks architectures:

- Fully Connected NN
- Convolutional NN
- Recurrent NN
- Graph NN

HLS backends:

- Vivado HLS
- Intel HLS
- Vitis HLS (experimental)

https://fastmachinelearning.org/hls4ml/

Integration with a hardware synthesis tool for ML

Python integration

hls4ml

```
1 from tensorflow.keras.models import load_model
2 from sklearn.metrics import accuracy_score
3 model = load_model('model_keras_MLP.h5')
4 model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
fc1 (Dense)                  (None, 60)                3900
_________________________________________________________________
relu1 (Activation)           (None, 60)                0
_________________________________________________________________
fc0 (Dense)                  (None, 40)                2440
_________________________________________________________________
relu0 (Activation)           (None, 40)                0
_________________________________________________________________
fc2 (Dense)                  (None, 30)                1230
_________________________________________________________________
relu2 (Activation)           (None, 30)                0
_________________________________________________________________
fc3 (Dense)                  (None, 10)                310
```

Integration with a hardware synthesis tool for ML

Python integration

hls4ml

```
import hls4ml
import plotting

config = hls4ml.utils.config_from_keras_model(teacherMLP, granularity='model')
config['Model'] = {'Precision' : 'ap_fixed<24,16>', 'ReuseFactor' : 1, 'Strategy' : 'Latency'}
print("-----------------------------------")
plotting.print_dict(config)
print("-----------------------------------")
hls_model = hls4ml.converters.convert_from_keras_model(teacherMLP,
                                                       hls_config=config,
                                                       output_dir='model_3/hls_model'
                                                      )
hls_model.compile()
```

Integration with a hardware synthesis tool for ML

Python integration

hls4ml

```
import hls4ml
import plotting
hls4ml.model.optimizer.OutputRoundingSaturationMode.layers = ['Activation']
hls4ml.model.optimizer.OutputRoundingSaturationMode.rounding_mode = 'AP_RND'
hls4ml.model.optimizer.OutputRoundingSaturationMode.saturation_mode = 'AP_SAT'

config = hls4ml.utils.config_from_keras_model(model, granularity='name')

config['Model'] = {'Precision' : 'ap_fixed<17,16>', 'ReuseFactor' : 1, 'Strategy' : 'Latency'}

config['LayerName']['fc1']['Precision']['weight'] = 'ap_fixed<9, 1>'

config['LayerName']['softmax']['Precision'] = 'ap_fixed<32,15>'

print("-----------------------------------")
plotting.print_dict(config)
print("-----------------------------------")
hls_model = hls4ml.converters.convert_from_keras_model(model,
                                                       hls_config=config,
                                                       output_dir='model_3/MLP_student_smr3765'
                                                      )

hls_model.compile()
```

Integration with a hardware synthesis tool for ML

Network description generated inside HLS project

hls4ml

```
63
64    layer3_t layer3_out[N_LAYER_3];
65    #pragma HLS ARRAY_PARTITION variable=layer3_out complete dim=0
66    nnet::dense_latency<input2_t, layer3_t, config3>(input1, layer3_out, w3, b3);
67
68    layer5_t layer5_out[N_LAYER_3];
69    #pragma HLS ARRAY_PARTITION variable=layer5_out complete dim=0
70    nnet::relu<layer3_t, layer5_t, relu_config5>(layer3_out, layer5_out);
71
72    layer6_t layer6_out[N_LAYER_6];
73    #pragma HLS ARRAY_PARTITION variable=layer6_out complete dim=0
74    nnet::dense_latency<layer5_t, layer6_t, config6>(layer5_out, layer6_out, w6, b6);
75
76    layer8_t layer8_out[N_LAYER_6];
77    #pragma HLS ARRAY_PARTITION variable=layer8_out complete dim=0
78    nnet::relu<layer6_t, layer8_t, relu_config8>(layer6_out, layer8_out);
79
80    layer9_t layer9_out[N_LAYER_9];
81    #pragma HLS ARRAY_PARTITION variable=layer9_out complete dim=0
82    nnet::dense_latency<layer8_t, layer9_t, config9>(layer8_out, layer9_out, w9, b9);
83
84    layer11_t layer11_out[N_LAYER_9];
85    #pragma HLS ARRAY_PARTITION variable=layer11_out complete dim=0
86    nnet::relu<layer9_t, layer11_t, relu_config11>(layer9_out, layer11_out);
87
88    layer12_t layer12_out[N_LAYER_12];
89    #pragma HLS ARRAY_PARTITION variable=layer12_out complete dim=0
90    nnet::dense_latency<layer11_t, layer12_t, config12>(layer11_out, layer12_out, w12, b12);
91
92    layer14_t layer14_out[N_LAYER_12];
93    #pragma HLS ARRAY_PARTITION variable=layer14_out complete dim=0
94    nnet::relu<layer12_t, layer14_t, relu_config14>(layer12_out, layer14_out);
95
96    layer15_t layer15_out[N_LAYER_15];
97    #pragma HLS ARRAY_PARTITION variable=layer15_out complete dim=0
98    nnet::dense_latency<layer14_t, layer15_t, config15>(layer14_out, layer15_out, w15, b15);
99
100   nnet::softmax<layer15_t, result_t, softmax_config17>(layer15_out, layer17_out);
101
```

Integration with a hardware synthesis tool for ML

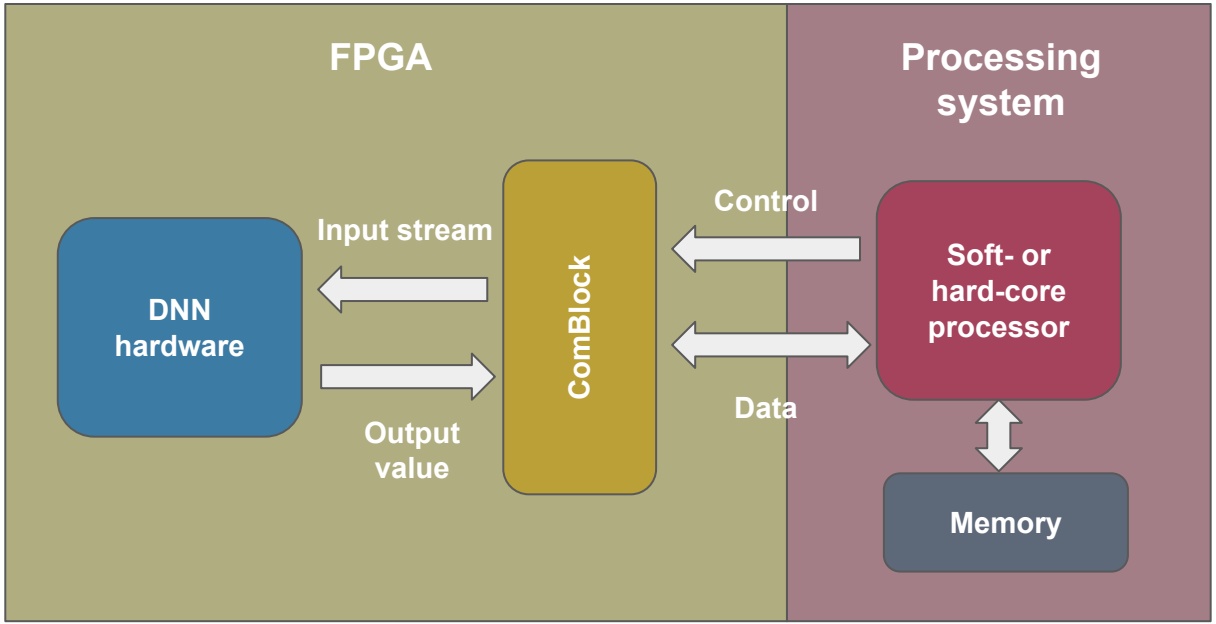
QKeras for quantization-aware training

hls4ml

```
# MLP architecture
# Create the student QKERAS
studentQ_MLP = keras.Sequential(
    [
        Input(shape=(30,)),
        QDense(20, name='fc1',
                kernel_quantizer=quantized_bits(9,1,alpha=1), bias_quantizer=quantized_bits(23,15,alpha=1)),
        QActivation(activation=quantized_relu(16,15), name='relu1'),
        QDense(10, name='fc2',
                kernel_quantizer=quantized_bits(9,1,alpha=1), bias_quantizer=quantized_bits(23,15,alpha=1)),
        QActivation(activation=quantized_relu(16,15), name='relu2'),
        QDense(10, name='fc6',
                kernel_quantizer=quantized_bits(9,1,alpha=1), bias_quantizer=quantized_bits(23,15,alpha=1)),
        QActivation(activation=quantized_relu(16,15), name='relu3'),

        QDense(4, name='output',
                kernel_quantizer=quantized_bits(32,15,alpha=1), bias_quantizer=quantized_bits(32,15,alpha=1)),
        Activation(activation='softmax', name='softmax')


    ],
    name="student",
)

print_qstats(studentQ_MLP)
```

C. Hardware assessment framework

Hardware assessment framework



Applications

ML and model compression techniques for SoC/FPGA

Applications

Gamma/neutron discrimination

ML and model compression techniques for SoC/FPGA

Applications

Gamma/neutron discrimination

Pest classification in fruit crops

ML and model compression techniques for SoC/FPGA

Applications

Gamma/neutron discrimination

Pest classification in fruit crops

Pulse shape discriminator for cosmic rays studies

Gamma/Neutron discrimination

ML and model compression techniques for SoC/FPGA

Applications

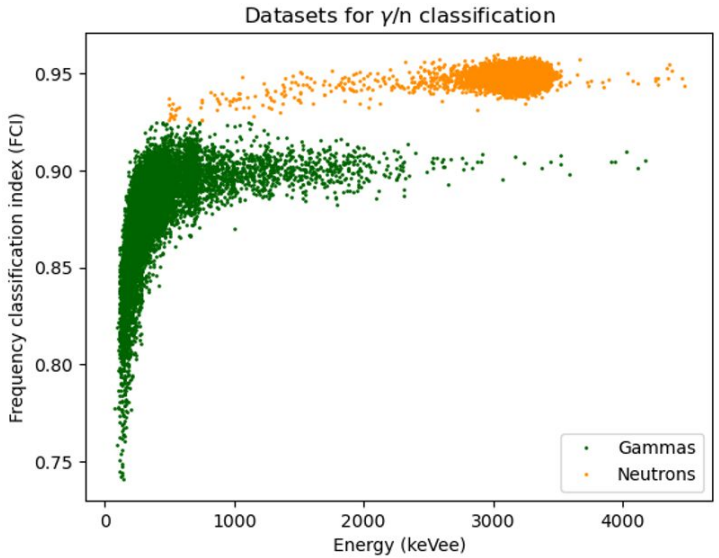
Gamma/neutron discrimination

- Tagged dataset of **gamma and neutron events** from Deuterium-Deuterium (DD) and Deuterium-Tritium (DT) generators.
- The dataset was recorded at the **Neutron Science Facility (NSF) of the Nuclear Science and Instrumentation Laboratory (NSIL), IAEA.**
- The detector is based on a small **CLYC** (Cs2LiYCl6:Ce) crystal (0.5 in diameter by 30 mm length) coupled to a 4-element SiPM array.
- The data were **sampled at 4 GSPS with 10-bits resolution** using a CAEN DT5761 digitizer.
- **The total gamma and neutron events in this dataset are 10913 and 27696, respectively.**

ML and model compression techniques for SoC/FPGA

Applications

Gamma/neutron discrimination



Morales, I. R., Crespo, M. L., Bogovac, M., Cicuttin, A., Kanaki, K., & Carrato, S. (2023). Gamma/neutron classification with SiPM CLYC detectors using frequency-domain analysis for embedded real-time applications. Nuclear Engineering and Technology.

Dataset from https://doi.org/10.5281/zenodo.8037059

ML and model compression techniques for SoC/FPGA

Applications

Gamma/neutron discrimination

Subsampling was done to allow the dataset to be used in **real-time embedded deployments**, leading to a **10 MSPS** final sampling rate.

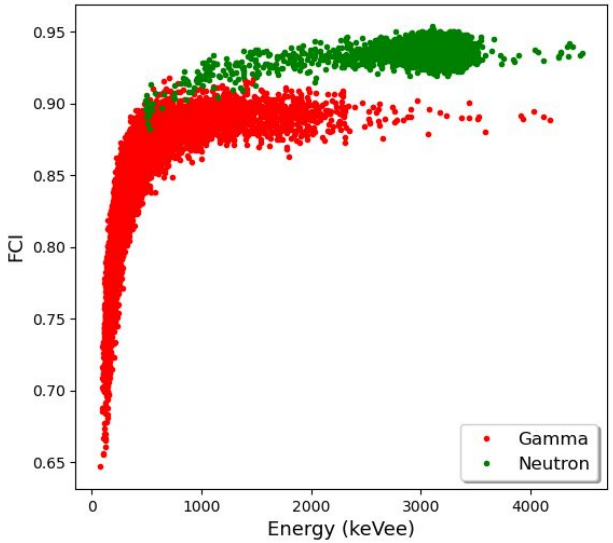
Dataset from https://doi.org/10.5281/zenodo.8037059

ML and model compression techniques for SoC/FPGA

Applications

Gamma/neutron discrimination

Subsampling was done to allow the dataset to be used in **real-time embedded deployments**, leading to a **10 MSPS** final sampling rate.



Dataset from https://doi.org/10.5281/zenodo.8037059

ML and model compression techniques for SoC/FPGA

Applications

Gamma/neutron discrimination

Teacher based on MLP

- 2,267 parameters
- Composed of
  - 4 FC layers
- Accuracy:
  - Gamma: 99.44%
  - Neutron: 98.22%

ML and model compression techniques for SoC/FPGA

Applications

Gamma/neutron discrimination

Teacher based on MLP

- 2,267 parameters
- Composed of
  - 4 FC layers
- Accuracy:
  - Gamma: 99.44%
  - Neutron: 98.22%

Student based on MLP

- **950** parameters
- Composed of
  - **3 FC** layers
- **9-bit fixed point**
- Sparsity: **30%**
- Accuracy:
  - Gamma: 98%
  - Neutron: 97.22%

ML and model compression techniques for SoC/FPGA

Applications

Gamma/neutron discrimination

- **SoC memory footprint in terms of resource utilization @100MHz**
  - Zedboard platform: **LUT below 41%, DSP 35%, BRAM 0%, FF 21%**

ML and model compression techniques for SoC/FPGA

Applications

Gamma/neutron discrimination

- **SoC memory footprint in terms of resource utilization @100MHz**
  - Zedboard platform: **LUT below 41%, DSP 35%, BRAM 0%, FF 21%**

- **SoC latency**
  - Zedboard platform: **45 clk cycles (0.225 us)**

Image classification based on CNN

ML and model compression techniques for SoC/FPGA

Applications

Pest classification in fruit crops

Precision agriculture on the edge

ML and model compression techniques for SoC/FPGA

Applications

Pest classification in fruit crops

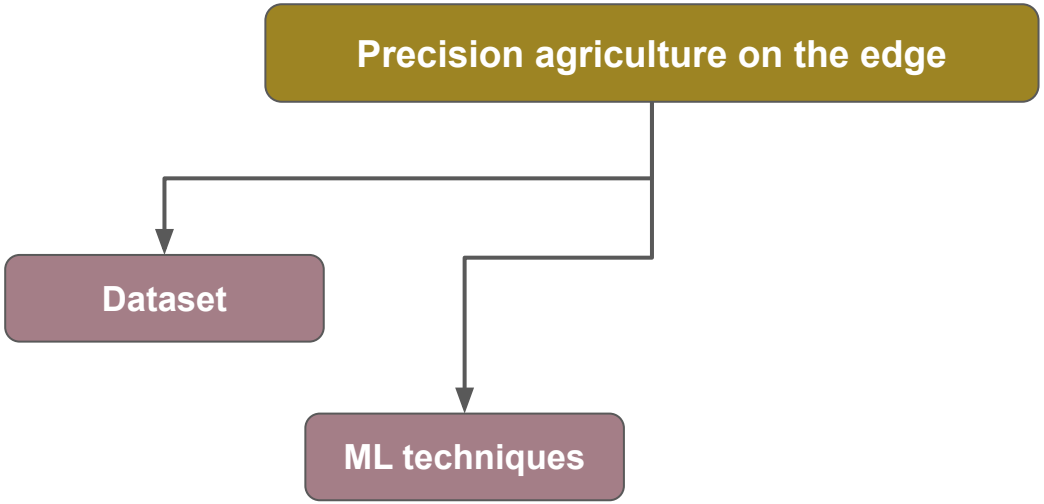
Precision agriculture on the edge

Dataset

ML and model compression techniques for SoC/FPGA

Applications

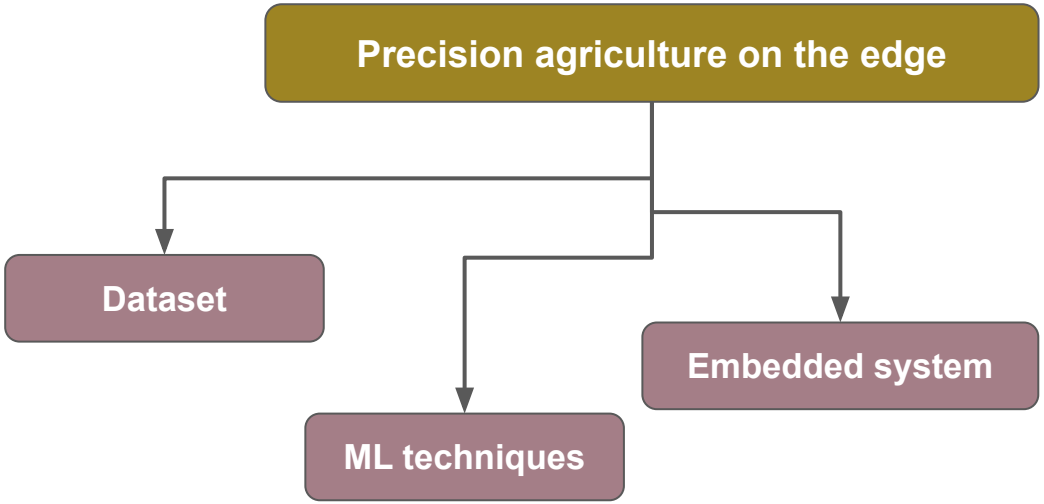
Pest classification in fruit crops



ML and model compression techniques for SoC/FPGA

Applications

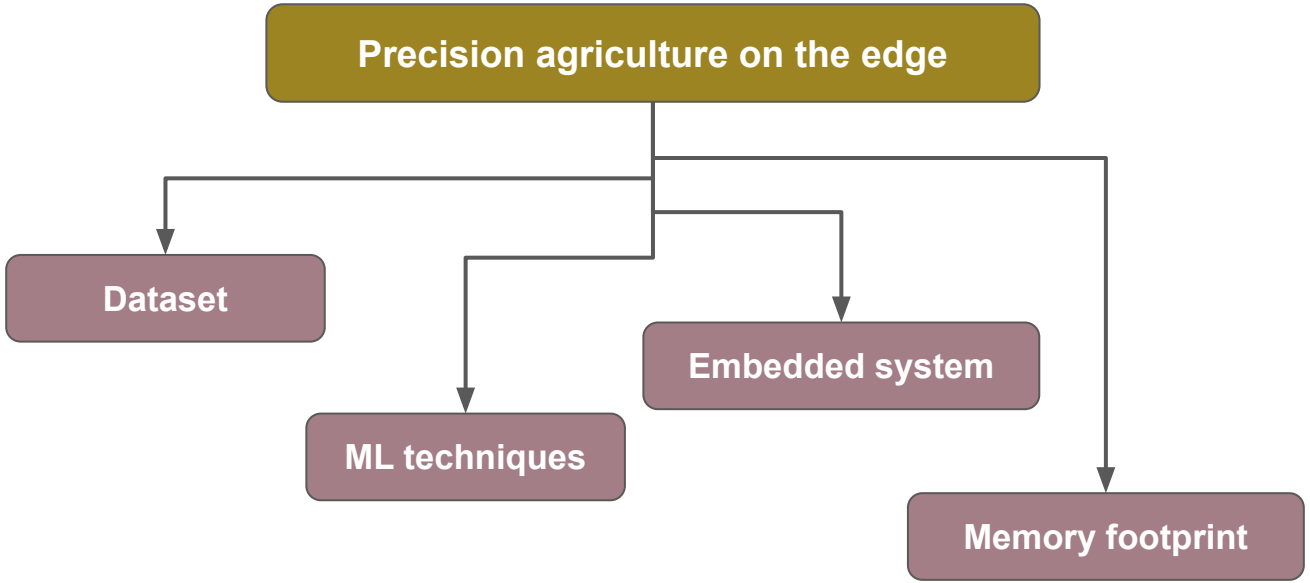
Pest classification in fruit crops



ML and model compression techniques for SoC/FPGA

Applications

Pest classification in fruit crops



ML and model compression techniques for SoC/FPGA

Applications

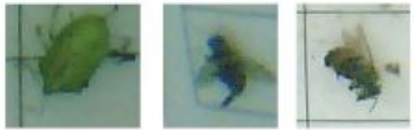
Pest classification in fruit crops

Precision agriculture on the edge

Nectras IoT trap

Captured image

Other insects



Lobesia botrana

ML and model compression techniques for SoC/FPGA

Applications

Pest classification in fruit crops

Pest24 [6]

A standard dataset available in the literature for training, granting a stable and effective performance.

ML and model compression techniques for SoC/FPGA

Applications

Pest classification in fruit crops

Pest24

A standard dataset available in the literature for training, granting a stable and effective performance.

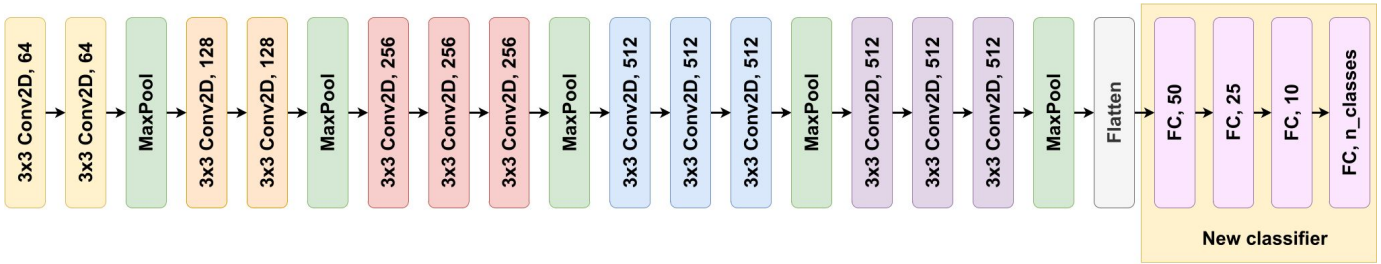
Arg

Images provided by the current system in Argentina.

ML and model compression techniques for SoC/FPGA

Applications

Pest classification in fruit crops

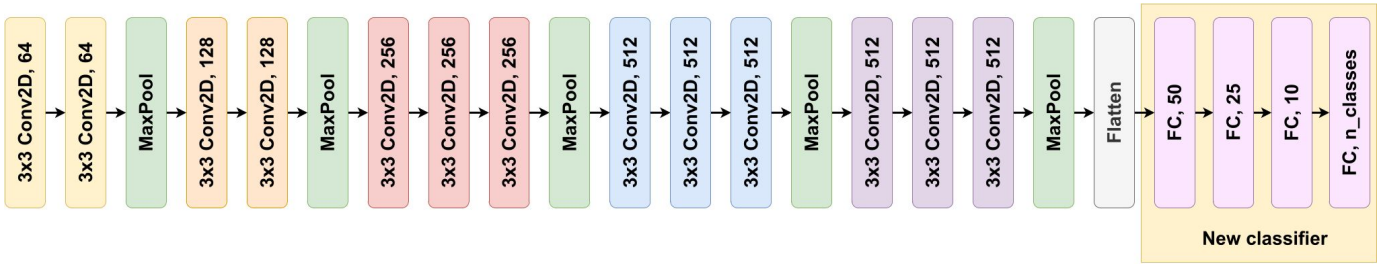


Teacher architecture based on VGG16 and obtained through transfer learning
– 14,818,706 parameters –

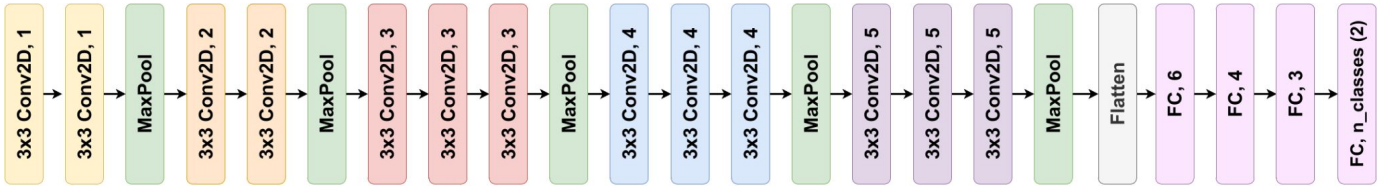
ML and model compression techniques for SoC/FPGA

Applications

Pest classification in fruit crops



Teacher architecture based on VGG16 and obtained through transfer learning
– 14,818,706 parameters –

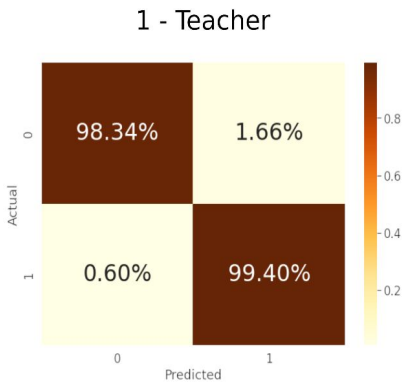


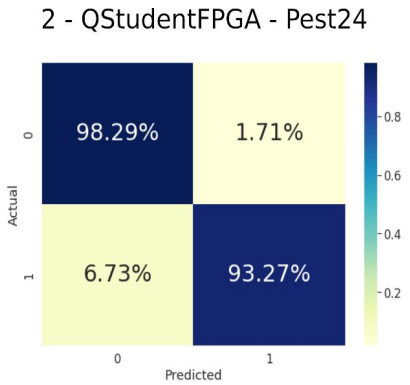
Compressed architecture.
Compression ratio: 7,409*x* – in number of parameters –

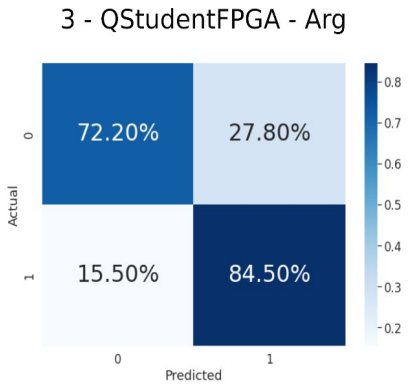
ML and model compression techniques for SoC/FPGA

Applications

Pest classification in fruit crops







ML and model compression techniques for SoC/FPGA

Applications

Pest classification in fruit crops

- **Overall accuracy**
  - Teacher architecture: **98.87%**
  - Student architecture: **95.78%**

ML and model compression techniques for SoC/FPGA

Applications

Pest classification in fruit crops

- **Overall accuracy**
  - Teacher architecture: **98.87%**
  - Student architecture: **95.78%**

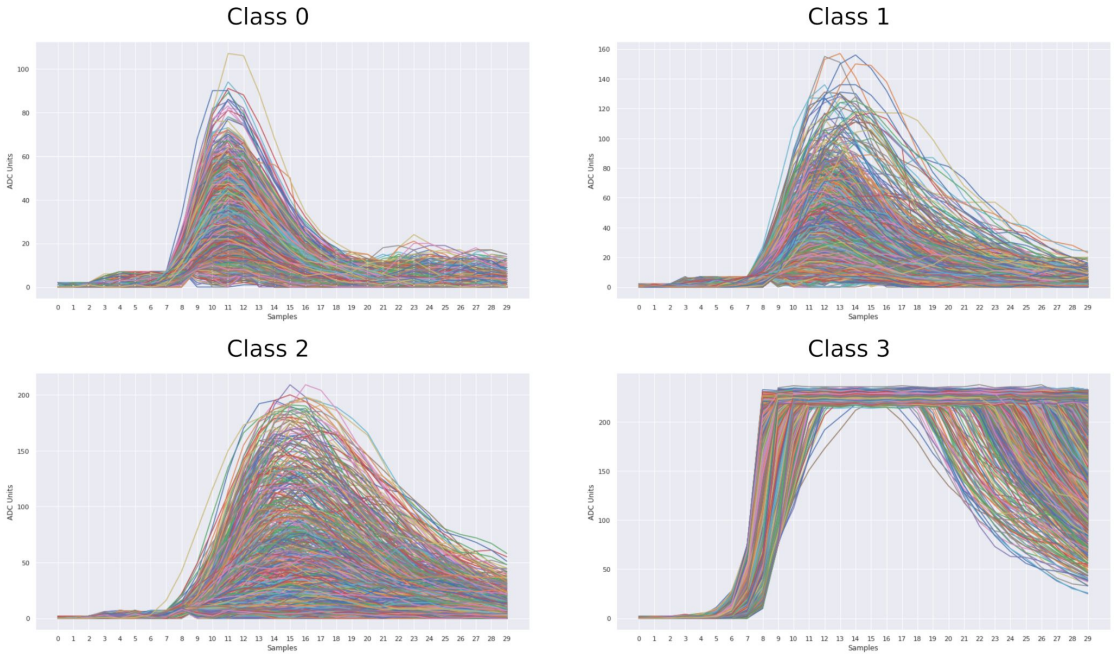
- **SoC memory footprint in terms of resource utilization @200MHz**
  - KRIA platform: **below 21%**
  - PYNQ-Z1 platform: **below 63%**

Pulse shape discriminator for cosmic rays studies

ML and model compression techniques for SoC/FPGA

Applications

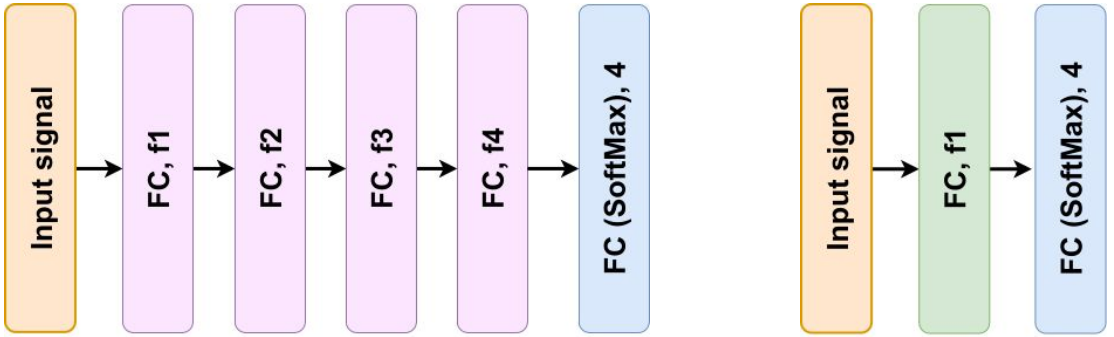
Pulse shape discriminator for cosmic rays studies



ML and model compression techniques for SoC/FPGA

Applications

Pulse shape discriminator for cosmic rays studies



Left. Teacher architecture based on MLP - **16,352** parameters.

Right: Distilled architecture - **529** parameters
Compression ratio: 30.91*x*.

ML and model compression techniques for SoC/FPGA

Applications

Pulse shape discriminator for cosmic rays studies

- **Overall accuracy**
  - Teacher architecture: **99.70%**
  - Student architecture: **98.96%**
    - **8-bit fixed point**
    - **Target sparsity: 20%**

ML and model compression techniques for SoC/FPGA

Applications

Pulse shape discriminator for cosmic rays studies

- **Overall accuracy**
  - Teacher architecture: **99.70%**
  - Student architecture: **98.96%**
    - **8-bit fixed point**
    - **Target sparsity: 20%**

- **SoC memory footprint in terms of resource utilization @200MHz**
  - Artix-7: **below 27%**

ML and model compression techniques for SoC/FPGA

Applications

Pulse shape discriminator for cosmic rays studies

- **Overall accuracy**
  - Teacher architecture: **99.70%**
  - Student architecture: **98.96%**
    - **8-bit fixed point**
    - **Target sparsity: 20%**

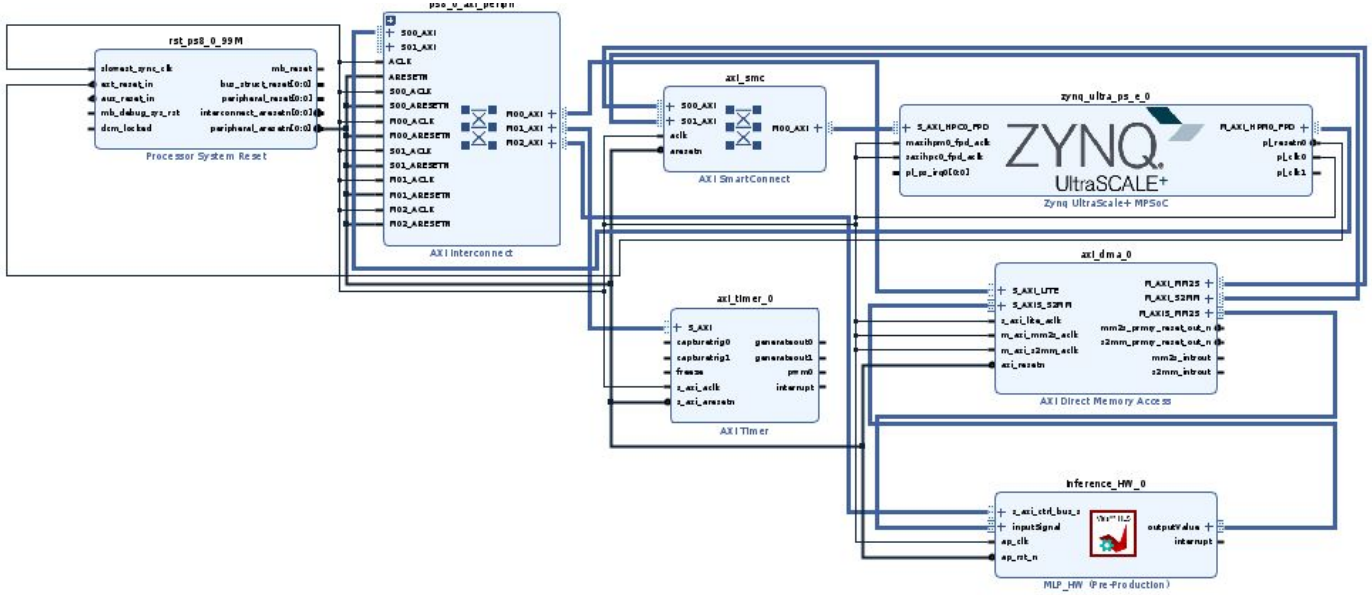
- **SoC memory footprint in terms of resource utilization @200MHz**
  - Artix-7: **below 27%**

- **SoC latency @200MHz**
  - Artix-7: **10 clock cycles**

ML and model compression techniques for SoC/FPGA

Applications - Connections example

Pulse shape discriminator for cosmic rays studies



ML and model compression techniques for SoC/FPGA

Applications

Pulse shape discriminator for cosmic rays studies

Pulse shape discriminator for cosmic rays

In [ ]:
```
from pynq import Overlay
```

In [ ]:
```
ol = Overlay("hw/inference_PYNQ.bit")
```

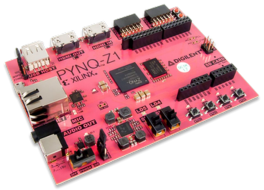
In [ ]:
```
ol.ip_dict
```

In [ ]:
```
dma = ol.axi_dma_0
dma_send = ol.axi_dma_0.sendchannel
dma_recv = ol.axi_dma_0.recvchannel
```

In [ ]:
```
from pynq import allocate
import numpy as np

data_size = 30
input_buffer = allocate(shape=(data_size,), dtype=np.uint32)
```

In [ ]:
```
x3 = [0, 2, 0, 0, 0, 0, 2, 14, 68, 231, 232, 232, 232, 230, 232,
      231, 233, 232, 231, 231, 232, 232, 231, 230, 232, 231, 232, 231, 231, 230]
for i in range(0, data_size):
    input_buffer[i] = x3[i]
```

In [ ]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(15,7))
plt.xlabel('Samples', fontsize=11)
plt.ylabel('Amplitude', fontsize=11)
plt.grid(True, alpha=1.0)
plt.plot(x3, 'o', label="Signal 1", color='navy', markersize=7, lw=1)
```



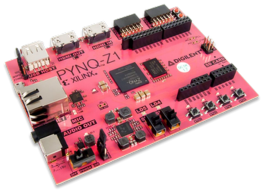
PYNQ™

ML and model compression techniques for SoC/FPGA

Applications

Pulse shape discriminator for cosmic rays studies

```
In [ ]: hls_ip = ol.inference_HW_0

In [ ]: hls_ip.register_map

In [ ]: # Initialize HLS IP core

        CONTROL_REGISTER = 0x0
        hls_ip.write(CONTROL_REGISTER, 0x81) # 0x81 will set bit 0

In [ ]: hls_ip.register_map

In [ ]: # Start the DMA transfer
        dma_send.transfer(input_buffer)

In [ ]: output_buffer = allocate(shape=(4,), dtype=np.uint32)

In [ ]: dma_recv.transfer(output_buffer)

In [ ]: for i in range(4):
            print((output_buffer[i]))
```



PYNQ™

Final remarks

- The experiments show that the proposed workflow successfully generates compressed models, leading to a **fully on-chip memory-mapped implementation** on the FPGA.
- The **integration of KD** into the ensemble of compression techniques contributes to achieving a balanced student model in terms of size, computational efficiency, and accuracy.
- The workflow addresses the entire development cycle: **from the ML-based architecture training to the hardware deployment**, overcoming the limitations outlined in previous works.
- Furthermore, the **ComBlock** in the hardware assessment framework facilitates the model performance evaluation in the FPGA by simplifying the communication interfaces.

References

[1] Choudhary, T., Mishra, V., Goswami, A., Sarangapani, J.: A comprehensive survey on model compression and acceleration. In: Artificial Intelligence Review, 53(7), pp. 5113--5155, 2020, doi: https://doi.org/10.1007/s10462-020-09816-7.

[2] Hinton, G.; Vinyals, O.; Dean, J.: Distilling the knowledge in a neural network. In: arXiv preprint arXiv:1503.02531, 2(7), 2015, doi: https://doi.org/10.48550/arXiv.1503.02531.

[3] Blalock, D.; Gonzalez Ortiz, J. J.; Frankle, J.; Guttag, J. (2020).: What is the state of neural network pruning?. In: Proceedings of machine learning and systems, 2, pp. 129--146, 2020.

[4] Liang, T.; Glossner, J.; Wang, L.; Shi, S.; Zhang, X.: Pruning and quantization for deep neural network acceleration: A survey. In: Neurocomputing, 461, pp. 370--403, 2021, doi: https://doi.org/10.1016/j.neucom.2021.07.045.

[5] Duarte, J.; Han, S.; Harris, P.; Jindariani, S.; Kreinar, E.; Kreis, B.; Ngadiuba, J.; Pierini, M.; Rivera, R.; Tran, N.; Wu, Z. : Fast inference of deep neural networks in FPGAs for particle physics. In: Journal of Instrumentation, 13(07), 2018, https://doi.org/10.1088/1748-0221/13/07/P07027.

[6] Wang, Qi-Jin; Zhang, Sheng-Yu; Dong, Shi-Feng; Zhang, Guang-Cai; Yang, J.; Li; R., Wang, Hong-Qiang: Pest24: A large-scale very small object dataset of agricultural pests for multi-target detection. In: Computers and Electronics in Agriculture, Volume 175, 2020, doi: https://doi.org/10.1016/j.compag.2020.105585.

[7] Molina, R. S.; Gil-Costa, V.; Crespo, M. L.; Ramponi, G.: High-Level Synthesis Hardware Design for FPGA-based Accelerators: Models, Methodologies, and Frameworks. In: IEEE Access, 2022, doi:10.1109/ACCESS.2022.3201107.

[8] García Ordóñez, L.G.; Molina, R.S.; Morales Argueta, I.R. Morales; Crespo, M.L.; Cicuttin, A. ; S. Carrato, G. Ramponi, H.E. Pérez Figueroa, and M.G. Ballina Escobar: Pulse Shape Discrimination for Online Data Acquisition in Water Cherenkov Detectors Based on FPGA/SoC. In: 37th International Cosmic Ray Conference (ICRC2021), 2021, doi: https://doi.org/10.22323/1.395.0274

[9] Molina, R. S., Carrer, V., Ballina, M., Crespo, M. L., Bollati, L., Sequeiro, D., ... & Ramponi, G. (2022, September). MI-based classifier for precision agriculture on embedded systems. In International Conference on Applications in Electronics Pervading Industry, Environment and Society (pp. 117-124). Cham: Springer Nature Switzerland.

[10] Wang, Q. J., Zhang, S. Y., Dong, S. F., Zhang, G. C., Yang, J., Li, R., & Wang, H. Q. (2020). Pest24: A large-scale very small object data set of agricultural pests for multi-target detection. Computers and Electronics in Agriculture, 175, 105585.

[11] Morales, I. R., Crespo, M. L., Bogovac, M., Cicuttin, A., Kanaki, K., & Carrato, S. (2023). Gamma/neutron classification with SiPM CLYC detectors using frequency-domain analysis for embedded real-time applications. Nuclear Engineering and Technology.

The Abdus Salam
International Centre
for Theoretical Physics

UNIVERSITÀ
DEGLI STUDI
DI TRIESTE

IAEA

Joint ICTP-IAEA School on
Systems-on-Chip based on
FPGA for Scientific Instrumentation
and Reconfigurable Computing

Thank you!!

Trieste - Italy
2023