The Abdus Salam International Centre for Theoretical Physics

ICTP SAIFR

IFT - UNESP INSTITUTO DE FÍSICA TEÓRICA

Núcleo de Computação Científica

SPRACE

FPGA for the Acceleration of Machine Learning Algorithms

Romina Molina

Multidisciplinary Laboratory

Joint ICTP, SAIFR and UNESP School on Systems-on-Chip,Embedded Microcontrollers and their Applications in Research and Industry- smr3557 -

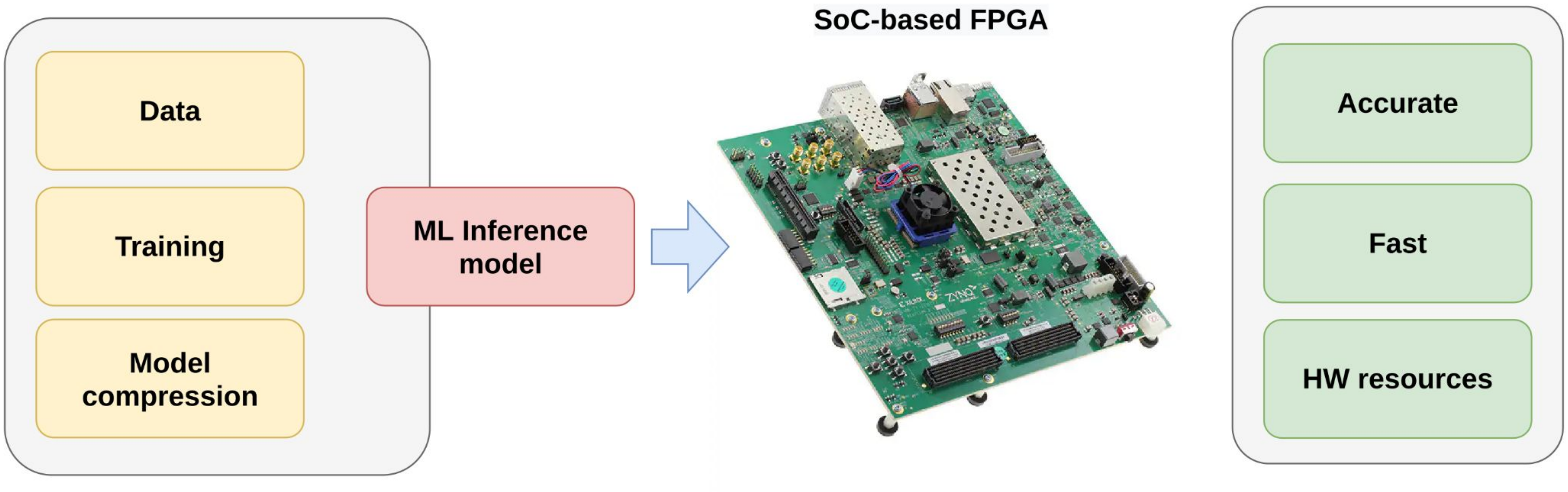
Outline

Outline

Introduction

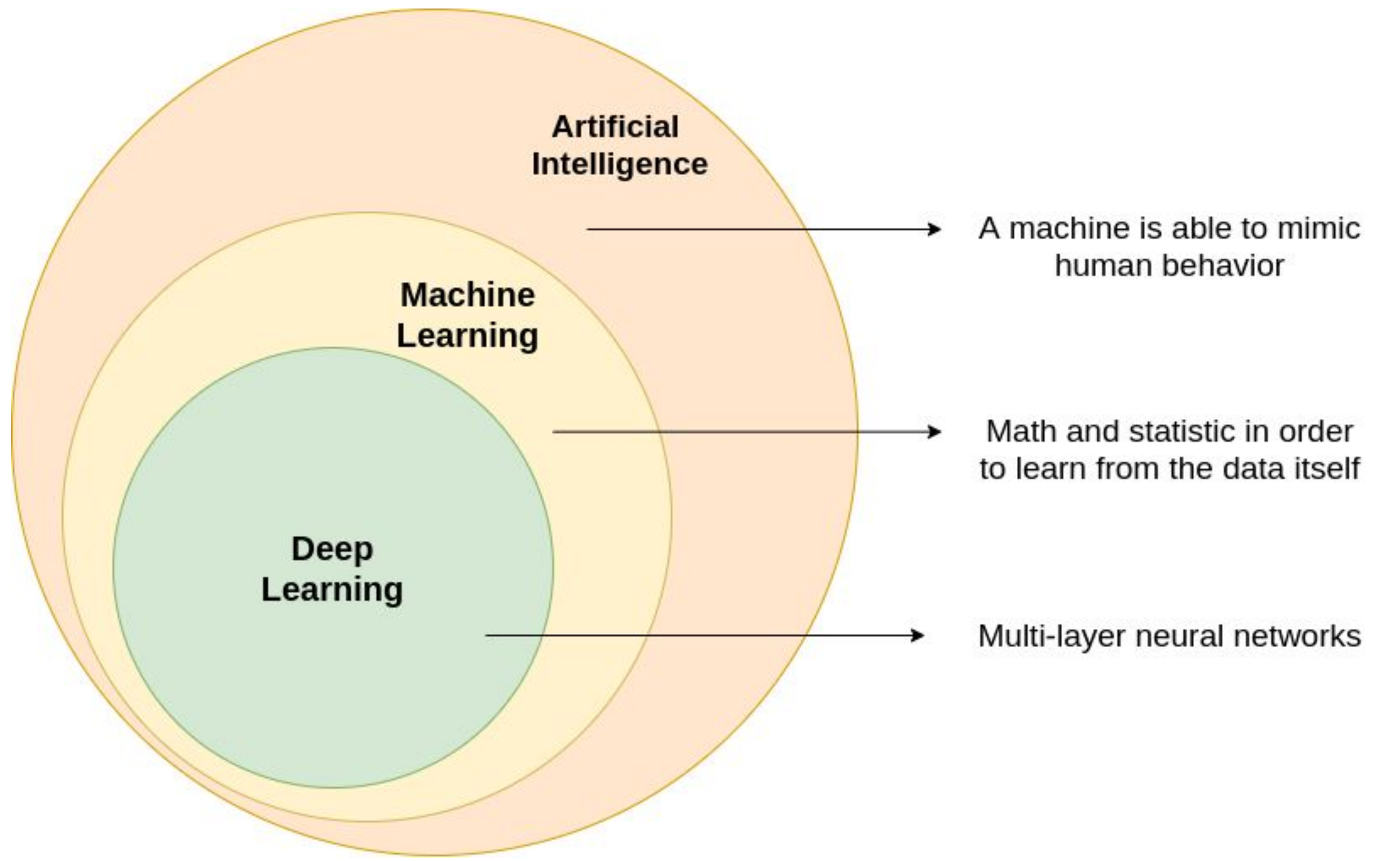
Introduction

Machine Learning and System On Chip



Machine Learning

Machine Learning


Artificial Intelligence
Machine Learning
Deep Learning
A machine is able to mimic human behavior
Math and statistic in order to learn from the data itself
Multi-layer neural networks

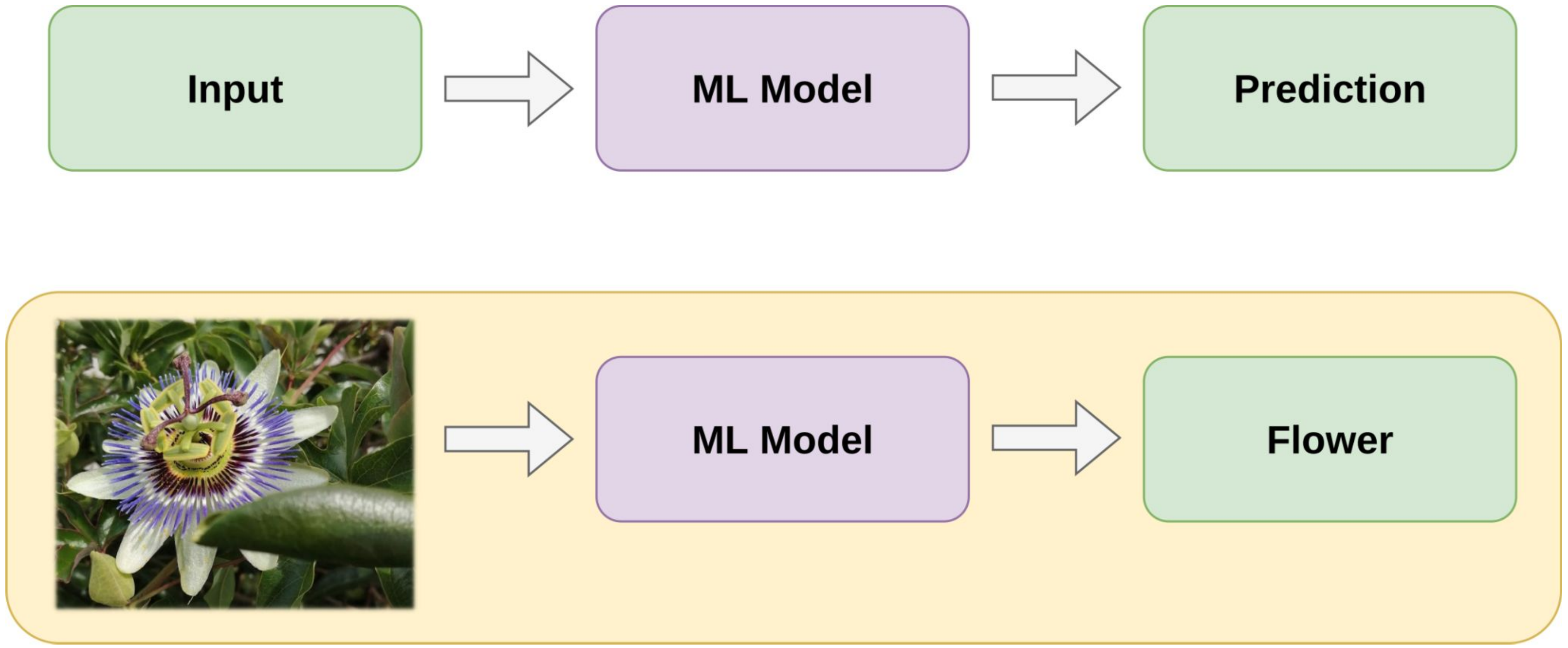
Machine Learning

"Learning can be defined as the process of estimating associations between inputs, outputs, and parameters of a system using a limited number of observations"

(Cherkassky et al. 2007)

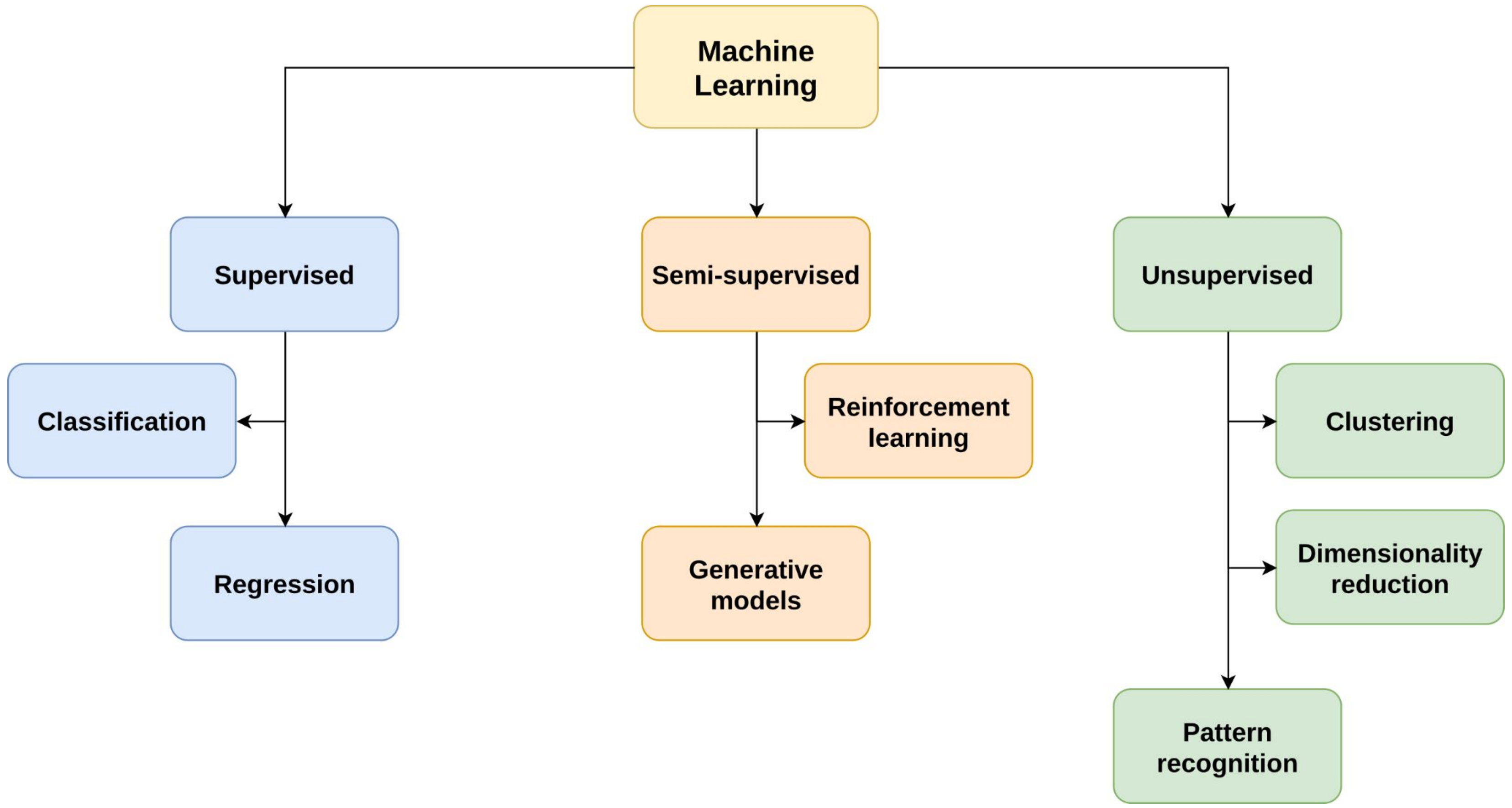
Machine Learning

Classification ML-based



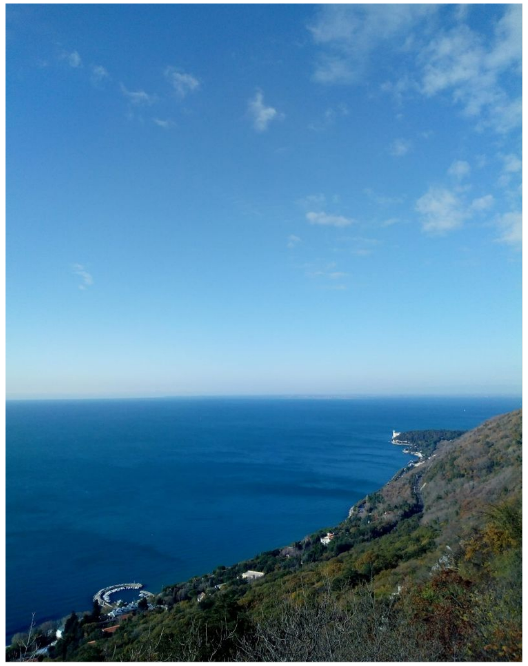
Machine Learning

Basic Classification



Machine Learning

Supervised learning

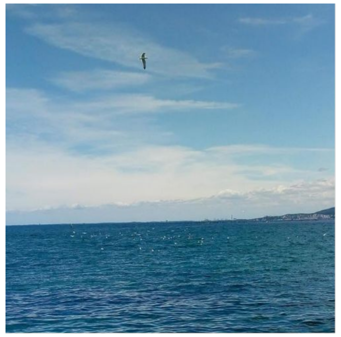


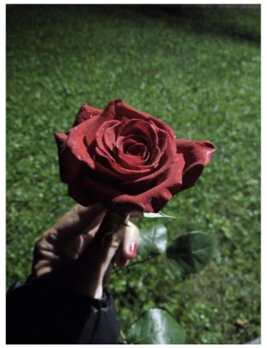
Sea

Flower

Machine Learning

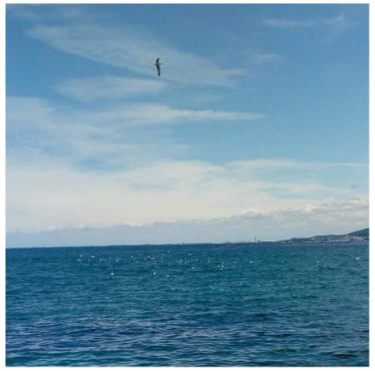
Unsupervised learning

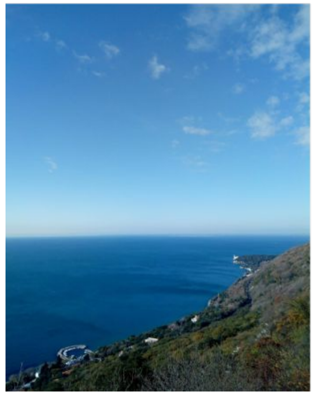




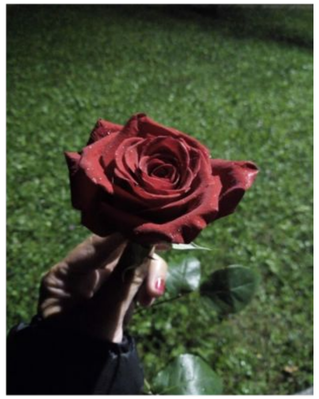
Machine Learning

Unsupervised learning





Sea

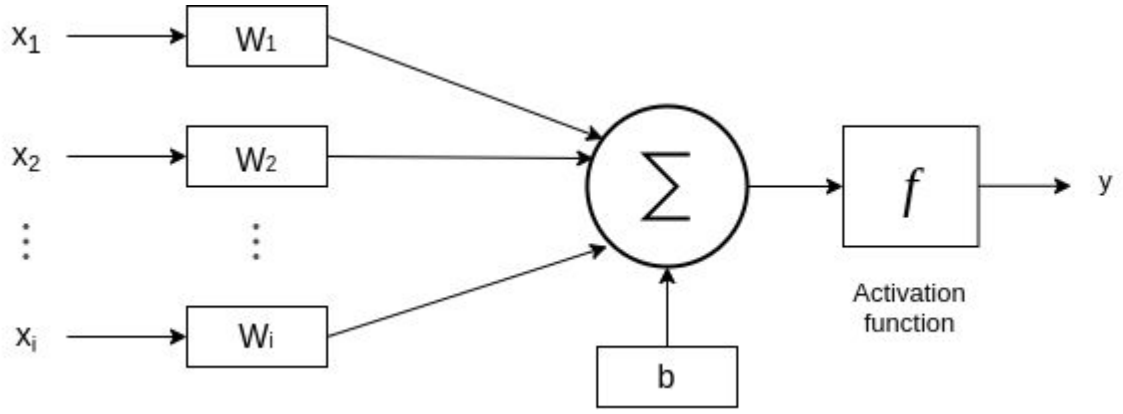


Flower

Machine Learning

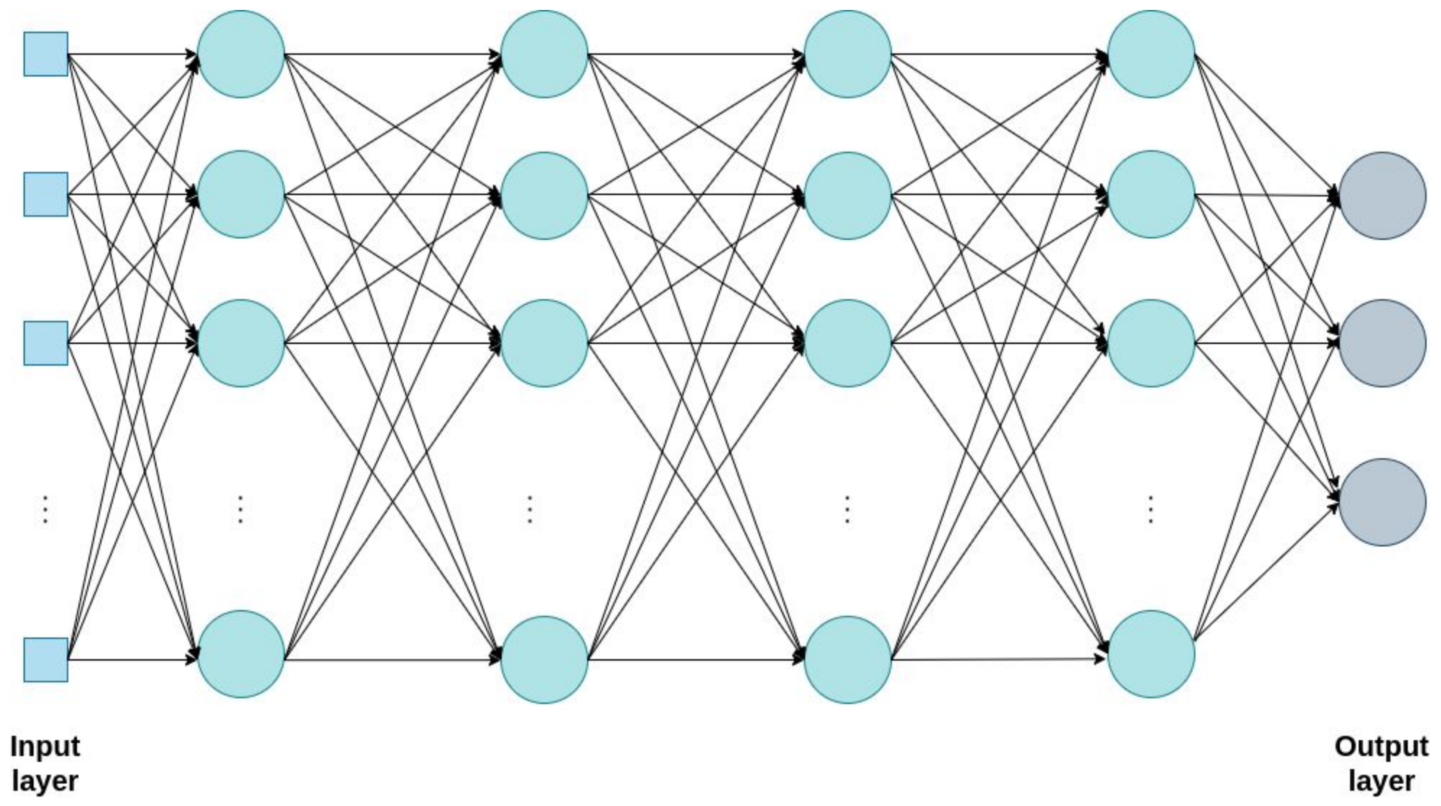
Artificial Neural Network

An Artificial Neural Network (ANN) is composed of neuron (or node) interconnections arranged in different layers.



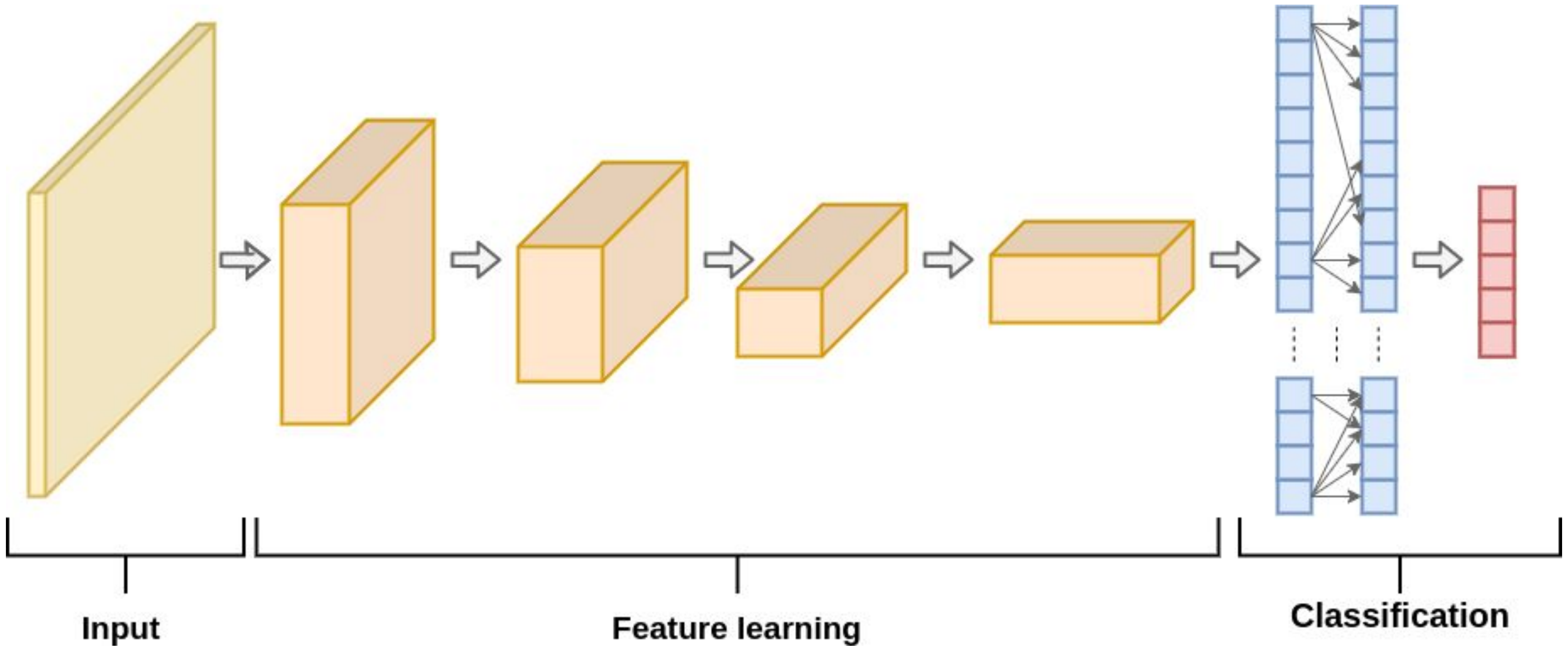
Machine Learning

Multi-Layer Perceptron (MLP) architecture

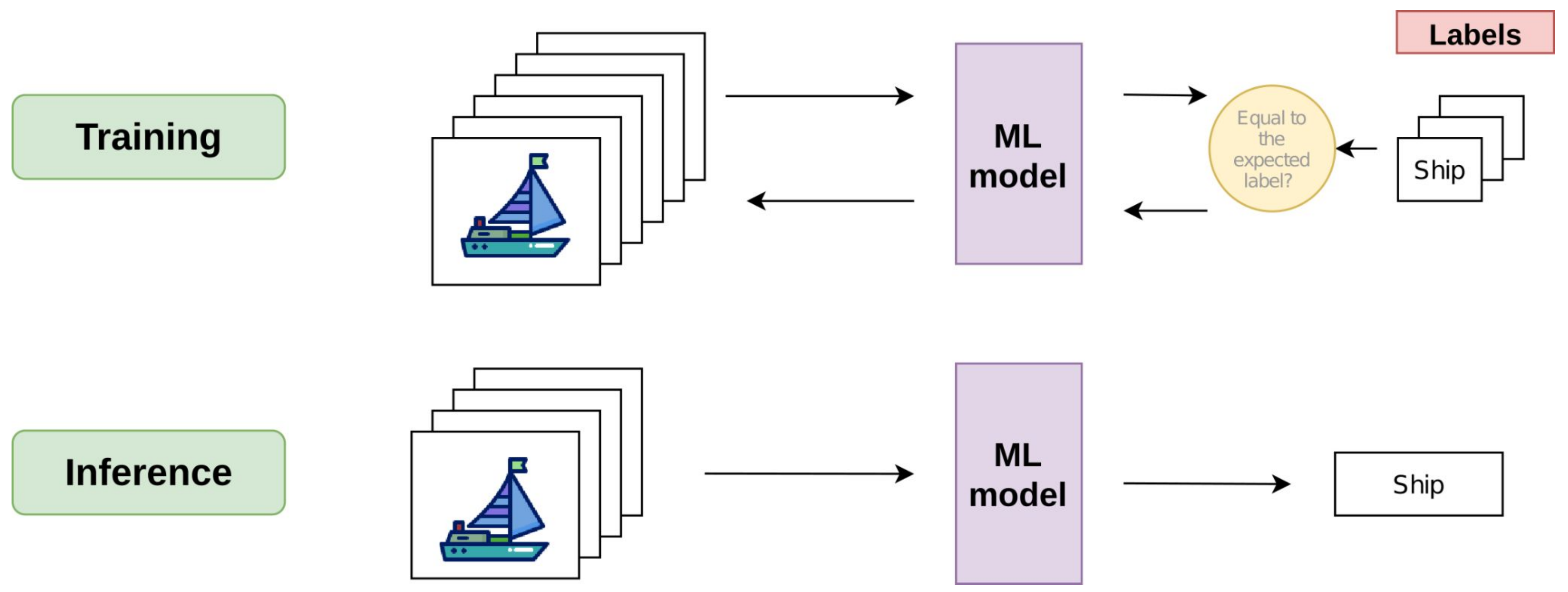


Machine Learning

Convolutional Neural Networks (CNN) architecture



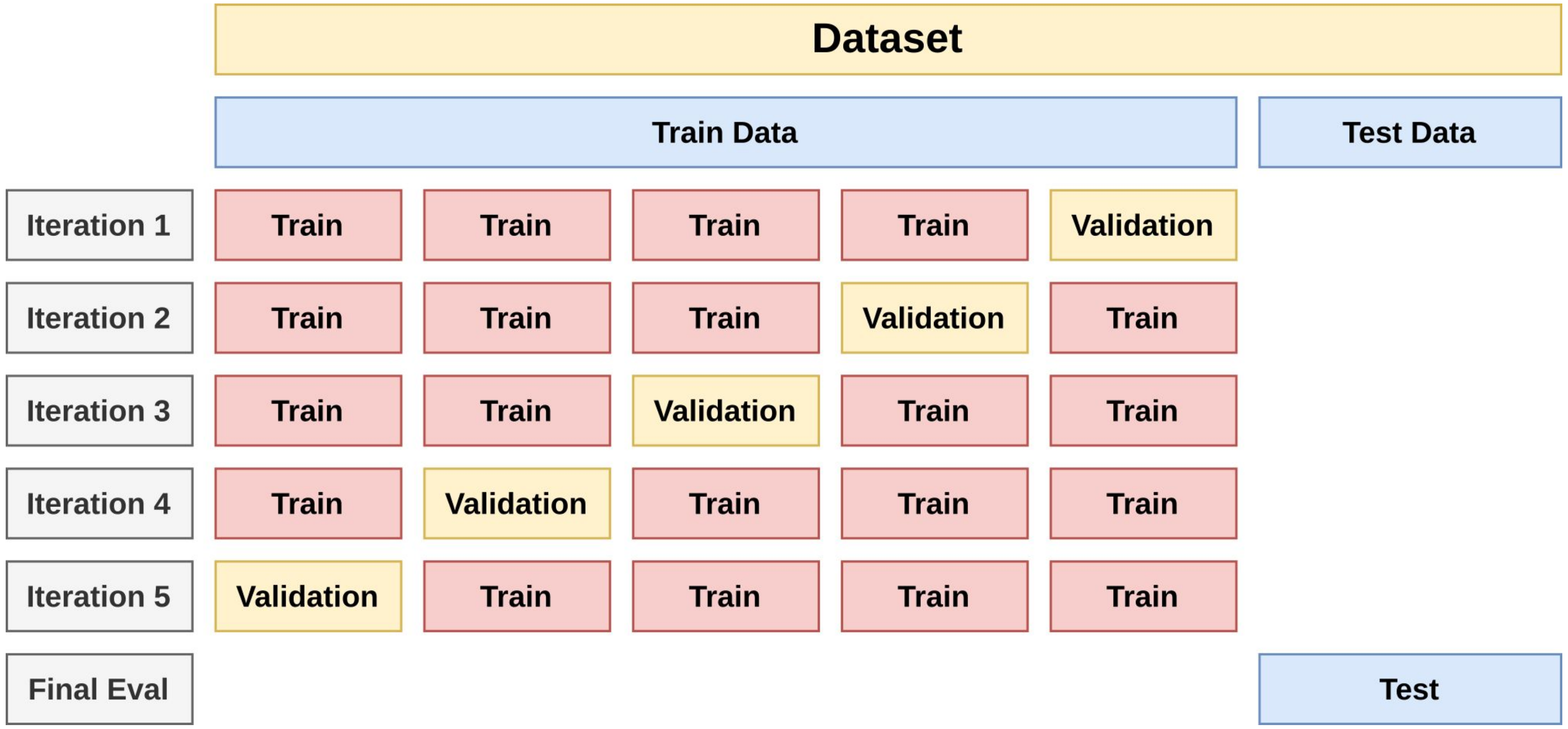
Machine Learning



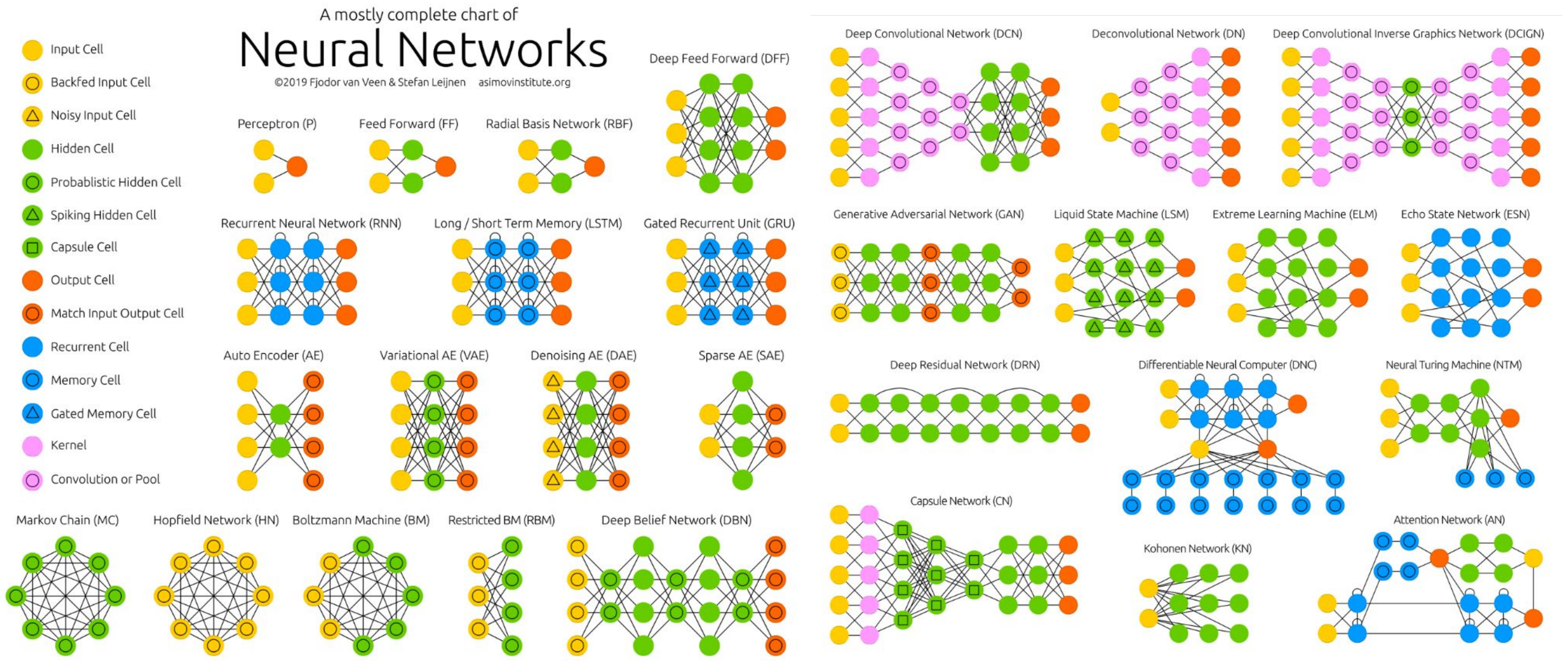
- In a classifier, an input is mapped into a specific class.
- Supervised training step to recognize patterns: the network compares its actual output with the desired output. The difference between these two values is adjusted with backpropagation.

Machine Learning

K-Fold Cross-Validation



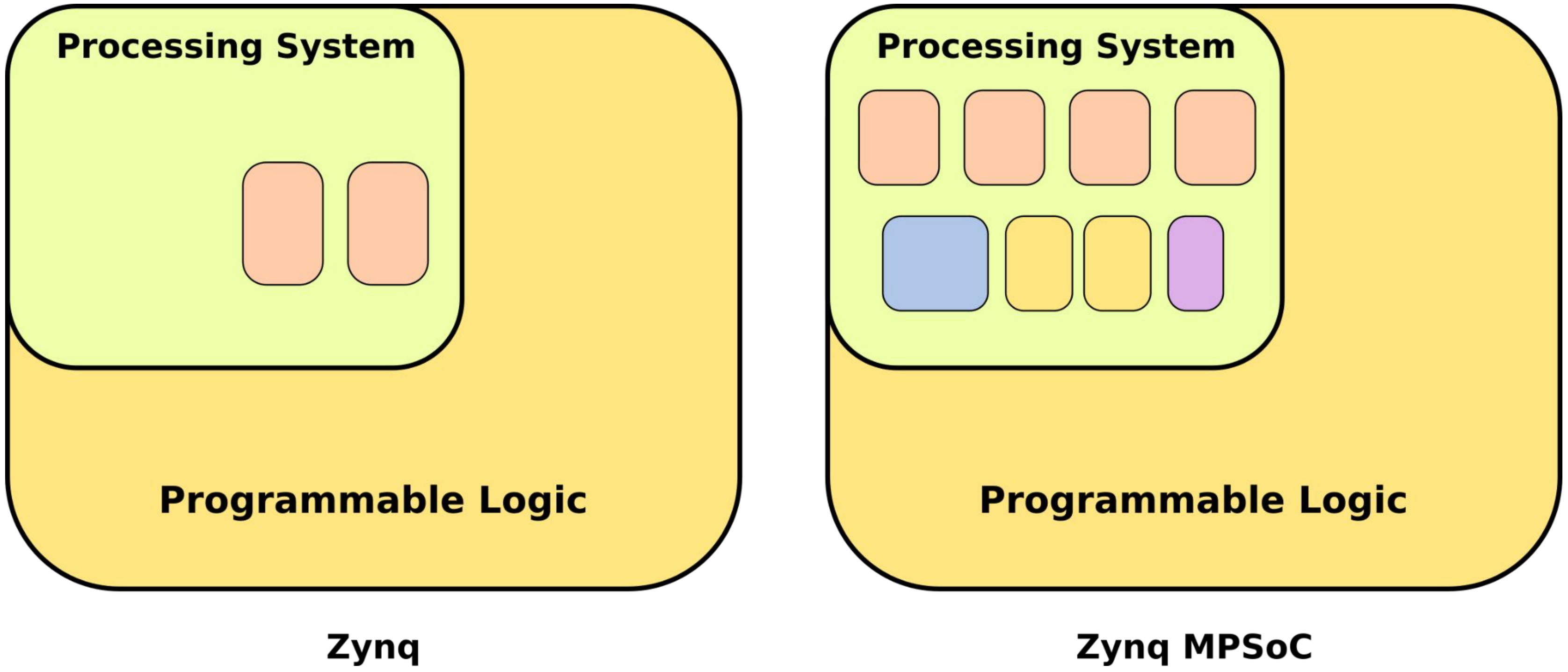
Iteration 1	Train	Train	Train	Train	Validation	
Iteration 2	Train	Train	Train	Validation	Train	
Iteration 3	Train	Train	Validation	Train	Train	
Iteration 4	Train	Validation	Train	Train	Train	
Iteration 5	Validation	Train	Train	Train	Train	
Final Eval						Test


A mostly complete chart of
Neural Networks
©2019 Fjodor van Veen & Stefan Leijnen asimovinstitute.org
Input Cell
Backfed Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Capsule Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Gated Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Differentiable Neural Computer (DNC)
Neural Turing Machine (NTM)
Capsule Network (CN)
Kohonen Network (KN)
Attention Network (AN)

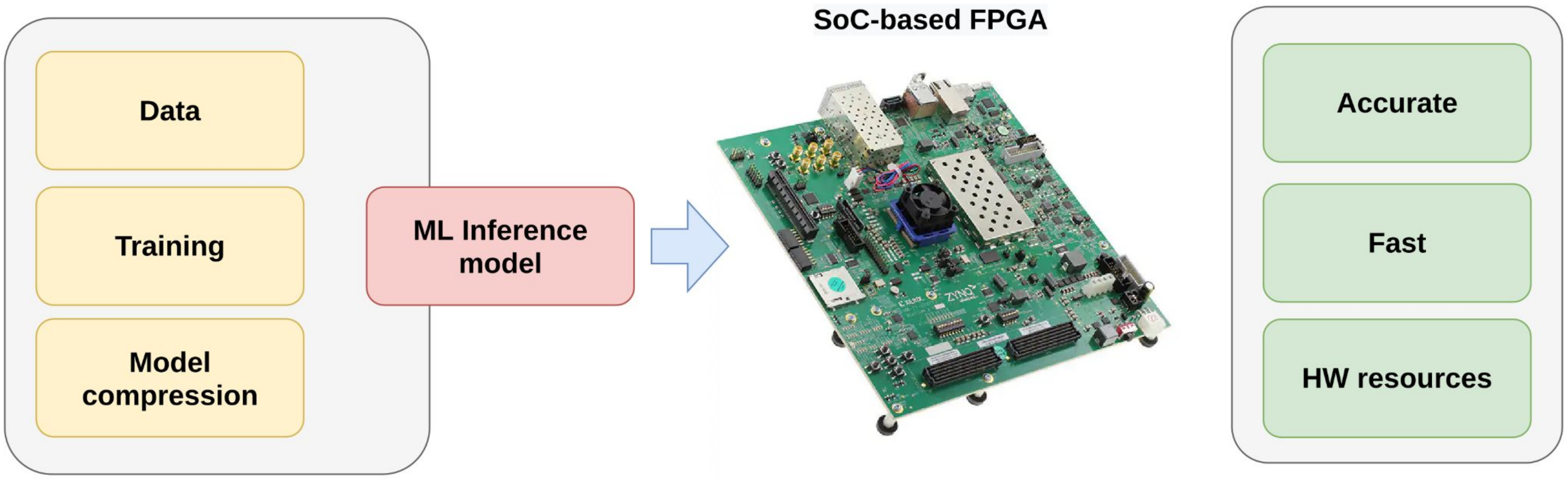
SoC-based FPGA

SoC-based FPGA

High level comparison of Zynq-7000 SoC and Zynq UltraScale+ MPSoC

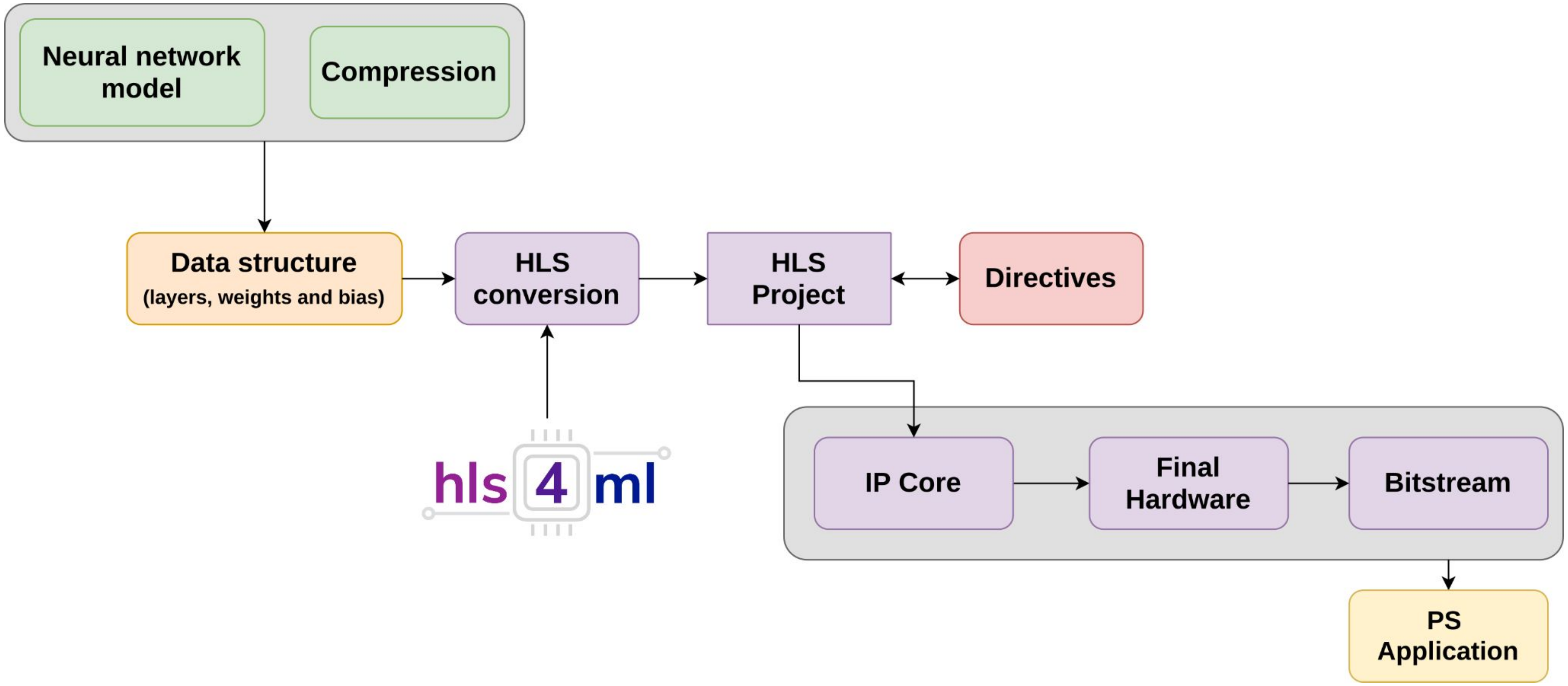


SoC-based FPGA



Acceleration of ML Inference

Acceleration of ML Inference



Acceleration of ML Inference

Considerations to map inference into FPGAs

- Clock frequency
- Bandwidth off-chip memory
- DSP, LUT, BRAM, FF
- Fixed point
- Power consumption

Acceleration of ML Inference

Considerations to map inference into FPGAs

- Low-precision arithmetic to reduce power consumption and increase throughput.
- Reduce memory footprint
  - NN model can be deployed into on-chip memory, avoiding DDR access and bottlenecks.
- Model compression techniques [1]

Acceleration of ML Inference

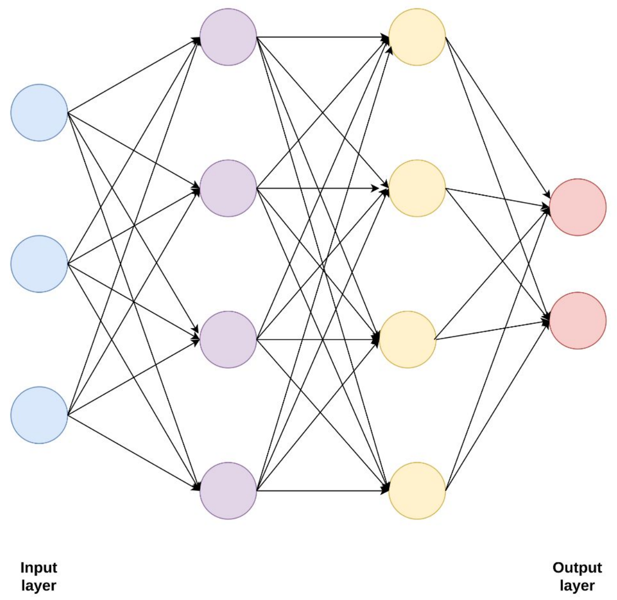
Considerations to map inference into FPGAs

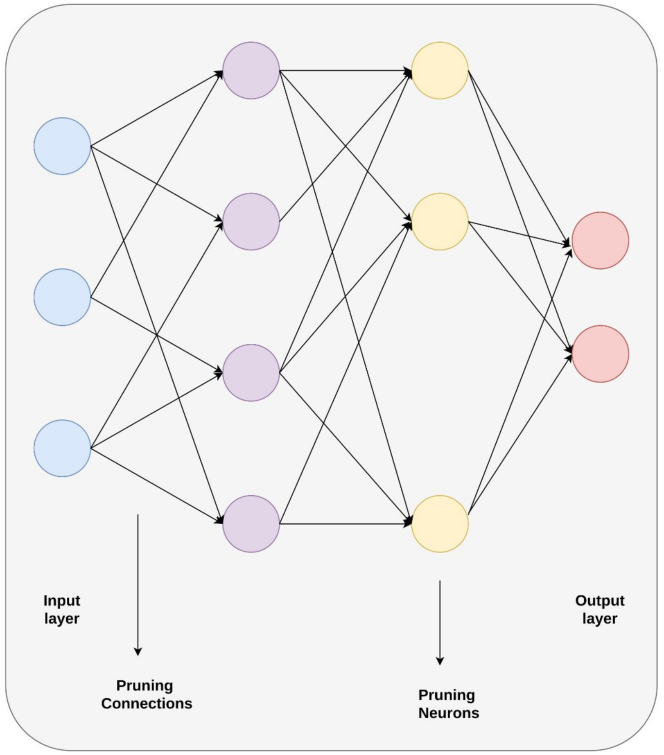
- **Model Compression**
    - Quantization (Q) and pruning (P) (train from scratch and pre-trained model)
        - Q: Reduce number of bits to represent weights and bias
        - P: Remove connections and/or neurons
    - Low-rank factorization (train from scratch and pre-trained model)
    - Compact convolutional filters (train from scratch)
    - Knowledge distillation (train from scratch)

Acceleration of ML Inference

Considerations to map inference into FPGAs

- Model Compression: Pruning

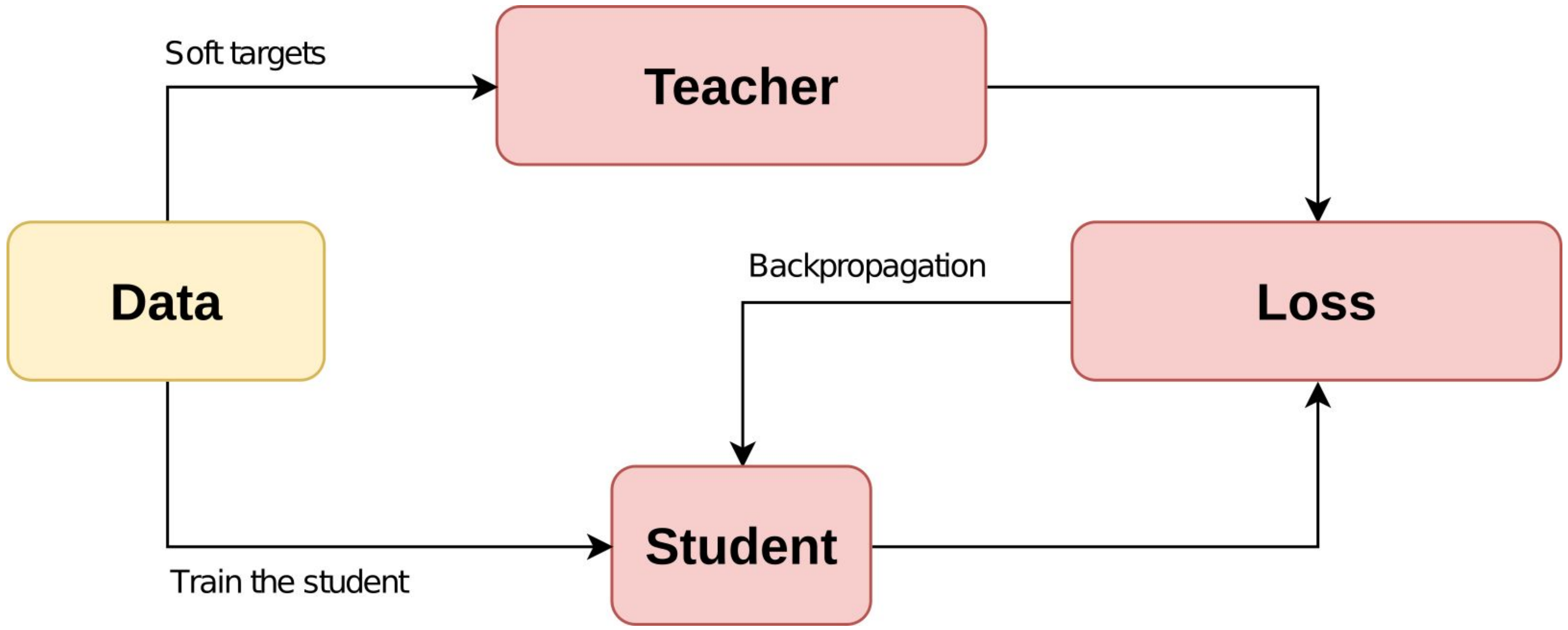




Acceleration of ML Inference

Considerations to map inference into FPGAs

- Model Compression: Knowledge Distillation



High-Level Synthesis (Vivado HLS / Vitis HLS)

High-Level Synthesis

Hardware design

- **Vivado HLS / Vitis HLS:**
  - It provides the facility to create RTL from a high level of abstraction.
  - It allows the optimization of the input code using directives to:
    - Reduce latency
    - Improve performance and throughput
    - Reduce resource utilization

Without directives, Vivado HLS /Vitis HLS will look minimize latency and improve concurrency

High-Level Synthesis

Hardware design - Directives

- **Minimize latency:** UNROLL, LOOP_FLATTEN, LOOP_MERGE.
- **Minimize throughput:** DATAFLOW, PIPELINE.
- **Improve bottleneck:** RESOURCE, ARRAY_PARTITION, ARRAY_RESHAPE.

```
#pragma HLS UNROLL
#pragma HLS PIPELINE
#pragma HLS ARRAY_PARTITION variable=layer3_out complete dim=0
```

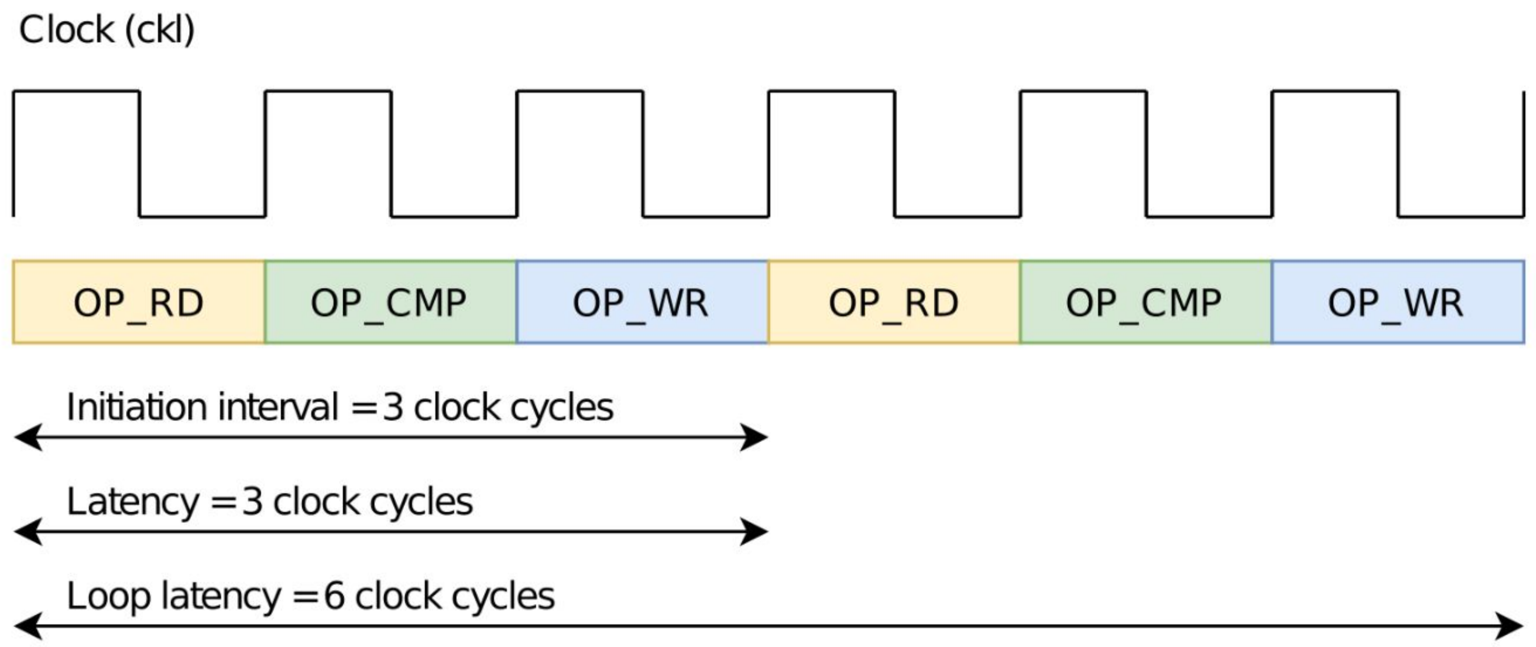
High-Level Synthesis

Hardware design - Directives

Loop Example

```
Loop_1: for(i=1; i<3; i++){
      OP_RD;
      OP_CMP;
      OP_WR;
}
```

OP_RD	Read
OP_CMP	Computation
OP_WR	Write



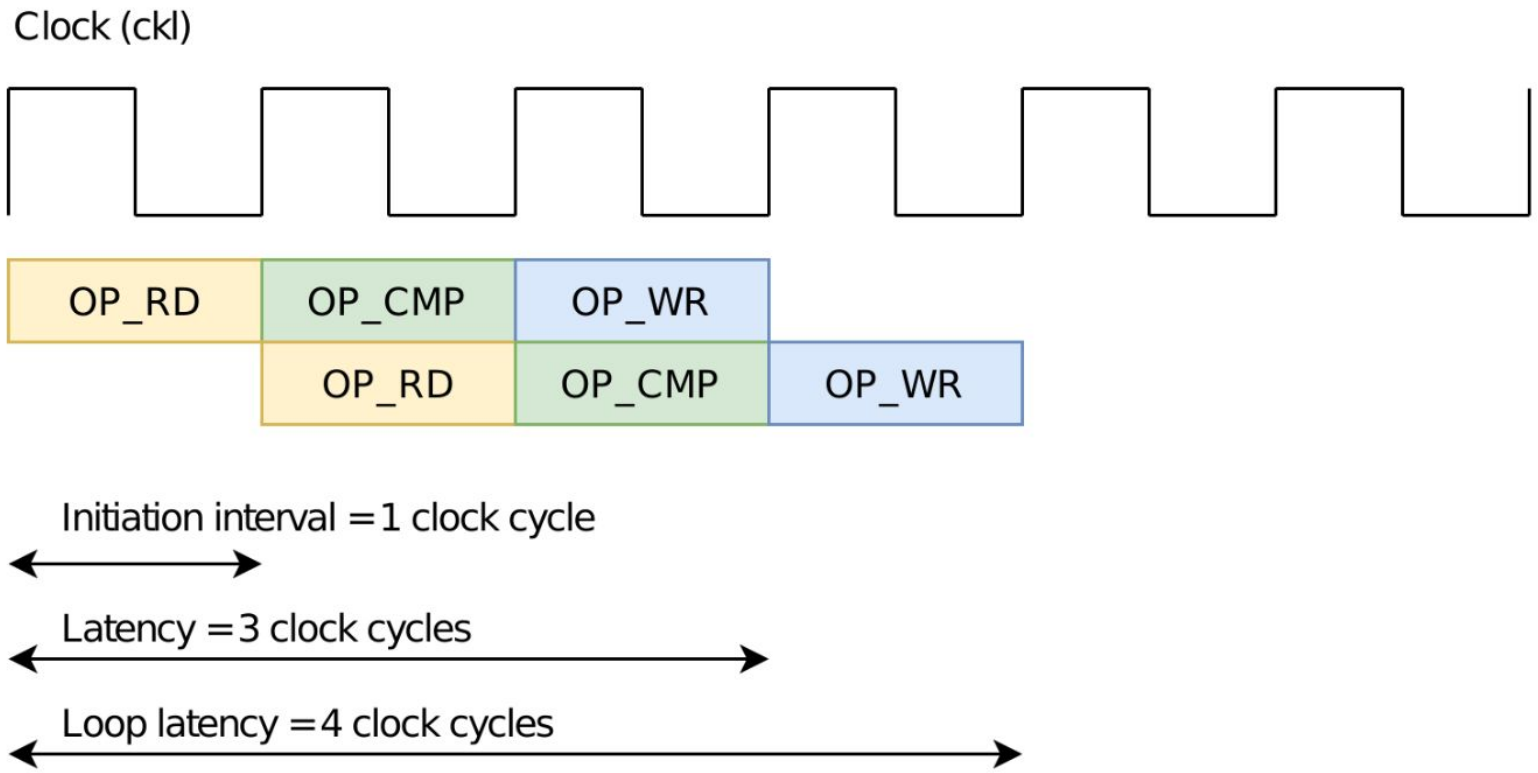
High-Level Synthesis

Hardware design - Directives

Loop + Pipeline

```
Loop_1: for(i=1; i<3; i++){
      OP_RD;
      OP_CMP;
      OP_WR;
}
```

🟨 OP_RD	Read
🟩 OP_CMP	Computation
🟦 OP_WR	Write



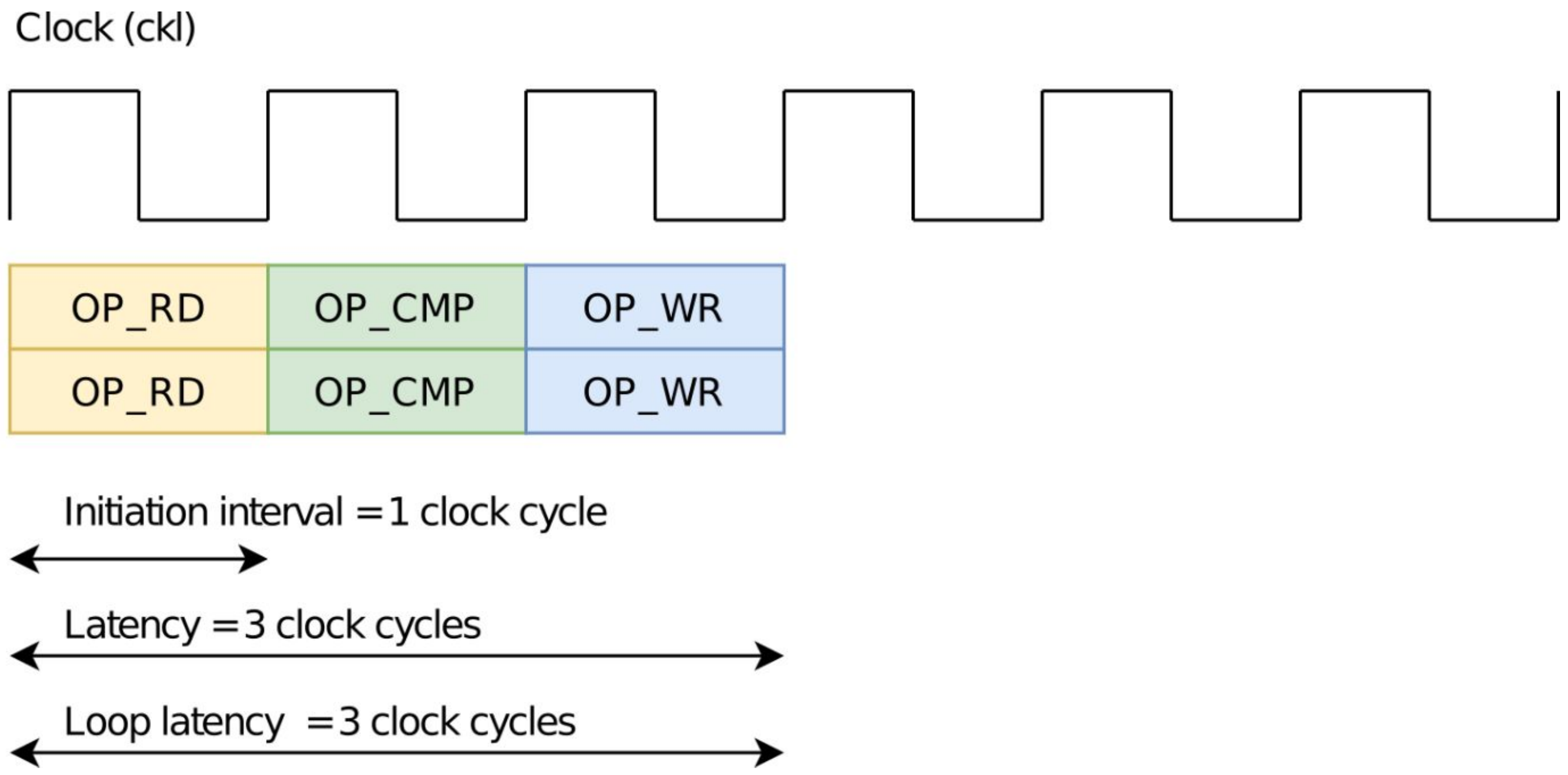
High-Level Synthesis

Hardware design - Directives

Loop + Unroll

```
Loop_1: for(i=1; i<3; i++){
      OP_RD;
      OP_CMP;
      OP_WR;
}
```

OP_RD	Read
OP_CMP	Computation
OP_WR	Write



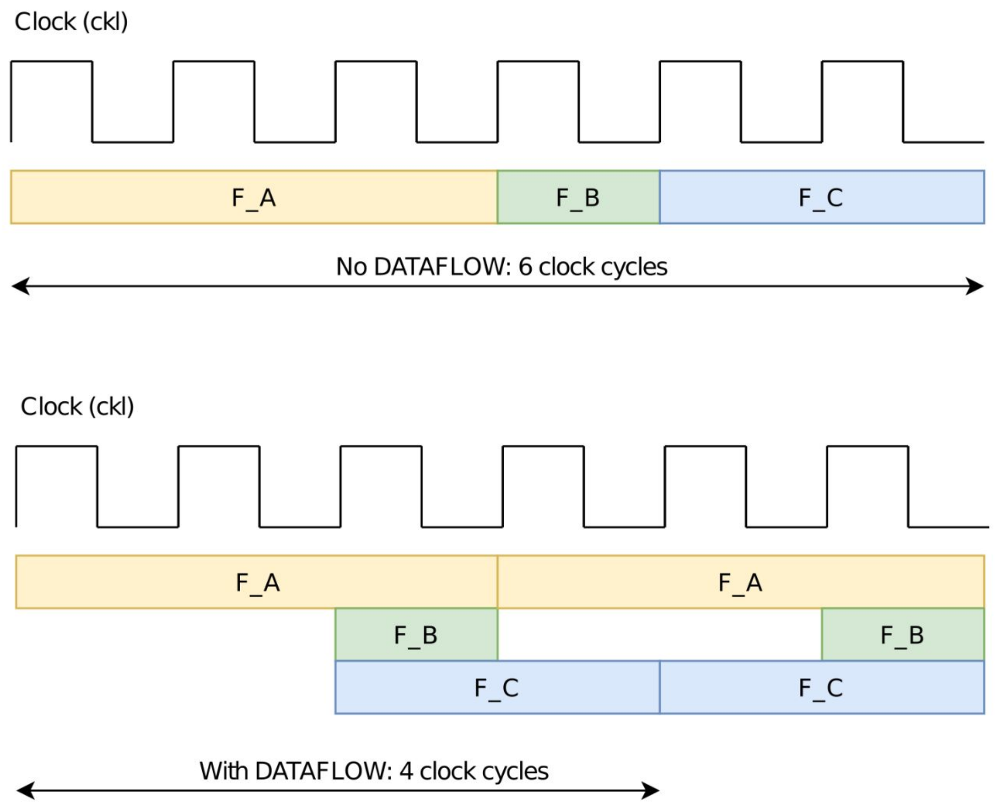
High-Level Synthesis

Hardware design - Directives

Loop + Dataflow

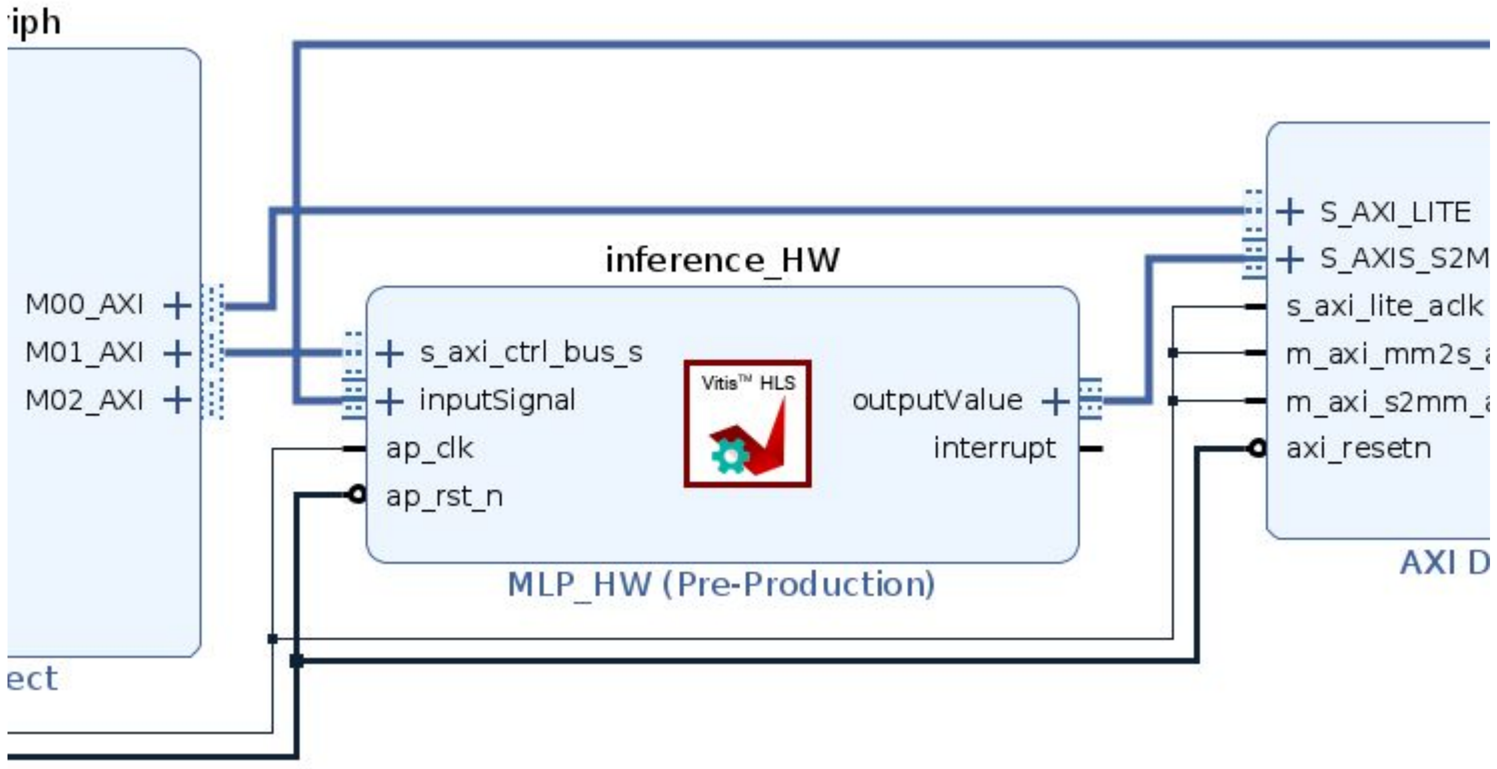
```
void function(a, b, c, d) {
     f_A(a, b, i1);
     f_B(c, i1, i2);
     f_C(i2, d);

return d;
}
```

F_A	Function f_A(a, b, i1)
F_B	Function f_B(c, i1, i2)
F_C	Function f_C(i2, d)



High-Level Synthesis

IP Core



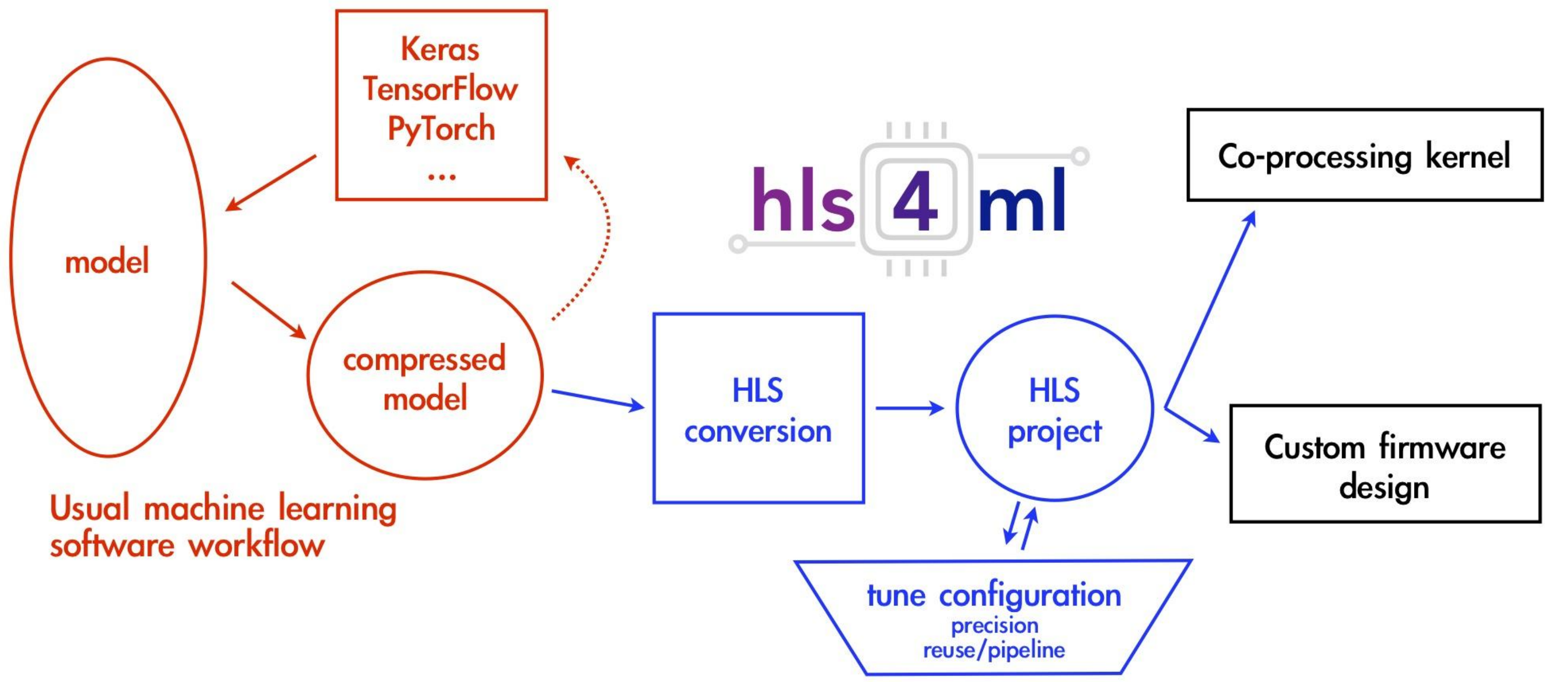
High-Level Synthesis for ML (hls4ml)

High-Level Synthesis for ML (hls4ml)

- Package for ML inference on SoC-FPGAs using HLS. (Duarte et. al)
- "Fast inference of deep neural networks (DNN) in FPGAs for particle physics" Duarte et al. [2]
- GitHub: https://github.com/fastmachinelearning/hls4ml-tutorial
- https://fastmachinelearning.org/hls4ml/

High-Level Synthesis for ML (hls4ml)

Design flow



High-Level Synthesis for ML (hls4ml)

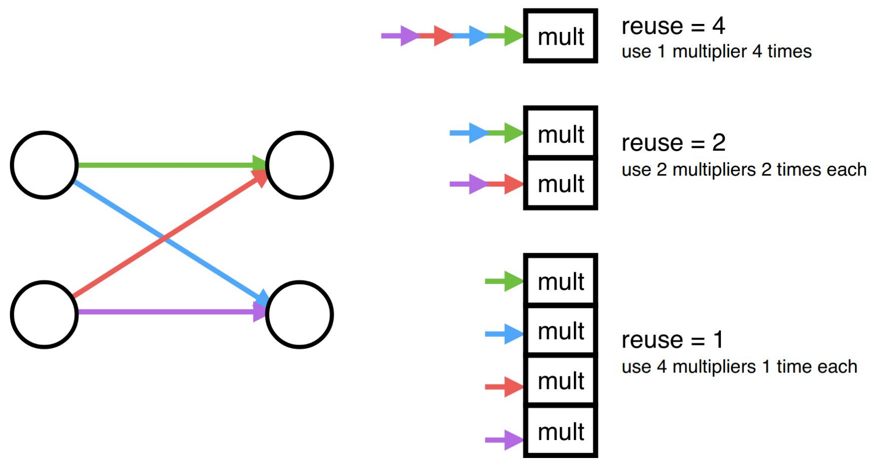
Features:

- HLS to create IP Core.
- Keras, TensorFlow, Pytorch.
- On-chip data structures.
- Quantization through ap_fixed data type in HLS.
    - typedef **ap_ufixed<10,8> din** (A 10-bit input: 8-bit integer value with 2 decimal places)
- Trade-off between resource utilization and latency/throughput.

High-Level Synthesis for ML (hls4ml)

Features:

- Pipelining to speed up the process by accepting new inputs after an initiation interval.
- Size/Compression
- Precision
- Dataflow/Resource Reuse
- Quantization Aware Training: Qkeras [3]

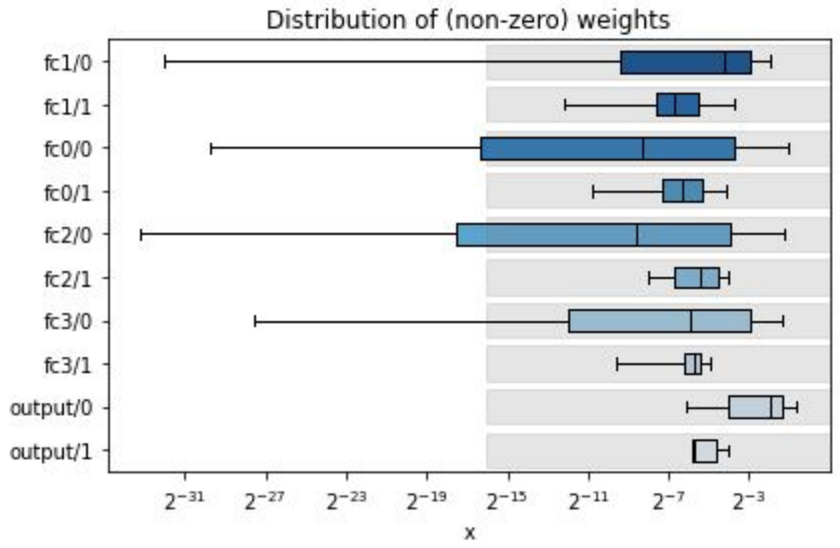


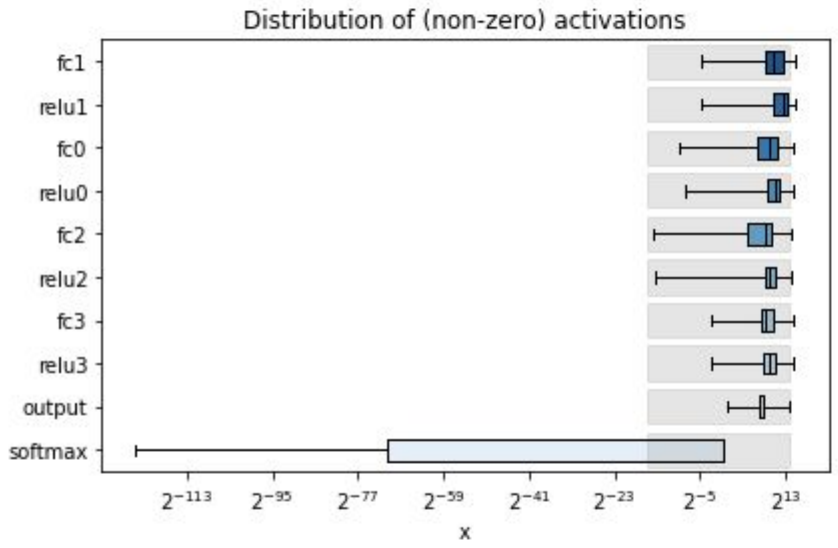
Reuse factor [2]

High-Level Synthesis for ML (hls4ml)

Features - Profiling

- Profiling to adjust precision
- **Method:** hls4ml.model.profiling.numerical





High-Level Synthesis for ML (hls4ml)

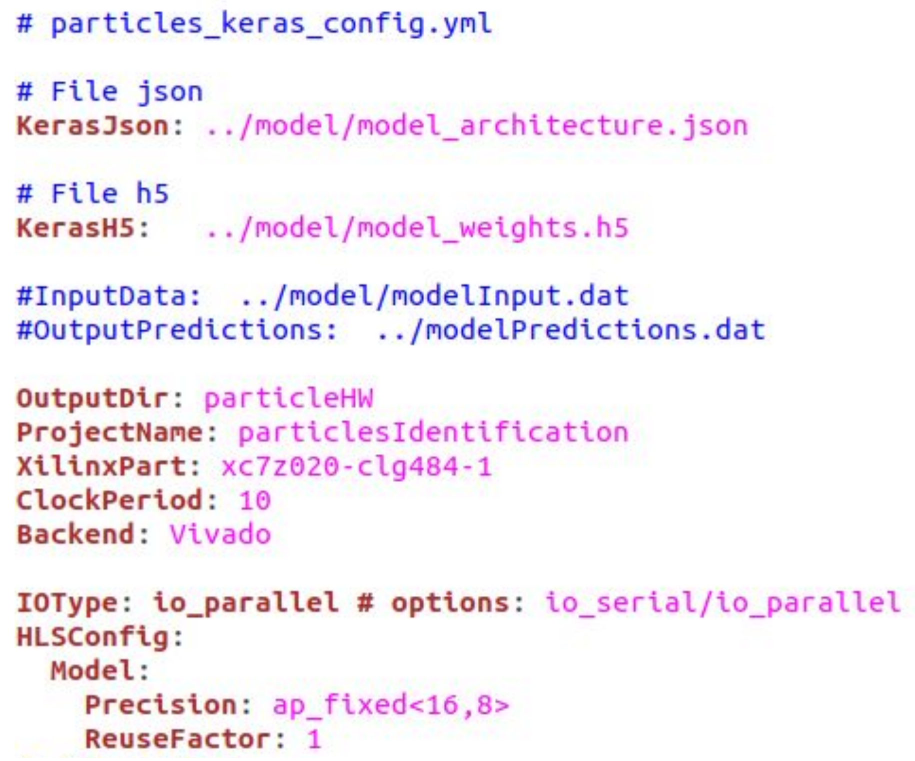
How we start with the tool?

- First, we have to download packages and dependencies. Then, we need to decide between:
    - Using command line
    - Using Jupyter Notebook

High-Level Synthesis for ML (hls4ml)

How we start with the tool? - Command line

- Configuration file (.yml).
- In this example, the file has the name model-config.yml
    - Files required: .json y .h5



```
# particles_keras_config.yml

# File json
KerasJson:  ../model/model_architecture.json

# File h5
KerasH5:    ../model/model_weights.h5

#InputData:   ../model/modelInput.dat
#OutputPredictions:   ../modelPredictions.dat

OutputDir: particleHW
ProjectName: particlesIdentification
XilinxPart: xc7z020-clg484-1
ClockPeriod: 10
Backend: Vivado

IOType: io_parallel # options: io_serial/io_parallel
HLSConfig:
  Model:
    Precision: ap_fixed<16,8>
    ReuseFactor: 1
```

High-Level Synthesis for ML (hls4ml)

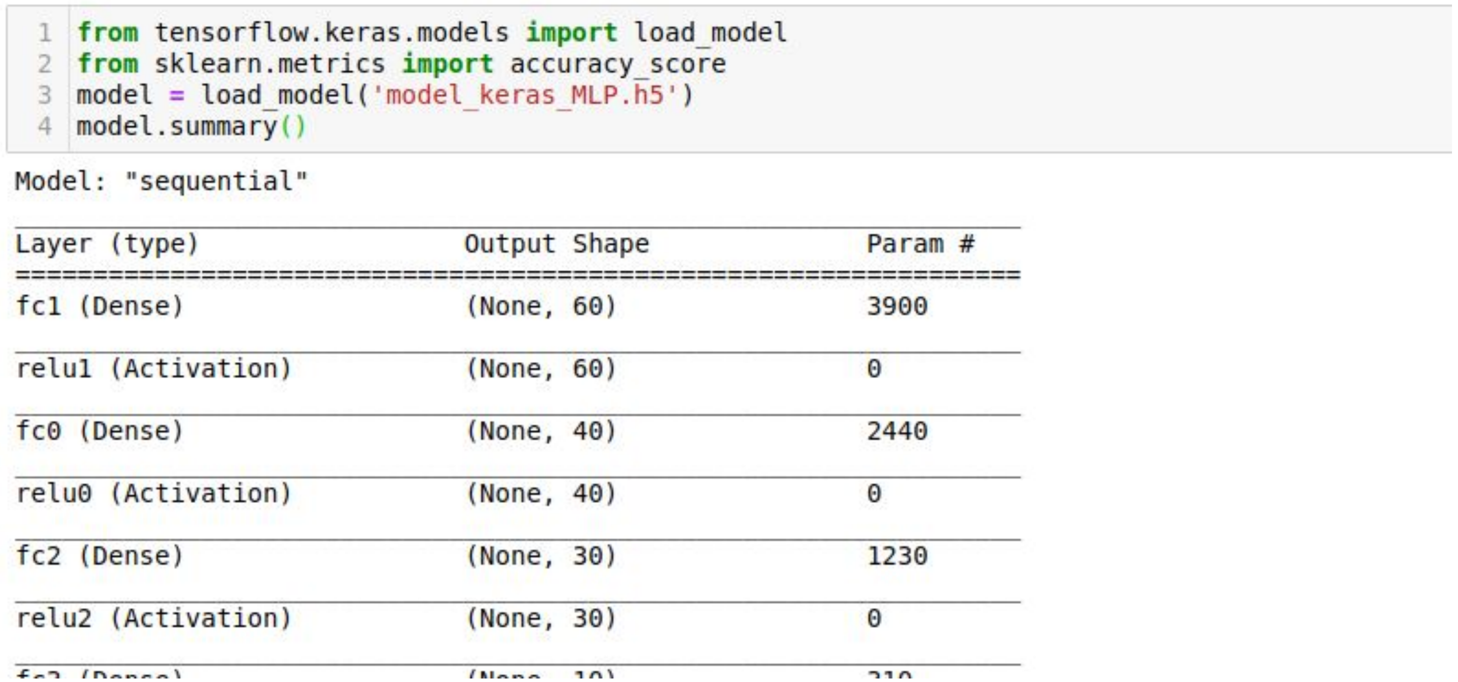
How we start with the tool? - Command line

- Commands for terminal execution
    - hls4ml convert -c model-config.yml
    - hls4ml build -p ProjectName -a
    - vivado_hls -f ProjectName.tcl "csim=1 synth=1 cosim=0 export=0"

- Following the information in the previous image, **ProjectName** was replaced by **particlesIdentification**:
    - hls4ml convert -c model-config.yml
    - hls4ml build -p particlesIdentification -a
    - vivado_hls -f particlesIdentification.tcl "csim=1 synth=1 cosim=0 export=0"

High-Level Synthesis for ML (hls4ml)

How we start with the tool? - Jupyter Notebook

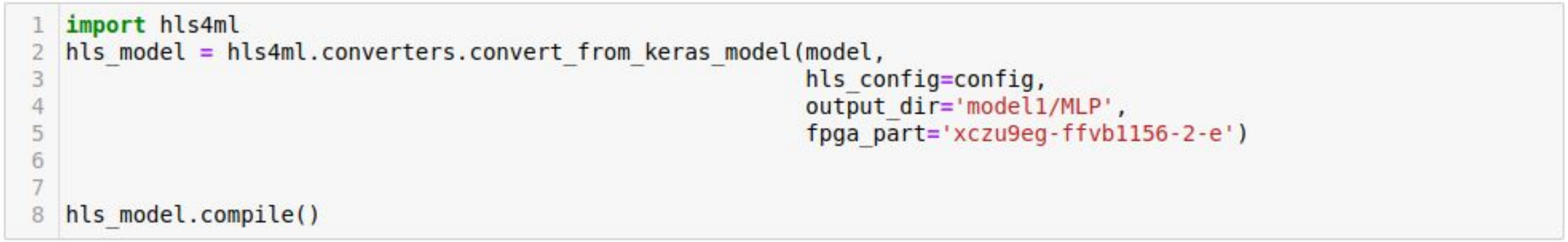


```python
from tensorflow.keras.models import load_model
from sklearn.metrics import accuracy_score
model = load_model('model_keras_MLP.h5')
model.summary()
```

```
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
```

High-Level Synthesis for ML (hls4ml)

How we start with the tool? - Jupyter Notebook

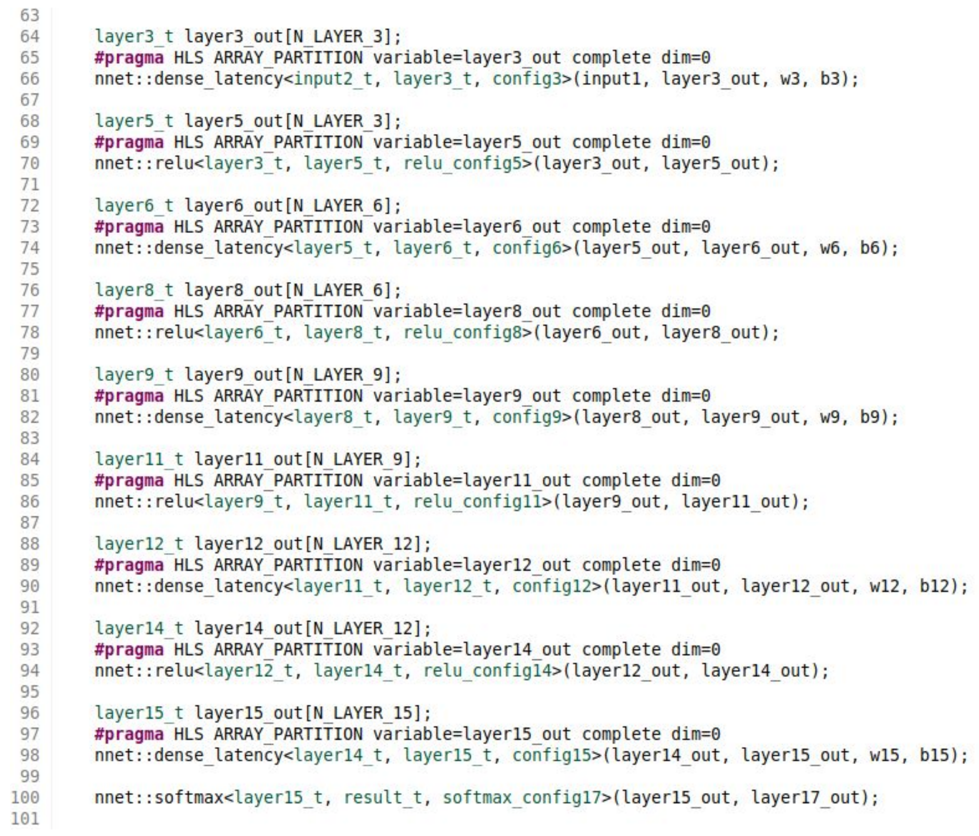


```python
import hls4ml
hls_model = hls4ml.converters.convert_from_keras_model(model,
                                                       hls_config=config,
                                                       output_dir='model1/MLP',
                                                       fpga_part='xczu9eg-ffvb1156-2-e')



hls_model.compile()
```

High-Level Synthesis for ML (hls4ml)

How we start with the tool? - Jupyter Notebook

Network description generated inside HLS project



```
layer3_t layer3_out[N_LAYER_3];
#pragma HLS ARRAY_PARTITION variable=layer3_out complete dim=0
nnet::dense_latency<input2_t, layer3_t, config3>(input1, layer3_out, w3, b3);

layer5_t layer5_out[N_LAYER_3];
#pragma HLS ARRAY_PARTITION variable=layer5_out complete dim=0
nnet::relu<layer3_t, layer5_t, relu_config5>(layer3_out, layer5_out);

layer6_t layer6_out[N_LAYER_6];
#pragma HLS ARRAY_PARTITION variable=layer6_out complete dim=0
nnet::dense_latency<layer5_t, layer6_t, config6>(layer5_out, layer6_out, w6, b6);

layer8_t layer8_out[N_LAYER_6];
#pragma HLS ARRAY_PARTITION variable=layer8_out complete dim=0
nnet::relu<layer6_t, layer8_t, relu_config8>(layer6_out, layer8_out);

layer9_t layer9_out[N_LAYER_9];
#pragma HLS ARRAY_PARTITION variable=layer9_out complete dim=0
nnet::dense_latency<layer8_t, layer9_t, config9>(layer8_out, layer9_out, w9, b9);

layer11_t layer11_out[N_LAYER_9];
#pragma HLS ARRAY_PARTITION variable=layer11_out complete dim=0
nnet::relu<layer9_t, layer11_t, relu_config11>(layer9_out, layer11_out);

layer12_t layer12_out[N_LAYER_12];
#pragma HLS ARRAY_PARTITION variable=layer12_out complete dim=0
nnet::dense_latency<layer11_t, layer12_t, config12>(layer11_out, layer12_out, w12, b12);

layer14_t layer14_out[N_LAYER_12];
#pragma HLS ARRAY_PARTITION variable=layer14_out complete dim=0
nnet::relu<layer12_t, layer14_t, relu_config14>(layer12_out, layer14_out);

layer15_t layer15_out[N_LAYER_15];
#pragma HLS ARRAY_PARTITION variable=layer15_out complete dim=0
nnet::dense_latency<layer14_t, layer15_t, config15>(layer14_out, layer15_out, w15, b15);

nnet::softmax<layer15_t, result_t, softmax_config17>(layer15_out, layer17_out);
```

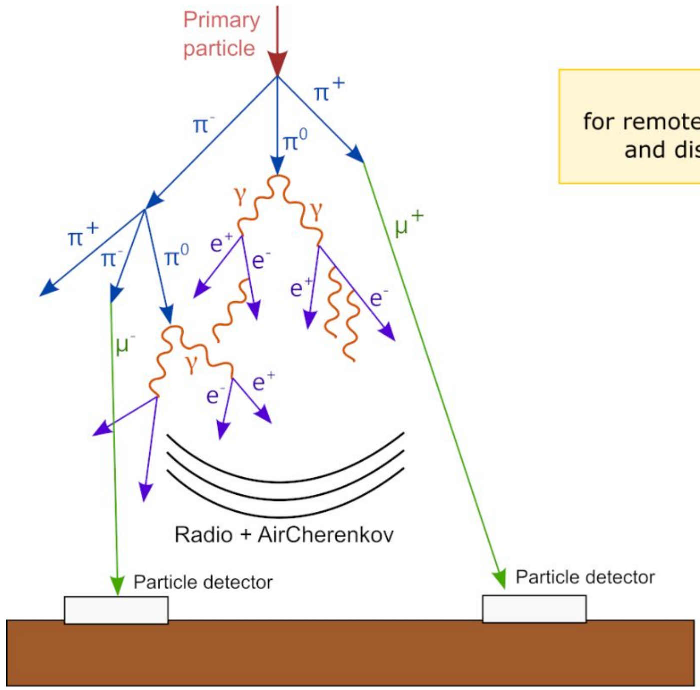
High-Level Synthesis for ML (hls4ml)

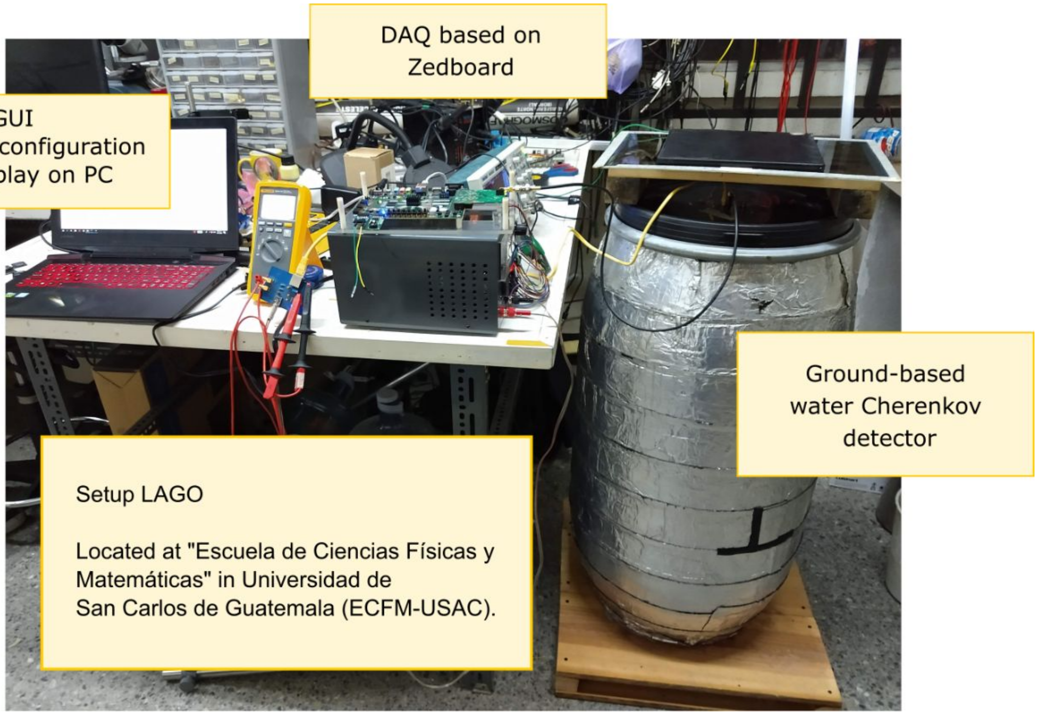
Case of study: Pulse Shape Discrimination for Water Cherenkov Detectors

High-Level Synthesis for ML (hls4ml)

Case of study: Pulse Shape Discrimination for Water Cherenkov Detectors

Experimental Setup




DAQ based on Zedboard
GUI for remote configuration and display on PC
Ground-based water Cherenkov detector

Setup LAGO

Located at "Escuela de Ciencias Físicas y Matemáticas" in Universidad de San Carlos de Guatemala (ECFM-USAC).

High-Level Synthesis for ML (hls4ml)

Case of study: Pulse Shape Discrimination for Water Cherenkov Detectors

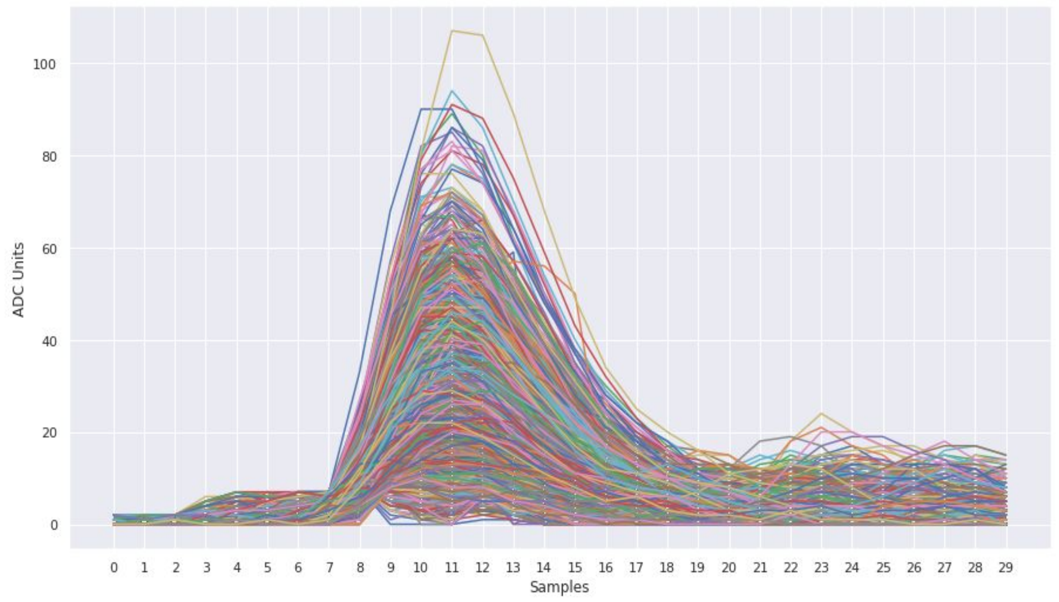
Experimental Setup

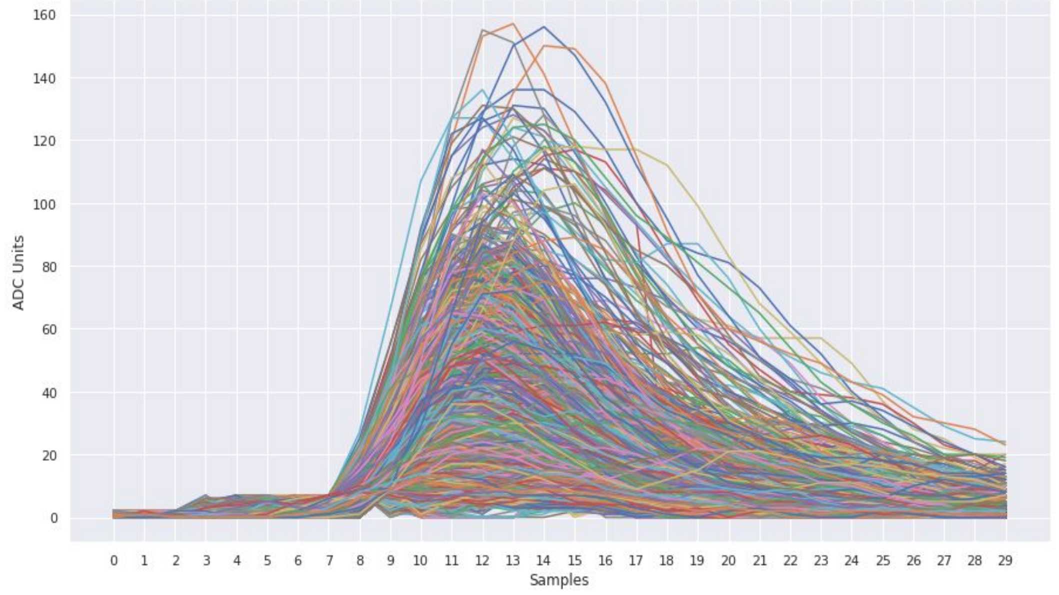
- Water Cherenkov detector (WCD) at the Escuela de Ciencias Físicas y Matemáticas - Universidad de San Carlos de Guatemala (ECFM-USAC).
- Feature extraction in the incoming signal to perform signal classification.
- Signal: 30 samples

High-Level Synthesis for ML (hls4ml)

Case of study: Pulse Shape Discrimination for Water Cherenkov Detectors

Different types of signals - Class 0 and 1

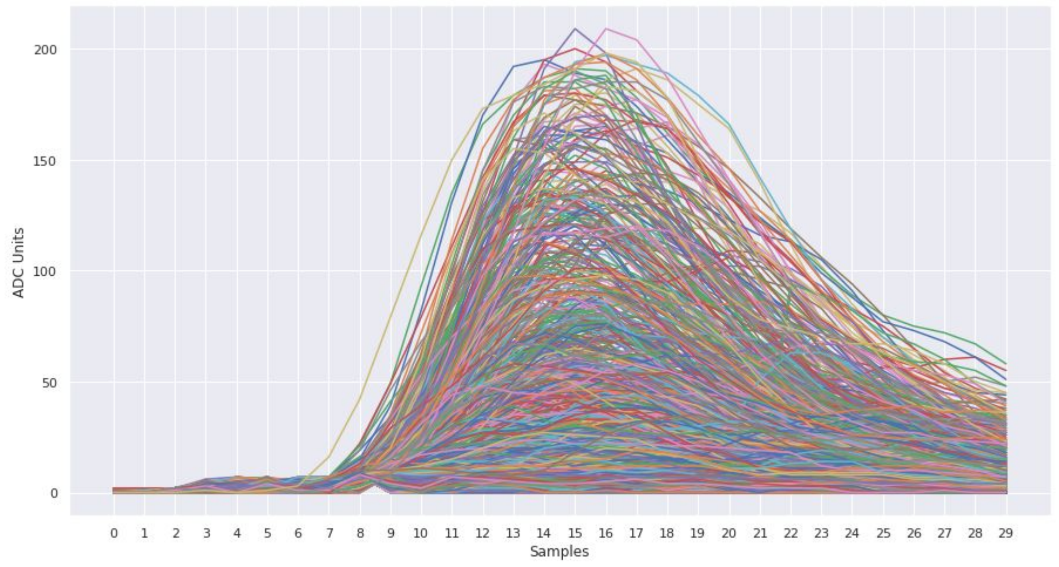


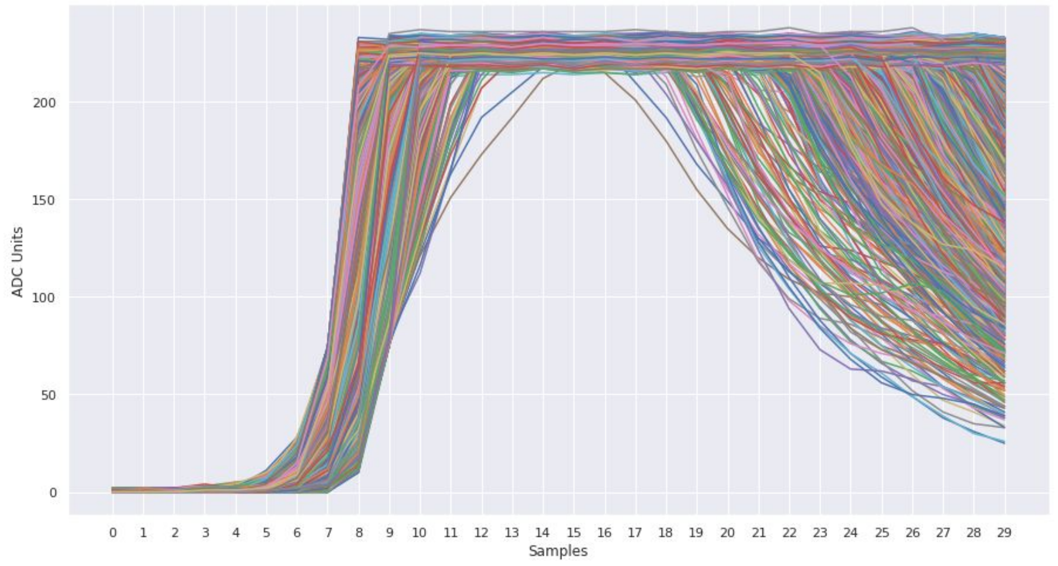


High-Level Synthesis for ML (hls4ml)

Case of study: Pulse Shape Discrimination for Water Cherenkov Detectors

Different types of signals - Class 2 and 3

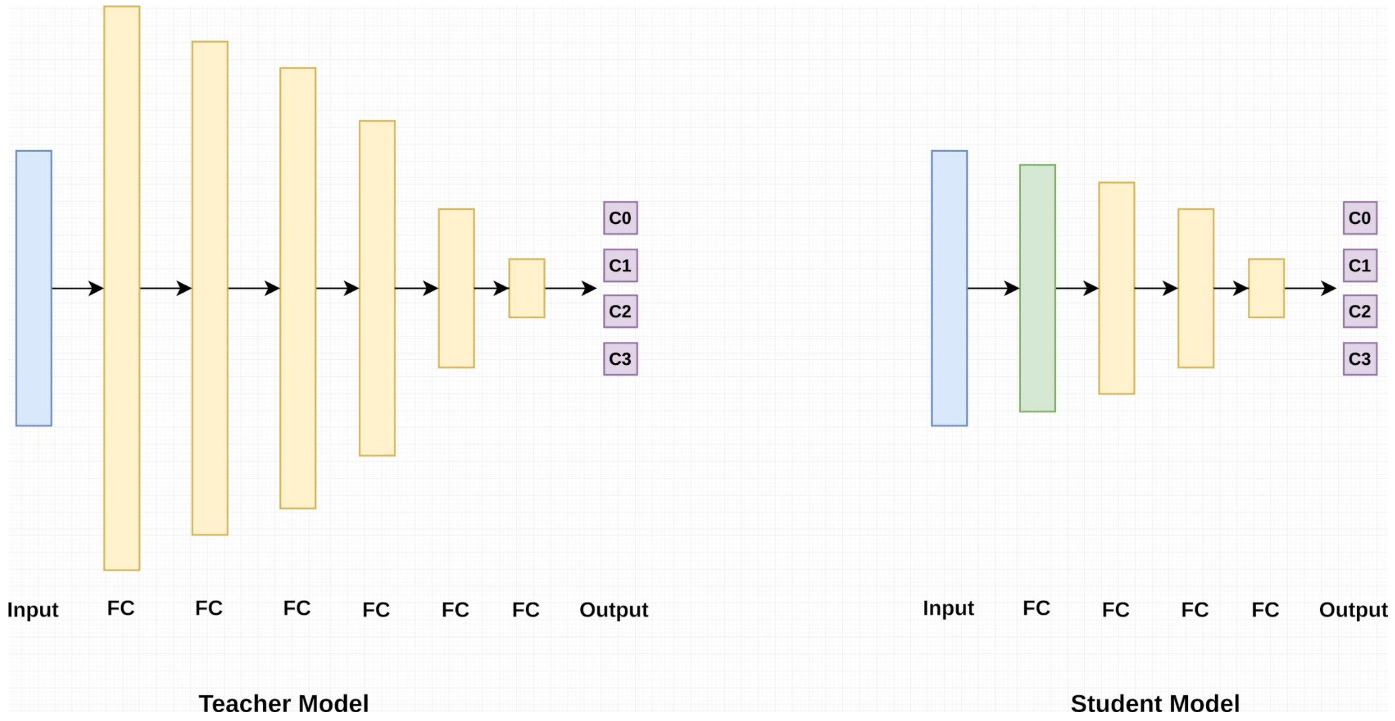




High-Level Synthesis for ML (hls4ml)

Case of study: Pulse Shape Discrimination for Water Cherenkov Detectors

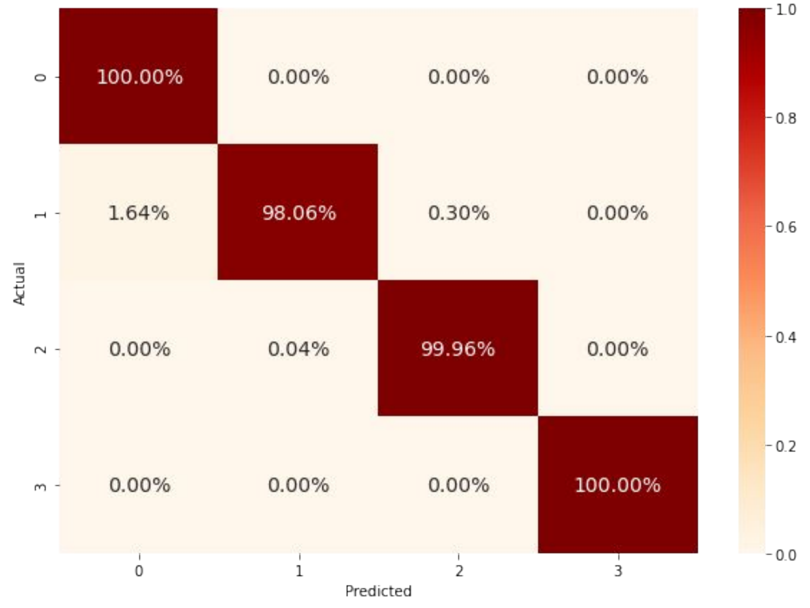
MLP architecture through an ensemble of compression techniques: Distillation, Quantization and Pruning

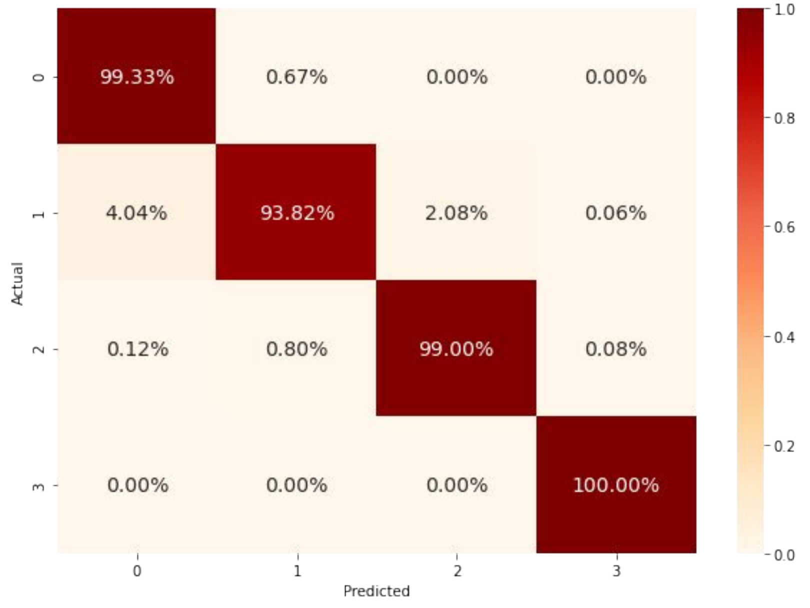


High-Level Synthesis for ML (hls4ml)

Case of study: Pulse Shape Discrimination for Water Cherenkov Detectors

Confusion Matrix before (left) and after (right) compression
Total params reduction: From 31,514 to 984
Overall accuracy: From 99.4% to 97%

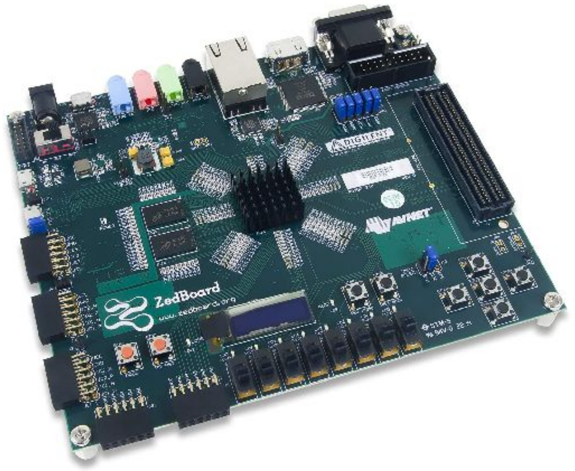




High-Level Synthesis for ML (hls4ml)

Case of study: Pulse Shape Discrimination for Water Cherenkov Detectors

Define the SoC architecture



High-Level Synthesis for ML (hls4ml)

Case of study: Pulse Shape Discrimination for Water Cherenkov Detectors

HLS reports - Clock @5ns

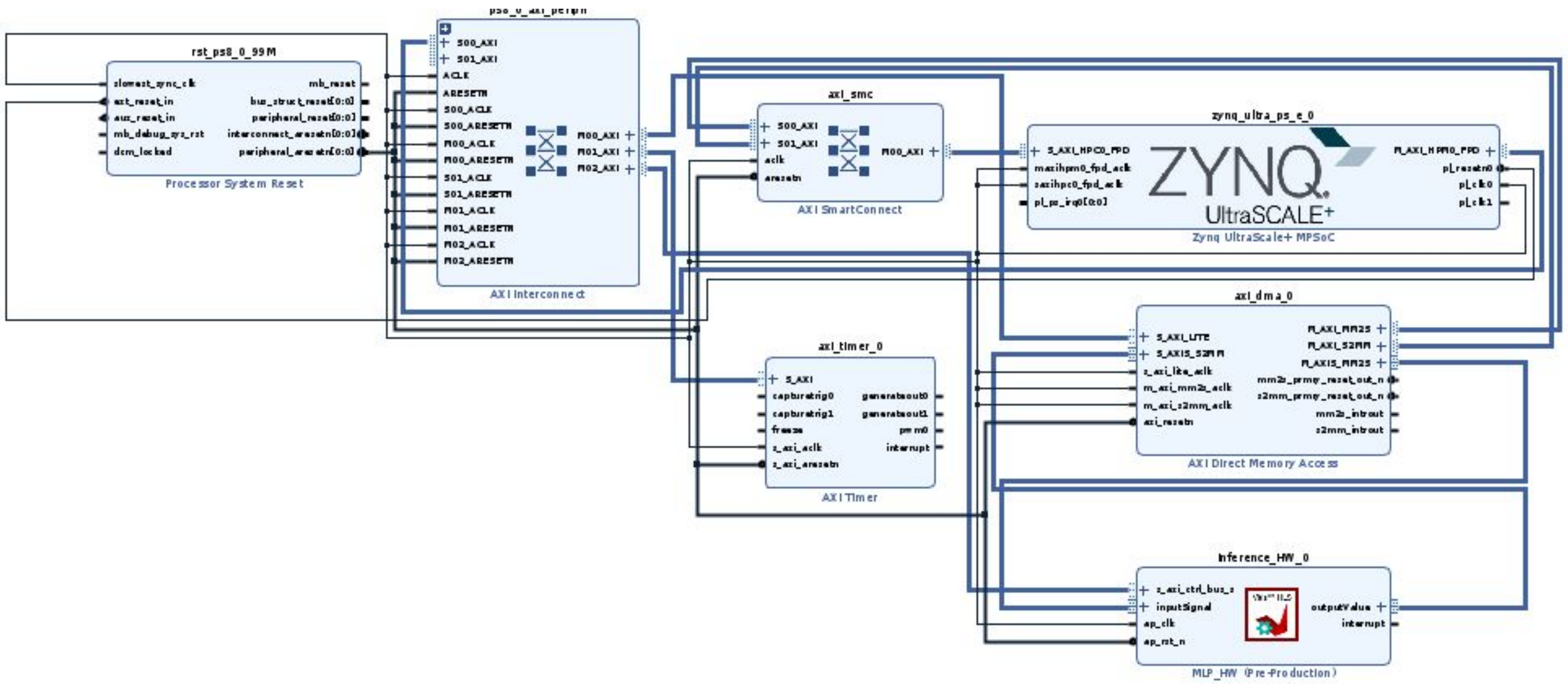
	Latency [clk]*	LUT	FF	BRAM	DSP
			PYNQ		
Sol_1_rf1	39	69%	75%	0%	**369%**
Sol_2_rf8	55	**72%**	24%	0%	50%
			KRIA		
Sol_3_rf1	20	49%	12%	0%	**69%**
			ZCU102		
Sol_4_rf1	20	20%	5%	0%	**34%**

*Latency only for inference

High-Level Synthesis for ML (hls4ml)

Case of study: Pulse Shape Discrimination for Water Cherenkov Detectors

Hardware created with Vivado IP Integrator



High-Level Synthesis for ML (hls4ml)

Case of study: Pulse Shape Discrimination for Water Cherenkov Detectors

Final resource usage reported by Vivado

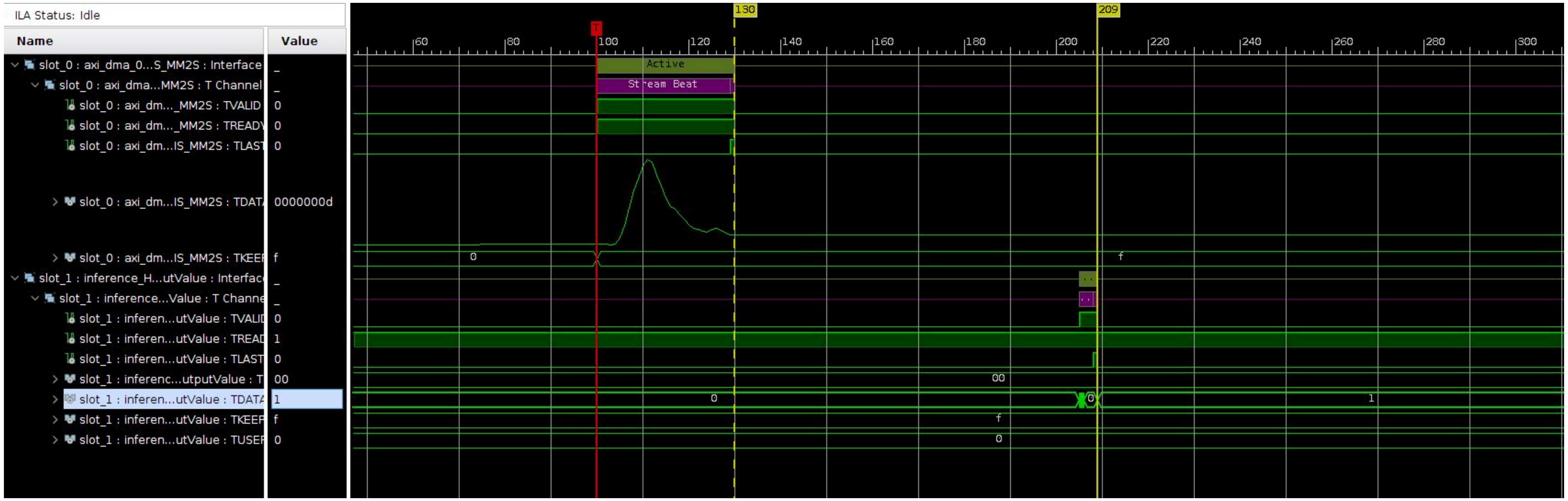
	LUT	FF	BRAM	DSP
PYNQ	44.6%	23%	34%	**50%**
KRIA	30%	7.8%	33%	**69%**
ZCU102	7.8%	2.8%	9.9%	**27%**

High-Level Synthesis for ML (hls4ml)

Case of study: Pulse Shape Discrimination for Water Cherenkov Detectors

Family / Part: Zynq-7000 / xc7z020clg400-1

Clock cycles for inference: 79 (Estimated by HLS: 55)


ILA Status: Idle

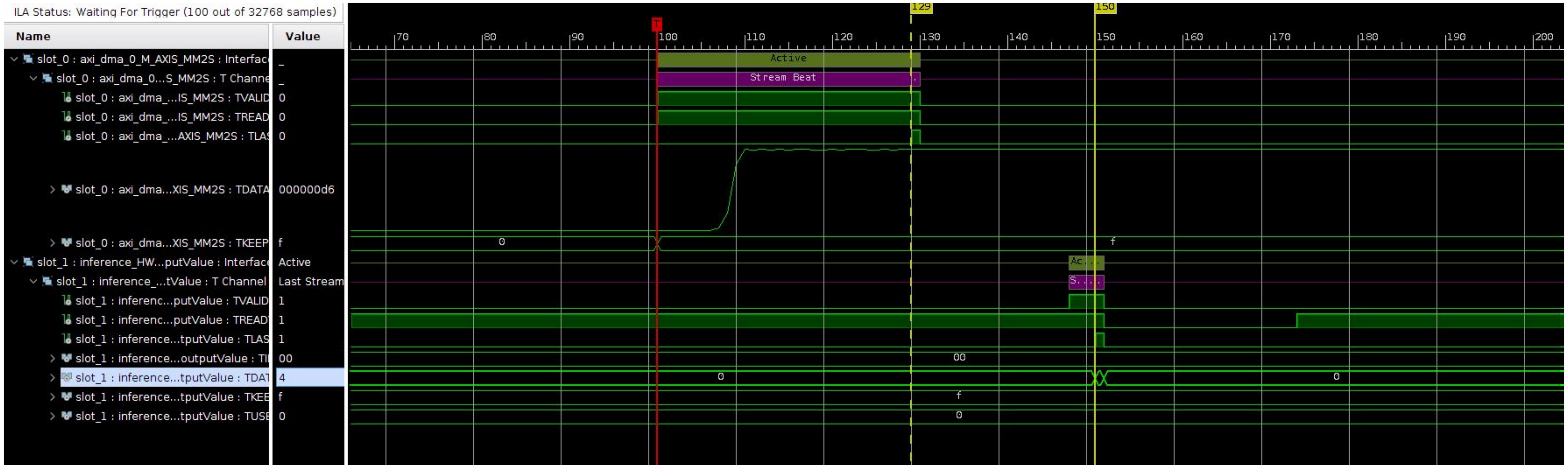
Name	Value
slot_0 : axi_dma_0...S_MM2S : Interface	_
slot_0 : axi_dma...MM2S : T Channel	_
slot_0 : axi_dm..._MM2S : TVALID	0
slot_0 : axi_dm..._MM2S : TREADY	0
slot_0 : axi_dm...IS_MM2S : TLAST	0
slot_0 : axi_dm...IS_MM2S : TDAT/	0000000d
slot_0 : axi_dm...IS_MM2S : TKEEI	f
slot_1 : inference_H...utValue : Interfac	_
slot_1 : inference...Value : T Channe	_
slot_1 : inferen...utValue : TVALII	0
slot_1 : inferen...utValue : TREAI	1
slot_1 : inferen...utValue : TLAST	0
slot_1 : inferenc...utputValue : T	00
slot_1 : inferen...utValue : TDAT/	1
slot_1 : inferen...utValue : TKEEF	f
slot_1 : inferen...utValue : TUSEF	0

High-Level Synthesis for ML (hls4ml)

Case of study: Pulse Shape Discrimination for Water Cherenkov Detectors

Family / Part: zynquplus / xczu9eg-ffvb1156-2-e

Clock cycles for inference: 21 (Estimated by HLS: 20)



ILA Status: Waiting For Trigger (100 out of 32768 samples)

Name	Value
slot_0 : axi_dma_0_M_AXIS_MM2S : Interfac	_
slot_0 : axi_dma_0...S_MM2S : T Channe	_
slot_0 : axi_dma_...IS_MM2S : TVALID	0
slot_0 : axi_dma_...IS_MM2S : TREAD	0
slot_0 : axi_dma_...AXIS_MM2S : TLAS	0
slot_0 : axi_dma...XIS_MM2S : TDATA	000000d6
slot_0 : axi_dma...XIS_MM2S : TKEEP	f
slot_1 : inference_HW...putValue : Interfac	Active
slot_1 : inference_...tValue : T Channel	Last Stream
slot_1 : inferenc...putValue : TVALID	1
slot_1 : inferenc...putValue : TREAD	1
slot_1 : inference...tputValue : TLAS	1
slot_1 : inference...outputValue : TI	00
slot_1 : inference...tputValue : TDAT	4
slot_1 : inference...tputValue : TKEE	f
slot_1 : inference...tputValue : TUSE	0

References

[1] Duarte, J.; Han, S.; Harris, P.; Jindariani, S.; Kreinar, E.; Kreis, B.; Ngadiuba, J.; Pierini, M.; Rivera, R.; Tran, N.; et al. Fast inference of deep neural networks in FPGAs for particle physics. J. Instrum. 2018, 13, P07027, doi:10.1088/1748-0221/13/07/p07027.

[2] Cheng, Y.; Wang, D.; Zhou, P.; Zhang, T. A Survey of Model Compression and Acceleration for Deep Neural Networks. arXiv 2017, arXiv:1710.09282

[3] Coelho, J.; Kuusela, A.; Zhuang, H.; Aarrestad, T.; Loncar, V.; Ngadiuba, J.; Pierini, M.; Summers, S. Ultra Low-latency, Low-area Inference Accelerators using Heterogeneous Deep Quantization with QKeras and hls4ml. arXiv 2020, arXiv:2006.10159.

[4] Garcia, L.G.; Molina, R.S.; Crespo, M.L.; Carrato, S.; Ramponi, G.; Cicuttin, A.; Morales, I.R.; Perez, H. Muon–Electron Pulse Shape Discrimination for Water Cherenkov Detectors Based on FPGA/SoC. Electronics 2021, 10, 224. https://doi.org/10.3390/electronics10030224

[5] Vivado Design Suite User Guide - High-Level Synthesis - UG902 (v2019.1) July 12, 2019

The Abdus Salam
International Centre
for Theoretical Physics

Thank you!