1st Mesoamerican Workshop on Reconfigurable X-ray Scientific Instrumentation for Cultural Heritage

Lab 5: ML on SoC-FPGA

Antigua Guatemala, June 2025

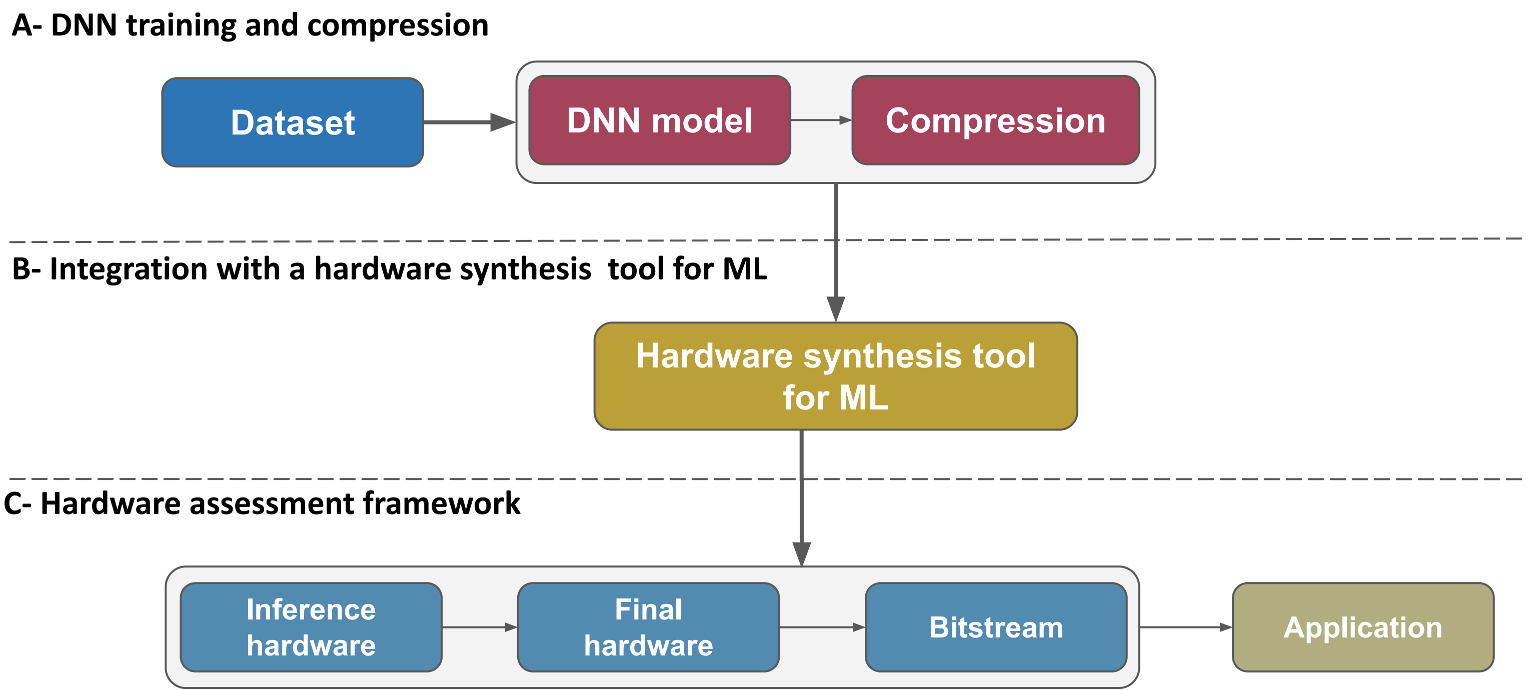
Romina Soledad Molina, Ph.D.

Objectives

- Learn how to deploy ML-based models on SoC-FPGA platforms.
- Learn and understand the workflow to compress ML-based model for resource constrained devices.
- Acquire knowledge of hls4ml package.
- Perform the generation and instantiation of the HLS-based ML IP core previously designed through Vitis HLS tool.
- Integrate and verify the complete hardware design.

Bridging Machine Learning and FPGAs

End-to-end workflow

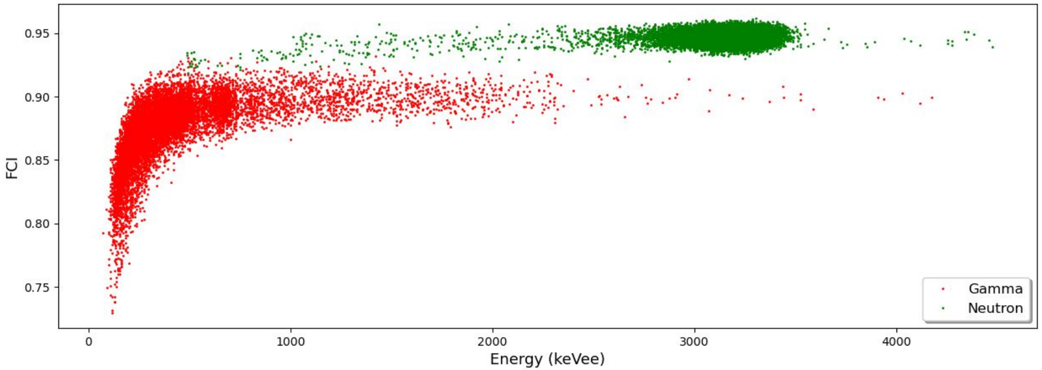


Available at https://github.com/RomiSolMolina/workflowCompressionML

Case study:
Gamma/neutron discrimination

Case study: Gamma/neutron discrimination

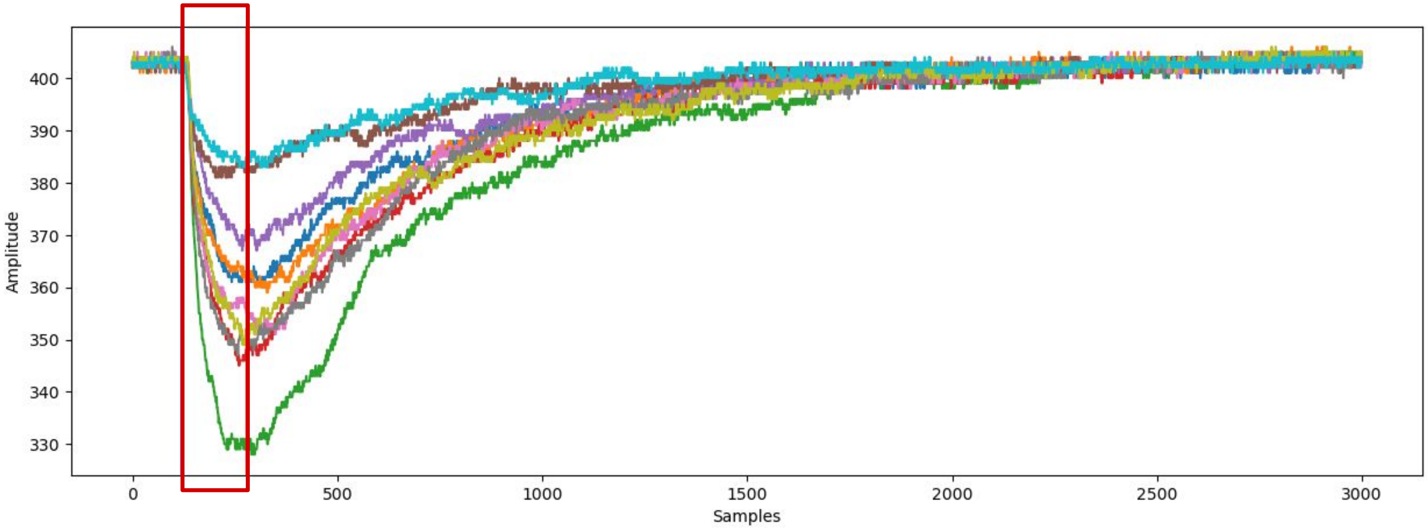
- The experimental data for this project were collected at the Neutron Science Facility, IAEA Laboratories, in Seibersdorf, Austria.
- The image below depicts the **gamma/neutron distribution** obtained using the method described in [GN], employed to generate the labeled dataset, consisting of two classes: class 0 corresponding to gamma and class 1 to neutron.



[GN] 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

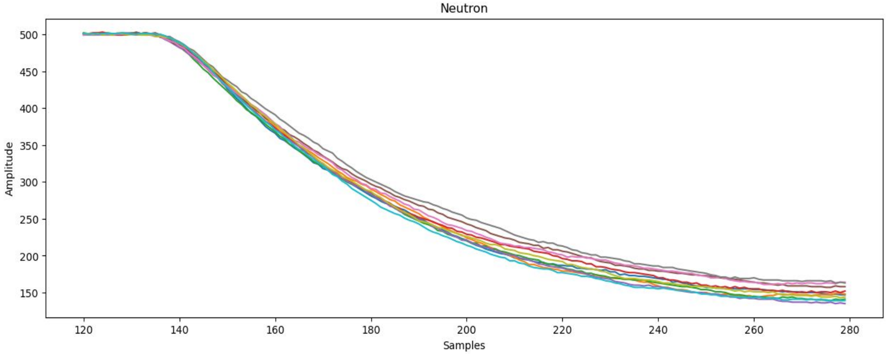
Case study: Gamma/neutron discrimination

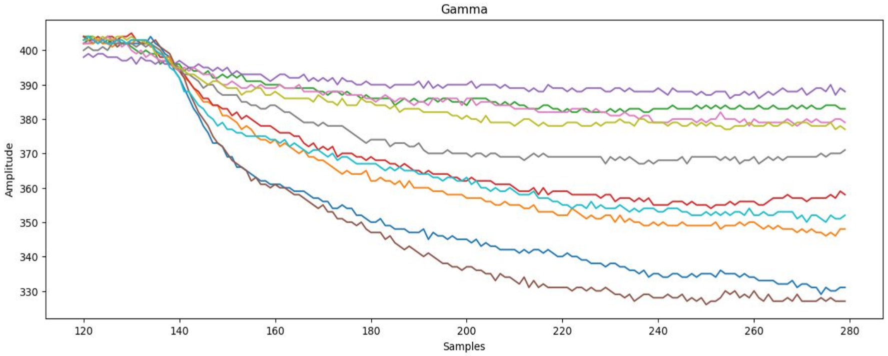
- The **primary information** in these types of signals is concentrated in the **leading edge**.
- The image below displays some of the original signal traces, along with the corresponding window that highlights the portion of the signal being cropped.



Case study: Gamma/neutron discrimination

- For this project, the signals used will consist of **161 samples**, extracted specifically from the leading edge.
- Samples of the final gamma and neutron traces are shown in the following figures.
- With this information, a dataset was generated to be used for the training, validation, and testing of the ML-based model.

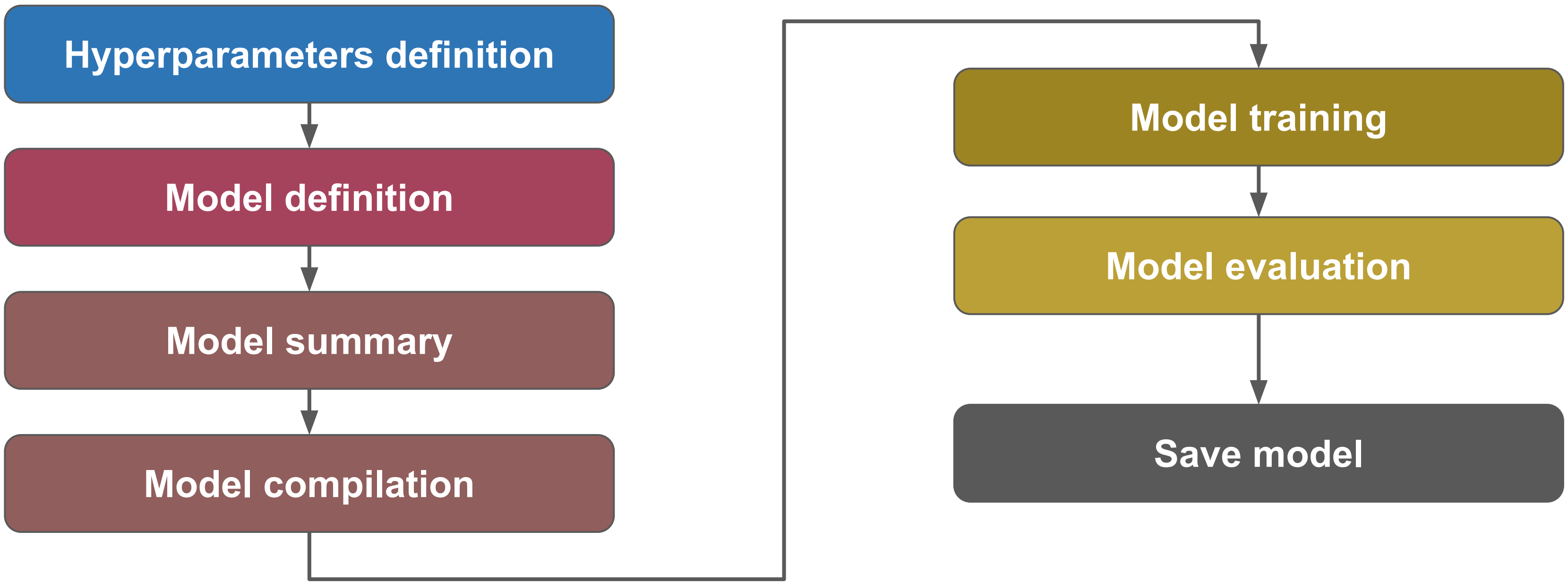




A. Model training and compression

Machine learning

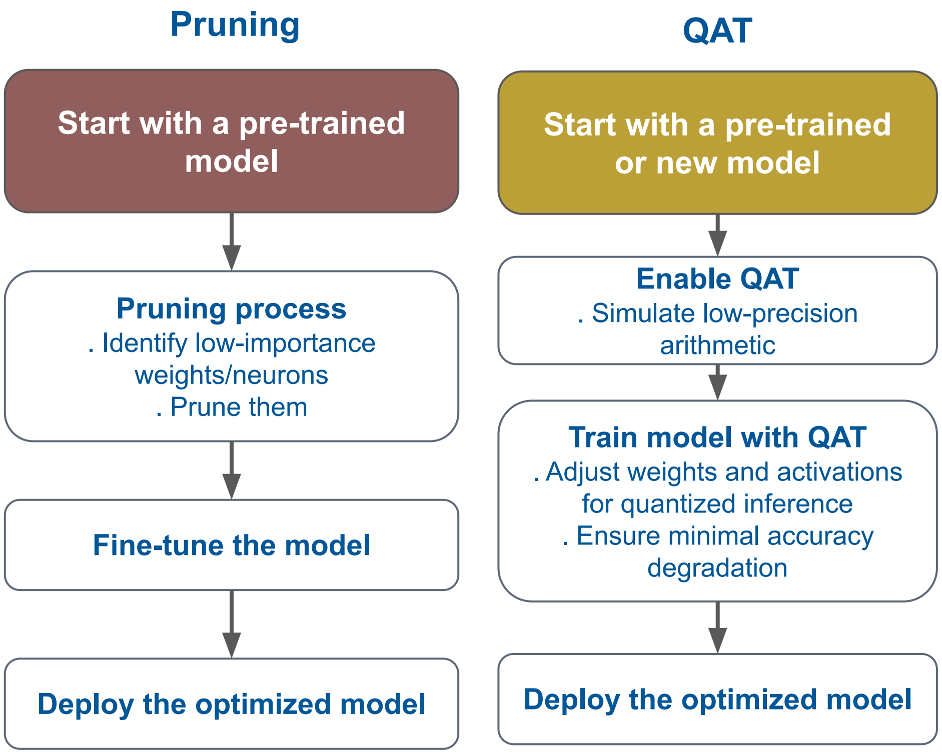
Training - General steps Keras+TensorFlow



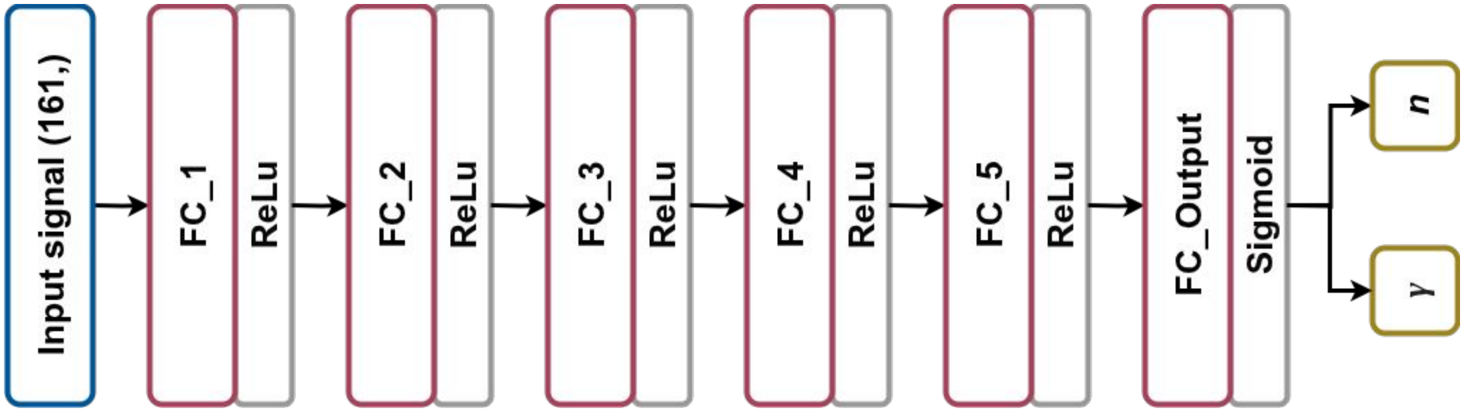
Machine learning

Compression

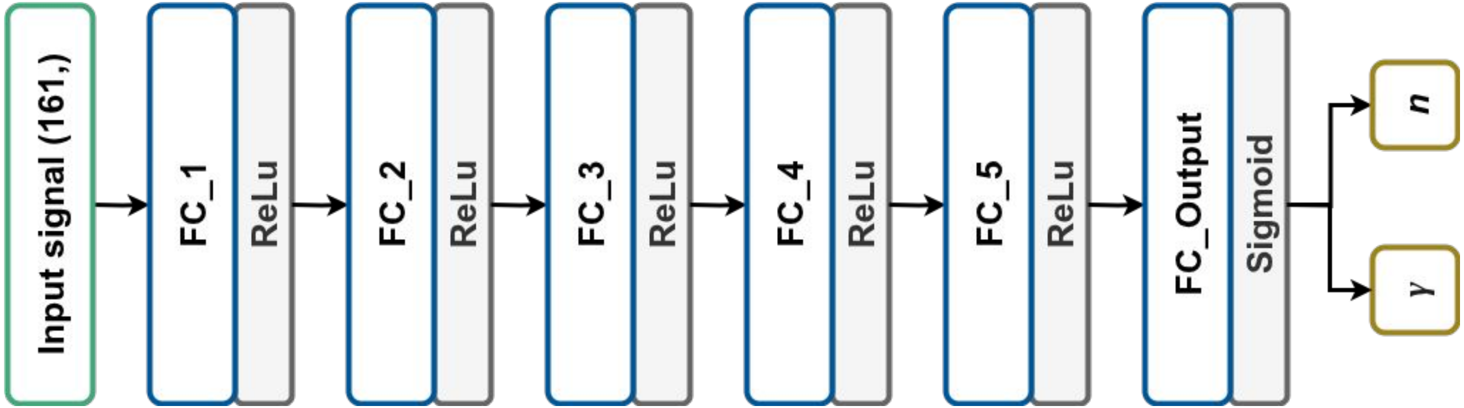
- Teacher training
- Compression
  - Pruning
  - Quantization-aware training (QAT)
  - Knowledge distillation (KD)
    - Student model
  - KD + QAT



Teacher architecture

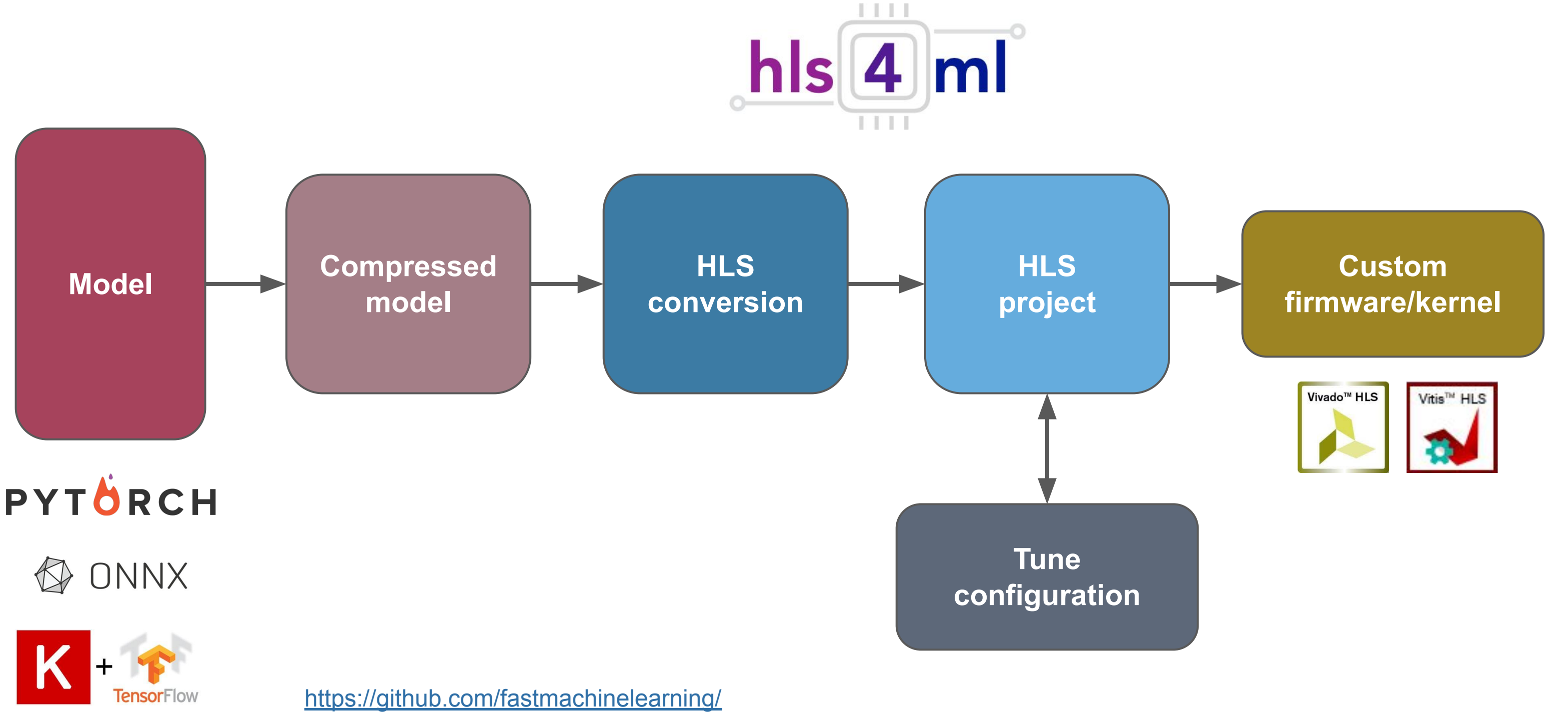


Student architecture



B. Integration with a hardware synthesis tool for ML

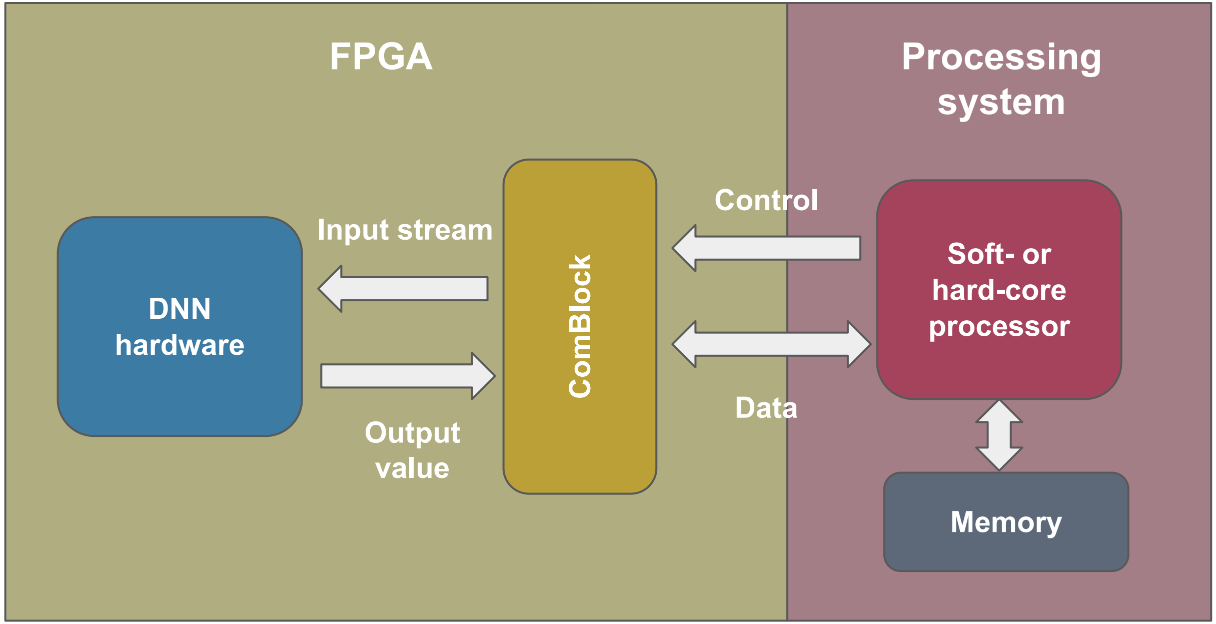
Integration with a hardware synthesis tool for ML



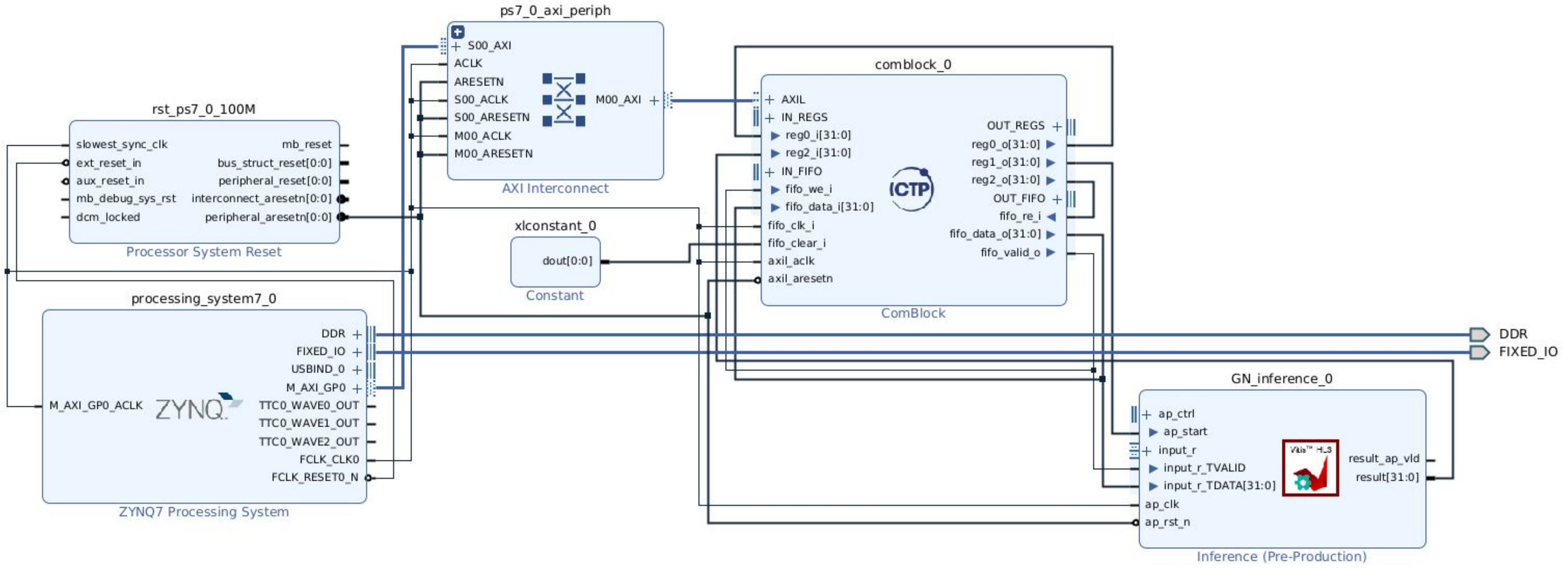
https://github.com/fastmachinelearning/

C. Hardware assessment framework

Hardware assessment framework



Hardware assessment framework



The Abdus Salam International Centre for Theoretical Physics

IAEA

unesco

1st Mesoamerican Workshop on Reconfigurable X-ray Scientific Instrumentation for Cultural Heritage

Lab 5: ML on SoC-FPGA

Antigua Guatemala, June 2025

Romina Soledad Molina, Ph.D.

IAEA International Atomic Energy Agency Atoms for Peace and Development

Secretaría Nacional de Ciencia y Tecnología

IUPAP

USAC TRICENTENARIA Universidad de San Carlos de Guatemala

The Abdus Salam International Centre for Theoretical Physics

Machine learning

Training - General steps Keras+TensorFlow

General overview

- The first two steps focus on **defining the hyperparameters and configuring the machine learning architecture**. Afterward, a model summary provides an overview of how the model was constructed.

- Once the model is created, parameters such as the optimizer, loss function, and metrics are configured using the **model.compile()** function.

- Finally, training is performed with the **model.fit()** function, where the dataset, batch size, number of epochs, and callbacks, among other settings, are specified.

Machine learning

Training - General steps Keras+TensorFlow

```
model= Sequential([

    Flatten(input_shape=(w, h)),
    Dense(256, activation='relu'),
    Dense(64, activation='relu'),
    Dense(32, activation='relu'),
    Dense(n_classes, activation='softmax')

])
```

Model definition

```
model.summary()
```

Model summary

Machine learning

Training - General steps Keras+TensorFlow

```
learningRate = 0.001
optimizer = Adam(learningRate)
Epochs = 32
Batch = 16
```

Defining some of the hyperparameters

Machine learning

Training - General steps Keras+TensorFlow

```
model.compile(loss='sparse_categorical_crossentropy', optimizer=op, metrics=['accuracy'])
```

Model compile

Loss: A metric that measures how far the model's predictions are from the actual values.
Optimizer: An algorithm that adjusts the weights of the neural network to minimize the loss function.
Learning Rate: A hyperparameter that controls the size of the adjustments the optimizer makes to the model's weights during each iteration.
Metrics: Additional values monitored during training to evaluate the model's performance. For example, accuracy (used in classification).

Machine learning

Training - General steps Keras+TensorFlow

```
history = model.fit(x_train_norm, y_train, epochs= 32, batch_size = 50, validation_split=0.2)
```

Model fit

x_train_norm: normalized dataset obtained by applying a transformation to x_train.

y_train: labels (or expected values) corresponding to the training data.

batch: number of samples processed before updating the model's weights.

epochs: number of times the model will go through the entire training dataset.

validation_split: percentage of the training dataset (x_train, y_train) reserved for validation.

Machine learning

Training - General steps Keras+TensorFlow

Plot the Accuracy and Loss from the **history** variable during training