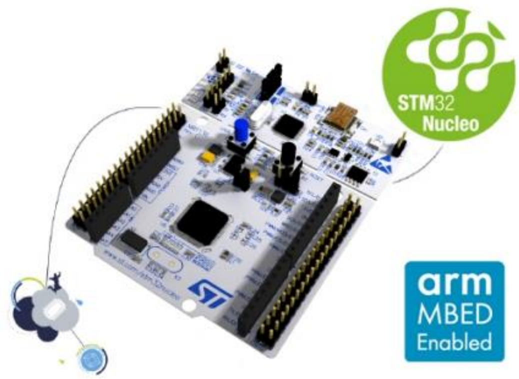
Lab 3 on embedded Artificial Intelligence on microcontroler

Polytech Nice Sophia

During this lab, you will use the software tools installed during Lab1 to optimize artificial neural networks for execution on the embedded platform Nucleo-64 depicted in the following figure.



Part I. Train a network

In this part, your goal is to define a Convolutional Neural Network reaching **between 98,5 and 99%** of good recognition (accuracy) on the MNIST Dataset. The accuracy of your model must be confirmed by an average **on 3 learning** and the validation **on 100 inferences** on laptop and on target!

(Obviously, the statistical studies would need more trials, but it is here a first introduction to CNN exploration)

To reach better accuracy you can change the following hyper-parameters of your CNN in the Build model and Train model parts of TF script:

- The number of **filters** in each convolution (*Conv2D*) layer
- The size of the **kernels** in each convolution layer (by default set to 3x3)
- The number of **Convolution layers**
- The use of ***MaxPool2D*** layers between *Conv2D* layers
- The number of ***Dense* layers**
- The number of **neurons** in each dense layer (except the output layer)
- The **activation function** in each layer
- The number of **epochs** of learning (how many time the network will learn the entire dataset)

When you get the expected accuracy, fill the following table as your result.

Layer	Output shape	Number of parameters	Kernel (If conv2D)
Input			
Total trainable parameters			
Number of Epochs			

Final model	Accuracy on test		Loss on test		Accuracy on validation
Learn 1					
Learn 2					
Learn 3					
Results	Average	Std deviation	Average	Std deviation	

Part II. Evaluate the performance of a network

Once you finished part I with a satisfying model, open STM32CubeIDE and import your model as explained in Lab1 (additional software -> network -> browse and select the corresponding .h5 file).

- **Analyze the original model layer per layer**
    - If your model does not satisfy the memory requirements, come back to tensor flow and reduce the size of the network (the number of parameters)
    - If the model is correct, fill the following table

	Type	Param #	MACC	MACC (%)	ROM (bytes)	ROM (%)	Bytes per Param
Layer 1							
Layer 2							
…							
TOTAL							

- **Validate on desktop and select compression factor**

Results during inference	Number of inferences (test dataset)	Accuracy	RMSE	MAE	MACC	ROM (bytes)
Original model (result from TF)	100					
Without compression						
Compression X4						
Compression X8						

Part III. Validate a network on target

Results during inference	Number of inferences (test dataset)	Accuracy without compression	RMSE	Total latency (ms)	CPU cycles	Cycles / MACC
Original model (result from TF)	100					
Model validated on Laptop without compression						
Model validated on Target without compression						
Model validated on Target with compression X4						
Model validated on Target with compression X8						

Part IV. Validation on the complete Test Dataset

Modify the initial Tensor Flow script in order to validate your model on the complete Test (10.000 inferences), and complete the table. (No learning is needed)

	Inferences	Accuracy	RMSE	MAE
Original model	100			
Original model under TF	10 000			
Validate model on desktop				
Validate model on desktop X4				
Validate model on desktop X8				

What are your conclusions on the implementation of CNN onto embedded platform? Summarize the observations made on accuracy, latency, memory footprint, compression…