Lab 2 on embedded Artificial Intelligence on microcontroler

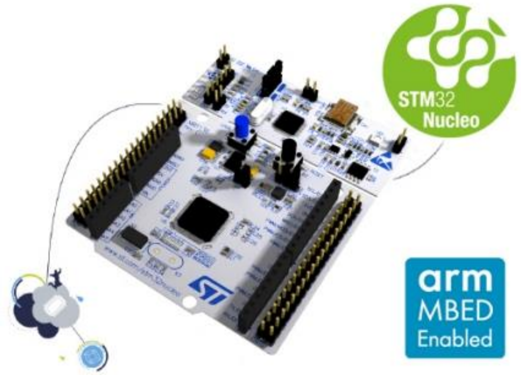
Polytech Nice Sophia

During this lab, you will make a first programmation of AI-based algorithms onto MCU (MicroControler Unit). This steps will then be reused during next Labs.

Hardware target

The hardware target is a Nucleo 64 board, equipped with a STM32L476 MCU. You will also need a USB type A to USB mini-B cable. Please look if you have a cable of this type, we don't have enough yet.

This MCU is based on the ARM Cortex M4 architecture and runs at a frequency of 80 MHz. The board provides 1 MB Flash and 128 KB SRAM.



Part1. Learn Neural Networks with Docker (each time you will train a CNN)

1. Connect to moodle, and download the provided TensorFlow script to train (during the lab) a predefined neural network
2. Open a windows powershell and go to the directory where you get the TF script and type the following command

docker.exe run -it --rm -p 8888:8888 -e JUPYTER_ENABLE_LAB=yes –v $((pwd).tostring()+"/d1-notebooks:/home/jovyan/test") jupyter/tensorflow-notebook

3. Accept the request for access rights from Docker
4. Copy the link beginning by http://127.0.0.1... provided in the powershell and paste it in a web browser.
   (Tested with Firefox 74.0)
5. Open the script provided in the test directory.
6. Then run All cells. It will launch the training of a Convolutional Neural Network (CNN) on a specific dataset (MNIST). After few minutes, you will obtain a CNN trained to recognize specific inputs. We will then use it to deploy the corresponding code onto an embedded platform. Explanations will come during the lab. Just execute the code.

Part 3. Program the CNN on the board with STM32CubeIDE (each time you will deploy a CNN on the board)

1. Run STM32CubeIDE.
2. Open the project you already created for STM32 micro-controler.
3. CubeMX perspective → Category "Connectivity" → "USART2"
4. "Configuration" panel → "Parameter Settings" tab → "Basic Parameters" → "Baud Rate" → "912600"
5. Unnecessary, should be configured by default. Otherwise, category " Additional Software" on the left menu → "STMicroelectronics.X-CUBE-AI"
   - "Configuration" panel → "Platform Settings" tab → "COM Port", "IPs or Components" drop-down "USART: asynchronous", "Found Solutions" drop-down USART2
6. "Configuration" panel → "Add network"
   - "Choose model…" drop-down → "Keras" "Saved model"
7. Select also the following:
   - the type of CNN model we are using: Keras
   - select the file with extension .h5 for the description of the pre-trained CNN in your TF folder (step 1.7)
   - the compression level: None (others to be tested during **Lab 3)**
   - validation inputs: select the file X_test from the directory where you trained your model in step 1.7.
   - validation outputs: select the file Y_test from the same directory
8. Optional:
   - Select Analyze to verify that you get the correct files.
   - Select Validation on Desktop to get the details of the CNN layer per layer
9. Select “Validation on target” to download the code on target, verify the configuration:
   - Use communication port: Automatic
   - Bauds: 912600 = enabled
   - **Tick the box “enabled”** in the part « Automatic compilation and download » without changing the default settings: ST-Link | PA2 | PA3 | STM32CubeIDE | SWD
10. Wait for the injection of the test vectors through the serial connection (10 000 vectors).
    - You will get the result of the inference phase of the CNN at the end of the injection.
    - Warning: the window printing the test vectors is freezing sometimes but is running in background
    - Warning: don’t put your PC on automatic standby during this step
11. If you get the result of the execution on the target, you are ready for **Lab 3**.

Annex 1 – Frequent errors with STM32CubeIDE

- If build fails with "error: static declaration of 'MX_USART2_UART_Init' follows non-static declaration":
  Project Manager → Advanced Settings → MX_USART2_UART_Init → Uncheck

Visibility
(see
https://community.st.com/s/question/0D53W00000WYWEpSAP/xcubeai-520-validation-static-declaration-of-mxusart2uartinit-follows-nonstatic-declaration
)

- If analyze fails with "ImportError: cannot import name 'get_all_providers' from 'onnxruntime.capi._pybind_state'":
  Software Packs → Manage Software Packs → STMicroelectronics → X-CUBE-AI 5.2.0 → Install
  Software Packs → Select Components
  → STMicroelectronics.X-CUBE-AI → Core → Uncheck
  → STMicroelectronics.X-CUBE-AI → Device Application → Not selected
  → Ok
  Software Packs → Select Components
  → STMicroelectronics.X-CUBE-AI → 5.2.0
  → STMicroelectronics.X-CUBE-AI → Core
  → STMicroelectronics.X-CUBE-AI → Device Application → Validation
  (see
  https://community.st.com/s/question/0D53W00000gevWYSAY/xcubeai-600-windows-importerrorcannotimportnamegetallprovidersfromonnxruntimecapipybindstate
)