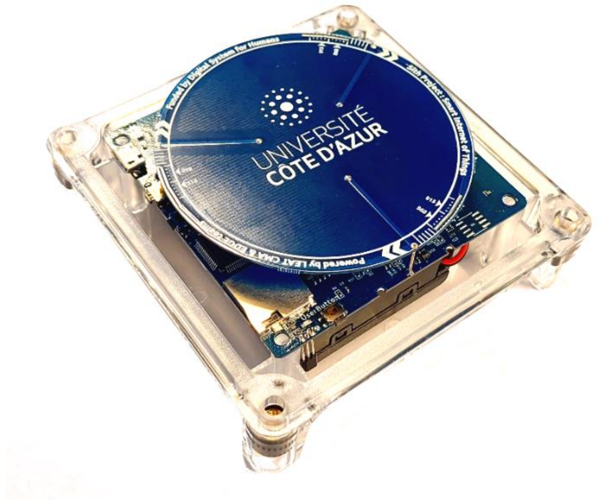
Lab V: preparation of the project for spoken keyword detection (spotting)

Polytech Nice Sophia

Hardware target

The target is a RFThings-AI Dev Kit board equipped with a STM32L476RGT6 Microcontroller. 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.



The goal of this lab is to perform classification of audio samples collected from the microphone mounted to the board.

Part I. Train the network onto an existing dataset

1. Read the script provided in the repository (lab5). In this script the wave data are read file by file. Each file corresponds to one sample of the Google Speech Command (GSC_v2) Dataset, which specifically consider the pronounced 10 first digits. Each file contains the audio data following a PCM coding, 16khz format with 16 bits signed integer precision on **a single channel**. The script first converts the data in float 32 bits representation. Since all the data do not have the same length, the second step consists in resizing all the samples in the same format (1 second) thanks to zero padding. The rest of the script fill the arrays corresponding to the train and test sets.
2. Adapt the previous notebooks to build and train a network dedicated to this new dataset.
    a. Build a network corresponding to the model M5 proposed in the following publication: https://arxiv.org/pdf/1610.00087.pdf
    b. Train the network to reach at least 90% on the test set
    c. Generate the code, compile and measure the resulting memory footprint
    d. Does it fit the embedded memory of the targeted device?

Part II. Optimize the network to fit the embedded constraints

Modify the previous M5 model in order to reduce the memory footprint: resize the input, reduce the filters and the 1D kernels. You can also increase the stride.
As usual, try to fit the memory constraint and conserve a good accuracy. Verify that the 16 bits quantization does not impact the initial accuracy.

Part III. Read the data from the microphone onto the MCU and make real-time prediction

a. Read the data from the microphone

The microphone used on the board captures analog signals but embeds an ADC. It is connected to the microcontroller through I2S serial communication: Inter-IC sound or IIS. It is a standard for the serial communication with external devices. It is composed at least of 3 signals: the clock signal bit, the signal word select line and the multiplexed data.
The data are automatically stored in a specific buffer by the DMA of the MCU.

- Look on internet to provide a definition of what is a DMA in a MCU architecture (report).

b. Generate the code and make prediction

As usual, compile your Arduino file with the description of your pretrained model and download it onto the board.
You can take a look at the loop function to understand the behavior: the callback function will fill the input buffer and the loop function will make a new prediction each second thanks to the previous boolean variable *ready_for_inference*.
You can adapt the gain coefficient for the amplification of the signal (×4 by default in the Arduino code) in your tests.

Verify the good prediction of your model onto the real data

- by pronouncing several times the 10 digits from 0 to 9 in English in the case of the digits from GSC

In the future context of your project this step will be repeated on specific data

- by pronouncing the words of your dataset
- or by playing the sound of recorded bird …
- this step depends on the application you target in your project

You can send the samples over the serial link to your laptop to reconstruct a PCM file and listen to your samples with the Audacity software:

- Uncomment the corresponding line
- Comment the line that prints the prediction into the serial console.
- Run the python script from the MicroAI_EDU repo: serial_client_i2s_pcm.py
- Import raw data of the “out.pcm” file with the correct parameters

MicroAI_EDU : https://bitbucket.org/edge-team-leat/microai_edu/src/master/

Part IV. Send the data with LoRAWan wireless communication and store it onto the server

The last step of the project consists on sending the predicted classifications onto a data server through wireless communication.
To do that, you will add to your embedded code the part communicating over the LoRAWan protocol. You can find the template of the code on the MicroAI_EDU repository (Lab6).
In order to get the data collected by the device, you will create an account on The Things Network (TTN) : https://www.thethingsnetwork.org/
Your account will enable to add the keys of your device and observe the data collected.
Verify the relevance of your predictions before deploying your device in its environment.

V. Report

The project will be evaluated through a report you will submit on Moodle at the end of the project time with the corresponding source code.
The report must contain the following sections:

- Description of your project (context, goals, environment of test)
- Description of the workflow to solve the problem described before (from training to embedded AI and communication). Add a figure illustrating that flow.
- Description of the CNN model obtained after training by taking inspiration on the first labs
- Presentation of the processing architecture of the sensor's data
    - Provide a definition of DMA and ADC
- Presentation of the results obtained, and the experimental setup to test the generalization of the training to real data (not the ones of the dataset).
- Analysis of the result of your project on the following criteria
    - Performance (accuracy on train, test and on real data after quantization)
    - Memory footprint
    - Latency
    - Energy consumption and battery lifetime analysis (provide assumptions of your estimation)
- Conclusion and possible evolutions of the project