Lab III: Automatic generation of embedded AI with MicroAI

Polytech Nice Sophia

Part I Automatic generation of embedded neural networks

Part I.1 Train your convolutional network

Work to be done: Reuse a pretrained convolutional network on UCI HAR (from Lab1)

You can now reuse a previous CNN that attain good accuracy (>90%) on this dataset.

Part I.2 Generate a complete convolutional network

Work to be done: Use the provided MicroAI software to automatically generate the C code of your network

Download from MicroAI_EDU repository the notebook Lab3_part1_*UCA-HAR_MicroAI.ypynb* and **adapt the code to your CNN**.

Run the notebook to i) remove the softmax layer (executed on your computer, not on the MCU), ii) install MicroAI and iii) call MicroAI to convert the code in C.

Note the parameters of the Converter constructor. They correspond to the quantization explained in the previous lab, but they can also be adapted to work on 8 bits during inference.

At this step, you should find in your working folder a subfolder 'output' containing one file per layer and a header file (*model.h*) containing all the code gathered in a single file (easier for compilation with ArduinoIDE).

Part I.3 Evaluate your network on desktop

Work to be done: Validate your network on your desktop with the data of the UCI HAR test set.

After launching the container:

```
docker run -it --rm --name jupyter-tensorflow -p 8888:8888 -v
"$(pwd)"/MicroAI_EDU:/home/jovyan/MicroAI_EDU jupyter/tensorflow-notebook
```

you must install the package related to gcc:

```
docker exec -it -u root jupyter-tensorflow apt update
docker exec -it -u root jupyter-tensorflow apt install -y g++
```

You can then use the command from the jupyter notebook to compile and test the two versions of your network: full precision (floating point) and quantized (fixed point).

Part I.4 Evaluate onboard your network on the test set

Work to be done: Validate your network on the board with the data of the UCI HAR test set

Replace the previous *model.h* file (lab 2) by the new one and download the code on the board and run the python code evaluate.py on your computer in order to validate it on the MCU:

Example on windows: *> python evaluate.py x_test_250.csv y_test_250.csv COM4*

Make sure you find a similar accuracy between the onboard prediction and the validation of your original keras model.

From the results obtained on desktop and onboard, fill the following table and provide a comparison between the different accuracies.

Part II Measurements of embed AI on MCU

Part II.1 Onboard mearurements

In this first step of measurements, you will evaluate the characteristics of the onboard prediction and fill the following table.
The code provided for the lab (evaluate.py) measures both the accuracy and the latency of the prediction (Latency per inference).
At compile time, Arduino IDE provides the Sketch use of the memory (ROM).
To get the RAM footprint, it is needed to go deeper in the organization of the generated binary file. This file is in ELF format and you can observe the sections of this file by running the size command (arm-none-eabi-size). To get the command line corresponding to your binary file, configure ArduinoIDE in verbose mode (File menu->preference->show verbose output during compilation) and copy the last command from the Arduino terminal to a powershell terminal.
You can estimate your RAM footprint by summing the size of the relevant sections.

	Floating point representation	**Fixed point representation**
Accuracy (%) on desktop	83,20%	83,20%
Accuracy (%) on board	83,20%	82,8%
Latency **L** (ms) on board	34,20ms	33,11ms
Memory footprint (ROM, KB)	62152 bytes (25%)	57904 bytes (23%)
Memory footprint (RAM, KB)	55376 bytes	50032 bytes

Warning:
*Verify that your conversion code in the Arduino file works both in floating and fixed point representations of the input data (especially by using generic type **number_t** and not **int**).*
Comment on the estimation:
With that simple method we don't have an exact measure of the size of the stack that is dynamically allocated during execution. But the code does not use intensively local variables and we assume that this estimation is close to the real memory use.

Part II.2 Estimation of embedded AI and device autonomy

From the previous measurements, you will now provide an estimation of the autonomy of the device when executing the CNN and making prediction on the sensor's data.
In order to make this estimation, we will make the following assumption:

- The microcontroller is supposed to behave in two power modes:
  - Active mode: The power consumption **P** of the board has been already measured. It corresponds to the instantaneous consumption of the board when running the code (read data and make prediction) and is 62 mW.
  - Low-power mode: it corresponds to the power consumption when the board only reads the samples (in polling mode). In that case, the power is supposed to be zero.
- Let's consider that the device will collect data over a window of 2,56 sec. The period **T** of the prediction task corresponds thus to this value.
  - In that time, the duration of the Active mode corresponds to the measured latency **L**
  - The duration of the low-power mode corresponds to the remaining time over the period **T**
- You now have to compute the average power consumption on the entire period **T**
- The battery pack used with our device is composed of **two/three** AA Alcaline batteries. Each battery has an energy of 3900 mWh.
- From the previous data, you should estimate the autonomy (in time) of the device.

	Unit of measure	**Estimation**
Power consumption of active mode	mW	62mW
Average power consumption per inference (over period T)	mW	0,799
Energy of the battery	mWh	7800/11700
Autonomy of the device with sleep mode	h	9762,2
Autonomy of the device without sleep mode (the case in our labs)	h	125,81
Autonomy of the device in case of wireless communication (propose a model for LoRA communication)	h	Average = 2,30mW 3391

(62*0,033 +0*(2,56-0,033))/ 2,56

7800/0,799

7800/62

(62*0,033 +0,05*(62 +15))/2,56

7800/2,30

On suppose consommation de
Lora 15mW pendant 50ms

In the next lab, you will reproduce the steps of part I onto your own data collected from the accelerometer of the board and make the prediction in real-time.