LAB7 APAI:

Deployment of Neural Networks on STM32

***Credits**: Alessio Burrello, Lorenzo Lamberti, Francesco Conti, Nazareno Bruschi, Davide Nadalini, Alberto Dequino. (University of Bologna)*

***Contacts**: alessio.burrello@unibo.it, nazareno.bruschi@unibo.it, f.conti@unibo.it*

How to deliver the assignment:

DEADLINE: 16/12/2021 at 4PM

Instructions:

- Use Virtuale platform to load your file: link
- update the .ipynb file, named as follows: LAB7_APAI_yourname.ipynb and the C project.

Important: the notebook must be pre-run by you. Outputs must be correct and visible when you download it

Links to COLAB exercise:

COLAB:
https://colab.research.google.com/drive/1UCozUW26KFQnB4ed1Dbchqj54I-Ymdll?usp=sharing
Solution: (...coming in 1 week...)

Requirements:

- STM32CubeMX Version 6.4
  - STM32Cube FW_<YOUR_FAMILY_CORE> V<XX>
  - STMicroelectronics X-CUBE-AI 7.0.0
- STM32CubeIDE
- PuTTY

Resources

- https://www.st.com/resource/en/user_manual/dm00570145-getting-started-with-xcubeai-expansion-package-for-artificial-intelligence-ai-stmicroelectronics.pdf
- https://www.digikey.it/en/maker/projects/tinyml-getting-started-with-stm32-x-cube-ai/f94e1c8bfc1e4b6291d0f672d780d2c0

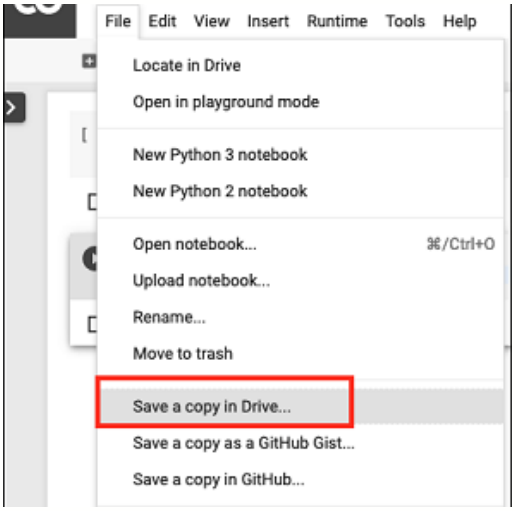
- http://www.emcu.eu/how-to-implement-printf-for-send-message-via-usb-on-stm32-nucleo-boards-using-atollic/
- Materials on GitHub

Lab session summary:

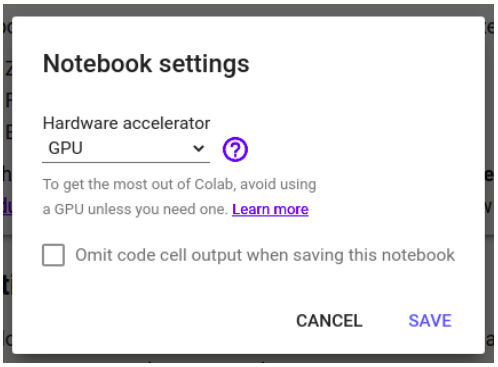
1. Train the network from LAB5 with Softmax instead of LogSoftmax;
2. Export the .onnx of the network and test it on CubeMX with random values;
3. Export a batch of data from the notebook and validate your application using real data;
4. Export a single image from the notebook and create the final application;

COLAB Setup:

1. Create your own COLAB copy of the notebook!
    - In Google Colab, use the menu: *File > Save a copy in Drive*



2. Activate/deactivate GPU: *Runtime -> Change runtime type*
    - ***Note**: If you use for too much time the GPU, your account will be limited to CPU for 24h.*



Task 0: Install the STM and other useful tools

The installation of the tools may require a lot of time, hence, start with it.

- Download from Virtuale the installers.
- Install CubeMX
- After the installation of CubeMX, open it and download the board firmware and XCubeAI library. You could find some references in the lab materials
- Install CubeIDE
- Install PuTTY

Task 1: Train the network from LAB5 with Softmax instead of LogSoftmax

```
class CNN(nn.Module):
    def __init__(self, n_classes=10, depth_mult=1.):
        super(CNN, self).__init__()
        # first_conv_channels=int(32*depth_mult)
        self.ConvBnRelu1 = ConvBnRelu(1, 32,  stride=1) # conv3x3: ch_in=1, ch_out=32, in=28x28, out=28x28/2
        self.ConvBnRelu2 = ConvBnRelu(32, 64,  stride=1) # conv3x3: ch_in=1, ch_out=32, in=28x28, out=28x28/2
        self.pool = nn.MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, return_indices=False, ceil_mode=False) # MaxPool2: in=14x14, out=14x14/2
        self.ConvBnRelu3 = ConvBnRelu(64, 128, stride=2) # conv3x3: ch_in=32, ch_out=64, in=7x7, out=7x7
        self.dropout = nn.Dropout(p=0.5, inplace=False)
        fc_in_size = 7*7*128
        self.fc = nn.Linear(fc_in_size, n_classes, bias=False)
        self.softmax = nn.LogSoftmax(dim=1)
```

Figure 1: The Custom CNN topology.

Task 2: Export the .onnx of the network and test it on CubeMX with random values

Description: this task is splitted into two different steps.

- in the colab, you have to export and download the graph.onnx with the topology and weights of your network;
- in the stm32cubemx you have to import your network in a project with the board assigned to your group; you can follow the detailed steps in the guide linked at the beginning of this file.
    a. Create a new project and select the your board in the board selector (note that all the peripherals of the board are automatically set to default values);
    b. Include the X-CUBE-AI package and add your network to the project;
    c. Go through the Analyze, Validate on Desktop, and Validate on Target with random number steps.

Output: which are the dimensions computed by the tool? Are they identical to the ones computed from pytorch?

BONUS: Are the CNN dimensions compatible with the memory storage of your board? If no, restart with a smaller CNN, otherwise you will have linker error during the validation on target. You can answer this question after analyzing the network with the CubeMX option. TIP: Reduce the input and output channels of the CNN conv layers according to the board memory constraints. Do not take into account accuracy drop.

Task 3: Export a batch of data from the notebook and validate your application using real data

Description:
On the Colab:

- take the first batch of validation images and the corresponding labels;
- export them in .npy format with numpy and download them

On STM32CubeMX:

- repeat the steps validate on desktop and validate on target with the new data.

Output: which is the accuracy obtained by the tool on a single batch?

Task 4: Export a single image from the notebook and create the final application

Description: Create a working application with a network that classifies an image and gives back the result through UART.
This task includes many sub-tasks:

1. Download a single image of the validation set in csv format. We will use this image inside our project. Use the csv writer to create the csv file.
2. Create the template of the C code using the generate. You have to select the Application Template inside the Software Packs -> Component Selector -> X-CUBE-AI -> Device Application. In the project manager you can define directories and the IDE (select STM32CubeIDE). PAY ATTENTION: Every time you generate the C code, the previous will be overwritten if you have not used the specific code spaces to add your code!
3. Add all the missing elements to classify the input image with the new application:
    a. Add the call to the init of the uart peripheric (USART3) to read the outputs on the uart (if is needed);
    b. Re-direct the printf to the uart to use it on the board. TIP: add the __io_putchar and the _write functions at the end of your code in the main and add #include <stdio.h> in the includes.

c. Add the file input.h where you save the data from your image in the .csv file. float input[28*28] = {CSV DATA}; and add it to the includes of the app_x-cube-ai.c
d. Updated the ai_run to put in the ai_input[0].data the image.
e. Change MX_X_CUBE_AI_Process() , acquire_and_process_data(), post_process() accordingly to let the network work.
f. Printf the output results in the data of ai_output to see the final score of each class; TIP: convert the output to float and print float numbers. Additional TIP: You need to abilitate the float print in the linker flags (-u print_float) → the IDE will suggest it.

4. Run your application using STM32CubeIDE:
    a. Open a SERIAL link with the correct COM from PuTTY with the correct baudrate. TIP: check in device manager the COM port and in UART initialization in your c code the baudrate value.
    b. Run the application from the IDE and read if the correct output is displayed on the UART.

Output: A working C project to classify images through our CNN deployed on the STM32 board