IESTI01 – TinyML

Lab 04

Adriano Carvalho Maretti

2020009562

Introduction

This following report will be about the fourth and last laboratory activity of the IESTI01 discipline. In this lab activity, the Person Detection Model should be deployed in the Arduino Nano 33 BLE board and modify the code a little bit to add some new features to the application, changing the LEDs dynamics a little bit, with the sentences that are printed on the Serial Monitor

The hardware used consisted in the Arduino Nano 33 BLE, connected to the computer with a micro-USB cable.

The software used to deploy the model was the Arduino IDE, where the code was developed and all the modifications and added features were built and incorporated to the project. The AI model was already trained and responding greatly, prior to the start of this project.

Image 1 – Arduino running the model

Data, training and testing

The data acquisition and treatment, the ML model training and testing were all done prior to the lab by the Harvard TinyML team. So, all the props to them for setting up this incredible set of data and building a great and applicable model.

First Deployment

The first deployment was made without modifying anything from the original code. The code was opened from the examples tab and flashed inside the MCU to run. In this first deployment, the intention was to understand how the ML model was running and what was the programming logic behind it, by analyzing the .cpp and .h files to understand its contents and working functionalities.

So, it runs capturing images from the camera each second (when capturing the blue LED turns on). Then, the LED turns green if the person is spotted and red in case it does not capture someone. In the Serial Monitor, the results of the inference are continually printed in the form of probability for each label, one of them being negative and the other positive (the model real guess).

Code Modifications

After changing the code the first time, I tried to turn on a yellow LED by combining the Green and Red colors of the RGB LED, so that it would turn to the color I wanted. But, unfortunately, it didn't work as planned and both LEDs produced an undesirable combination of colors that made the project look bugged.

Then, I tried to work with de power indicator LED blinking, just to test how this would work, and the application looked bugged again because it seemed like the board was turning on and off continually.

So, I opted to do the modifications proposed as they are in the class, and it worked all just fine. The built in LED indicated the camera taking the user's pictures, and the LED turns blue when a person is detected and red when it doesn't, printing exactly this output on the Serial Monitor while the model is running at its normal pacing. The code bellow is the original code modified to do the tasks described above. All the added code is present in the RespondToDetection function.

```
void RespondToDetection(tflite::ErrorReporter* error_reporter,
                        int8_t person_score, int8_t no_person_score) {
```

```
  static bool is_initialized = false;
  if (!is_initialized) {
    // Pins for the built-in RGB LEDs on the Arduino Nano 33 BLE Sense
    pinMode(LEDR, OUTPUT);
    pinMode(LEDG, OUTPUT);
    pinMode(LEDB, OUTPUT);
    pinMode(LED_BUILTIN, OUTPUT);
    is_initialized = true;
  }

  // Note: The RGB LEDs on the Arduino Nano 33 BLE
  // Sense are on when the pin is LOW, off when HIGH.

  // Switch the person/not person LEDs off
  digitalWrite(LEDB, HIGH);
  digitalWrite(LEDR, HIGH);
  digitalWrite(LEDG, HIGH);
  digitalWrite(LED_BUILTIN, HIGH);

  // Flash the blue LED after every inference.
  digitalWrite(LED_BUILTIN, LOW);
  delay(100);
  digitalWrite(LED_BUILTIN, HIGH);

  // Switch on the green LED when a person is detected,
  // the red when no person is detected
  if (person_score > no_person_score) {
    digitalWrite(LEDB, LOW);
    digitalWrite(LEDR, HIGH);
    TF_LITE_REPORT_ERROR(error_reporter, "Person detected",
                         person_score, no_person_score);
  } else {
    digitalWrite(LEDB, HIGH);
    digitalWrite(LEDR, LOW);
    TF_LITE_REPORT_ERROR(error_reporter, "No person detected",
                         person_score, no_person_score);
  }


}
```

Deployment and running

When trying to deploy the model, some errors were faced between executions. The microcontroller seemed to not accept to compile the model and try to flash it in without pressing the RESET button. So, after this was discovered,

I could run the three modifications I described briefly in the Code Modifications section of this same document, the last modification being the one shown in the images bellow.

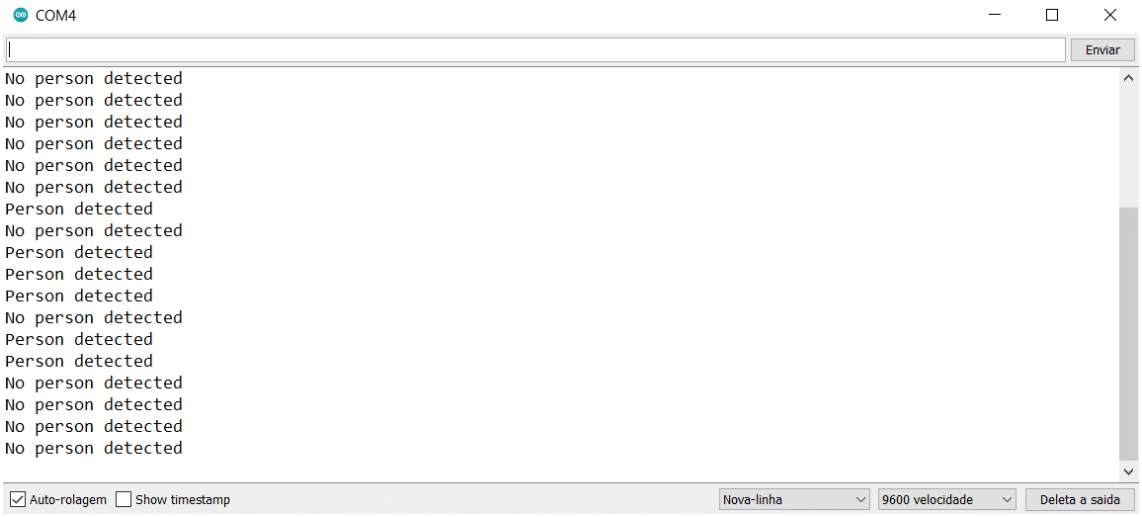

Image 2 – Serial Monitor printing the outputs

Image 3 – Model detected someone

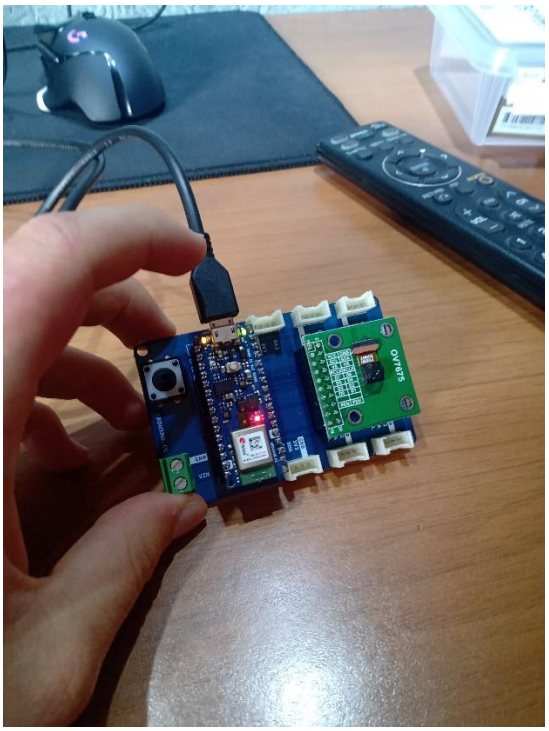

Image 4 – Model not detecting anyone

Conclusion

The, to conclude this lab, it was noticed that great image processing models can be done in embedded devices as well, even with a not great camera and really low processing, storage and memory powers. By modifying the C++ code present in the already built Harvard project, I was able to understand more of the Arduino IDE and the model as a whole. Being able to modify something gives you the task of understanding how that code is working and what it is doing in each line, so you can add your own code without messing up what is already working

In general, this was a great learning process, and I thank the teacher a lot for postponing the submission of this lab so I could do it calmly and learning as I must. Thank you very much once again.