IESTI01 – TinyML

Lab 02

Adriano Carvalho Maretti

2020009562

Introduction

This lab exercise consisted in building a TinyML model on Edge Impulse Studio that could identify, from a microphone, if the user was speaking one of the keywords that were defined before, and would turn a RGB led on if the condition was achieved.

The hardware used to run the model was an Arduino Nano 33 BLE MCU, like the one shown in the image below

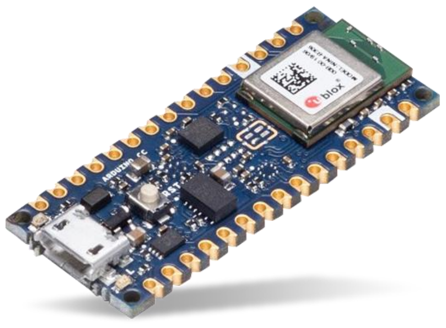
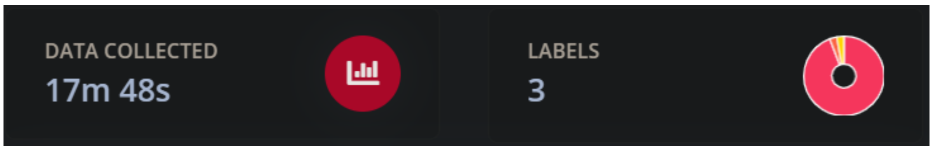


Image 1 – Arduino Nano 33 BLE

The software used to build the model was composed of the Edge Impulse Studio (the website used to build and train the model), Edge Impulse CLI (the cli used to pair the website with the MCU), the Arduino CLI, the Edge Impulse firmware (loaded inside the board), the Arduino IDE (the development environment used to load the ML model inside the board and code what would happen if a determined label would be inferenced by the running model.

Data Acquisition

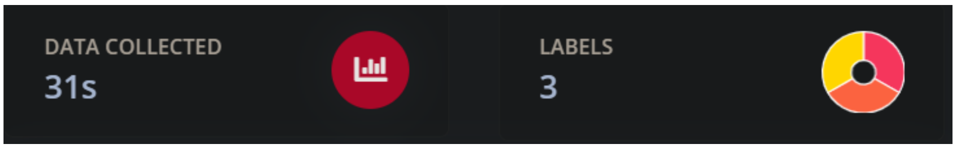
The data acquisition was made based that the model would inference between 3 different labels: ambient_noise, Unifei and Jarvis. The ambient noise data collection was made out of an already existing database from Edge Impulse, containing 1006 different sounds. The Unifei and Jarvis labels were collected manually, being 30 one second samples each.



DATA COLLECTED
17m 48s
LABELS
3

Image 2 – Training dataset

The test dataset was made out of 30 different samples, being 10 of each label the model would support, all collected manually.

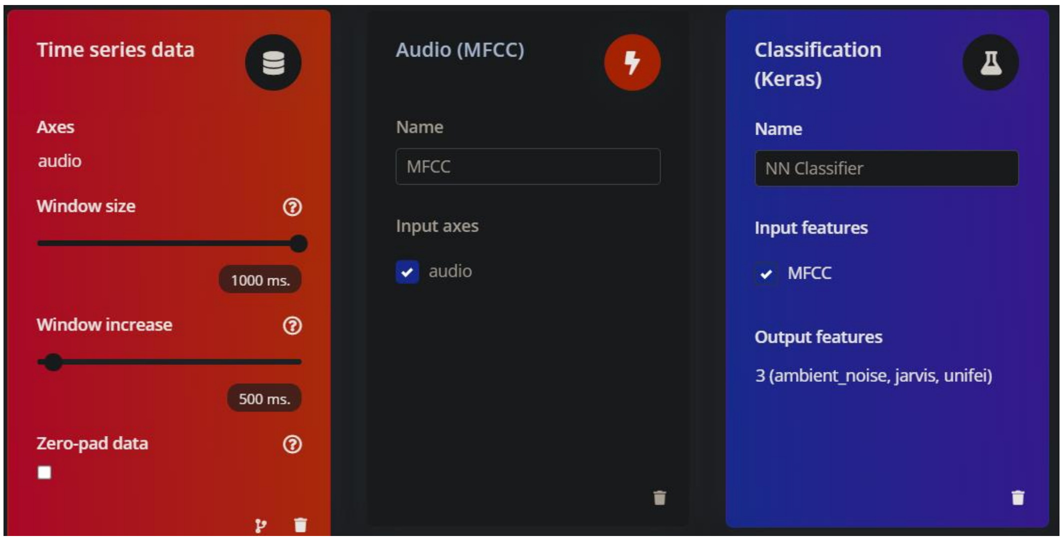


DATA COLLECTED
31s
LABELS
3

Image 3 – Test dataset

Impulse Design

The impulse was designed to resolve this specific problem, being composed of a Time Series Data Analysis, a Audio MFCC layer and a Keras classification.



Time series data
Axes
audio
Window size
1000 ms.
Window increase
500 ms.
Zero-pad data

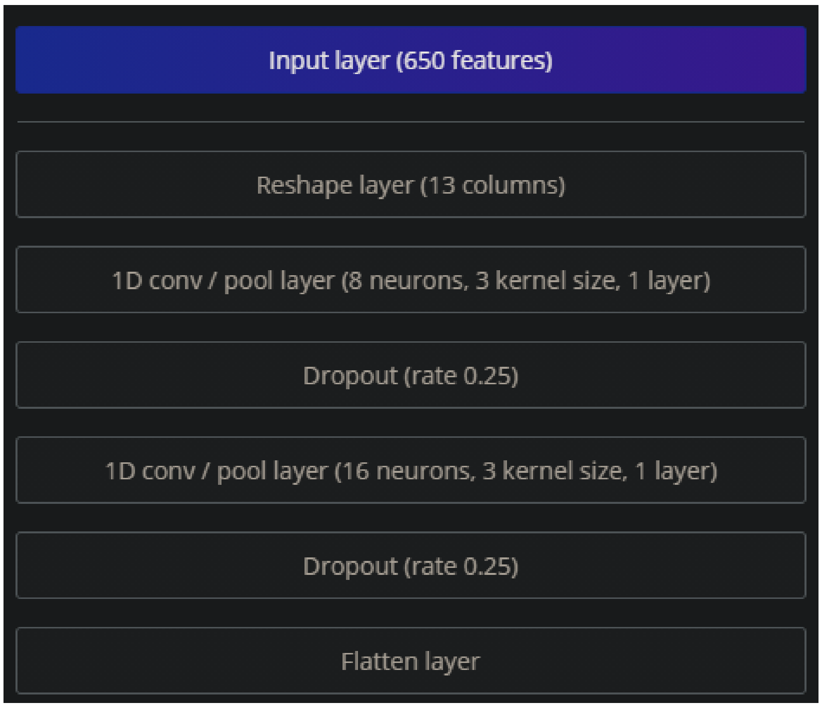
Audio (MFCC)
Name
MFCC
Input axes
✓ audio

Classification (Keras)
Name
NN Classifier
Input features
✓ MFCC
Output features
3 (ambient_noise, jarvis, unifei)

Image 4 – Impulse Design

Training the model

After the data was collected and the Impulse was properly built, the next step was to use the data to train a model to identify what sound was coming in the microphone. The model was trained in the standard number and type of layers Edge Impulse Studio passed.



Input layer (650 features)

Reshape layer (13 columns)

1D conv / pool layer (8 neurons, 3 kernel size, 1 layer)

Dropout (rate 0.25)

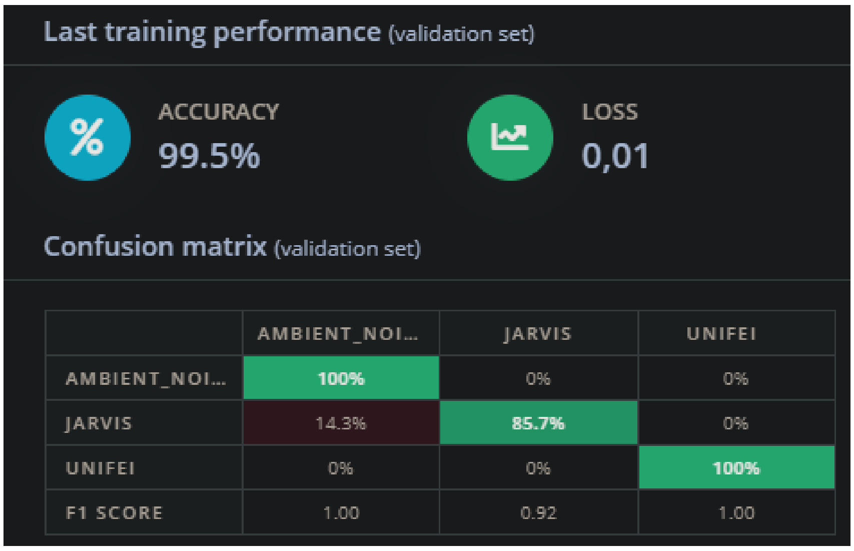
1D conv / pool layer (16 neurons, 3 kernel size, 1 layer)

Dropout (rate 0.25)

Flatten layer

Image 5 – NN classifier

So, after training was completed, this was how the results looked like from the training data perspective, and the result was really great in terms of the accuracy and loss that were obtained: 99.5% and 0,01 respectively.



Last training performance (validation set)

ACCURACY 99.5%

LOSS 0,01

Confusion matrix (validation set)

	AMBIENT_NOI...	JARVIS	UNIFEI
AMBIENT_NOI...	100%	0%	0%
JARVIS	14.3%	85.7%	0%
UNIFEI	0%	0%	100%
F1 SCORE	1.00	0.92	1.00

Image 6 – Training results confusion matrix

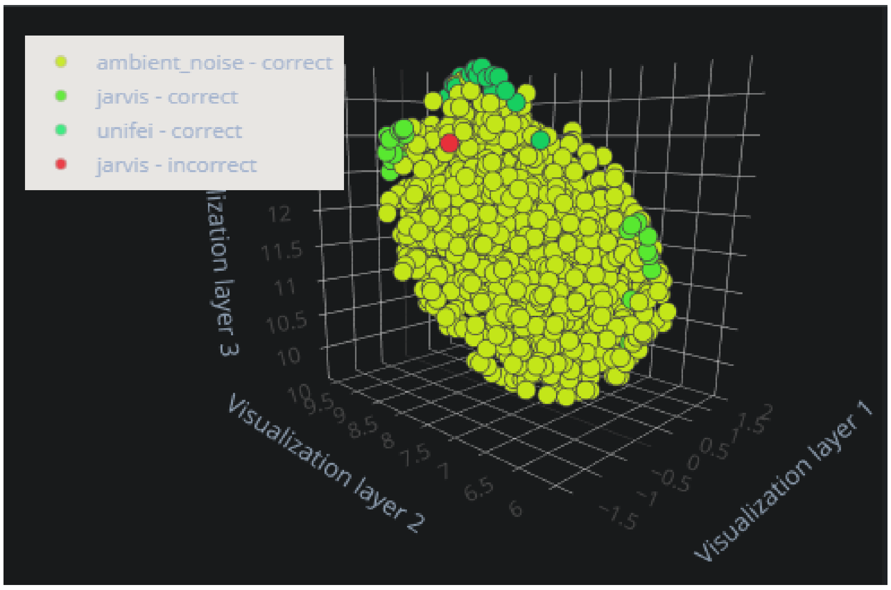
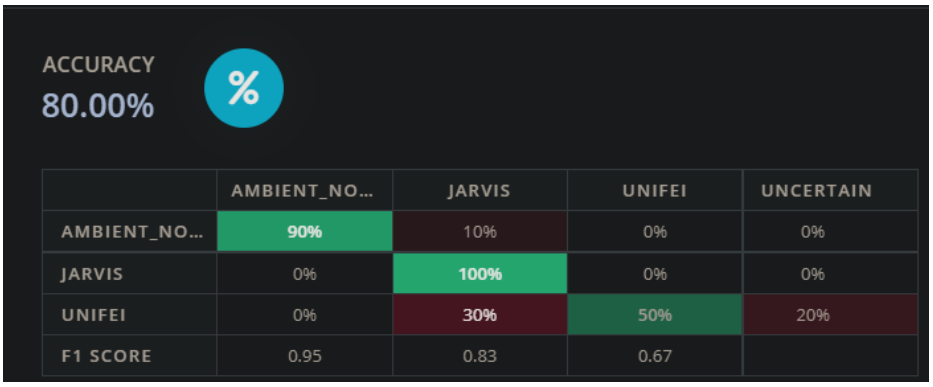


Image 7 – Training dataset Feature Explorer

Testing Model (Edge Impulse Studio)

After the model was properly trained and obtained a great validation dataset accuracy (99.5%), it's time to test the model with real world sounds that weren't present in the original dataset, but later collected and inserted in the Edge Impulse Studio. The model achieved a good 80% accuracy when being tested, with some confusion from the model when the label was "unifei".



ACCURACY
80.00%

	AMBIENT_NO...	JARVIS	UNIFEI	UNCERTAIN
AMBIENT_NO...	90%	10%	0%	0%
JARVIS	0%	100%	0%	0%
UNIFEI	0%	30%	50%	20%
F1 SCORE	0.95	0.83	0.67	

Image 8 – Test data confusion matrix

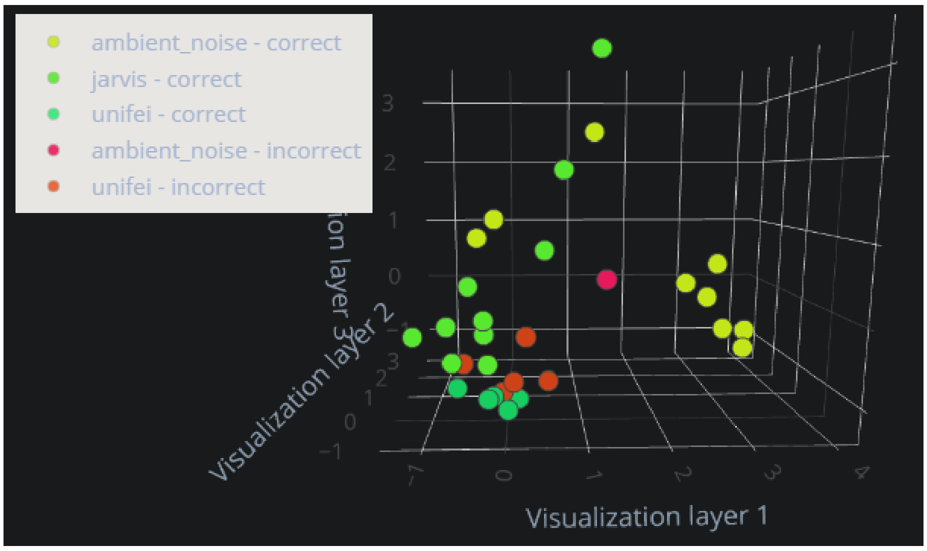
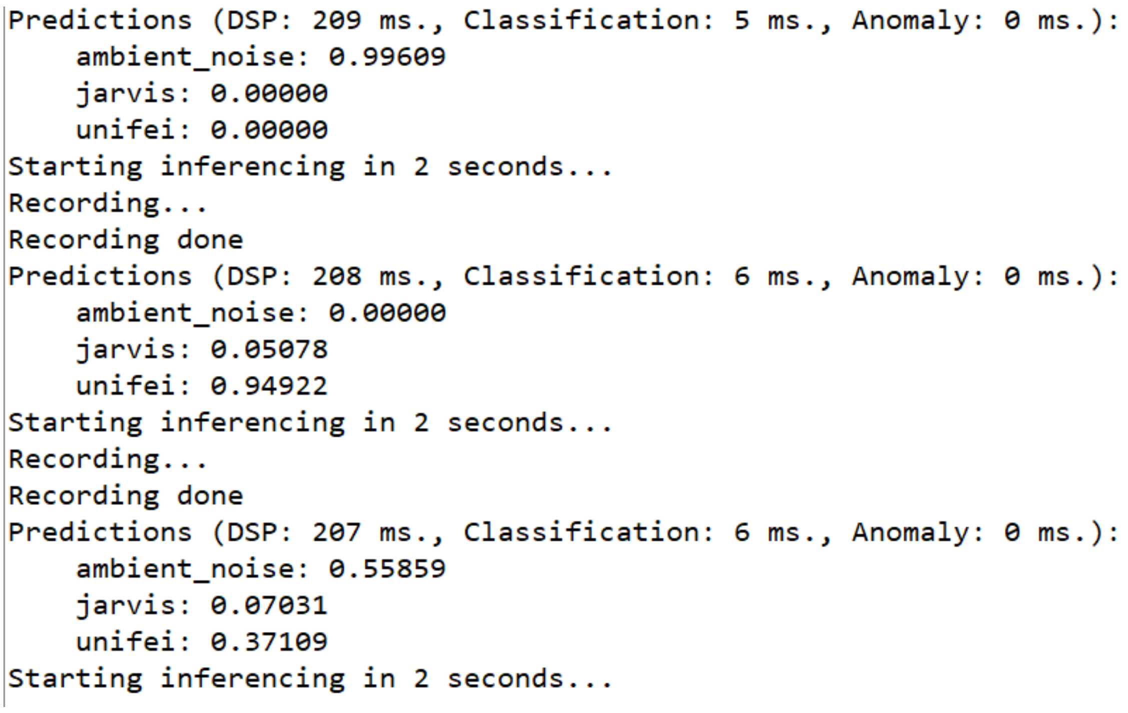


Image 9 – Test data Feature Explorer

Deployment

After all those steps were taken, the model was compiled into an Arduino Library and sent to a .zip file. Then, it was uploaded to the Arduino IDE and uploaded inside the Arduino Nano 33 BLE and the model inferenced some labels after recording sometimes:



```
Predictions (DSP: 209 ms., Classification: 5 ms., Anomaly: 0 ms.):
    ambient_noise: 0.99609
    jarvis: 0.00000
    unifei: 0.00000
Starting inferencing in 2 seconds...
Recording...
Recording done
Predictions (DSP: 208 ms., Classification: 6 ms., Anomaly: 0 ms.):
    ambient_noise: 0.00000
    jarvis: 0.05078
    unifei: 0.94922
Starting inferencing in 2 seconds...
Recording...
Recording done
Predictions (DSP: 207 ms., Classification: 6 ms., Anomaly: 0 ms.):
    ambient_noise: 0.55859
    jarvis: 0.07031
    unifei: 0.37109
Starting inferencing in 2 seconds...
```

Image 10 – Model inferences on the serial monitor

Then, the standard code was modified to include a function that triggered the RGB led depending on the label returned. The added code was based on this two functions:

```
void turnLedsOff (void) {
  digitalWrite(LEDR, HIGH);
  digitalWrite(LEDG, HIGH);
  digitalWrite(LEDB, HIGH);
}
void turnLedOn(unsigned int prediction) {
  switch(prediction) {
      case 0:
        turnLedsOff();
        digitalWrite(LEDR, LOW);
        break;
      case 1:
        turnLedsOff();
        digitalWrite(LEDG, LOW);
        break;
      case 2:
        turnLedsOff();
        digitalWrite(LEDB, LOW);
        break;
    }
}
```

So, the functionality working can be seen in this link in a recorded YouTube video.

Conclusion

After training, testing and deployment on the board, it was concluded that the model is working pretty satisfactory. It had an 80% accuracy when tested on Edge Impulse and was working pretty well when running inside the Arduino Nano. To increase the accuracy for the test set, it could be added more samples of other people speaking the keyword, more samples of different distances between the mic and the person and so on.

But, anyway, the obtained result was a good model with an efficient keyword detection system, that could be used to a great deal of things.

Ps.: If you want to check out the code used, just access my Github account.