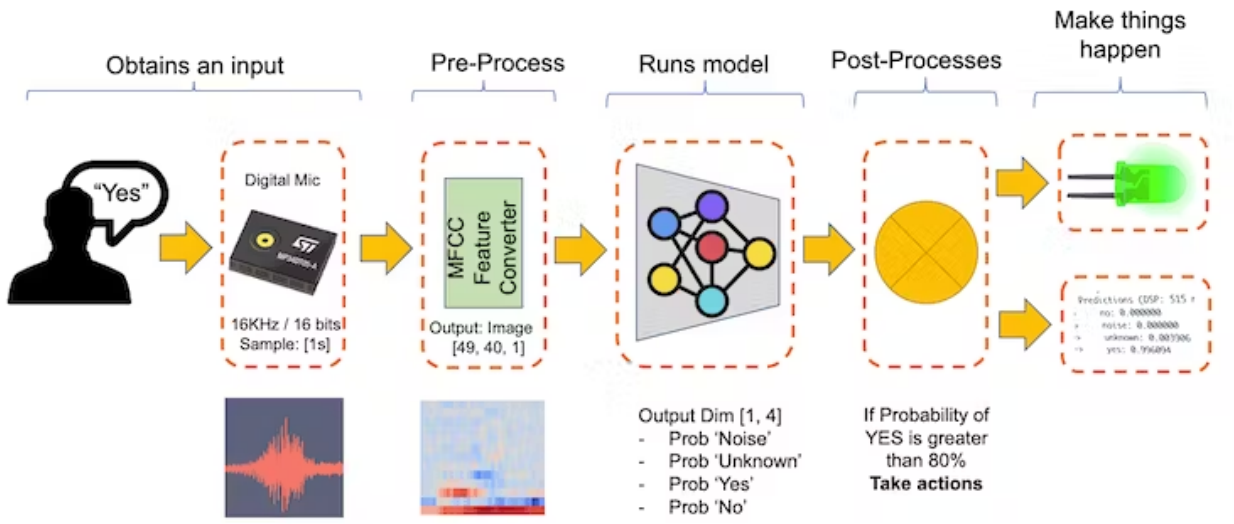
EECS 690/700 EmbeddedML Lab #4

KWS

In this lab, you will create a KWS project on Edge Impulse and deploy the trained model onto the XIAO ESP32S3 board.

Credit: this lab is based on the following material developed by Prof. Marcelo Rovi
https://www.hackster.io/mjrobot/tinyml-made-easy-keyword-spotting-kws-5fa6e7

The KWS Project Overview



Our KWS application will recognize four classes of sound:

- YES (Keyword 1)
- NO (Keyword 2)
- NOISE (no keywords spoken, only background noise is present)
- UNKNOW (a mix of different words than YES and NO)

Dataset

We will be using the speech command dataset by Pete Warden. The dataset has >1000 samples of data for each of 35 keywords it supports.

- Download the keywords dataset.
- Unzip the file in a location of your choice.

Note that all audio recordings are 16-bit samples recorded at 16KHz sampling speed. You can (in fact are encouraged) to add additional samples of your own voice.

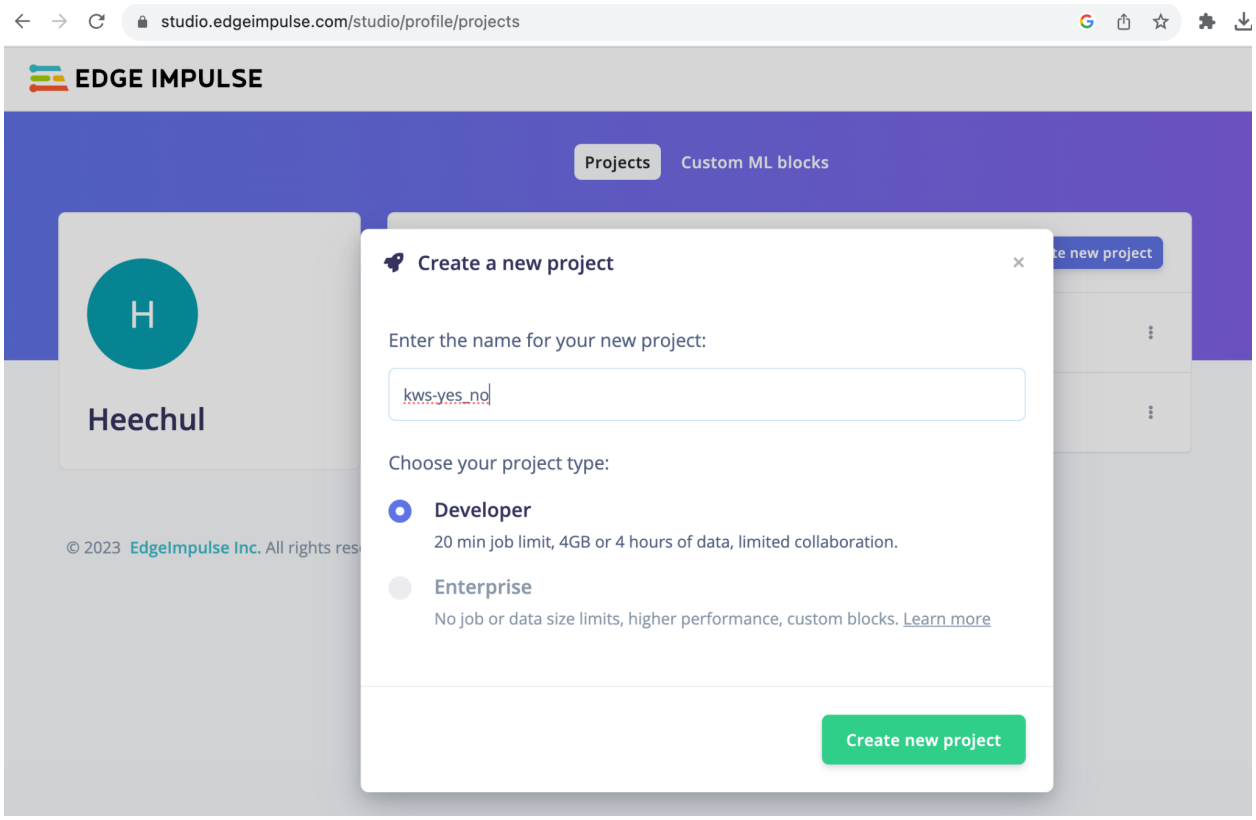
Part 1. Training KWS model with Edge Impulse

We will be using Edge Impulse for KWS application.

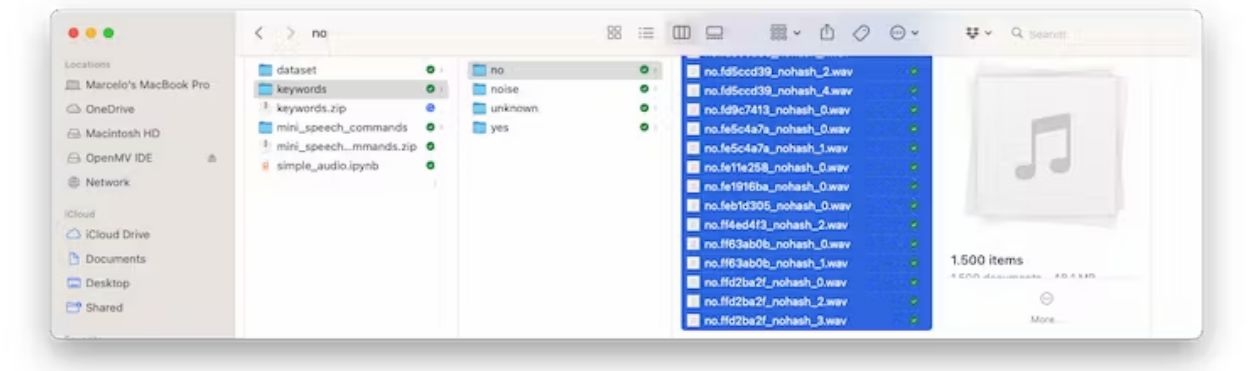
Task 1.1. Creating a new Edge Impulse project

Go to https://edgeimpulse.com and create an account.

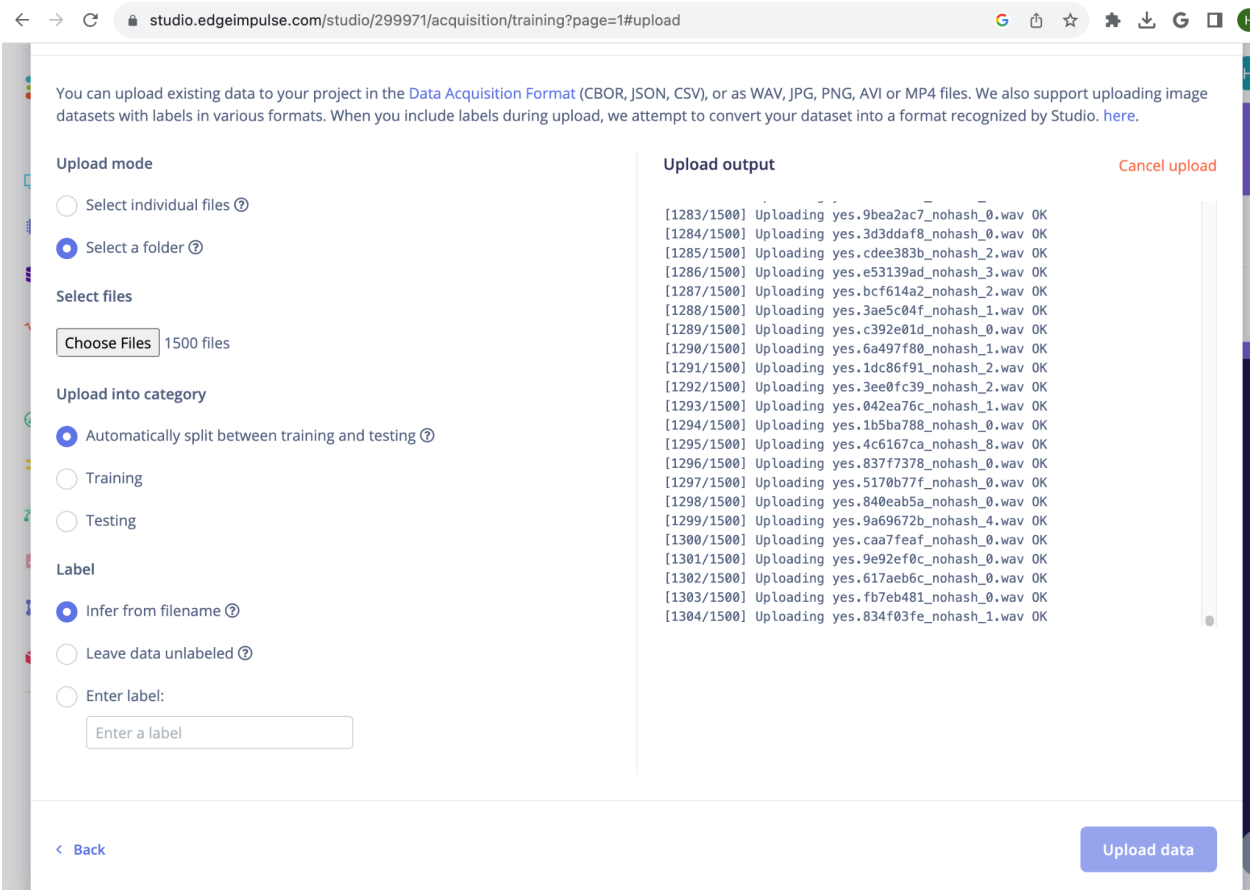
Once you create an account, create a KWS project (any name is fine).



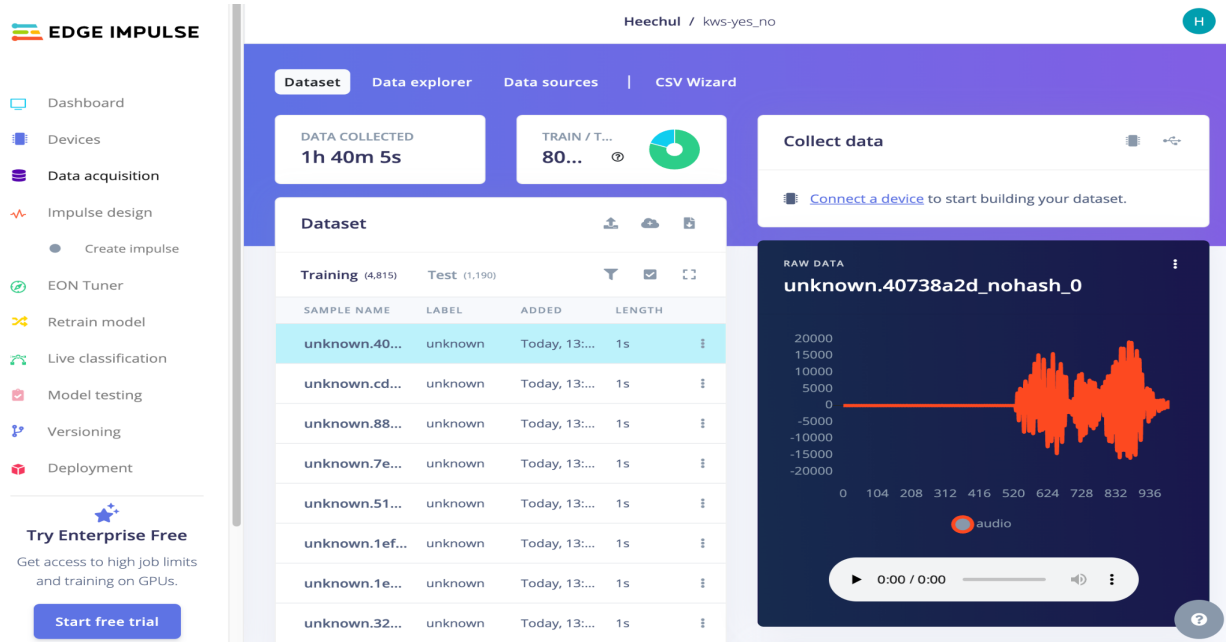
Once the project is created, select the Upload Existing Data tool in the Data Acquisition section. Choose the files to be uploaded:



And upload them to the Studio (You can automatically split data in train/test). Repeat to all classes and all raw data.



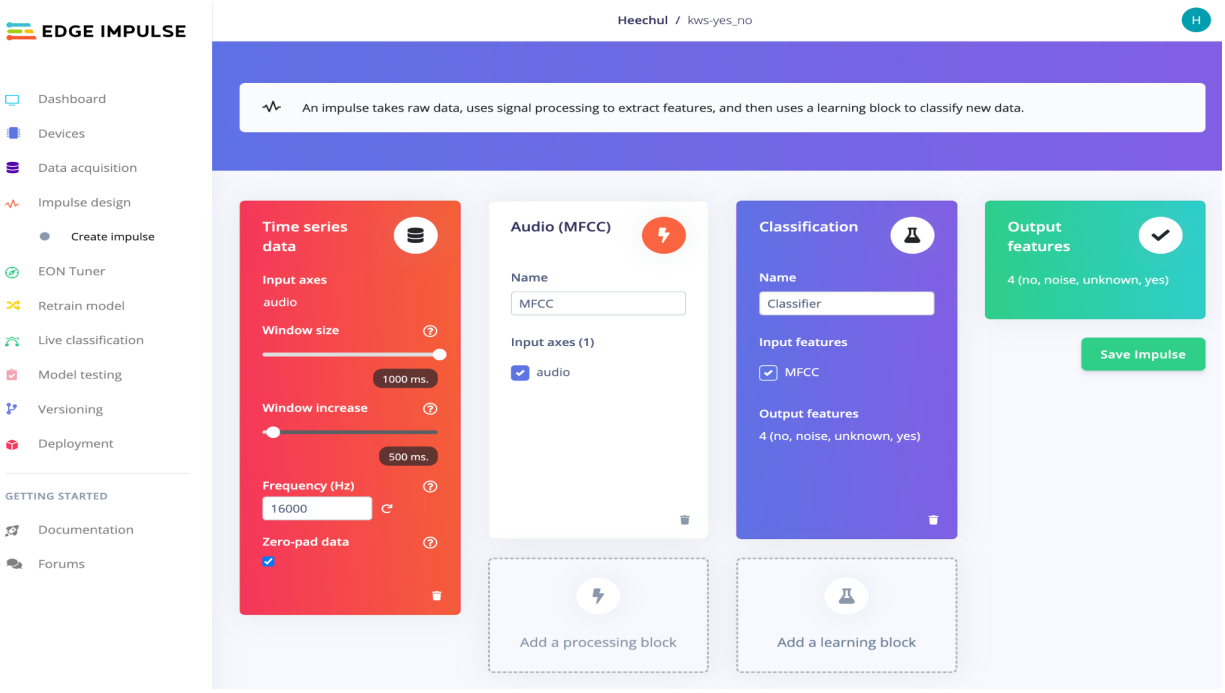
The samples will now appear in the Data acquisition section.



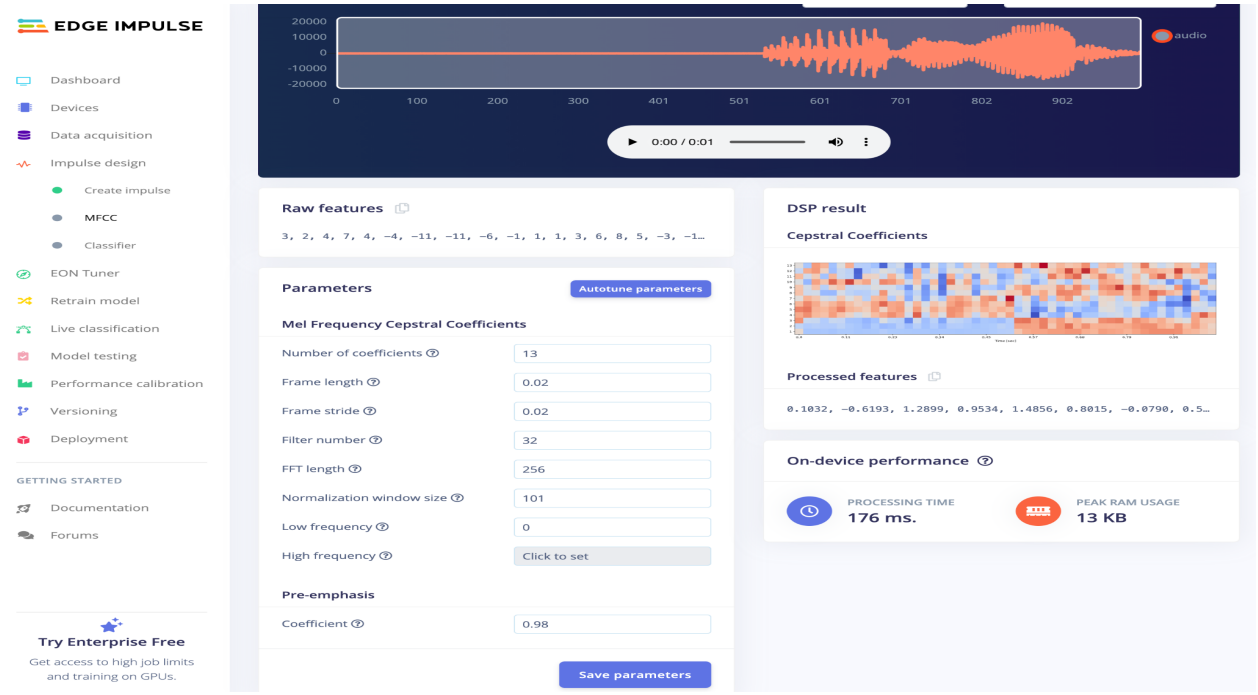
Task 1.2. Creating Impulse (pre-processing / model definition)

Now that we have data, we need to pre-process it so that we can use convolutional neural network models. Specifically, we will perform MFCC operations to transform the time-series audio signals into an image before feeding it to the neural network.

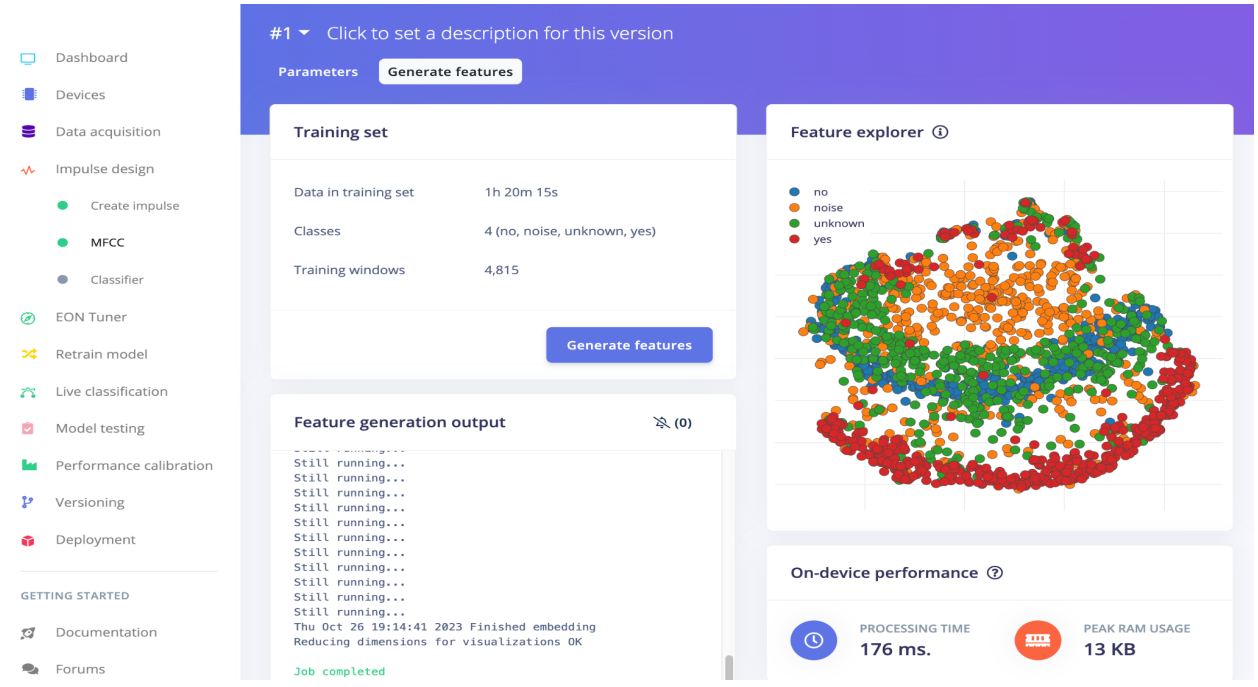
On the EdgeImpulse website, click the ‘Create impulse’ menu on the left.



Save Impulse.



You can adjust parameters (but encourage you to use the default values if you are not sure). Save parameters.

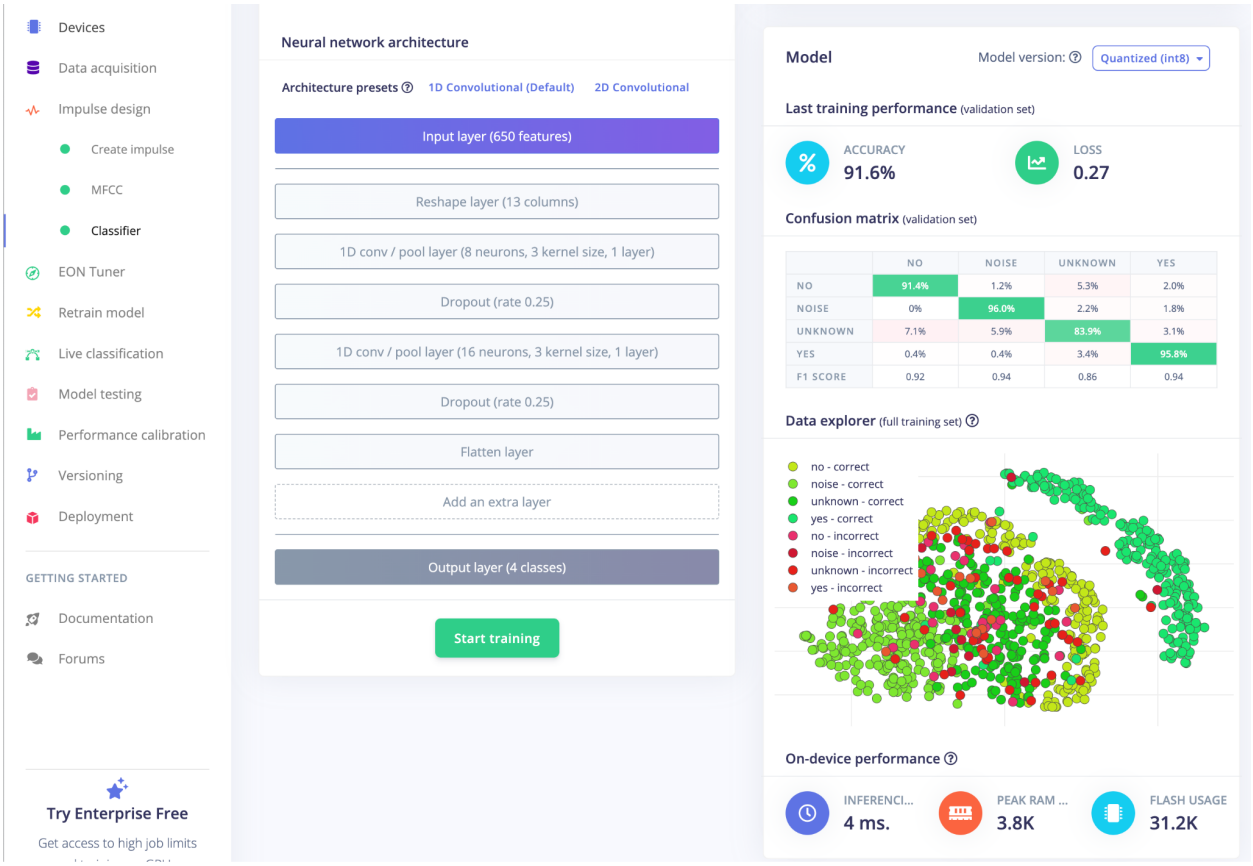


Generate features (which might take some time) and see the visualization results to see if there is a good separation between the classes.

Task 1.3. Model Design and Training

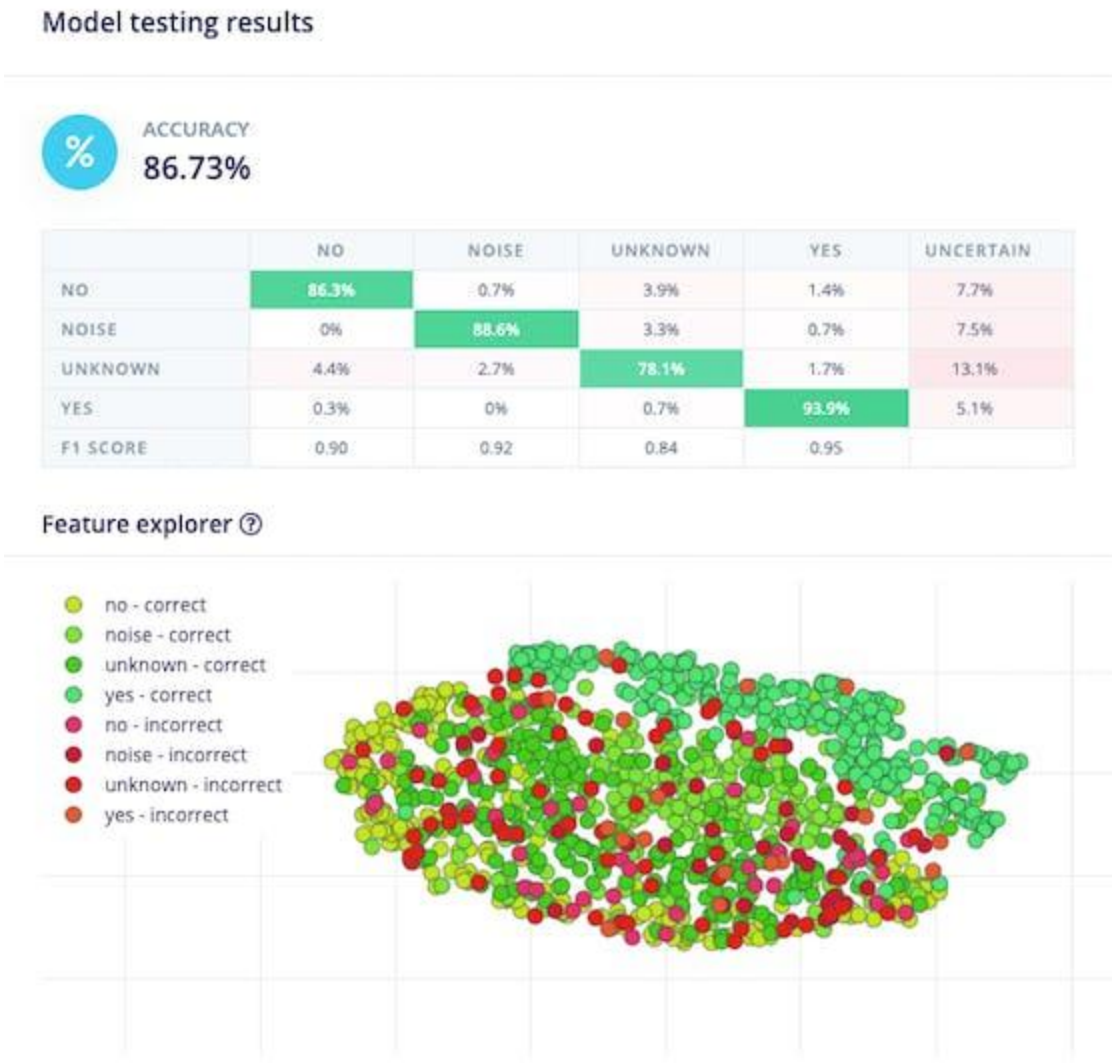
We will use a Convolution Neural Network (CNN) model. The basic architecture is defined with two blocks of Conv1D + MaxPooling (with 8 and 16 neurons, respectively) and a 0.25 Dropout. And on the last layer, after Flattening four neurons, one for each class:

As hyper-parameters, we will have a Learning Rate of 0.005 and a model that will be trained by 100 epochs. We will also include data augmentation, as some noise.

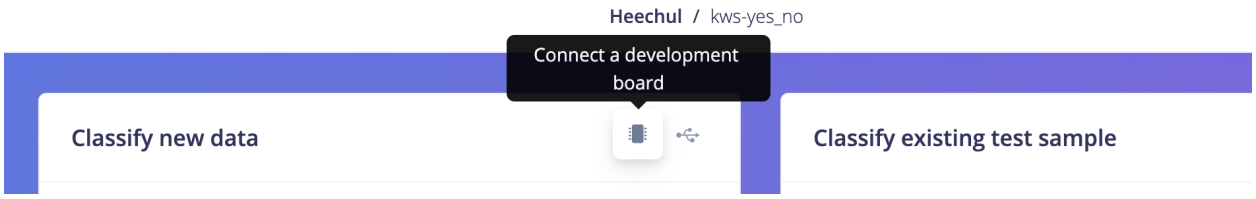


Task 1.4. Testing

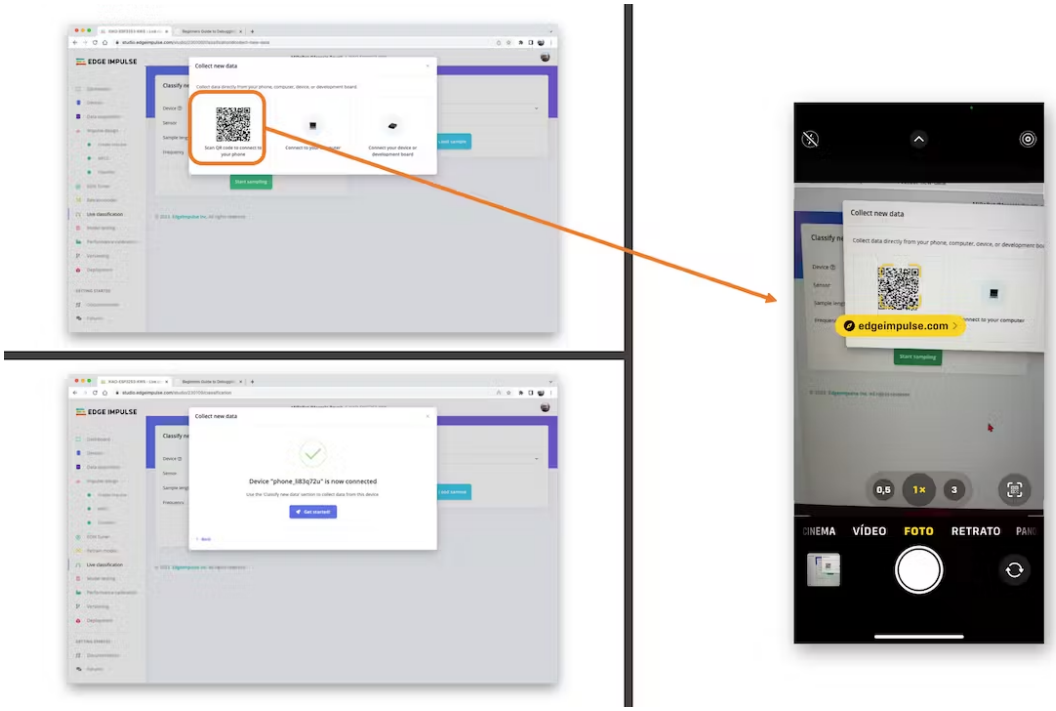
Testing the model with the data put apart before training (Test Data), we got an accuracy of approximately 87%.



We can proceed with the project, but it is possible to perform Live Classification using a smartphone before deployment on our device. Go to the *Live Classification* section and click on *Connect a Development board*:



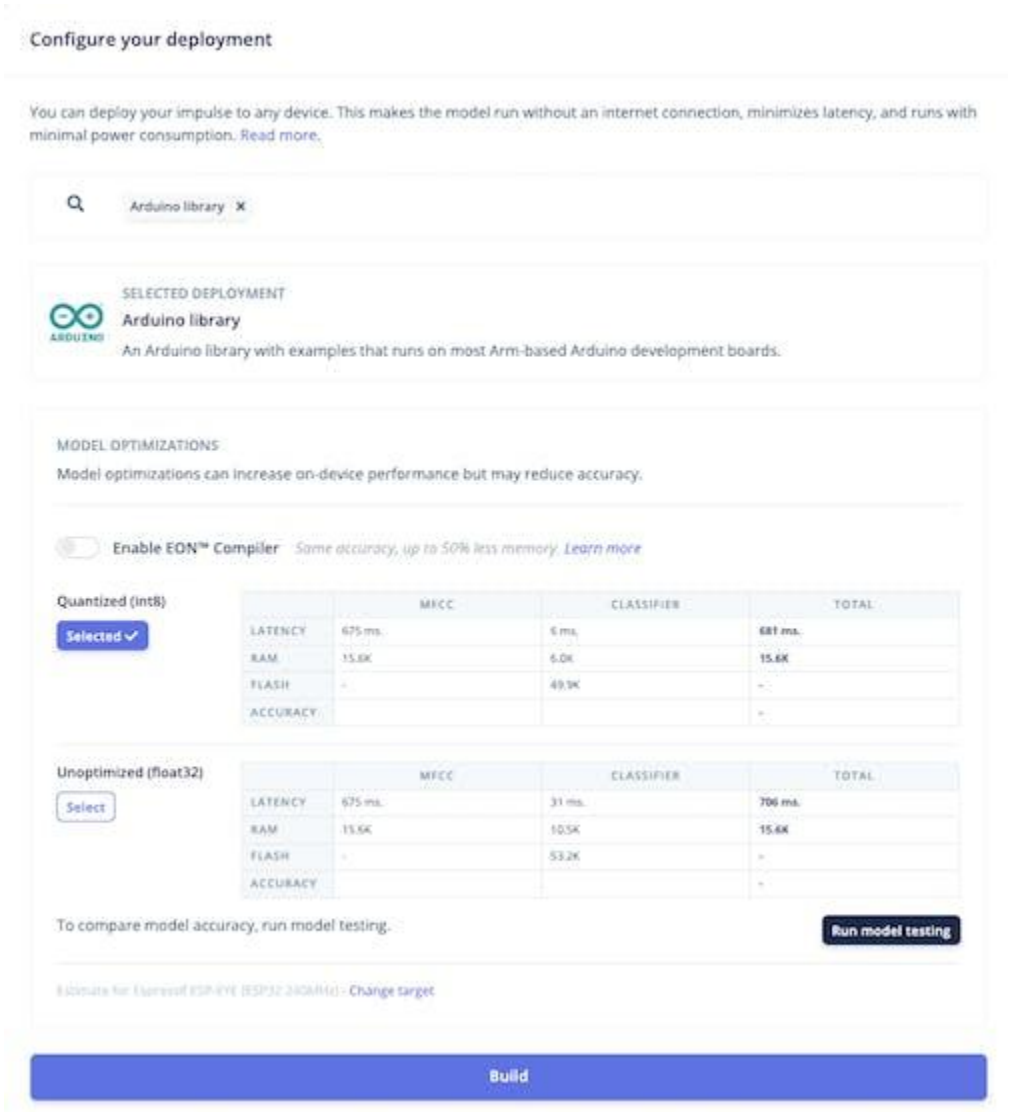
Point your phone to the barcode and select the link.



Your phone will be connected to the Studio. Select the option Classification on the app, and when it is running, start testing your keywords, confirming that the model is working with live and real data:

Task 1.5. Deployment and Inference

The Studio will package all the needed libraries, preprocessing functions, and trained models, downloading them to your computer. You should select the option Arduino Library, and at the bottom, choose Quantized (Int8) and press the button Build.



After some time, you will be downloading a single zip file (e.g., ei-kws-yes_no-arduino-1.0.4.zip)

Part 2. Testing the model on the ESP32-S3 MCU

Task 2.1. Setup a project.

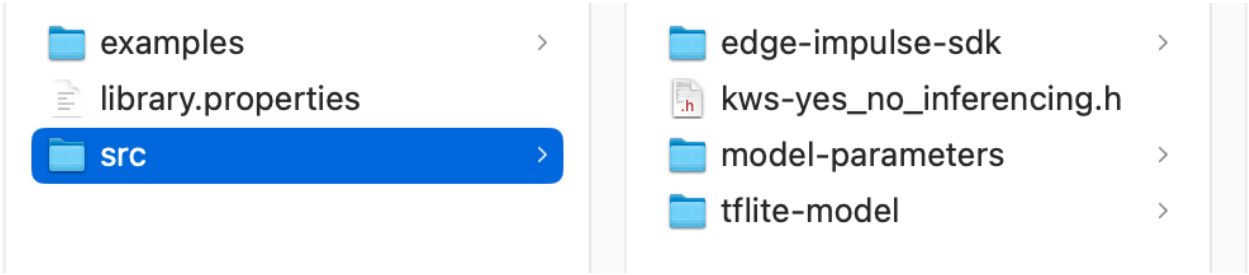
Download the sample project as follows.

```
$ mkdir -p ~/Documents/PlatformIO
$ cd ~/Documents/PlatformIO
$ git clone https://github.com/heechul/EmbeddedML
or
$ git pull
```

Task 2.2. Add the project folder

Add the '`l4-kws`' folder in VSCode.

Task 2.3. Copy the downloaded model into the project folder



Unzip the downloaded zip file
Copy the '`tflite-model`' and '`model-parameters`' folders into the '`l4-kws/src`' folder.
(do NOT copy the `edge-impulse-sdk` folder)

Task 2.4. Build & Upload to the ESP32-S3 board.

Hit the build and upload button.

References

https://www.hackster.io/mjrobot/tinyml-made-easy-keyword-spotting-kws-5fa6e7
https://docs.edgeimpulse.com/docs/pre-built-datasets/keyword-spotting