Deep Learning at the Edge – Activity 1

Task 1: Sound Classification/ Keyword Spotting

Introduction

In this lab, we will go through the steps of creating a speech recognition or keyword spotting (KWS) system. The same basic steps can be used to create a device that recognizes and classifies other sounds, too!

We will start with a simple, pre-made dataset and build a classifier for those sounds. It is best to find sounds made with something other than your voice!

In your Edge Impulse account, create a new project – **my-keyword-spotting-project**

Required Hardware

For collecting sound data, you should have access to a recording device. This can be a smartphone, webcam, laptop, etc.

For deploying, you can use your smartphone for a simple demo.

1. Collect Data

We will start with a pre-made keyword spotting dataset from Edge Impulse based on a subset of data in the Google Speech Commands Dataset, with added noise from the Microsoft Scalable Noisy Speech Dataset. It contains 25 minutes of data per class, split up in 1 second windows, sampled at 16,000Hz. The dataset contains:

- Yes - one second samples with only the word "yes" in it.
- No - one second samples with only the word "no" in it.
- Unknown - one second samples of other words.
- Noise - one second samples of background or static noise.

Importing this dataset

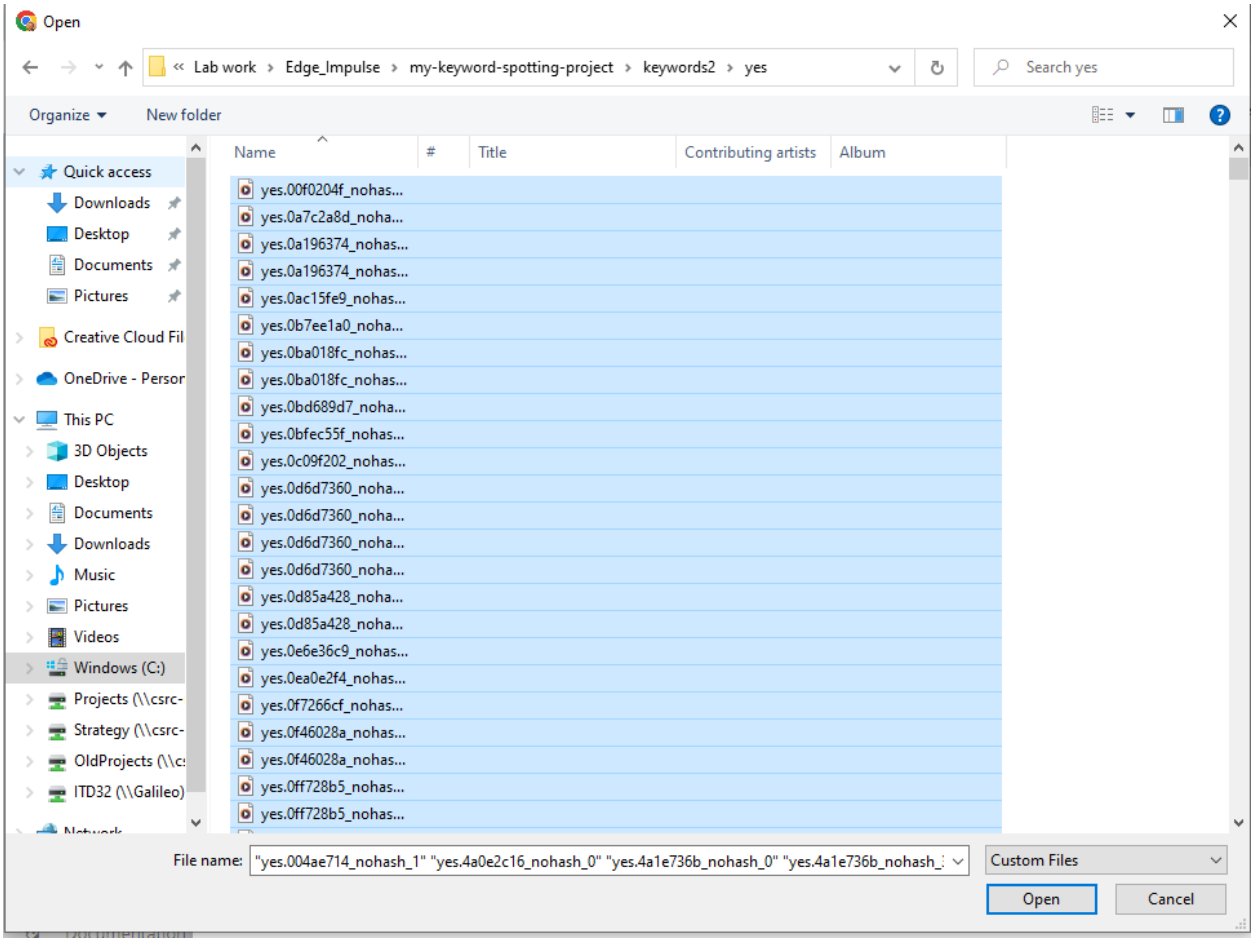
You can import this dataset to your Edge Impulse project by:

1. Downloading the keywords dataset
2. Unzip the file in a location of your choice
3. In your project, go to **Data acquisition** and click on the 'Upload icon'. Follow the instructions on the screen.

2. Upload Data

In your project in Edge Impulse. Head to the **Data Acquisition** page. Click **Let's collect some data**. Select the **Go to the upload data** option.

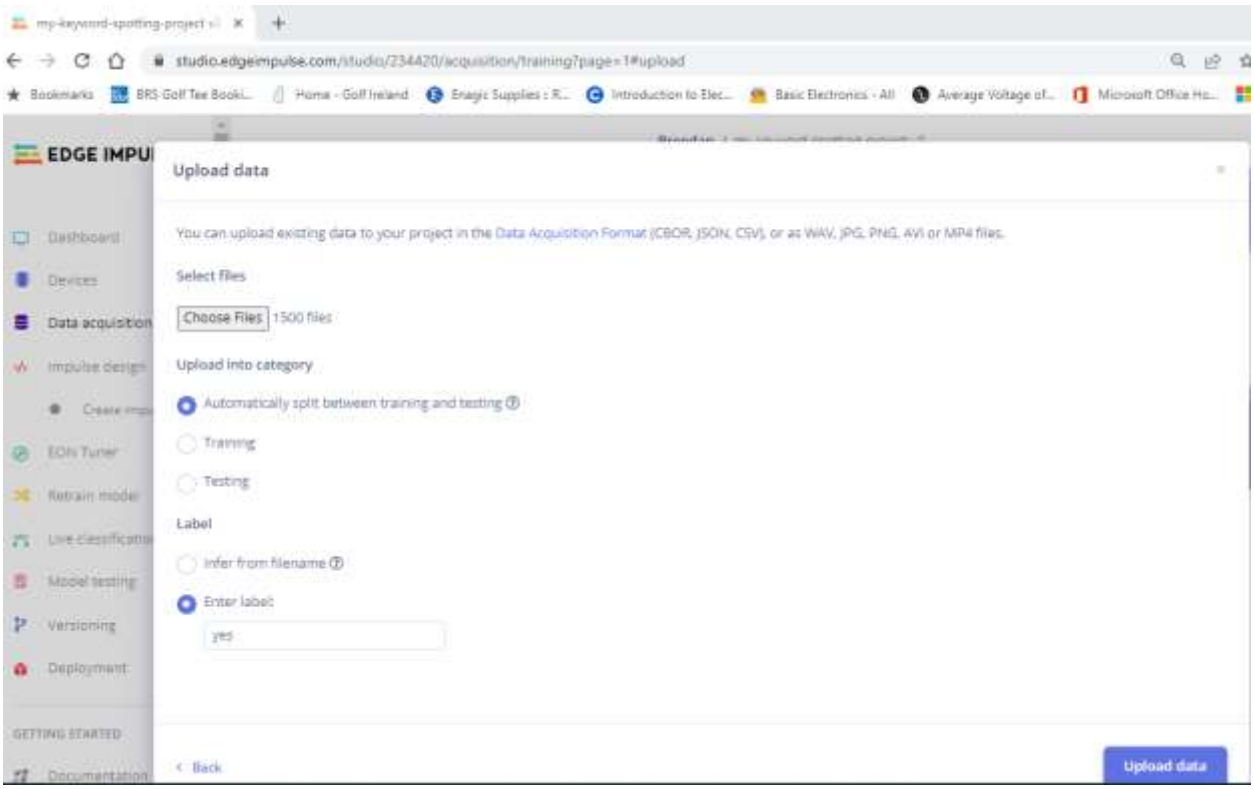
On this new page, click **Choose files**. Select all the 'yes' wav files from your unzipped sample set.



Click **Open**.

Leave *Automatically split between training and testing* selected. If the file naming scheme is as outlined above, leave *Infer from filename* selected. If not, select *Enter label* and give your samples a label (e.g. "yes"). I prefer to Enter a label just to make sure labelling is correct.

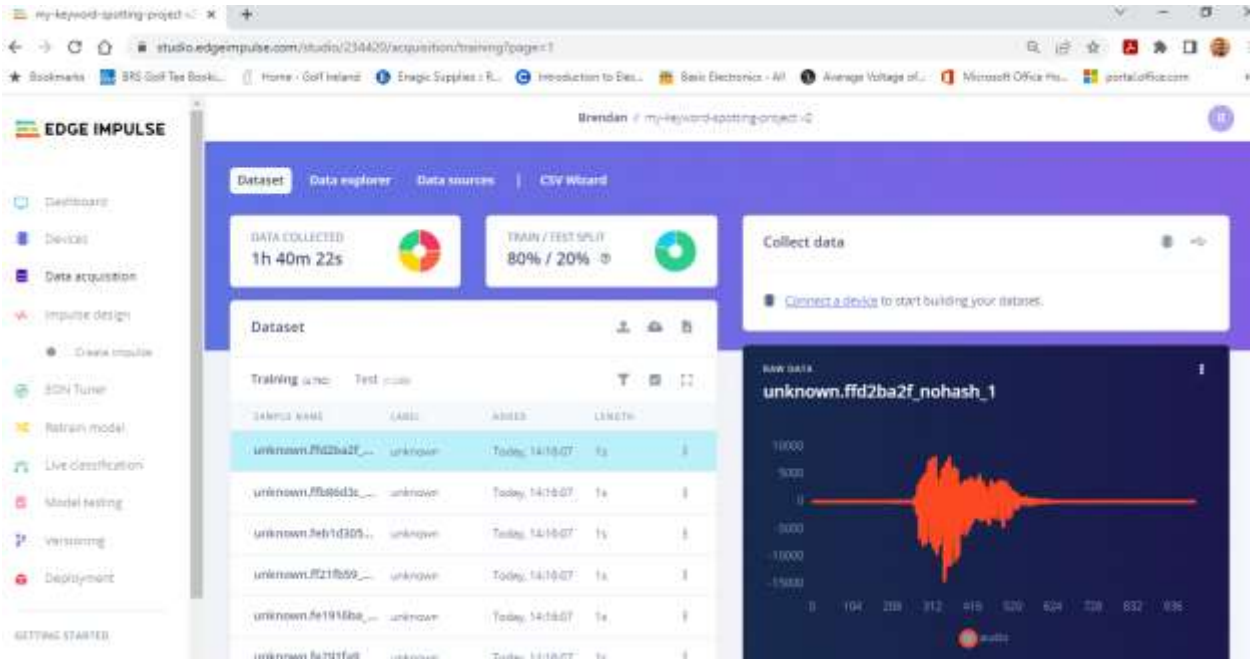
Deep Learning at the Edge – Activity 1



Click **Begin upload**.

Repeat this process for the *no, unknown* and *noise* sets that you extracted from the *keywords2* dataset ZIP file at the beginning. Make sure to add the correct labels for these samples – i.e. 'no', 'unknown' and 'noise'.

Click on the **Data acquisition** link to go back to the Data Acquisition page. Here, make sure that all of your samples are present and that they are divided between the training and test sets (there should be about 20% of the samples in the test set).

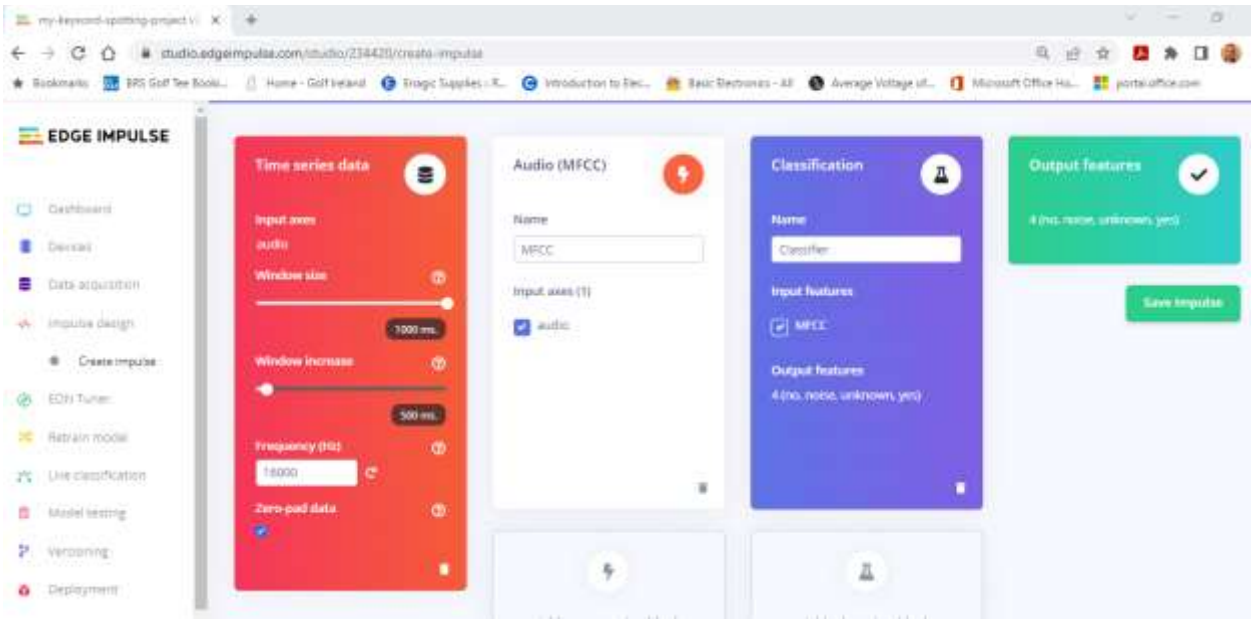


Alternate method: you can also collect audio data straight from your Edge Impulse project! Go to *Data acquisition* in a new project and connect your smartphone. (Here, I will demonstrate this method of adding some extra 'yes' and 'no' sample sounds into the train/test datasets– for eg. collect 10 seconds of data and then split into 10 separate sound clips.

3. Feature Extraction

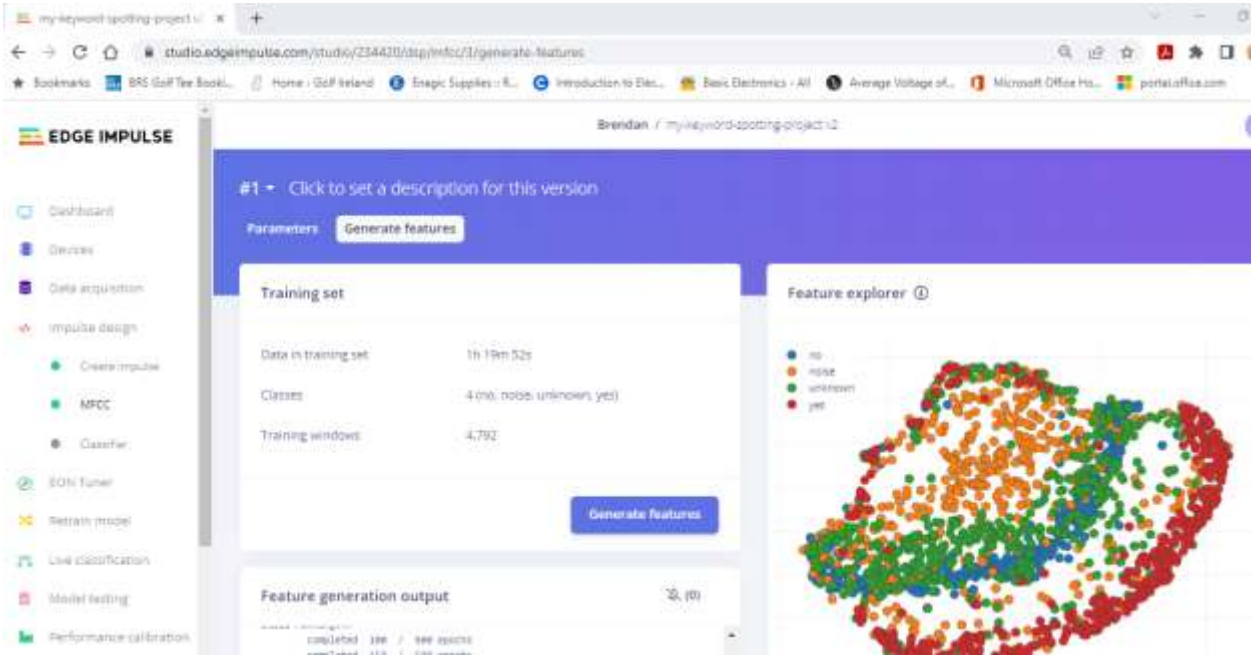
Navigate to the **Impulse design** page of your project. Add an **Audio (MFCC)** processing block and a **Classification** learning block.

Deep Learning at the Edge – Activity 1



Click **Save Impulse.**

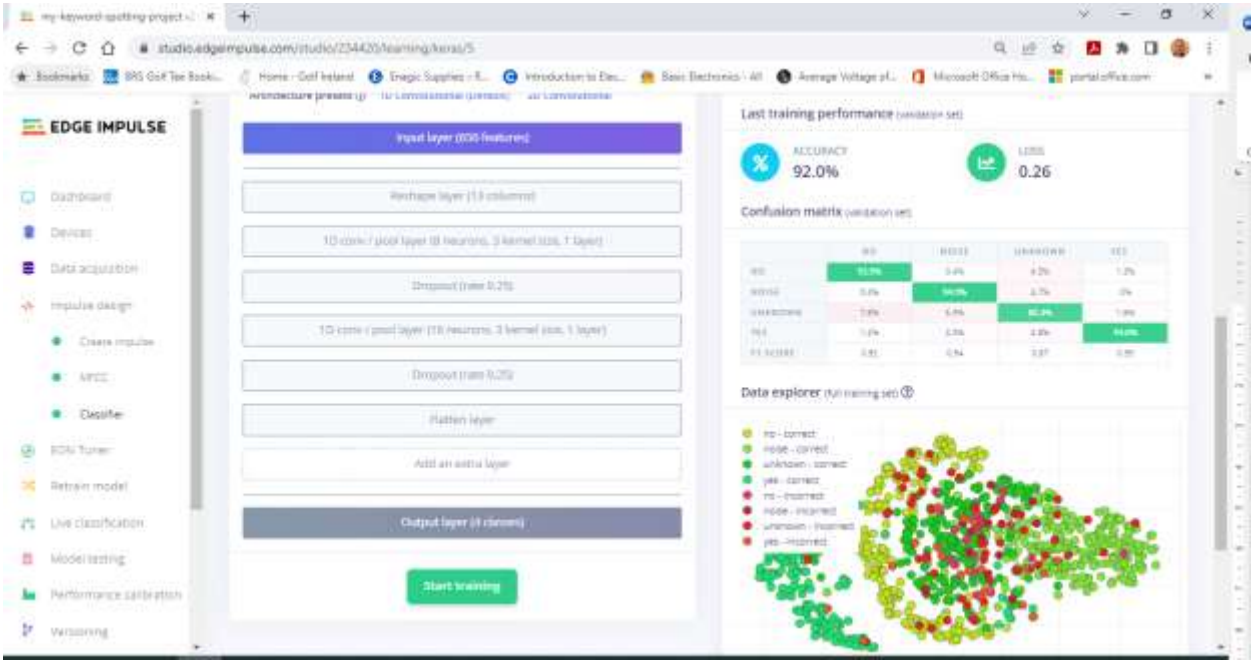
Go to the **MFCC** page and click on the **Generate Features** tab. Click the **Generate Features** button, and wait a moment while your audio samples are converted into spectrograms. When it's done, take a look at the *Feature explorer* to see if you can identify separation among your classes.



These features do not look like they are separated very well. However, it is enough to get started for a project like this.

4. Model Training

Navigate to the **Classifier** page. Leave all of the hyperparameters at their defaults and click **Start training**. When it's done, scroll down to view the *Confusion matrix* of the validation data.

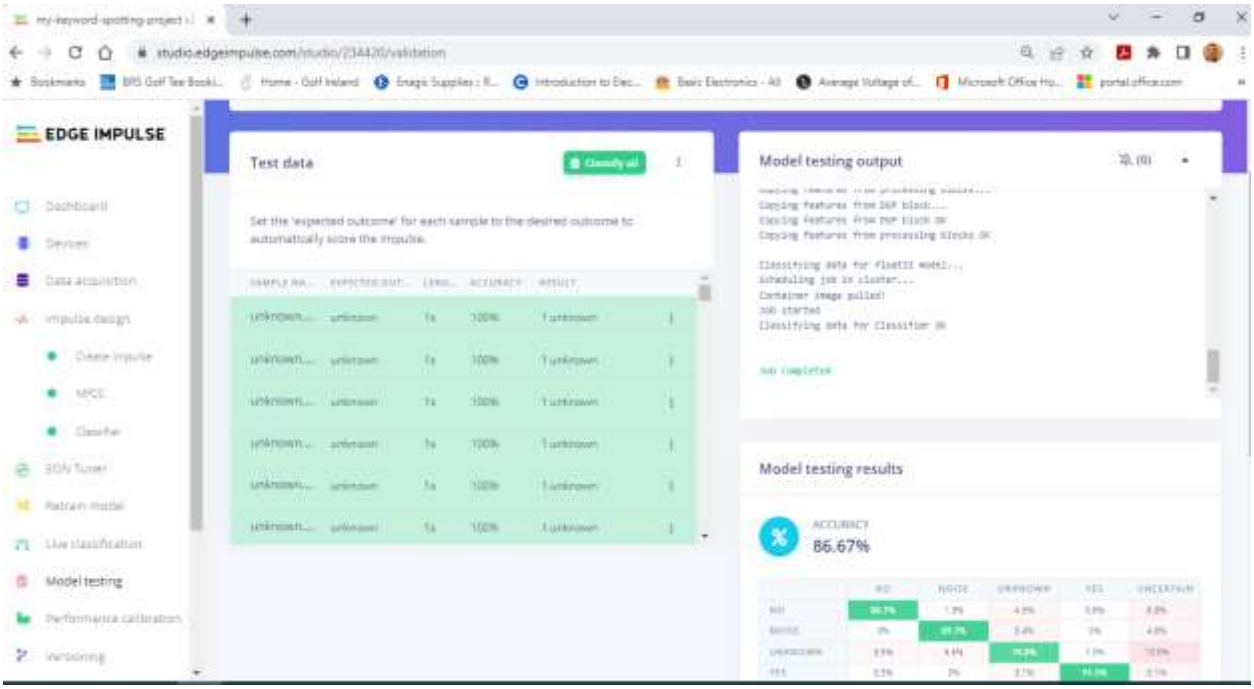


Note the *F1 scores* of each class and the *total accuracy*.

We have achieved a training accuracy of 92% which is quite good. Feel free to try changing some of the hyperparameters and re-training your model to see if it improves the per-class accuracy. Note that there is no easy solution here: much of creating a better model is trial and error, and sometimes, you simply don't have enough (or the right kind of) data to train a good model. In those cases, it's back to the drawing board to gather data!

5. Testing

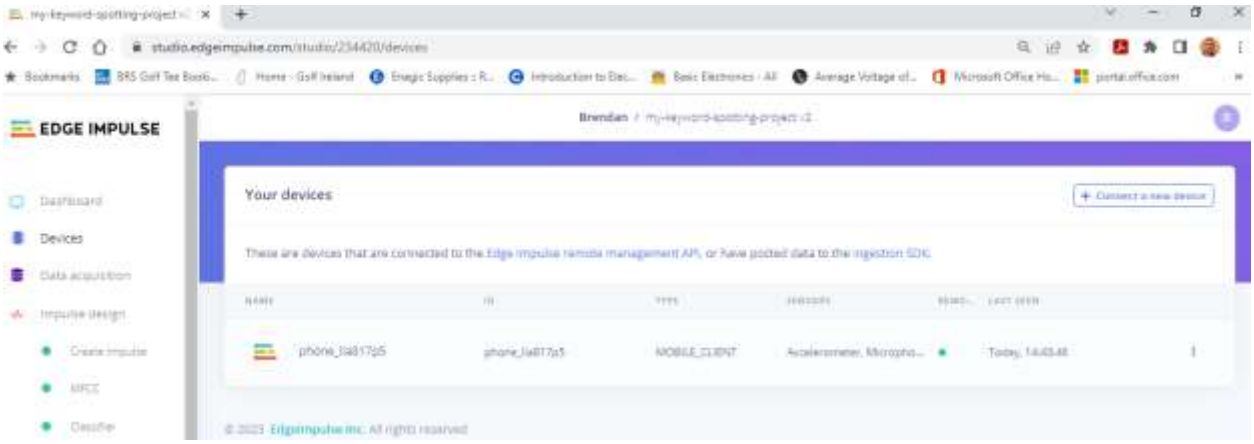
Head to the **Model testing** page and classify all of the test data.



If you're happy with the test results, continue to the deployment step. 86% is quite good – anything less than this, than you probably need to collect more data and adjust hyperparameters or change the model type.

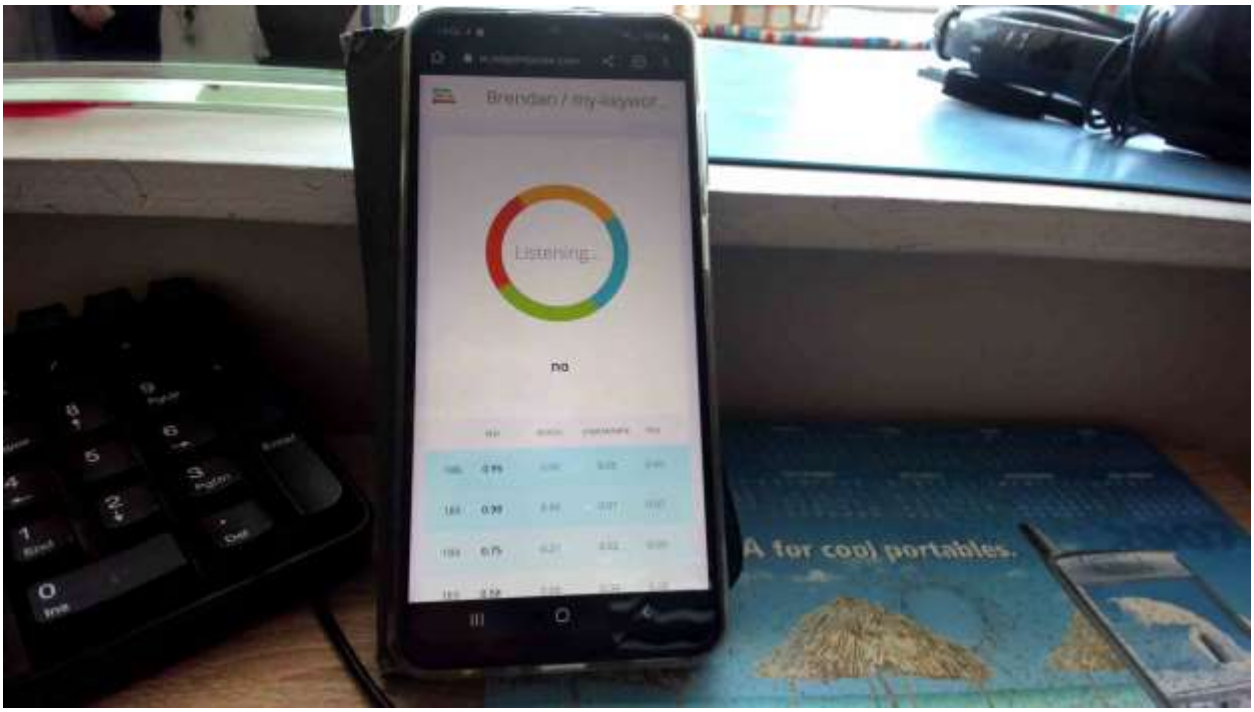
6. Deployment

Connect your smartphone to your project (if you have not done so already). Go to the **Devices** page and scan the QR code to add your smartphone as a device.



Head to *smartphone.edgeimpulse.com* on your phone's browser. Scroll down and click **Switch to classification mode**. If asked, allow the program to access your phone's microphone.

Hold your phone's microphone near various sources of sound to see if it can identify them.



7. Conclusion

I hope this has helped you get started using machine learning with audio data! As you've probably seen, choosing a model and adjusting hyperparameters can be a messy process with a lot of trial and error. A lot of time goes into creating a robust model that works in most environments, especially when it comes to identifying sounds!

Please note that this project is just a start. It will require a lot more data and effort to get a working model ready for production.