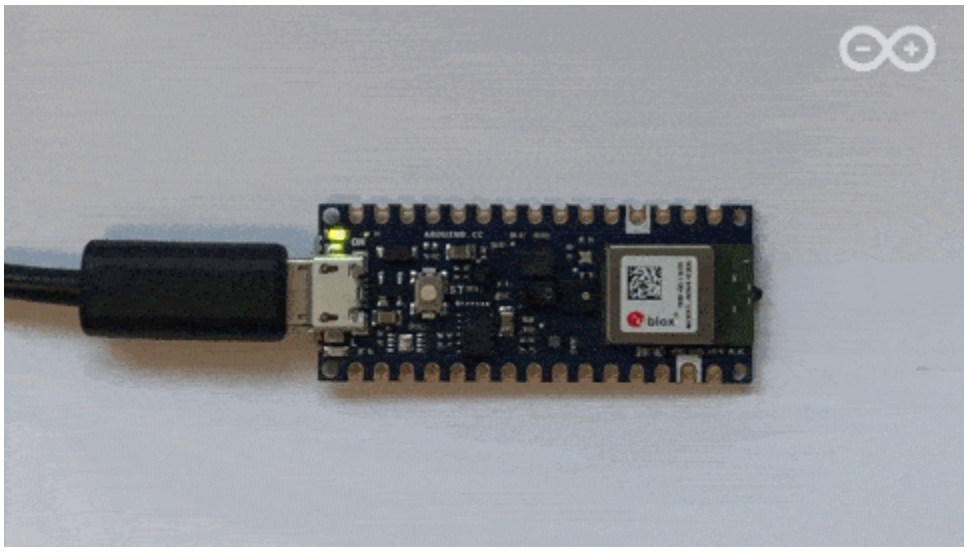
BOISE STATE UNIVERSITY

DEPLOYING A KWS MODEL IN ARDUINO IDE AND TRAINING YOUR OWN DATA WITH EDGE IMPULSE

Since the KWS editable colab file is no longer available, we will first look at how the Arduino nano BLE 33 works with the existing inferred KWS model, change the LEDS for “yes” and “no,” and then shift to edge impulse IDE for training with your own dataset or an existing dataset.

MICRO SPEECH EXAMPLE (KEYWORDSPOTTING)

This example shows how to run a 20 kB model that can recognize 2 keywords, "yes" and "no", from speech data. The application listens to its surroundings with a microphone. Depending on the device's capabilities, it indicates when it has detected a word by lighting an LED or displaying data on a screen.



The code has a small footprint (around 166 kilobytes on a Cortex M4) and only uses about 54 kilobytes of additional RAM for working memory.

THE ARDUINO CODE

1. Go to **File -> Examples**. You should see an entry within the list named **Arduino_TensorFlowLite.** Select it and click **micro_speech** to load the example.
2. Use the Arduino IDE to build and upload the example. Once it is running, you should see the built-in LED on your device flashing. The built-in LED will flash on/off for each inference cycle. Saying the word "yes" will cause the green LED to remain on for 3 seconds. The current model has fairly low accuracy, so you may have to repeat "yes" a few times. Saying the word "no" will cause the red LED to light up. The blue LED will be lit for certain "unknown" sounds.

***Word recognition should occur at a distance of approximately 1.5 feet in a low-noise environment.*

The program also outputs inference results to the serial port, which appear as follows:

```
Heard unknown (202) @22032ms
Heard yes (212) @25200ms
Heard yes (213) @27648ms
Heard no (212) @29792ms
Heard no (206) @31840ms
Heard yes (205) @33984ms
Heard yes (207) @36112ms
```

The number after each detected word is its score. By default, the program only considers matches as valid if their score is over 200, so all of the scores you see will be at least 200.

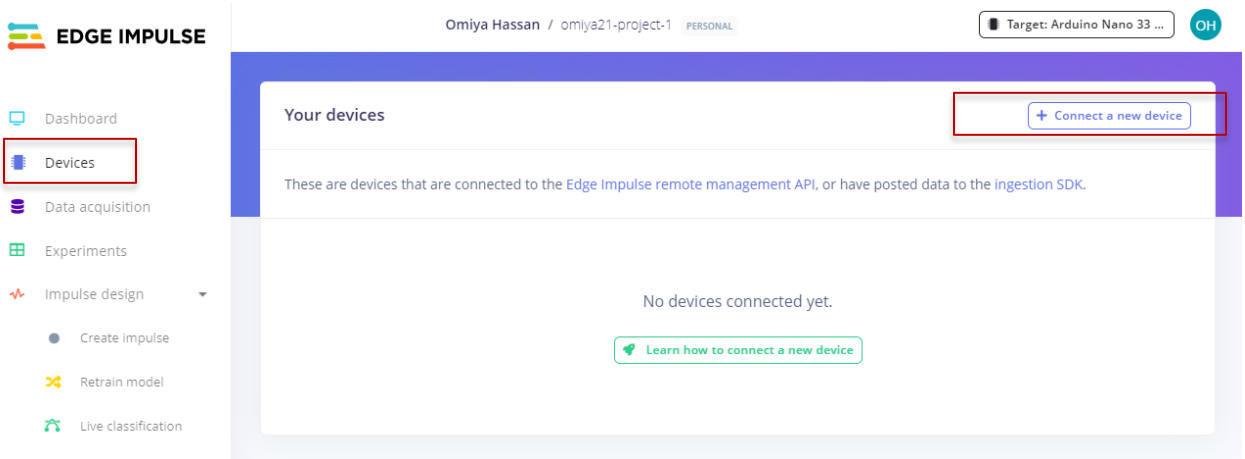
When the program is run, it waits several seconds for a USB-serial connection to be available. If there is no connection available, it will not output data. To see the serial output in the Arduino desktop IDE, do the following:

1. Open the Arduino IDE
2. Connect the Arduino board to your computer via USB
3. Press the reset button on the Arduino board
4. Within 5 seconds, go to **Tools -> Serial Monitor** in the Arduino IDE. You may have to try several times, since the board will take a moment to connect.

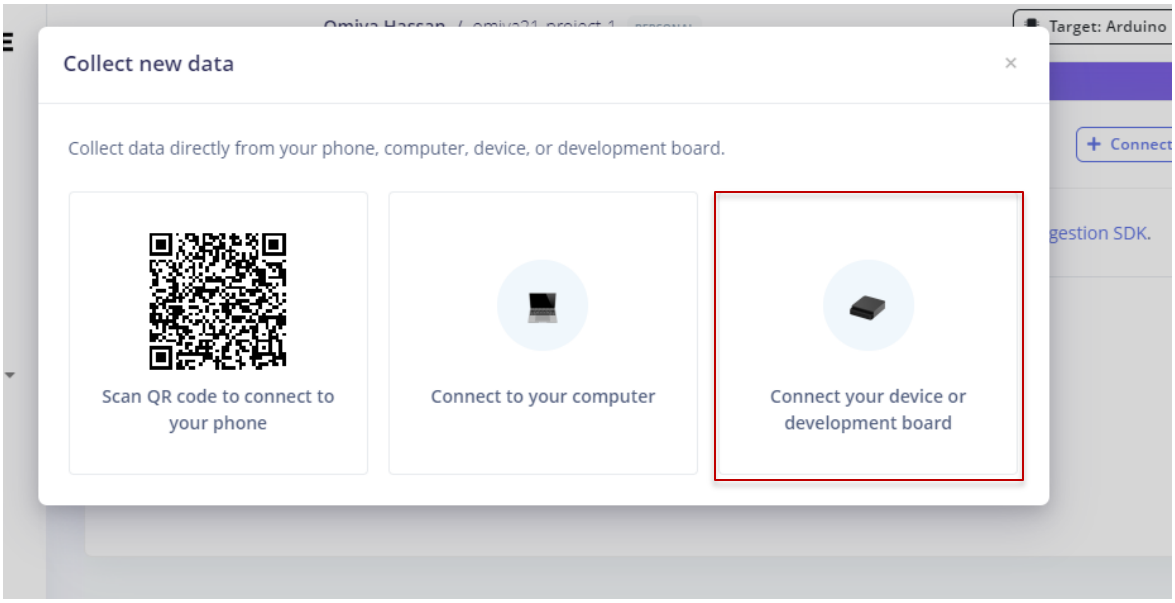
If you don't see any output, repeat the process again.

USING YOUR OWN VOICE/ EXISTING DATASET WITH EDGE IMPULSE

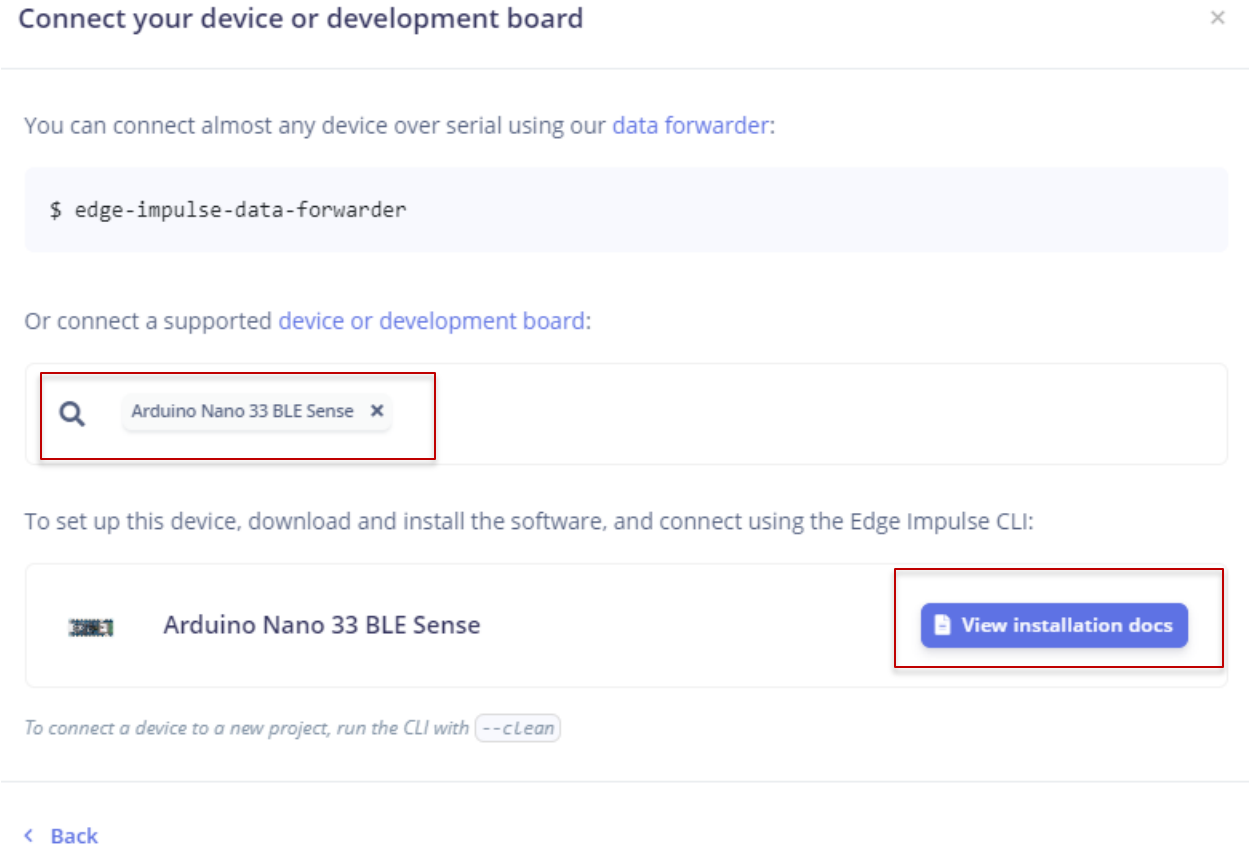
1. Create an account in the edge impulse (it's free)
2. After you signed up you need to add your Arduino Device in the Device Tab
3. Click on the "+ connect a new device"



4. It should prompt you to another popup window and you select "connect your device or dev board"



5. Search for "Arduino Nano 33 BLE Sense Microcontroller and click on the "view installation docs"



6. It will prompt you to another tab and you will need to download all the programs and files necessary to connect your microcontroller in the following installation guide.
7. If you are using Widnwos 11 OS and having trouble installing the Node.js,follow the instructions from the Github Link provided: https://github.com/IAU5/Nodejs_edgeImpulse_CLI/tree/main

CREATE PROJECT AND CONNECT YOUR BOARD

1. Go to your edge Impulse account and create a new project with the title of your choice
2. Bring up your terminal or command prompt and type: `edge-impulse-daemon`
    a. If you want to connect a different type of board (not Nano BLE 33) then you need to type: `edge-impulse-daemon –clean`
3. Type in your edge impulse email address and password
4. Connect to the project you created in the edge impulse project board and click enter
5. Go to your edge impulse account and double check if your Arduino board has been successfully connected

KWS WITH EDGE IMPULSE

Pre-Built Dataset: This is a prebuilt dataset for a keyword spotting system 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 mins of data per cycle, split it up to 1 second windows and sampled at 16kHz.

1. Select "Yes" and "No", Unknown and Noise
2. You can import the dataset to your Edge Impulse project through Uploader in the studio or via the CLI (docs).
    a. Download the keyword dataset
    b. Unzip the folder in the location of your choice
3. For uploading in the Edge Impulse studio simply click on the upload button/icon
4. For importing with CLI, open a terminal or command prompt and navigate to the place where you extracted the file, then run

```
$ edge-impulse-uploader --clean
$ edge-impulse-uploader --label noise --category split noise/*.wav
$ edge-impulse-uploader --label unknown --category split unknown/*.wav
$ edge-impulse-uploader --label no --category split no/*.wav
$ edge-impulse-uploader --label yes --category split yes/*.wav
```

Create your own dataset:

1. Go to **Account>Data Acquisition** and in the record new data section label your data as "Left".
2. Keep your **sample rate** and **frequency** the same
3. Select **"built-in microphone"** in the **Sensor** option
4. Say "Left" multiple times within the sampling second by pausing for 1 sec. *(use different voices to increase robustness of your model.)*
5. After sampling the record tab should show you the audio wave signal
6. Select the "split" sample option and select the places where you said "Left*". (it should automatically define the "Left" voices but tinker with it a bit)*
7. Create the split samples and it will create the "Left" samples in multiple numbers.
8. Play each sample to make sure the "left" sounded correct and clear.
9. Create more for generating better training data
10. Do the same procedure for "Right" and then for some background noise

Data Acquisition Tab

1. Remember to split your data into 80% training and 20% testing

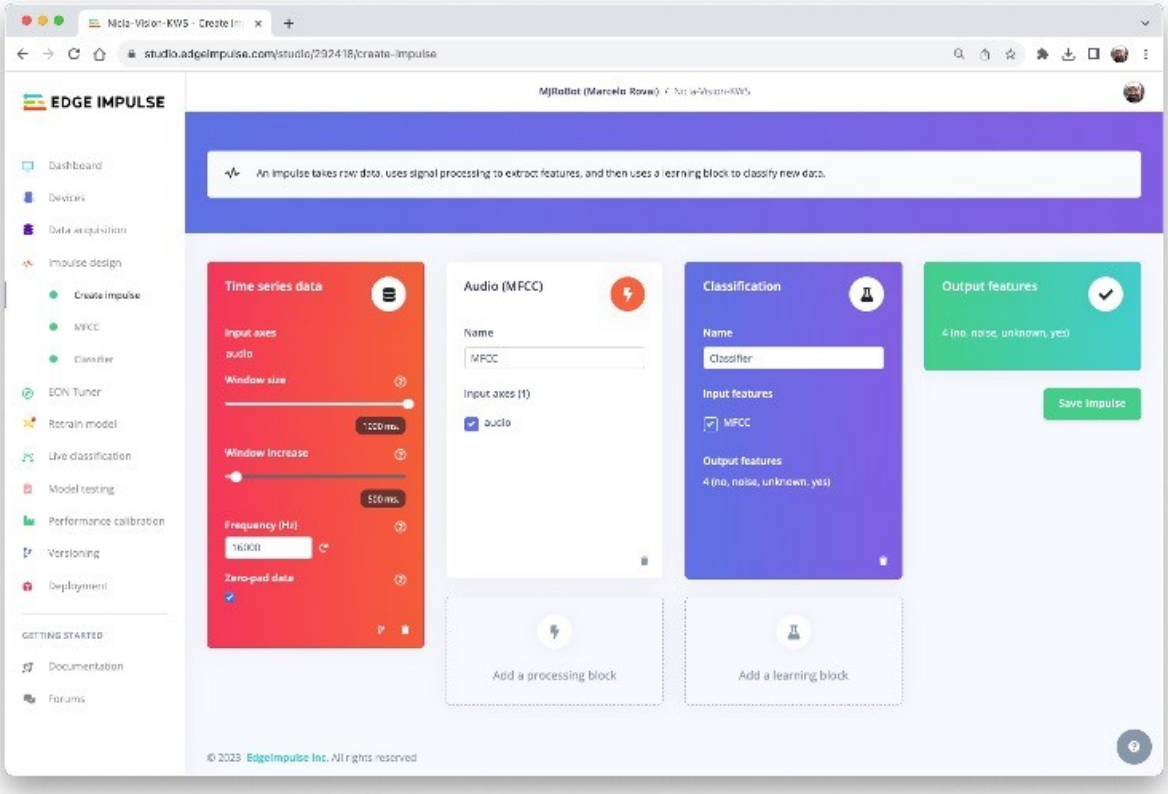
Create your Impulse

1. Go to **Impulse design> Create Impulse**

2. In add processing Block tab select “Audio MFCC”

MFCC is called Mel Frequency Cepstral Coefficient which is a mathematical model used to split human voice.

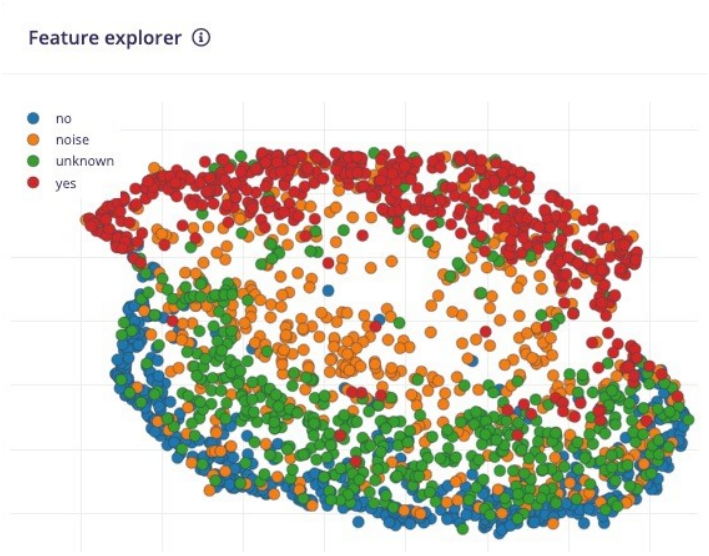
3. Then in Add **a learning block** you can select “Classification (keras)” model. The classification block is an untrained model that we will train with our selected and generated dataset. (*You can also play with the “transfer learning model” option which is a robust model already trained on existing dataset.)*
4. Select the “MFCC” option in the “classification block” and Save impulse



Edge Impulse “Create Impulse” Dashboard after selecting necessary blocks.

Feature Explorer

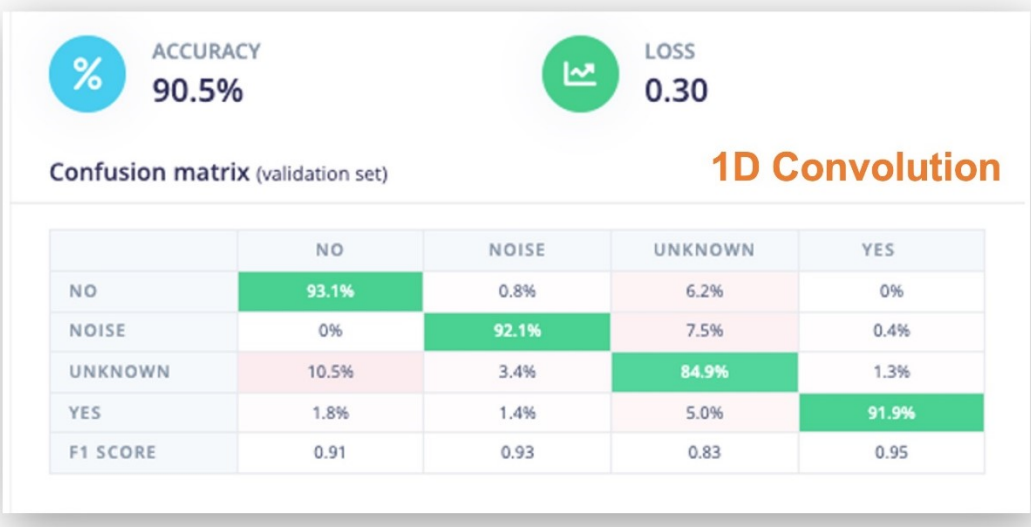
- Go to **Impulse Design > MFCC**
- You can check out each dataset and their corresponding image. Make necessary tweaks if needed.
- Then Click on the “save parameters” button
- It will prompt you to “Generate Feature” tab Click on “Generate Feature” button
- The generate feature tab will take in your audio datasets and generate MFCC
- After proper generation of data it will prompt you to “Feature Explorer” tab which will give you a UMAP of your dataset samples.
- This will be easy to identify whether your distribution is grouped and distributed correctly.



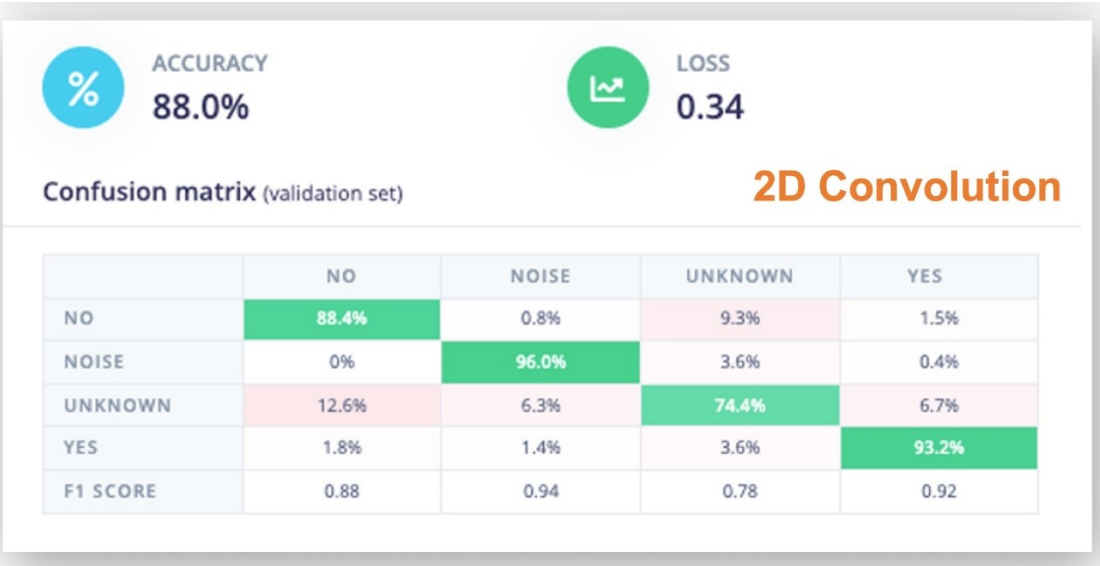
The actual features will be generated. Using UMAP, a dimension reduction technique, the Feature explorer shows how the features are distributed on a two-dimensional plot.

Classifier

- Go to **NN Classifier** tab
- You can mark your Neural Network training either in block level or expert mode
- For the first training work with block level and select 1D CNN
- Click on the Save & Train button
- In the training output you are able to see your loss, accuracy and val_accuracy in the "training output" tab over each epoch.
- After training you can check your confusion matrix
- Train your dataset with the 1D and the 2D CNN models with the same hyperparameters.



	NO	NOISE	UNKNOWN	YES
NO	93.1%	0.8%	6.2%	0%
NOISE	0%	92.1%	7.5%	0.4%
UNKNOWN	10.5%	3.4%	84.9%	1.3%
YES	1.8%	1.4%	5.0%	91.9%
F1 SCORE	0.91	0.93	0.83	0.95



ACCURACY
88.0%

LOSS
0.34

Confusion matrix (validation set)

2D Convolution

	NO	NOISE	UNKNOWN	YES
NO	88.4%	0.8%	9.3%	1.5%
NOISE	0%	96.0%	3.6%	0.4%
UNKNOWN	12.6%	6.3%	74.4%	6.7%
YES	1.8%	1.4%	3.6%	93.2%
F1 SCORE	0.88	0.94	0.78	0.92

It is also interesting to pay attention to the 1D Confusion Matrix from the sample model. The F1 Score for yes is 95%, and for no, 91%. That was expected by what we saw with the Feature Explorer (no and unknown at close distance). In trying to improve the result, you can inspect closely the results of the samples with an error.

**For community version you are only able to train your model until 20 mins so be cautious of building your model.

Under The Hood

If you want to understand what is happening "under the hood," you can download the pre-processed dataset (MFCC training data) from the Dashboard tab and run this Jupyter Notebook, playing with the code For example, you can analyze the accuracy by each epoch.

REPORT

1. Document your test result using the "yes" and "no" dataset and achieve accuracy over 90%
2. Generate your own dataset by saying "Left" and "Right" or your two words of choice. Remember to include noise and silence as well for better training
3. Include the UMAP of your generated dataset
4. Train your own model and report the evaluation metrics of your model using the "classifier" option.
5. Document "on device" section after training to see your inference time, memory utilization and peak performance.
6. Document feature explorer section and include the generated graph after testing.

Helpful Links to Get you Started

1. Keyword Spotting using Edge Impulse – Shawn Hymel
2. KWS using Mic and Edge Impulse- RoboCraze
3. Keyword Spotting (KWS) using NiclaV

Happy Inference!