

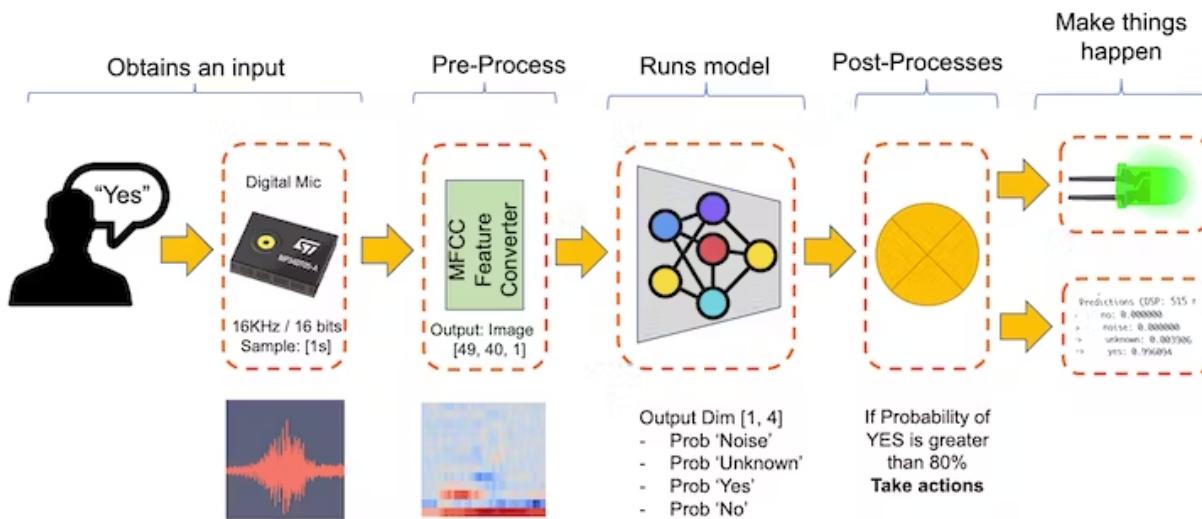
EECS 690/700 EmbeddedML Lab #4

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.

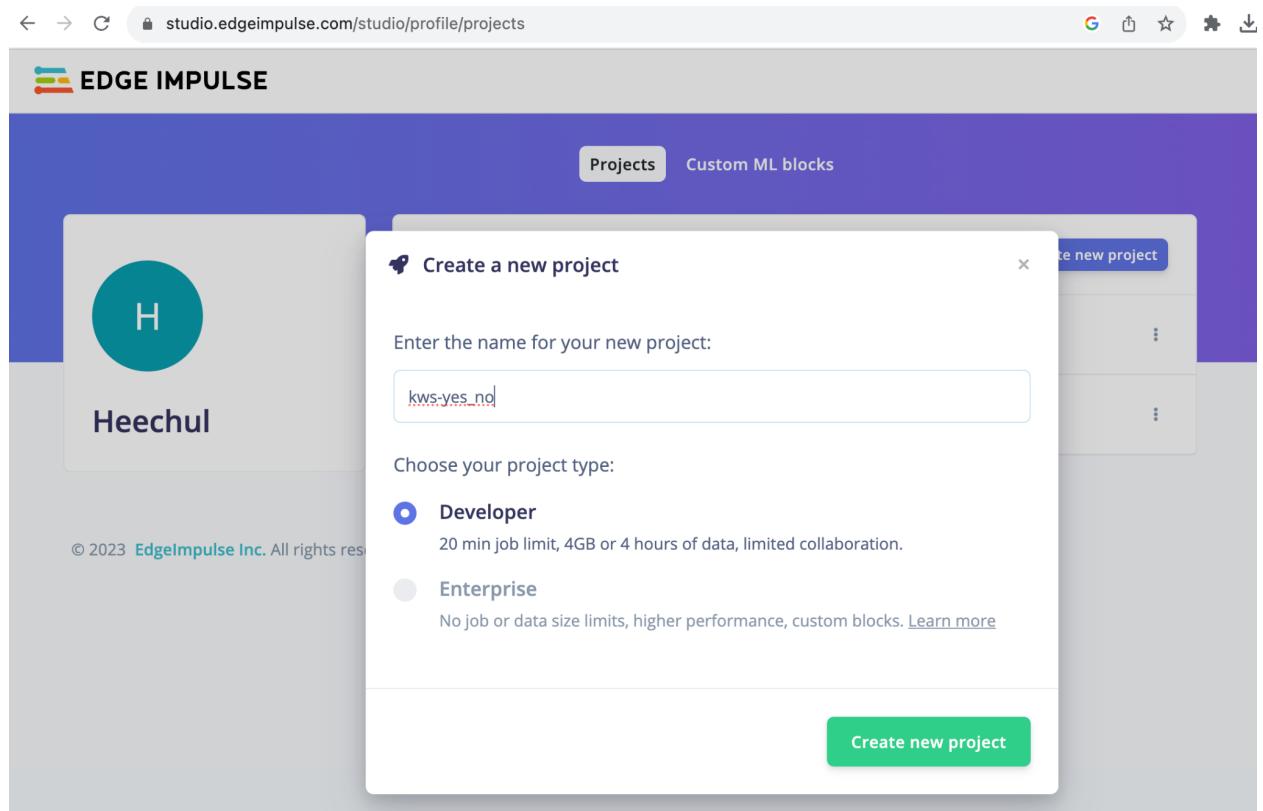
Part 1. Training KWS model with Edge Impulse

We will be using Edge Impulse for KWS application.

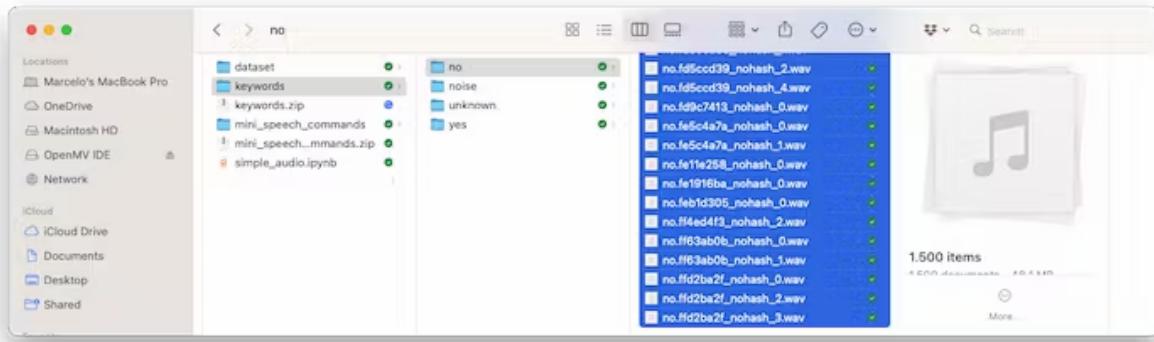
Task 1.1. Creating a new Edge Impulse project

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.

studio.edgeimpulse.com/studio/299971/acquisition/training?page=1#upload

You can upload existing data to your project in the [Data Acquisition Format](#) (CBOR, JSON, CSV), or as WAV, JPG, PNG, AVI or MP4 files. We also support uploading image datasets with labels in various formats. When you include labels during upload, we attempt to convert your dataset into a format recognized by Studio. [here](#).

Upload mode

- Select individual files [?](#)
- Select a folder [?](#)

Select files

[Choose Files](#) 1500 files

Upload into category

- Automatically split between training and testing [?](#)
- Training
- Testing

Label

- Infer from filename [?](#)
- Leave data unlabeled [?](#)
- Enter label:

Upload output

```
[1283/1500] Uploading yes.9bea2ac7_nohash_0.wav OK
[1284/1500] Uploading yes.3d3ddaf8_nohash_0.wav OK
[1285/1500] Uploading yes.cdee383b_nohash_2.wav OK
[1286/1500] Uploading yes.e53139ad_nohash_3.wav OK
[1287/1500] Uploading yes.bcf614a2_nohash_2.wav OK
[1288/1500] Uploading yes.3ae5c04f_nohash_1.wav OK
[1289/1500] Uploading yes.c392e01d_nohash_0.wav OK
[1290/1500] Uploading yes.6a497f80_nohash_1.wav OK
[1291/1500] Uploading yes.1dc86f91_nohash_2.wav OK
[1292/1500] Uploading yes.3ee0fc39_nohash_2.wav OK
[1293/1500] Uploading yes.042ea76c_nohash_1.wav OK
[1294/1500] Uploading yes.1b5ba788_nohash_0.wav OK
[1295/1500] Uploading yes.4c6167ca_nohash_8.wav OK
[1296/1500] Uploading yes.8377f7378_nohash_0.wav OK
[1297/1500] Uploading yes.5170b77f_nohash_0.wav OK
[1298/1500] Uploading yes.840eab5a_nohash_0.wav OK
[1299/1500] Uploading yes.9a69672b_nohash_4.wav OK
[1300/1500] Uploading yes.caa7feaf_nohash_0.wav OK
[1301/1500] Uploading yes.9e92ef0c_nohash_0.wav OK
[1302/1500] Uploading yes.617ae6bc_nohash_0.wav OK
[1303/1500] Uploading yes.fb7eb481_nohash_0.wav OK
[1304/1500] Uploading yes.834f03fe_nohash_1.wav OK
```

Cancel upload

[Back](#) [Upload data](#)

The samples will now appear in the Data acquisition section.

The screenshot shows the Edge Impulse web interface. On the left, there's a sidebar with various menu items: Dashboard, Devices, Data acquisition, Impulse design (selected), EON Tuner, Retrain model, Live classification, Model testing, Versioning, Deployment, and a Try Enterprise Free section with a Start free trial button. The main area has tabs for Dataset, Data explorer, Data sources, and CSV Wizard. A summary card shows 'DATA COLLECTED 1h 40m 5s' and 'TRAIN / T... 80...' with a progress bar. Below is a 'Dataset' table with columns for SAMPLE NAME, LABEL, ADDED, and LENGTH. The table lists several entries like 'unknown.40...' and 'unknown.51...'. To the right is a 'Collect data' section with a 'Connect a device' button. At the bottom is a 'RAW DATA' visualization for 'unknown.40738a2d_nohash_0' showing a red waveform against a dark background.

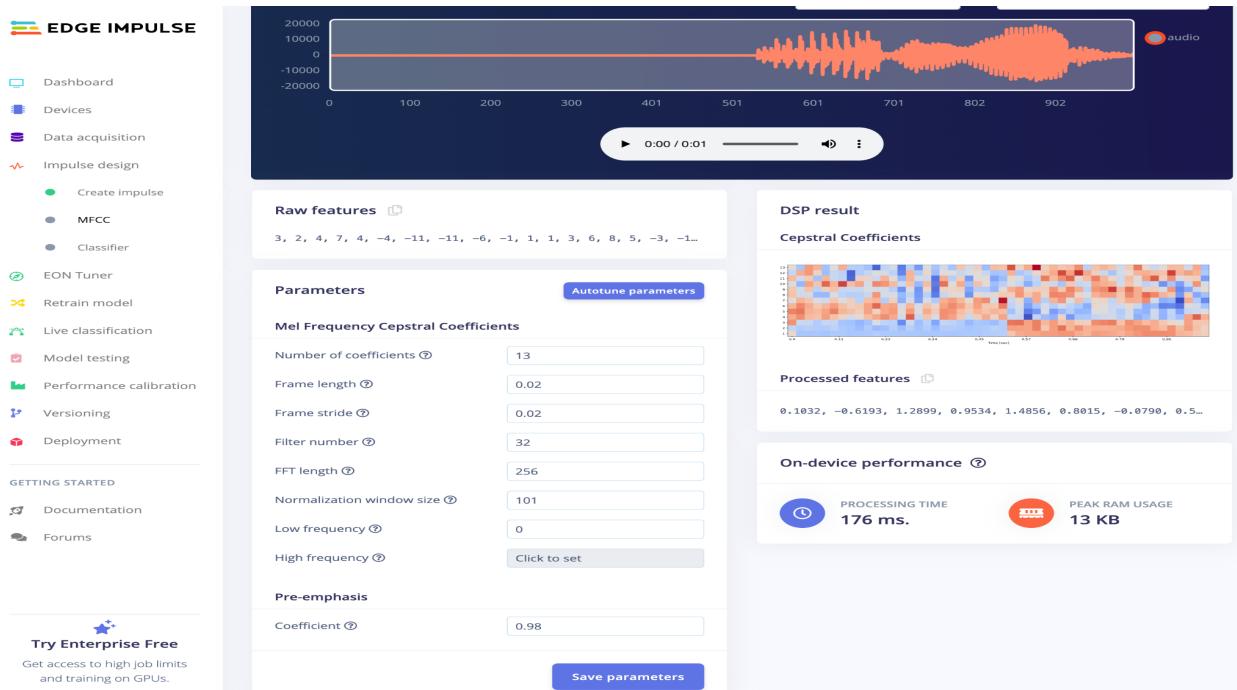
Task 1.2. Creating Impulse (pre-processing / model definition)

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.

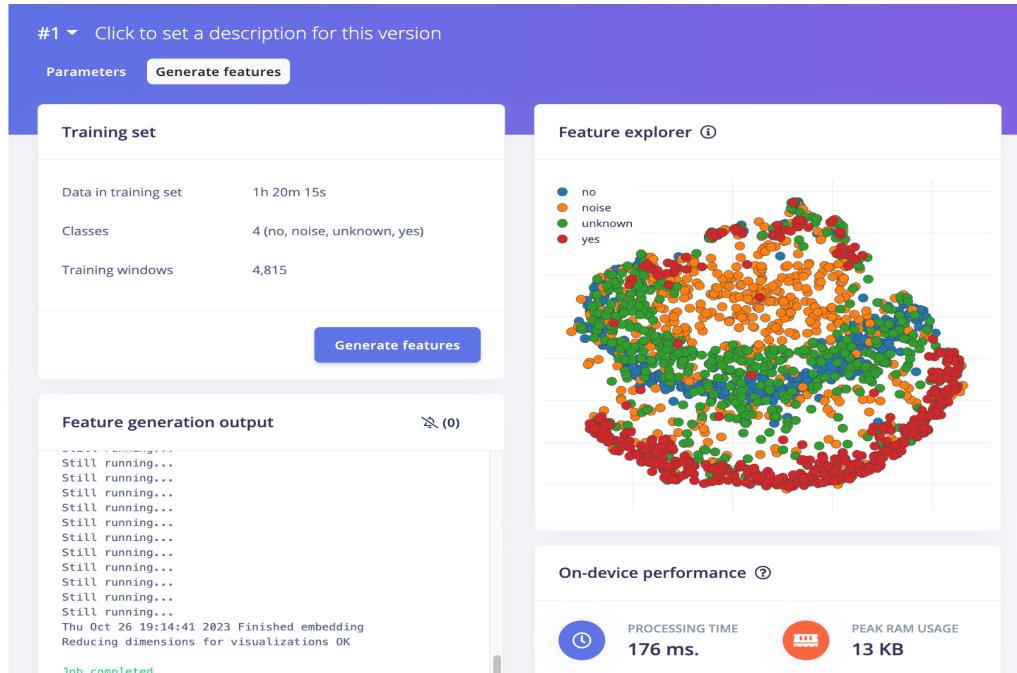
On the Edgelimpulse website, click the ‘Create impulse’ menu on the left.

The screenshot shows the 'Create impulse' configuration page. It features four main blocks: 'Time series data' (red background), 'Audio (MFCC)' (light blue background), 'Classification' (purple background), and 'Output features' (green background). The 'Time series data' block includes settings for Input axes (audio), Window size (1000 ms), Window increase (500 ms), Frequency (Hz) (16000), and Zero-pad data (checked). The 'Audio (MFCC)' block has a Name (MFCC) and Input axes (1). The 'Classification' block has a Name (Classifier) and Input features (MFCC). The 'Output features' block lists 4 categories: no, noise, unknown, yes. At the bottom are buttons for 'Save Impulse' and dashed boxes for 'Add a processing block' and 'Add a learning block'.

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.

The screenshot shows a user interface for building and testing machine learning models. On the left, a sidebar lists various tools and resources under categories like Devices, Data acquisition, Impulse design, EON Tuner, Retrain model, Live classification, Model testing, Performance calibration, Versioning, Deployment, and Getting Started. A prominent 'Try Enterprise Free' button is at the bottom of the sidebar.

The main area is divided into several sections:

- Neural network architecture:** Shows the current model structure:
 - 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
 - Add an extra layer (button)
 - Output layer (4 classes)
- Model:** Displays training statistics:
 - Model version: Quantized (int8)
 - Last training performance (validation set): Accuracy 91.6%, Loss 0.27
 - Confusion matrix (validation set):

	NO	NOISE	UNKNOWN	YES
NO	91.4%	1.2%	5.3%	2.0%
NOISE	0%	96.0%	2.2%	1.8%
UNKNOWN	7.1%	5.9%	83.9%	3.1%
YES	0.4%	0.4%	3.4%	95.8%
F1 SCORE	0.92	0.94	0.86	0.94
 - Data explorer (full training set): A scatter plot showing data points categorized by outcome (no - correct, noise - correct, unknown - correct, yes - correct, no - incorrect, noise - incorrect, unknown - incorrect, yes - incorrect).
 - On-device performance:
 - Inference time: 4 ms.
 - Peak RAM usage: 3.8K
 - Flash usage: 31.2K

Task 1.4. Testing

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

Model testing results



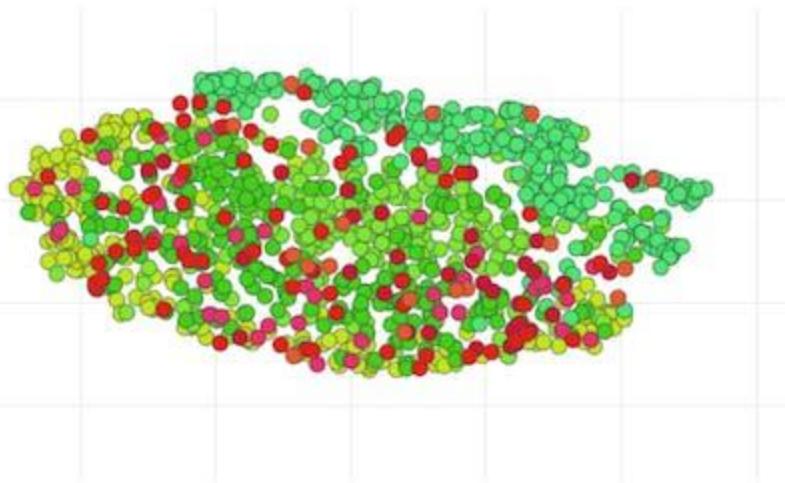
ACCURACY

86.73%

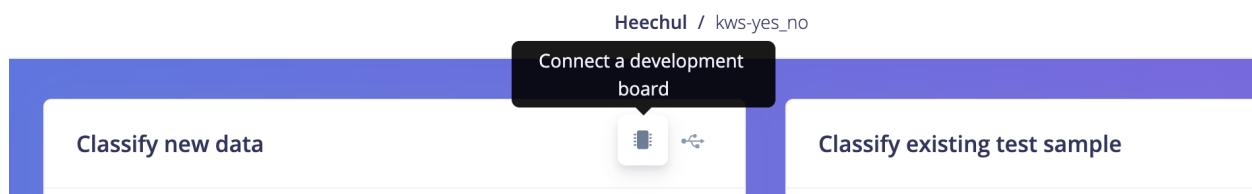
	NO	NOISE	UNKNOWN	YES	UNCERTAIN
NO	86.3%	0.7%	3.9%	1.4%	7.7%
NOISE	0%	88.6%	3.3%	0.7%	7.5%
UNKNOWN	4.4%	2.7%	78.1%	1.7%	13.1%
YES	0.3%	0%	0.7%	93.9%	5.1%
F1 SCORE	0.90	0.92	0.84	0.95	

Feature explorer (?)

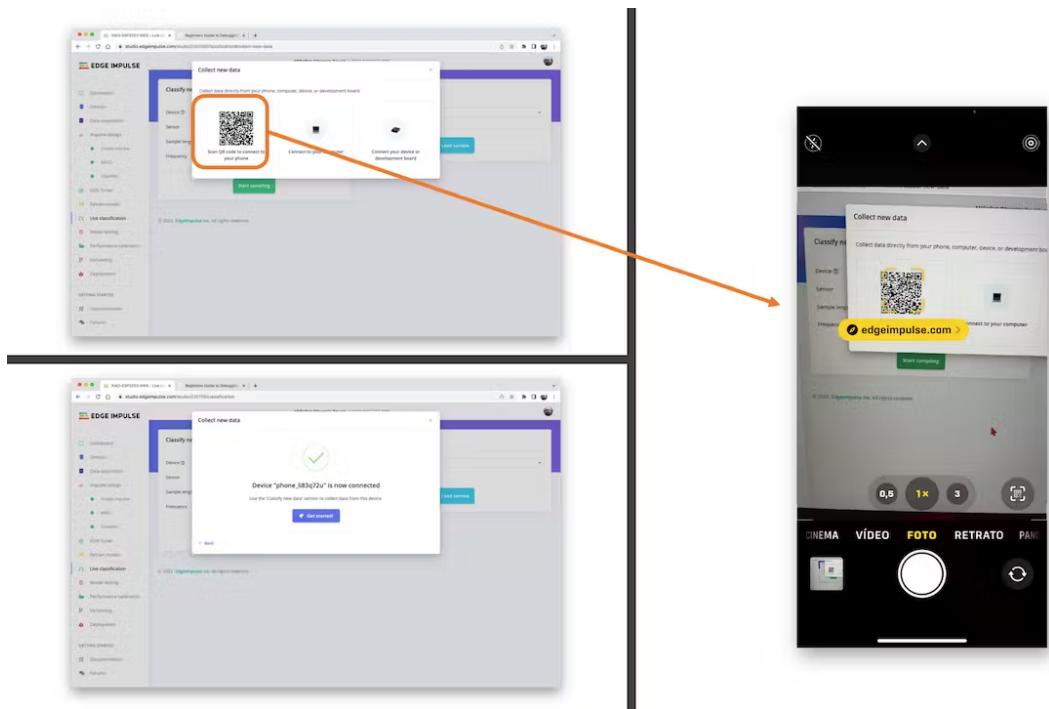
- no - correct
- noise - correct
- unknown - correct
- yes - correct
- no - incorrect
- noise - incorrect
- unknown - incorrect
- yes - incorrect



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.

Configure your deployment

You can deploy your impulse to any device. This makes the model run without an internet connection, minimizes latency, and runs with minimal power consumption. [Read more](#).

SELECTED DEPLOYMENT
Arduino library
An Arduino library with examples that runs on most Arm-based Arduino development boards.

MODEL OPTIMIZATIONS
Model optimizations can increase on-device performance but may reduce accuracy.

Enable EON™ Compiler Same accuracy, up to 50% less memory. [Learn more](#)

Quantized (int8)	MFCC	CLASSIFIER	TOTAL
Selected ✓	LATENCY 675 ms.	6 ms.	681 ms.
	RAM 15.6K	6.0K	15.6K
	FLASH -	49.9K	-
	ACCURACY -	-	-

Unoptimized (float32)	MFCC	CLASSIFIER	TOTAL
Select	LATENCY 675 ms.	31 ms.	706 ms.
	RAM 15.6K	10.5K	15.6K
	FLASH -	53.2K	-
	ACCURACY -	-	-

To compare model accuracy, run model testing. [Run model testing](#)

Estimate for Expresso ESP-EYE (ESP32 240MHz) - Change target

[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.

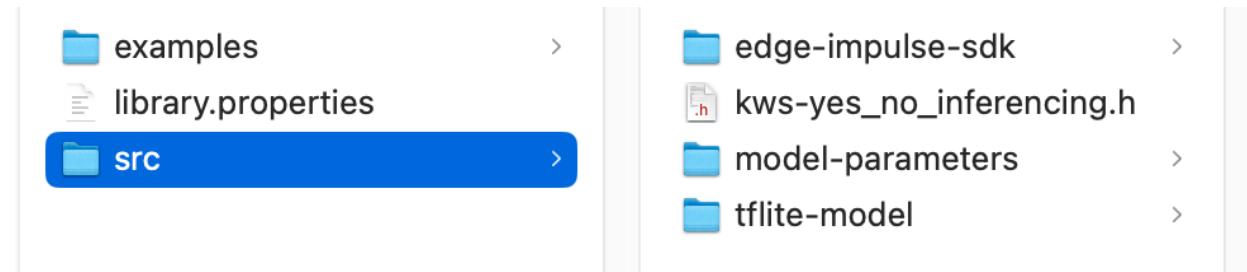
Download the sample project as follows.

```
$ 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 '14-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 '14-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>