ABOUT THIS PROJECT

There is a huge need for inexpensive, easily deployable solutions for COVID-19 and other flu related early detection. Together with the UN, Hackster, Edge Impulse and many others we recently launched the [UN Covid Detect & Protect Challenge](http://undpcovid19.hackster.io/) aiming to create easily deployable solutions for the prevention and detection of flu in developing countries. In this tutorial we show how to use [Edge Impulse](https://www.edgeimpulse.com/) machine learning on an Arduino Nano BLE Sense to detect the presence of coughing in real-time audio. We built a dataset of coughing and background noise samples, and applied a highly optimized TInyML model, to build a cough detection system that runs in real time in under 20 kB of RAM on the Nano BLE Sense. This same approach applies to many other embedded audio pattern matching applications, for example elderly care, safety and machine monitoring. This project and dataset was originally started by Kartik Thakore to help in the COVID-19 effort.

### Getting started

This tutorial has the following requirements:

* Basic understanding of software development and Arduino
* Installed Arduino IDE or CLI
* Android or iOS mobile phone
* [Arduino Nano BLE Sense](https://store.arduino.cc/usa/nano-33-ble-sense) or equivalent Cortex-M4+ board with a microphone (optional)

We will use Edge Impulse, an online development platform for machine learning on edge devices. [Create a free account by signing up here](https://studio.edgeimpulse.com/). Log into your account, and give your new project a name by clicking on the title. We call ours the “Arduino Cough Tutorial”.



### Collecting the dataset

The first step in any machine learning project is to collect a dataset that represents known samples of data that we would like to be able to match on our Arduino device. To get started we have created a small dataset with 10 minutes of audio in two classes, “cough” and “noise”. We will show how to import this dataset into your Edge Impulse project, add your own samples or even start your own dataset from scratch. This dataset is small and has a limited number of coughing and soft background noise samples. Thus the dataset is appropriate only for experimentation, and the model produced in this tutorial is able to differentiate only between quiet background noise and a small range of coughs. We encourage you to extend the dataset with a wider range of coughing, background noise and other classes like human speech to improve performance.

**Note:** Forcing yourself to cough is very hard on your vocal cords, be careful when collecting data and testing!

First download our cough dataset and extract the file on your PC at a location of your choice: <https://cdn.edgeimpulse.com/datasets/cough.zip>

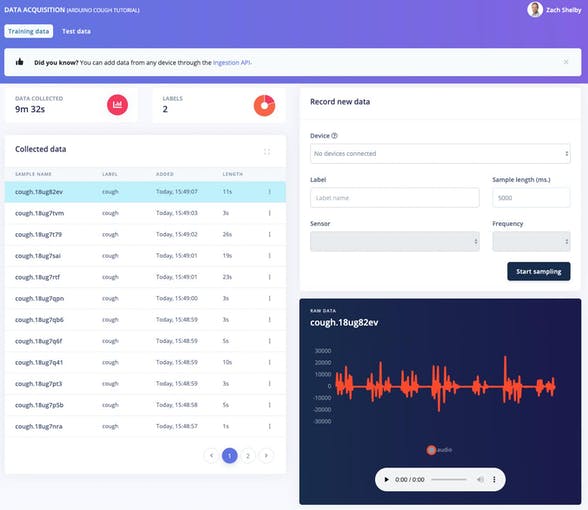
You can import this dataset to your Edge Impulse project using the [Edge Impulse CLI Uploader](https://docs.edgeimpulse.com/docs/cli-uploader). Install the Edge Impulse CLI following these [Installation](https://docs.edgeimpulse.com/docs/cli-installation) instructions.

Open a terminal or command prompt, and navigate to the folder where you extracted the file.

Run:

$ edge-impulse-uploader --clean  
$ edge-impulse-uploader --category training training/\*  
$ edge-impulse-uploader --category testing testing/\*

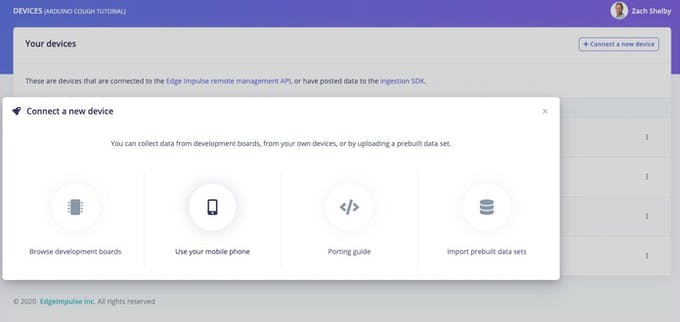
You will be prompted for your Edge Impulse username, password, and the project where you want to add the dataset. The dataset samples will now be visible on the Data acquisition page. By clicking on a sample, we can see what the sample looks like, and hear the audio by clicking on the play button below each graph.

[](javascript:openLightBox('f5d711fdfc',%200);)

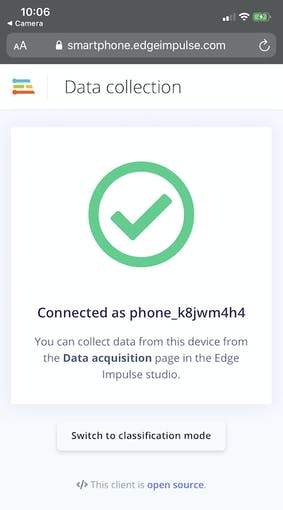
10 minutes of cough and noise data samples are sufficient to get started. You can optionally extend the dataset with your own cough and background noise samples. We can collect new data samples directly from devices from the **Data acquisition** page. Audio samples in WAV format can also be uploaded using the [Edge Impulse CLI Uploader](https://docs.edgeimpulse.com/docs/cli-uploader).

**Importantnote:** Models used in real world applications should be trained and tested with as diverse a dataset as possible. This initial dataset is relatively small, and the model will thus perform inconsistently when exposed to different types of background noise or coughs from different people.

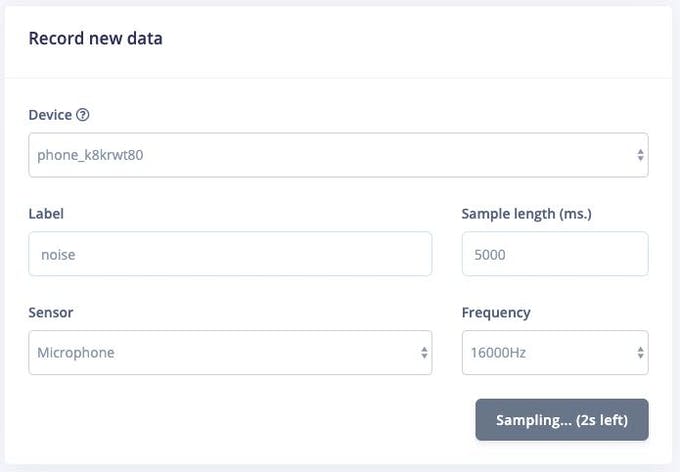
The easiest way to get started is to collect audio data using your mobile phone ([full tutorial here](https://docs.edgeimpulse.com/docs/using-your-mobile-phone)). Go to the **Devices**page, and click on the ‘+ Connect a new device’ button on the upper right. Select ‘Use your mobile phone’. This will produce a unique QR code to open a web application on your phone browser. Take a picture of the QR code, and select to open the link.

[](javascript:openLightBox('02cf123da9',%200);)

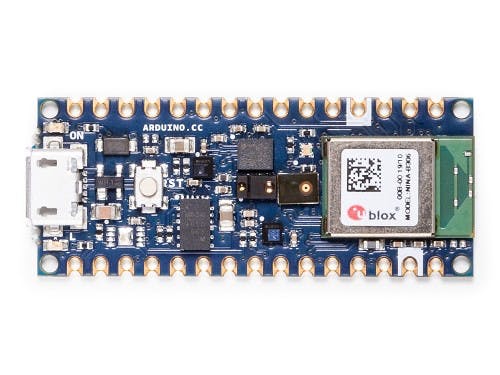
The web application will connect with your Edge Impulse project, and should look like this:

[](javascript:openLightBox('977a9e3b23',%200);)

We can now collect audio data samples directly from the phone from the **Data acquisition** page of Edge Impulse. In the ‘Record new data’ section, type in a label of either ‘cough’ or ‘noise’, make sure ‘Microphone’ is selected as the Sensor, and click ‘Start sampling’. Your phone will now collect an audio sample, and add it to your dataset.

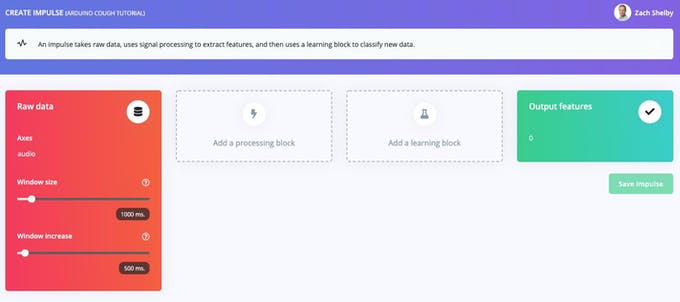
[](javascript:openLightBox('d2d9b7dbb4',%200);)

Collecting audio data directly from the Nano BLE Sense board is also supported. [Follow these instructions](https://docs.edgeimpulse.com/docs/arduino-nano-33-ble-sense) to install the Edge Impulse firmware and daemon. Once the device is connected to Edge Impulse, you can collect data samples just like with your mobile phone above.

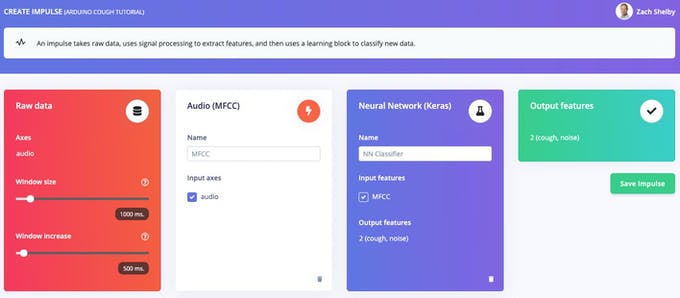
[](javascript:openLightBox('d95ca4550b',%200);)

### Creating your impulse

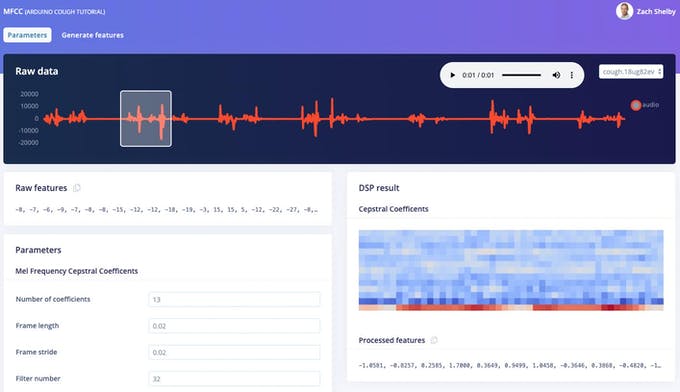
Next we will select signal processing and machine learning blocks, on the **Create impulse** page. The impulse will start out blank, with Raw data and Output feature blocks. Leave the default settings of a 1000 ms Window size and 500 ms Window increase. This means our audio data will be processed 1 s at a time, starting each 0.5 s. Using a small window saves memory on the embedded device, but means that we need sample cough data that without large breaks in between coughs.

[](javascript:openLightBox('fb46a2721d',%200);)

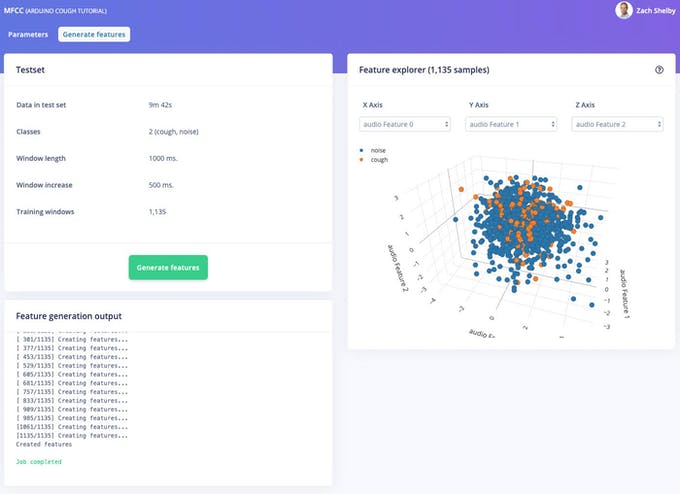
Click on ‘Add a processing block’ and select the **Audio (MFCC)** block. Next click on ‘Add a learning block’ and select the **Neural Network (Keras)** block. Click on ‘Save Impulse’. The audio block will extract a spectrogram for each window of audio, and the neural network block will be trained to classify the spectrogram as either a ‘cough’ or ‘noise’ based on our training dataset. Your resulting impulse will look like this:

[](javascript:openLightBox('a8afc719d3',%200);)

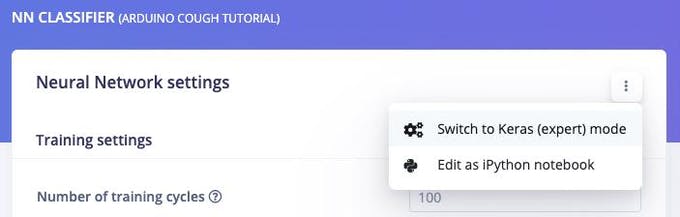
Next we will generate features from the training dataset on the **MFCC** page. This page shows what the extracted spectrogram looks like for each 1 second window from any of the dataset samples. We can leave the parameters to their defaults.

[](javascript:openLightBox('9508fe199a',%200);)

Next click on the ‘Generate features’ button, which then processes the entire training dataset with this processing block. This creates the complete set of features that will be used to train our Neural Network in the next step. Press the ‘Generate features’ button to start the processing, this will take a couple minutes to complete.

[](javascript:openLightBox('f8d80493b1',%200);)

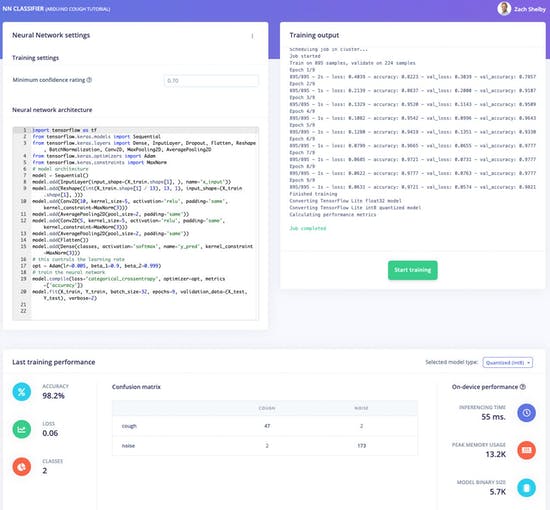
We can now proceed to setup and train our neural network on the **NN Classifier** page. The default neural network works well for continuous sounds like water running. Cough detection is more complicated, so we will configure a richer network using [2D convolution across the spectrogram](https://medium.com/x8-the-ai-community/audio-classification-using-cnn-coding-example-f9cbd272269e) of each window. 2D convolution processes the audio spectrogram in a similar way to image classification. Press the upper right corner of the ‘Neural Network settings’ section, and select ‘Switch to Keras (expert) mode’.

[](javascript:openLightBox('63d72ba6c5',%200);)

Replace the ‘Neural network architecture’ definition with the following code and set the ‘Minimum confidence rating’ setting to ‘0.70’. Then proceed to click the ‘Start training’ button. Training will take a few seconds.

import tensorflow as tf  
from tensorflow.keras.models import Sequential  
from tensorflow.keras.layers import Dense, InputLayer, Dropout, Flatten, Reshape, BatchNormalization, Conv2D, MaxPooling2D, AveragePooling2D  
from tensorflow.keras.optimizers import Adam  
from tensorflow.keras.constraints import MaxNorm  
# model architecture  
model = Sequential()  
model.add(InputLayer(input\_shape=(X\_train.shape[1], ), name='x\_input'))  
model.add(Reshape((int(X\_train.shape[1] / 13), 13, 1), input\_shape=(X\_train.shape[1], )))  
model.add(Conv2D(10, kernel\_size=5, activation='relu', padding='same', kernel\_constraint=MaxNorm(3)))  
model.add(AveragePooling2D(pool\_size=2, padding='same'))  
model.add(Conv2D(5, kernel\_size=5, activation='relu', padding='same', kernel\_constraint=MaxNorm(3)))  
model.add(AveragePooling2D(pool\_size=2, padding='same'))  
model.add(Flatten())  
model.add(Dense(classes, activation='softmax', name='y\_pred', kernel\_constraint=MaxNorm(3)))  
# this controls the learning rate  
opt = Adam(lr=0.005, beta\_1=0.9, beta\_2=0.999)  
# train the neural network  
model.compile(loss='categorical\_crossentropy', optimizer=opt, metrics=['accuracy'])  
model.fit(X\_train, Y\_train, batch\_size=32, epochs=9, validation\_data=(X\_test, Y\_test), verbose=2)

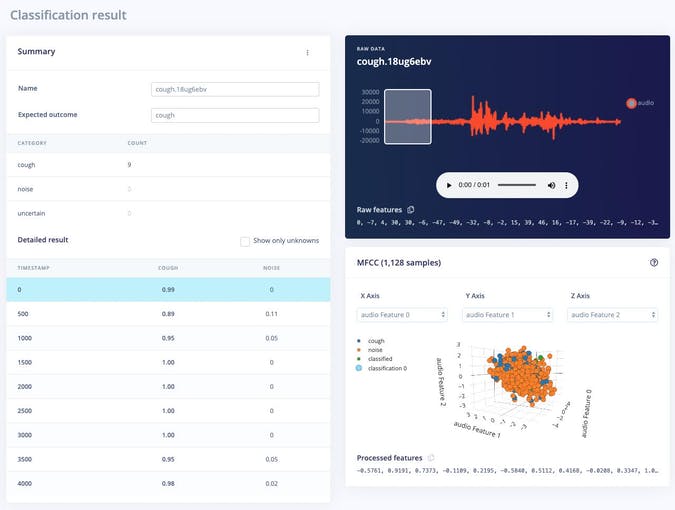
The page will show the training performance and on-device performance, which should like this depending on your dataset:

[](javascript:openLightBox('07f2e060cc',%200);)

Our Arduino cough detection algorithm is now ready to try out!

### Training and testing

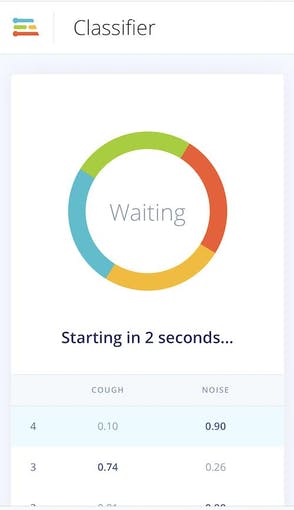
The **Live classification** page allows us to test the algorithm both with the existing testing data that came with the dataset, or by streaming audio data from your mobile phone or Arduino device. We can start with a simple test by choosing any of the test samples, and pressing ‘Load sample’. This will classify the test sample and show the results:

[](javascript:openLightBox('575215eb5e',%200);)

We can also test the algorithm with live data. Start with your mobile phone by refreshing the browser page on your phone we opened earlier. Then select your device in the ‘Classify new data’ section and press ‘Start sampling’. You can similarly stream audio samples from your Nano BLE Sense when connected to the project via the edge-impulse-daemon as in the data collection step.

### Deployment

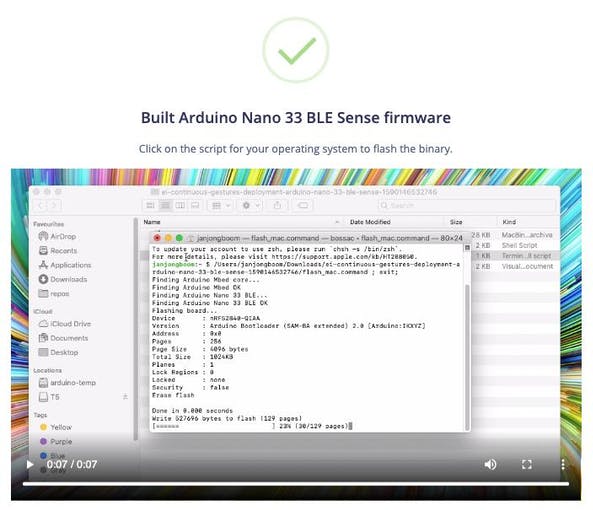
We can easily deploy our cough detection algorithm to the mobile phone. Go to the browser window on your phone and refresh, then press the ‘Switch to classification mode’ button. This will automatically build the project into a WebAssembly package and execute it on your phone continuously (no cloud required after that, even go to airplane mode!)

[](javascript:openLightBox('58f1cfbff9',%200);)

Next we can deploy the algorithm to the Nano BLE Sense by going to the **Deployment** page. Select the ‘Arduino Nano 33 BLE Sense’ under ‘Build firmware’ and then click ‘Build’.

[](javascript:openLightBox('35de22b785',%200);)

This will build a complete firmware for the Nano BLE Sense including your latest algorithm. Follow the instructions on the screen to flash your Arduino board with the binary.

[](javascript:openLightBox('d951444e9a',%200);)

Once the Arduino is flashed, we can open a serial port to the device while it is plugged into USB at 115, 200 baud. Once the serial port is open, press enter to get a prompt and then:

> AT+RUNIMPULSE  
Inferencing settings:  
 Interval: 0.06 ms.  
 Frame size: 16000  
 Sample length: 1000 ms.  
 No. of classes: 2  
Starting inferencing, press 'b' to break  
Recording...  
Recording done  
Predictions (DSP: 495 ms., Classification: 84 ms., Anomaly: 0 ms.):   
 cough: 0.01562  
 noise: 0.98438  
Starting inferencing in 2 seconds...  
Recording...  
Recording done  
Predictions (DSP: 495 ms., Classification: 84 ms., Anomaly: 0 ms.):  
 cough: 0.01562  
 noise: 0.98438  
Starting inferencing in 2 seconds...  
Recording...  
Recording done  
Predictions (DSP: 495 ms., Classification: 84 ms., Anomaly: 0 ms.):  
 cough: 0.86719  
 noise: 0.13281  
Starting inferencing in 2 seconds...  
Recording...  
Recording done  
Predictions (DSP: 495 ms., Classification: 84 ms., Anomaly: 0 ms.):  
 cough: 0.01562  
 noise: 0.98438

### Future work

The sky is the limit with TinyML, sensors and Edge Impulse on Arduino, here are some ideas for further work:

* Extend the default dataset with your own coughing and background sounds, remember to retrain periodically and test. You can set up unit tests under the Testing page to ensure that the model is still working as it is extended.
* Add a new class and data for human sounds that are not coughing, like background speech, yawning etc.
* Start with a new dataset, collecting audio samples to detect something new. Hint: You can upload just the noise class data from this dataset to get started!
* [From these instructions deploy to Arduino Library](https://docs.edgeimpulse.com/docs/running-your-impulse-arduino) as part of an Arduino Sketch to show cough detection using the LED or a display
* Make use of other sensors like the 3-axis accelerometer of the Nano BLE Sense by [following this tutorial](https://docs.edgeimpulse.com/docs/continuous-motion-recognition).