Table Tennis Deep Trainer

Debbie Liske
Wearable Devices - Harvard DGMD S-14

Introduction and Motivation

This project proposes a Table Tennis Trainer that provides feedback to improve stroke technique

Uses Deep Learning to train a model to recognize the problem with the swing and inform the player of their tendency and indicate whether a swing was proper or one of two forms of improper technique - thus utilizing a three class classifier to help the player improve skills.

The purpose of this project is to design a product that can help a player learn the proper technique for a forehand drive, but the design and method can be applied to any stroke or serve.

The advantage of this method is that a table (or ball) is not required if the model is trained by an experienced player with the goal and characteristics of the student in mind. This enables the player to learn the basic movements of a skill at the cognitive stage of learning and is also known as “shadow practicing”.

This project is focused on classification of the angle of the racket in the player’s hand. If the face of the racket is too vertical, the racket is considered to be too open and thus the player should decrease the vertical angle by aiming the tip of the racket forward and down. If, on the other hand, the racket face is too horizontal, the racket is considered too closed and the face of the racket should be angled more vertically. During training, the device will classify each swing and record the results.

The ultimate goal of the project is unique in that the feedback is an immediate audio response as to whether the swing was too open, too closed, or proper mechanics. There are two options for training: 1) When the player successfully swings a forehand drive of proper technique, the player hears a beep reinforcing the swing. If the classifier does not classify a good swing, no sound is heard. This is comparable to “clicker training” used with animals in which positive reinforcement is given upon success, and no sound is made otherwise. 2) An extension to this is to recommend to open or close the angle of the racket if the swing is poor based on text to speech conversion, using a human voice. Given time constraints, only the classification piece has been implemented in phase one.

Process - Part 1

Acquire equipment

- STM32 IOT Discovery Board
- Peripherals

Connect board to Edge Impulse using USB

- Power
- Acquire data

Train (Collect data) for:

- Correct technique
- Open racket face
- Closed racket face

Generate features

Build Neural Network architecture and determine hyperparameters

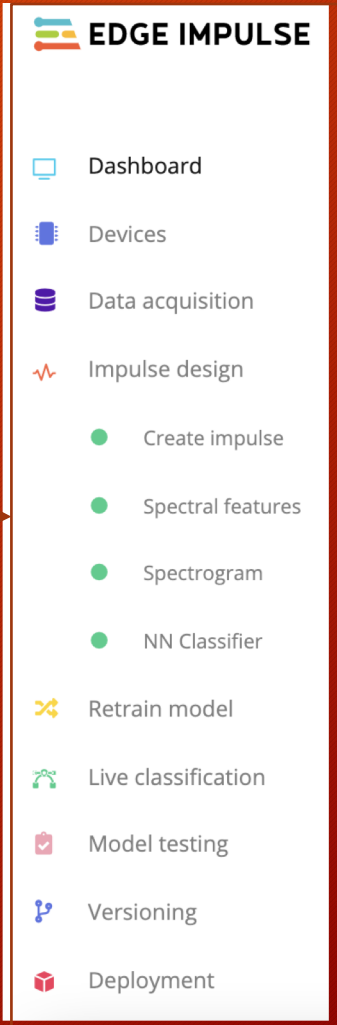
Train model using train-test split of the acquired data to create a 3-class neural network classifier

Assess model performance with accuracy metrics and confusion matrix

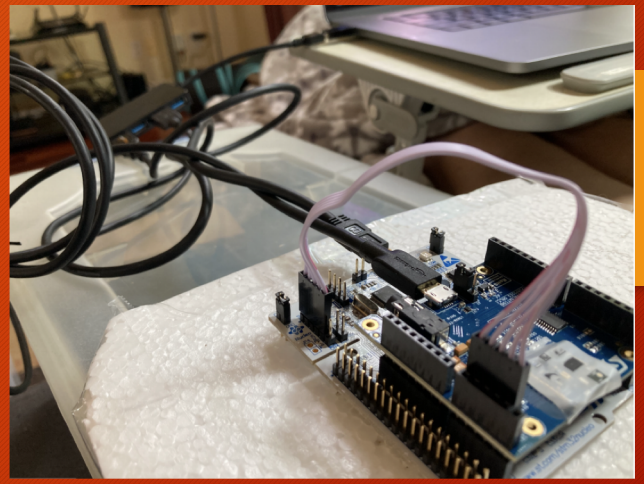
Perform Live Classification to demonstrate how it works while powered and sending data to Edge Impulse

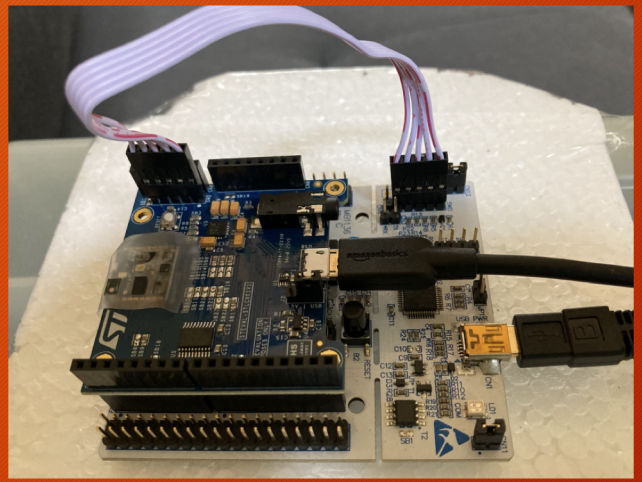
Input Data

- Attach Discovery Kit to Edge Impulse
- Receive and Label data from the sensors
- Determine components of model
- Generate spectral features
- Generate accY spectrogram features
- Design Neural Network and train the model
- Test the model classification skill
- Review all / any test result(s)
- Publish the model with current version
- Flash the board with binary




USB3.0 HUB



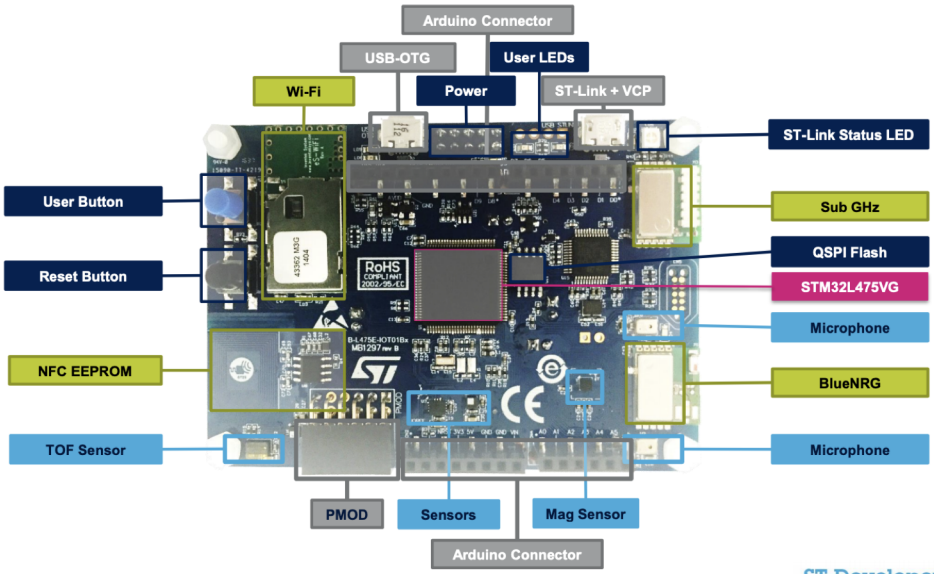


data_file6.txt

```
TimeStamp: 00:08:21.94

ACC_X: 20, ACC_Y: -18, ACC_Z: 1009

GYR_X: 2240, GYR_Y: -3150, GYR_Z: -770

MAG_X: -288, MAG_Y: -108, MAG_Z: -222

PRESS: 1007.48

TimeStamp: 00:08:22.94

ACC_X: 17, ACC_Y: -17, ACC_Z: 1013

GYR_X: 2170, GYR_Y: -3010, GYR_Z: -840

MAG_X: -300, MAG_Y: -112, MAG_Z: -213

PRESS: 1007.48

TimeStamp: 00:08:23.94

ACC_X: 19, ACC_Y: -17, ACC_Z: 1011

GYR_X: 2240, GYR_Y: -3080, GYR_Z: -770

MAG_X: -295, MAG_Y: -109, MAG_Z: -219

PRESS: 1007.48
```

```python
import serial
import time
import datetime
portname = '/dev/cu.usbmodemFFFFFFFEFFFF1'
ser = serial.Serial(portname, 9600)
start_time = time.time()
sampleTime = 60 #sec
f    = open("/Users/debbieliske/Downloads/data_file6.txt", "w")

while (time.time()-start_time) <= sampleTime:
    s = ser.readline()
    line = s.decode('utf-8').replace('\r\n','')
    time.sleep(.1)
    f.write(line+"\r\n")     # Appends output to file

ser.flush()
ser.close()
f.close()
```

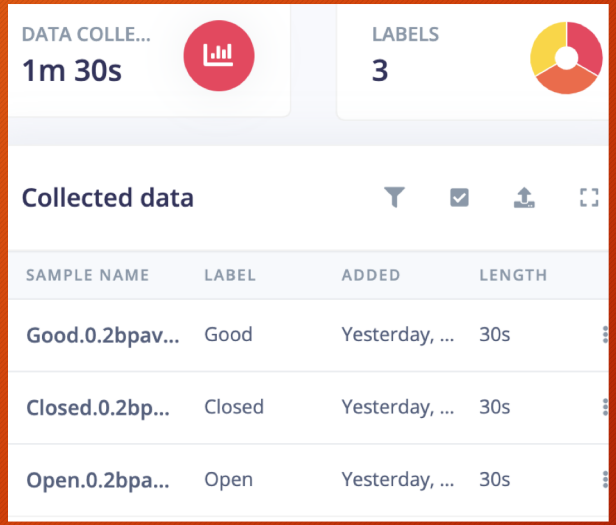
STM32L475 Discovery IoT Node

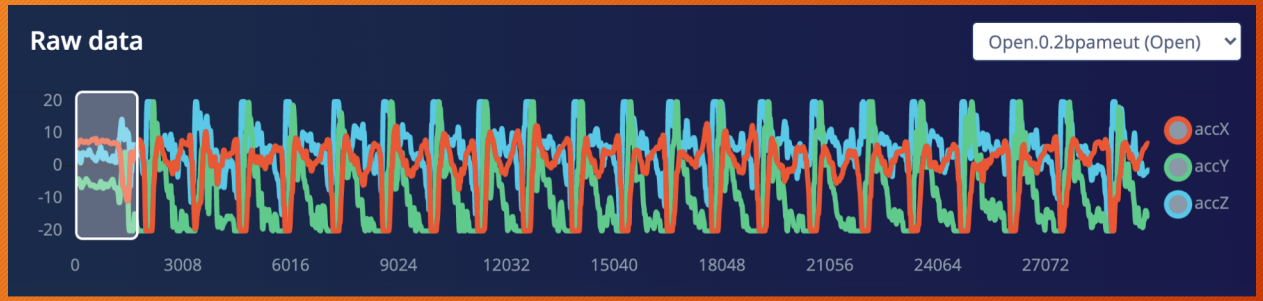


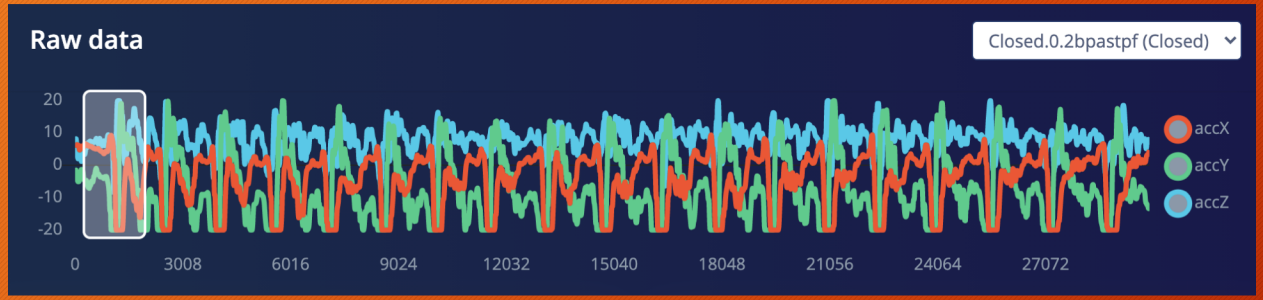
life.augmented

ST Developers Conference


EDGE IMPULSE
Project info
Keys
Export
Debbie Liske
Dashboard
Devices
Data acquisition
Debbie Liske / Table Tennis Deep Trainer
This is your Edge Impulse project. From here you acquire new training data,
design impulses and train models.


DATA COLLE...
1m 30s
LABELS
3
Collected data
SAMPLE NAME
LABEL
ADDED
LENGTH
Good.0.2bpav...
Good
Yesterday, ...
30s
Closed.0.2bp...
Closed
Yesterday, ...
30s
Open.0.2bpa...
Open
Yesterday, ...
30s





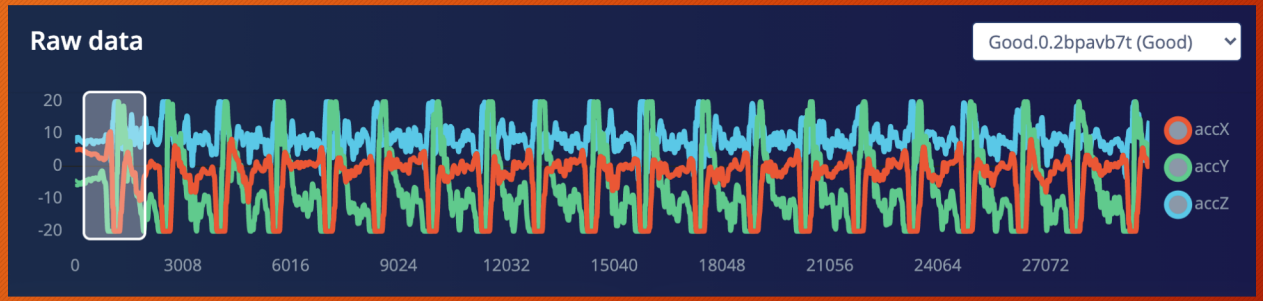
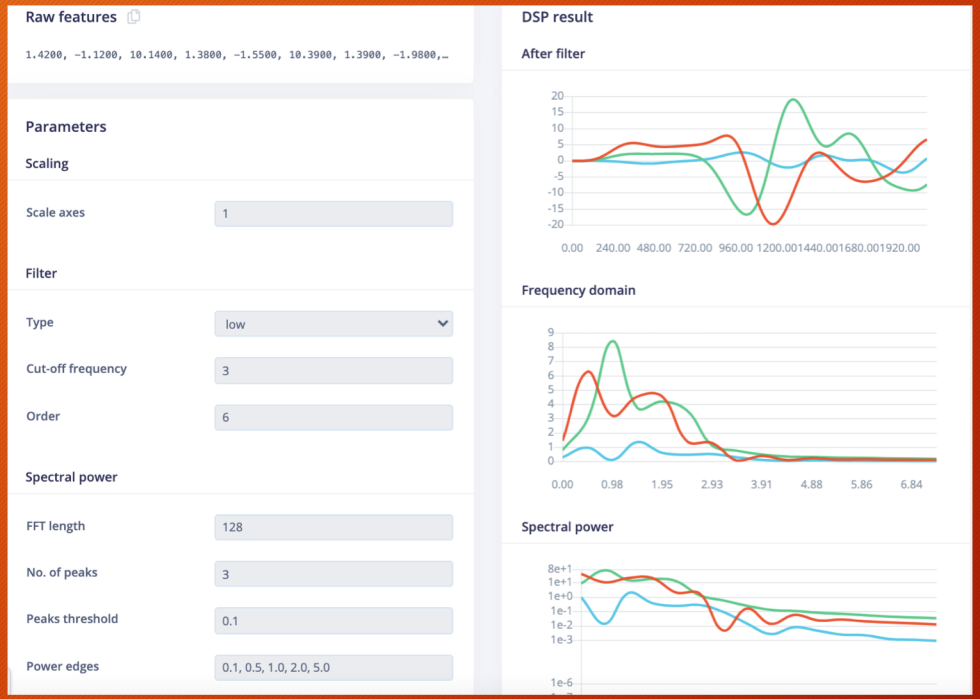


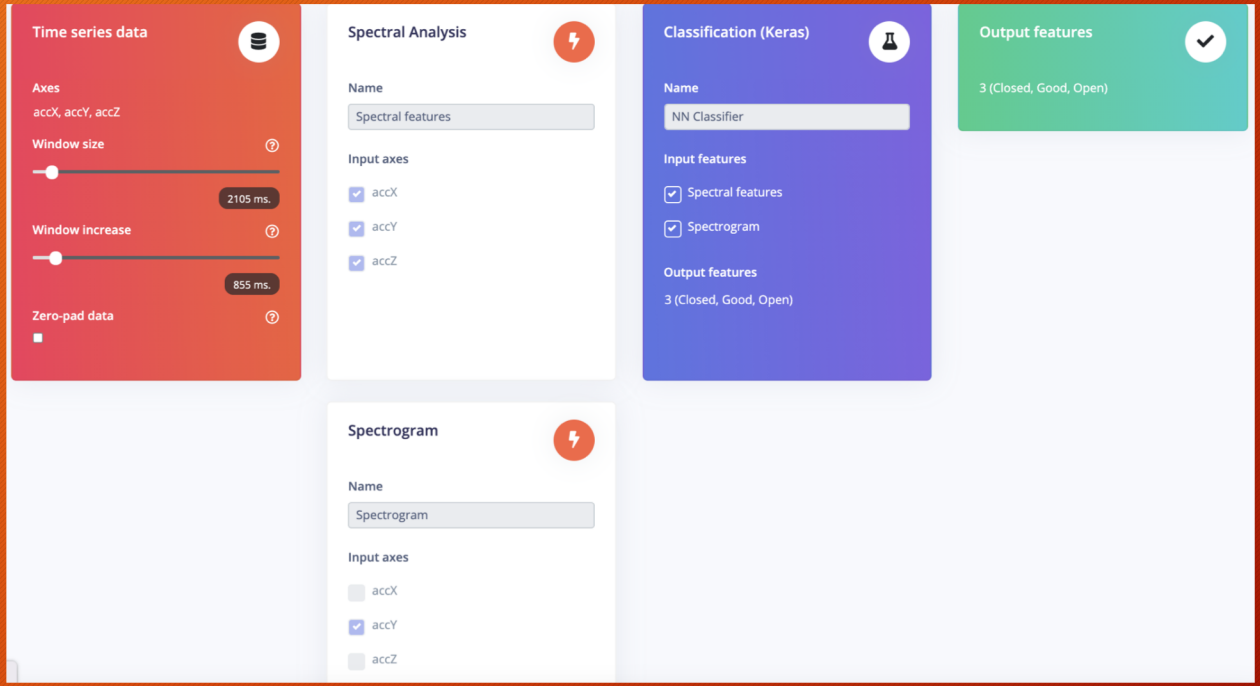
Figure1: Open (Slightly Improper) Technique Accelerometer Data

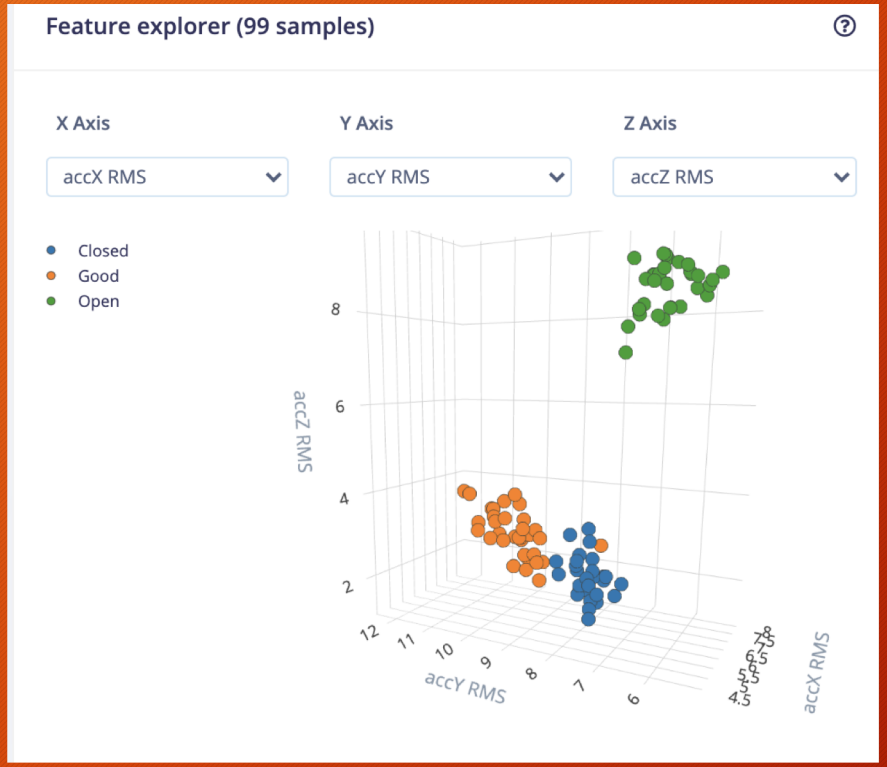
Figure2: Closed (Slightly Improper) Technique Accelerometer Data

Figure3: Proper Technique Accelerometer Data

Signal Feature Extraction




Time series data
Axes
accX, accY, accZ
Window size
2105 ms.
Window increase
855 ms.
Zero-pad data
Spectral Analysis
Name
Spectral features
Input axes
accX
accY
accZ
Classification (Keras)
Name
NN Classifier
Input features
Spectral features
Spectrogram
Output features
3 (Closed, Good, Open)
Output features
3 (Closed, Good, Open)
Spectrogram
Name
Spectrogram
Input axes
accX
accY
accZ


Feature explorer (99 samples)
X Axis
accX RMS
Y Axis
accY RMS
Z Axis
accZ RMS
Closed
Good
Open
accZ RMS
8
6
4
2
12
11
10
9
8
7
6
accY RMS
accX RMS

Neural Network settings

Training settings

Number of training cycles: 50

Learning rate: 0.0005

Minimum confidence rating: 0.60

Neural network architecture

Input layer (8,483 features)

Dense layer (20 neurons)

Dropout (rate 0.1)

Dense layer (10 neurons)

Dense layer (10 neurons)

Output layer (3 features)

Model

Model version: Quantized (int8)

Last training performance (validation set)

ACCURACY 95.0%

LOSS 0.21

Confusion matrix (validation set)

	CLOSED	GOOD	OPEN
CLOSED	100%	0%	0%
GOOD	11.1%	88.9%	0%
OPEN	0%	0%	100%
F1 SCORE	0.91	0.94	1.00

Feature explorer (full training set)

The feature explorer is only supported when you have a single DSP block.

On-device performance

INFERENCING TIME 25 ms.

PEAK RAM USAGE 10.3K

FLASH USAGE 183.2K

Live Classification

Summary

Name: testing.0.2at5n06k

Expected outcome: testing

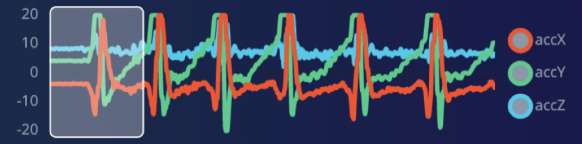
CATEGORY	COUNT
Closed	3
Good	4
Open	0
uncertain	3

Detailed result — Show only unknowns

TIMESTAMP	CLOSED	GOOD	OPEN
0	0.68	0.29	0.03
848	0.72	0.27	0
1696	0.69	0.27	0.04
2544	0.27	0.73	0
3392	0.40	0.59	0

RAW DATA

testing.0.2at5n06k

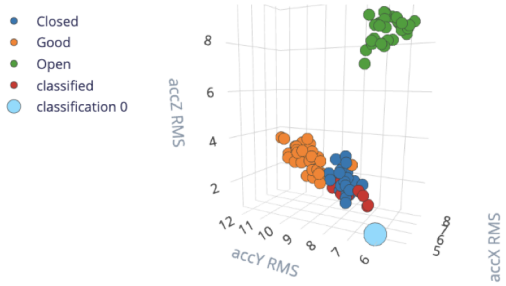


Raw features

```
-3.7400, 4.1600, 8.4400, -3.7900, 4.1600, 8.4600, -3.7600, 4.2200, 8…
```

Spectral features (109 samples)

X Axis: accX RMS | Y Axis: accY RMS | Z Axis: accZ RMS



Processed features

Models Ready For Deployment

Spectral features training data	NPY file
Spectral features training labels	NPY file
Spectral features testing data	NPY file
Spectral features testing labels	NPY file
Spectrogram training data	NPY file
Spectrogram training labels	NPY file
Spectrogram testing data	NPY file
Spectrogram testing labels	NPY file
NN Classifier model	TensorFlow Lite (float32)
NN Classifier model	TensorFlow Lite (int8 quantized)
NN Classifier model	TensorFlow Lite (int8 quantized with float32 input and ...
NN Classifier model	TensorFlow SavedModel

Select optimizations *(optional)*

Model optimizations can increase on-device performance but may reduce accuracy. Click below to analyze optimizations and see the recommended choices for your target. Or, just click Build to use the currently selected options.

Enable EON™ Compiler
Same accuracy, up to 50% less memory.
Open source.

Build output

```
Creating job... OK (ID: 1193595)

Writing templates OK

Scheduling job in cluster...
Job started
Creating archive...
Creating archive OK
```

Built STM32Cube.MX CMSIS-PACK

Learn how to integrate this library

Process - Part 2 (to be completed at a later date)

Acquire equipment

- Speaker and Audio Amplifier
- 3.7 - 5V Converter
- Arduino and supplies

Power the STM32 IoT Discovery Board with battery

(Use Wi-Fi or Bluetooth to collect data)

Export model from Edge Impulse

Open model / code in IDE

Edit C/C++ code to trigger a sound on a successful (“good” class) forehand drive.

Thank You

See main documentation for more details.
Here is the associated video:
https://www.youtube.com/watch?v=AxE7QkXZY3I