Table Tennis Deep Trainer

DGMD S-14 Wearable Devices and Computer VIsion (Summer 2021)
Debbie Liske

Abstract

Wearable devices have become an important technique to capture data for activity recognition, and advanced sports programs are beginning to incorporate sophisticated methods of capturing this data, analyzing the results and building models to aid in the development of athletic skills training. Table tennis is a complex technical sport that requires high levels of skill and concentration that can only be developed through practice and repetition. This project proposes a Table Tennis Trainer that provides feedback to improve stroke technique. There are many strokes in table tennis and numerous styles of play. This project focuses on the Forehand Drive. There is almost an infinite combination of things at play when swinging a racket, but this trainer is tuned to the individual and the specific conditions. One indicator of success of a forehand drive is the angle of the racket in the player's hand. The Table Tennis Deep Trainer 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.

Introduction

Motivation

In table tennis there are many styles of play such as the offensive attacker or the defensive blocker, and many types of strokes such as the drive, loop, or push. There are different ways to hold the racket such as shake hand or pen hold, and different types of rackets with various blades, rubber, and pips. There are of course forehand and backhand swings and different tables with different bounces. Even the size of the ball has changed over time as many years ago it was smaller, but at a professional level, the game was not watchable as the ball was moving too quickly. With so many styles, techniques, methods of play, strokes, etc., it is nearly impossible to create a trainer that works for everyone. However, with the advancement of wearable technology and artificial intelligence, devices can be trained and tuned for specific individuals as well as the general public.

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.

Related Work

STMicroelectronics has a Tennis swing classifier created by B. Lam and G. Schreiber that is intended to help users improve their tennis swing technique by classifying ideal and deficient tennis swings via accelerometer and gyroscope sensor data. [1]. P. Blank created a device called Smart Racket that is specifically designed for table tennis and is designed as a real-time feedback device for analysis but was a thesis project and thus had ample time to consider various investigations and impacts. [2]

System Architecture

This section includes a description of the Software & Developing Tools including (software, the laptop setup and the sensor/hardware information.)

Initial development environment:
A MacBook Pro was used along with the STM32CubeIDE to perform initial data collection using the STM32 Nucleo Development board with the SensorTile sensors. The SensorTile Development Kit is the STEVAL-STLKT01V1 and includes the SensorTIle, cradle expansion board and programming table to connect the cradle to the Nucleo development board.

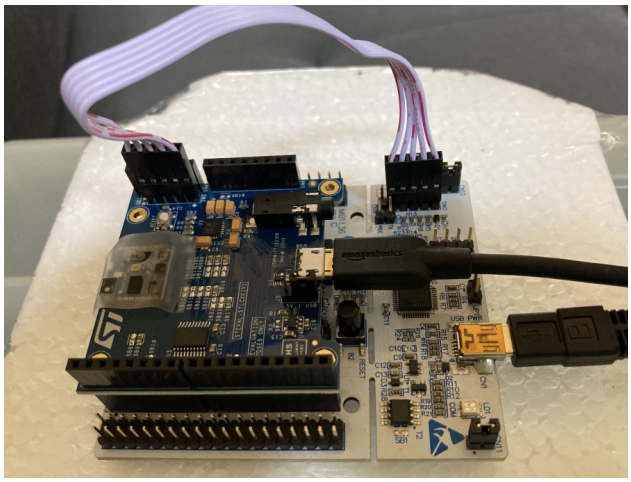


Figure: SensorTile, SensorTile Cradle Expansion, and Nucleo Development Boards

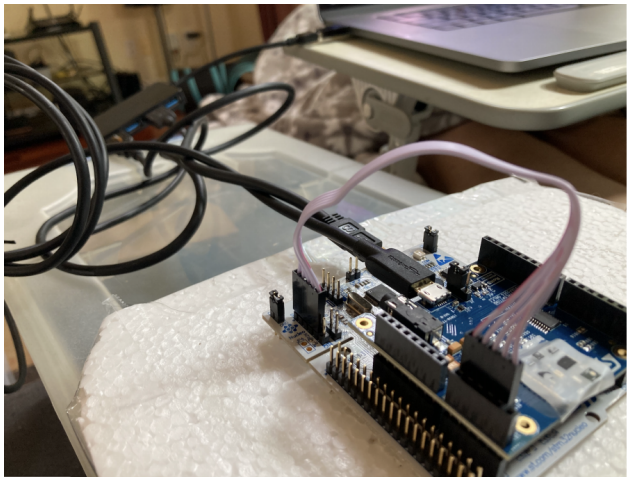


Figure: SensorTile board connect via USB to Macbook

The board were connect to the Mac using two USB cables and a USB Hub:

Initial Sensing

During development, although not all included in the final product, the following sensors were available: Accelerometer, Gyroscope, Humidity, Magnetometer, Pressure, Temperature. The STM32CubeIDE was used to build and run code for the SensorTile boards and via the USB serial port, data was captured via Python code and loaded into a .csv file using the code shown below. Output from the .csv file was then read into a dataframe and analysed.

```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()
```

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
```

Figure: Python code and collected data

The analysis using this code and data is not shown here because the data captured by Edge Impulse was the final data used in the project.

Final System Architecture

After initial analysis was performed, a decision was made to utilize the STM32 IOT Discovery Kit: B-L475E-IOT01A1 to capture data using the Edge Impulse online development platform to capture the raw data. Below is the description of the Discovery board.

Hardware

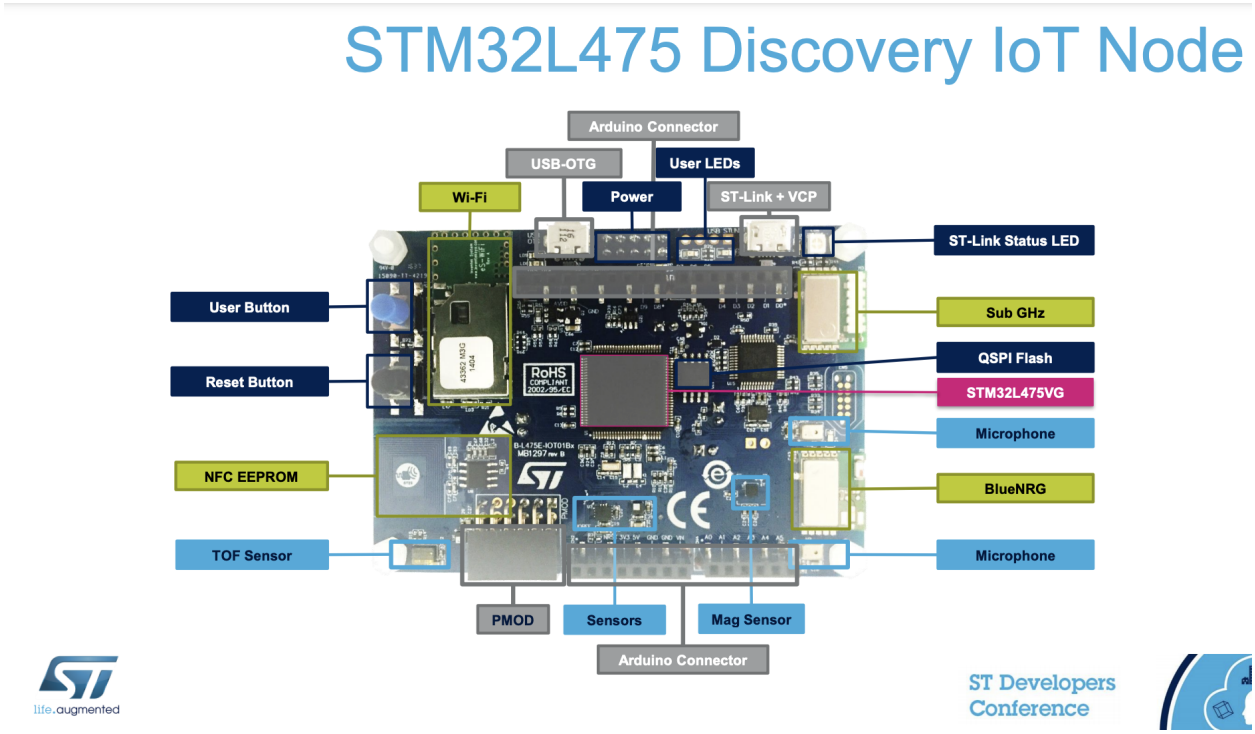
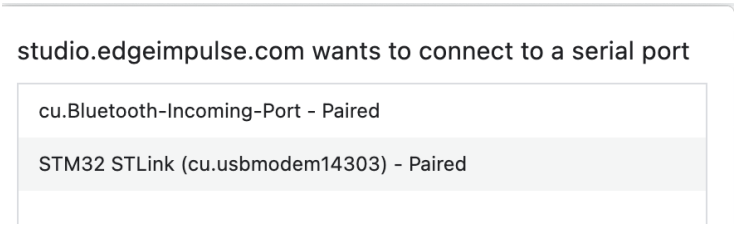


Figure: B-L475E-IOT01A1 IoT Discovery Kit

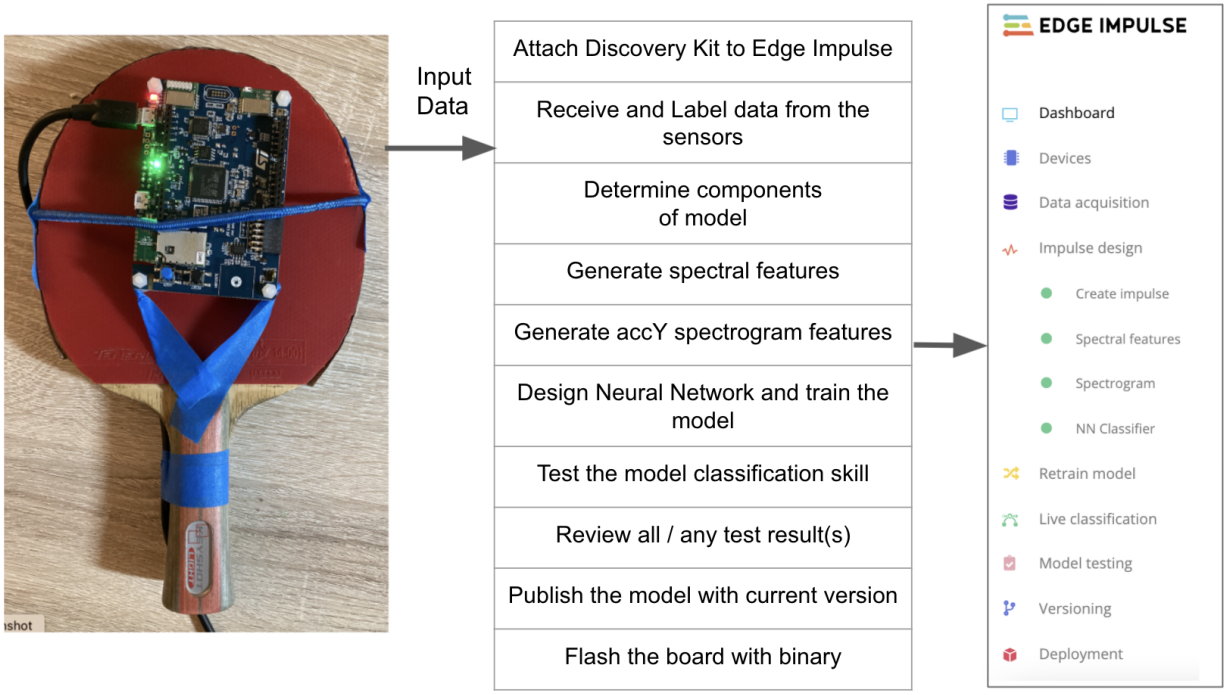
Development Platform

Edge Impulse is an online development platform for machine learning on edge devices.

By connecting the Discovery Kit to the Mac using USB, I was then able to select the STM32 that was paired with Edge Impulse via the STLink usb modem connection.

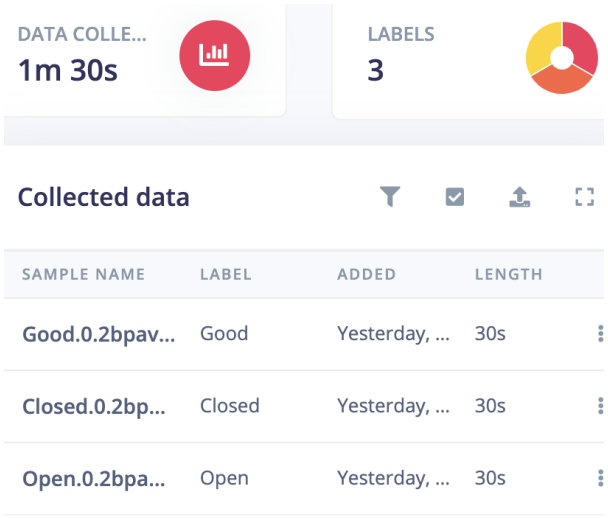


Architecture Diagram



System Sensor Data Acquisition (Data Collection)

During data collection, 30 seconds of forehand drives were performed with the STM32 IoT Discovery Kit for ***each*** class. The device was attached to a table tennis racket and powered by a USB cable which was also used to transfer data to the computer. The results were collected in Edge Impulse and labeled as either *good, closed, or open* as described in the introductory paragraphs of this document.



Data Analysis

Proper strokes are portrayed with the following waveforms for the x, y, and z axes from the accelerometer. Looking closely at the images, the differences are

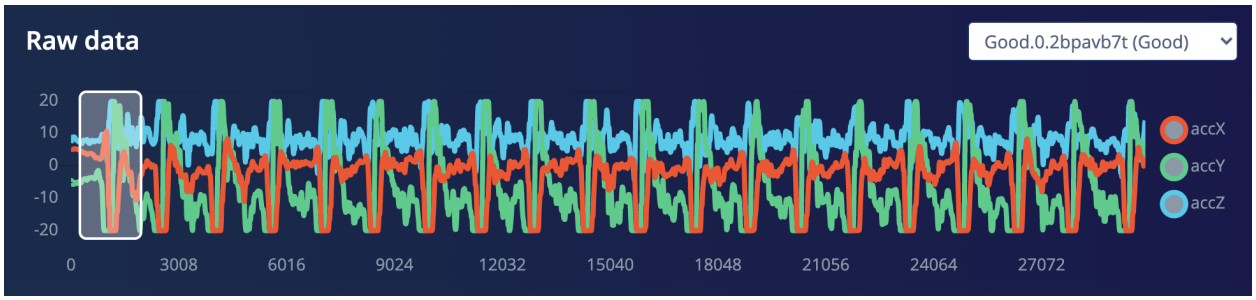


Figure: Proper Technique Accelerometer Data

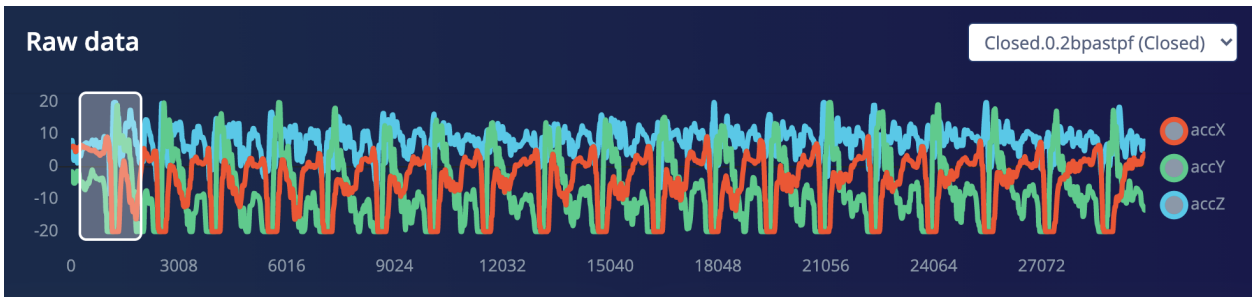


Figure: Closed (Slightly Improper) Technique Accelerometer Data

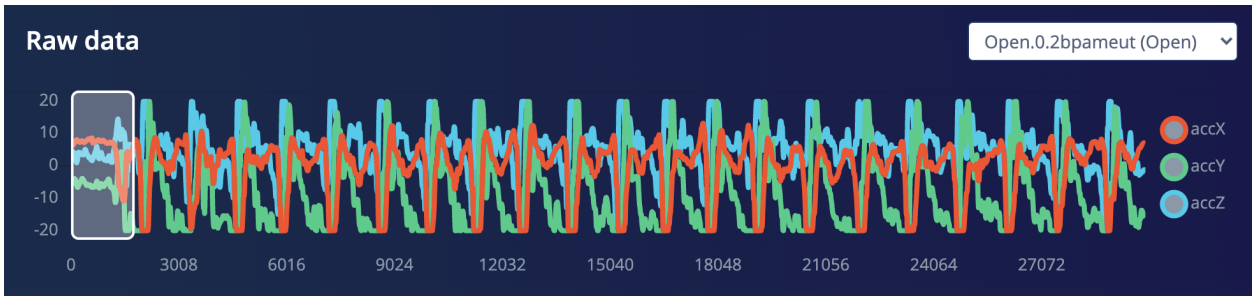


Figure: Open (Slightly Improper) Technique Accelerometer Data

Signal Feature Extraction

Edge Impulse uses the concept of an “Impulse” which is a concept that takes raw data, uses signal processing to extract features, and then uses a learning block to classify new data.

Signal Features

Spectral Features:

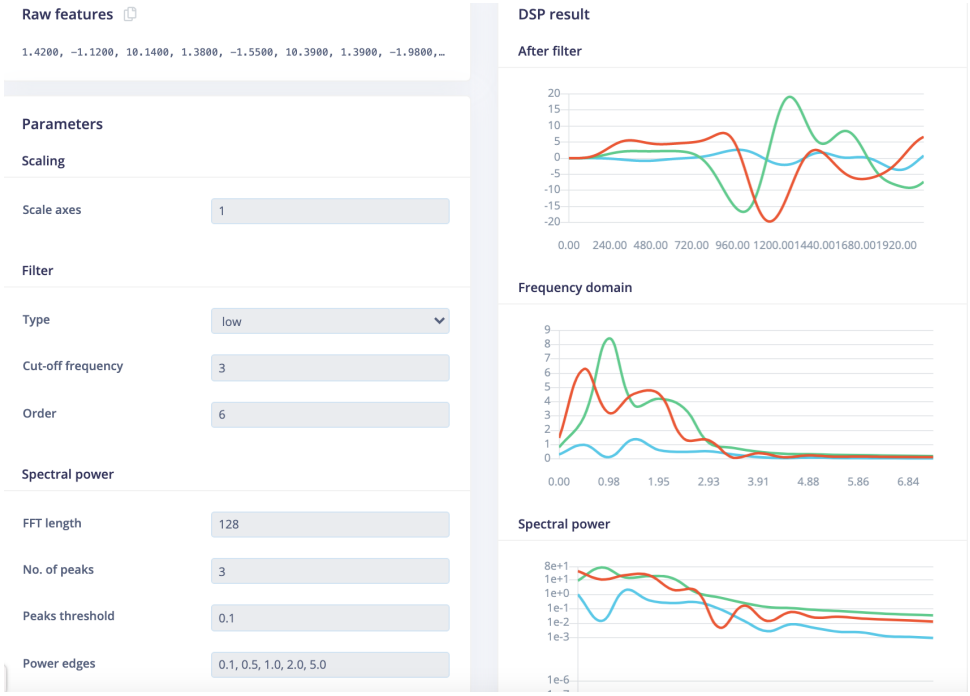


Figure: Spectral Features

In addition to the above features, a Spectrogram of the Y acceleration axis was utilized to help with the classification of the good and closed classes which were closer in 3D space than the open class.

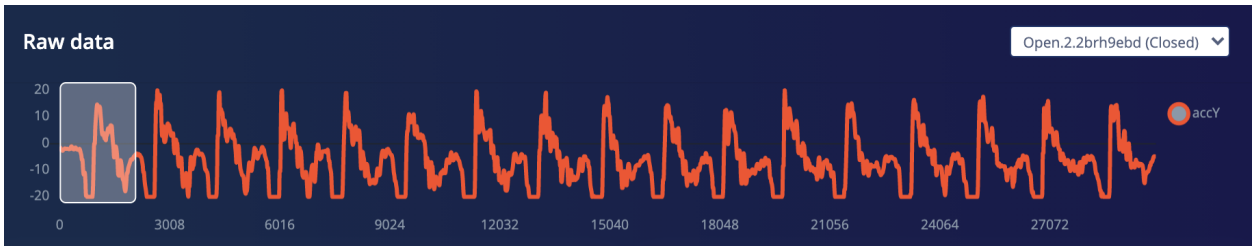
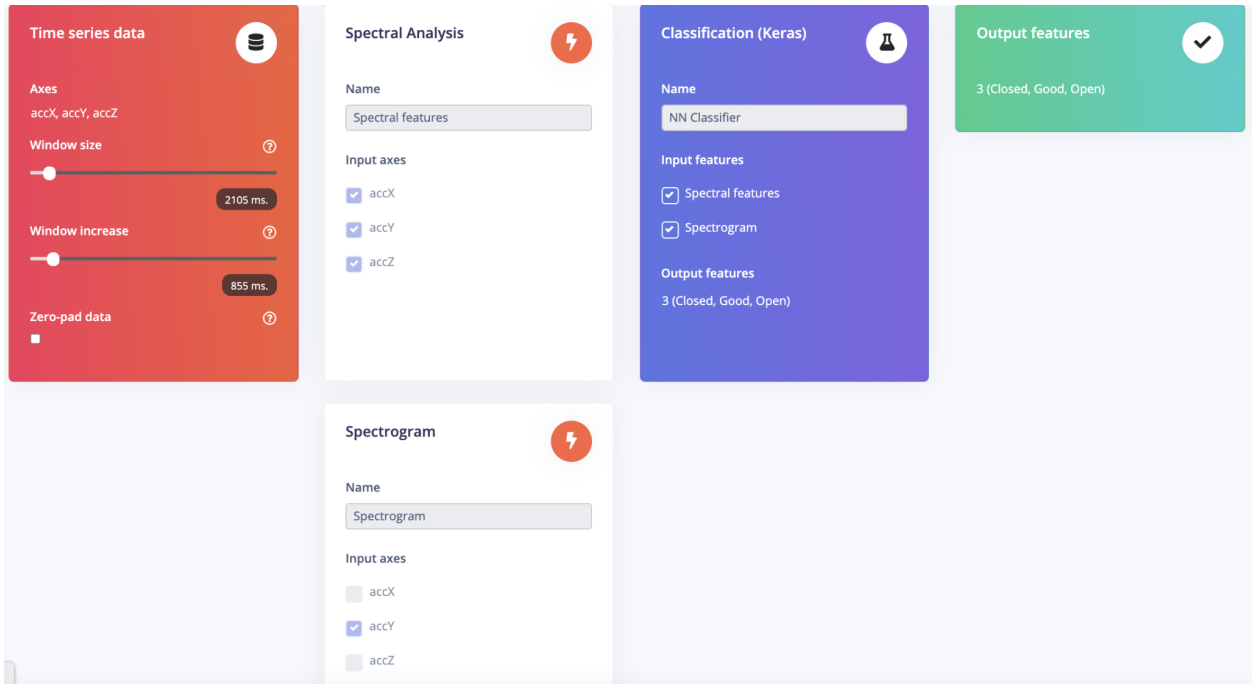


Figure: Spectrogram ACC_Y

Neural Network Implementation



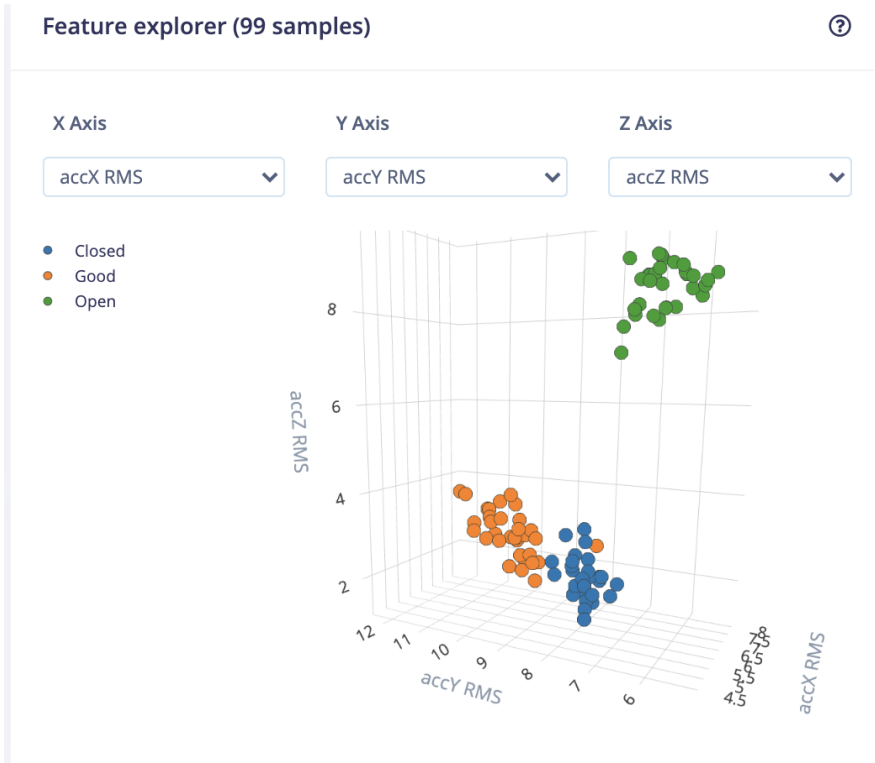


Figure: Spectral Analysis

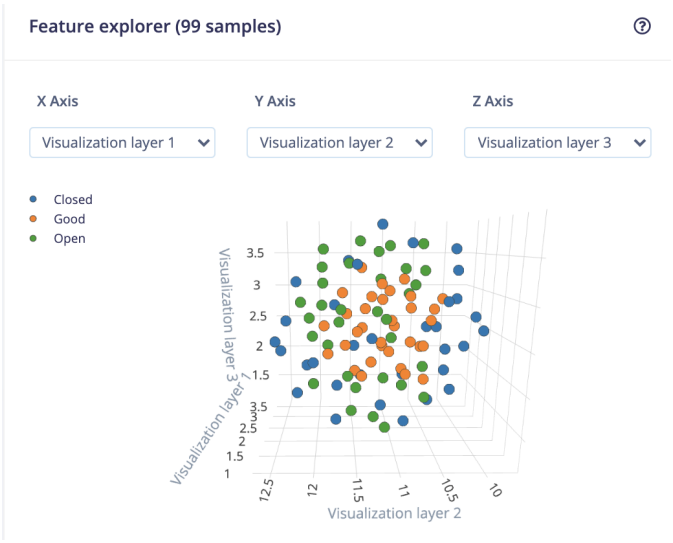


Figure: Spectrogram

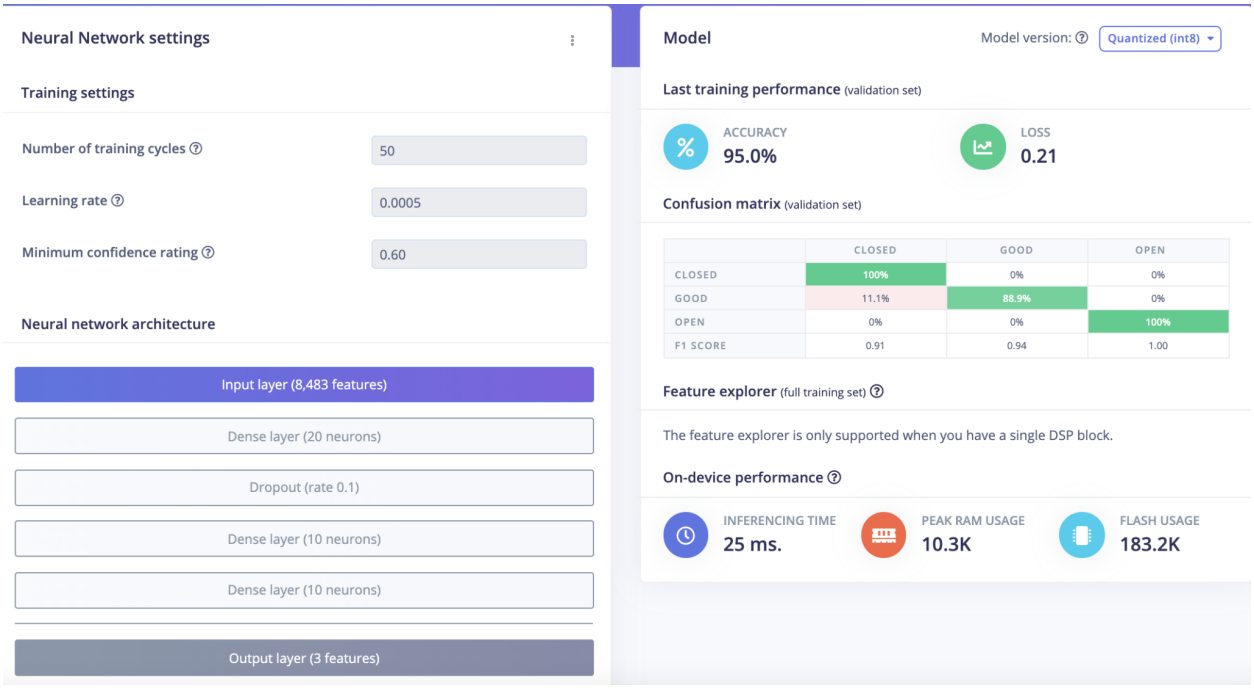
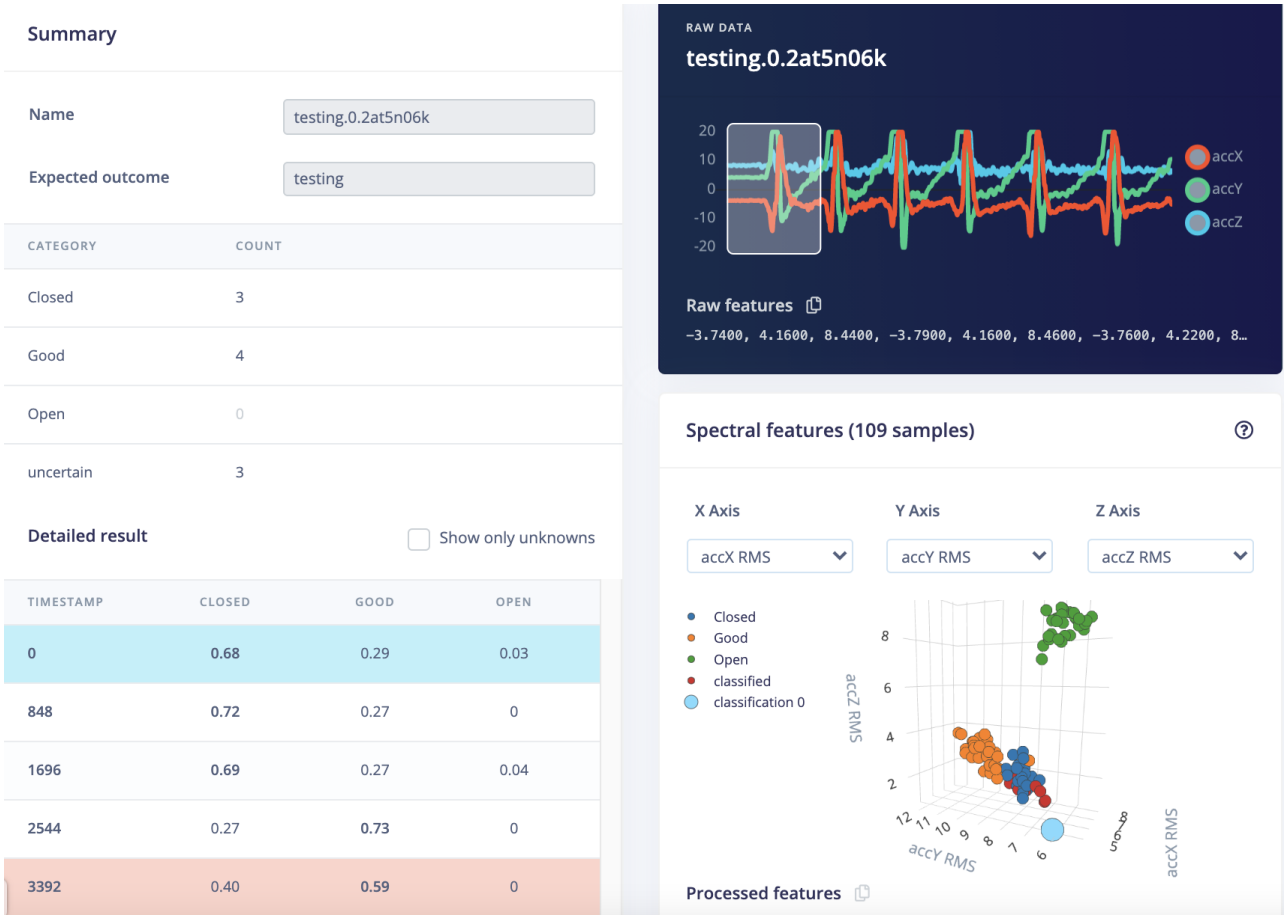


Figure: Neural Network and Model Performance

Demonstration

Live Classification - TEST 1:



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

Live Classification - TEST 2

I was able to learn from the mistakes I made in the first Live Classification test and did it again, this time for 30 seconds and developed a timing that coincided with the training data. This time I was able to correct my technique and create 22 proper swings as shown below:

CATEGORY	COUNT
Closed	0
Good	22
Open	0
uncertain	11

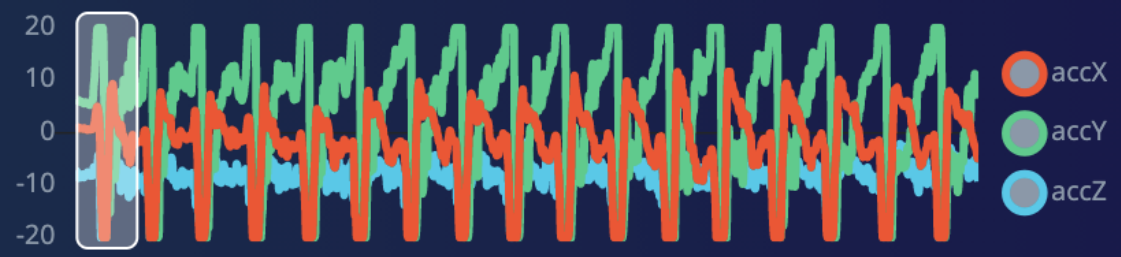
Detailed result

☐ Show only unknowns

TIMESTAMP	CLOSED	GOOD	OPEN
0	**0.55**	0.43	0.02
848	0.40	**0.60**	0
1696	0.36	**0.64**	0
2544	0.20	**0.80**	0
3392	0.46	**0.53**	0
4240	0.48	**0.52**	0
5088	**0.50**	0.49	0
5936	**0.51**	0.49	0
6784	**0.56**	0.44	0

RAW DATA

testing.3.2brjvee2



Raw features

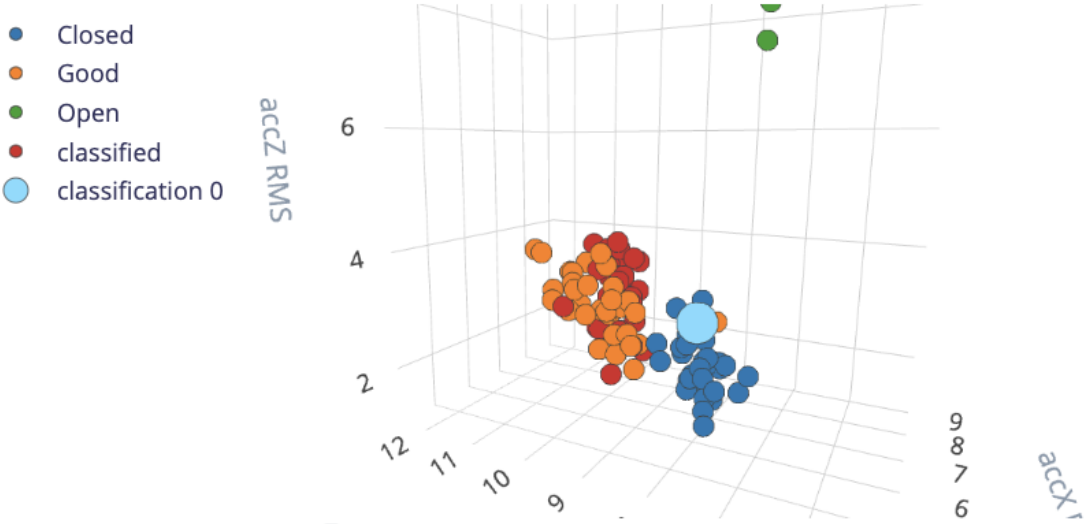
0.7200, 6.3400, -7.1700, 0.7500, 6.1400, -7.3200, 0.8200, 6.1000, -7.8000,…

Spectral features (132 samples)

X Axis: accX RMS

Y Axis: accY RMS

Z Axis: accZ RMS



Processed features

6.3189, 0.9921, 4.9414, 4.4643, 0.1694, 0.0000, 0.0000, 0.4971, 2.5003, 2.…

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

Source Code and Repository

This is a link to the public project on Edge Impulse:
https://studio.edgeimpulse.com/public/39443/latest

List of Milestones / Steps

1. Using Edge Impulse, collect data for 3 classes using the accelerometer sensor based on the angle of the racket (too open, too closed, and correct). Thirty seconds of each class were captured via the STM32 IoT Discovery Board attached to a racket using a USB connection to power the board and transmit the data.
2. Perform data analysis and report findings and graphs
3. Generate features (Edge Impulse) and perform any feature engineering
4. Build neural network model based on the training data as input
5. Perform Live Classification and develop a confusion matrix with accuracy results

Results & Discussion

Although the project was complete in that it collected, processed and analyzed data, trained a neural network based on a train/test set, performed live classification, and created npy and binary files for deployment, there are still pieces that need to be addressed including flashing the device, connecting to the arduino, and connecting speaker and an audio amplifier to the arduino, and modifying the C code, once the model is converted to C, such that there is an immediate feedback audio sound upon a swing of the racket.

Conclusions

The Table Tennis Deep Trainer is only in phase one at this time and I will continue the project once this report has been submitted.

Future Work

The next steps to be continued after this first version is complete is to do the following:

Once Edge Impulse creates the deployable files, I should be able to generate an H5 file that I can move to the IDE and import into C code. And then modify the C to generate a beep upon a certain condition. I would need to:

1. Create code in C that would trigger the beep in the speaker if the classification was correct
2. Compile the code and flash the Discovery board
3. Connect the board to the USB port and thus it is powered and running the model

4. I would then perform the swing and if it was correct, the speaker would beep, otherwise it would be quiet.

References:

1 - Tennis_Motion_ML_Classifier_Reference_Design.pdf
2 - (PDF) Smart Racket – Instrumented Racket as Real-time Feedback Device in Table Tennis

Appendix

Milestones

- Week 1.
  - Setting up STM32Cube IDE
  - Collect hardware
  - Work through tutorials
- Week 2.
  - Download and set up Anaconda and learn some data analytics
  - Work with Edge Impulse and the phone
  - Learn interface and how to build model
- Week 3.
  - More tutorials and data analytics,
  - Formulating project idea
  - Meeting with professor, TF, and table tennis coach.
  - Ordering the STM32 Discovery Kit
- Week 4.
  - Collect Training data with SensorTile and EDGE Impulse.
  - Perform live classification using the accelerometer data.
  - Using the tutorials, work on flashing the device.
  - Meet with Jose and discuss next steps.
  - Address the need for an external battery
  - Review embedded ML options
- Week 5.
  - Setup the STM32 Discovery Kit.
  - Get a speaker for audio output.
  - Consider connecting with the Arduino.

    - Address the need for utilizing multiple sensor types and not just the accelerometer. The discovery kit also has the ability to collect data from the gyroscope, magnetometer, and several others.
- Week 6.
    - Collect Training data with a kit mounted on racket and data analytics.
    - Training data can be collected with shadow practicing and using an actual tennis ball and table is not required for this project given time constraints.
    - Testing data with kit mounted on racket and data analytics
    - Build and train the model in keras/TF
    - Convert the code to an embedded form in C/C++
    - Flash the sensorTile processor
- Week 7.
    - Final testing
    - Perform data analytics
    - Prepare presentation
- Week 8.
    - Project Presentation