CS322 MINI PROJECT REPORT

Motion Classification

Using Deep Learning on ESP32

Team Members:
Kartik Mouli 2001CS35
Rohit Ranjan 2001CS56

Report By -
Kartik Mouli (2001CS35)

Aim

The aim of this project is to Implement a Neural Network on ESP32.

About

Motion Classification is a simple and at the same time great example of what machine learning can do. It uses lots of "messy" data to classify things.

In this project we will make a classifier with 4 classes :idle, up_down, left_right and circle.

Equipment used

Hardware Components

- ESP32 Module

The ESP32 is a very versatile System On a Chip (SoC) that can be used as a general purpose microcontroller with quite an extensive set of peripherals including Wi-Fi and Bluetooth wireless capabilities.

- MPU6050 Sensor

MPU6050 sensor module is complete 6-axis Motion Tracking Device. It combines 3-axis Gyroscope, 3-axis Accelerometer and Digital Motion Processor all in small package.

- Breadboard

A breadboard is a construction base used to build semi-permanent prototypes of electronic circuits.

- Jumper Wire

A jump wire is an electrical wire, or group of them in a cable, with a connector or pin at each end which is normally used to interconnect the components of a breadboard.

Software apps and online services

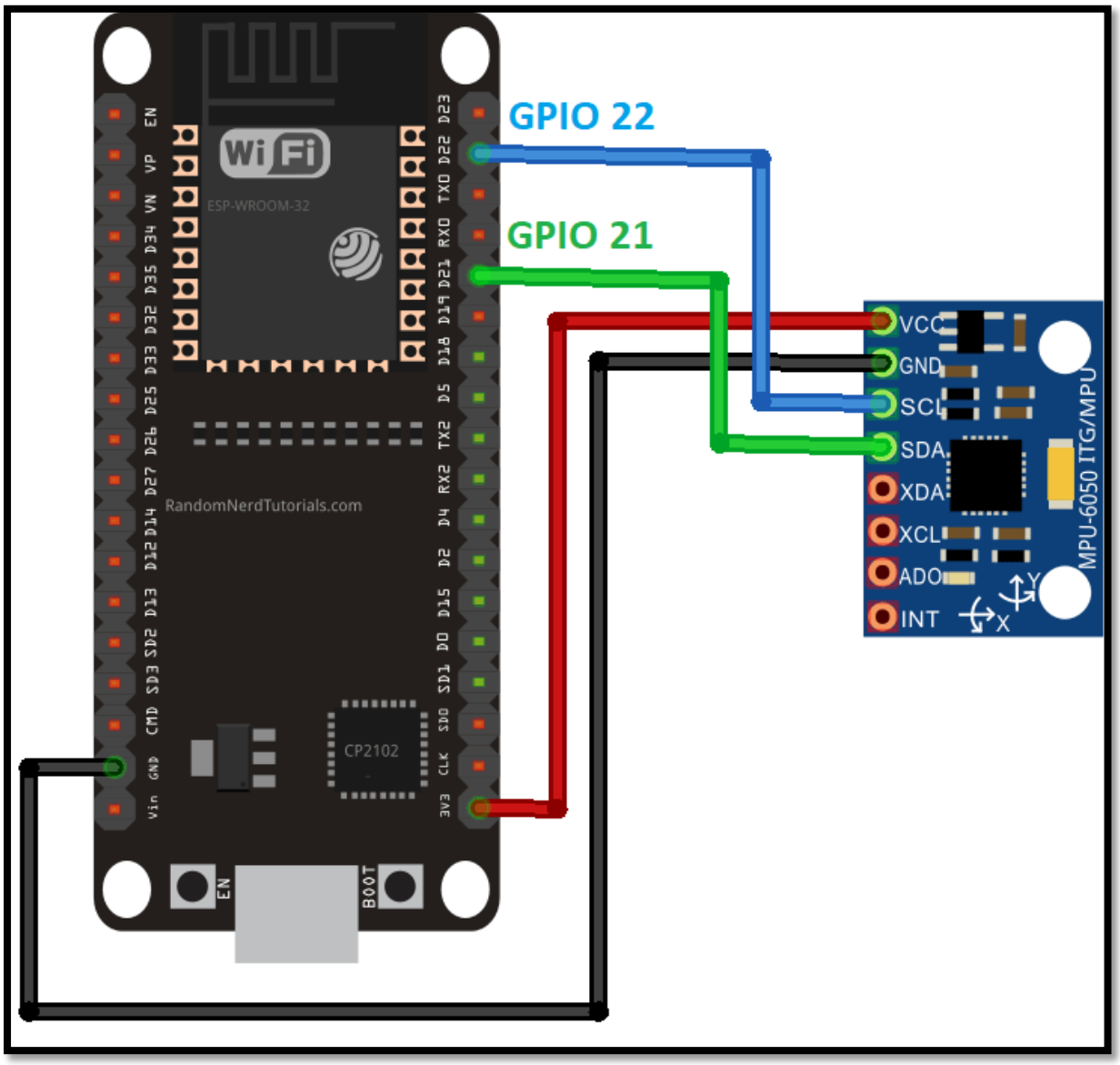
- Edge Impulse Studio

Edge Impulse is the leading development platform for machine learning on edge devices, free for developers and trusted by enterprises.

- Arduino IDE

The Arduino Integrated Development Environment or Arduino Software (IDE) - contains a text editor for writing code, a message area, a text console, a toolbar with buttons for common functions and a series of menus. It connects to the Arduino hardware to upload programs and communicate with them. The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Schematic



Hardware Connections

ESP32 -> MPU6050
D22 (GPIO 22) – SCL
D21 (GPIO 21) – SDA
3V3 – VCC
GND – GND

Project Contribution

Project has major 4 parts:

1) Testing hardware components
2) Data Acquisition
3) Creating ML Model
4) Deployment of Model

Rohit Ranjan (2001CS56) – Part 1 and 2
Kartik Mouli (2001CS35) – Part 3 and 4

Working

The MPU6050 Sensor collect data in 1 sec interval and send over to ESP32 module which has Motion Classification model deployed on it. It classifies the motion and gives output on serial monitor.

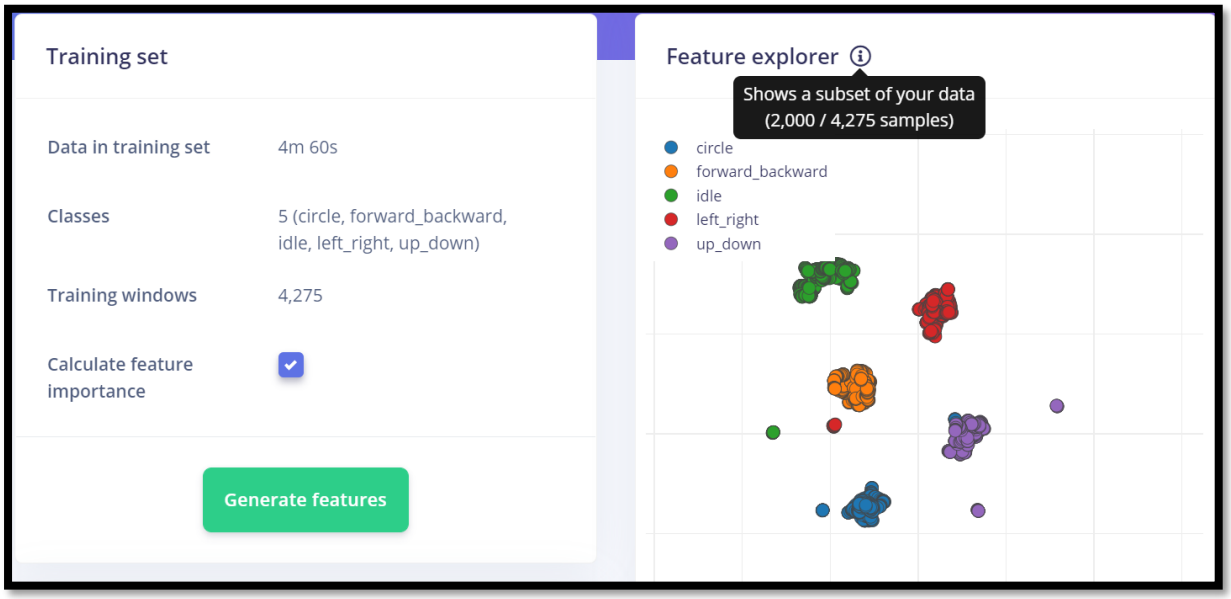
We have used Edge Impulse Studio which provides great environment for data acquisition and creating ML model. We deployed the model on ESP32 using Arduino IDE.

Part 1 and 2 will be explained by Rohit Ranjan (2001CS56).

Creating ML model

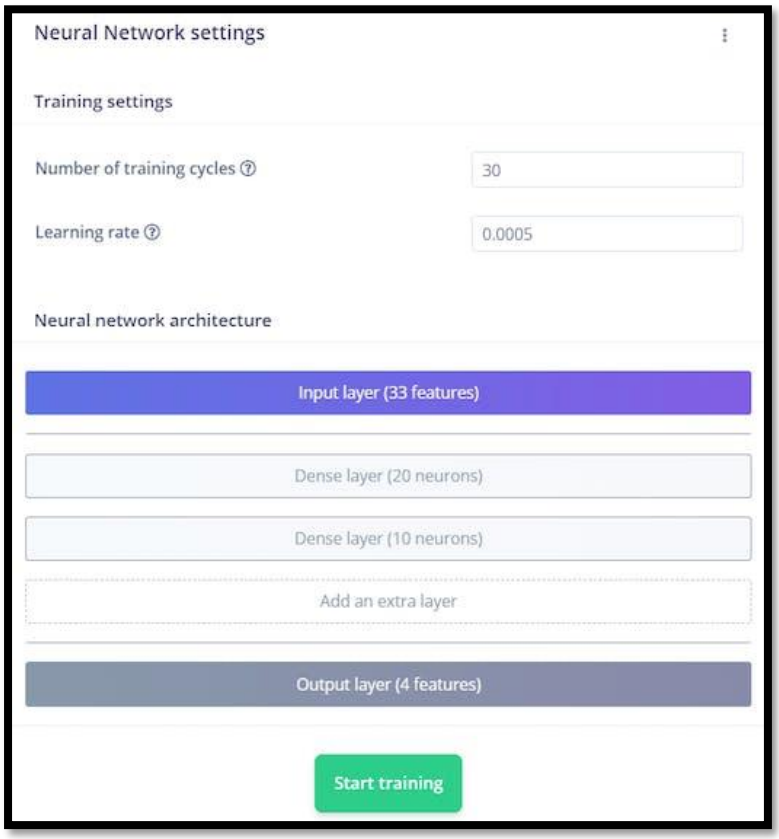
- Feature Explorer

We can see how well our classes are separated using the feature explorer:



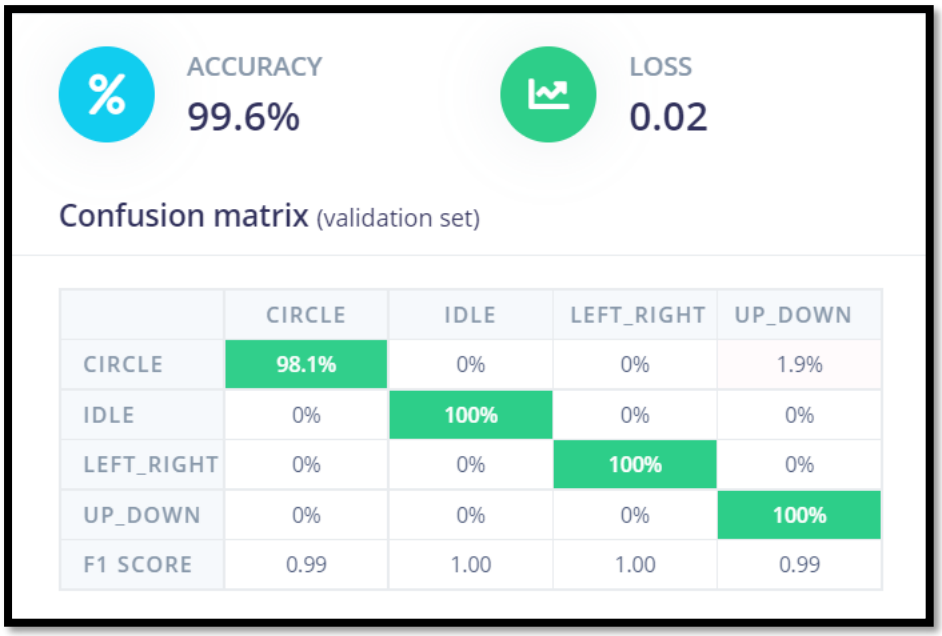
- Neural Network Implementation

```
import tensorflow as tf
from tensorflow .keras.models import Sequential
from tensorflow .keras.layers import Dense, Inputlayer, Dropout, Conv1D, Conv2D, Flatten, Reshape, MaxPooling1D, MaxPooling2D, BatchNormalization
from tensorflow .keras.optimizers import Adam
sys.path .append('./resources/libraries')

# model architecture
model = Sequential()
model.add(Dense(20, activation='relu ',
                activity_regularizer=tf.keras.regularizers.l1(0.00001)))
model.add(Dense(10, activation='relu ',
                activity_regularizer=tf.keras.regularizers.l1(0.00001)))
model.add(Dense(classes, activation='softmax ', name='y_pred'))

# this controls the learning rate
opt = Adam(lr=0.0005, beta_1=0.9, beta_2=0.999)

# this controls the batch size,or you can manipulate the tf.data"
BATCH_SIZE = 32
train_dataset = train_dataset.batch(BATCH_SIZE, drop_remainder=False)
validation_dataset = validation_dataset.batch(BATCH_SIZE, drop_remainder=False)
callbacks.append(BatchloggerCallback(BATCH_SIZE, train_sample_count))

# train the neural net1ork
model.compile(loss='categorical_crossentropy ',
              optimizer=opt, metrics=['accuracy'])
model.fit(train_dataset, epochs=30, validation_data=validation_dataset,
          verbose=2, callbacks=callbacks)
```



Model Performance

Confusion Matrix



	CIRCLE	IDLE	LEFT_RIGHT	UP_DOWN
CIRCLE	98.1%	0%	0%	1.9%
IDLE	0%	100%	0%	0%
LEFT_RIGHT	0%	0%	100%	0%
UP_DOWN	0%	0%	0%	100%
F1 SCORE	0.99	1.00	1.00	0.99

Result

For circle

```
Predictions (DSP: 14 ms., Classification: 0 ms.):
    circle: 0.98438
    idle: 0.00391
    left_right: 0.00000
    up_down: 0.01172
```

For idle

```
Features (14 ms.): 0.011759 0.385262 1.128474 -
Running impulse...
Predictions (time: 0 ms.):
circle: 0.000000
idle:   0.996094
left_right:     0.000000
up_down:        0.000000
```

For left_right

```
Running impulse...
Predictions (time: 0 ms.):
circle: 0.000000
idle:   0.000000
left_right:     0.996094
up_down:        0.000000
```

For up_down

```
Predictions (time: 0 ms.):
circle: 0.011719
idle:   0.000000
left_right:     0.000000
up_down:        0.988281
```

Conclusion

It was possible to implement a Neural Network using TinyML on ESP32 and a MPU6050 sensor.