BONAFIDE CERTIFICATE

Certified that this project report entitled "**GESTURE RECOGNITION USING MPU6050"** is a bonafide work of **ANUJ SINGH CHAUHAN – 20BRS1098** who carried out the Project work under my supervision and guidance for **CSE2039-FUNDAMENTALS OF ARTIFICIAL INTELLIGENCE.**

Dr. R. Jothi

Associate Professor

School of Computer Science and Engineering (SENSE),

VIT University, Chennai

Chennai – 600 127.

ABSTRACT

Gestures come natural to humans but for a machine i.e. computers, it's a very hard task to recognize gestures and understand what it symbolizes. In this project we are trying to make a gesture recognition system which monitors the larger movement of the body, in this case our hands, to figure out what activity or motion it is undergoing and then take appropriate actions accordingly. We are not using conventional camera to recognize gestures as using cameras is very impractical, in congested situations cameras would not have the proper field of view, so to tackle this problem we are proposing a sensor based movement recognition module, it uses MPU6050 which is a Inertial Measurement Unit (IMU) which can sense the acceleration and rotational acceleration about the three axes. The data can be used to train a neural network and then used to classify movements.

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. R. Jothi,** Associate Professor, School of Electronics Engineering, for her consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work. .

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

TABLE OF CONTENTS

1. INTRODUCTION

1.1 OBJECTIVES AND GOALS

The objective of our project is to create a cheap gesture recognition module, which can be used at places where the traditional cameras cannot work, that might be due to low levels of light or compactness of the space which will not allow the camera to use its field of view.

1.2 APPLICATIONS

The applications for this project is immense,

- Sports, sportsmen and women can use these module to recognize the way their body moves and then compare it to ideal form in which they should move.
- Smart phones and other smart devices, they can detect whether the movement that they are experiencing is some random movement or a certain type of movement i.e. the movement of picking up the phone vs. movement of walking with the phone in our hands.
- Smart home security systems, it can recognise the way each family member opens the door, so when someone who is not recognised opens the door it can detect that there is an anomaly and inform the owners.
- Virtual Reality, the world is moving towards virtual reality and this kind of cheap gesture recognition modules are very important for us to make this virtual reality available to everyone at a cheaper cost.

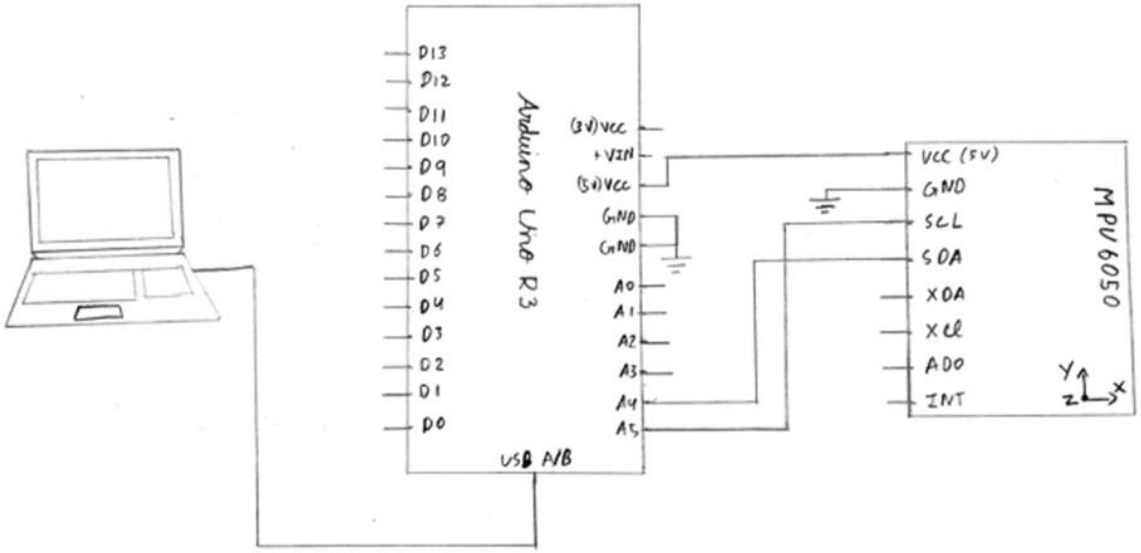
1.3 FEATURES

Features of this module is:

- Compact
- Cheap
- No field of view required

2. DESIGN

2.1 BLOCK DIAGRAM

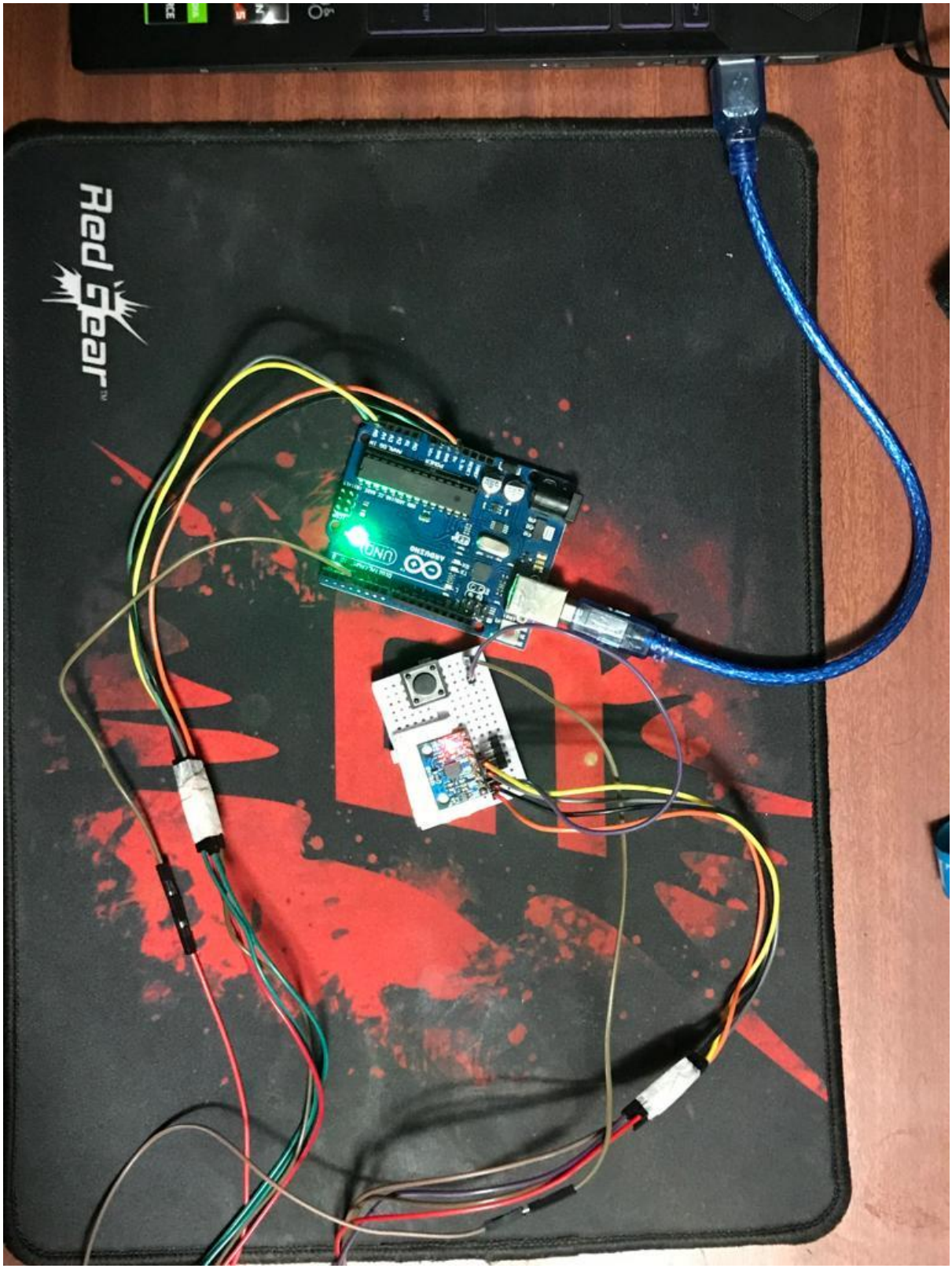


2.2 HARDWARE ANALYSIS

The hardware for this project consists of:-

- Arduino Uno R3: We are using Arduino Uno R3 microcontroller to interface the MPU6050 IMU, and normalize its data
- MPU6050: The MPU6050 is an IMU (Inertial Measurement Unit), which can sense the acceleration and rotation about x, y and z axes. This will sense our motions and send this data to the Arduino.
- PC: The final gesture classification is done on the PC using neural networks.

2.3 SNAPSHOTS



3. SOFTWARE

3.1 CODING

Arduino Uno Code:

Initializing all the variables, *numSamples* is the number of samples we need to take for each gestures, so we are taking 140 samples per gesture.

```
const int numSamples = 140;
int samplesRead = numSamples;
int Button = 3;
```

Now we need to record all the data that MPU6050 is sending, so the data is stored in form of a Vectors. The *mpu.readNormalizeAccel()* and *mpu.readNormalizeGyro()* read the normalized values in from of m/s2.

```
if(digitalRead(Button) == LOW){
//if(1){
  samplesRead = 0;
  while(samplesRead < numSamples){
    Vector rawAccel = mpu.readRawAccel();
    Vector normAccel = mpu.readNormalizeAccel();
    Vector rawGyro = mpu.readRawGyro();
    Vector normGyro = mpu.readNormalizeGyro();

    samplesRead++;
```

These Vectors are then send to the PC for further processing.

Recording Data on PC:

Serial library is used to interface Arduino to Python script, So we Import serial. To store the data we use xlwt library.

```
import serial
import xlwt
from xlwt import Workbook
```

Now we open the workbook to store the data in it.

```
wb = Workbook()
sheet1 = wb.add_sheet('Sheet 1')
```

Initialize the variable to receive the data from Arduino.

```
arduinoSerialData = serial.Serial('com3', 115200)
```

Now we record all the data from Arduino and store it the excel file that we have opened.

```
sheet1.write(rowCnt, 0, x)
sheet1.write(rowCnt, 1, y)
sheet1.write(rowCnt, 2, z)
sheet1.write(rowCnt, 3, gx)
sheet1.write(rowCnt, 4, gy)
sheet1.write(rowCnt, 5, gz)
rowCnt = rowCnt+1
```

Neural Network Model:

Importing all the important Libraries.

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import tensorflow as tf
```

Initializing all the variables and constants for the whole project.

```
SEED = 1337
np.random.seed(SEED)
tf.random.set_seed(SEED)

GESTURES = [
    "punch",
    "flex",
    "wave"
]

SAMPLES_PER_GESTURE = 140

NUM_GESTURES = len(GESTURES)
```

Load all the data into one data frame, df.

```
for gesture_index in range(NUM_GESTURES):
    gesture = GESTURES[gesture_index]
    print(f"Processing index {gesture_index} for gesture '{gesture}'.")

    output = ONE_HOT_ENCODED_GESTURES[gesture_index]

    df = pd.read_csv("D:/VIT SEM-4/ArdProject/" + gesture +".csv")
    num_recordings = int(df.shape[0]/SAMPLES_PER_GESTURE)

    print(f"\tThere are {num_recordings} recordings of the {gesture} gesture.")
```

Normalize the data values, Acceleration is between -4 to +4 so to normalize it we can add 4 and divide the whole thing by 8, similarly gyroscope is between -2000 to +2000 so to normalize it we add 2000 and divide the whole with 4000. Do this for each data entry of each and every gesture.

```
for i in range(num_recordings):
    tensor = []
    for j in range(SAMPLES_PER_GESTURE):
        index = i*SAMPLES_PER_GESTURE + j
        tensor += [
            (df['aX'][index] + 4)/8,
            (df['aY'][index] + 4)/8,
            (df['aZ'][index] + 4)/8,
            (df['gX'][index] + 2000)/4000,
            (df['gY'][index] + 2000)/4000,
            (df['gZ'][index] + 2000)/4000
        ]
    inputs.append(tensor)
    outputs.append(output)
```

Now we have loaded and normalized our data, but still our data is in a sequence i.e. all the data representing punch is together, it will not be ideal to train our model, So we go ahead and randomize our input data. After randomizing it we will split our data into 3 parts Train dataset, Test dataset and validation dataset.

```
num_inputs = len(inputs)
randomize = np.arange(num_inputs)
np.random.shuffle(randomize)

inputs = inputs[randomize]
outputs = outputs[randomize]

TRAIN_SPLIT = int(0.6 * num_inputs)
TEST_SPLIT = int(0.2 * num_inputs + TRAIN_SPLIT)

inputs_train, inputs_test, inputs_validate = np.split(inputs,
                                                      [TRAIN_SPLIT, TEST_SPLIT])
outputs_train, outputs_test, outputs_validate = np.split(outputs,
                                                         [TRAIN_SPLIT, TEST_SPLI

print("Data set randomization and splitting complete.")
inputs_train.shape, inputs_test.shape, inputs_validate.shape
```

We create a Dense Neural Network model, using keras.Sequential with total 3 layers, 1 input layer, 1 hidden layer, and 1 output layer.

The input layer has 50 perceptrons, the hidden layer have 15 perceptrons and the output layer has perceptrons equal to the number of gestures, i.e. 3.

The input layer and the hidden layer use the relu activation function, and the output layer uses softmax activation function as it has to classify the gestures.

```
model = tf.keras.Sequential([
    tf.keras.layers.Dense(50, activation = 'relu'),
    tf.keras.layers.Dense(15, activation = 'relu'),
    tf.keras.layers.Dense(NUM_GESTURES, activation = 'softmax')
])
```

While compiling the model for optimizer we use rmsprop optimizer, loss function is MSE and the metrics is MAE.

After compiling the model we fit the model with the training dataset.

```
model.compile(optimizer = 'rmsprop', loss = 'mse', metrics = ['mae'])

history = model.fit(inputs_train, outputs_train, epochs = 600, batch_size = 1,
                    validation_data = (inputs_validate, outputs_validate))
```

We use this trained model to predict on test dataset.

```
predictions = model.predict(inputs_test)

print("predictions = \n", np.round(predictions, decimals = 3))
print("actual = \n", outputs_test)
```

Now that the model is trained we can save our model.

```
model.save("Gesture_recognition.h5")
```

Script for live classification of gestures:

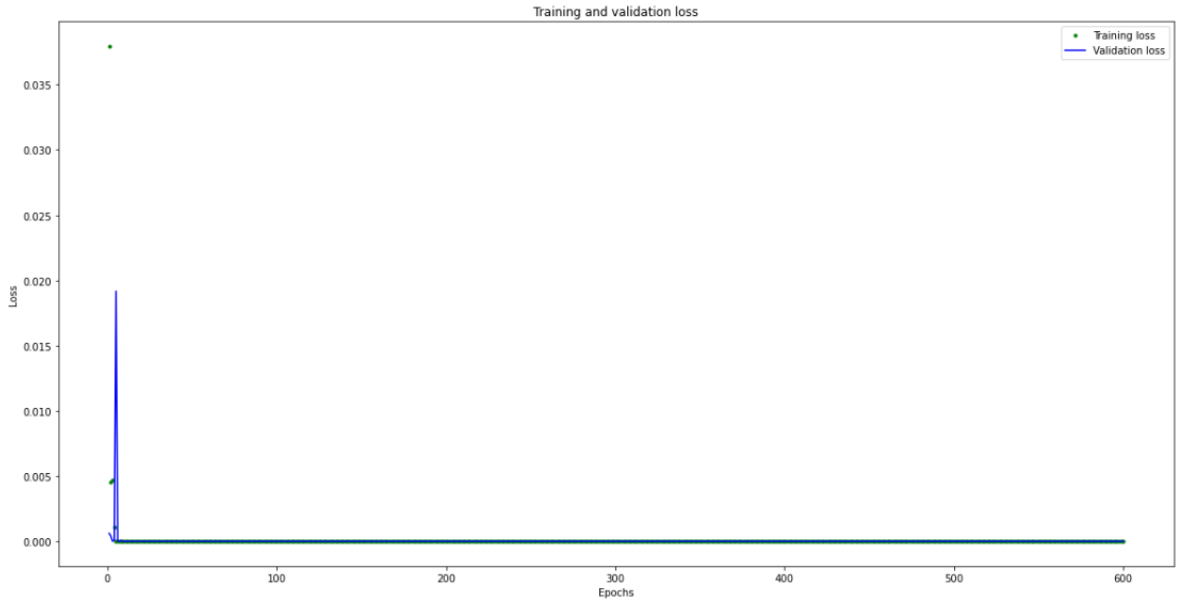
We create another script to interface with arduino but now we have to load our neural network model in this script and rather than storing the values in a excel file we have feed this data to the model for it classify the gesture.

```
rowCnt = 0
trained_model = tf.keras.models.load_model("Gesture_recognition.h5")
print(trained_model.summary())
```

```
if(cnt == 140):
    ten = getgest(ten)
    predictions = trained_model.predict(ten.reshape(1, 840))
    ind = np.where(np.round(predictions, decimals = 3) == 1)
    print("The output is ",np.where(np.round(predictions, decimals = 3)))
    cnt = 0
    ten = []
```

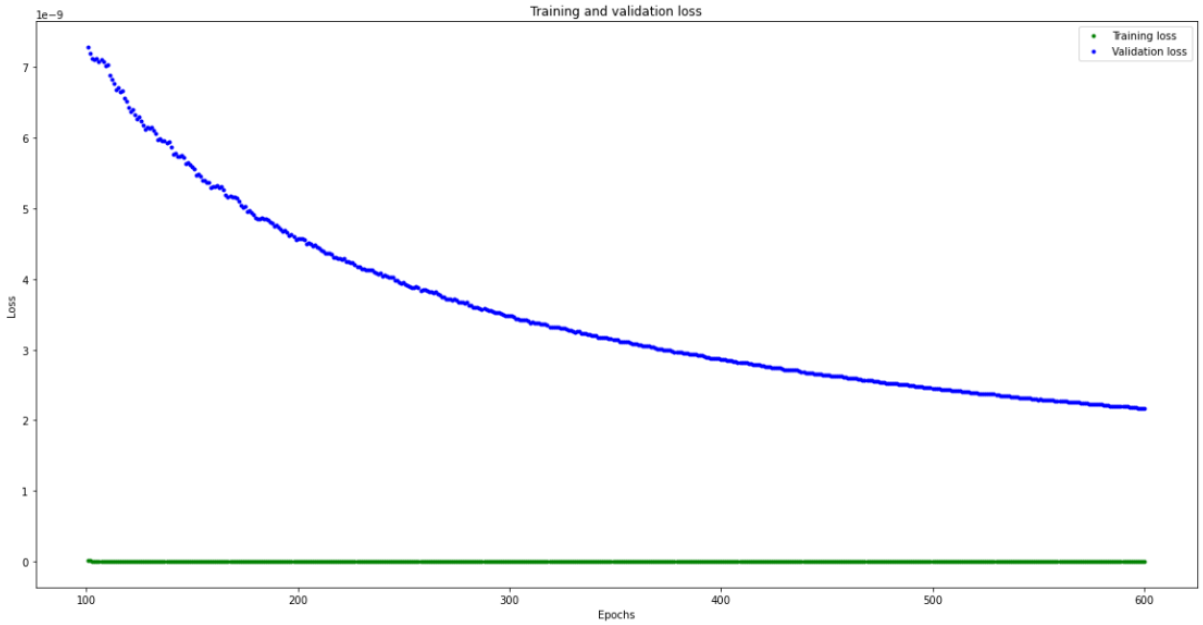
3.2 ANALYSIS

Graphs After training the model:



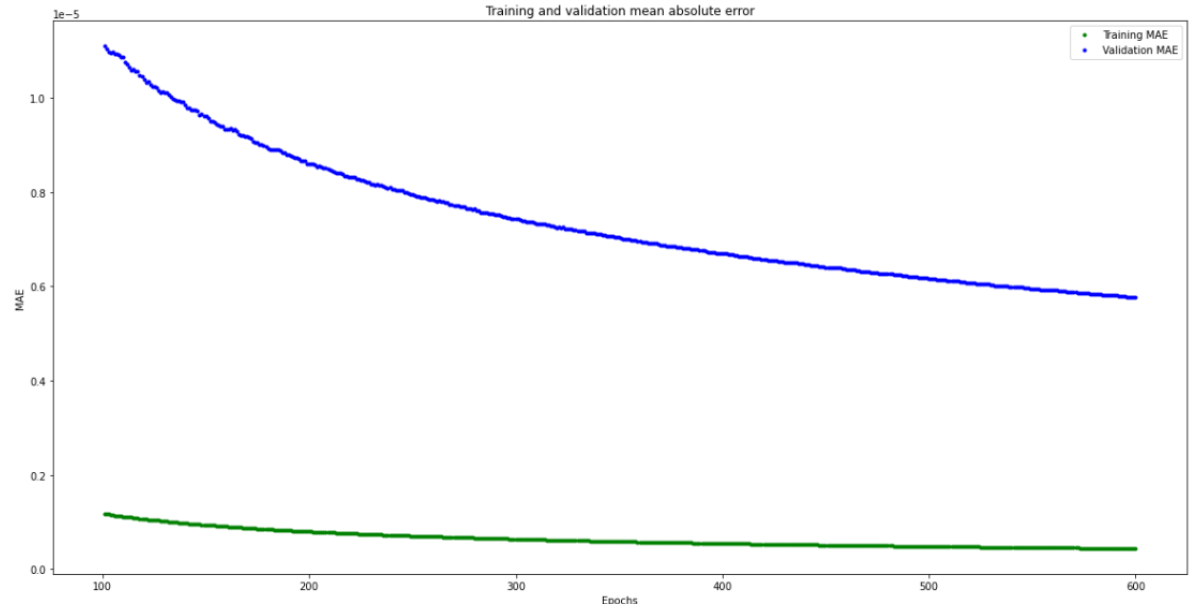
Graph 1: Training and validation loss

In Graph 1 we can see that the loss in the initial epochs is very high compared to the rest of the epochs, so now we remove the first 100 epochs from the graph and then see the losses vs epochs.



Graph 2: Training and validation loss (reduced epochs)

Now we can see the training loss and the validation loss, the training loss is very less as it has already learned the training dataset, the validation loss on the other hand tells us that the model is learning as the validation loss is also decreasing after every epoch.



Graph 3: Training and validation mean absolute error

The mean absolute error for training dataset is again less from the start as the model as learned the training data set, but as we see that the MAE for validation is also reducing so the model is learning.

After predicting on the test dataset we check the accuracy.

```
accuracy_score(outputs_test,np.round(predictions, decimals = 1))
```

```
1.0
```

The output is in the form of an array if the 0^{th} index is 1 it means the gesture was a punch, if 1^{st} index is 1 it means the gesture was a Flex and if the 2^{nd} index is 1 it means the gesture was a Wave.

```
The output is in the form [Punch, Flex, Wave] [[1. 0. 0.]]   Punch
The output is in the form [Punch, Flex, Wave] [[1. 0. 0.]]   Punch
The output is in the form [Punch, Flex, Wave] [[0. 1. 0.]]   flex
The output is in the form [Punch, Flex, Wave] [[0. 0. 1.]]   wave
The output is in the form [Punch, Flex, Wave] [[1. 0. 0.]]   Punch
```

4. CONCLUSION AND FUTURE WORK

3.1 RESULT, CONCLUSION AND INFERENCE

So the project worked as expected, the gesture recognition module is successful in classifying 3 different types of gestures. The DNN is giving 100% accuracy in during the testing phase.

3.2 FUTURE WORK

This project is just a prototype, and for future work:-

- Work can be done to make it a completely wireless module using Bluetooth module to communicate with the server and use the model of Arduino that can be run tensorflow code.

REFERENCES

[1] https://blog.tensorflow.org/2019/11/how-to-get-started-with-machine.html