IE417/EL530-Introduction to Embedded Artificial Intelligence

Lab 2

Building a Gesture Recognition System

Group name : Embedded Minds

Ruturajsinh Chauhan(202101146)

Devansh Shrimali(202101492)

Raj Chauhan(202101049)

Jalp Patel(202101267)

Gesture-Controlled YouTube Using Arduino Nano BLE 33 Sense and Edge Impulse

1. Introduction

This project aims to control YouTube using gestures detected by an accelerometer on the Arduino Nano BLE 33 Sense, leveraging Edge Impulse for machine learning. The gesture inputs control YouTube through Python and the PyAutoGUI library by simulating keyboard shortcuts.

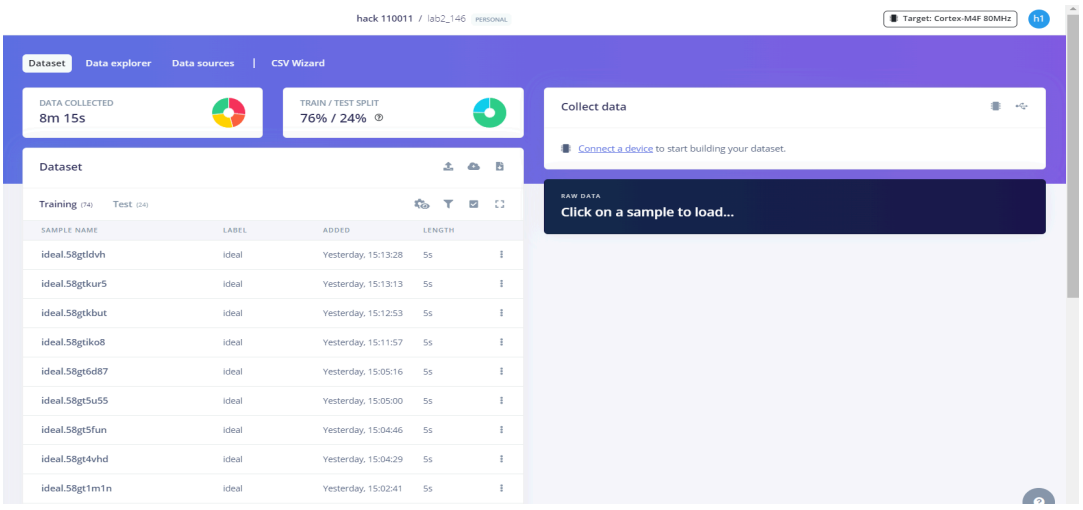
2. Data Collection

Data was collected using the onboard IMU (Inertial Measurement Unit) of the Arduino Nano BLE 33 Sense. The IMU measures acceleration on three axes—X, Y, and Z. The acceleration data is sampled and then processed by Edge Impulse.

Steps for Data Collection:

- • Sensor Data: We collected data on three axes: accX, accY, and accZ.
- • Gestures: We recorded three gestures—circle, up_down, and pan.
- • Circle: Corresponds to the next video action.
- • Up_down: Corresponds to the mute/unmute action.
- • Pan: Corresponds to the play/pause action.

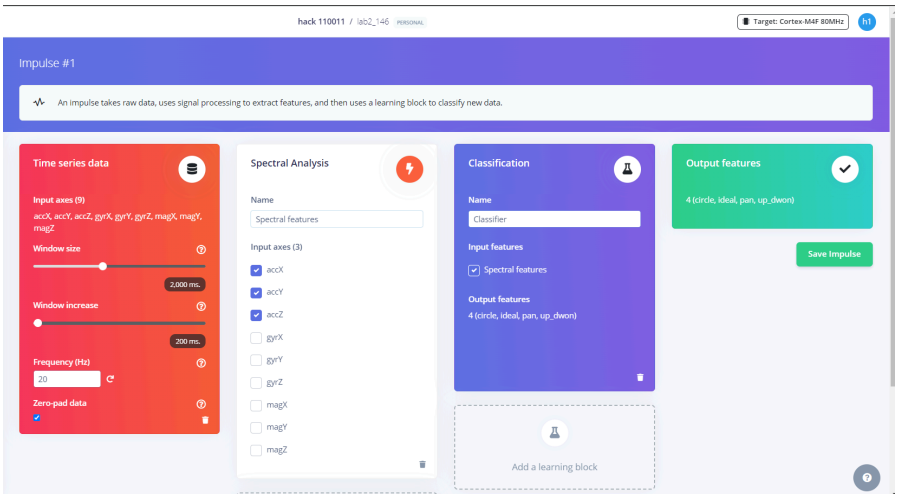
Below is the dataset visualization during the collection process:



3. Spectral Analysis

Edge Impulse performs spectral analysis on the collected data. Spectral features are extracted from the raw accelerometer data, which is then used to train machine learning models. The analysis helps in identifying patterns corresponding to the gestures.

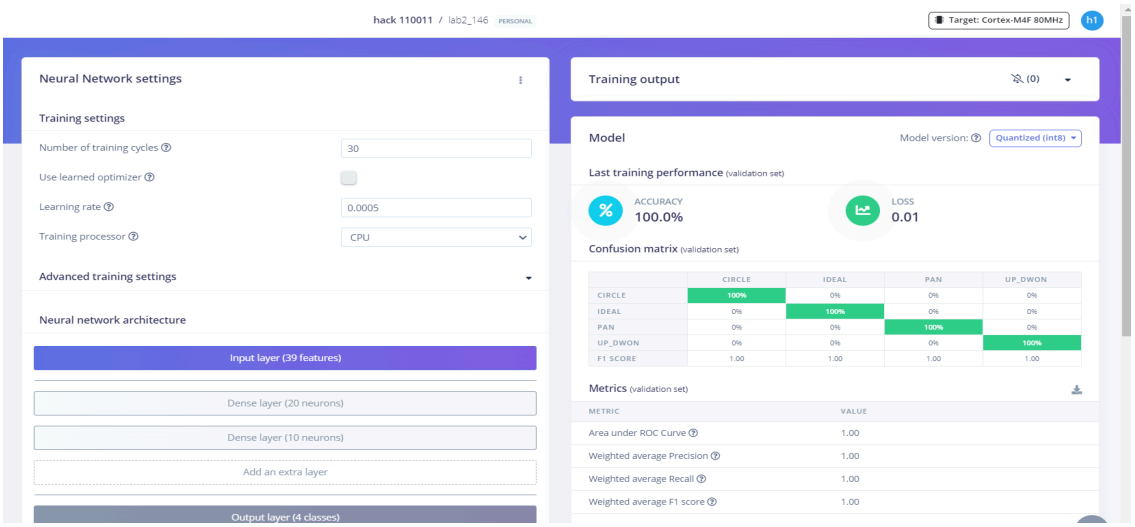
Below is a visualization of the features extracted using spectral analysis:



4. Model Training

We used Edge Impulse to train a model for classifying the gestures based on the accelerometer data. The model is trained on a dataset comprising three gestures: circle, up_down, and pan.

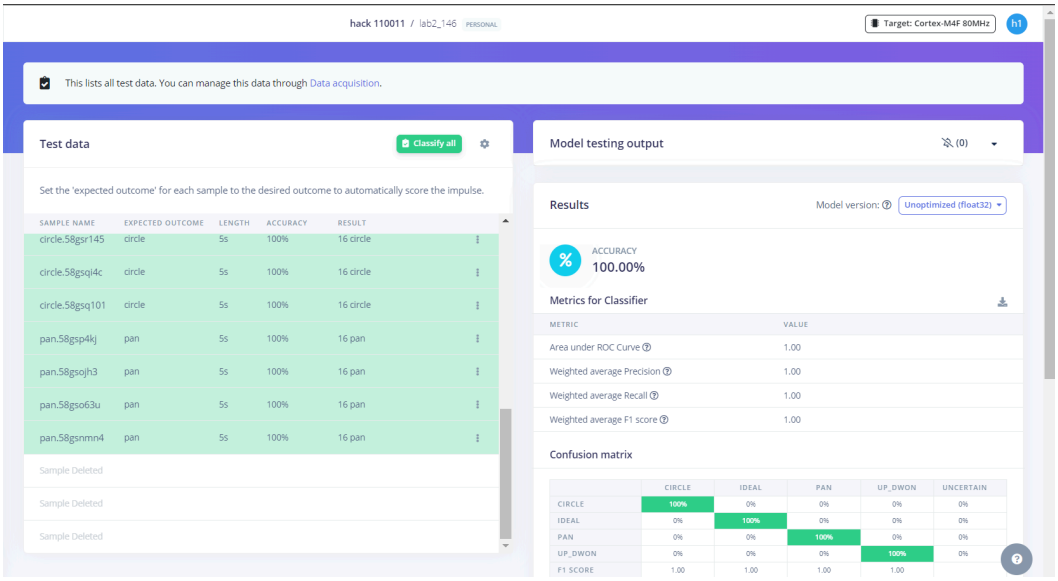
Below is a diagram of the model classification process:



5. Testing and Results

Once the model was trained, we tested its performance by feeding it new gesture data and verifying if it correctly classified the actions. The model's performance was satisfactory for all the gestures with minimal errors.

Below are the classification outputs and testing results:



6. Code Implementation

- Reads the gesture commands from the Serial port.
- Uses the PyAutoGUI library to simulate YouTube keyboard controls:
    - "next" → right arrow key (next video)
    - "mute" → 'm' key (mute/unmute)
    - "play" → 'k' key (play/pause)

The Python code used for gesture control with YouTube, leveraging the trained model and PyAutoGUI, is provided below:

```
import serial

import pyautogui

import time

# Set the correct COM port (check your Arduino IDE for the correct
port)

ser = serial.Serial('COM6', 115200, timeout=1)

while True:

    if ser.in_waiting > 0:

        line = ser.readline().decode('utf-8').strip()

        print(f"Received: {line}")



        if "next" in line:

            pyautogui.press('right')  # Next video

        elif "mute" in line:

            pyautogui.press('m')  # Mute/unmute

        elif "play" in line:

            pyautogui.press('k')  # Play/pause

    time.sleep(0.1)
```

7. Arduino Code

- Collects motion data from the accelerometer on the Nano BLE 33 Sense.
- Uses the Edge Impulse model to classify the gestures (like "circle", "up_down", and "pan").
- Sends a corresponding command ("next", "mute", "play") to the computer via Serial.

The Arduino Nano BLE 33 Sense code for collecting accelerometer data and sending it to the Edge Impulse platform is provided below:

```
#include <lab2_146_inferencing.h>   // Edge Impulse inference library
#include <Arduino_LSM9DS1.h>        // IMU sensor library

#define CONVERT_G_TO_MS2    9.80665f
#define MAX_ACCEPTED_RANGE  2.0f

void setup() {
    Serial.begin(115200);
    while (!Serial);  // Wait for Serial connection
    Serial.println("Gesture Recognition with YouTube Control");

    if (!IMU.begin()) {
        Serial.println("Failed to initialize IMU!");
    } else {
        Serial.println("IMU initialized");
    }

    if (EI_CLASSIFIER_RAW_SAMPLES_PER_FRAME != 3) {
        Serial.println("Error: Classifier input size should be 3 (for 3
       sensor axes)");
        return;
    }
}

float getSign(float number) {
    return (number >= 0.0) ? 1.0 : -1.0;
}

void loop() {
    delay(10);
    Serial.println("Sampling...");
```

```
    float buffer[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE] = { 0 };

{   for (size_t ix = 0; ix < EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE; ix += 3)
1000);  uint64_t next_tick = micros() + (EI_CLASSIFIER_INTERVAL_MS *
        IMU.readAcceleration(buffer[ix], buffer[ix + 1], buffer[ix + 2]);

        for (int i = 0; i < 3; i++) {
            if (fabs(buffer[ix + i]) > MAX_ACCEPTED_RANGE) {
                buffer[ix + i] = getSign(buffer[ix + i]) *
                  MAX_ACCEPTED_RANGE;
            }
        }

        buffer[ix + 0] *= CONVERT_G_TO_MS2;
        buffer[ix + 1] *= CONVERT_G_TO_MS2;
        buffer[ix + 2] *= CONVERT_G_TO_MS2;

        delayMicroseconds(next_tick - micros());
    }
  signal_t signal;
    int err = numpy::signal_from_buffer(buffer,
          EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE, &signal);
    if (err != 0) {
        Serial.println("Failed to create signal from buffer");
        return;
    }

    ei_impulse_result_t result = { 0 };
    err = run_classifier(&signal, &result, false);
    if (err != EI_IMPULSE_OK) {
        Serial.print("Error: ");
        Serial.println(err);
        return;
    }

    // Example for controlling YouTube:
    String controlCommand = "";
    if (result.classification[0].value > 0.8) {
        controlCommand = "next";  // Circle gesture for next video
    } else if (result.classification[3].value > 0.8) {
        controlCommand = "mute";  // up_down gesture for mute/unmute
    } else if (result.classification[2].value > 0.8) {
        controlCommand = "play";  // pan gesture for play/pause
    }

    if (controlCommand != "") {
```

```
        Serial.print("Control: ");
        Serial.println(controlCommand);
    }
}
```

8. Video Link

lab2_out_146_049_267_492.mp4