Edge Computing Laboratory
Lab Assignment 5

Name: Gaurav Gadekar
Class: TY AIEC Batch C
Enrollment No: MITU23BTCSD120
Roll No: D2233120

Title
The “Hello World” of Edge Impulse Platform

Introduction

Edge Impulse is a development platform for machine learning on edge devices, targeted at developers who want to create intelligent device solutions. The "Hello World" equivalent in Edge Impulse would typically involve creating a simple machine learning model that can run on an edge device, like classifying sensor data or recognizing a basic pattern.

Objective

TinyML: Building and Training a Model

Materials Required

Raspberry Pi 4 / Nano BLE Sense Board

Theory

GPIO (General Purpose Input/Output) pins on the Raspberry Pi are used for interfacing with other electronic components. BCM numbering refers to the pin numbers in the Broadcom SOC channel, which is a more consistent way to refer to the GPIO pins across different versions of the

Here’s a high-level overview of steps you'd follow to create a "Hello World" project on Edge Impulse:

Steps to Configure the Edge Impulse:

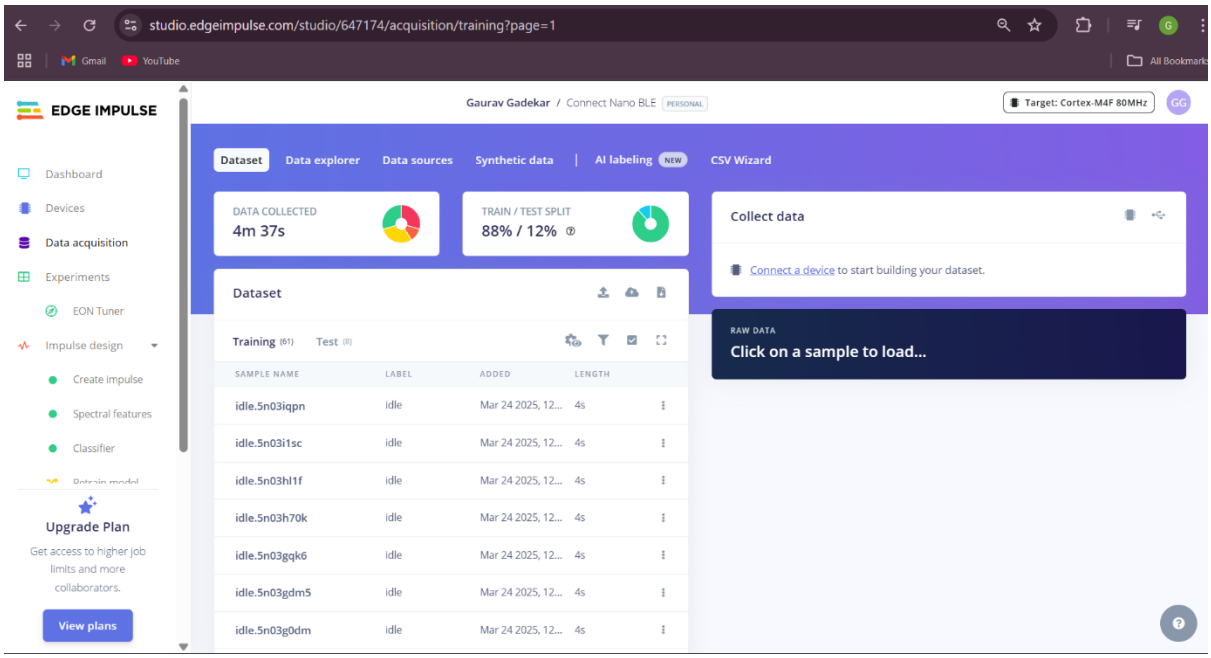
1. Create an Account and New Project:
    - Sign up for an Edge Impulse account.
    - Create a new project from the dashboard.
2. Connect a Device:
    - You can use a supported development board or your smartphone as a sensor device.
    - Follow the instructions to connect your device to your Edge Impulse project.
3. Collect Data:

- Use the Edge Impulse mobile app or the Web interface to collect data from the onboard sensors.
- For a "Hello World" project, you could collect accelerometer data, for instance.

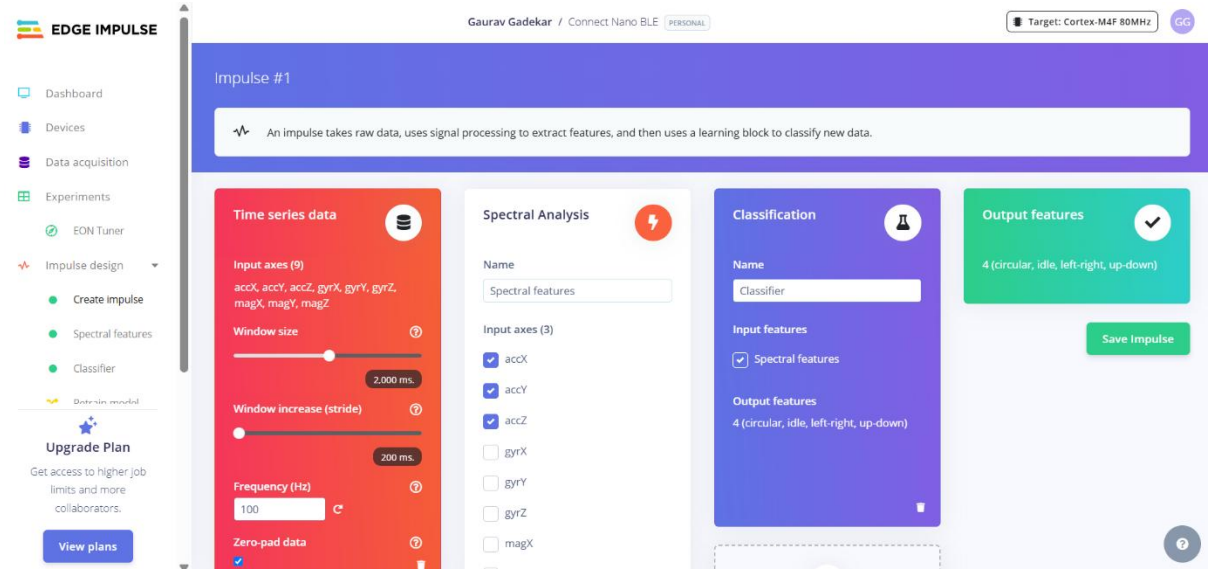
4. Create an Impulse:
    - Go to the 'Create impulse' page.
    - Add a processing block (e.g., time-series data) and a learning block (e.g., classification).
    - Save the impulse, which defines the machine learning pipeline.
5. Design a Neural Network:
    - Navigate to the 'NN Classifier' under the 'Learning blocks'.
    - Design a simple neural network. Edge Impulse provides a default architecture that works well for most basic tasks.
6. Train the Model:
    - Click on the 'Start training' button to train your machine learning model with the collected data.
7. Test the Model:
    - Once the model is trained, you can test its performance with new data in the 'Model Testing' tab.
8. Deploy the Model:
    - Go to the 'Deployment' tab.
    - Select the deployment method that suits your edge device (e.g., Arduino library, WebAssembly, container, etc.).
    - Follow the instructions to deploy the model to your device.
9. Run Inference:
    - With the model deployed, run inference on the edge device to see it classifying data in real-time.
10. Monitor:
    - You can monitor the performance of your device through the Edge Impulse studio.

Screenshots:

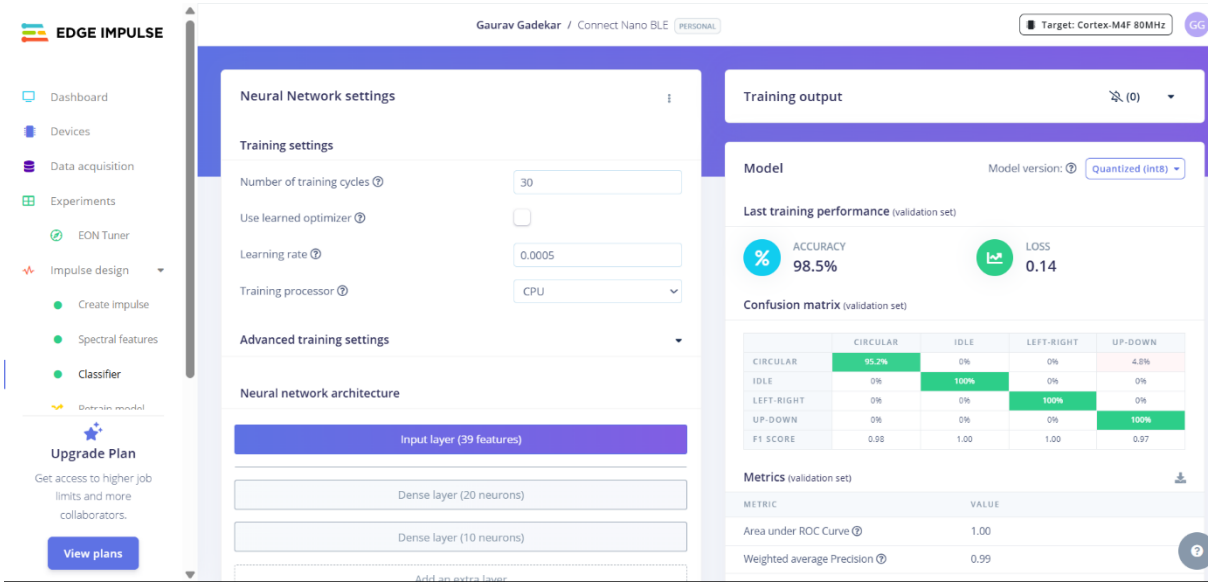
1. Dataset Image



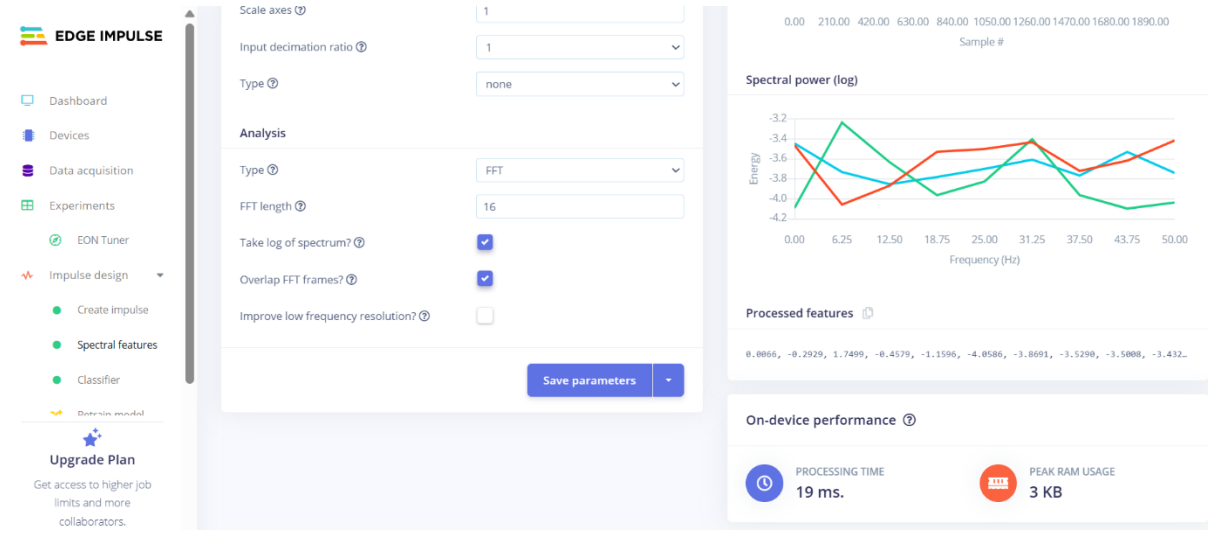
2. Feature extraction - Image



3. Accuracy / Loss - Confusion Matrix – image



4. Validation Result – Image



5. Copy the code of Arduino Sketch

nano_ble33_sense_accelerometer_continuous.ino | nano_ble33_sense_microphone_continuous.ino

```
#include <Arduino_LSM9DS1.h>

/* Edge Impulse ingestion SDK
 * Copyright (c) 2022 EdgeImpulse Inc.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 *
 */

/* Includes ---------------------------------------------------------------- */
#include <Gesture_Recognition_inferencing.h>
#include <Arduino_LSM9DS1.h> //Click here to get the library: https://www.arduino.cc/reference/en/libraries/arduino_lsm9ds1/

/* Constant defines -------------------------------------------------------- */
#define CONVERT_G_TO_MS2    9.80665f
/**
 * When data is collected by the Edge Impulse Arduino Nano 33 BLE Sense
 * firmware, it is limited to a 2G range. If the model was created with a
 * different sample range, modify this constant to match the input values.
 * See https://github.com/edgeimpulse/firmware-arduino-nano-33-ble-sense/blob/master/src/sensors/ei_lsm9ds1.cpp
 * for more information.
 */
#define MAX_ACCEPTED_RANGE  2.0f

/*
 ** NOTE: If you run into TFLite arena allocation issue.
 **
 ** This may be due to may dynamic memory fragmentation.
 ** Try defining "-DEI_CLASSIFIER_ALLOCATION_STATIC" in boards.local.txt (create
 ** if it doesn't exist) and copy this file to
 ** `<ARDUINO_CORE_INSTALL_PATH>/arduino/hardware/<mbed_core>/<core_version>/`.
 **
 ** See
 ** (https://support.arduino.cc/hc/en-us/articles/360012076960-Where-are-the-installed-cores-located-)
 ** to find where Arduino installs cores on your machine.
 **
 ** If the problem persists then there's not enough memory for this model and application.
```

```
/* Private variables ------------------------------------------------------- */
static bool debug_nn = false; // Set this to true to see e.g. features generated from the raw signal
static uint32_t run_inference_every_ms = 200;
static rtos::Thread inference_thread(osPriorityLow);
static float buffer[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE] = { 0 };
static float inference_buffer[EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE];

/* Forward declaration */
void run_inference_background();

/**
* @brief      Arduino setup function
*/
void setup()
{
    // put your setup code here, to run once:
    Serial.begin(115200);
    // comment out the below line to cancel the wait for USB connection (needed for native USB)
    while (!Serial);
    Serial.println("Edge Impulse Inferencing Demo");

    if (!IMU.begin()) {
        ei_printf("Failed to initialize IMU!\r\n");
    }
    else {
        ei_printf("IMU initialized\r\n");
    }

    if (EI_CLASSIFIER_RAW_SAMPLES_PER_FRAME != 3) {
        ei_printf("ERR: EI_CLASSIFIER_RAW_SAMPLES_PER_FRAME should be equal to 3 (the 3 sensor axes)\n");
        return;
    }

    inference_thread.start(mbed::callback(&run_inference_background));
}

/**
 * @brief Return the sign of the number
 *
 * @param number
```

6. Screen shot of Arduino Terminal - Result

```
5:43:05.997 -> Predictions (DSP: 104 ms., Classification: 0 ms., Anomaly: 0 ms.): left-right  [ 0, 0, 0, 10, 0, 0, ]
5:43:06.323 -> Predictions (DSP: 106 ms., Classification: 0 ms., Anomaly: 0 ms.): left-right  [ 0, 0, 0, 10, 0, 0, ]
5:43:06.634 -> Predictions (DSP: 106 ms., Classification: 0 ms., Anomaly: 0 ms.): left-right  [ 0, 0, 1, 9, 0, 0, ]
5:43:06.901 -> Predictions (DSP: 106 ms., Classification: 0 ms., Anomaly: 0 ms.): left-right  [ 0, 0, 1, 8, 1, 0, ]
5:43:07.214 -> Predictions (DSP: 106 ms., Classification: 0 ms., Anomaly: 0 ms.): left-right  [ 0, 0, 1, 7, 2, 0, ]
5:43:07.523 -> Predictions (DSP: 106 ms., Classification: 0 ms., Anomaly: 0 ms.): uncertain  [ 0, 0, 1, 6, 3, 0, ]
5:43:07.848 -> Predictions (DSP: 106 ms., Classification: 0 ms., Anomaly: 0 ms.): uncertain  [ 0, 0, 2, 5, 3, 0, ]
5:43:08.172 -> Predictions (DSP: 106 ms., Classification: 0 ms., Anomaly: 0 ms.): uncertain  [ 0, 0, 2, 4, 4, 0, ]
5:43:08.448 -> Predictions (DSP: 106 ms., Classification: 0 ms., Anomaly: 0 ms.): uncertain  [ 0, 0, 3, 3, 4, 0, ]
```

```
15:43:18.041 -> Predictions (DSP: 104 ms., Classification: 0 ms., Anomaly: 0 ms.): idle  [ 0, 0, 10, 0, 0, 0, ]
15:43:18.373 -> Predictions (DSP: 106 ms., Classification: 0 ms., Anomaly: 0 ms.): idle  [ 0, 0, 10, 0, 0, 0, ]
15:43:18.686 -> Predictions (DSP: 106 ms., Classification: 0 ms., Anomaly: 0 ms.): idle  [ 0, 0, 10, 0, 0, 0, ]
15:43:18.970 -> Predictions (DSP: 106 ms., Classification: 0 ms., Anomaly: 0 ms.): idle  [ 0, 0, 10, 0, 0, 0, ]
15:43:19.314 -> Predictions (DSP: 104 ms., Classification: 0 ms., Anomaly: 0 ms.): idle  [ 0, 0, 10, 0, 0, 0, ]
15:43:19.589 -> Predictions (DSP: 106 ms., Classification: 0 ms., Anomaly: 0 ms.): idle  [ 0, 0, 10, 0, 0, 0, ]
15:43:19.936 -> Predictions (DSP: 106 ms., Classification: 0 ms., Anomaly: 0 ms.): idle  [ 0, 0, 10, 0, 0, 0, ]
15:43:20.206 -> Predictions (DSP: 104 ms., Classification: 0 ms., Anomaly: 0 ms.): idle  [ 0, 0, 10, 0, 0, 0, ]
15:43:20.542 -> Predictions (DSP: 106 ms., Classification: 0 ms., Anomaly: 0 ms.): idle  [ 0, 0, 9, 0, 1, 0, ]
```

Conclusion:- Created and deployed ML model with sound based data on edge device