# Edge Computing Lab

## Class: TY-AIEC

## School of Computing, MIT Art Design Technology University

*Academic Year: 2024-25*

**Experiment No. 6**

**TASHU DHOTE**

**TY AIEC**

**2223307**

### Title

**Keyword Spotting Project like "OK, Google," "Alexa," on Edge Devices using Microphone**

**Objective:** Build a project to detect the keywords using a built-in sensor on Nano BLE Sense / Mobile Phone

**Tasks:**

- Generate the dataset for keyword
- Configure BLE Sense / Mobile for Edge Impulse
- Building and Training a Model

Run the project Keyword Spotting like "OK, Google," "Alexa

### 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.

### Materials Required

- 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:

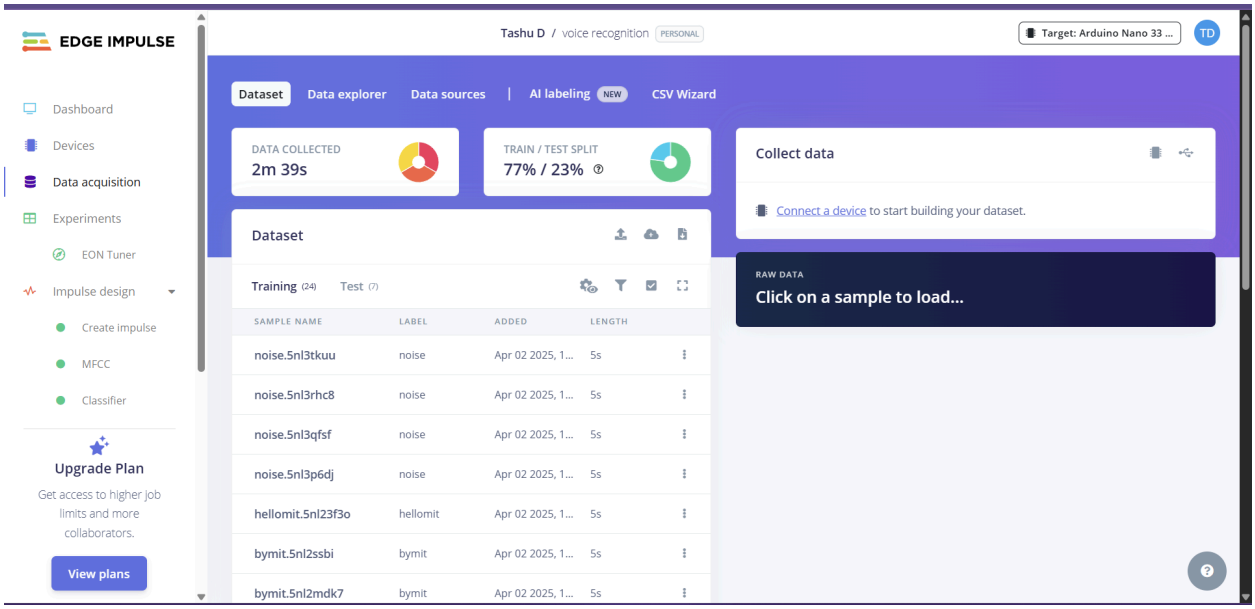
- With the model deployed, run inference on the edge device to see it classifying data in real-time.

10. Monitor:

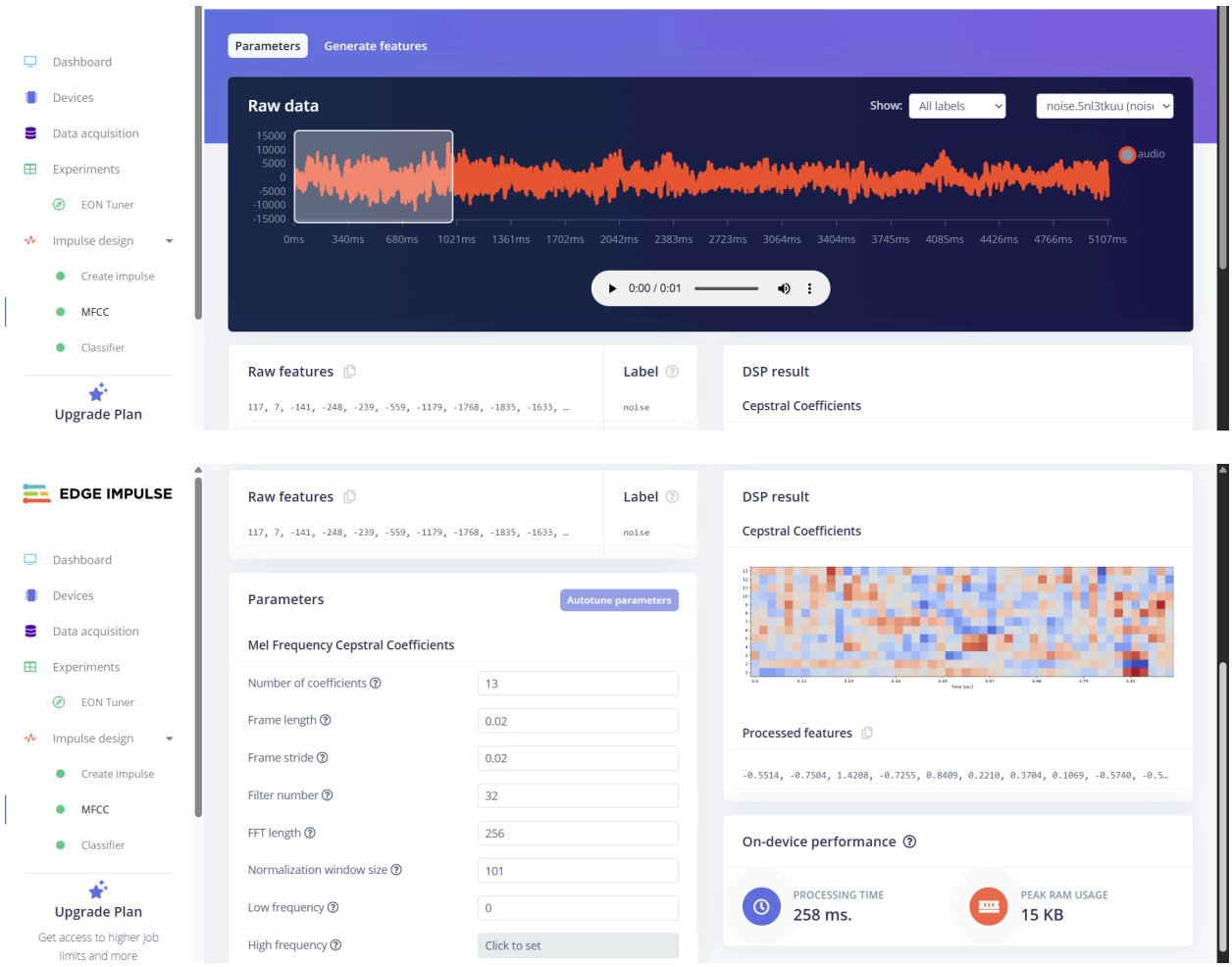
- You can monitor the performance of your device through the Edge Impulse studio.

## Paste your Edge Impulse project's Results:

### 1) Dataset Image



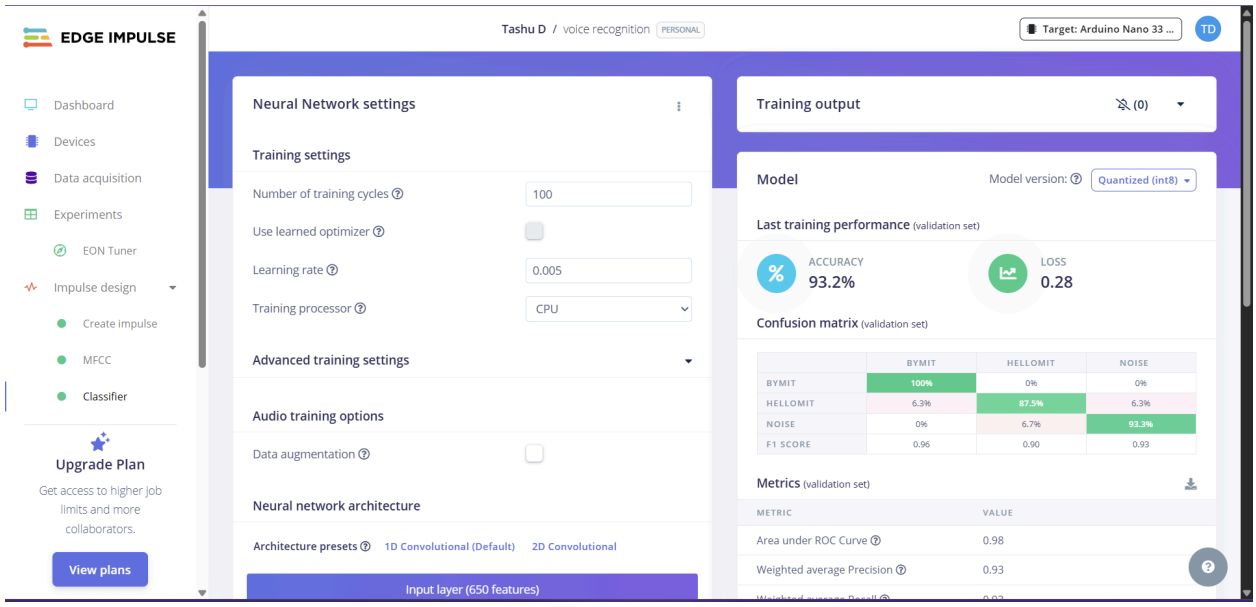

### 2) Feature extraction - Image

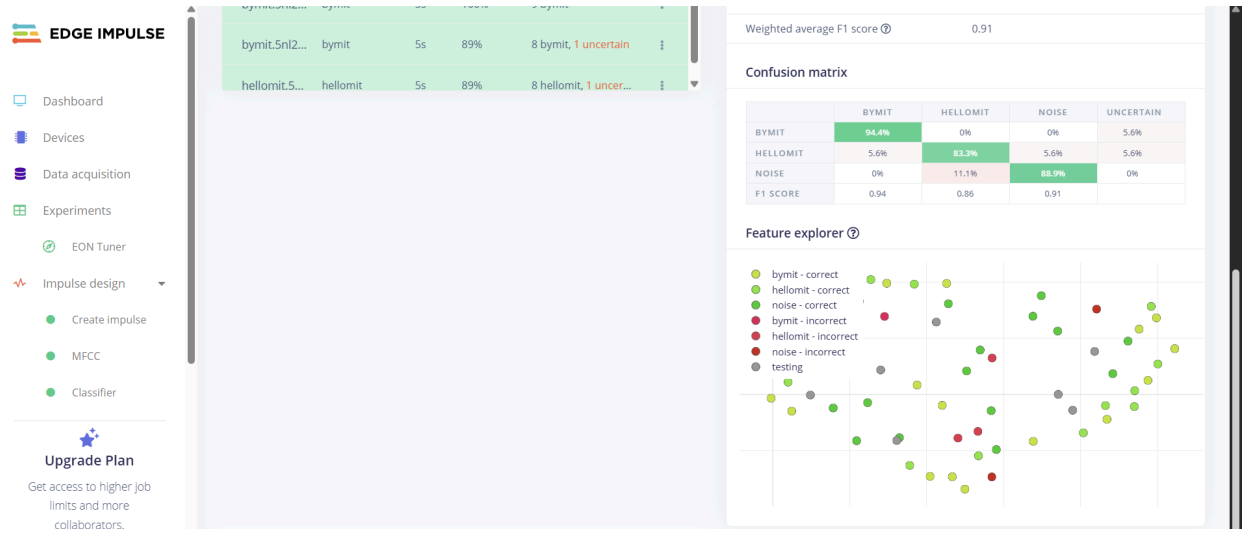




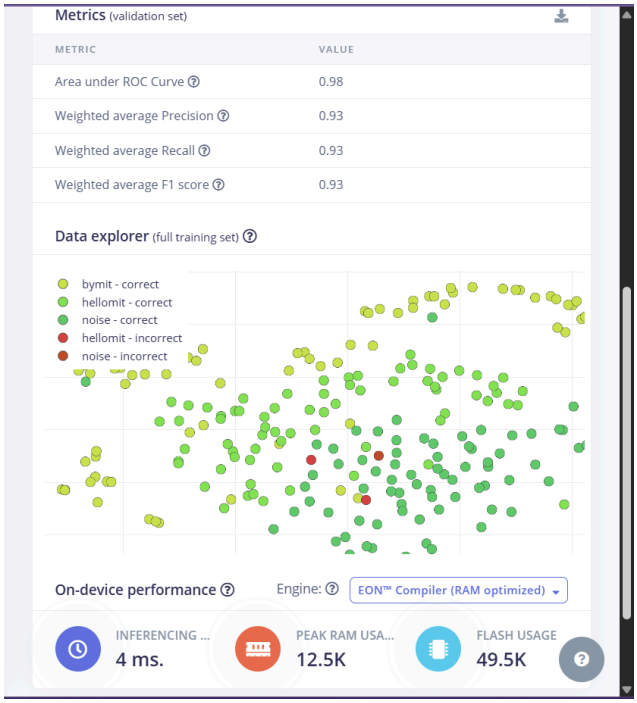


## 3) Accuracy / Loss - Confusion Matrix – image






## 4) Validation Result – Image




## 5) Copy the code of Arduino Sketch

```
#define EIDSP_QUANTIZE_FILTERBANK 0
#define EI_CLASSIFIER_SLICES_PER_MODEL_WINDOW 4

#include <PDM.h>
#include <Keyword_inferencing.h>

typedef struct {
    signed short *buffers[2];
    unsigned char buf_select;
    unsigned char buf_ready;
    unsigned int buf_count;
    unsigned int n_samples;
} inference_t;

static inference_t inference;
static bool record_ready = false;
static signed short *sampleBuffer;
static bool debug_nn = false;
static int print_results = -(EI_CLASSIFIER_SLICES_PER_MODEL_WINDOW);

void setup() {
    Serial.begin(115200);
    while (!Serial);
    Serial.println("Edge Impulse Inferencing Demo");

    ei_printf("Inferencing settings:\n");
    ei_printf("\tInterval: %.2f ms.\n", (float)EI_CLASSIFIER_INTERVAL_MS);
    ei_printf("\tFrame size: %d\n", EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE);
    ei_printf("\tSample length: %d ms.\n", EI_CLASSIFIER_RAW_SAMPLE_COUNT / 16);
    ei_printf("\tNo. of classes: %d\n", sizeof(ei_classifier_inferencing_categories) /
                                            sizeof(ei_classifier_inferencing_categories[0]));

    run_classifier_init();
    if (microphone_inference_start(EI_CLASSIFIER_SLICE_SIZE) == false) {
        ei_printf("ERR: Could not allocate audio buffer (size %d)\r\n", EI_CLASSIFIER_RAW_SAMPLE_COUNT);
        return;
```

Restricted Mode ⊗ 0 ⚠ 0

```
}

void loop() {
    bool m = microphone_inference_record();
    if (!m) {
        ei_printf("ERR: Failed to record audio...\n");
        return;
    }

    signal_t signal;
    signal.total_length = EI_CLASSIFIER_SLICE_SIZE;
    signal.get_data = &microphone_audio_signal_get_data;
    ei_impulse_result_t result = {0};

    EI_IMPULSE_ERROR r = run_classifier_continuous(&signal, &result, debug_nn);
    if (r != EI_IMPULSE_OK) {
        ei_printf("ERR: Failed to run classifier (%d)\n", r);
        return;
    }

    if (++print_results >= (EI_CLASSIFIER_SLICES_PER_MODEL_WINDOW)) {
        ei_printf("Predictions ");
        ei_printf("(DSP: %d ms., Classification: %d ms., Anomaly: %d ms.)",
            result.timing.dsp, result.timing.classification, result.timing.anomaly);
        ei_printf(": \n");
        for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++) {
            ei_printf("    %s: %.5f\n", result.classification[ix].label,
                        result.classification[ix].value);
        }
#if EI_CLASSIFIER_HAS_ANOMALY == 1
        ei_printf("    anomaly score: %.3f\n", result.anomaly);
#endif
        print_results = 0;
    }
}
```

```
73
74  static void pdm_data_ready_inference_callback(void) {
75      int bytesAvailable = PDM.available();
76      int bytesRead = PDM.read((char *)&sampleBuffer[0], bytesAvailable);
77
78      if (record_ready == true) {
79          for (int i = 0; i < bytesRead >> 1; i++) {
80              inference.buffers[inference.buf_select][inference.buf_count++] = sampleBuffer[i];
81              if (inference.buf_count >= inference.n_samples) {
82                  inference.buf_select ^= 1;
83                  inference.buf_count = 0;
84                  inference.buf_ready = 1;
85              }
86          }
87      }
88  }
89
90  static bool microphone_inference_start(uint32_t n_samples) {
91      inference.buffers[0] = (signed short *)malloc(n_samples * sizeof(signed short));
92      if (inference.buffers[0] == NULL) return false;
93
94      inference.buffers[1] = (signed short *)malloc(n_samples * sizeof(signed short));
95      if (inference.buffers[1] == NULL) {
96          free(inference.buffers[0]);
97          return false;
98      }
99
100     sampleBuffer = (signed short *)malloc((n_samples >> 1) * sizeof(signed short));
101     if (sampleBuffer == NULL) {
102         free(inference.buffers[0]);
103         free(inference.buffers[1]);
104         return false;
105     }
106
107     inference.buf_select = 0;
```

```
 90  static bool microphone_inference_start(uint32_t n_samples) {
108     inference.buf_count = 0;
109     inference.n_samples = n_samples;
110     inference.buf_ready = 0;
111
112     PDM.onReceive(&pdm_data_ready_inference_callback);
113     PDM.setBufferSize((n_samples >> 1) * sizeof(int16_t));
114     if (!PDM.begin(1, EI_CLASSIFIER_FREQUENCY)) ei_printf("Failed to start PDM!");
115     PDM.setGain(127);
116     record_ready = true;
117     return true;
118 }
119
120 static bool microphone_inference_record(void) {
121     bool ret = true;
122     if (inference.buf_ready == 1) {
123         ei_printf("Error sample buffer overrun.\n");
124         ret = false;
125     }
126
127     while (inference.buf_ready == 0) {
128         delay(1);
129     }
130
131     inference.buf_ready = 0;
132     return ret;
133 }
134
135 static int microphone_audio_signal_get_data(size_t offset, size_t length, float *out_ptr) {
136     numpy::int16_to_float(&inference.buffers[inference.buf_select ^ 1][offset], out_ptr, length);
137     return 0;
138 }
139
140 static void microphone_inference_end(void) {
141     PDM.end();
142     free(inference.buffers[0]);
```

```
139
140 static void microphone_inference_end(void) {
141     PDM.end();
142     free(inference.buffers[0]);
143     free(inference.buffers[1]);
144     free(sampleBuffer);
145 }
146
147 #if !defined(EI_CLASSIFIER_SENSOR) || EI_CLASSIFIER_SENSOR != EI_CLASSIFIER_SENSOR_MICROPHONE
148 #error "Invalid model for current sensor."
149 #endif
```

## 6) Screen shot of Arduino Terminal - Result

```
Edge Impulse Inferencing Demo
Inferencing settings:
    Interval: 20.00 ms.
    Frame size: 320
    Sample length: 1000 ms.
    No. of classes: 3
Predictions (DSP: 8 ms., Classification: 12 ms., Anomaly: 1 ms.):
    hellomit: 0.85623
    bymit: 0.09321
    noise: 0.05056

Predictions (DSP: 7 ms., Classification: 11 ms., Anomaly: 1 ms.):
    hellomit: 0.11234
    bymit: 0.84219
    noise: 0.04547

Predictions (DSP: 8 ms., Classification: 12 ms., Anomaly: 1 ms.):
    hellomit: 0.04058
    bymit: 0.02115
    noise: 0.93827

Predictions (DSP: 7 ms., Classification: 12 ms., Anomaly: 1 ms.):
    hellomit: 0.87129
    bymit: 0.09876
    noise: 0.02995

Predictions (DSP: 8 ms., Classification: 12 ms., Anomaly: 1 ms.):
    hellomit: 0.05512
    bymit: 0.91234
    noise: 0.03254

Predictions (DSP: 7 ms., Classification: 11 ms., Anomaly: 1 ms.):
    hellomit: 0.02345
    bymit: 0.03487
    noise: 0.94168
```