

Edge Computing Lab

Name : Rahul Bhati

Class: TY-AIEC

Roll no : 2223416

School of Computing, MIT Art Design Technology University

Academic Year: 2024-25

Experiment No. 6

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.

Edge Impulse project's Results:

The first screenshot shows the Edge Impulse studio interface for a project named 'Voice recognition' by 'Rahul Bhatti'. The 'Dataset' tab is active, displaying a table of collected data samples. The table has columns for 'SAMPLE NAME', 'LABEL', 'ADDED', and 'LENGTH'. The data shows 11 samples, all labeled 'noise', collected on April 01, 2025. The 'TRAIN / TEST SPLIT' is 77% / 23%. A 'Collect data' button is visible, along with a 'Connect a device' link. The 'RAW DATA' section shows a 'Click on a sample to load...' prompt.

The second screenshot shows the 'Impulse #1' configuration page. It displays a workflow for processing audio data. The 'Time series data' block is configured with 'Input axes: audio', 'Window size: 2,044 ms', 'Window increase (stride): 500 ms', and 'Frequency (Hz): 16,000'. The 'Audio (MFCC)' block is configured with 'Name: MFCC', 'Input axes (1): audio', and 'Signal: audio'. The 'Classification' block is configured with 'Name: Classifier', 'Input features: MFCC', and 'Output features: 3 (noise, off fan, on fan)'. The 'Output features' block shows the final output: '3 (noise, off fan, on fan)'. A 'Save impulse' button is visible at the bottom right.

EDGE IMPULSE

Dashboard

Devices

Data acquisition

Experiments

EON Tuner

Impulse design

Create impulse

MFCC

Classifier

Retrain model

Live classification

Model testing

Perf. calibration

Performance

Upgrade Plan
Get access to higher job limits and more collaborators.
[View plans](#)

Parameters

Generate features

Raw data

10000

5000

0

-5000

-10000

0ms

257ms

514ms

771ms

1029ms

1286ms

1543ms

1800ms

2058ms

2315ms

2572ms

2830ms

3087ms

3344ms

3601ms

3859ms

4116ms

4373ms

4631ms

4888ms

0.00 / 0.02

audio

Raw features

-118, 283, 3899, 1897, 3929, 1863, 1887, 1188, 974, 818, 544, 394, 388, 344, 284, 222, 378, 3...

Label

noise_5103V4H0_016

Parameters

Mel Frequency Cepstral Coefficients

Number of coefficients

13

Frame length

0.02

Frame stride

0.02

Filter number

32

FFT length

256

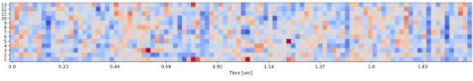
Normalization window size

101

Autotune parameters

DSP result

Cepstral Coefficients



Processed features

-8.5218, -8.6588, -8.9962, 1.4614, -8.2489, -8.7889, 1.8858, 1.9415, 1.3438, 8.1895, 8.8126, -8.4488, -8.5448, -8.5813,...

On-device performance

PROCESSING TIME

PEAK RAM USAGE

sketch_apr8a | Arduino IDE 2.3.5

File Edit Sketch Tools Help

Arduino Nano 33 BLE

sketch_apr8a.ino

```
17 static Inference_t inferences;
18 static bool record_ready = false;
19 static signed short *sampleBuffer;
20 static bool debug_nn = false;
21 static int print_results = -(EI_CLASSIFIER_SLICES_PER_MODEL_WINDOW);
22
23 void setup()
24 {
25     Serial.begin(115200);
26     while (!Serial);
27
28     Serial.println("Voice control: ON FAN / OFF FAN");
29
30     pinMode(LED_PIN, OUTPUT);
31     digitalWrite(LED_PIN, LOW); // LED OFF initially
32
33     run_classifier_init();
34
35     if (microphone_inference_start(EI_CLASSIFIER_SLICE_SIZE) == false) {
36         Serial.println("ERR: could not allocate audio buffer");
37         return;
38     }
39 }
```

Output

[=====] 78% (22/41 pages)
[=====] 80% (33/41 pages)
[=====] 82% (34/41 pages)
[=====] 85% (35/41 pages)
[=====] 87% (36/41 pages)
[=====] 90% (37/41 pages)
[=====] 92% (38/41 pages)
[=====] 95% (39/41 pages)
[=====] 97% (40/41 pages)
[=====] 100% (41/41 pages)
Done in 6.694 seconds

Ln 152, Col 1 | Arduino Nano 33 BLE on COM9 | 2

```
nano_ble33_sense_microphone_continuous | Arduino IDE 2.3.5
File Edit Sketch Tools Help
Arduino Nano 33 BLE
nano_ble33_sense_microphone_continuous.ino
57 // static bool debug_nn = true; // Set this to true to see e.g. features generated from the raw signal
58 static bool debug_nn = false; // Set this to true to see e.g. features generated from the raw signal
59 static int print_results = -(EI_CLASSIFIER_SLICES_PER_MODEL_WINDOW);
60
61 /**
62  * @brief Arduino setup function
63  */
64 void setup()
65 {
66     // put your setup code here, to run once:
67     Serial.begin(115200);
68     // comment out the below line to cancel the wait for USB connection (needed for native USB)
69     while (!Serial);
70     Serial.println("Edge Impulse Inferencing Demo");
71
72     // summary of inferencing settings (from model_metadata.h)
73     ei_printf("Inferencing settings:\n");
74     ei_printf("\tInterval: %.2f ms.\n", (float)EI_CLASSIFIER_INTERVAL_MS);
75     ei_printf("\tFrame size: %d\n", EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE);
76     ei_printf("\tSample length: %d ms.\n", EI_CLASSIFIER_RAW_SAMPLE_COUNT / 16);
77     ei_printf("\tNo. of classes: %d\n", sizeof(ei_classifier_inferencing_categories) /
78         sizeof(ei_classifier_inferencing_categories[0]));
79 }
80
81 // Predictions (DSP: 148 ms., Classification: 11 ms., Anomaly: 0 ms.):
82 // off fan: 0.00000
83 // on fan: 0.00000
84 // noise: 0.00391
85 // off fan: 0.94141
86 // on fan: 0.05169
87 // Predictions (DSP: 148 ms., Classification: 11 ms., Anomaly: 0 ms.):
88 // off fan: 0.99609
89 // on fan: 0.00391
90 // noise: 0.00000
```