Edge Computing Lab

Class: TY-AIEC

School of Computing, MIT Art Design Technology University

Academic Year: 2024-25

Experiment No. 9

Name: Viresh Kamlapure
Class: TY AIEC
Enrollment No: MITU23BTCSD082
Roll No: D2233082

Introduction

The Object Detection using Camera on Edge Computing Devices

Objective: Build a project to detect an object using Edge Computing

Tasks:

- Generate the dataset for customized object
- Configure Edge Impulse for Object Detection
- Building and Training a Model
- Deploy on Edge Computing Device

Introduction

Edge Impulse is a development platform for machine learning on edge devices, targeted at developers who want to create intelligent device solutions. The " Camera "sensor reading 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:

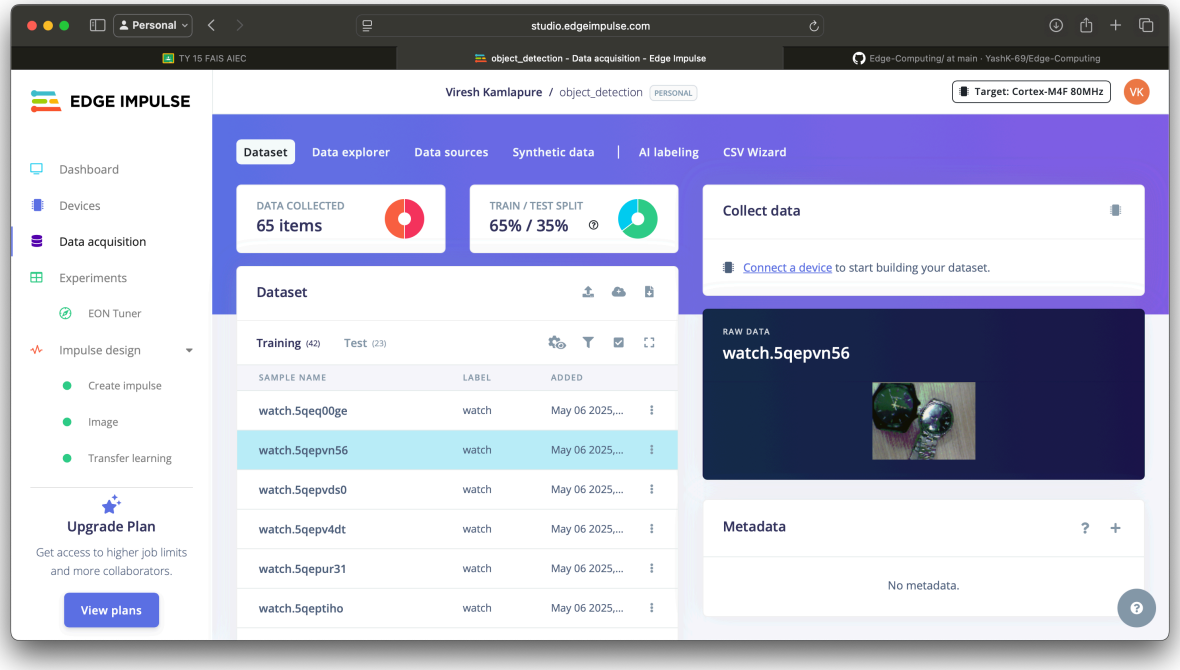
- With the model deployed, run inference on the edge device to see it classifying data in real-time.

10. Monitor:

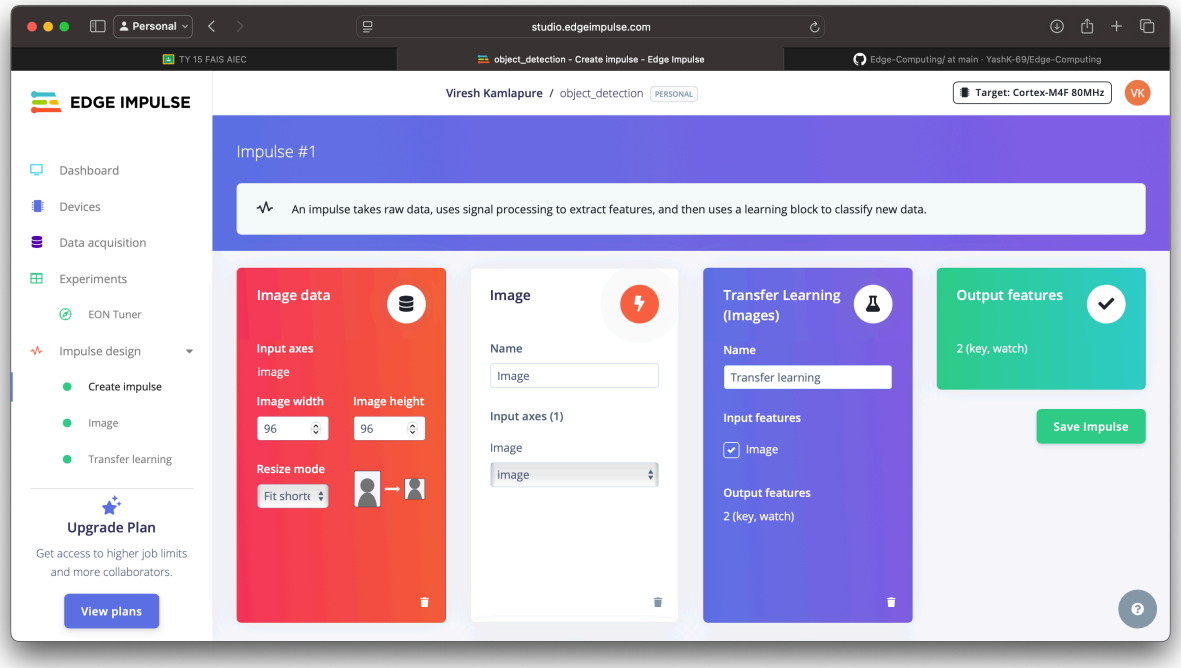
- You can monitor the performance of your device through the Edge Impulse studio.

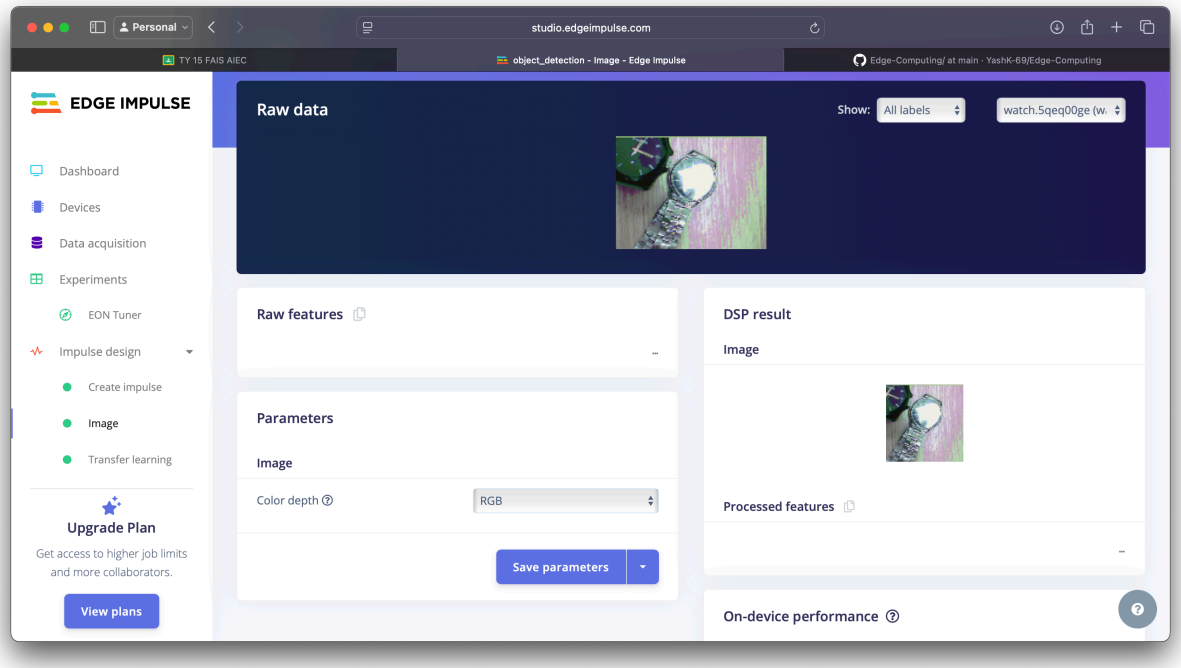
Paste your Edge Impulse project's Results:

1) Dataset Image

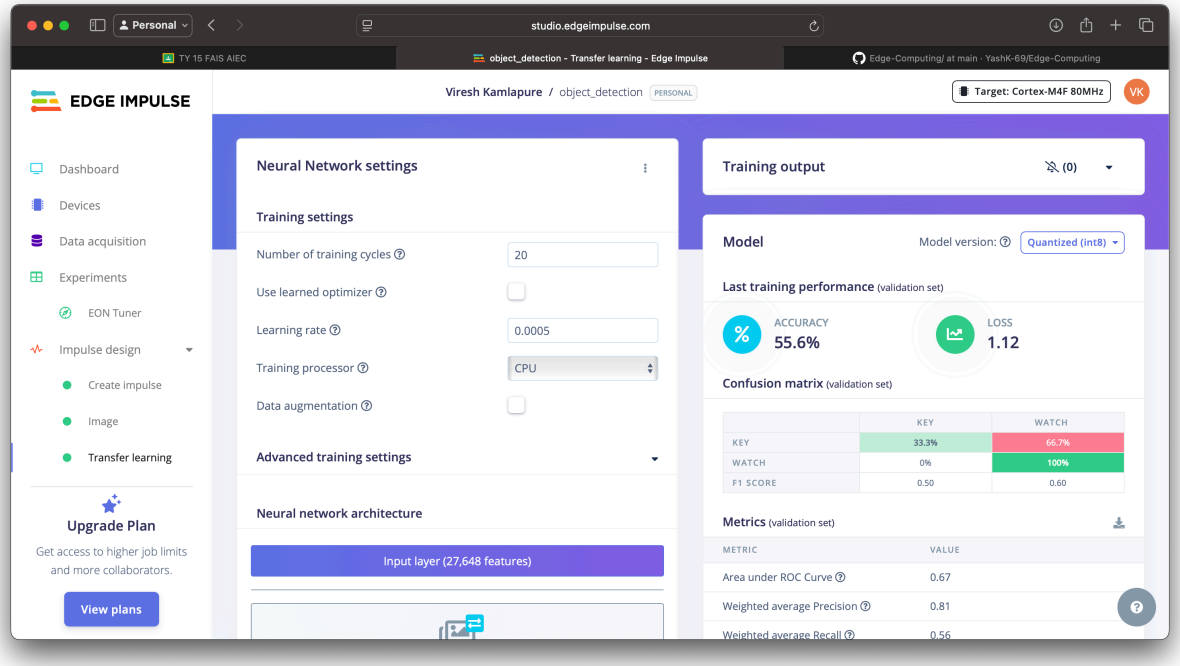


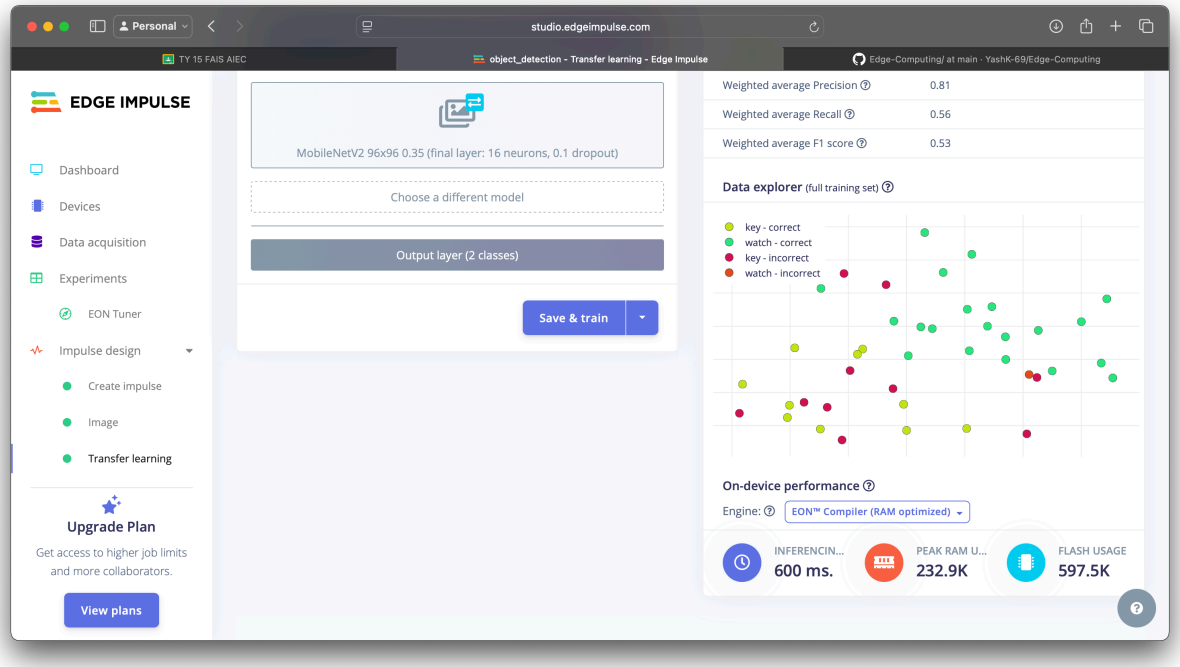
2) Feature extraction - Image



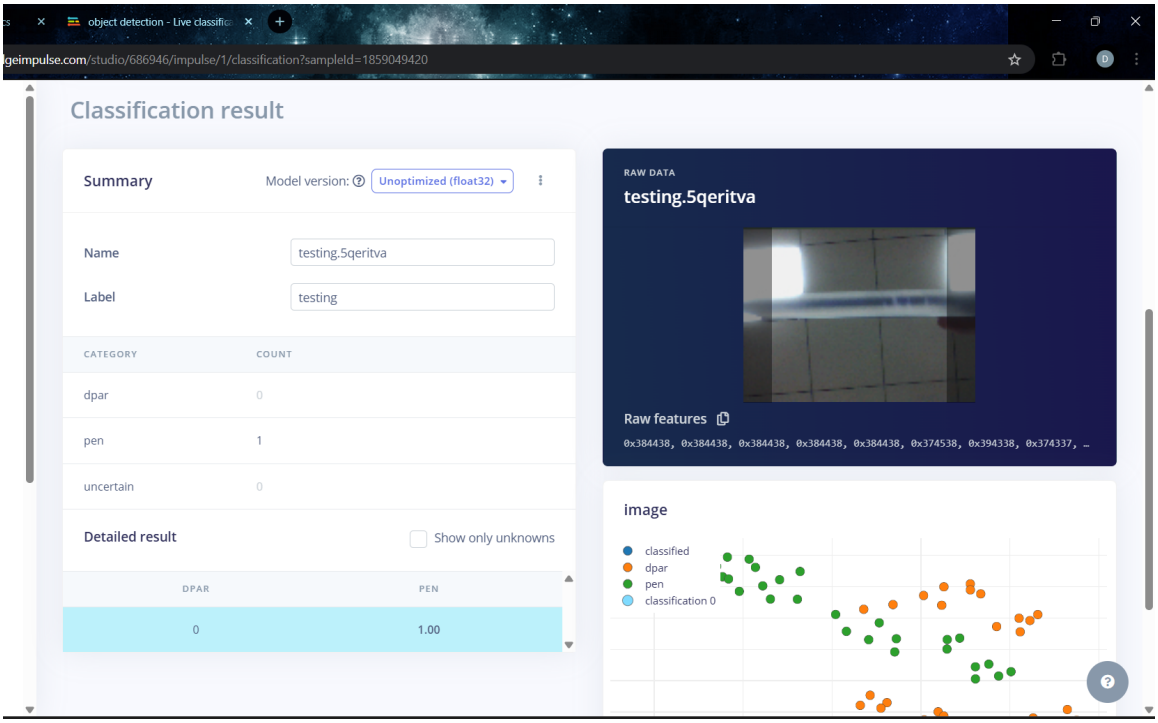


3) Accuracy / Loss - Confusion Matrix – image

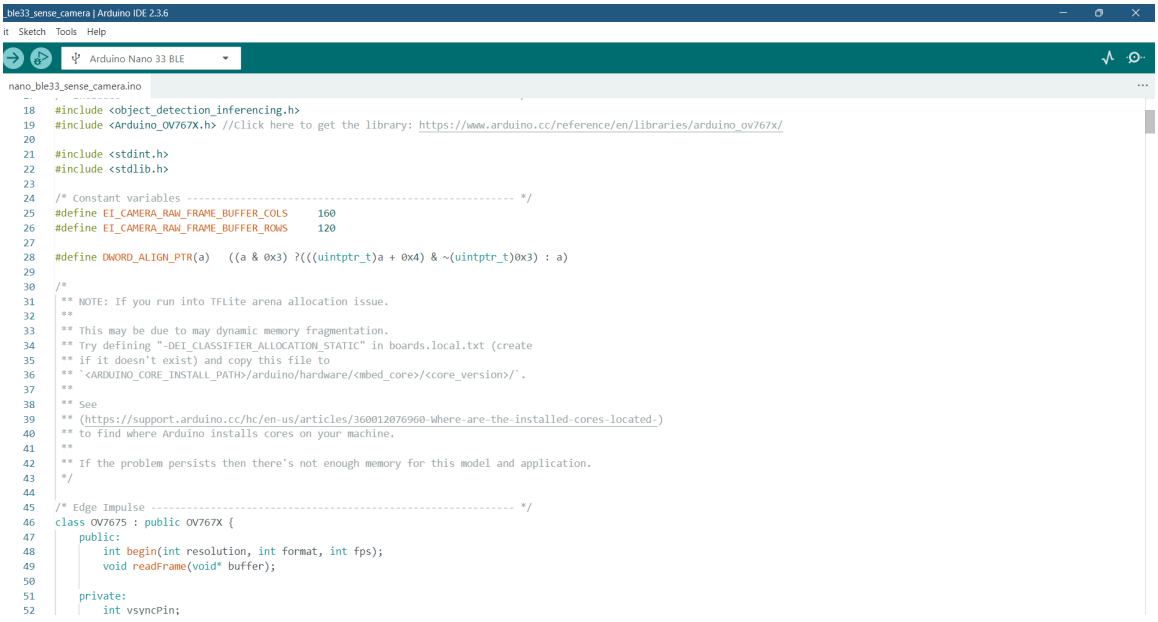


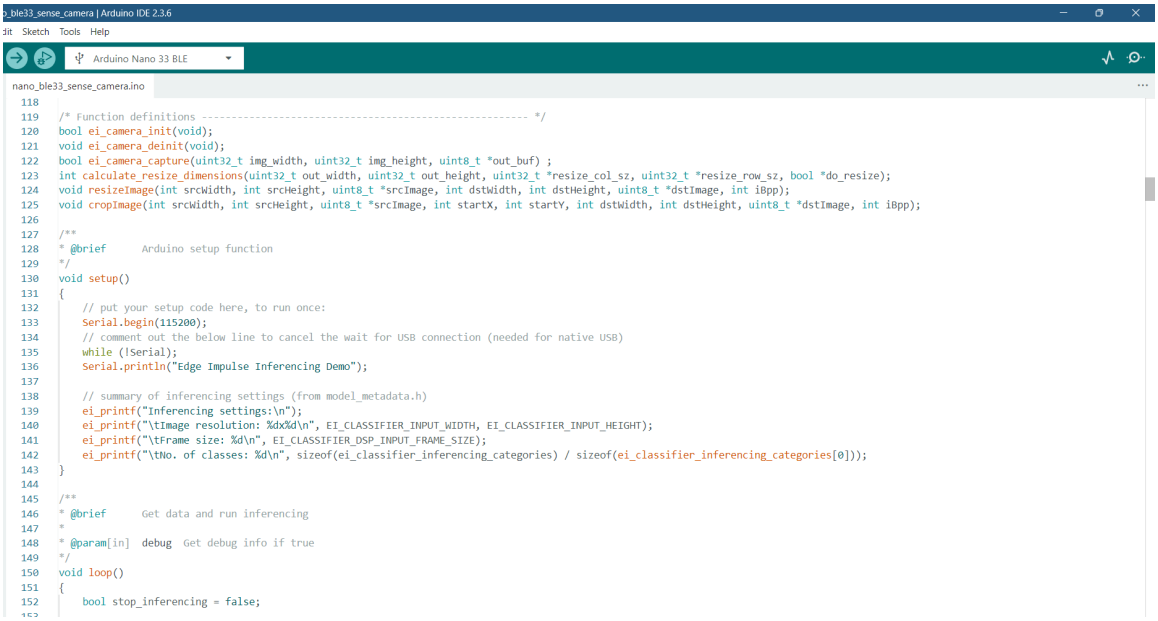


4) Validation Result – Image



5) Copy the code of Arduino Sketch


_ble33_sense_camera | Arduino IDE 2.3.6
it Sketch Tools Help
Arduino Nano 33 BLE
nano_ble33_sense_camera.ino
#include <object_detection_inferencing.h>
#include <Arduino_OV767X.h> //Click here to get the library: https://www.arduino.cc/reference/en/libraries/arduino_ov767x/
#include <stdint.h>
#include <stdlib.h>
/* Constant variables --- */
#define EI_CAMERA_RAW_FRAME_BUFFER_COLS 160
#define EI_CAMERA_RAW_FRAME_BUFFER_ROWS 120
#define DWORD_ALIGN_PTR(a) ((a & 0x3) ?(((uintptr_t)a + 0x4) & ~(uintptr_t)0x3) : a)
/*
** 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.
*/
/* Edge Impulse --- */
class OV7675 : public OV767X {
public:
int begin(int resolution, int format, int fps);
void readFrame(void* buffer);
private:
int vsyncPin;


_ble33_sense_camera | Arduino IDE 2.3.6
dit Sketch Tools Help
Arduino Nano 33 BLE
nano_ble33_sense_camera.ino
/* Function definitions --- */
bool ei_camera_init(void);
void ei_camera_deinit(void);
bool ei_camera_capture(uint32_t img_width, uint32_t img_height, uint8_t *out_buf) ;
int calculate_resize_dimensions(uint32_t out_width, uint32_t out_height, uint32_t *resize_col_sz, uint32_t *resize_row_sz, bool *do_resize);
void resizeImage(int srcWidth, int srcHeight, uint8_t *srcImage, int dstWidth, int dstHeight, uint8_t *dstImage, int iBpp);
void cropImage(int srcWidth, int srcHeight, uint8_t *srcImage, int startX, int startY, int dstWidth, int dstHeight, uint8_t *dstImage, int iBpp);
/**
* @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");
// summary of inferencing settings (from model_metadata.h)
ei_printf("Inferencing settings:\n");
ei_printf("\tImage resolution: %dx%d\n", EI_CLASSIFIER_INPUT_WIDTH, EI_CLASSIFIER_INPUT_HEIGHT);
ei_printf("\tFrame size: %d\n", EI_CLASSIFIER_DSP_INPUT_FRAME_SIZE);
ei_printf("\tNo. of classes: %d\n", sizeof(ei_classifier_inferencing_categories) / sizeof(ei_classifier_inferencing_categories[0]));
}
/**
* @brief Get data and run inferencing
*
* @param[in] debug Get debug info if true
*/
void loop()
{
bool stop_inferencing = false;

o_ble33_sense_camera | Arduino IDE 2.3.6

dit Sketch Tools Help

Arduino Nano 33 BLE

nano_ble33_sense_camera.ino

```
                bb.height);
        }

    // Print the prediction results (classification)
#else
        ei_printf("Predictions:\r\n");
        for (uint16_t i = 0; i < EI_CLASSIFIER_LABEL_COUNT; i++) {
            ei_printf("  %s: ", ei_classifier_inferencing_categories[i]);
            ei_printf("%.5f\r\n", result.classification[i].value);
        }
#endif

    // Print anomaly result (if it exists)
#if EI_CLASSIFIER_HAS_ANOMALY
        ei_printf("Anomaly prediction: %.3f\r\n", result.anomaly);
#endif

#if EI_CLASSIFIER_HAS_VISUAL_ANOMALY
        ei_printf("Visual anomalies:\r\n");
        for (uint32_t i = 0; i < result.visual_ad_count; i++) {
            ei_impulse_result_bounding_box_t bb = result.visual_ad_grid_cells[i];
            if (bb.value == 0) {
                continue;
            }
            ei_printf("  %s (%f) [ x: %u, y: %u, width: %u, height: %u ]\r\n",
                    bb.label,
                    bb.value,
                    bb.x,
                    bb.y,
                    bb.width,
                    bb.height);
        }
#endif

        while (ei_get_serial_available() > 0) {
            if (ei_get_serial_byte() == 'b') {
```

o_ble33_sense_camera | Arduino IDE 2.3.6

dit Sketch Tools Help

Arduino Nano 33 BLE

nano_ble33_sense_camera.ino

```
//
// Extends the OV767X library function. Reads buf_rows VGA rows from the
// image sensor.
//
void OV7675::readBuf()
{
    int offset = 0;

    uint32_t ulPin = 33; // P1.xx set of GPIO is in 'pin' 32 and above
    NRF_GPIO_Type * port;

    port = nrf_gpio_pin_port_decode(&ulPin);

    for (int i = 0; i < buf_rows; i++) {
        // rising edge indicates start of line
        while ((*hrefPort & hrefMask) == 0); // wait for HIGH

        for (int col = 0; col < bytes_per_row; col++) {
            // rising edges clock each data byte
            while ((*pclkPort & pclkMask) != 0); // wait for LOW

            uint32_t in = port->IN; // read all bits in parallel

            in >>= 2; // place bits 0 and 1 at the "bottom" of the register
            in &= 0x3f03; // isolate the 8 bits we care about
            in |= (in >> 6); // combine the upper 6 and lower 2 bits

            raw_buf[offset++] = in;

            while ((*pclkPort & pclkMask) == 0); // wait for HIGH
        }

        while ((*hrefPort & hrefMask) != 0); // wait for LOW
    }
} /* OV7675::readBuf() */
```

6) Screen shot of Arduino Terminal - Result

```
 *
 */

/* Includes ---------------------------------------------------------------- */
#include <object_detection_inferencing.h>
#include <Arduino_OV767X.h> //Click here to get the library: https://www.arduino.cc/reference/en/libraries/arduino_ov767x/

#include <stdint.h>
#include <stdlib.h>
```

Output Serial Monitor

Message (Enter to send message to 'Arduino Nano 33 BLE' on 'COM3') New Line 9600 baud

```
13:31:48.450 -> Edge Impulse Inferencing Demo
13:31:48.450 -> Inferencing settings:
13:31:48.450 ->         Image resolution: 96x96
13:31:48.450 ->         Frame size: 9216
13:31:48.450 ->         No. of classes: 2
13:31:48.450 ->
13:31:48.450 -> Starting inferencing in 2 seconds...
13:31:50.432 -> Taking photo...
```