

ESP32-S3 STICK with TinyML

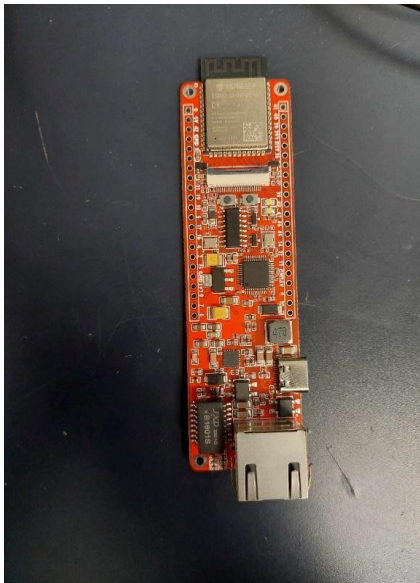
INTRODUCTION:

Hello everyone! In this tutorial, I will demonstrate how to utilize our pre-trained machine learning model with custom data. Specifically, we will be using Edge Impulse to detect the colors. In our previous tutorial, we relied on a PC and a Python script for face detection. However, in this tutorial, we'll shift to using S3 SoC instead of a PC.

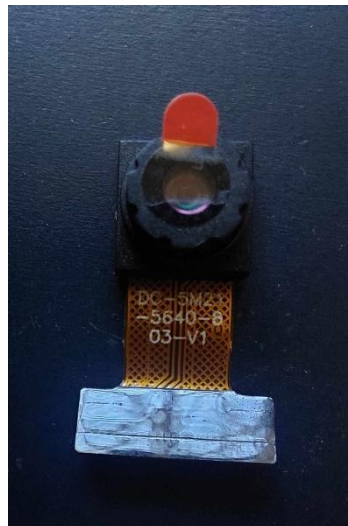
List of hardware you need:

- 1x USB-C cable
- ESP32-S3 stick
- Camera Ov2640 or Ov5640

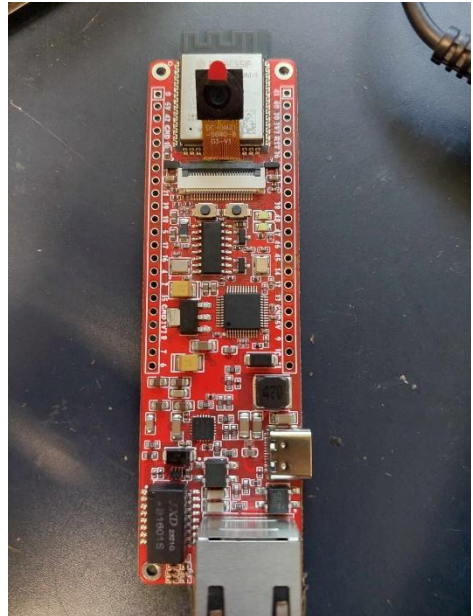
ESP32 STICK



Ov5640 camera



Connected camera to stick

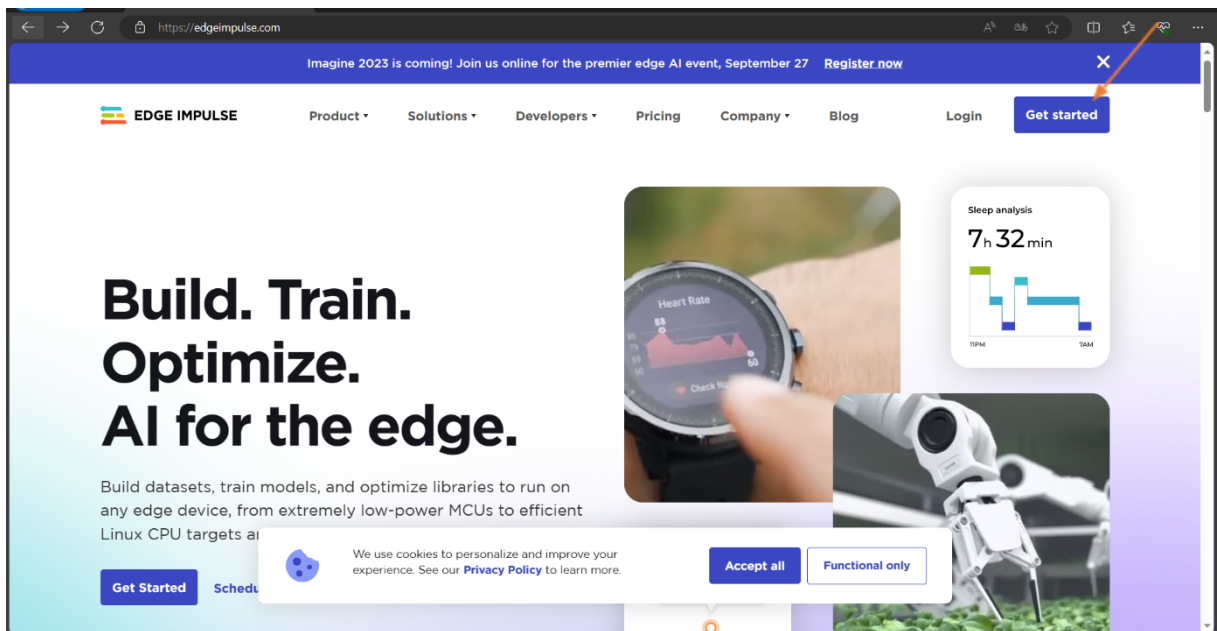


1. Software

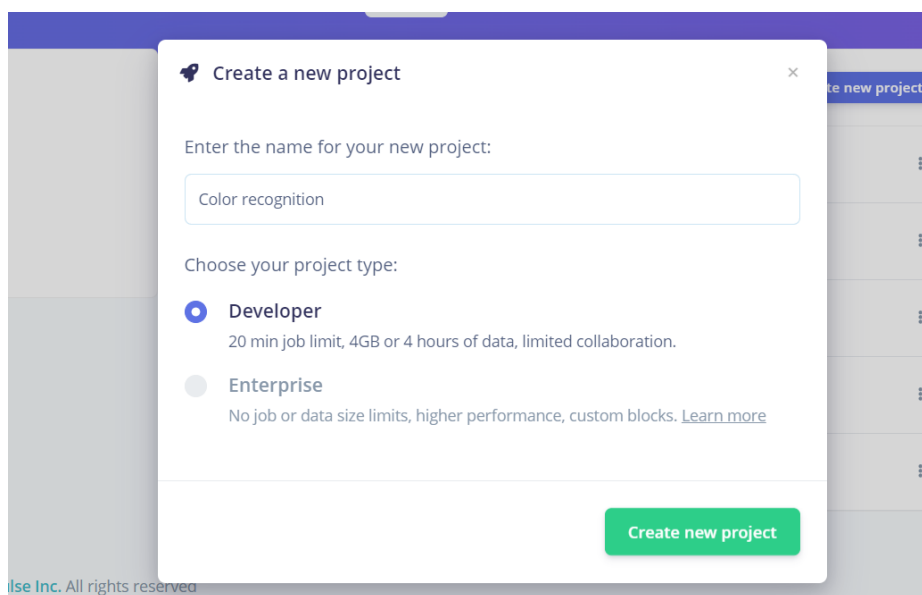
To set up TinyML, we'll be using the Arduino IDE. While we could use VScode as we did in the previous tutorial, Arduino IDE offers a simpler approach because it allows us to include libraries in the form of ZIP files. Additionally, please note that for our pre-trained model, you'll need to register on the Edge Impulse website.

2. Registration

On Edge Impulse website click on button 'Get started' and register

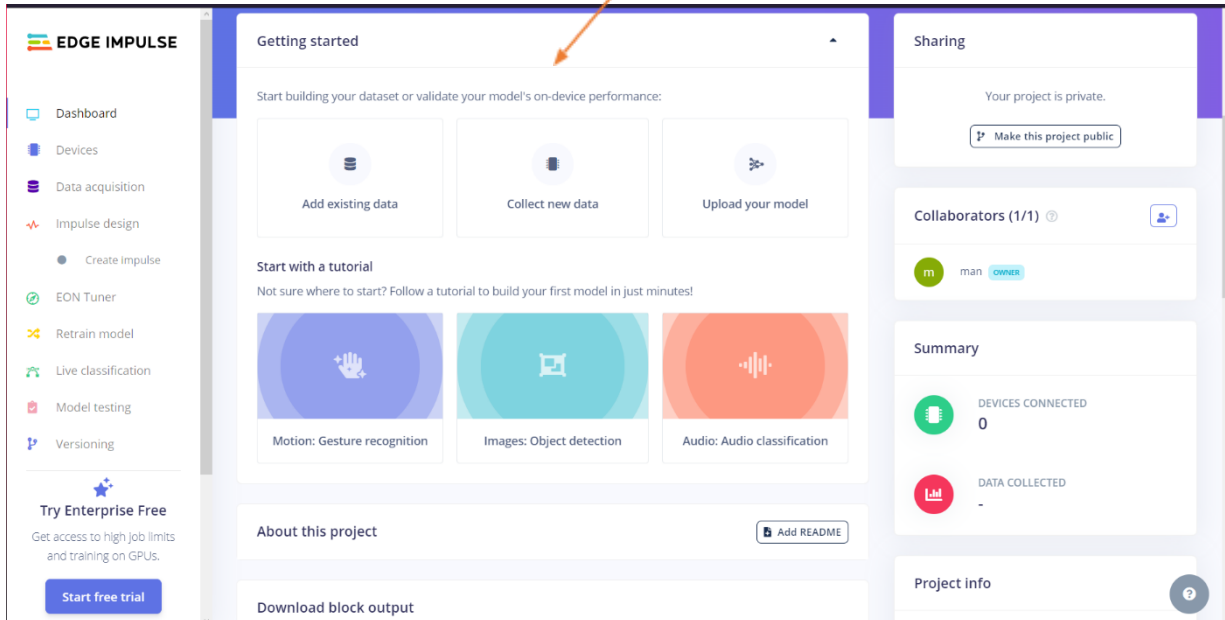


Create a project, enter the name of a project for me it's 'color recognition'



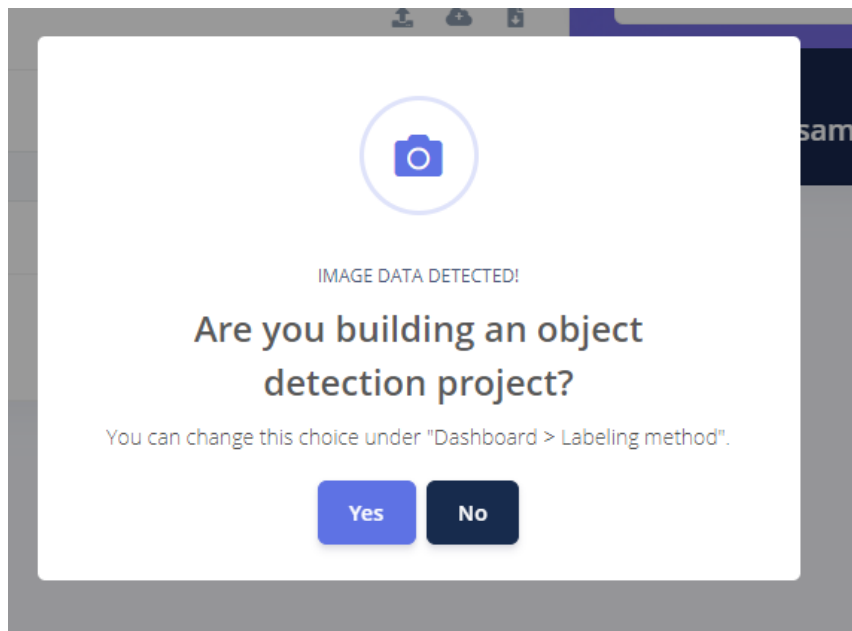
3. Data

After creating project you should be on dashboard, click on 'Collect new data', if you see window with collecting data by computer, phone or board close it, but if you want to you can. Click on 'add data' and 'upload data'.

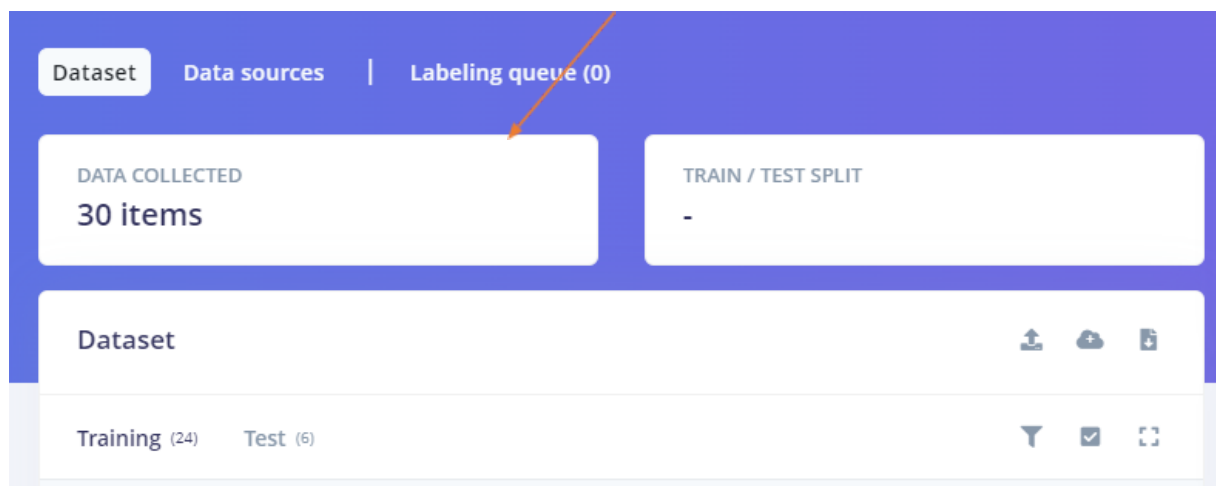


If

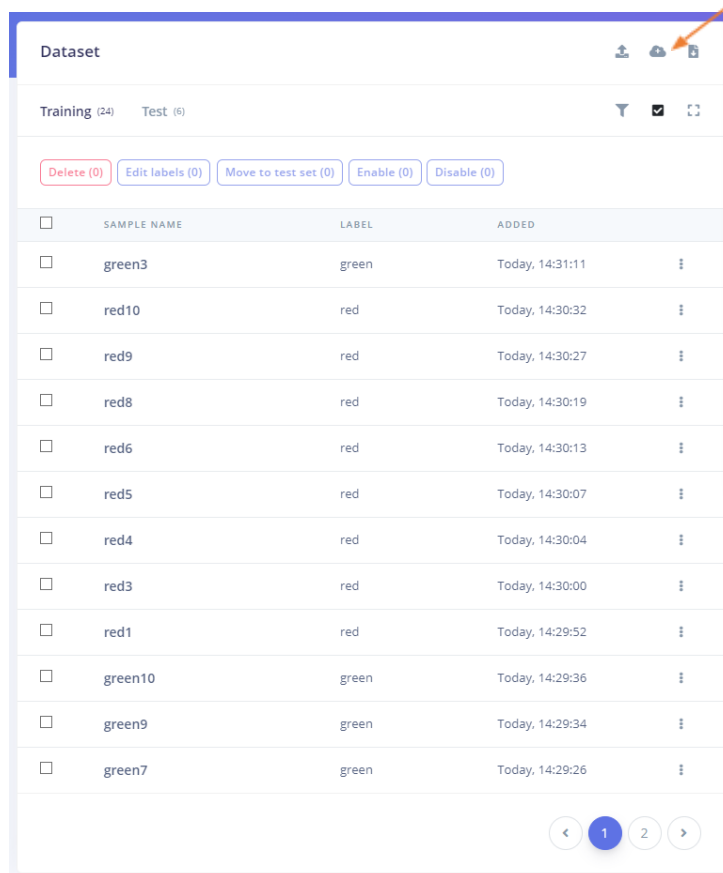
you see this while uploading data click 'No'



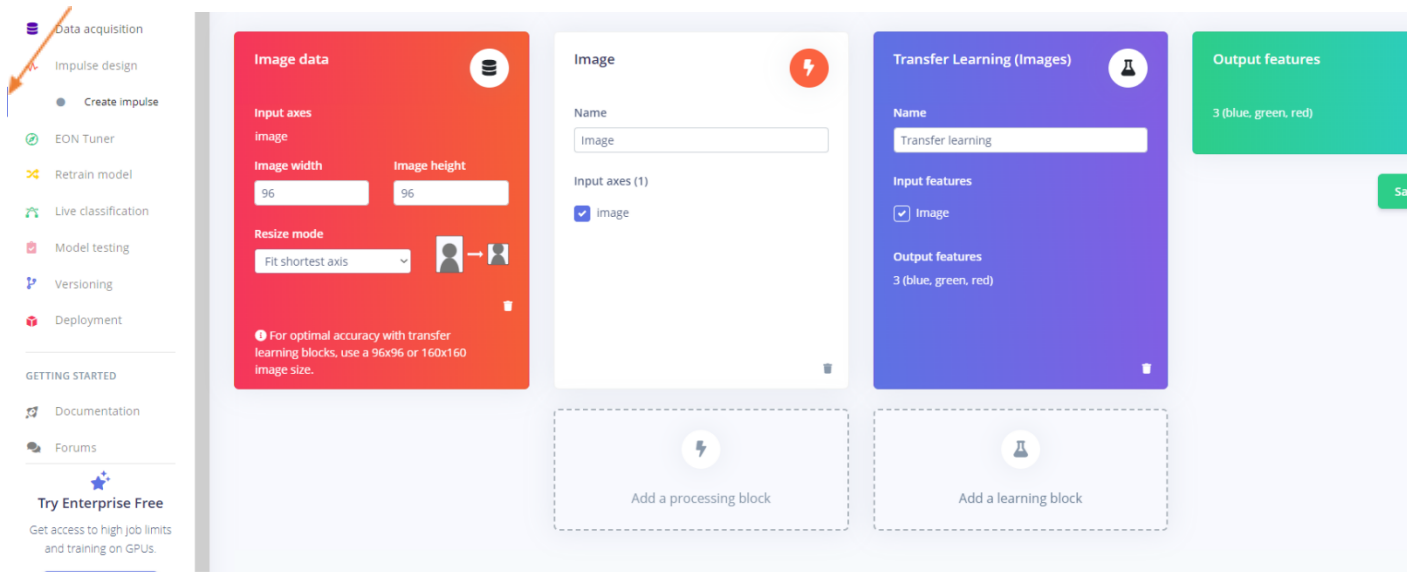
You need to save photo in Labeling queue



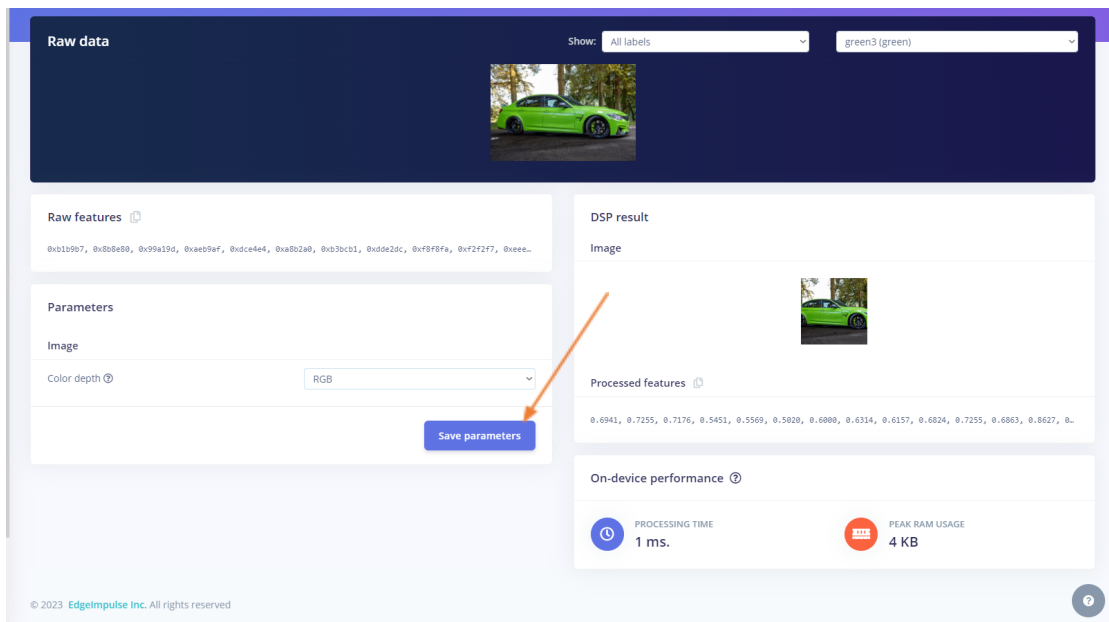
Once you have uploaded your data, click on the icon that allows you to select multiple items. From there, you can edit the names of the labels, such as "red," "green," and "blue." If you wish to create a testing dataset, you can move two photos of each color into the testing dataset.



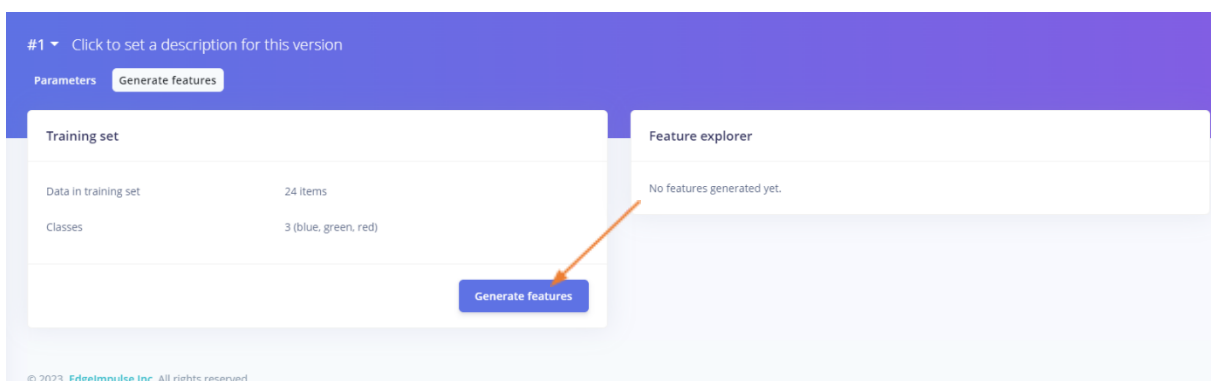
Next on sidebar click on 'Create impulse'



Save parameters, color RGB



Generate features, and after generating click on sidebar 'Transfer learning'



After that select target for ESP-EYE because it's similar to our stick and click 'start training'

The screenshot shows a training configuration page. At the top right, a dropdown menu is set to 'Target: espressif ESP-EYE (ESP32 240MHz)'. The left sidebar contains sections for 'Neural Network settings', 'Training settings', 'Advanced training settings', and 'Neural network architecture'. Under 'Training settings', 'Number of training cycles' is 20 and 'Learning rate' is 0.0005. Under 'Neural network architecture', the 'Input layer' has 27,648 features, the model is 'MobileNetV2 96x96 0.35 (final layer: 16 neurons, 0.1 dropout)', and the 'Output layer' has 3 classes. A green 'Start training' button is at the bottom.

#1 Click to set a description for this version

Target: espressif ESP-EYE (ESP32 240MHz)

Neural Network settings

Training settings

Number of training cycles 20

Learning rate 0.0005

Data augmentation ☐

Advanced training settings

Neural network architecture

Input layer (27,648 features)

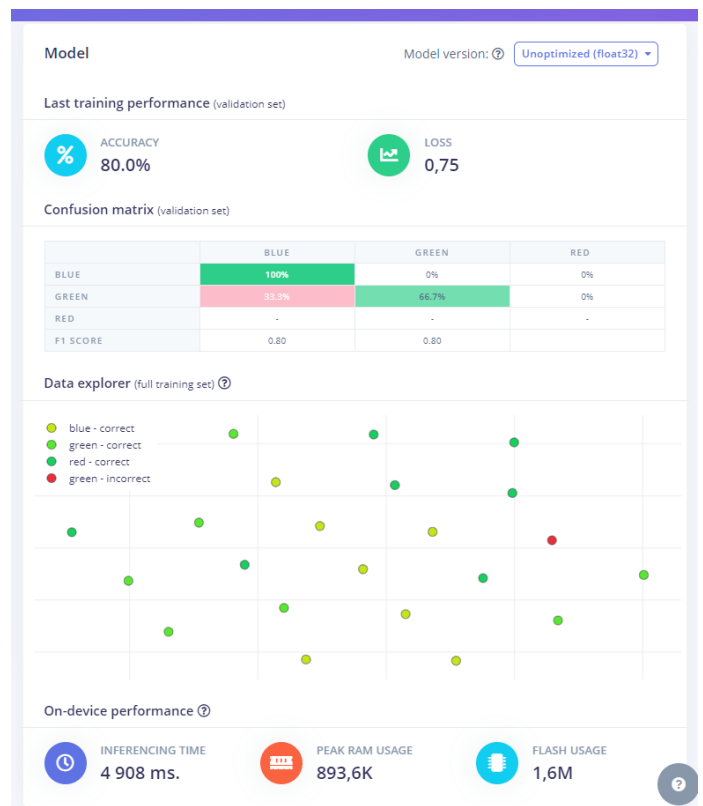
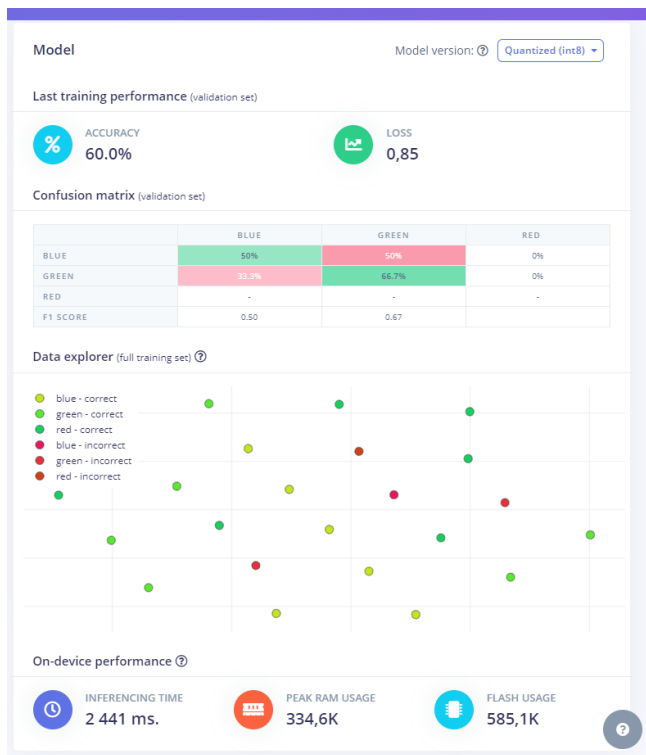
MobileNetV2 96x96 0.35 (final layer: 16 neurons, 0.1 dropout)

Choose a different model

Output layer (3 classes)

Start training

Accuracy is 60% for optimized and 80% for unoptimized but **inferencing time** is twice as much as optimized. Accuracy could be better, but for training we had only 24 pictures and for testing 6 pictures.




In sidebar click on 'Deployment', choose Arduino library, turn off EON compiler because for ESP32-S3 Edge Impulse hasn't released SDK yet and click on build

Configure your deployment

You can deploy your impulse to any device. This makes the model run without an internet connection, minimizes latency, and runs with minimal power consumption. [Read more.](#)

Arduino library ✕

 **SELECTED DEPLOYMENT**
Arduino library
An Arduino library with examples that runs on most Arm-based Arduino development boards.

MODEL OPTIMIZATIONS

Model optimizations can increase on-device performance but may reduce accuracy.

☐ **Enable EON™ Compiler** *Same accuracy, up to 50% less memory. [Learn more](#)*

Quantized (int8)
Selected ✓

	IMAGE	TRANSFER LEARNING	TOTAL
LATENCY	15 ms.	2 441 ms.	2 456 ms.
RAM	4.0K	353.1K	353.1K
FLASH	-	683.4K	-
ACCURACY			-

Unoptimized (float32)
Select

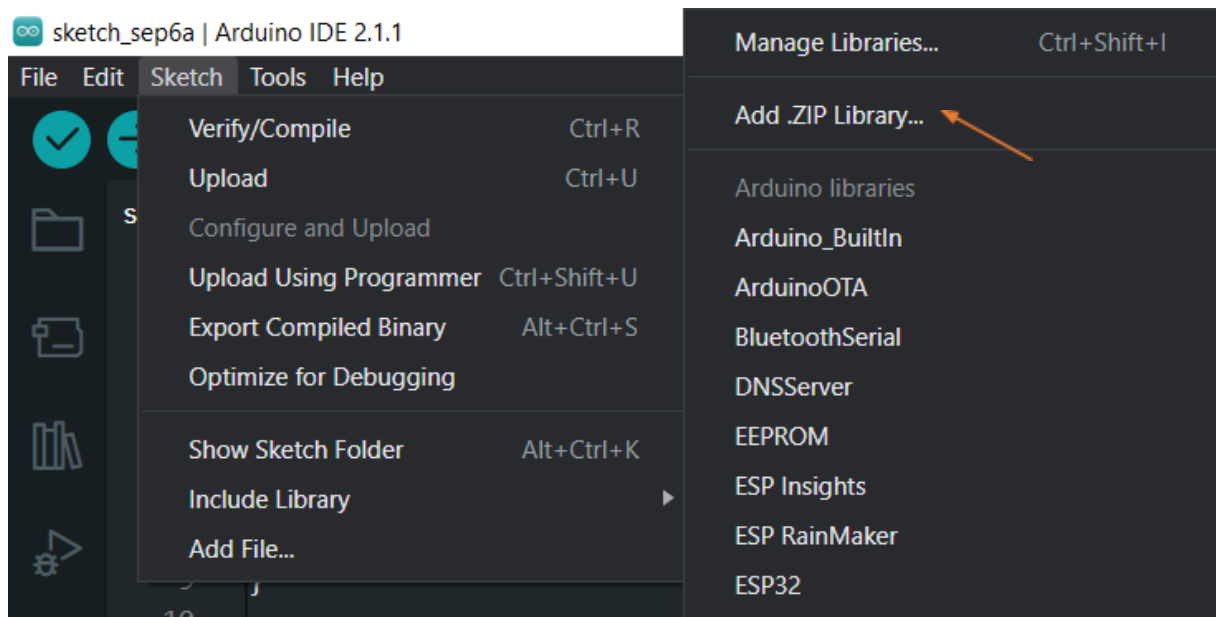
	IMAGE	TRANSFER LEARNING	TOTAL
LATENCY	15 ms.	4 908 ms.	4 923 ms.
RAM	4.0K	1.1M	1.1M
FLASH	-	1.7M	-
ACCURACY			-

To compare model accuracy, run model testing for all available optimizations. [Run model testing](#)

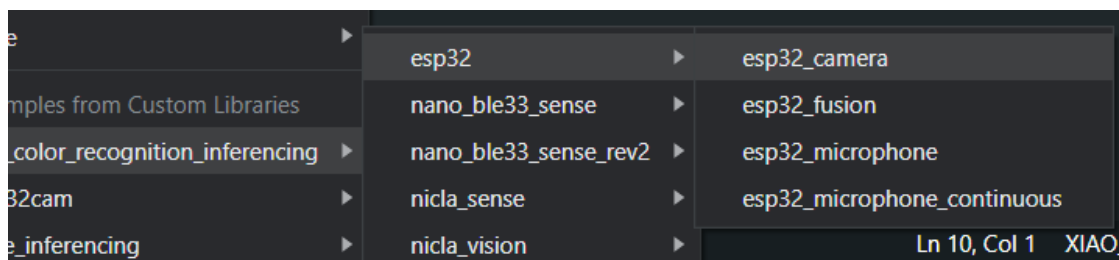
Estimate for Espressif ESP-EYE (ESP32 240MHz) - [Change target](#)

Build

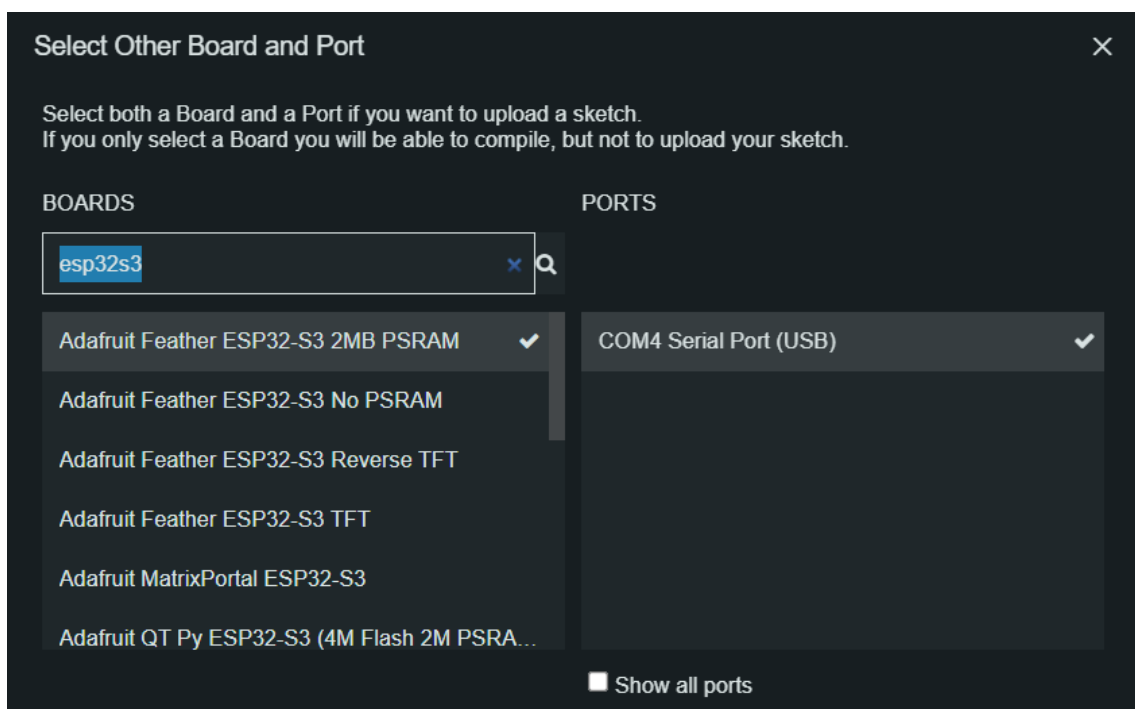
After downloading zip file include file in Arduino IDE



After library is installed go to File>Examples and scroll to the bottom for examples from custom libraries. Library name should be same name as project and choose camera



You can select this board with PSRAM or XIAO_ESP32S3, it's same SoC



You must configurate pins, in this screenshot you can delete it and set up like this below:

```
#define CAMERA_MODEL_ESP_EYE // Has PSRAM
// #define CAMERA_MODEL_AI_THINKER // Has PSRAM
#if defined(CAMERA_MODEL_ESP_EYE)
#define PWDN_GPIO_NUM -1
#define RESET_GPIO_NUM -1
#define XCLK_GPIO_NUM 4
#define SIOD_GPIO_NUM 18
#define SIOC_GPIO_NUM 23
#define Y9_GPIO_NUM 36
#define Y8_GPIO_NUM 37
#define Y7_GPIO_NUM 38
#define Y6_GPIO_NUM 39
#define Y5_GPIO_NUM 35
#define Y4_GPIO_NUM 14
#define Y3_GPIO_NUM 13
#define Y2_GPIO_NUM 34
#define VSYNC_GPIO_NUM 5
#define HREF_GPIO_NUM 27
#define PCLK_GPIO_NUM 25
#elif defined(CAMERA_MODEL_AI_THINKER)
#define PWDN_GPIO_NUM 32
#define RESET_GPIO_NUM -1
#define XCLK_GPIO_NUM 0
#define SIOD_GPIO_NUM 26
#define SIOC_GPIO_NUM 27
#define Y9_GPIO_NUM 35
#define Y8_GPIO_NUM 34
#define Y7_GPIO_NUM 39
#define Y6_GPIO_NUM 36
#define Y5_GPIO_NUM 21
#define Y4_GPIO_NUM 19
#define Y3_GPIO_NUM 18
#define Y2_GPIO_NUM 5
#define VSYNC_GPIO_NUM 25
```

```
#include "esp_camera.h"

// Select camera model - find more camera
// https://github.com/espressif/arduino-esp32

#define CAMERA_MODEL_XIAO_ESP32S3 // Has

#define PWDN_GPIO_NUM    -1
#define RESET_GPIO_NUM   -1
#define XCLK_GPIO_NUM     15
#define SIOD_GPIO_NUM     4
#define SIOC_GPIO_NUM     5

#define Y9_GPIO_NUM       16
#define Y8_GPIO_NUM       17
#define Y7_GPIO_NUM       18
#define Y6_GPIO_NUM       12
#define Y5_GPIO_NUM       10
#define Y4_GPIO_NUM       8
#define Y3_GPIO_NUM       9
#define Y2_GPIO_NUM       11
#define VSYNC_GPIO_NUM    6
#define HREF_GPIO_NUM     7
#define PCLK_GPIO_NUM     13
```

Don't forget to disable the ESP NN acceleration. Failure to do so may result in the following error message:

[illegible]

Go to:

Documents\Arduino\libraries\Car_color_recognition_inferencing\src\edge-impulse-sdk\classifier\ei_classifier_config.h

And change this line from 1 to 0 and save:

```
#define EI_CLASSIFIER_TFLITE_ENABLE_ESP_NN 0
```

You must do some changes in this code because generated code by EI it doesn't work for this stick, I mean code works but in serial monitor it doesn't have any outputs

Change every: Serial > Serial0

ei_printf > Serial0.printf

it should look like this:

```
void setup()
{
    // put your setup code here, to run once:
    Serial0.begin(115200);
    //comment out the below line to start inference immediately after upload
    while (!Serial0);
    Serial0.println("Edge Impulse Inferencing Demo");
    if (ei_camera_init() == false) {
        Serial0.printf("Failed to initialize Camera!\r\n");
    }
    else {
        Serial0.printf("Camera initialized\r\n");
    }

    Serial0.printf("\nStarting continuous inference in 2 seconds...\n");
    ei_sleep(2000);
}
```