02

Traffic Light Detection For color Blindness People Using Edge Impulse And ESP32-Camera

An affordable and accessible solution for safer traffic navigation

- Problem & Proposed Solution
- Requirements
- Edge Impulse
- FOMO
- Traffic Light Colors Model On Edge Impulse
- ESP32-CAM+ TL MODEL CODE
- ALL SYSTEM

Problem

Traffic lights are essential for road safety.
Color blindness affects 1 in 12 men and 1 in 200 women worldwide.
Difficulty distinguishing red, yellow, and green can cause confusion and safety risks.

Proposed Solution

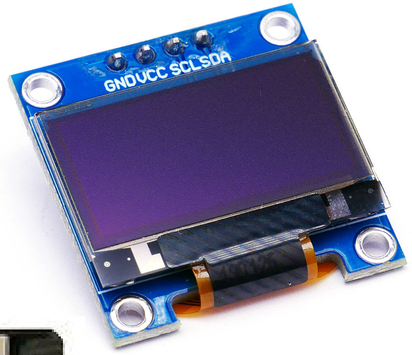
AI-powered traffic light detection system using Edge Impulse.
Runs on a low-cost ESP32-CAM microcontroller.
Detects traffic light colors and outputs real-time results to assist individuals.

Requirements

Core Component

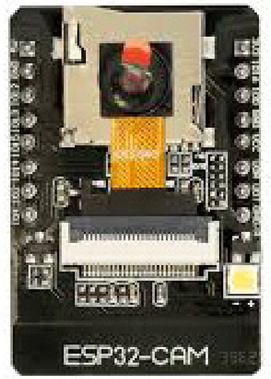
1. ESP32-CAM Module:

- A low-cost microcontroller
- It is responsible for capturing real-time images of traffic lights.



2. Edge Impulse Platform:

- Edge Impulse is a versatile platform designed for creating Ai models that run efficiently on edge devices



3. OLED Display:

- Display Results of Detection.

EDGE IMPULSE

What is Tiny ML??

TinyML refers to running machine learning models on small, low-power devices

After training data using edge impulse on a FOMO model,
TinyML allows the deployment of the model onto resource-constrained devices for real-time, low-power inference.

Generating Deployment Code!!

Edge Impulse provides a pre-compiled library for the target device (e.g., ESP32-CAM) in the form of:

- Arduino Library

- C++ code for TensorFlow Lite Micro

- These libraries include all the necessary code for running the TF Lite model on the target device.

Before We Start With FOMO ...

Object Detection Using Bounding Boxes

“Are there faces in the image , where and what size are they?”

Object Detection Using Centroids

“Are there faces in the image , where are they?”

Heatmap:

Grid-based heatmap predicts the likelihood of an obj at specific point(centeroids).

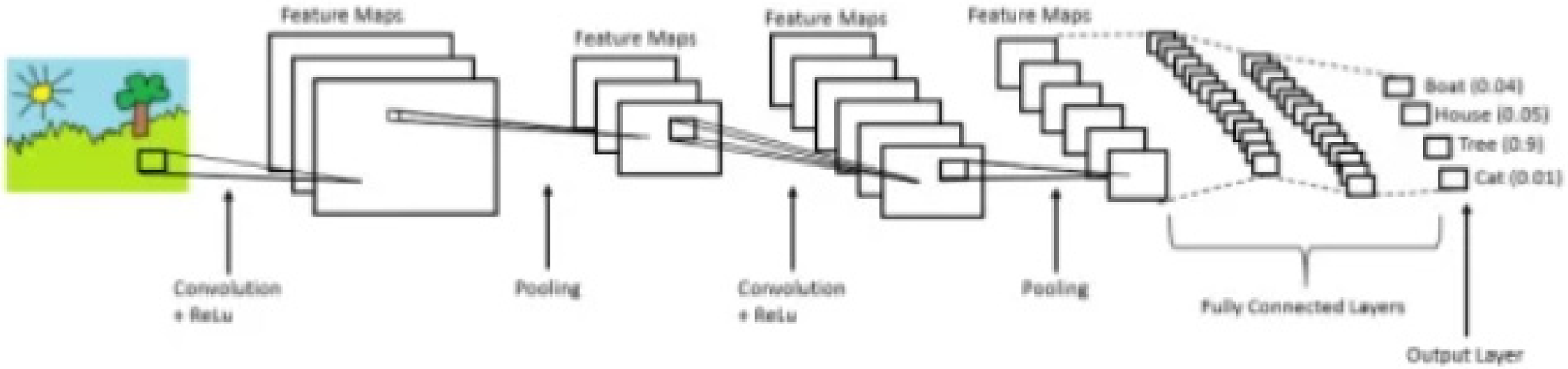

FOMO = Centroid Detection + Grid-based heatmap

FOMO

Faster Objects More Objects

- based on mobilenet architecture
- ultra fast , measured on Rpi to give 60 fps
- Capable of segmentation and counting objects

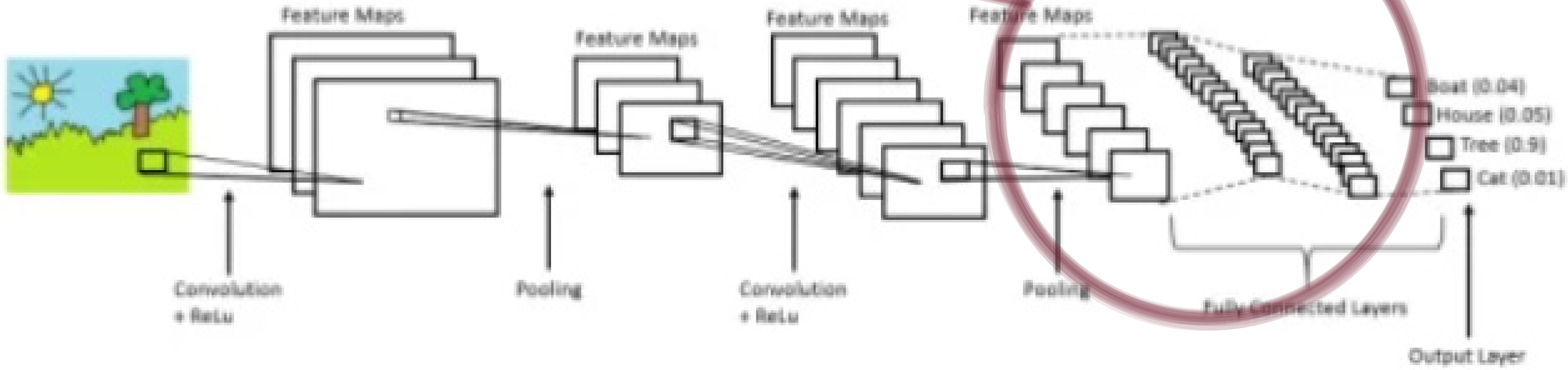
MobileNet V2



FOMO

Replace It With Fully connected layers to reduce dimensions to match required grid size

Simplified : replaces bounding box with HeatMap (dividing image into a grid each grid cell predicts the probability of an object centeroid belonging to particular class).



Key Defferences

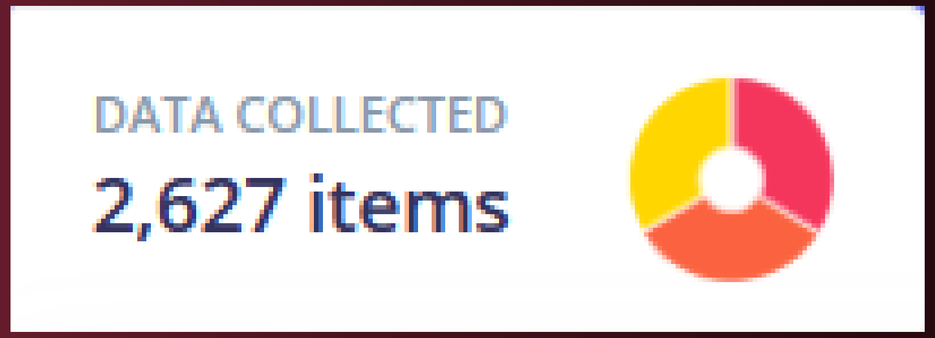
Asbect	MobileNetSSD / YOLO	FOMO
BOUNDING BOXES	predicts bounding boxes (x,y,width,hight)	No Bounding, only centroid locations.
HEAT MAP USAGE	No	uses a grid-based heatmap for centeroid detection.
COMPLEXITY OF PREDECTION	Requires seperate outputs for box regression a and classification	Single probability output per grid cell.
LATENCY AND RESOURSES USAGE	Higher latency and resource requirements	extremely low latency and resource usage.

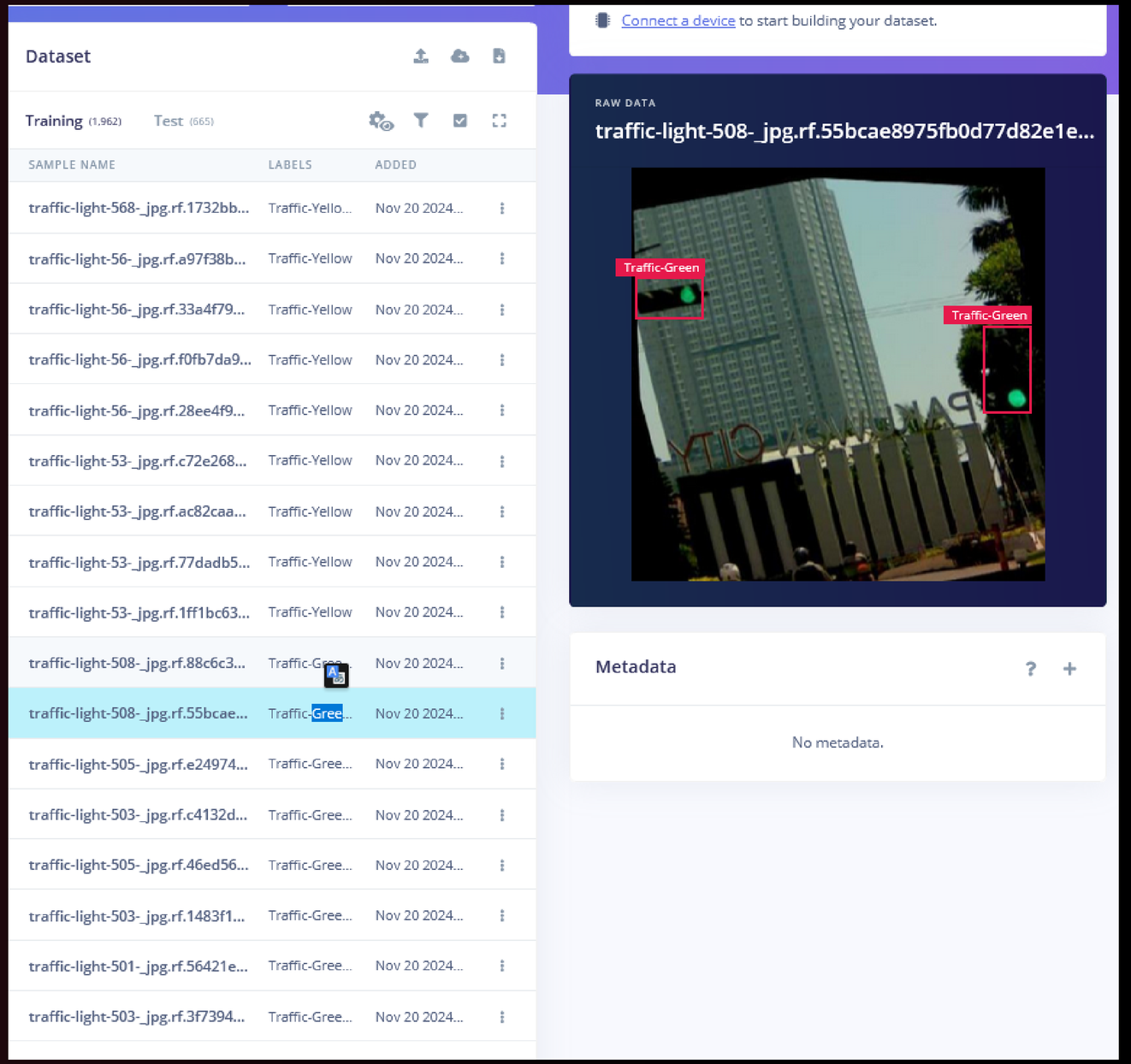
Traffic Light Colors Detection Model On Edge Impulse:

1. Data Collection

- Source: The dataset is obtained from Roboflow as a YOLOv5 format with pre-labeled images and a corresponding YAML file.

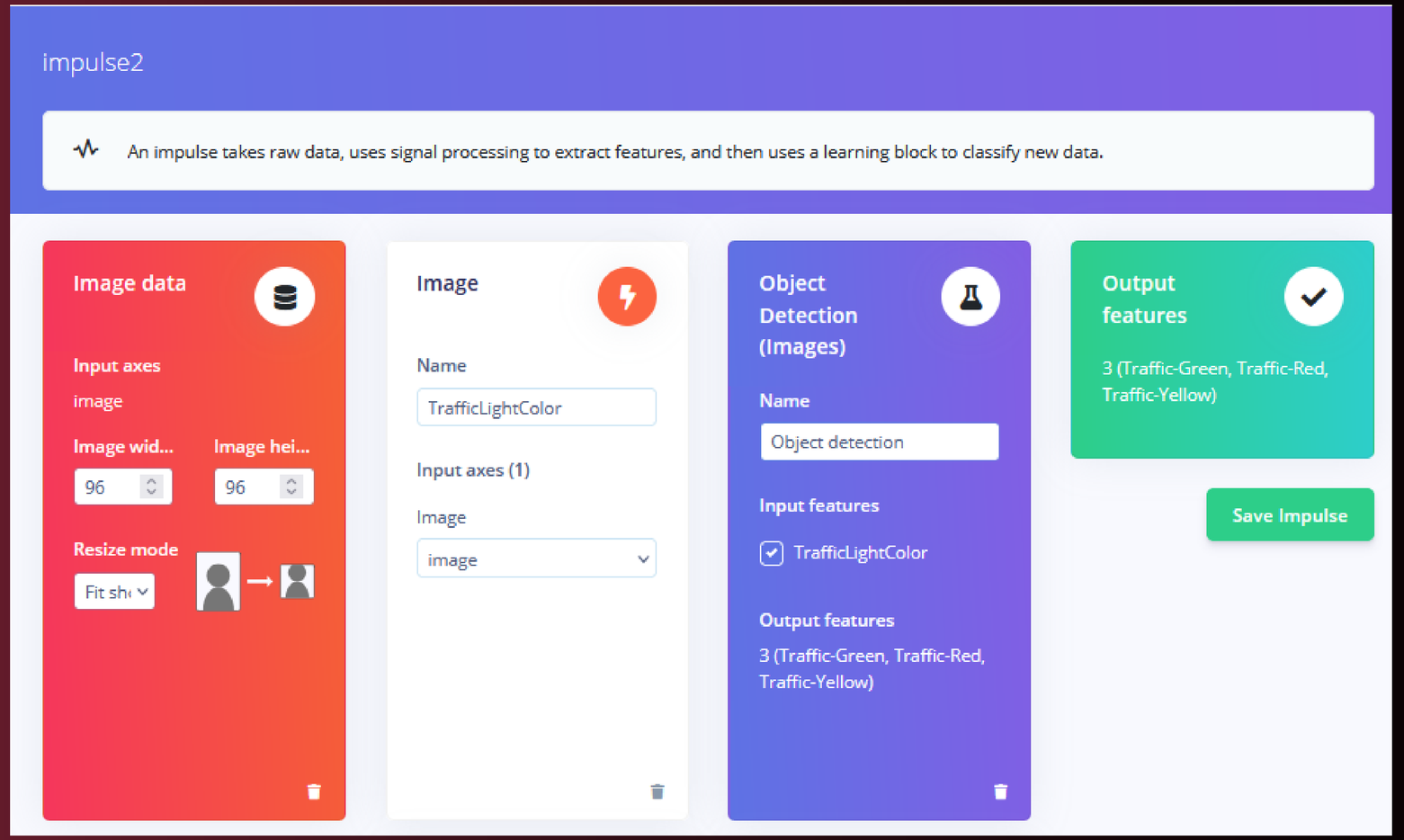
- Dataset Details: Total images: 2627, Labels: Three classes (Red Traffic light, Yellow Traffic light, Green Traffic light).





2 . Create Impulse

the impulse pipeline is all about cleaning, processing,
and arranging the raw data before the actual training phase.



3. Data Splitting:

Split the collected dataset into training and test sets (80% training, 20% testing).

4 . Training Process:

- Model Type: FOMO (Faster Objects, More Objects) neural network architecture.
  - Specifically optimized for object detection on microcontrollers.
  - Focuses on the center points of detected objects without full-scale bounding box segmentation.

- Training Details:
  - Input dimensions: 96x96 pixels.
  - Output dimensions: 3 classes (red, yellow, green).
  - Training process: Iterative optimization of neural network weights to minimize classification error.
- Output: A trained FOMO model capable of detecting traffic light states.

Neural Network settings

Training settings

Number of training cycles: 50

Use learned optimizer: ☐

Learning rate: 0.01

Training processor: CPU

Data augmentation: ☑

Advanced training settings

Validation set size: 20 %

Split train/validation set on metadata key:

Batch size: 32

Profile int8 model: ☑

Neural network architecture

Input layer (27,648 features)

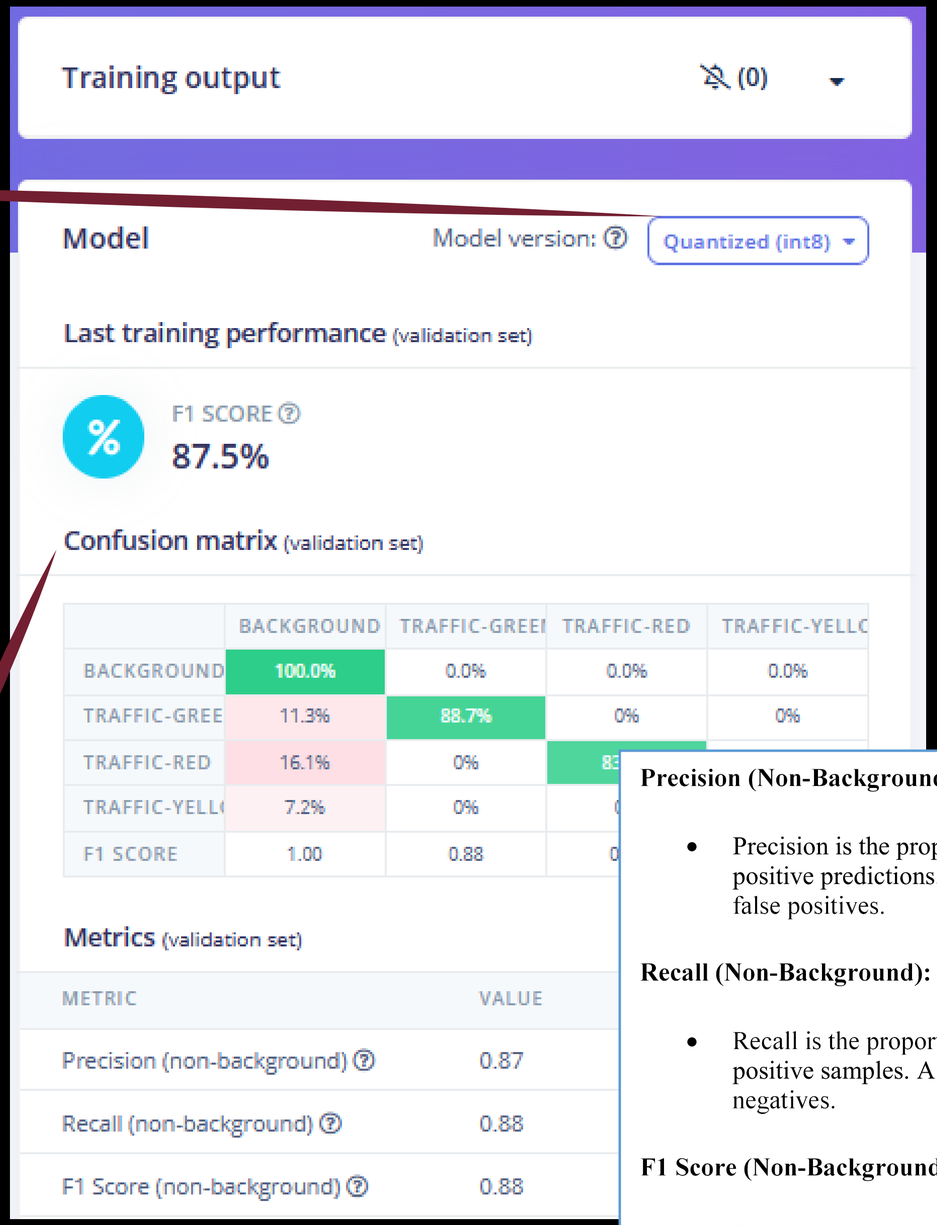
FOMO (Faster Objects, More Objects) MobileNetV2 0.35

Choose a different model

Output layer (3 classes)

Model Version: `Quantized (int8)`

- The model has been quantized to `int8`, which means it uses 8-bit integer representations instead of 32-bit floating-point numbers. This reduces the model's size and memory requirements, making it ideal for deployment on resource-constrained devices like the ESP32-CAM.



- Rows: Represent the actual class (ground truth).
- Columns: Represent the predicted class

Background:

100% of the background samples were correctly classified as background.

Traffic-Green:

- 88.7% of the traffic-green samples were correctly identified.
- 11.3% were misclassified as traffic-red.

Traffic-Red:

- 83.9% of the traffic-red samples were correctly identified.
- 16.1% were misclassified as traffic-green.

- **Traffic-Yellow:**
  - 92.8% of the traffic-yellow samples were correctly identified.
  - 7.2% were misclassified as background.

Precision (Non-Background): `0.87`

- Precision is the proportion of true positive predictions among all positive predictions. A high precision value means the model avoids false positives.

Recall (Non-Background): `0.88`

- Recall is the proportion of true positive predictions among all actual positive samples. A high recall value means the model avoids false negatives.

F1 Score (Non-Background): `0.88`

- The overall F1 score for non-background classes, reflecting the model's balance between precision and recall.

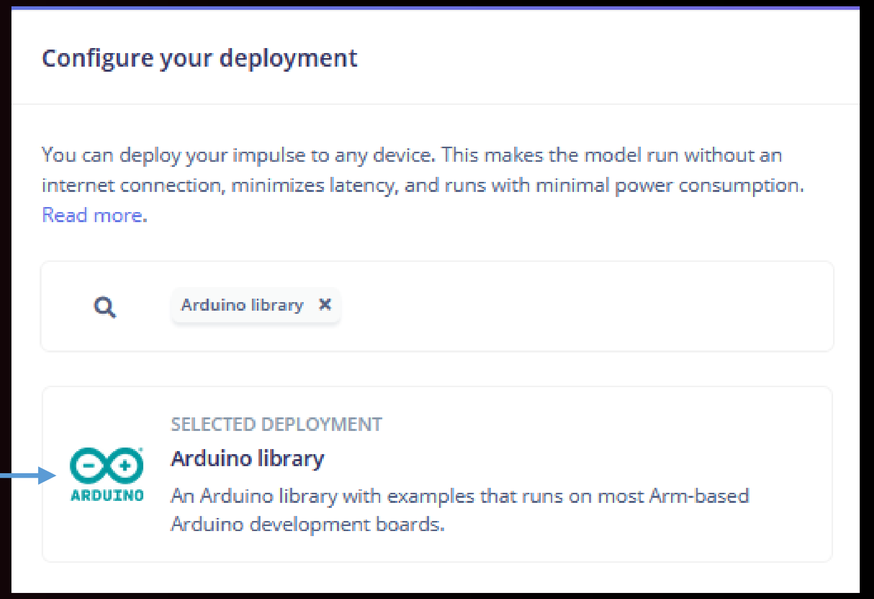
5 . Model Deployment :

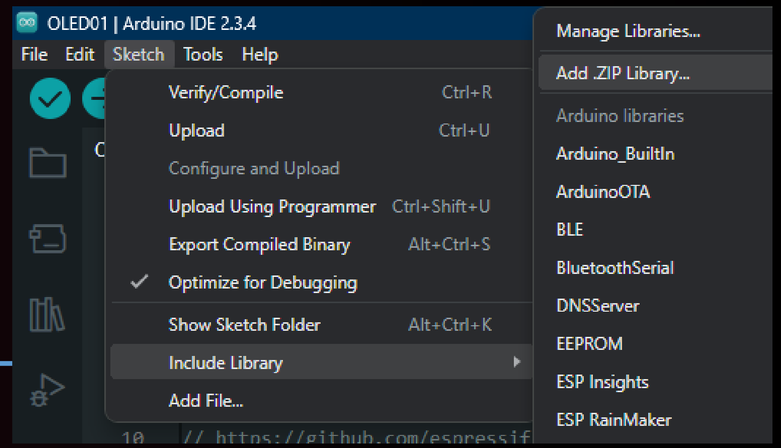
Deploying the trained Edge Impulse model on an ESP32-CAM involves generating the Arduino library from Edge Impulse, integrating it into the Arduino IDE, and then programming the ESP32-CAM to perform inference in real-time. Below is the detailed process:

Step 1: Export the Model

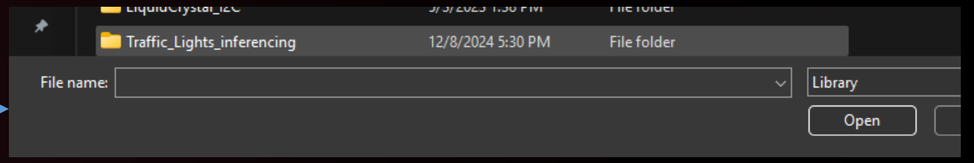
1. **Generate the Arduino Library:**
   - In the Edge Impulse dashboard, navigate to the **Deployment** tab.
   - Select **Arduino library** as the deployment target.
   - Click **Build** to generate the library. This will create a `.zip` file containing the model and necessary files for Arduino.

2. **Download the Library:**
   - Download the generated `.zip` file to your local system.





Step 2: Import the Library into Arduino IDE



TRY THE MODEL :)

Esp32Cam-Code-WorkFlow