

Problem Statement

Various sensors such as Inductive loop sensors, Infrared Sensors, Ultrasonic Sensor, Magnetic Sensors are used in parking lots to determine the number of vehicles parked. High installation and maintenance cost, durability issues are few of the major disadvantages. There is a need for cost effective and efficient solution.

Objective

To implement Machine Learning with ESP32 Cam for

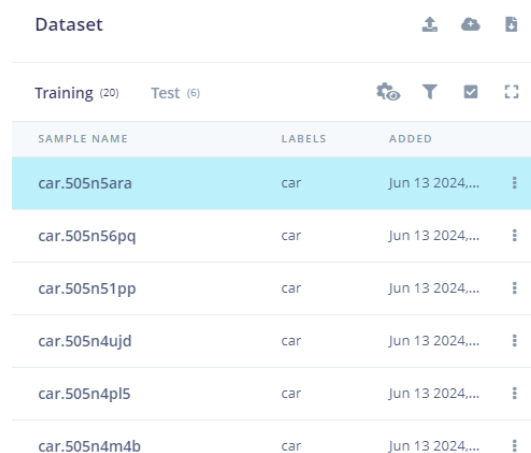
- Determining the status of vehicles parked by using object classification and detection.
- Develop a cost effective solution for smart parking application.

Methodology

Edge Impulse is a leading development platform for Machine Learning on edge devices. The Arduino zip library is installed and deployed on the ESP32 Cam board.

A Toy car is used for demonstrating the proof of concept.

1. **Data Collection:** Collecting various angles of top view of the car using the Data Acquisition tab in Edge Impulse.



The screenshot shows the 'Dataset' tab in the Edge Impulse interface. It displays a table with columns for 'SAMPLE NAME', 'LABELS', and 'ADDED'. The first row is highlighted in light blue. Above the table, there are tabs for 'Training (20)' and 'Test (6)', and icons for settings, filters, checkboxes, and refresh. The table lists six samples, all labeled 'car', with their respective sample names and the date they were added (June 13, 2024).

Dataset		
Training (20) Test (6)		
SAMPLE NAME	LABELS	ADDED
car.505n5ara	car	Jun 13 2024,...
car.505n56pq	car	Jun 13 2024,...
car.505n51pp	car	Jun 13 2024,...
car.505n4ujd	car	Jun 13 2024,...
car.505n4pl5	car	Jun 13 2024,...
car.505n4m4b	car	Jun 13 2024,...

Fig 1: Figure representing the data acquisition tab of Edge impulse.

2. **Labelling the data:** The data is labelled in the Labelling Queue tab of edge impulse.



Fig 2: Figure representing the data labelling tab of Edge impulse.

3. **Addition of blocks:** Addition of Image processing block and Object detection Learning block and setting the size of image as 48X48.

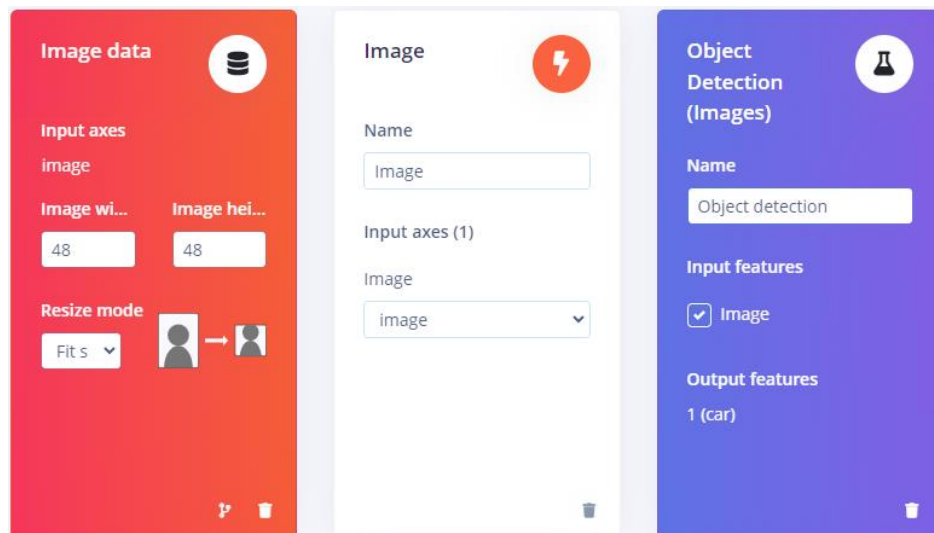


Fig 3: Figure representing the addition of processing and learning block of Edge impulse.

4. **Feature Generation:** Setting the image to Grayscale and generating parameters.

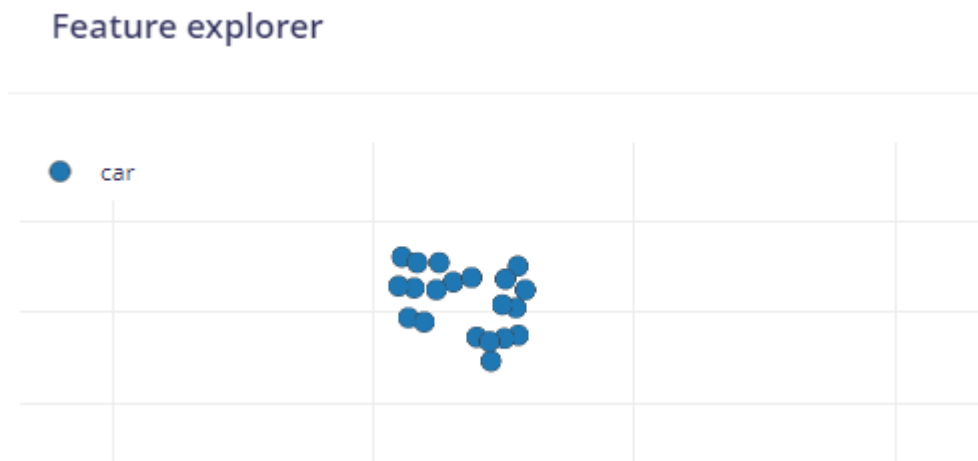


Fig 4: Figure representing the features generated.

5. **Selection of Classification algorithm:** Selecting classification algorithm such as FOMO (Faster Objects and More objects) designed specifically for edge devices. Setting the parameters for ESP32 cam with respect to Learning rate and training cycle.

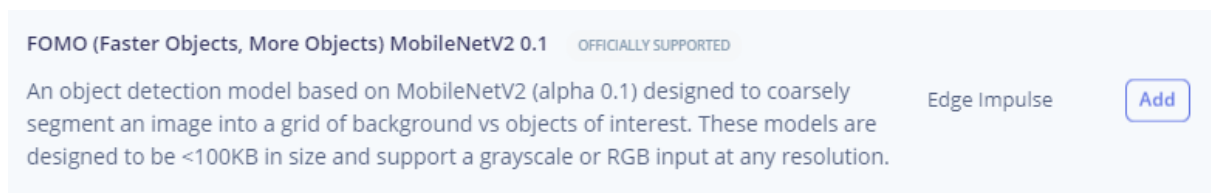


Fig 5: Figure representing the selection of Algorithm for ESP32 edge device

6. **Training and Testing:** Training and testing of the model. Analysing the F1 score.

Last training performance (validation set)

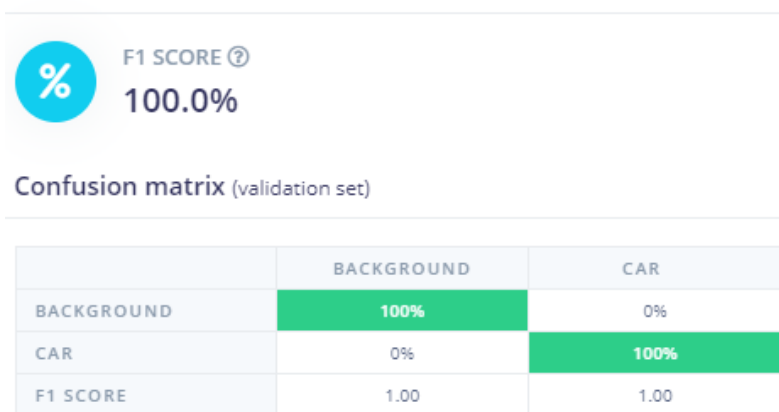


Fig 6: Figure representing the F1Score of the Model Testing Results

- 7. Live Classification:** Evaluating the model by providing new unseen data.



Fig 7 Results of classification on unseen data

- 8. Making Changes to the code:** Few changes such as selecting the appropriate CAM board. Making changes to the code which prints the Result on to the serial monitor.
- 9. Deployment:** Generating an Arduino compatible library and deploying the model using the Arduino IDE onto the ESP32 Cam Board by connecting the ESP32 CAM board through the USB Port.

Results

The Results can be viewed on the Arduino IDE Serial monitor. When the ESP32 cam is made to face the top view of the car, “Car Parked” is printed on the Serial monitor in the absence of the car “No Car” is printed onto the Serial monitor.

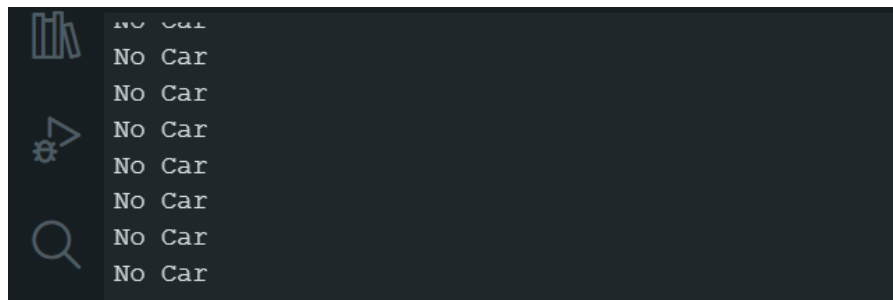


Fig 8: No Car printed on Serial monitor when car is not placed below the **camera**

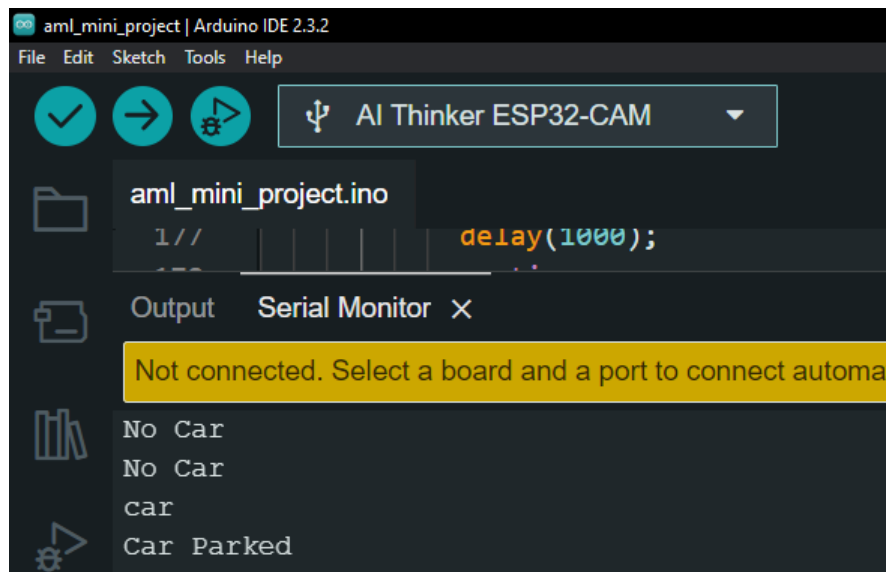


Fig 9: "Car Parked" printed on Serial monitor when car is placed below the camera