

BMI/CEN 598: Embedded Machine Learning Final Project (F4)

Edge Vision: User Defined Object-Counting using Raspberry Pi

Manuscript

Arizona State University

Submitted By:

Jayaram Atluri

Pavan Kalyan Majjiga

Swapnil Meshram

Shubh Patel

Instructor: Hassan Ghasemzadeh

Lab: Embedded Machine Intelligence Lab (EMIL)

Abstract:

This project uniquely combines the Raspberry Pi 4's processing capability and the high-resolution capabilities of its Camera Module V2 (8MP, 1080p) to create a user-defined object-counting system. Utilizing the sophisticated Faster R-CNN ResNet-50 model, EdgeVision offers unmatched real-time, high-resolution object recognition and tracking. The project not only exhibits the actual implementation of complex machine learning models in edge computing but also showcases a unique user interface that enables for interactive examination of video feeds for diverse applications.

Introduction:

EdgeVision reshapes the landscape of edge computing in video analytics. By integrating cutting-edge machine learning algorithms with the sturdy and versatile Raspberry Pi 4 and its Camera Module V2, the project delivers a revolutionary solution for real-time video analysis. This integration is crucial in several sectors, including public safety, traffic monitoring, and retail analytics, delivering a scalable, efficient, and user-friendly solution for object detection and counting. EdgeVision illustrates the promise of embedded machine learning in redefining daily technological applications.

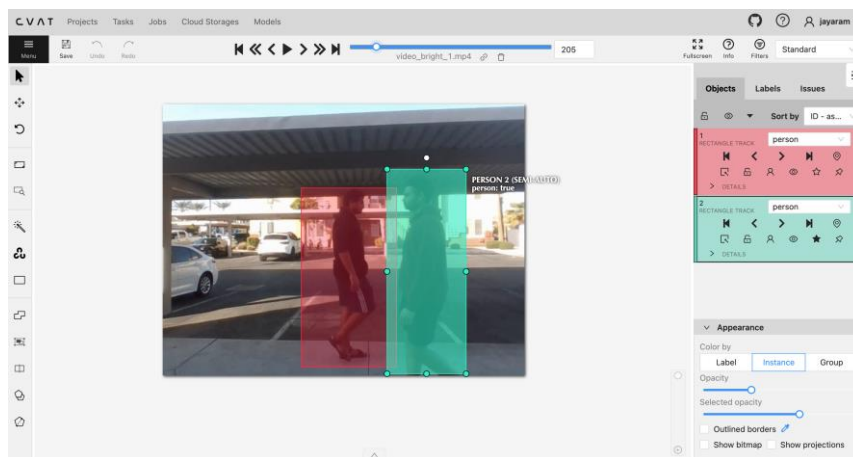
Problem Statement:

The EdgeVision project was intended to address two key challenges in the field of video analytics:

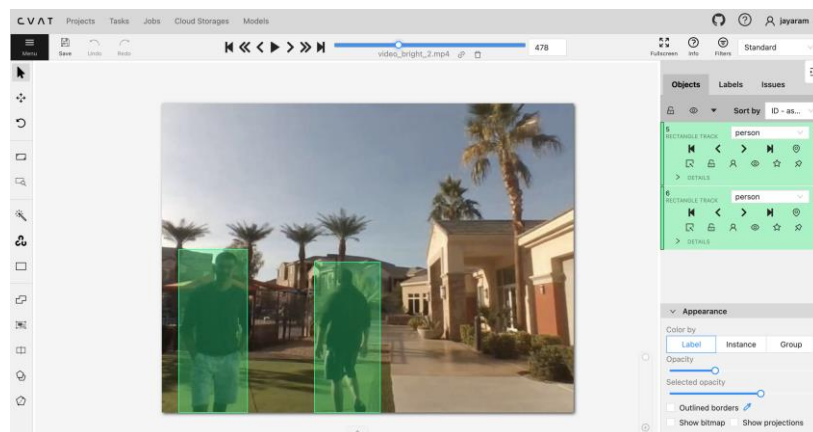
- 1) Achieving High Accuracy: The project targets the need for highly precise object detection and tracking across varied environmental and lighting circumstances, a problem in conventional video analysis systems.
- 2) Compact and Effective Technology Implementation: A primary objective is to efficiently integrate this advanced technology on a compact platform like the Raspberry Pi 4. This requires balancing the balance between processing power, energy consumption, and real-time responsiveness, making the solution viable for wide-ranging applications.

Data Collection and Annotation:

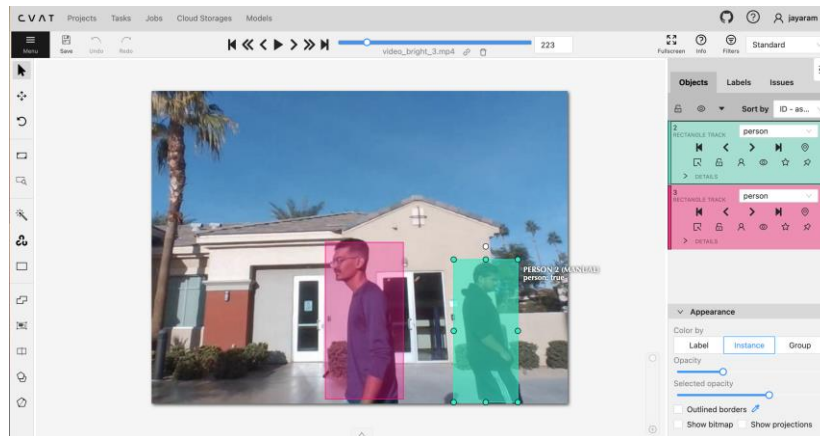
To construct and optimize the machine learning model, our project group deployed the Raspberry Pi Camera Module V2 to acquire high-quality video data. We collected data at various lighting settings, background environment and movement scenarios, crucial for training a robust object detection system. The annotation process, undertaken with great attention to detail on 'cvat.ai', required labelling each object in the video frames with bounding boxes. This rigorous method guaranteed that the model learned to recognize and track objects reliably, comprehending their dimensions and movements in varied settings, forming the core of EdgeVision's machine-learning capabilities.



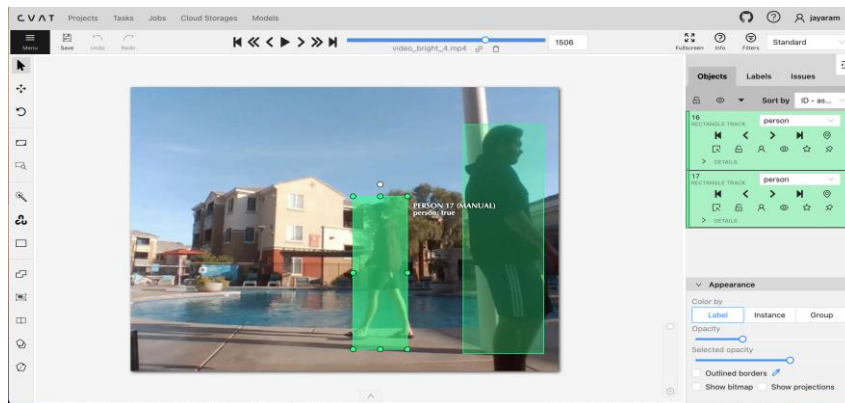
Annotation 1: Labelled sample in outdoor bright lighting condition, consisting of 1554 frames at a rate of 24 frames per second (fps).



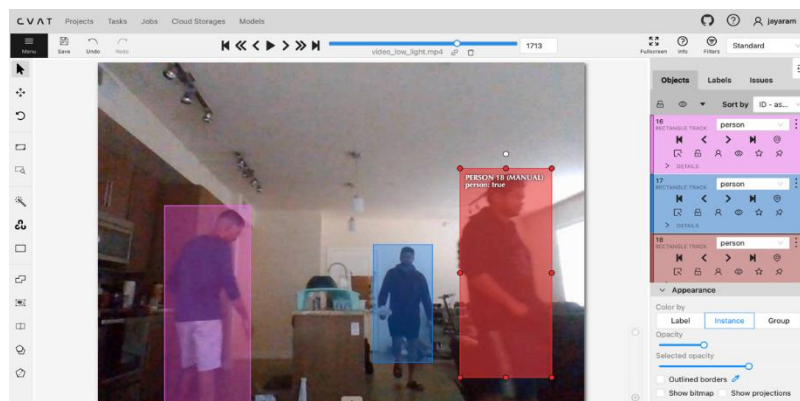
Annotation 2: Labelled sample in outdoor bright lighting condition, comprising 1421 frames at 24 frames per second (fps).



Annotation 3: Labelled sample in outdoor bright lighting condition, with a total of 1499 frames at 24 frames per second (fps).



Annotation 4: Labelled sample in outdoor bright lighting condition, featuring 2160 frames at 24 frames per second (fps).



Annotation 5: Labelled sample in indoor low lighting condition, encompassing 2518 frames at 24 frames per second (fps).

[< Back to projects](#)

Actions ⋮

edgevision [🔗](#)

Project #80061 created by jayaram on November 29th 2023

Assigned to

Project description

Edit

Issue Tracker [🔗](#)

[🔗](#) Raw

[🔗](#) Constructor

```






{
  "name": "person",
  "id": 1560006,
  "color": "#16f698",
  "type": "rectangle",
  "x": 1560006,
  "y": 1560006,
  "x2": 1560006,
  "y2": 1560006
}

```

Done

Reset

Raw label for Annotation

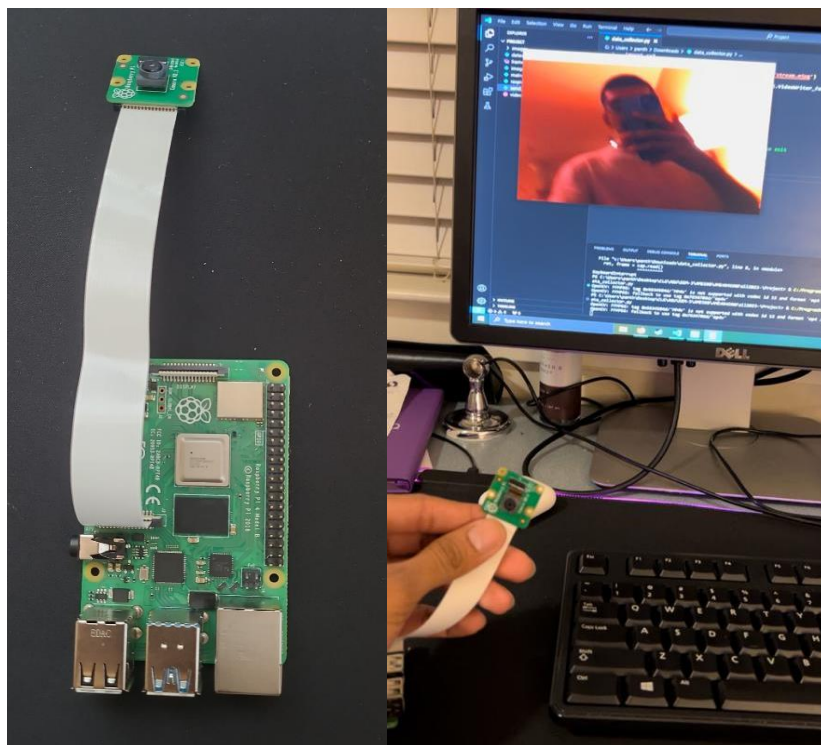
| | | | |
|---|--|--|--------------------------------------|
|  | #403746: Annotation 5 Created by jayaram on December 5th 2023 Last updated 33 minutes ago | <div> <div>• 1 annotating • 1 total</div> <div></div> </div> | <div>Open</div> <div>Actions ⋮</div> |
|  | #403744: Annotation 4 Created by jayaram on December 5th 2023 Last updated an hour ago | <div> <div>• 1 annotating • 1 total</div> <div></div> </div> | <div>Open</div> <div>Actions ⋮</div> |
|  | #403742: Annotation 3 Created by jayaram on December 5th 2023 Last updated an hour ago | <div> <div>• 1 annotating • 1 total</div> <div></div> </div> | <div>Open</div> <div>Actions ⋮</div> |
| Train | | | |
|  | #403741: Annotation 2 Created by jayaram on December 5th 2023 Last updated 3 hours ago | <div> <div>• 1 annotating • 1 total</div> <div></div> </div> | <div>Open</div> <div>Actions ⋮</div> |
|  | #403730: Annotation 1 Created by jayaram on December 5th 2023 Last updated 4 hours ago | <div> <div>• 1 annotating • 1 total</div> <div></div> </div> | <div>Open</div> <div>Actions ⋮</div> |

Labelling Sub Tasks

Overview of Project Modules

1) Raspberry Pi 4 and Video Streaming for Data Collection

- Enhanced Capabilities: The Raspberry Pi 4, with its fast CPU, greater RAM, and enhanced connectivity choices, provides a solid foundation for the project. It handles intense computational needs of video processing and machine learning techniques.
- We used WIFI in the Raspberry Pi by Python socket library and threading libraries to process the http request and streamed the video from camera to laptop at 24 fps and (680, 480) pixels resolution.

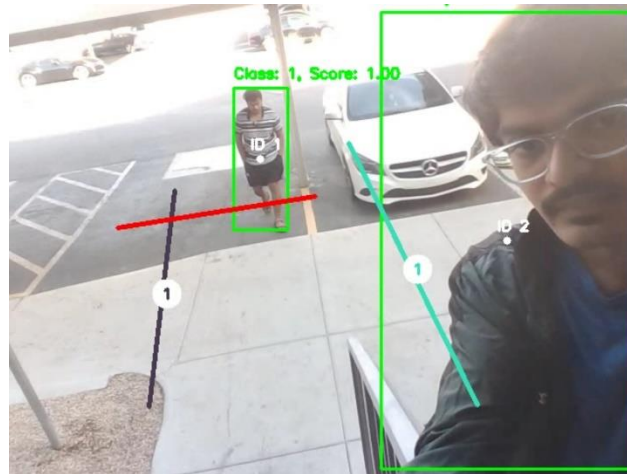


Raspberry Pi and Camera Integration with Video Streaming

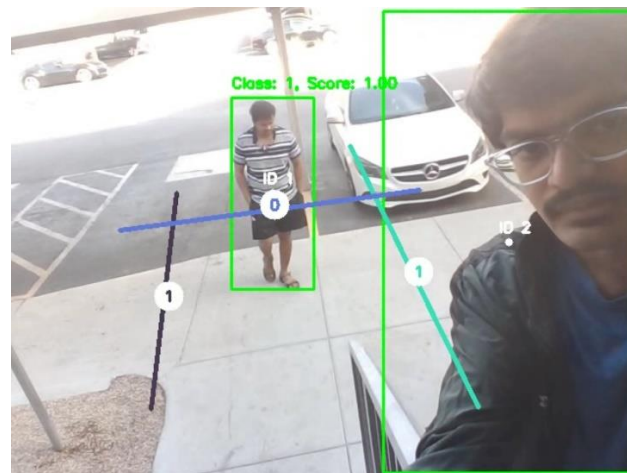
2) LineSegmentCounter

- Interactive Functionality: Users can draw and change line segments on the video interface, enabling real-time counting and analysis of object movements. This module's capabilities is crucial for applications like monitoring traffic flow at crossroads or assessing customer footfall in retail locations.
- Technical Details: The module employs event-driven programming to respond to user inputs, adjusting line positions and calculating object counts dynamically as objects passes through the defined lines. Once the mouse is click it displays

start point on the image for the new line draws a temporary red line until mouse is released at end point. It stores all the line segments drawn by the user and displays on the video.



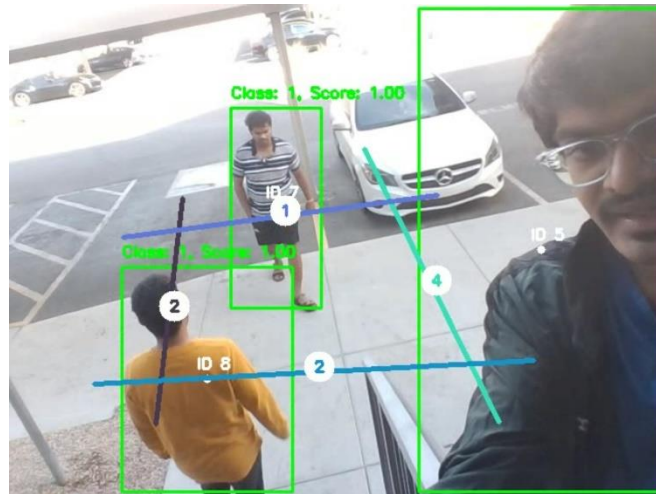
Temporary Red Line Segment while drawing counter on Video



New Counter Drawn on the Video and Stored

3) ObjectDetector

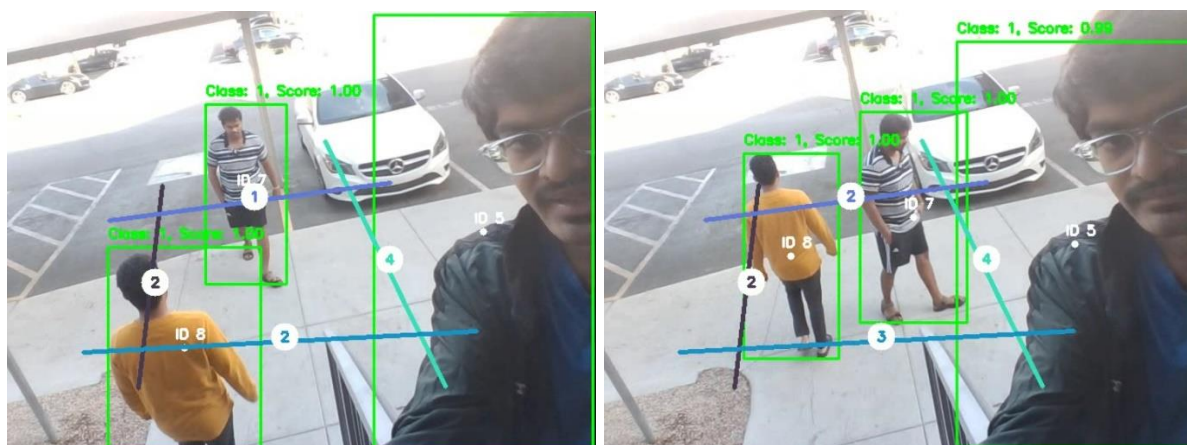
- **Faster R-CNN ResNet-50 Integration:** The ObjectDetector module capitalizes on the Faster R-CNN ResNet-50 architecture, recognized for its efficient and precise object identification. It adeptly handles various object sizes and occlusions, maintaining high detection accuracy.
- **Performance Optimization:** Special care has been taken to optimize the model for the Raspberry Pi 4, ensuring that it works effectively without overwhelming the device's processing capabilities.



Multiple Object Detection

4) CentroidTracker

- **Tracking Algorithms:** The CentroidTracker utilizes Euclidean distance based nearest neighbour algorithm to associate same objects from all the identified object's centroids across previous and current video frame. Also, it keeps track of the disappearance of an object and if object reappears after few frames within the threshold distance then it is associated with the same object. This makes it possible for continued tracking of objects if object detection is missed and even in settings with object overlap or transient occlusion.
- **Integration with ObjectDetector:** The tracker works in conjunction with the ObjectDetector, using the detected bounding boxes to determine and track object in the video feed.



Multiple Objects Tracked and Counted simultaneously by User Defined Counters

Technology and Model Description

- Faster R-CNN ResNet-50 Architecture Excellence: The Faster R-CNN ResNet-50 model stands as a cutting-edge solution in object detection. It combines the deep learning prowess of ResNet-50 with the efficiency of the Faster R-CNN's region proposal network, making it perfect for real-time object detection.
- Deep Learning Capabilities: The ResNet-50 backbone in the model is responsible for extracting complicated information from the input video frames, while the region proposal network effectively offers potential object positions, which are subsequently tuned for precise detection and classification.

Detailed Implementation and Workflow:

1) Video Capture and Processing

- High-Resolution Input: Video data acquired by the Raspberry Pi Camera Module V2 gets into the system at high resolution, giving the detailed images necessary for accurate object detection.
- Real-Time Analysis: The video frames are processed in real-time by the Raspberry Pi. This requires numerous processes, each crucial to the system's overall effectiveness.

2) Machine Learning Pipeline

- Feature Extraction: At first, the ResNet-50 model processes the video frames, extracting a rich set of characteristics needed for detecting objects.
- Region Proposal: These properties are then examined by the Faster R-CNN's region proposal network, providing possible locations where objects might be found.
- Object Detection: For each proposed region, the model adds further convolutional layers to refine the detection, finding the object's class and specific location.

- Object Tracking: Concurrently, the CentroidTracker leverages this detection data to follow objects across frames, keeping their identities and tracking routes.

3) User Interaction and Data Output

- LineSegmentCounter Integration: As objects are detected and tracked, the LineSegmentCounter module communicates with this data. It watches the objects' trajectories with the user-defined line segments, counting each crossing and delivering real-time analytics.
- Output Visualization: The system provides a visual representation where users may see the identified objects, their tracking trajectories, and the number of instances of objects crossing the defined lines, all overlayed on the live video feed.

Code Implementation and Functionalities

1) Data Collection Codes

- Introduction: This section discusses the scripts important in collecting data for the EdgeVision project. It consists of raspicam_feed.py and data_collector.py scripts, each playing a critical role in video data streaming and collection.
- a) raspicam_feed.py
- Overview: This script builds a server to stream video from the Raspberry Pi camera over HTTP. It exhibits network programming, threading, and managing real-time video streams in Python using Socket and threading libraries.
 - Key Components:
 - Imports and Dependencies: Libraries enabling video streaming, server handling, and threading.
 - StreamingOutput Class: Handles the camera output stream and manages frame buffering and synchronization.
 - StreamingHandler Class: Manages requests made through HTTP to serve video stream.

- Camera and Server Initialization: Set up the Raspberry Pi Camera and streams images.
- b) data_collector.py
- Overview: This script is meant for capturing and recording video streams, which is important for collecting raw data for training model for object detection.
 - Key Components:
 - Import and Module: Utilizes the OpenCV library for real-time computer vision.
 - Video Capture Setup: Captures video through a network stream, primarily a Raspberry Pi camera server.
 - Video Output: Sets up cv2.VideoWriter for saving the captured video.
 - Video Capture Loop: Regularly reads and records frames from the video stream.

2) Project Implementation Codes

Introduction: This section explores the main operational scripts of the EdgeVision project, containing app.py, draw_count.py, and oop_detection.py. These scripts are crucial in object detection, tracking, and data processing. Object Oriented Methodology was used for working simultaneously on different aspects of the project.

- a) app.py
- Overview: Crucial to the project, this script analyses video streams for object detection and counting, incorporating OpenCV for various computer vision tasks.
 - Key Components:
 - Imports and Modules: Essential modules for object detection, tracking, and video processing.
 - Video Capture Setup: Captures video from network source.
 - Object Detection and Counting Initialization: Sets up object detection and counting classes.
 - Video Output Setup: Sets up video output for capturing processed streams.

- Processing Loop: Analyses frames for object detection, displays, and records results.

b) draw_count.py

- Overview: This program has the LineSegmentCounter class for interactive line drawing on video frames and displays the count of the objects passed through the line segments.

- Classes and their main methods:

i. **LineSegmentCounter**:

- a. draw_stored_segment: It's the main method that needs to be run for drawing and displaying the line segment counters on the frame. Using the center information of the objects in the current frame and previous frame, it detects whether the object has passed through the line segment or not.
- b. mouse_event: It detects the mouse click, draws a temporary line segment, registers new line segment once the mouse button is released and assigns a color to it.
- c. draw_text_on_segments: It updates the counter and writes count on the line segment.
- d. do_intersect: It takes (x, y) co-ordinates of two line segments as an input and returns 'True' if they intersect.

c) oop_detection.py

- Overview: Includes the ObjectDetector class that leverages a Faster R-CNN model with a ResNet-50 framework for object detection and as CentroidTracker.

- Classes and their methods:

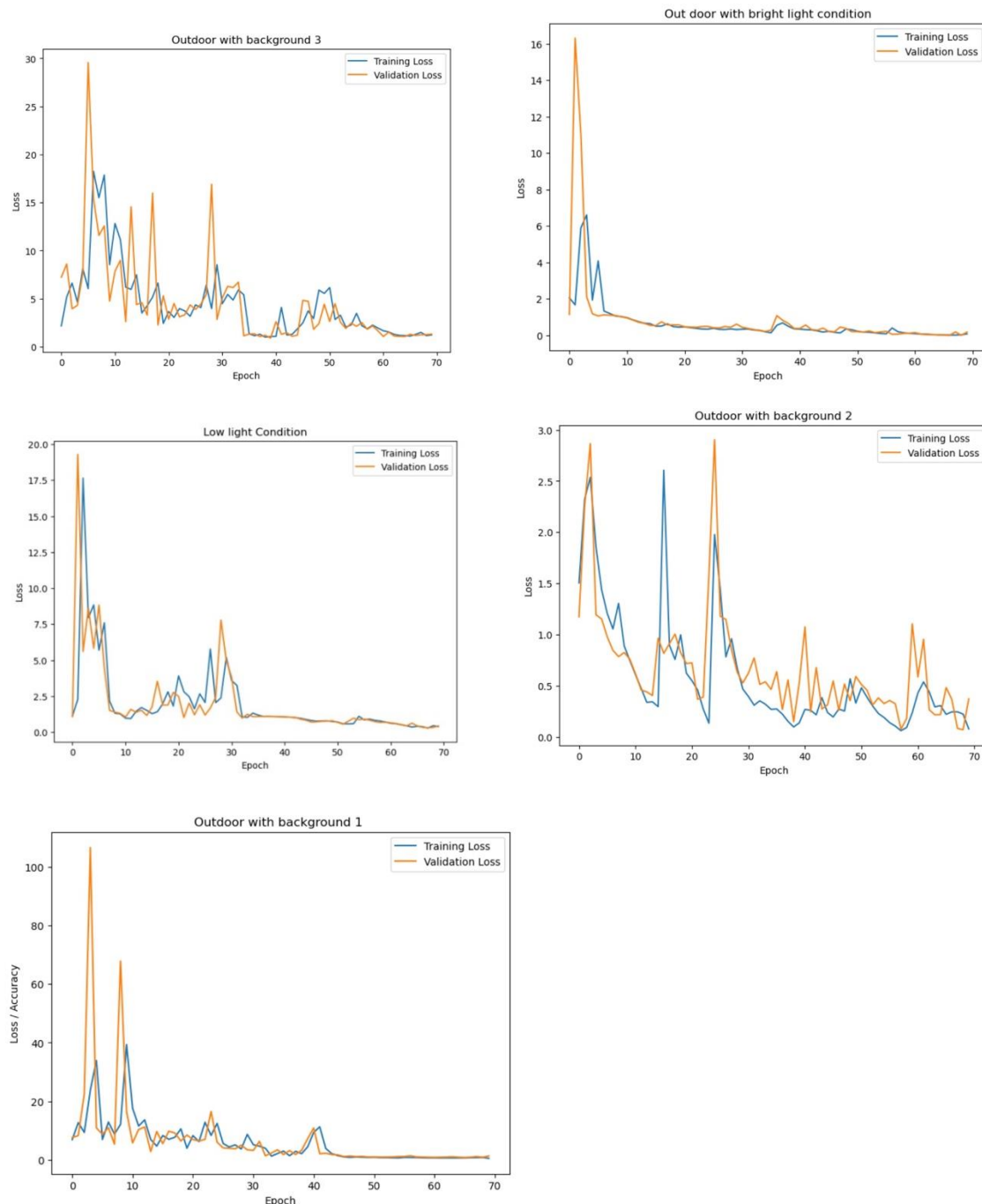
i. **ObjectDetector**: Sets up with target class and confidence threshold; loads and prepares the Faster R-CNN model.

- a. detect_objects: It detects objects with confidence threshold >0.7 and draws the bounding box around the object in the frame. It also returns the center of the bounding boxes which can be used for tracking the detected objects.

- ii. **CentroidTracker:** It tracks the detected objects using the center information from previous frame and current frame. video analysis.
 - a. **register_objects:** If the new detection is far from the threshold distance then register it as new object or else it is the same object. It also updates the center information of the objects.
 - b. **cleanup_unused_objects:** Sometimes object detection gets missed for few frames, so if the detection again appears after few frames and it is within the threshold distance it registers it as the same object. If the detection is missing for more than few frames, then the object information is deleted.

Results and Observations

After extensive testing in various environments, EdgeVision's performance and accuracy were validated. The model accuracies and losses for different conditions are as follows:



Accuracy Reported on Bright Condition was 95.6% and in Low Light Condition was 82.5%

Conclusion and Future Work

EdgeVision represents a significant breakthrough in embedded machine learning by showcasing the potential of integrating advanced machine learning models with small computing systems. Future upgrades may concentrate on improving energy efficiency, increasing adaptability to different environments, and exploring the possibility of integrating with other IoT devices for more extensive applications.