Dynamic Traffic Signal Management System

Pranav Vanama[1]

IIT GUWAHATI, Assam, India

Abstract. This paper presents a real-time traffic management system that utilizes a fine-tuned YOLOv8 that was fine-tuned on a dataset containing traffic images. The estimated density at the traffic signal dynamically controls the traffic signal. Here we simulate the scenario using Python's Turtle and OpenCV libraries.

Keywords: YOLO · Turtle.

1 Introduction

Traffic congestion in urban areas leads to increased delays, accidents, and pollution. Traditional traffic systems, with static signal timings, struggle to handle varying traffic conditions efficiently. In typical timings like around 5PM we see that all schools and offices will be done and people will travel back to their homes, we observe that few lanes are busy compared to others that could not be adapted by traditional static signal timings. Therefore, urban traffic congestion is a growing concern in many parts of the world. This project provides a dynamic traffic management system that adjusts signal timings based on real-time vehicle density estimation using OpenCV and YOLOv8.

2 Methodology

2.1 Dataset

The YOLOv8 model was trained on a Kaggle "Top-View Vehicle Detection Image Dataset". This dataset provides top-view images of vehicles, suitable for traffic density estimation. The dataset focuses on the 'Vehicle' class which will include wide variety of vehicles. Each image has been standardized to a 640x640 resolution. The dataset is divided into Training and Validation sets with 536 and 90 images respectively. Augmentation, including a 50% probability for horizontal flip is applied on the training set.

2.2 YOLOv8 Model Training

YOLOv8 - You Only Look Once version 8

The YOLO model is widely used for detecting objects.

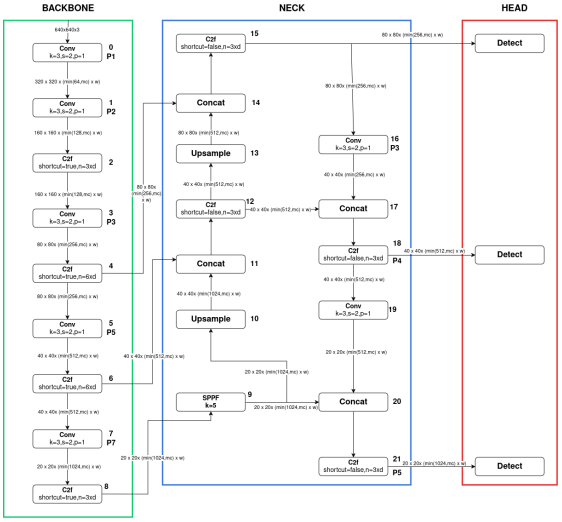

Fig. 1. YOLOv8 architecture

We can load the YOLOv8 model from a library called **ultralytics** in python. YOLOv8 is the first version of the YOLO series to offer an official package that can be easily installed using pip.

The loaded model is already trained with COCO dataset. We fine-tune the existing model with our kaggle dataset which will enhance the model ability to recognize and detect vehicles in top-view images.

The model was fine-tuned for 100 epochs. Training was executed on a CUDA-enabled GPU with an initial learning rate of 0.0001 and a dropout rate of 0.1

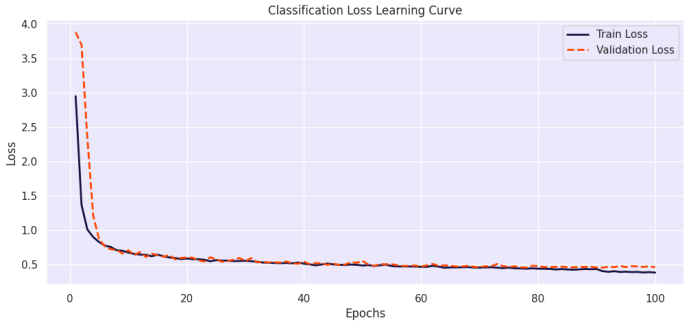

Fig. 2. Learning curve

Testing and Evaluation : In **98%** of cases the model successfully detects the presence of a vehicle when there is one. And only in 1% of the cases the model detects a non-vehicle or background as a vehicle.

In object detection, we care about what is in the image and where it is. So we need a metric that takes both the aspects into the account i.e our metric should check wether the position of the object is predicted correctly or not and the class also.

mAP - Mean Average precision

mAP is a widely used metric in such cases.

We predict all the boxes(position) and classes for all the images. Then we calculate IoU(Intersection over Union) it is the ratio of areas, of intersection over the union of the predicted box and ground truth box. We set a certain threshold and all the objects in an image with IoU greater than that threshold are taken into account and the precision and recall are calculated for each image and a Precision v/s Recall curve is plotted for each class.Average precision for a class is the area under the PR curve. mAp is the mean of AP of all the classes.

For our fine-tuned model we got a **mAP50 as 97.5% and a mAP50-95 as 74.3%**.

2.3 Traffic signal simulation

The simulation is created using Python's **Turtle** library to visually represent traffic signal behavior.

- The fine-tuned model is loaded into a Python script for vehicle detection.
- We use a folder of traffic jam images to test our model.
- I am assuming 4 lanes are present. For each lane, a random image is selected and the number of vehicles is detected using the trained model.
- Based on the vehicle count in each lane, a green signal is assigned for that lane. And this is visually displayed through the Turtle simulation.
- The time-to-vehicle ratio may vary so we should customize it accordingly, by default we are setting it to 1.
- I also added a camera function which when implemented opens the webcam and takes snapshot so that we can use it.

3 Results

The result is a real time dynamic traffic management system which can take snapshots of the vehicles at the traffic signal and based on the vehicle count which is predicted by our YOLOv8 fine-tuned model we can adjust the time of green signal at the traffic signal. This will result in lesser chance of congestion hence reducing time waste, less air pollution, less sound pollution, fewer accidents and adaptability according to the situations.

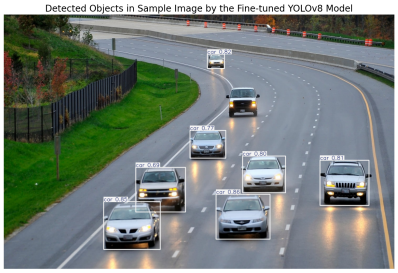

Fig. 3. Vehicle detection

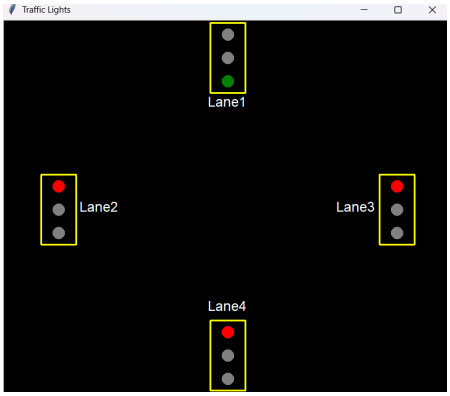

Fig. 4. turtle simulation

4 Future Work

- Integrate it with IoT-devices for scalability.
- Add emergency vehicle detection. So that we can change the green signal to that lane until the emergency vehicle is gone by.
- Create a web-based dashboard for traffic stats.
- We can also add pedestrian signal in all the lanes.

References

1. Ultralytics YOLOv8 Documentation. https://docs.ultralytics.com/models/yolov8/
2. Kaggle Dataset. https://www.kaggle.com/datasets/farzadnekouei/top-view-vehicle-detection-image-dataset
3. OpenCV Documentation. https://docs.opencv.org/4.x/index.html
4. Python Turtle Graphics Documentation. https://docs.python.org/3/library/turtle.html
5. Pranav, Dynamic Traffic Management System – GitHub Repository. https://github.com/25-pranav-25/dynamic-traffic-management