Real-Time Vehicle Detection in Traffic Monitoring using YOLO model

James Nguyen

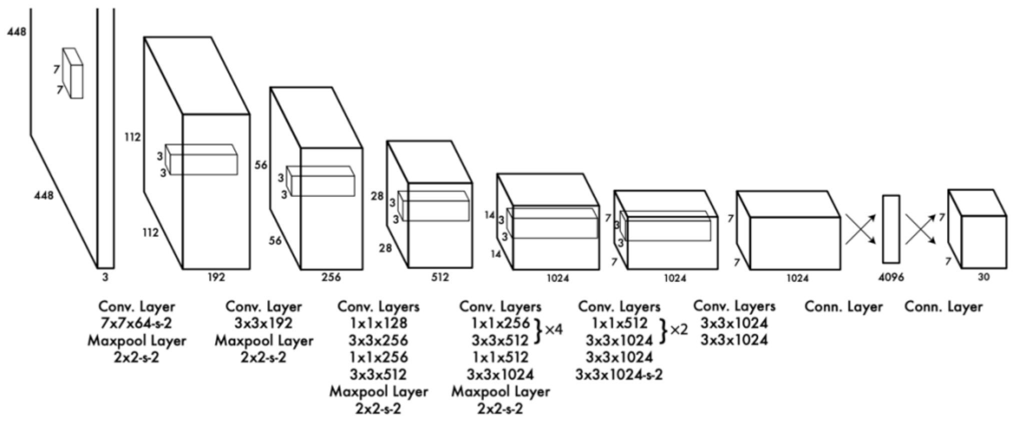
1. Introduction

CCTV for traffic monitoring has been used for years, but with the development of nowadays technologies and the help of AI, people now go beyond passive video recording to perform real-time traffic analysis. For instance, by utilizing YOLO model, we can analyze surveillance cameras to detect and analyze vehicles on the road in real-time. The model can process video streams from traffic cameras, count the number of vehicles and determines congestion levels based on predefined thresholds from a given frame. The goal of this project is to monitor traffic conditions, which can be used for smart city applications and traffic management or notification.

2. Related Work

As object detection algorithms have developed continuously, there are variety of models like R-CNN, Faster R-CNN, or SSD (Single Shot Detection) that are able to do really well in detecting objects from videos. Faster R-CNN is a two-stage detector that first proposes regions of interest and then classifies them. It is well-known for high accuracy, especially when requiring precise localization. However, faster R-CNN model's computational complexity results in slower inference times, making it less suitable for real-time applications. For instance, studies have shown that Faster R-CNN achieves mAP scores but at the cost of inference speeds only around 3-5 FPS. This limits its practicality in scenarios like live traffic monitoring. On the other hand, SSD is a fast, one-stage object detection model that performs both classification and localization in a single pass, making it suitable for real-time applications. It also achieves a good trade-off between speed and accuracy, reaching up to 60 FPS with around 60-70 mAP. However, SSD model's performance tends to drop with small, distant objects or when the road is crowded, which limits its effectiveness in complicated traffic scenarios.

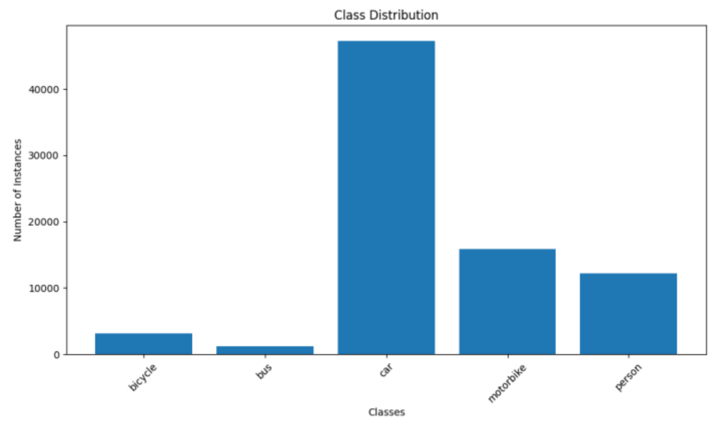
YOLO model offers a single-stage detection approach like SSD model, enabling faster processing compared to other two-stage detection approach. YOLO model incorporates architectural enhancements such as anchor-free detection and advanced feature fusion techniques inspired by GoogleNet model. These improvements have led to significant gains in both speed and accuracy. Comparative analyses show that YOLO model achieves mAP scores comparable to those of Faster R-CNN, while maintaining inference speeds suitable for real-time applications, often surpassing 90-130 FPS.

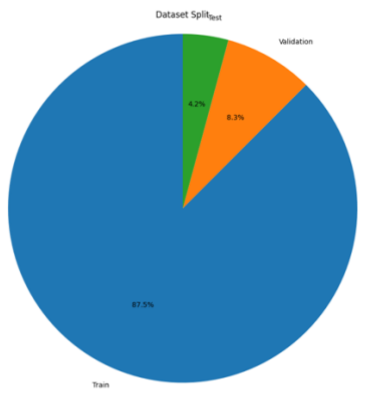


YOLO model architecture – 2016

3. Implementation

The dataset being used for this project consists of vehicles on the street images extracted from surveillance cameras. The data is classified into 6 classes representing 6 types of vehicles: bicycle, bus, car, motorbike, person, and truck. The original dataset is also applied augmentations to ensure the robustness and generalization. These augmentations include horizontal flipping, saturation adjustment ranging from -61% to +61%, brightness variation between -25% and +25%, and random noise affecting up to 2% of pixels. These transformations were used to simulate different environmental and lighting conditions to simulate real-world inference.

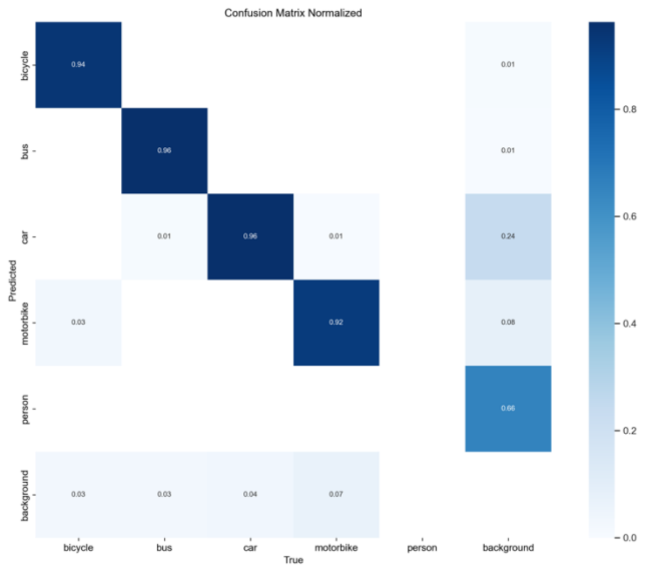


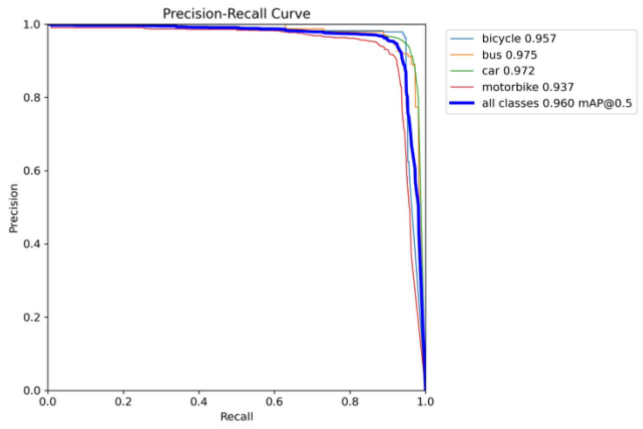


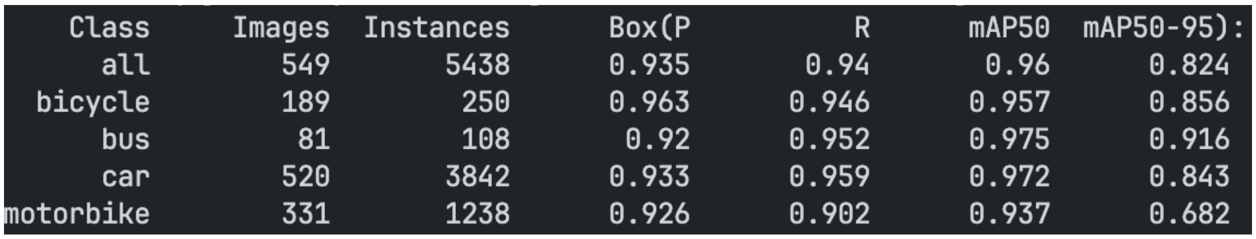
While feeding to the model, all training images were resized to 768×768 pixels to make sure that the model is still able to detect vehicles from further distance. For YouTube live streams, I use Streamlink and OpenCV libraries to handle taking the live stream to local machine and framing, writing on the videos. Streamlink is used to retrieve and buffer a 720p stream from YouTube, which then is passed into the YOLOv8 model for prediction. I use YOLOv8s model for this project because of its stability and high accuracy when compared to other more recent models like YOLOv9, YOLOv10, or YOLO11. Each video stream is handled by a thread using the Streamlink library, and OpenCV to process and display the video frames. The output of each frame is rendered with bounding boxes, class labels, and confidence scores, then displayed via OpenCV in real-time. A vehicle counter is also integrated into the pipeline to estimate crowd density, showing "CROWDED" when the number of detected vehicles exceeded a predefined threshold of 10 vehicles in the frame at a time. The confidence threshold is set to 0.4 to make sure the balance between minimizing false positives and retaining small or partially covered vehicles. Additionally, I also include a summary window displaying the crowd status of each stream. To ensure efficiency, only the most recent frame per stream is retained using a queue, and performance is tracked using live FPS metrics.

4. Result

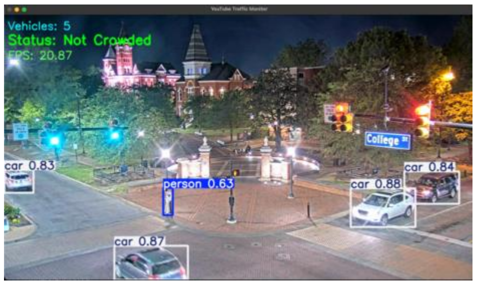
The performance of the YOLOv8s model is evaluated using both quantitative metrics and qualitative observations from real-world video streams. For the test set, the model achieved mAP50 approximately 0.96, while stricter for mAP50-95, around 0.82. These metrics suggest strong object localization and classification performance, especially given the compact size of the YOLOv8s model. In addition to mAP, the model demonstrated a precision score of 0.935 and a recall score of 0.94, indicating a good balance between minimizing false positives and capturing as many true vehicle detections as possible.

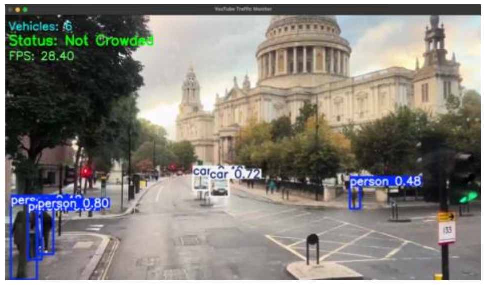


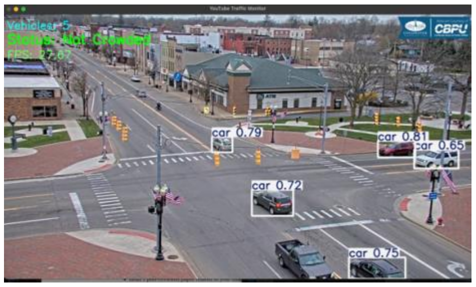



```
    Class     Images  Instances      Box(P          R      mAP50  mAP50-95):
      all        549       5438      0.935       0.94       0.96      0.824
  bicycle        189        250      0.963      0.946      0.957      0.856
      bus         81        108       0.92      0.952      0.975      0.916
      car        520       3842      0.933      0.959      0.972      0.843
motorbike        331       1238      0.926      0.902      0.937      0.682
```

In live video scenarios, the model consistently achieved around 30 FPS, mostly due to not strong computer graphic (Laptop GPU NVIDA 4070). However, for nighttime, or low light condition, the fps is reduced significantly with much lower accuracy. But this is already predicted as lack of dataset mentioned above.







5. Discussion

One of the primary issues is due to the limitation in dataset. While it has around 5800 images before augmentations, the dataset does not include different weather conditions, nighttime, or different camera angles. Therefore, the model struggles to detect vehicles on less common road environments such as rain, fog, or nighttime. Adding to the dataset more samples from low-light and diverse traffic conditions would likely improve robustness.

Furthermore, since the system relies on extracting live YouTube streams using Streamlink library, any network fluctuation or frame drop could lead to skipped frames in the OpenCV display. This is one of the reasons the FPS archived in this project is not too high, around 30 FPS. Besides that, the dropped FPS issue also depends a lot on the computer power. This inconsistency affected the smoothness of detection output and made it harder to maintain a reliable measure of FPS.

6. Reference

Fernández-Sanjurjo, M., Jr., Bosquet, B., Mucientes, M., Brea, V. M., & Centro Singular de Investigación en Tecnoloxía Intelixentes (CiTIUS). (2019). Real-Time Visual Detection and tracking system for traffic monitoring. *Preprint Submitted to Engineering Applications of Artificial Intelligence*. https://apps.citius.usc.es/fd/media/publications/456/beforePeersReview/fernandez-sanjurjo19_eaai_1_20210618111923282.pdf

Khazukov, K., Shepelev, V., Karpeta, T., Shabiev, S., Slobodin, I., Charbadze, I., & Alferova, I. (2020). Real-time monitoring of traffic parameters. *Journal of Big Data*, *7*(1). https://doi.org/10.1186/s40537-020-00358-x

Azimjonov, J., & Özmen, A. (2021). A real-time vehicle detection and a novel vehicle tracking systems for estimating and monitoring traffic flow on highways. *Advanced Engineering Informatics*, *50*, 101393. https://doi.org/10.1016/j.aei.2021.101393

Joseph, R., Santosh, D., Ross, G., & Ali, F. (2015). You only look once: Unified, Real-Time Object Detection. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1506.02640