


Red-light Running Detection

Thien Doanh Le

(International University, Ho Chi Minh City, Vietnam; Vietnam National University, Ho Chi Minh City, Vietnam)

 <https://orcid.org/0009-0002-9261-5223>, ITCSIU22237@student.hcmiu.edu.vn)


Duc Luan Dang

(International University, Ho Chi Minh City, Vietnam; Vietnam National University, Ho Chi Minh City, Vietnam)

ITCSIU19157@student.hcmiu.edu.vn)


Thi Quynh Nhu Duong

(International University, Ho Chi Minh City, Vietnam; Vietnam National University, Ho Chi Minh City, Vietnam)

 <https://orcid.org/0009-0007-0728-1406>, ITCSIU22202@student.hcmiu.edu.vn)

Kha Tu Huynh*

(International University, Ho Chi Minh City, Vietnam; Vietnam National University, Ho Chi Minh City, Vietnam)

 <https://orcid.org/0000-0001-8262-4703>, hktu@hcmiu.edu.vn)

Abstract: Red-light Running increases the risk of collisions and traffic accidents. When a car runs a red light, it can cause a collision with other vehicles moving along the main road, causing serious accidents and even leading to casualties. In Vietnam, many traffic accidents are caused by red-light running. This research paper presents a novel approach for detecting red-light running violations for Vietnamese intersections by leveraging object detection techniques and the YOLO (You Only Look Once) algorithm, a deep neural learning model that uses convolutional neural network architecture (CNNs) for object detection in real-time. The proposed system utilizes CCTV video footage to capture video frames, which are then processed through a trained YOLOv8 model to identify red-light violators. The system's performance is evaluated based on detection accuracy and processing speed and validated against a custom build dataset extracted from CCTV footages of Vietnamese streets. The experimental results demonstrate high accuracy and processing efficiency up to 93.4% mAP50, 89.2% precision and 92.6% recall, indicating that the proposed approach is suitable for deployment in the context of Vietnamese traffic conditions. The proposed system has significant potential to enhance road safety and mitigate the incidence of red-light running violations in Vietnam.

Keywords: Red-light Running, YOLOv8, intersections, traffic lights, traffic violation.

Categories: H.3.1, H.3.2, H.3.3, H.3.7, H.5.1

DOI: 10.3897/jucs.150763

* Corresponding Author: Kha Tu Huynh (hktu@hcmiu.edu.vn)

1 Introduction

With the progressive urbanization and increasing population numbers in Vietnam, the frequency of vehicular traffic has grown tremendously. The rapid increase in traffic has raised significant safety concerns among both authorities and citizens. A contributing factor to the increase of road dangers is the frequently observed flagrant violation of red-light traffic signals, known as red light running, mostly by motorists. This behavior is not only life-threatening but also leads to a rising rate of pedestrian, cyclist, and driver accidents, consequently causing fatalities. Vietnamese streets have a distinctive traffic environment that makes it difficult to enforce traffic laws efficiently due to a variety of road users, fluctuating vehicle speeds, and complex intersections [Le, H. L. et al. (2010)]. Due to the sheer number of cars and the complexity of urban traffic, traditional techniques of monitoring and enforcement were not able to keep up. Although existing traffic monitoring technologies are available and have already been put in use, their installation costs can reach as high as \$70,000 [Aleksandar, S. (2020)]. For that reason, an economical and sophisticated Red Light Running Detection System designed specifically for Vietnamese roadways is required to address this problem and improve road safety.

The main motivation behind developing a Red-Light Running Detection System using CCTV camera footage stems from the potential to leverage technology in addressing the challenges associated with red light violations on Vietnamese streets. The convergence of computer vision offers a cost-effective, accurate, and real-time solution for red light running, aiming to improve road safety and reduce accidents. Utilizing CCTV footage and image processing algorithms, this system can identify instances of vehicles passing through red lights, detecting and recording violations in real-time.

2 Related Works

The detection and prevention of red-light violations, speeding, and other traffic-related incidents are critical concerns in modern transportation systems. Several research papers have addressed these issues using various approaches.

One common method is the use of computer vision techniques, such as object detection algorithms, to analyze video data captured by cameras at intersections. [De Goma, J. C. et al.(2020)] proposed a method which achieves high accuracy rates, up to 100% for detecting red-light runners and 92.1% for speeding violations; however, it may be limited by issues such as lighting conditions, camera angles, and video quality..

Another approach is the use of machine learning algorithms, such as co-training-based methods [Momin, Bashirahamad F. et al. (2015)]. The primary feature selected for detection is "haar." A haar-training classifier is trained, and Adaboost is used to create a strong classifier for vehicle detection. This research enhances video surveillance and traffic monitoring by enabling efficient vehicle detection and attribute-based vehicle search. It provides a method for efficient vehicle management and criminal investigations, enhancing the efficiency of traffic management and traffic monitoring. After detecting vehicles, the next step is to search for specific vehicles based on attributes such as color, date and time, speed, and direction of travel. This research contributes to the field of video surveillance and traffic monitoring by providing a method for efficient vehicle detection and attribute-based vehicle search, which has applications in traffic management and criminal investigations.

In 2017, [Brasil, R. H., & Machado, A. M. C. (2017)] proposed a red-light runner detection system consisting of a camera and a computer embedded in vehicles. An

algorithm was used to processes recorded videos, and a prototype was developed for monitoring work vehicles without interfering with driving, acting as an educational tool. Tests were conducted using street videos and a benchmark video. The video processing time is less than one-tenth of the video duration and achieves an impressive accuracy of approximately 95.8%.

Some research papers have focused on developing novel algorithms for detecting red-light runners, such as the “dilemma zone” approach [Zaheri, D., & Abbas, M.(2015)] or the use of vehicle speed and trajectory data [Luo, D., Huang, X., & Qin, L. (2008)]. The system first identifies the red traffic signal and then analyzes the movement of the vehicle within the detection zone. The approach uses cubic splines interpolation to fit vehicle tracks and detects red light violations, and Hough transform to detect stop lines in X-Y space, eliminating false positives and enhancing road safety by deterring red light running. This method can achieve high accuracy rates, up to 96%, but may be limited by specific scenarios or conditions.

In addition, some studies have explored using probabilistic models, such as Bayesian networks [Chen, X., Zhou, L., & Li, L. (2019)]. The paper suggested a solution for the critical issue of red-light-running (RLR) at signalized intersections through a unique method. The authors propose a probabilistic stop-or-go prediction model based on Bayesian networks (BN) using continuous vehicle trajectory data obtained from radar sensors. This model aims to improve RLR prediction accuracy and interpretability compared to traditional methods using embedded traffic sensors. The study’s findings reveal that the BN model outperforms other machine learning models and the inductive loop detection (ILD) model, offering a promising approach for enhancing traffic safety at signalized intersections through probabilistic RLR prediction based on continuous trajectories.

In [Li, M., Chen, X., Lin, X., Xu, D., & Wang, Y. (2018)], the research addresses the critical safety concern of red light running (RLR) at signalized intersections by analyzing nine months of RLR events extracted from high-resolution traffic data collected by loop detectors at three intersections. This paper identifies key factors that significantly impact RLR behaviors, including occupancy time, time gap, used yellow time, time left to yellow start, the behavior of preceding vehicles during yellow, and the presence of vehicles in adjacent lanes. To predict RLR in real time, a modified rare events logistic regression model is proposed, which outperforms standard logistic regression models. This research leverages readily available loop detector data and also offers a great potential for practical applications in intersection safety improvement.

In [Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016)], the paper proposes a methodology that combines high-resolution signal controller data with conventional stop bar loop detection to identify vehicles that enter the intersection after the start of the red signal, a critical moment for RLR crashes. The methodology was validated through on-site video collection at several locations and refined to reduce false RLR indications. A case study demonstrated that increasing the green signal time on the side street by 4% led to a significant 34% reduction in daily RLR counts and 1.7 times decrease in the likelihood of RLR incidents, offering a cost-effective safety improvement. Law enforcement and transportation agencies can utilize this technique to efficiently manage and deploy safety resources, especially in cases with limited crash histories or infrequent incidents at intersections.

Some of the studies have also applied deep learning methods, including CNN, for object detection or classification applications such as pothole detection. Examples include the works of [Xiong N., He J., Park J.H., Cooley & D., Li Y.(2009)] and the detection of vehicles in various lanes and traveling directions, for instance, [Lim Kuoy Suong, and

Kwon Jangwoo, (2017)]. These methods have been shown to attain high accuracy levels but perhaps with high demands in volumes of training data and computational power.

In [Hongyu Huang et al. (2012)], the authors research a method to extract a mobility model for META using data from the GPS of taxis in the city. The proposed system collected data from 4000 taxis equipped with GPS and map-matched them to the road map using heuristic algorithms. A prototype system could estimate the travel patterns and characteristics of taxi movement.

Specifically, it appears that the authors of these papers have applied different methodologies to handle issues such as traffic red-light-running incidents, detection of potholes, and classification of vehicles. After testing the proposed systems and methods, there is promising accuracy and efficiency for application in real-world road situations to enhance safety and traffic management. Building upon these promising results, the next phase of research explores more advanced methodologies to tackle similar traffic-related challenges. In particular, we propose the integration of YOLOv8, a state-of-the-art object detection model, in combination with DeepSORT, a robust multi-object tracking algorithm.

3 Methodology

The red-light running detection system in this paper was built using YOLOv8 and the Deep SORT algorithm. It also includes collecting traffic images at intersections to ensure that the YOLOv8 model is compatible and can adapt to the environment for vehicle and red-light classification and detection. When the dataset is ready, the model is trained, validated, and tested. The integrated DeepSORT algorithm and YOLOv8 models form a cohesive red-light running detection and tracking system. The system operating process is shown in Figure 1.

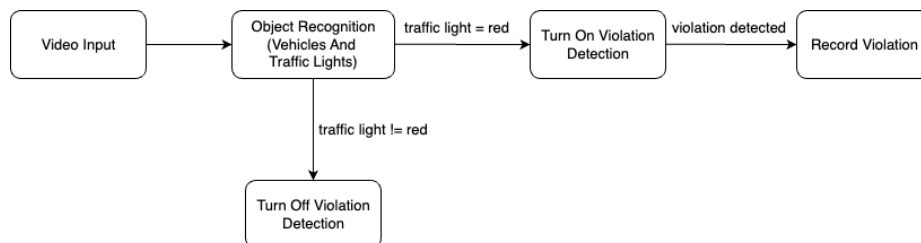


Figure 1: The System Operating Process

Firstly, the overview of YOLOv8 model used in the red light detection problem will be presented.

3.1 YOLOv8 Overview

YOLO, which is short for “You Only Look Once”, is an algorithm and architecture used extensively in computer vision tasks to detect objects in real time. What makes YOLO different from other object detection methods is because of its speed and efficiency. YOLO can quickly and accurately identify objects in images or video frames. Unlike

two-stage detection models like R-CNN, YOLO tackles object detection as a single regression problem. It directly predicts bounding boxes and class probabilities for the image at once using a single convolutional network, eliminating the need for region detection and classification steps [Ultralytics. (2023)].

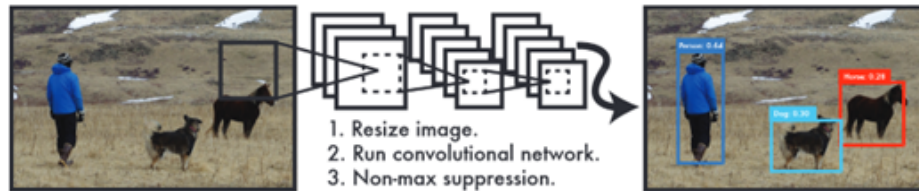


Figure 2: The YOLO Detection System

YOLOv8, the latest integration of YOLO, was chosen due to its reputation for delivering high accuracy detection and shorter inference times when from the older YOLO versions. Compared to its predecessor YOLOv5, YOLOv8 introduces two key changes. First, it shifts from Anchor Boxes to Anchor-Free Detection, departing from predefined templates for object localization and classification [Krishnakumar, M. (2023)]. This change addresses issues of rigidity and struggles with irregular object shapes. Second, YOLOv8 employs Mosaic Data Augmentation during training, combining four distinct images to help the model learn to detect objects in various positions and partial occlusion, although this augmentation is only used for the final 10 training epochs due to its potential impact on performance [Krishnakumar, M. (2023)].

Despite being labeled as the latest state-of-the-art [Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012)], YOLOv8 still remains in the developmental phase and is susceptible to modifications in the future, so there is currently no available official paper or documentation regarding the performance of this architecture. However, the Roboflow Team had performed several tests to help verify the legitimacy of this new architecture using their RF100 dataset, the research was then published on their website which yielded promising results. Particularly in real-world applications where YOLOv8 displays its superiority over its predecessor. Anyhow, this highly advanced ability of YOLOv8 in detection is very much dependent upon high-quality and correctly labeled data for correct detection and tracking in challenging traffic scenarios. Therefore, data collection is the important step for training a model and will be presented in the following section.

3.2 Data collection

Collecting high-quality datasets is the most essential part for training machine learning models effectively. Initially, the dataset collection involved locating appropriate locations for capturing traffic images. Therefore, around 2 hours' worth of Vietnamese intersection CCTV footages was collected. After that, those footages were imported into Photoshop to be sliced up frame by frame at around ten frame per second (FPS). The final dataset consists of around 3135 images, containing a total of 8 different classes (5 vehicle classes and 3 traffic light classes), with 3035 images used for training, 368 images for validation, and 121 images for testing.

In the final dataset, the class with the most annotations were "motorbike," followed by "car," "red-light," "green-light," "bike," "truck," "yellow-light," and "bus."

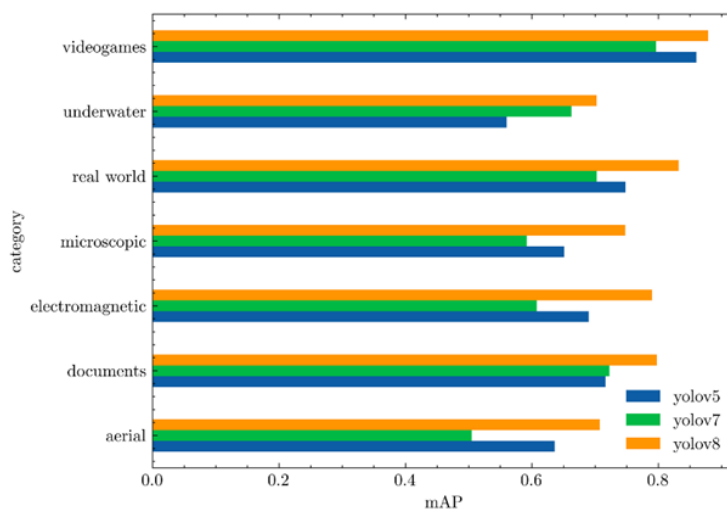


Figure 3: YOLOs average mAP@.50 against RF100 categories [Roboflow (2022)]

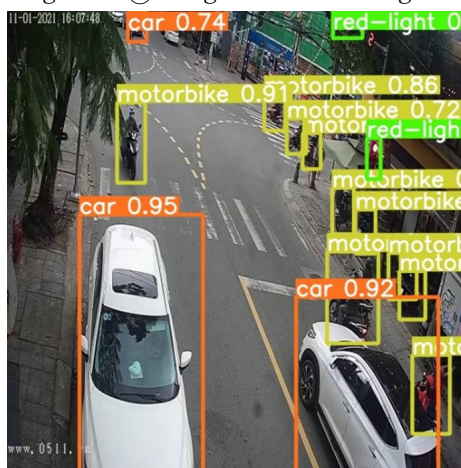


Figure 4: An annotated frame

After the data is collected in sufficient quantity and quality with the necessary characteristics, the red light detection model will be trained and evaluated for feasibility and effectiveness.

3.3 Model Training and Evaluation

Once the dataset is prepared, the training process can be initiated. Since YOLO is built upon the foundation of Convolutional Neural Networks (CNNs), it is commonly trained using supervised learning. The training process begins with gathering the necessary images, which will then be annotated. The annotated images will form a dataset which

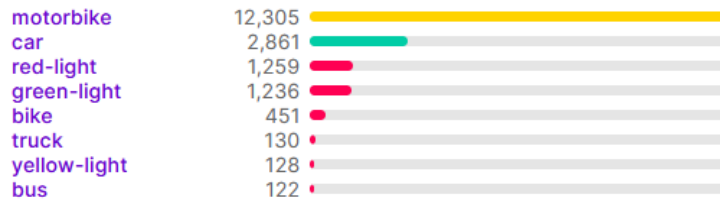


Figure 5: The number of annotations for each class

can be used for training models. In order to train the model, we used the Google Colab template provided by Roboflow [Jacob, S., Francesco. (2023)]. YOLOv8m, the third variant of the YOLOv8 model series, was selected for the training stage due to its capability to produce relatively high-accuracy detections compared to YOLOv8n and YOLOv8x, while also delivering faster inference times when compared to YOLOv8L and YOLOv8X [Figure 3].

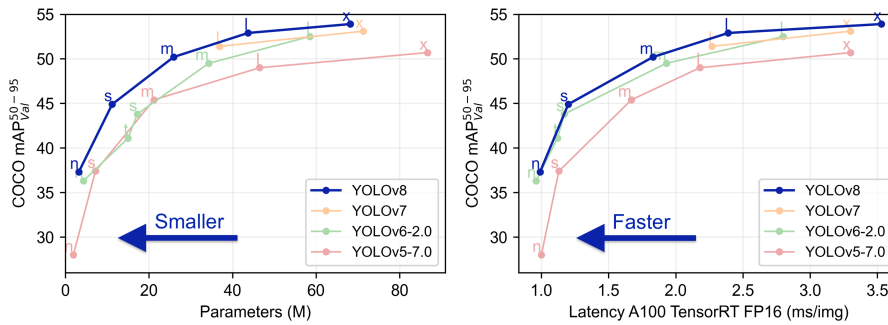


Figure 6: Comparison between YOLOv8 to others YOLO models [Jacob, S., Francesco. (2023)]

Each model generation underwent hyperparameter tuning with an image size configured to 640 pixels with around 120 epoch iterations. Once the training phase is finished the model undergoes validation and testing procedures to calculate metrics such as precision, recall, and mAP. If the model doesn't meet performance expectations, modifications will be made to the dataset to improve the results in the next training session. This iterative process continues until the model with a satisfactory level of performance is created. The model training process is presented in [Figure 7].

Tracking in deep learning is the task of predicting the positions of objects throughout a video. This task is typically done by getting the initial set of detections, assigning unique IDs, and tracking them throughout frames of the video feed while maintaining the given IDs. DeepSORT (Deep Learning-based SORT) is an advanced object-tracking framework that combines deep learning techniques with the traditional SORT (Simple Online and Realtime Tracking) algorithm to provide robust and accurate tracking of objects in video sequences. While SORT tracking algorithm shows strong performance in terms of tracking accuracy and precision, it still has limitations in scenarios involving occlusion,

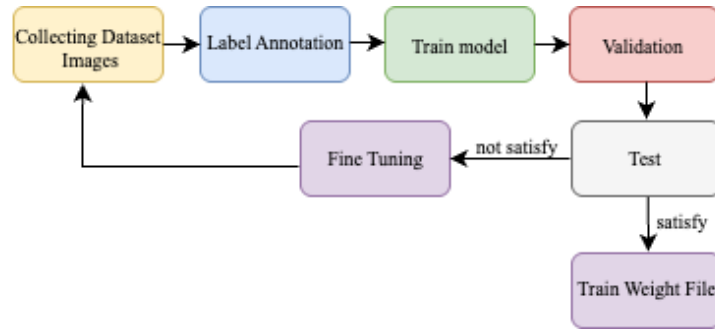


Figure 7: Model training process

this often leads to the creation of high ID numbers [Chen, X., Zhou, L., & Li, L. (2019)]. DeepSORT, on the other hand, employs deep learning to create appearance embeddings – compact numerical representations of an object’s visual characteristics. This means that even when objects are occluded and their motion features are undetectable, DeepSORT can still recognize objects based on their unique appearances which allow it to maintain tracking continuity and correctly associate an object with its original track, even when it becomes temporarily invisible due to occlusion [Chen, X., Zhou, L., & Li, L. (2019)].

For object tracking, the authors use the DeepSORT technique which will be presented in the following content.

3.4 DeepSORT Tracking Algorithm Overview

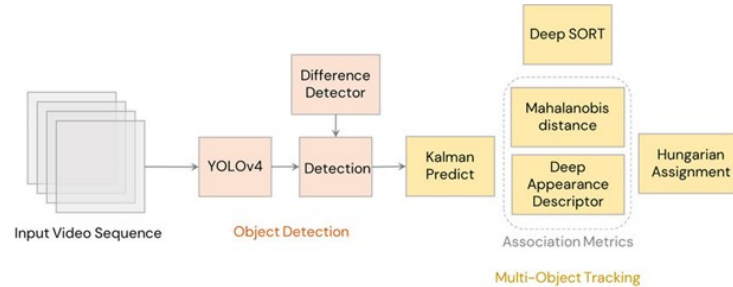


Figure 8: DeepSORT architecture [Chen, X., Zhou, L., & Li, L. (2019)]

DeepSORT (Deep Simple Online and Realtime Tracker) extends the SORT algorithm by incorporating deep learning-based appearance features to improve object tracking. It utilizes YOLOv4 for object detection, a Kalman Filter for motion prediction, and a Deep Appearance Descriptor for object re-identification. The Mahalanobis distance metric and the Hungarian Assignment algorithm enhance data association, significantly reducing identity switches. Figure 8 illustrates the detailed architecture of DeepSORT.

In vehicle tracking, DeepSORT is widely used in traffic monitoring, autonomous driving, and intelligent transportation systems. It enables robust multi-object tracking,

handling occlusions and varying speeds, ensuring accurate vehicle movement analysis and traffic flow optimization.

3.5 Implementation

To implement the system, we will need to download the Ultralytics library, which comes bundled with the YOLOv8 models [Roboflow (2022)]. Additionally, since we will be using a NVIDIA GPUs to run YOLO, it is important to install CUDA, a parallel computing platform and API created by NVIDIA. It utilizes the computational power of NVIDIA GPUs to accelerate various computing tasks, including object detection. The first step in developing the system is to design a simple graphical user interface (GUI) for easy navigation. Therefore, a GUI was designed using PyQt5 containing five buttons and a table.

To identify instances of vehicles crossing at a red traffic light, coordinates for the area where violations are monitored need to be set. This task must be done manually because every intersection is unique. Once the coordinates are defined, it's necessary to implement a straightforward logic for enabling and disabling the detection process. This implies that the system will initiate the violation detection process only when the red light is active and deactivate the detection when either the yellow or green light is on.



Figure 9: Different stages of traffic light. (1) On green light the system will halt its violation tracking. (2) The system resume violation detection on red-light

In order to determine whether a vehicle performed red-light running or not, we must examine the vehicle's position in relation to the defined restricted region. This can be accomplished by placing a point at the center of each vehicle's bounding box. To determine the center point coordinates of bounding boxes, we used the following formula:

$$C_x = \frac{x_0 + (x_1 - x_0)}{2} \quad (1)$$

$$C_y = \frac{y_0 + (y_1 - y_0)}{2} \quad (2)$$

C_x : the x coordinate of the center point

C_y : the y coordinate of the center point

x_0 : the x-coordinate of the first end point

x_1 : the x-coordinate of the second endpoint

y_0 : the y-coordinate of the first endpoint

y_1 : the y-coordinate of the second endpoint

Once the x and y coordinates of the center point have been calculated, we can utilize them for tracking the positions of vehicles. By monitoring the changes in the values of C_x and C_y and comparing them with the given limit lines over individual frames, we can trace the location and movement of the vehicles. This process allows us to detect and mark potential red-light running violations or instance, in [Figure 9], when a vehicle is within the area where $x_0 < C_x < x_1$ and $y_0 < C_y < y_1$ then it will be categorized as a violation. When a violation is detected, it will be displayed on the table. The displayed

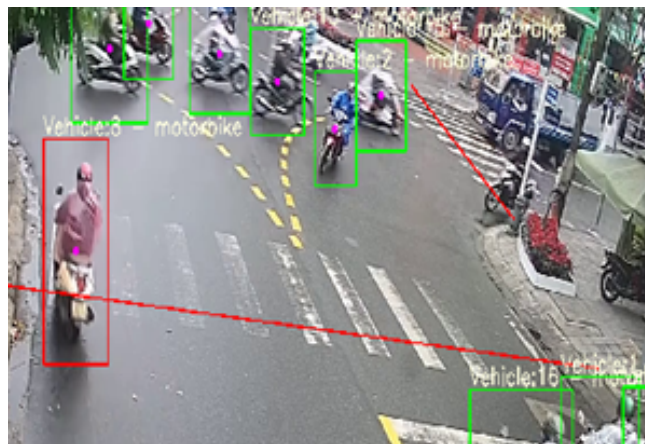


Figure 10: Motorbike was caught performing red-light running.

information includes the violator's ID, vehicle type, the timestamp from the video when the violation occurred, and an image snippet capturing the moment of the violation. Referring to [Figure 10], the vehicle with ID number 6 has been observed crossing the red light, and therefore, this incident should be recorded as a violation in the table. Nevertheless, there is an issue with the approach. In most cases, Vietnamese traffic laws do not penalize vehicles that make right turns on a red light. In the case of [Figure 9], where there is no traffic sign prohibiting vehicles from turning right during a red light. Therefore, it is necessary to establish an additional limit line to exclude marked vehicles from the list of violators. If a vehicle was marked as a potential violator when crossing the first line at red light, it will be removed from the list of violations when turning right to cross the second line.

Following the detailed implementation of YOLOv8 and DeepSORT, it is crucial to evaluate the system's performance under real-world conditions to validate its effectiveness. The experimental results, which provide insight on the suggested approach's overall impact, accuracy, and efficiency in a range of traffic conditions, are presented in the following section.


	IDs	Type	Timestamp	Image
1	8	motorbike	00:01	

Figure 11: Violation was added to table.

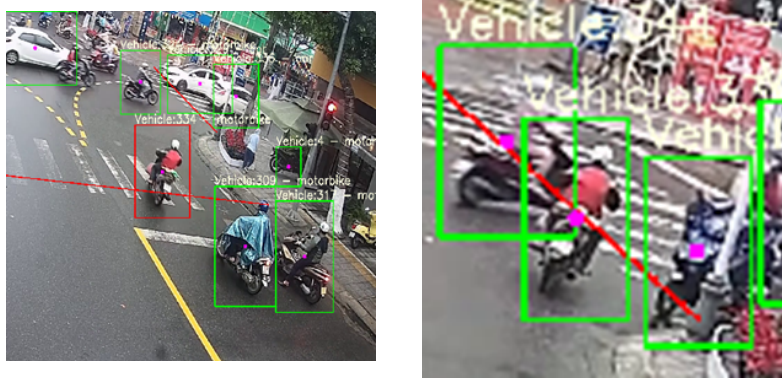


Figure 12: (1) Motorbike got marked for red-light running. (2) Motorbike is removed from the red-light violation list after crossing the second limit line.

4 Experiment Results

In this section, we present the outcomes of our experiments conducted to evaluate the performance of the system. To begin, our initial step involves assessing the performance of the YOLOv8 model. Subsequently, the system was set to go through a stress test, aiming to test its efficacy and to uncover its limitations. Everything was done using a computer with these specifications [Table 1]:

4.1 Model Evaluation Metrics

To evaluate the model detection performance, these following metrics was used:

Component	Specification
GPU	NVIDIA GeForce GTX 1660 (1,408 CUDA Cores)
Processor	AMD Ryzen 5 2600 Six-Core Processor (12 CPUs)
RAM	16GB

Table 1: Specification of the computer used for testing.

4.1.1 Recall and precision.

$$Precision = \frac{TP}{TP + FP} \quad (3)$$

$$Recall = \frac{TP}{TP + FN} \quad (4)$$

Precision determines how many of the predicted positive instances are actually true positives. True Positive (TP) is the number of correctly predicted positive instances, and False Positives (FP) is the number of instances that were predicted as positive but are actually incorrect predictions.

Recall measures how many of the actual positive instances were correctly predicted as positive by the model. Moreover, False Negative (FN) is the number of instances that are actually positive but were predicted as negative (missed predictions)

4.1.2 Average precision (AP)

$$AP = \int_0^1 P(R) dR \quad (5)$$

To compute AP, we first calculate precision and recall values at multiple thresholds for the predictions. These precision-recall pairs form a curve, and AP is the area under this curve which can be seen in [Table 2].

4.1.3 Mean Average Precision (mAP)

$$mAP = \frac{\sum_i^N AP_i}{N} \quad (6)$$

mAP is a metric used to evaluate the performance of object detection or recognition models across multiple classes or categories. It is represented by the average of the Average Precision (AP) values calculated for each class I where N is the number of object classes.

In our study, we employed two mAP variants: mAP50, evaluating precision at 50% recall, and mAP50-95, assessing precision across the 50% to 95% recall range.

4.2 Training Results and Discussions

In our evaluation of the custom trained YOLOv8 model, we considered a range of training evaluation metrics mentioned in [4.1] to evaluate the model. These metrics were calculated in order to provide insights into the model’s capabilities and areas for improvement. Here is the result after we validated the model using a validation dataset consisting of 252 images.

Class	Images	Instance	Precision	Recall	mAP50	mAP50-95
all	252	3040	0.892	0.926	0.934	0.763
bike	252	66	0.886	0.827	0.875	0.600
bus	252	16	0.756	0.971	0.856	0.730
car	252	414	0.938	0.928	0.964	0.845
green-light	252	209	0.969	0.990	0.992	0.865
motorbike	252	2068	0.935	0.938	0.977	0.790
red-light	252	207	0.953	0.986	0.985	0.834
truck	252	31	0.827	0.806	0.866	0.682
yellow-light	252	29	0.870	0.966	0.959	0.735

Table 2: Validation results of the model after training.

The model can accurately detect vehicles such as “motorbike” and “car” with high precision and performs relatively well on “bike” [Table 2]. However, it still struggles when detecting other types of transport like “truck,” and “bus” [Table 2]. The model trained on Vietnamese urban streets, where cars and motorbikes are the primary transport vehicles, caused confusion between classes due to the frequency of their appearances. To improve the model’s accuracy, additional images need to be added to the existing dataset, as the difference in vehicle appearances can lead to confusion.

To further illustrate the performance of each class, a confusion matrix is created to visualize the classification results. This matrix displays the true positive, true negative, false positive, and false negative values for each class in the classification model. It effectively demonstrates how well the model has predicted the different classes and where it might be struggling.

Based on [Figure 13], the relatively high score for the motorbike class within the background category, around 0.70 in the confusion matrix, indicates a notable challenge in accurately distinguishing motorbikes from other elements in the dataset. This suggests that the model struggles to discern motorbikes distinctly, potentially misclassifying them as part of the background category due to overlapping features or a lack of specific patterns unique to motorbikes.

However, despite this discrepancy highlighted in the evaluation metrics, the practical impact of this misclassification has not been a significant problem when put into practice. Upon deploying the system, despite the result of the confusion matrix, the overall performance was not substantially affected, maintaining the system’s functionality.

The precision-recall curve illustrates the model’s performance across different object categories [Figure 14]. The precision scores reveal the model’s ability to accurately

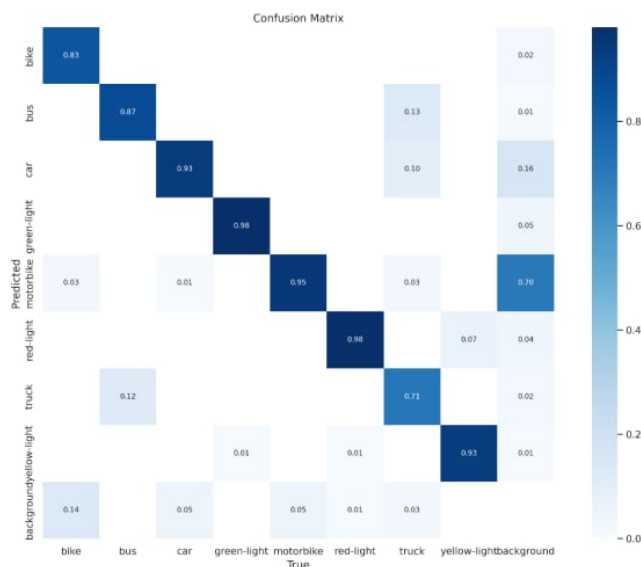


Figure 13: The Confusion Matrix of All Classes.

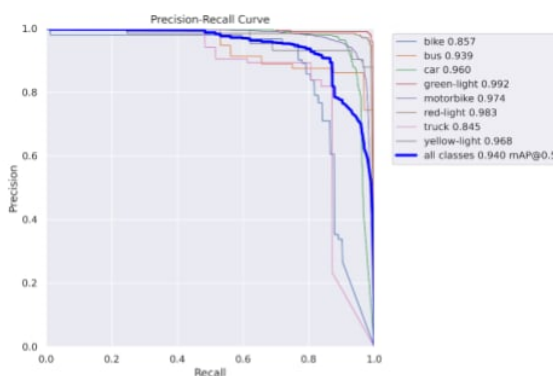


Figure 14: Precision-Recall Curve.

recognize specific classes in the dataset. Notably, the model achieves high precision in identifying various classes, such as cars at 0.960, red-light at 0.983, and an impressive 0.992 for green light. Important categories like bikes and buses also exhibit respectable precision scores of 0.857 and 0.939, respectively, indicating the model’s capability in distinguishing these entities. Additionally, the model shows a strong precision of 0.974 for motorbike classification. Though the precision for the truck class is slightly lower at 0.845, the overall mean average precision (mAP@0.5) for all classes collectively stands at a commendable 0.940. This analysis emphasizes the model’s precise classification across diverse object categories, highlighting its effectiveness in recognizing objects

within the dataset.



Figure 15: Testing the trained model by employing modified dataset images to simulate harsh weather conditions.

A series of diverse testing images is presented, intentionally altered to replicate harsh weather conditions, including low gamma settings for reduced visibility and the introduction of static effects to simulate heavy rain and foggy environments. These modifications aim to closely resemble real-world adverse weather scenarios, challenging the model's recognition capabilities. Remarkably, the conducted testing exhibited promising outcomes, with the model successfully detecting and identifying objects correctly in 100% of the provided images. This impressive performance underscores the model's ability to navigate and recognize objects amidst challenging environments, demonstrating its robustness and potential for effective real-world application in adverse weather conditions.

The effectiveness of a CCTV system relies heavily on the quality of footage and the positioning of cameras. Low quality footage can cause issues in detecting objects or generating false positives. Additionally, the camera's location may miss certain violations. Weather conditions and lighting can also significantly impact the system's effectiveness, with footage captured during severe weather conditions or with minimal lighting at night rendering the entire video unusable.

5 Conclusion

The Red-Light Running Detection System, implemented with the YOLOv8 model and the DeepSORT tracking algorithm, seeks to combat instances of red-light violations occurring on Vietnamese roadways. The system employs computer vision and deep learning methodologies to enhance road safety and mitigate accidents. Empirical investigations

demonstrate that the YOLOv8 model exhibits commendable performance in accurately detecting a wide range of vehicle categories. However, it encounters difficulties in accurately distinguishing less common types such as “bus” and “truck”. Additional fine-tuning and augmentation of data are required to enhance precision. The system exhibits promising potential, however, it is imperative to address and update its limitations in order to achieve optimal performance in real-world scenarios. Overall, the development of a Red-Light Running Detection System using computer vision and deep learning could be a significant step towards improving road safety in Vietnam and addressing the growing concerns related to red-light violations. As urbanization and traffic congestion continued to rise, leveraging technology to enhance law enforcement and traffic management became increasingly important in ensuring the safety of all road users.

In the future works, we will study DeepSORT [Azhar, M. I. H. et al. (2020, August)] incorporating additional features from deep learning models (such as object embeddings) to improve accuracy when tracking objects across multiple frames in our system. The comparison of DeepSORT technique and other techniques for object detection problems will also be studied further. Besides, Fuzzy-based approaches have shown a good performance in image processing [García-Zamor et al. (2024)] so we will research fuzzy-based approach as one of our coming works.

References

- [Aleksandar, S. (2020)] Aleksandar, S. (2020, October 19):“The average installation cost per intersection of an Adaptive Traffic Control System (ATCS) is \$65,000”;Transportation Research Board.
- [Azhar, M. I. H. et al. (2020, August)] Azhar, M. I. H., Zaman, F. H. K., Tahir, N. M., & Hashim, H. (2020, August)“People tracking system using DeepSORT”. In 2020 10th IEEE international conference on control system, computing and engineering (ICCSCE) (pp. 137-141). IEEE.
- [Brasil, R. H., & Machado, A. M. C. (2017)] Brasil, R. H., & Machado, A. M. C. (2017):“Automatic detection of red light running using vehicular cameras”; IEEE latin america transactions, 15(1), 81-86.
- [Chen, X., Zhou, L., & Li, L. (2019)] Chen, X., Zhou, L., & Li, L. (2019):“Bayesian network for red-light-running prediction at signalized intersections”; Journal of Intelligent Transportation Systems, 23(2), 120-132.
- [De Goma, J. C. et al.(2020)] De Goma, J. C., Bautista, R. J., Eviota, M. A. J., & Lopena, V. P. (2020, April):“ Detecting red-light runners (RLR) and speeding violation through video capture”;2020 IEEE 7th international conference on industrial engineering and applications (ICIEA);pp. 774-778; IEEE
- [García-Zamor et al. (2024)] [García-Zamora, D., Cruz, A., Neres, F., Santiago, R. H., de Hierro, A. F. R. L., Paiva, R., ... & Bustince, H. (2024)]“Admissible OWA operators for fuzzy numbers”. Fuzzy Sets and Systems, 480, 108863.
- [Hongyu Huang et al. (2012)] Hongyu Huang, Daqiang Zhang, Yanmin Zhu, Minglu Li, Min-You Wu, (2012):“ A Metropolitan Taxi Mobility Model from Real GPS Traces”, Journal of Universal Computer Science 18(9),1072-1092.
- [Jacob, S., Francesco. (2023)] Jacob, S., Francesco. (2023, January 11; “What is YOLOv8? The Ultimate Guide. Roboflow”; “<https://blog.roboflow.com/whats-new-in-yolov8/>”
- [Krishnakumar, M. (2023)] Krishnakumar, M. (2023, April 13): “A gentle introduction to yolov8”; “<https://wandb.ai/mukilan/wildlife-yolov8/reports/A-Gentle-Introduction-to-YOLOv8-Vmldzo0MDU5NDA2>”

- [Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012)] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012): "Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25."
- [Le, H. L. et al. (2010)] Le, H. L., Nguyen, T. H., Dang, Q. T., & Nguyen, T. D. (2010) "Some Applications of Advance Technology in Solving Transport Means Surveillance Problem in Vietnam"; *International Journal of the Society of Materials Engineering for Resources*, 17(1), 5-8.
- [Li, M., Chen, X., Lin, X., Xu, D., & Wang, Y. (2018)] Li, M., Chen, X., Lin, X., Xu, D., & Wang, Y. (2018): "Connected vehicle-based red-light running prediction for adaptive signalized intersections"; *Journal of Intelligent Transportation Systems*, 22(3), 229-243.
- [Lim Kuoy Suong, and Kwon Jangwoo, (2017)] Lim Kuoy Suong, and Kwon Jangwoo, (2017): "Detection of Potholes Using a Deep Convolutional Neural Network", *Journal of Universal Computer Science* 24(9), 1244-1257
- [Luo, D., Huang, X., & Qin, L. (2008)] Luo, D., Huang, X., & Qin, L. (2008, August): "The research of red light runners video detection based on analysis of tracks of vehicles"; 2008 International Conference on Computer Science and Information Technology (pp. 734-738), IEEE.
- [Momin, Bashirahamad F. et al. (2015)] Momin, Bashirahamad F., and Tabssum M. Mujawar: "Vehicle detection and attribute-based search of vehicles in video surveillance system."; 2015 international conference on circuits, power and computing technologies [ICCPCT-2015]; IEEE, 2015.
- [Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016)] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016): "You only look once: Unified, real-time object detection"; in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779-788.
- [Roboflow (2022)] Roboflow (2022), Notebooks, GitHub: "<https://github.com/roboflow/notebooks>"
- [Sanyam (2022)] Sanyam (2022, June 21): " Understanding Multiple Object Tracking using DeepSORT. LearnOpenCV" "<https://learnopencv.com/understanding-multiple-object-tracking-using-deepsort/>"
- [Ultralytics. (2023)] Ultralytics. (2023); Ultralytics, GitHub: "<https://github.com/ultralytics/ultralytics>"
- [Xiong N., He J., Park J.H., Cooley & D., Li Y.(2009)] Xiong N., He J., Park J.H., Cooley & D., Li Y.(2009): "A Neural Network Based Vehicle Classification System for Pervasive Smart Road Security", *Journal of Universal Computer Science*, vol. 15, no. 5 (2009), 1119-1142
- [Zaheri, D., & Abbas, M.(2015)] Zaheri, D., & Abbas, M. (2015, September): "An algorithm for identifying red light runners from radar trajectory data"; 2015 IEEE 18th International Conference on Intelligent Transportation Systems (pp. 2683-2687), IEEE.