VIETNAM NATIONAL UNIVERSITY OF HOCHIMINH CITY
THE INTERNATIONAL UNIVERSITY
SCHOOL OF COMPUTER SCIENCE AND ENGINEERING


# Red-light Running Detection

By
Đặng Đức Luân
ITITIU19157

A thesis submitted to the School of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
Bachelor of Information Technology

Ho Chi Minh City, Vietnam
2024

**Red-light Running Detection**

APPROVED BY:

_____ ,
Dr. Huynh Kha Tu

_____
Dr. Nguyen Trung Ky

_____
(*Typed Committee name here*)

_____
(*Typed Committee name here*)

_____
(*Typed Committee name here*)

THESIS COMMITTEE
(Whichever applies)

## ACKNOWLEGMENTS

I want to sincerely thank Dr. Huynh Kha Tu for all of her help with this thesis, including her amazing advice, constant support. Her mentorship greatly shaped the results and direction of this research. Aside from improving this thesis, Dr. Huynh Kha Tu's dedication and thoughtful suggestions have had a significant impact on my development as a researcher. I sincerely appreciate her invaluable contributions to this project and all of the assistance she provided, including the time she spent trying to help me over the summer.

Lastly, I would also like to thank the entire faculty, staff, and teachers at IU. Their support and academic knowledge have been invaluable to my learning experience, building a collaborative and intellectually engaging atmosphere that helped my development as a researcher.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# ABSTRACT

As the world population becomes larger, the number of vehicles on the road increases dramatically, leading to a rise in road violations. One of the most common types of violation is red-light running (RLR). This thesis presents a comprehensive study about a cost-effective RLR detection system. By incorporating neural networks and object detection methods, the development of this system was able to detect RLR violations with high accuracy. This thesis presents a comprehensive study of a new approach for red light running detection system using the YOLOv8 detection model. The main goal of this research is to improve road safety by developing a precise and dependable method of identifying traffic signal offenses, with a focus on red-light violations. This thesis uses state-of-the-art YOLO architecture to develop a reliable and affordable solution that can be easily applied to stop the increasing number of traffic violations related to driving under red lights. The research involves various processes, such as collecting datasets from CCTV footage and different sources across the internet. Additionally, different techniques, such as data augmentation, are applied to enhance the model's performance in real-world applications. The processes include training, validating, and testing the object detection model. To track violations, the research also incorporates DeepSORT as the proposed tracking algorithm on top of YOLOv8 to flag detected violators. Furthermore, there is an implementation of a graphics user interface (GUI) to facilitate easy navigation and handling for potential users. The end system was able to achieve relatively high performance, with 92.4% precision, 91.2% recall, and 94.0% mAP50.

# CHAPTER 1

# INTRODUCTION

## 1.1. Background

With the rapid urbanization and exponential increase in vehicular traffic witnessed in nations like Vietnam, the management of traffic flow and road safety has become a critical societal concern. The increase in the number of vehicles on the roads, along with other factors such as diverse road user behaviors, makes it extremely difficult for law enforcement to keep up. Out of many things that cause accidents on roads, RLR (Red-Light Running) is one of the most common violations committed in Vietnam and a major reason for crashes, injuries, and deaths. It is estimated that 20% of traffic accidents occurring at signalized intersections result from not following traffic lights [1]. According to WHO, the number of road traffic accidents in low-income countries such as Vietnam increased in the period from 2013 to 2016, and the risks of road death in low-income nations are also three times higher than that of high-income countries [2].

To address these challenges, the Vietnamese government has implemented traffic surveillance camera system around the country [3]. However, these projects are typically extremely costly and only limited to areas with heavy traffic, such as city centers and highways. For instance, in Ba Ria – Vung Tau Province, a traffic surveillance camera system comprising 89 cameras required approximately 3.5 million USD, averaging around 40,000 USD per camera. [4]. Due to the extremely high installation and maintenance costs, many intersections with moderate and low traffic were left unmonitored, which is why there is a need for an alternative solution.

To address these challenges, the Vietnamese government has implemented traffic surveillance camera system around the country [3]. However, these projects are typically extremely costly and only limited to areas with heavy traffic, such as city centers and highways. For instance, in Ba Ria – Vung Tau Province, a traffic surveillance camera system comprising 89 cameras required approximately 3.5 million USD, averaging around 40,000 USD per camera. [4]. Due to the extremely high installation and maintenance costs, many intersections with moderate and low traffic were left unmonitored, which is why there is a need for an alternative solution.

By leveraging object detection models, image processing techniques, and gathering CCTV footage from cross intersections, authorities can establish a system capable of identifying red light runners without relying on costly infrastructure surveillance projects. This approach

enables the utilization of existing camera networks to monitor traffic violations, enhancing road safety measures. Such a system, designed for cost-efficiency and wider coverage, facilitates prompt intervention and enforcement against red light violations. Additionally, the integration of these technologies allows for the exploration of further enhancements in traffic management and accident prevention strategies, fostering a more comprehensive approach to ensuring safer roadways at every corner of the country.

## 1.2. Problem Statement

The limitations of existing traffic surveillance systems in Vietnam present a critical problem, particularly regarding the ineffective monitoring of moderate and low traffic intersections due to high costs. The current approach, relying on costly traffic surveillance camera systems concentrated in areas with heavy traffic flow, such as city centers and highways, leaves numerous intersections unmonitored.

The limitations of existing traffic surveillance systems in Vietnam present a critical problem, particularly regarding the ineffective monitoring of moderate and low traffic intersections due to high costs. The current approach, relying on costly traffic surveillance camera systems concentrated in areas with heavy traffic flow, such as city centers and highways, leaves numerous intersections unmonitored.

To address this issue, emerging technologies in computer vision and machine learning provide promising methods for automated object monitoring and recognition. One such technology is Convolutional neural networks (CNN), which has been shown to be more and more capable of detecting complex scenarios such as identifying intricate patterns and diverse objects within images. CNN's capability to learn and discern hierarchical features enables it to comprehend visual data hierarchically, from simple to complex elements.

This approach allows CNN to detect and classify objects in images, irrespective of variations in size, orientation, or context. Its ability to adapt has proven particularly effective in discerning complex scenarios, such as distinguishing between different vehicle types, pedestrians, and traffic signals within urban environments. Moreover, CNN's application extends to addressing challenges like red light running detection at intersections. By analyzing visual cues captured by surveillance cameras, CNN can swiftly identify instances of vehicles violating traffic signals, aiding in the automatic detection of red-light runners. Its ability to process and interpret visual information allows for the recognition of these violations in real-time, contributing to enhanced traffic management and road safety.

Among all CNN architectures, the YOLO (You Only Look Once) framework stands out as a significant technology in real-time object detection, for its impressive ability to identify objects swiftly and consistently with remarkable speed and accuracy [5]. The YOLOv8 detection model, the latest addition to the YOLO object detection family, while being a very light weight model, it possesses efficient real-time object detection capabilities better than its previous predecessors [6] and can quickly identifying and categorizing objects within images and videos. This proficiency makes it an ideal choice for developing an advanced RLR detection (RRD) system. Unlike extensive IoT traffic surveillance camera systems relying on Onboard Image Processing (OIP), Light Detection and Ranging (LDR), and sensors within cameras for data transmission, this RRD system operates as software performing calculations and image processing on personal computers rather than within the cameras themselves. By leveraging existing CCTV networks, this RRD system can be implemented without extensive hardware upgrades, utilizing the infrastructure already in place, significantly reducing costs and complexity.

## 1.3. Scope and Objectives

The scope of this thesis involves developing a system capable of detecting red light running violations using CCTV camera footage as input. The system aims to detect and classify various vehicle types and traffic lights, while also tracking instances where a vehicle commits a red-light violation in real-time. Subsequently, it will record these violations, which can then be stored in either CSV or PDF formats. Additionally, this thesis will delve into evaluating the custom trained YOLOv8 model, calculating crucial metrics like mean average precision (mAP), precision, recall, and object class losses. It will also showcase the model's performance in simulated heavy weather conditions to assess its robustness.

The primary objective of thesis is to conduct research and develop a real-time system utilizing CCTV footage to detect and record instances of red-light violations while evaluating the performance of a custom trained YOLOv8 model under varying conditions. To achieve this goal, several smaller objectives need to be addressed. These objectives include tasks such as:

Conducting an extensive literature review of existing research papers on topics related to RLR detection, while exploring the various techniques used in each study. This task aims to gain a better understanding of the prior research regarding system implementation, effectiveness, and comparative benefits in conjunction with this thesis.

Collecting frames from CCTV footage to compile a dataset for training the object detection model. The dataset will consist of a blend of regular images and modified images that simulate

specific situations like poor visibility and severe weather conditions. By creating mixed dataset, it enhances the model's robustness by exposing it to diverse situations, preventing overfitting and enabling better generalization to real-world scenarios.

Once the dataset is compiled, the subsequent step involves manual annotation of these images, an essential preparation phase for training the object detection model. Annotation refers to the process of labeling or marking specific objects within images, outlining their boundaries, and categorizing them according to predefined classes. This annotation process aids the model in learning and recognizing objects accurately during training. Additionally, annotations facilitate the model's understanding of various object classes, such as vehicles and traffic lights, enabling it to differentiate between them when deployed for RLR detection.

When the dataset is fully annotated, the training process for a custom YOLOv8 model begins. This phase involves feeding the annotated dataset into the YOLOv8 pre-trained model prepared by Ultralytics [6], allowing it to learn and discern patterns, features, and object classes from the labeled images. Training involves iteratively adjusting the model's parameters to minimize errors and enhance its ability to accurately detect objects, including vehicles and traffic signals, especially during instances of red-light violations. This iterative process aims to optimize the model's performance, enabling it to recognize and classify objects effectively in various scenarios encountered within traffic surveillance footage.

Following the training phase, the model requires evaluation to assess its performance accurately. This evaluation process involves computing essential metrics such as mean average precision (mAP), recall, and precision. Evaluating these metrics provides a comprehensive understanding of the model's strengths and weaknesses, guiding potential fine tuning to enhance its efficiency in real-time RLR detection scenarios to prepare it for testing.

During the testing phase, the trained object detection model undergoes rigorous assessments to validate its performance and functionality. This phase involves subjecting the model to various real-world scenarios resembling the conditions encountered in traffic surveillance. The model is evaluated using distinct datasets or footage that wasn't part of the training set, ensuring its ability to generalize and detect RLR instances accurately in new, unseen environments. Testing involves analyzing the model's response and accuracy in identifying vehicles, traffic lights, and instances of red-light violations in diverse scenarios, including different weather conditions, varying visibility, and traffic densities.

After obtaining a suitable custom-trained model, the next phase involves initiating the image processing stage. Through the implementation of tracking algorithms, the system gains the capability to identify instances of RLR violations in real-time. This involves the systematic

detection and marking of such violations within the processed images. The utilization of tracking algorithms enhances the system's ability to precisely identify and track objects by giving them an ID number, allowing for the efficient and accurate identification of RLR instances. This stage is crucial in ensuring the system's ability to recognize and record violations.

Once tracking and image processing features are implemented, the construction of the RLR system can begin. As this system is intended to be user-friendly software, it necessitates the design of a graphical user interface (GUI) to streamline interactions between users and the system. The GUI plays a crucial part in simplifying user engagement, allowing for intuitive and straightforward interactions with the system. The GUI should have displayed the detected violation, provided options to export this recorded data to the computer's file system.

## 1.4. Assumption and Solution

The success of this thesis depends on a number of important assumptions. First of all, it is assumed that the study's use of publicly available camera footage is always accessible and of a high quality. This assumption is based on the belief that the collected footage is clear and reliable, despite the fact that variations in lighting, weather, and camera angles may impact the accuracy of object detection algorithms.

Secondly, the study assumes a level of consistency in the flow of traffic and conditions captured from public camera recordings. It assumes that observed traffic behavior, flow, and adherence to traffic signals, including red lights, remain relatively stable over time, allowing for effective violation analysis and detection. Furthermore, it presumes that any fluctuations or deviations from these patterns do not significantly hinder the detection and analysis of red-light violations using object detection methodologies.

Another important assumption is the existence of an infrastructure that is capable of managing and gathering large amounts of video data from public cameras. This includes having enough computational capacity, reliable network connectivity, and advanced data processing capabilities to effectively handle and collect the large video streams in order to prepare them for analysis.

Addressing these assumptions requires practical solutions. To begin, the thesis suggests that preprocessing techniques such as image enhancement and noise reduction be used to improve data quality. Furthermore, diversifying the dataset with different scenarios aims to help the object detection model adapt better to changing conditions.

Furthermore, making sure that the object detection model can adapt to small changes in how traffic flows is important. It means improving the model so it can learn from these changes and adjust itself. This way, even the smallest differences from how traffic usually is will get noticed and checked more carefully. Also, regularly updating the model with new information about different traffic scenarios to help make the system find red-light violations more accurately. By doing so, the model stays sharp and reliable, even if traffic patterns change a bit over time.

Moreover, addressing the assumption of robust infrastructure involves advocating for upgrades in computational resources, network stability, and data processing methods. Investing in powerful hardware, optimized network configurations, and streamlined data processing techniques would improve the system's ability to manage and collect large amounts of video data for comprehensive analysis

## 1.5. Structure of thesis

The literature review, methodology, implementation and results, discussion and evaluation, conclusion and future work are the five main sections which together make up this thesis.

The literature review section will focus on previous studies, academic publications, and published studies that are relevant to the topic. The goal of this section is to provide a deeper understanding of prior studies, focusing on their implementation, associated technologies, and effectiveness in comparison to the research done in this thesis.

The methodology section will present the method taken in creating the RLR detection system. It describes in detail the procedures of collecting datasets, and annotating them, selecting deep learning models, preprocessing data, training the proposed model, and evaluation methodologies, along with the metrics that were selected for performance evaluation. Furthermore, this section will provide definitions for terms that are relevant to the thesis's research.

The implementation and results section will cover the practical implementation of the system developed based on the methodology described. It includes the implementation of deep learning algorithms for object detection using CCTV footage, the model deployment process, and the outcomes that were obtained. It will showcase the system's performance in detecting red-light violations, providing statistical data, and visual representations of how the system performed.

The discussion and evaluation segment will objectively evaluate the outcomes achieved, exploring their significance, limitations, and implications. It will evaluate the strengths and weaknesses of the developed system, discussing the alignment of findings with the initial

research objectives. This section will also explore possible improvements or future directions for refinement.

The conclusion will summarize the key findings, emphasizing the contributions made by the developed system in RLR detection using deep learning and CCTV footage. It will go over the significance of the research, summarize the main outcomes, and propose methods for future research or system improvements.

# CHAPTER 2

# LITURATURE REVIEW/RELATED WORK

## 2.1. Detecting Red-Light Runners (RLR) and Speeding Violation through Video Capture

The conclusion will summarize the key findings, emphasizing the contributions made by the developed system in RLR detection using deep learning and CCTV footage. It will go over the significance of the research, summarize the main outcomes, and propose methods for future research or system improvements.

In Goma, Bautista, Eviota, & Lopena (2020) [7], the research proposes a deep neural network method, the Single Shot Detector (SSD), for vehicle detection in video clips, addressing the growing concern of traffic violations leading to road accidents. This system aims to increase the effectiveness of law enforcement by quickly identifying violations. In addition to identifying typical violations like speeding and running a red light, it also reduces the burden of police officers by automating detection, particularly in situations where human error may occur.

The method used in the research involves acquiring video data from a specific intersection and testing three distinct Convolutional Neural Network (CNN) architectures, MobileNet, Inception V2, and ResNet 50, through the SSD model to detect traffic violations accurately. In particular, the prototype achieved high accuracy, with up to 100% accuracy for RLR and 92.1% accuracy for speeding under optimal conditions.

The study also elaborates on data acquisition, processing, feature extraction, bounding boxes generation, and multi-scale feature mapping within the SSD model. It discusses non-maxima suppression for refining detections and delves into the specific methodologies for red-light runner and over-speeding detection, detailing the computational processes and formulas used for speed prediction and distance measurement.

The discoveries are outlined in the results and discussion sections, which also highlight the performance of various feature extractors and their accuracy rates, processing speeds, and error percentages. MobileNet showed the fastest processing speed (0.08s per frame) but with a higher error percentage, while ResNet50 exhibited slower processing (0.15s per frame) yet achieved higher accuracy.

All things considered, this study offers a thorough analysis of automated traffic infraction detection, highlighting the importance of selecting feature extractors according to user preferences for speed or accuracy and outlining the future plans for the field's development.

## 2.2. Vehicle detection and Attribute based search of vehicles in video surveillance system

The research by Momin, & Mujawar (2015) [8] proposed an approach to vehicle detection and attribute-based search in video surveillance systems, acknowledging the limitations of traditional methods in urban scenarios with varied lighting and crowded scenes. It presents a co-training method for detection using Haar features, utilizing an Adaboost-based classifier, and subsequently extracts semantic attributes for vehicle retrieval.

The study highlights the difficulties in identifying automobiles in various scenarios and emphasizes the significance of attribute-based searches in criminal investigations. It detects motion using Haar-like features and adjusts the aspect ratio sliding window to accommodate different kinds of vehicles.

The detection process involves multiple steps: data collection from surveillance cameras, identification of regions of interest (ROIs), foreground blob extraction using background subtraction, and subsequent analysis of vehicle shapes. The paper highlights the utilization of motion and Haar features for vehicle detection, combined to minimize false alarms. Attributes critical for identifying vehicles such as date and time, direction of travel, color, and speed are precisely extracted for each detected vehicle. This method utilizes optical flow computation to verify vehicle direction, quantization of colors in HSL space to estimate dominant vehicle colors, and centroid motion for speed calculation.

Results from experiments show the algorithm's effectiveness in detecting vehicles, with motion and Haar-based detection showing a 5% reduction in false alarms compared to Haar-only detection. The study showcases detection outputs from various scenarios, revealing robustness to different lighting conditions and crowded scenes, while acknowledging room for improvement, particularly in color estimation under varying lighting conditions.

The paper concludes by highlighting the proposed method's potential for vehicle detection and attribute-based search in video surveillance systems. It suggests further enhancements for accuracy, such as incorporating additional features like color and speed, to improve the system's overall performance.

## 2.3 Vehicle detection and Attribute based search of vehicles in video surveillance system

The paper by Yung, & Lai (2001) [9] presents a creative approach for detecting red light runners on video, diverging significantly from conventional red light camera systems. This method extracts traffic light states and vehicle movements solely from video data, in contrast to existing systems that depend on physical or electronic connections to traffic light control systems or buried loop detectors.

The methodology involves two key stages:

Traffic Light Sequence Construction: This step involves detecting and constructing a sequence of traffic lights present at a junction. It identifies red, yellow, and green regions by color-based detection in the video frames, utilizing the spatial and temporal relationships between these lights. The algorithm locates traffic lights based on their colors, shapes, sizes, and relative positions in consecutive frames. Once identified, the algorithm establishes the traffic light sequence based on these detections.

Vehicle Motion Estimation: After identifying the traffic light sequence, the method estimates the motion of vehicles beyond the stop line while the traffic light is red. This is achieved by creating virtual loop detectors (VLD) that mimic the function of physical loop detectors but exist solely in the video space. The VLDs are placed beyond the stop line, and a motion estimation algorithm is applied to detect movements in the direction of the road, filtering out pseudomotions caused by shadows, pedestrians, or turning vehicles.

This system's implementation and evaluation involved various field trials, demonstrating its capability to detect multiple red-light runners across multiple lanes. Moreover, it showcased resilience against realistic challenges such as minimal traffic light visibility, shadows causing pseudo-motions, low contrast scenarios, pedestrian movements, and turning vehicles. The conventional approach to red light enforcement relies on systems interconnected with traffic light control and loop detectors, typically triggering a camera to capture red light violations. This paper challenges this methodology, proposing a video-based analysis leveraging spatial and temporal cues from the footage.

Through various trials and evaluations, this approach demonstrated high accuracy in identifying red light violations despite adverse conditions such as poor visibility, shadows, and diverse vehicle and pedestrian movements.

In summary, this paper presents a groundbreaking method that capitalizes on video analysis to detect red light violations, showcasing resilience against real-world challenges and demonstrating promising potential for enhancing road safety. This innovative approach

signifies a shift from traditional red light camera systems, relying solely on video analysis to detect violations without direct connectivity to traffic control systems. The method's success in multiple real-world scenarios shows its potential for enhancing road safety by autonomously identifying red light runners, potentially reducing the frequency of accidents caused by such violations

## 2.4 An Algorithm for Identifying Red Light Runners from Radar Trajectory Data

In Zaheri, & Abbas (2015) [10], the research paper addresses a critical issue in traffic safety: the dilemma zone and red light running at signalized intersections. This analysis revolves around a study proposing a new algorithm to predict red light runners and differentiate them from other right turners at red lights. The paper's main goal is to enhance intersection safety by accurately identifying red light runners, thus reducing accidents, and enhancing the flow of traffic.

The research approach uses a thorough data collection procedure with the second-generation intersection safety data collection and evaluation system from the Virginia Tech Signal Control and Operations Research and Education System (VT-SCORES) lab in order to accomplish this goal. Vehicle trajectories, speeds, and other pertinent parameters were continuously measured in real time in the field by utilizing Wavetronix radar technology installed at a particular intersection.

The analysis and results section presents findings based on one month of data. The paper illustrates a clear correlation between red light running occurrences and specific times of the day, such as peak traffic hours and weekends. The comparison of radar-based measurement with loop detectors is crucial in highlighting the model's effectiveness. The validation process demonstrates a high accuracy rate of 96%, emphasizing the model's reliability in identifying red light runners.

In terms of implications and future works, the paper stresses the significance of the proposed model in traffic safety and suggests potential future research directions. It mentions the model's applicability to emergency vehicles, trucks, and buses, hinting at its scalability and adaptability.

The study excels in addressing a critical issue in transportation safety and proposes a viable algorithm to identify red light runners. The methodology is robust, employing both theoretical models and practical field data. However, the paper could benefit from more comprehensive insights into the limitations of the proposed model and potential challenges in its real-world implementation.

Overall, the paper represents a significant advancement in traffic safety research, offering a promising solution to mitigate red light running incidents at intersections. Its high accuracy rate and systematic approach make it a valuable contribution to the field.

## 2.5 The Research of Red Light Runners Video Detection Based on Analysis of Tracks of Vehicles

In Luo, Huang, and Qin (2008) [11], the research paper delves into the critical domain of red-light runner detection in computer vision applications. It introduces a new methodology centered around analyzing vehicle tracks to address the limitations of existing detection algorithms. The proposed approach aims to rectify undetected instances and error-prone identifications common in current existing algorithms.

The paper first highlights the significant threat to road safety posed by drivers disregarding traffic signals, especially at intersections. It highlights the prevalence of accidents due to red light violations, underscoring the importance of efficient red light running detection systems. While traditional systems rely on inductive loop detectors (ILD), they suffer from drawbacks such as surface destruction, rigidity, and high costs, prompting the need for alternative methods.

The paper proposes leveraging video technology and analysis techniques to redefine the principles of red-light camera (RLC) systems. It advocates for a shift from ILD-based RLCs to those employing virtual loop detectors (VLD) in video frames. However, it acknowledges the shortcomings of VLD-based systems, particularly in cases of undetected or mistakenly detected violations, especially with right-turning vehicles. The methodology proposed in the paper primarily revolves around the analysis of vehicle tracks. It involves multiple steps, including stop line detection using the Hough transform, vehicle motion estimation, and the fitting of moving points using cubic splines interpolation. These techniques aim to accurately identify vehicle movements and track violations concerning red lights.

The system overview section provides a functional block diagram depicting the traffic light detection, stop line detection, and subsequent vehicle motion estimation and tracking. Each section details the algorithms and methods utilized, emphasizing the sequential nature of the proposed system. Stop line detection involves employing the Hough transform to identify the stop line's location, crucial for defining the field of detection and determining red light violations. The paper discusses the nuances of detecting the stop line and its relevance to subsequent analysis. Moving points fitting, essential before analyzing vehicle tracks, is conducted using cubic splines interpolation. The algorithm for fitting moving points is explained in detail, highlighting the mapping of points in the X-Y space, and ensuring

continuity in the Y-axis for accurate analysis. The red-light runners detection algorithm employs an analysis of vehicle tracks to discern violations. Three situations straight, left, and right turns are distinguished, with specific criteria set for identifying red light violations based on the sequence of moving points in the vehicle tracks.

Trials and results stage present a prototype implementation showcasing the effectiveness of the proposed methodology. Results demonstrate successful vehicle motion estimation, tracking, and the accurate detection of red-light runners, validating the efficacy of the methodology.

In conclusion, the paper introduces a comprehensive methodology leveraging vehicle track analysis for red light runner detection. While the proposed approach demonstrates promise and accuracy in trials, it also emphasizes the need for further validation and refinement in real-world applications.

## 2.6 Bayesian network for red-light-running prediction at signalized intersections.

In the paper from Chen, Zhou, and Li (2019) [12], it explores a critical aspect of traffic management, predicting stop-or-go behavior at signalized intersections, focusing particularly on the detection and anticipation of RLR (RLR) vehicles. By utilizing Bayesian Network (BN) models and analyzing continuous trajectory data collected by radar sensors, the study aims to enhance the accuracy of forecasting a vehicle's decision when encountering a yellow traffic signal.

In setting the stage for the research, the researchers utilize radar sensor data obtained from a T intersection. This dataset includes vehicle information like trajectories, speed, acceleration, and distances from the stop line. Such comprehensive data forms the backbone for constructing a robust stop-or-go prediction model. The study design encompasses a comparative analysis involving BN models, Linear Regression (LR), Support Vector Machines (SVM), Random Forests (RF), and the contrast between continuous trajectories and Inductive Loop Detector (ILD)-based prediction methodologies.

Preprocessing this extensive dataset involves extracting pertinent attributes from 17-day historical radar data. The focus is on identifying "interested" vehicles—those within a critical distance from the stop line during a yellow signal onset. These vehicles are categorized into distinct groups based on their position in traffic—alone, leading a platoon, following a platoon, and considering the leader's decision. Crucial attributes, such as distance to the stop line, speed, acceleration, and car-following behavior, are identified as significant determinants for accurate stop-or-go predictions.

The main point of the research lies in the development and evaluation of Bayesian Network models. These models are strategically designed to categorize a vehicle's stop-or-go behavior based on attributes like speed, acceleration, and distance precisely at the yellow onset time. Additionally, the models account for car-following behaviors, a factor known to impact decision-making at signalized intersections. Various BN structures are experimented with, comparing their performance against SVM, RF, and ILD-based methods. The distinct advantage of BN models lies in their provision of probabilistic outputs, offering enhanced interpretability and decision-making support.

The observational findings obtained from the study showcase the clear advantage of BN models utilizing radar data over ILD-based prediction methods in forecasting stop-or-go behavior at signalized intersections. Particularly noteworthy is the higher detection rate achieved by BN models when maintaining a low false alarm rate. This aspect holds significance in potentially minimizing accidents related to RLR vehicles. The study emphasizes the superiority of continuous trajectory data obtained from radar sensors over ILD data for achieving greater prediction accuracy.

In terms of future development, the researchers chart a path for future research initiatives. This includes broadening the scope by incorporating additional attributes such as lane-changing behavior to enrich prediction models. Moreover, the study advocates for leveraging advanced learning techniques to establish relationships between attribute nodes and harnessing evolving connected vehicle technologies for more precise data collection. A proposed online learning system to dynamically update the RLR vehicle training set could further enhance prediction accuracy over time.

In conclusion, this research contributes invaluable insights into the optimization of traffic management and safety at signalized intersections. By capitalizing on continuous trajectory data and employing BN models, the study showcases the potential for significantly improving the accuracy of predicting stop-or-go behavior, ultimately fostering safer roadways and efficient traffic flow.

## 2.7 Connected vehicle-based RLR prediction for adaptive signalized intersections.

In the search of Li, Chen, Lin, Xu, and Wang (2018) [13], the research focuses on developing a system for predicting and classifying Red Light Running (RLR) behavior at signalized intersections using connected vehicle (CV) data. The primary objective of this research is to develop and compare two distinct methodologies for RLR prediction: one leveraging

continuous data collected from radar sensors (CV-based) and the other relying on fixed-sensor data (ILD-based). The goal is to determine which method is more accurate at detecting RLR behavior, which is a vital component in averting potentially dangerous intersection collisions.

The CV-based method involves extracting several attributes from vehicle trajectories precisely at the onset of a red light. These attributes include speed, acceleration, and the distance of vehicles from the stop line. This data serves as the basis for classifying vehicles into RLR and non-RLR categories. One significant consideration in this classification process is the selection of the time difference between a vehicle's arrival at the stop line and the red-light onset time, crucial in determining the likelihood of RLR behavior. The research highlights the importance of calibrating this parameter for optimal prediction accuracy, striking a balance between RLR and non-RLR classification.

To refine the dataset for analysis, the study filters out instances of yellow-light-running vehicles. The resultant dataset, consisting of 9,329 trajectories, is then used for training and validation. Utilizing this dataset, the CV-based method, using radar data, demonstrates its superiority by effectively distinguishing 628 RLR vehicles from 8,701 non-RLR vehicles based on continuous measurements.

In the evaluation section, the study adopts several metrics like false alarm rate, missing rate, and receiver operating characteristic (ROC) curves. These metrics provide a comprehensive assessment of the models' predictive capabilities. ROC curves, for instance, depict the comparative performance between the LS-SVM (Least Squares Support Vector Machine) models applied to CV data and ILD-based methods. The evaluation reveals that LS-SVM models, specifically when applied to CV data, exhibit superior predictive accuracy over ILD-based approaches. The research also introduces the concept of weighted LS-SVM classifiers, aiming to minimize missing rates while considering the significance of each class (RLR and non-RLR). This exploration into weighted models reflects the study's commitment to enhancing prediction accuracy by adjusting the classifier's sensitivity to each class.

Furthermore, the study examines the influence of time difference on prediction accuracy. Through varied scenarios and time differences, the research consistently demonstrates the CV-based approach's superiority over fixed-sensor-based methodologies. This investigation substantiates the CV-based method's advantages, showcasing lower false alarm rates and consistently higher prediction accuracy across different settings. However, the study acknowledged challenges in predicting some peculiar RLR behaviors, like vehicles initially stopping at red lights and then proceeding, or vehicles crossing the stop line well after the red-

light onset. These uncommon behaviors posed difficulties in their prediction due to their infrequency in the dataset.

About future advancements, the research proposes incorporating additional behavioral aspects like lane-changing and car-following behaviors. Moreover, implementing an online learning framework and integrating historical data with real-time information are suggested strategies to enhance prediction accuracy further.

In summary, this research presents a comprehensive exploration into the prediction of RLR behavior at signalized intersections. Leveraging CV-based methodologies offers a promising avenue for significantly improving accuracy in identifying RLR incidents, thus contributing to enhanced intersection safety. The findings shed light on the intricacies of RLR prediction and pave the way for future advancements leveraging connected vehicle technologies to mitigate traffic risks at intersections.

## 2.8 Automatic detection of red light running using vehicular cameras.

The research by RH Brasil, and Machado in 2017 [14] presented a novel approach to tackle red light violations using vehicular cameras and onboard processing. The primary goal was to detect instances of vehicles running red lights, especially in areas lacking fixed sensors or monitoring systems. This proposed system aimed to utilize onboard cameras and processing algorithms within vehicles to identify red light violations, functioning as an educational tool rather than for punitive measures.

The study outlined a multi-stage process involving image capture, preprocessing, segmentation based on color spaces (specifically HSV), and detection/recognition techniques for identifying traffic lights. The system focused on detecting red lights, filtering out yellow and green, and employing algorithms to recognize and track the traffic signal within captured images.

In order to test the system's accuracy and processing efficiency, recorded videos from various locations and scenarios were used in the testing. Particularly, the results demonstrated good accuracy rates when it came to correctly identifying red lights during daytime recordings. However, challenges emerged with false positives and negatives, particularly influenced by the positioning of the camera and the quality of recorded videos. In terms of real-time processing, the study demonstrated potential feasibility but also highlighted the need for further improvements in system parameters, camera quality, and processing algorithms. Challenges related to nighttime recordings, specific traffic signal configurations (such as arrows limiting vehicle movement), and the impact of varying lighting conditions on accuracy were identified.

The research proposed potential future directions, including testing the system on low-power systems like Raspberry Pi, exploring higher-quality camera recordings to reduce color distortion, addressing nighttime detection challenges, and evaluating parallel processing for improved efficiency.

Overall, the findings suggest that while the system showed promise in detecting red light violations using vehicular cameras, refinements and further testing are necessary to enhance accuracy, especially in varying environmental conditions and traffic signal configurations. Moreover, real-time implementation and cost-effective deployment in vehicles remain areas for future exploration and optimization.

## 2.9 A Discrete Tracking Based-on Region For Red-Light Running Detection.

In Budiardjo, and Ramli (2013) [15], the paper addresses the critical issue of RLR at intersections, a pervasive safety concern worldwide. Focusing on Indonesia, the study delves into the complexities of RLR, emphasizing its danger and the need for an intelligent infrastructure system to mitigate risks effectively. By implementing tracking algorithms based on RFID technology, the research goal is to predict and detect RLR incidents by measuring vehicle speed, distance from the intersection, and acceleration.

The study's model and simulation techniques use RFID readers strategically placed around intersections, capturing vehicle IDs, timestamps, and locations to predict RLR likelihood. Utilizing data processing, the research defines a "dilemma zone" based on kinematic equations, evaluating vehicle positions, speeds, and reaction times during yellow light phases. The developed algorithms can anticipate RLR during yellow phases and detect violations during red phases, providing insights into individual and clustered vehicle behaviors, distances from intersections, and speeds at critical points. The model, implemented successfully via Scilab software, offers a foundation for comprehensive RLR prediction and detection systems based on RFID technology.

Regarding the methodology, the study presents a discrete tracking approach based on RFID reader networks for the purpose of predicting and identifying RLR at intersections. It identifies the "dilemma zone," where vehicles struggle to stop or clear the intersection during yellow phases. By harnessing RFID data and kinematic equations, the study models vehicle behaviors, speeds, and distances, crucial in predicting RLR incidents. The algorithms developed demonstrate the potential to anticipate RLR during yellow signals and detect violations during red phases. The research offers insights into individual vehicle behaviors and cluster tendencies, shedding light on acceleration, distances from intersections, and speeds at critical moments.

In summary, this research pioneers an innovative strategy employing discrete tracking algorithms based on RFID technology to address the pervasive issue of RLR at intersections. By utilizing RFID reader networks, the study models and predicts RLR incidents by analyzing vehicle speeds, distances, and accelerations during crucial signal phases. The defined "dilemma zone" helps evaluate the potential for RLR violations, and the developed algorithms demonstrate promising capabilities in predicting RLR during yellow signals and detecting violations during red phases. Implemented through Scilab software, this pioneering approach lays a robust foundation for creating comprehensive RLR prediction and detection systems using RFID technology.

## 2.10 Innovative countermeasures for red light running prevention at signalized intersections: A driving simulator study.

In Hussain, Alhajyaseen, Brijs, Pirdavani, and Brijs (2020) [16], the research focuses on addressing the critical issue of RLR (RLR) in traffic intersections, a pervasive safety concern both in Indonesia and globally. The study aims to model and simulate vehicle speed through intersections to predict potential RLR instances, utilizing discrete tracking algorithms based on RFID (Radio Frequency Identification) technology.

The introduction outlines the consequence of RLR incidents and their impact on traffic safety, citing statistical data on fatalities and injuries caused by intersection-related crashes. It also references existing research attempting to mitigate RLR through measures like dynamic signal adjustments and collision avoidance systems.

The paper introduces a discrete tracking algorithm based on RFID technology, explaining its functionality in tracking vehicles' movements within a reader network. This algorithm aims to predict RLR by evaluating the dilemma zone, where vehicles may struggle to safely stop or clear the intersection during the yellow phase. The study proceeds with an intersection model and simulation setup using RFID reader networks to detect and track vehicles, focusing on RLR prediction and detection algorithms. The proposed algorithms analyze vehicle speed, entry times during yellow and red phases, and distances from the intersection to predict and detect potential RLR instances.

The research methodology involves simulating vehicle behaviors, individual and clustered, to understand their speeds, distances, and likelihood of RLR. It presents data formats for timestamp databases and collected vehicle speed databases, emphasizing the system's ability to determine average speeds and track clusters of vehicles.

The evaluation process involves determining vehicle distances from the intersection at yellow onset, calculating speeds at yellow fire, assessing accelerations, and predicting RLR instances based on these parameters. The paper discusses the simulation's ability to predict RLR for individual vehicles and clusters, emphasizing the system's dependency on the leading vehicle's behavior.

Overall, the paper introduces an innovative approach using RFID-based discrete tracking algorithms to model and predict RLR at intersections. The simulation methodology effectively analyzes vehicle behaviors, demonstrating the system's potential to enhance traffic safety by identifying potential RLR instances for intervention and prevention.

## 2.11 RLR traffic violations: A novel time-based method for determining a fine structure.

In Ghorghi, Zhou, and Zech (2016) [17], the paper conducted a detailed analysis within the framework of the City of Opelika's Red Light Camera (RLC) program to establish a method for determining fines related to RLR (RLR) violations. The study utilized VISSIM traffic models to calculate the likelihood of crashes following the red signal onset at intersections, emphasizing the safety during the all-red phase and the escalated risk thereafter. Integral to the research was the incorporation of data from the National Highway Traffic Safety Administration (NHTSA) to evaluate crash costs and the utilization of the Highway Capacity Software (HCS) to estimate the delays experienced by road users due to all-red intervals. These estimations were pivotal in formulating a fine structure considering crash probabilities and delay costs associated with RLR violations.

The study's highlight was the creation of a time-based fine structure that aligns with the risk posed by RLR violations and the delays incurred by road users. The fines were structured to reflect the level of risk drivers take during different phases of a red signal, ensuring a correlation between imposed fines and potential dangers associated with RLR violations. However, this comprehensive framework is not without its limitations. The study acknowledges the need for real-world violation data to validate the simulations and models used in the analysis. The absence of detailed crash and violation data poses a challenge, limiting the accuracy of the developed models. Nonetheless, the framework serves as a foundational step, providing a guideline for municipalities and jurisdictions to adopt similar methodologies tailored to their specific traffic conditions and local requirements.

In general, the research offers a structured methodology for devising fines that are objective and rooted in the risk assessment of RLR violations. This could be invaluable for policymakers

seeking to create effective traffic safety measures that discourage RLR violations by ensuring fines correspond with the potential risks and societal costs incurred. The spatial adaptability of the methodologies proposed here opens avenues for broader application in different geographical and traffic contexts. Further research and validation using real-world data are essential to refine and enhance the accuracy of these methodologies.

## 2.12 Adopting Machine Learning Methods to Predict Red light Running Violations

In A Jahangiri, Rakha, and Dingus (2015) [18], the research conducted by Arash Jahangiri, Hesham A. Rakha, and Thomas A. Dingus focused on predicting red light running (RLR) violations at signalized intersections using machine learning techniques. RLR violations contribute significantly to intersection-related crashes, necessitating proactive identification and intervention. Factors influencing driver behavior at intersections, such as vehicle speed, Time to Intersection (TTI), Distance to Intersection (DTI), age, and gender, were examined. However, obtaining driver-related factors like age and gender in practice posed challenges, prompting a focus on kinetic factors like speed and acceleration, which could be obtained through monitoring vehicle movement via cameras or onboard devices.

Their study aimed to develop prediction models using machine learning, specifically Support Vector Machine (SVM) and Random Forest (RF), using a dataset collected through video cameras. A critical aspect was defining a monitoring period corresponding to the onset of the yellow signal, considered the critical point influencing driver decisions regarding stopping or proceeding. This period, between certain DTI values, was crucial for capturing decision-making data.

To build these models, the researchers employed feature selection using the mRMR method, identifying the most relevant factors. They extracted 17 features and, through mRMR, narrowed it down to five essential features: DTI at yellow onset, mean and max velocity over the monitoring period, standard deviation of acceleration over the period, and Required Deceleration Parameter (RDP) at yellow onset.

The SVM and RF models achieved high prediction accuracies of 97.9% and 93.6%, respectively, in identifying RLR violations. The selection of monitoring periods and critical factors such as yellow onset proved instrumental in the accuracy of these models. Additionally, considerations were made for communication latency and prediction time, essential for real-time application, showing that the models could provide quick alerts to prevent potential crashes.

In summary, the research highlighted the significance of both the yellow signal onset and the defined monitoring period in predicting RLR violations. It emphasized the importance of specific kinetic factors in decision-making and showcased the effectiveness of machine learning methods, particularly SVM and RF, in accurately identifying potential violations. The models developed in this study hold promise for real-time application, providing timely alerts to drivers, potentially mitigating intersection-related crashes.

# CHAPTER 3

# METHODOLOGY

## 3.1. Some Important Concepts

### 3.1.1. Computer Vision

Computer vision is a major field within computer science and artificial intelligence that focuses on developing algorithms, methodologies, and systems allowing machines to interpret and understand visual data from the digital world, such as images or videos. It makes use of mathematical models, statistical methods, and machine learning methodologies, particularly convolutional neural networks (CNNs) and deep learning architectures to extract meaningful information, features, and patterns from visual inputs. Researchers work hard to enhance the precision, effectiveness, resilience, and adaptability of algorithms concerning these tasks. This is frequently achieved by capitalizing on extensive annotated datasets and investigating inventive network structures and training methods.

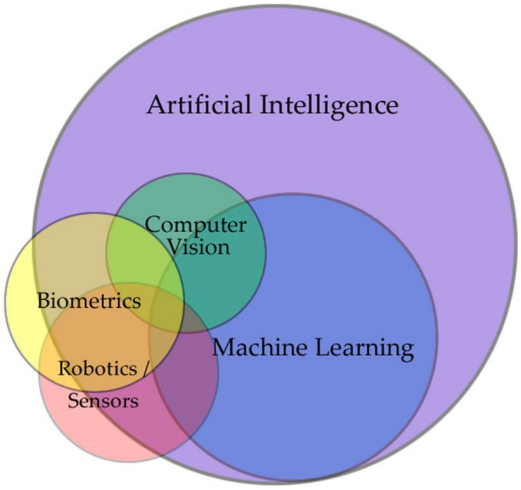


**Figure 1 – Computer vision in the field of AI. [19]**

The primary goal of computer vision research is to tackle basic problems in object recognition, scene analysis, visual perception, and image understanding. This includes tasks such as pose estimation, object detection, semantic segmentation, instance segmentation, image captioning, and image classification.

Furthermore, computer vision research frequently touches on related fields like robotics, computational photography, image processing, and human-computer interaction. The development of applications and systems with practical implications, covering fields like autonomous vehicles, healthcare diagnostics, surveillance, augmented reality, and more, is facilitated by collaborative efforts across these domains.

### 3.1.2. Deep Learning

Deep learning is a subfield of artificial intelligence that focuses on the design and training of complicated neural networks inspired by the human brain to process and comprehend vast amounts of data. Deep learning takes its name from the structural intricacy of these neural networks, which are made up of numerous layers and allow for the learning of detailed patterns and characteristics inside datasets. This hierarchical structure allows these networks to increasingly process and interpret information as it flows through the network layers, comparable to the human brain's information processing ability.

The concept of deep learning is defined by neural networks, which are made up of linked layers of artificial neurons or nodes. These networks are trained on large datasets, learning to recognize complex patterns and correlations in the data. During the training phase, huge amounts of labeled data are sent into the network, allowing it to repeatedly alter its internal parameters (weights and biases) to maximize performance and produce correct predictions or classifications. Nonlinearities are introduced by activation functions within neurons, assisting the network in representing complex data linkages. Deep neural networks may learn hierarchical data representations because of the architecture's depth, with lower layers catching simple information and deeper levels recognizing complicated, abstract patterns.

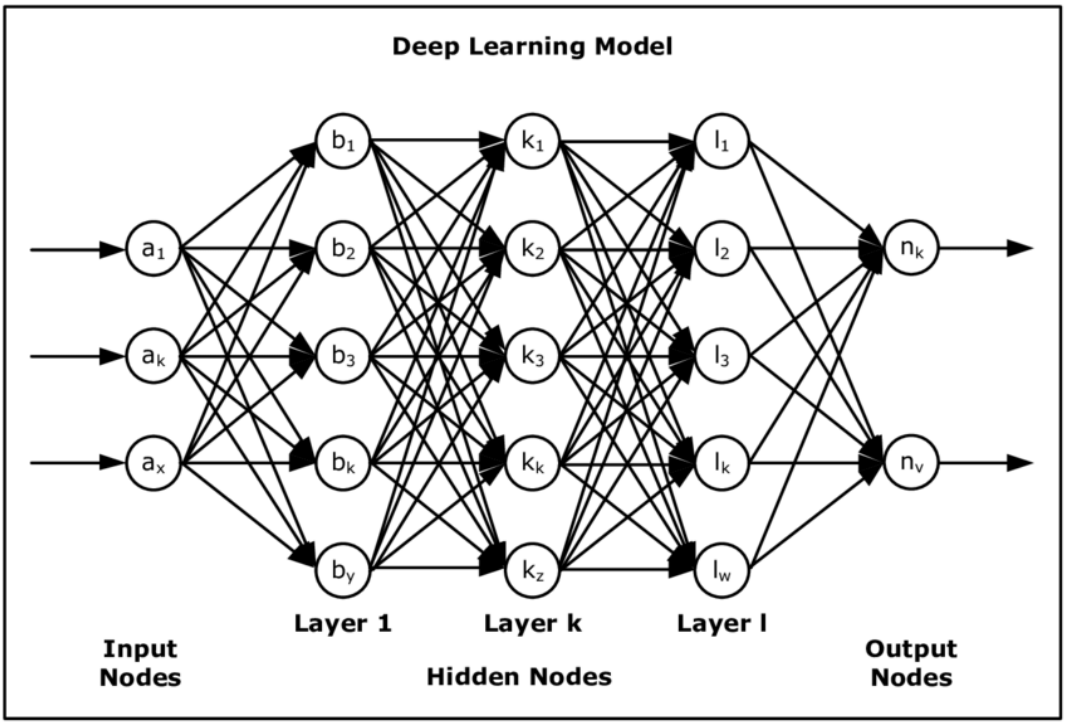


**Figure 2- Deep Learning Model. [20]**

Among the many domains, industries, and technologies for which deep learning algorithms are applied are image identification, natural language processing, and decision-making. These

algorithms are defined by their capacity to automatically learn complex data representations and extract fundamental characteristics. From speech recognition systems like virtual assistants and language translation applications to healthcare advancements in medical imaging for disease diagnosis and prognosis, the capability of deep learning is only limited to how machines understand and process information that is provided.

### 3.1.3. Object Detection and Object Detection Models

One of the most fundamental of computer vision is object detection, which enables machines to perceive and process visual information the same way that humans do. This complex process requires identifying and precisely localizing multiple objects within images or video frames. Its primary goal is to enable computational systems to not only recognize various objects in visual data but also to accurately determine their specific locations. This is accomplished by enclosing these objects in bounding boxes that define their spatial boundaries within the image and assigning categorical labels or tags to these identified objects, allowing for easier classification and categorization.

Object detection model training is an important process that requires large datasets annotated with bounding box coordinates and corresponding object labels. These models refine their understanding of object characteristics and spatial configurations through iterative learning, gradually improving their accuracy in localizing and classifying objects within images. Object detection has a wide range of applications in a variety of industries and domains. Object detection can be applied in a wide range of real-world applications, from autonomous vehicles, where it aids in the identification of pedestrians, vehicles, and road signs for safe navigation, to surveillance systems and medical imaging, where precise object identification is critical for security and diagnostic accuracy.

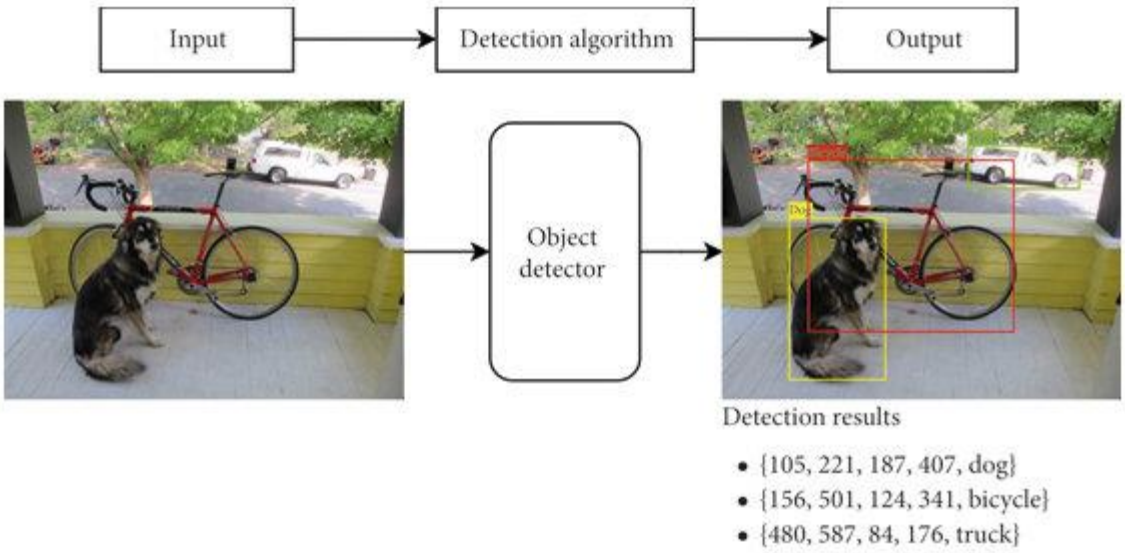


**Figure 3 – Object detection demonstration. [21]**

Advanced algorithms and sophisticated object detection architectures, such as Convolutional Neural Networks (CNNs), are at the core of object detection. CNNs are renowned for their remarkable ability to extract intricate and hierarchical features from images. These neural networks are critical in allowing machines to recognize patterns, textures, and shapes in visual data, facilitating object recognition and localization. Object detection techniques range from two-stage detectors like Faster R-CNN, which involve region proposal and classification, to one-stage detectors like YOLO (You Only Look Once), which are designed for faster real-time object detection by directly predicting bounding boxes and class probabilities.

### 3.1.4. Deep Neural Network

Deep neural networks, inspired by the structure and function of the human brain, represent a type of machine learning model. What distinguishes them is their architecture, composed of numerous layers of interconnected nodes (neurons). These neurons are categorized into various types of layers, such as input, hidden, and output layers, forming a hierarchical network that processes information step by step. The word "deep" refers to the integration of multiple hidden layers inside the network, enabling it to learn complex patterns and representations from the given data.

In practice, each neuron of the neural network takes input signals, analyzes them via weighted connections and activation functions, and sends the results to neurons in the next layer. During the training phase, the network changes the weights and biases associated with these connections to minimize the discrepancy between anticipated outputs and actual target values.

These networks learn by iteratively refining their internal representations, through a process known as backpropagation. During training, the network produces predictions, calculates the error or loss between these predictions and the true values, and then changes the weights and biases to minimize this error. Iterative learning lets the network identify complicated patterns and features in the data, allowing it to generate correct predictions or classifications.

The architecture of deep neural networks makes them able to automatically learn hierarchical data representations. Lower layers of these networks tend to capture simple details in images, such as edges or color gradients, but higher layers combine these features to recognize more abstract and high-level patterns in the data, such as objects or structures. This hierarchical learning capability is one of the reasons why deep neural networks excel at tasks involving complicated data, such as picture and speech recognition, natural language processing, and others.

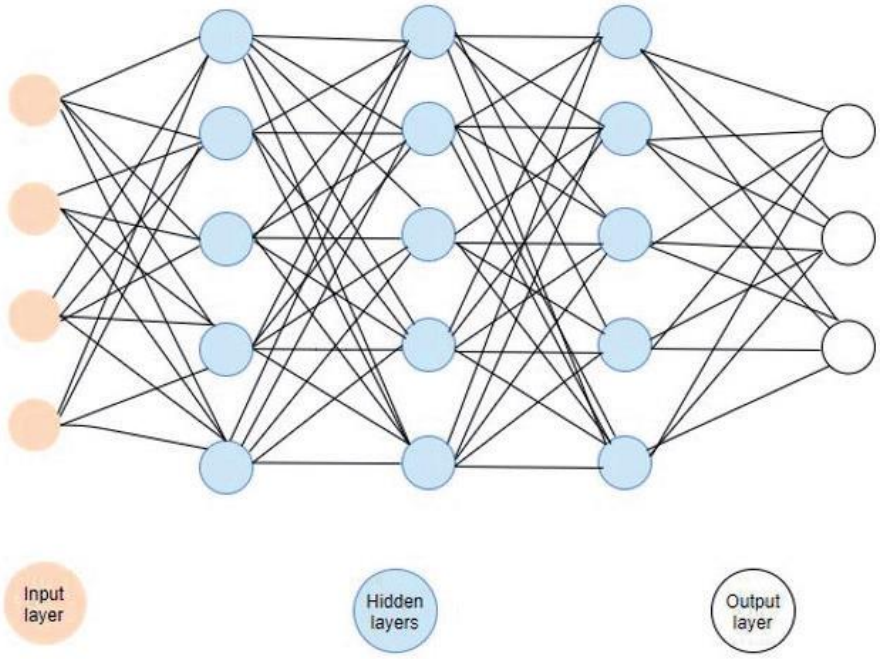


**Figure 4 – Deep neural network architecture. [22]**

There are different types of deep neural networks specializing in different domains and data types, each equipped with unique architectures to cater to specific tasks. Convolutional Neural Networks (CNNs) excel in processing visual data tasks like image classification and object detection by utilizing hierarchical feature extraction from images. Recurrent Neural Networks (RNNs) are mainly used for sequential data analysis, making them ideal for tasks involving time series prediction, natural language processing, and speech recognition. Generative Adversarial Networks (GANs) consist of two networks working synchronously to generate synthetic data, it is widely used in image generation and data augmentation.

### 3.1.5. Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) are a type of artificial neural network created primarily for analyzing and evaluating visual data, which made them extremely effective in computer vision applications. These networks are designed to emulate the complicated human visualization system, with a hierarchical architecture that allows them to distinguish patterns and characteristics inside images with incredible accuracy. Convolutional layers are at the heart of CNNs, where filters, or kernels, systematically glide across input images, extracting localized properties such as edges, textures, or shapes. This process, called convolution, generates feature maps that highlight where these features exist within the input data.

Furthermore, CNNs incorporate pooling layers after convolutional layers, lowering the spatial dimensions of the recovered features while keeping their crucial information. By extracting the most important information from feature maps, pooling techniques such as max pooling and average pooling improve computational efficiency and help prevent the networks from overfitting.

The biggest strength of CNNs lies in their ability to learn hierarchical representations of visual data. As the data progresses through the network's layers, from early convolutional layers to deeper ones, it gradually captures increasingly complex and abstract features. This hierarchical learning enables CNNs to discern intricate patterns, such as object parts and their arrangements, facilitating tasks like image recognition and object detection. CNN's architecture typically ends with fully connected layers that use the learned features for classification or regression tasks. These layers combine the high-level representations derived from previous layers, allowing the network to make predictions or decisions based on the input data. When the network is trained, this involves iteratively adjusting its internal parameters using optimization algorithms and backpropagation to minimize the difference between its predictions and the actual targets in the training data.

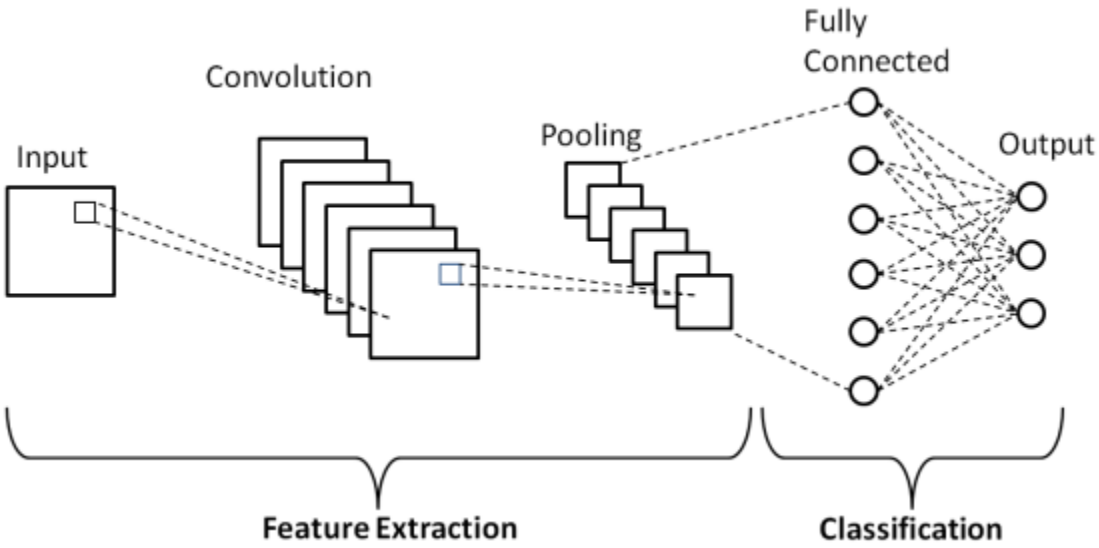


**Figure 5 – Different layers of a CNNs model. [23]**

CNNs have found widespread applications in various domains, revolutionizing fields like healthcare, autonomous vehicles, and security surveillance. Their ability to extract meaningful information from visual data has propelled advancements in tasks such as medical image analysis, object recognition in autonomous driving systems, and facial recognition in security applications. As research in the field progresses, CNNs continue to improve, with ongoing exploration into novel architectures, interpretability, and transfer learning, promising further advancements in their capabilities and applications.

### 3.1.6. Backpropagation

Backpropagation is an important algorithm in neural network training, especially Convolutional Neural Networks (CNNs). It provides a framework of how these networks process data. Backpropagation is mainly involved with the iterative correction of the network's internal parameters weights and biases by measuring the gradient of the loss function with respect to these parameters.

The backpropagation process starts by sending input data across the network in order to generate predictions. These predictions are then compared to the actual target values using a predetermined loss function, which measures the difference between the expected and real values. The goal is to reduce the loss or cost of the function, which helps improve the overall accuracy of the network's predictions. Backpropagation operates in reverse, beginning at the output layer and progressing backward through the network. Using the chain rule from calculus, it computes the gradient of the loss function with respect to the network's parameters. This gradient reflects the size and direction of change in each parameter necessary to lessen the loss.

The calculated gradients are then used to update the weights and biases in optimization techniques such as gradient descent or its derivatives (e.g., Adam, RMSprop). These updates are carried out iteratively across successive epochs or batches of data, progressively moving the network parameters in the direction of the loss function minimization.

The efficiency and effectiveness of backpropagation lies in its ability to propagate the error signal backward through the network, attributing portions of the error to each parameter. This attribution enables the network to adjust its parameters in a way that improves its predictions, ultimately enhancing its ability to generalize and make accurate predictions on new, unseen data.

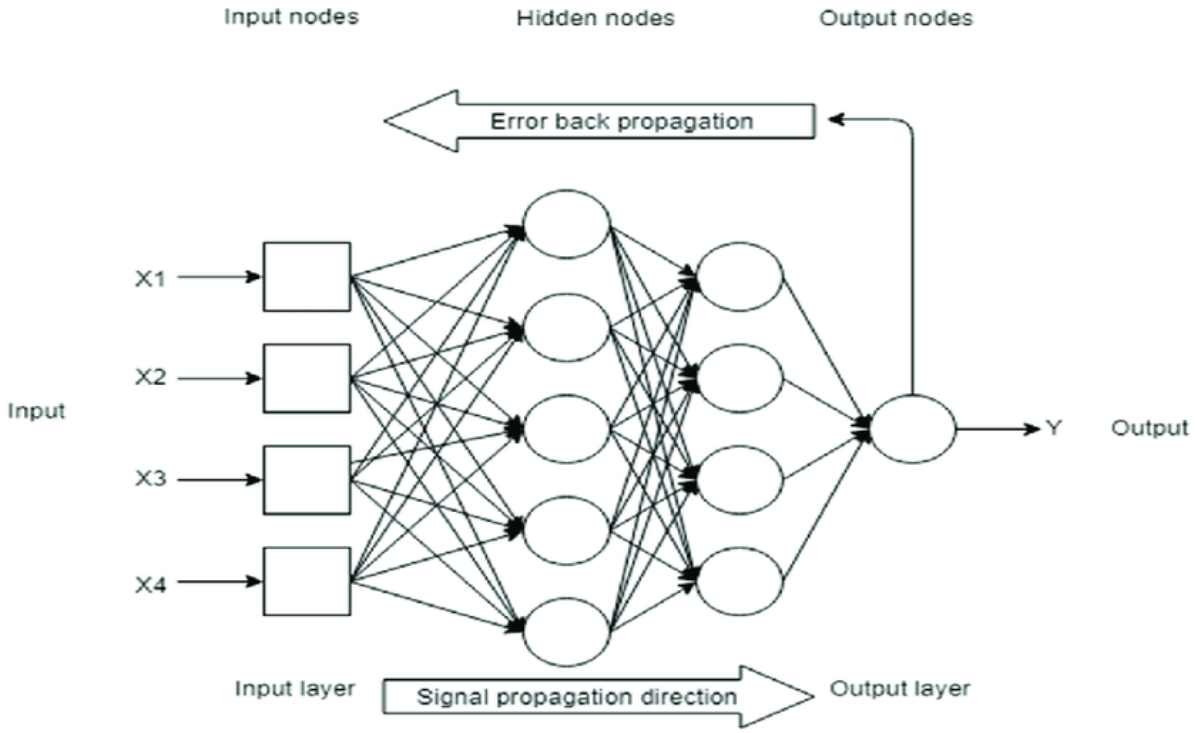


**Figure 6 – Backpropagation process. [24]**

The iterative nature of backpropagation allows the neural network to modify its parameters throughout numerous epochs or iterations through the training data, progressively boosting its capacity to generate correct predictions. However, while backpropagation is strong, it is only one component of neural network training, which also includes regularization techniques, hyperparameter tweaking, and appropriate network designs to attain optimal performance.

### 3.1.7. Tracking Algorithms

Tracking algorithms are algorithms used to track and predict the position, trajectory, or behavior of objects over time. These algorithms are important in fields like computer vision, robotics, satellite tracking, and others. They allow systems to analyze data sequences, such as

frames of video as well as sensor measurements, to calculate the continuous state of one object or multiple objects in a dynamic environment.

Tracking algorithms often use a variety of techniques, such as object identification, feature extraction, matching, and motion estimates, at their heart. Object detection helps in the initial detection of potential objects, whereas feature extraction retrieves distinguishing properties such as edges, corners, or descriptors that represent things. These characteristics serve as tracking identifiers. A matching algorithm help associating these features or detected objects across frames. This process involves establishing correspondence between features in consecutive frames or associating detected objects in subsequent frames to ensure continuity in tracking. Various methods, like the Kanade-Lucas-Tomasi (KLT) tracker or correlation-based trackers, are used for matching features [25].

There are several tracking algorithms, each with distinct features that make them appropriate for different scenarios. One of the most common types of tracking algorithm is Kalman filters. Kalman filters are recursive, Bayesian-based algorithms that use incomplete data collected over time to estimate the state of a system. They forecast an object's future state based on its current state and then update it with new measurements, effectively combining predictions and measurements to obtain an optimal estimate.

In more nonlinear and complex scenarios, traditional methods may struggle due to their reliance on assumptions of linearity and Gaussian noise. Instead, techniques like particle filters step in to navigate these intricate environments. These filters operate by maintaining a collection of weighted samples, often referred to as particles, to encapsulate various potential object states. Their strength lies in their ability to handle situations where system behavior deviates from Gaussian distributions or exhibits high nonlinearity. Particle filters excel in adapting to dynamic and unpredictable settings, effectively maneuvering through changing environments and uncertain dynamics. This adaptability allows them to track objects robustly, even when faced with occlusions, erratic movements, or varying appearances, making them a versatile choice in challenging tracking scenarios.

## 3.2. General Strategies to Use CNN for Object Detection

### 3.2.1. Define Problems

In image detection tasks, defining the problem is the foundational step that sets the basis for effective model building. It involves establishing clear objectives, constraints, and outlining the specific classes or categories to which images will be assigned, as well as specifying the types of objects the final model is capable of detecting based on the given objectives. For example,

in autonomous vehicles development, object detection is used to detect objects such as pedestrians, cars, and traffic signs, allows vehicles to recognize and classify objects in their environment for navigation and safety. Meanwhile, in retail and e-commerce, object detection is used to detect products, logos, and barcodes, which help with product categorization, product checkout, inventory management, etc.

### 3.2.2. Images Collection

Collecting images for an image detection task involves employing various methods to acquire a wide range of essential pictures. Typically, the process begins by searching image database sites such as Google Images and Pinterest to find high detail images suitable for basic classification. In more advanced image detection tasks, data is often collected from real-life scenarios, like public CCTV footage, street cameras, or even sensors installed in vehicles. These real-world sources provide a more diverse and complex collection of images, reflecting different perspectives, lighting conditions, and contexts, enhancing the dataset. However, an ideal dataset usually requires a mixture of both real-world and detailed images to help in the comprehensive understanding and accurate detection of objects. Real-world images offer contextual richness, while high-detail images provide specific features important for precise classification. Combining these varied types of images enriches the dataset, allowing machine learning models to learn diverse representations and patterns more effectively.

### 3.2.3. Data preprocessing

In object detection, data preprocessing acts like a kind of preparation before the actual learning starts. It is like getting things ready for the machine to understand the pictures better. This involves performing preparatory work to assist the model in better understanding the images. Also, adjusting the colors or brightness of the images to make them look similar helps the machine learn better. Another important thing is to change the pixel values of the pictures to fit into a specific range. This process, known as normalization, is essential for uniformly distributing intensity levels across all images. By scaling pixel values within a specific range, often between 0 and 1 or -1 and 1, the images become more consistent and amenable to the learning process.

### 3.2.4. Dataset Annotation

Dataset annotation refers to the process of labeling or tagging images in a dataset with specific information or metadata. In the field of object detection, annotation refers to the marking or

identification of objects or features within an image in order to provide supervised information to the machine learning model during training.

Annotations may come in many forms, such as bounding boxes that outline the location and boundaries of objects, semantic segmentation masks that define pixel-level object delineation, and categorical labels that assign images to predefined categories or classes. By associating visual features with corresponding labels or classifications, these annotations provide essential instruction to the model. For example, in a dataset containing images of animals, annotations could include drawing bounding boxes around each animal in the image, as well as labels indicating the type of animal (e.g., dog, cat, bird).

### 3.2.5. Model Selection

In image processing, model selection involves choosing the best machine learning architecture or algorithm for the specific requirements and objectives of the task at hand. This process takes into account factors such as the project's complexity, available computational capacity, the scope and size of the dataset, and the expected performance metrics.

For image-related tasks like classification, object detection, segmentation, or image generation, various machine learning models or architectures exist, each with its strengths and features for different scenarios. Some popular models include Convolutional Neural Networks (CNNs) like VGG, ResNet, Inception, YOLO and architectures specifically designed for specific tasks like U-Net for segmentation or Generative Adversarial Networks (GANs) for image generation.

### 3.2.6. Model Training

Training an object detection model involves a sequence of technical procedures designed to enable the model to recognize and localize objects within images or video frames. During training, the model learns by minimizing a defined loss function that quantifies the difference between predicted and actual bounding boxes. Combinations of localization loss (measuring the accuracy of bounding box predictions) and classification loss (ensuring correct object class predictions) are common loss functions. To minimize this loss, optimization techniques such as stochastic gradient descent (SGD) or different versions of it adjust the model's parameters.

Training an object detection model often requires multiple iterations (epochs) over the dataset. Fine-tuning involves adjusting hyperparameters, such as learning rate and batch size, to enhance model performance. Additionally, techniques such as data augmentation (flipping, rotation, and zooming) can improve the model's robustness to variations in the input data.

### 3.2.7. Evaluation and Testing

Evaluation in object detection focuses on determining models' performance and accuracy in identifying and localizing objects within images or videos. It compares how well a model recognizes and specifies objects to a predefined standard. This procedure involves comparing the predictions made by the model, such as bounding boxes around objects, to the dataset's annotated or labeled objects. The goal of evaluation is to comprehend how accurately and effectively the model identifies objects of interest, regardless of the metrics used.

Testing in object detection means verifying the model's performance on new, previously unseen data to ensure its reliability and generalization outside the training dataset. The goal of testing is to see if the model can generalize what it learned from the training phase to function properly in real-world scenarios. It ensures that the model's performance is not limited to specific events observed during training but also extends to unexpected circumstances, enhancing its practical effectiveness and reliability in a wide range of situations.

## 3.3. Evaluation Metrics

### 3.3.1. Precision

Testing in object detection means verifying the model's performance on new, previously unseen data to ensure its reliability and generalization outside the training dataset. The goal of testing is to see if the model can generalize what it learned from the training phase to function properly in real-world scenarios. It ensures that the model's performance is not limited to specific events observed during training but also extends to unexpected circumstances, enhancing its practical effectiveness and reliability in a wide range of situations.

Precision determines the ratio of correctly predicted positive instances to the total number of positive instances predicted by the model. It is expressed mathematically as the number of true positive detections divided by the sum of true positive and false positive detections.

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

where,

**True Positives (TP)**: Instances where the model correctly detects the presence of an object.

**False Positives (FP)**: Instances where the model incorrectly detects the presence of an object that is not there.

A high precision value in object detection suggests that when the model predicts the presence of an object, it is highly likely to be correct. It is useful for determining how well the model

avoids false alarms or false positives. In situations where false positives can have serious consequences, such as in security or critical detection systems, a high precision value is critical.

### 3.3.2. Recall

The model's ability to accurately recognize all relevant instances or objects within an image or dataset is measured by recall, also known as sensitivity or true positive rate. It is calculated mathematically as the ratio of the number of true positive detections to the total number of actual positive instances in the dataset.

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

where,

**True Positives (TP)**: Instances where the model correctly detects the presence of an object.

**False Negatives (FN)**: Instances where the model fails to detect an object that is present.

High recall in the context of object detection indicates that the model can detect and capture most of the actual objects of interest. A high recall value indicates fewer false positives or missed detections, which can be crucial in applications where identifying all instances of an object class is important, such as medical imaging or safety-critical systems like autonomous vehicles.

### 3.3.3. Average Precision (AP)

Average Precision (AP) is a metric used in object detection to evaluate the precision-recall curve. It measures how well a model balances precision and recall across different thresholds or levels of confidence. It involves calculating the region below the precision-recall curve, which represents the trade-off between precision and recall as the model's confidence threshold varies.

$$AP = \Sigma_n (R_n - R_{n-1}) \times P_n \quad (3)$$

Here:

$\boldsymbol{R_{n-1}}$ and $\boldsymbol{R_n}$ : consecutive recall values

$\boldsymbol{P_n}$ : the precision at the recall level $R_n$

This formula (3) computes AP by summing the products of the recall $(R_n - R_{n-1})$ change and the corresponding precision at each recall level $(P_n)$. This calculation approximates the area

under the precision-recall curve, quantifying the model's precision across different recall thresholds.

### 3.3.4. Mean Average Precision (mAP)

In object detection tasks, the Mean Average Precision (mAP) metric is used to evaluate a model's overall performance across different object classes. It involves calculating the Average Precision (AP) for each class separately and then averaging the AP values to provide a single measure of the model's performance.

$$mAP = \frac{1}{N} \Sigma_{i=1}^{N} AP_i \quad (4)$$

where,

**N**: the total number of object classes

$\boldsymbol{AP}_i$: the Average Precision for each class $i$

The mAP metric offers an established evaluation of the model's performance across different object classes, taking into account both the precision and recall trade-offs for each class individually before averaging these results to assess the overall efficacy of the object detection model.

In addition to mAP, another useful measure in object detection evaluation is Mean Average Precision at an Intersection over Union (mAP@IoU). It is a modification of the standard Mean Average Precision (mAP) that incorporates the concept of Intersection over Union (IoU). IoU calculates the overlap between predicted and actual bounding boxes for objects. mAP@IoU evaluates the model's ability to precisely localize objects in images by considering varying IoU thresholds and calculating the average precision across these thresholds.

## 3.4. Research Methods

The dataset is created through a process called data fusion, which involves combining datasets from different sources. These images then undergo a normalization process to resize them to 640x640. After this step, the image set is expanded through an augmentation process before being prepared for training to obtain the final dataset. Subsequently, the final dataset is partitioned into three distinct subsets for training, evaluation, and testing purposes.

The process of training begins by feeding the established training dataset into a pre-trained YOLOv8 model, then training with stochastic gradient descent (SGD) as optimization method. Following the training stage, the model is evaluated using a variety of metrics to determine its accuracy and performance on the evaluation dataset. Following that, the model is put to the test by being run through a separate testing dataset. During this phase, graphs and charts are created

based on the testing results to visually represent and analyze the model's performance, providing valuable insights into the model's effectiveness and robustness in real-world applications.

To summarize the research methods, process a flowchart is provided below:

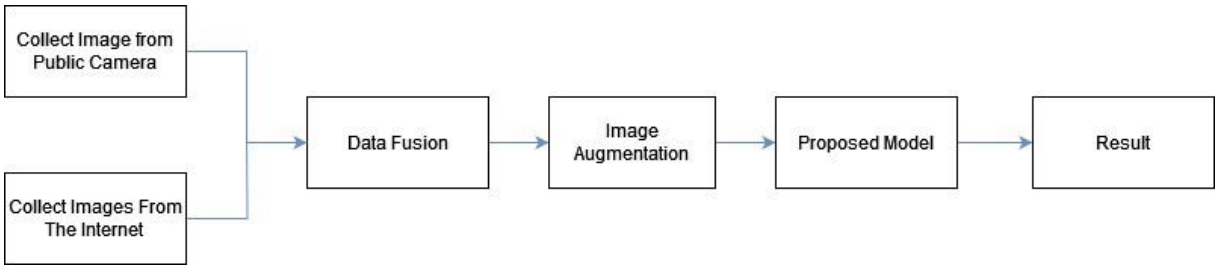


**Figure 7 – Research method.**

### 3.4.1. Data Augmentation

Data augmentation is a machine learning technique that uses different alterations on existing data to artificially improve the diversity of a dataset. This commonly involves adding random variations to images, such as rotation, scaling, flipping, changes in brightness or contrast, and other distortions, in the context of the original image. The expanded dataset is then used to train machine learning models, improving their capacity to generalize to new data and overall performance.

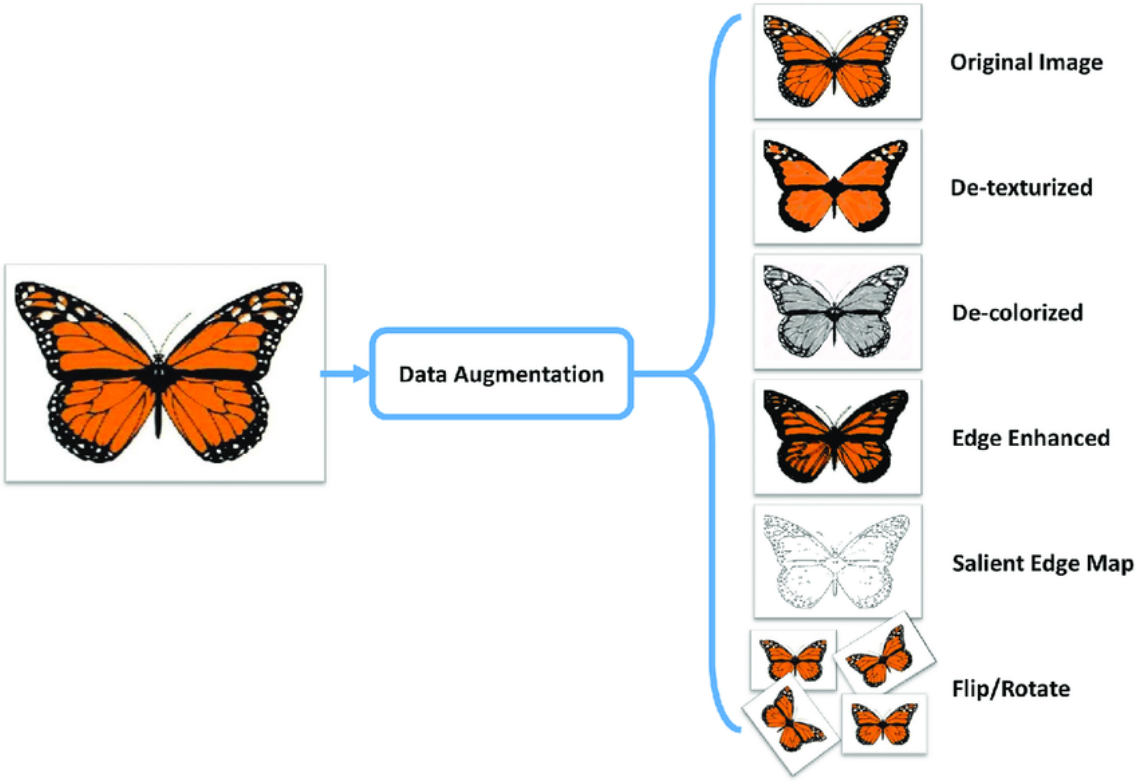


**Figure 8 – Data augmentation [26]**

The implementation of data augmentation is important for a RLR detection model. It increased robustness by exposing the model to a broader range of variations in input data. Given that red light running violations can occur under different conditions, including changing lighting, weather, and traffic scenarios, the ability of the model to adapt to these changes is significant.

### 3.4.2. Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent (SGD) is an optimization algorithm often used in machine learning, especially for training models. Fundamentally, SGD is an iterative optimization technique that modifies a model's parameters in order to minimize a certain loss function. SGD is "stochastic" because, instead of processing the entire dataset at once, it uses random mini-batches, or subsets, of the training data during each iteration. To prevent biases linked to order, the training data is randomly shuffled after the model's parameters are initialized. A mini-batch is chosen at random for each iteration, and the gradient of the loss function with respect to the model parameters is then calculated. SGD is represented by the formula:

$$\theta_{new} = \theta_{old} - \eta \nabla J(\theta_0; x^i, y^i) \quad (5)$$

where:

$\theta_{new}$: the updated parameters after iteration.

$\theta_{old}$: the updated parameters before iteration.

$\eta$: the learning rate which is a positive scalar determining the step size for parameter updates.

$J(\theta_0; x^i, y^i)$: the loss function that measures the difference between the model's prediction and the actual target.

$\nabla J(\theta_0; x^i, y^i)$: the gradient of the loss function.

## 3.5. Proposed Technologies

### 3.5.1. Labeling Tools

In order to label the dataset images for this project, Roboflow [27], which is an advanced and user-friendly platform for data annotation, was utilized. Roboflow streamlines the complicated task of image annotation by offering a suite of annotation tools designed to accelerate the labeling process. This platform is flexible and compatible with other machine learning frameworks since it supports multiple annotation formats, including COCO JSON, Pascal VOC XML, and YOLO format.

### 3.5.2. Proposed Deep Learning Architecture

#### 3.5.2.1. YOLO Architecture

The YOLO (You Only Look Once) architecture revolutionized object detection by treating the task as a regression problem. In this YOLO, object detection is solving as a regression problem by directly predicting spatially separated bounding boxes and associated class probabilities from full images in a single evaluation [28]. The entire detection pipeline is unified into a single neural network, allowing end-to-end optimization directly on detection performance.

The core idea behind YOLO is to divide the input image into a grid, typically with dimensions S×S [29]. Each grid cell takes responsibility for predicting objects within its region, making the detection process grid-centric. Inside of each grid cell, YOLO predicts multiple bounding boxes denoted as $B$, alongside with class probabilities ($C$) where C is the number of classes in the dataset [30]. For every bounding box, the network predicts five values: $x, y, w, h, conf$, where $(x, y)$ are the coordinates of the box's center, $(w, h)$ are the width and height of the box, and $conf$ is the confident score [4]. The output tensor of the YOLO network has dimensions $S \times S \times (B \times 5 + C)$, in which $B \times 5$ correspond to the bounding box predictions, including coordinates and confidence scores, and the $C$ represent class probabilities.

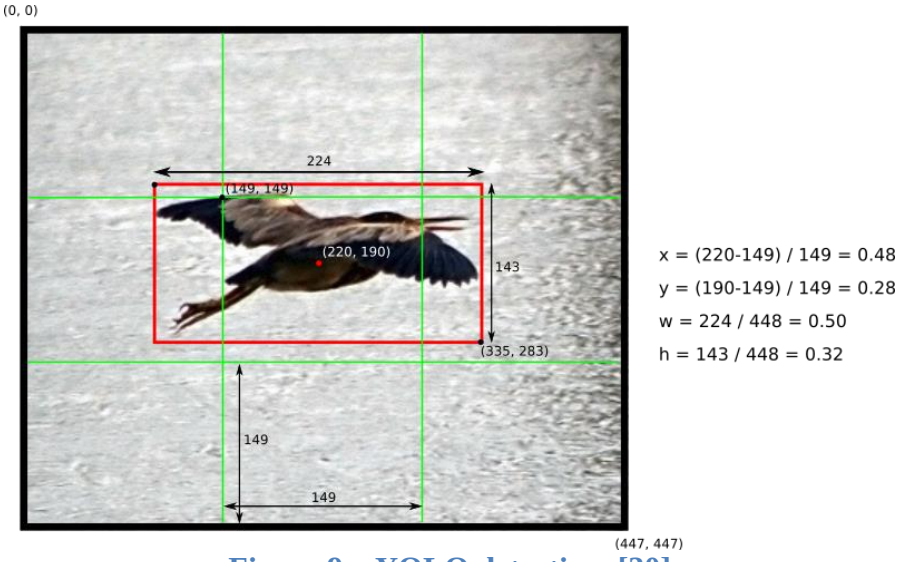


**Figure 9 – YOLO detection. [30]**

#### 3.5.2.2. YOLO Loss Function

The YOLO loss function is an essential part of the model's training process, instructing the optimization of the parameters of the neural network in order to improve detection accuracy.

The YOLO loss function, is a combination of four different loss function: Localization Loss ($L_{coord}$), Confidence Loss ($L_{conf}$), Classification Loss ($L_{class}$), and Confidence Loss for Non-object cells ($L_{noobj}$). Each component handling different aspects of the detection process.

First of all, Localization Loss function ($L_{coord}$) is responsible for evaluating the accuracy of the predicted bounding box coordinates compared to the ground truth coordinates.

$$L_{coord} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] +$$

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[ \left( \sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left( \sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

(6)

The function help penalizes the model for inaccuracies in predicting both the center coordinates ($x$,$y$) and the width-height dimensions ($w$, $h$) of the predicted bounding boxes.

Secondly, Object Confidence Loss function ($L_{conf}$) is represented by

$$L_{conf} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left( C_i - \hat{C}_i \right)^2 \quad (7)$$

It evaluates how well the predicted confidence scores ($C_i$) align with the ground truth confidence scores ($\hat{C}_i$) . This loss is computed only for cells containing objects, and it penalizes the model for deviations in confidence predictions.

Moreover, Classification Loss function ($L_{class}$) represented by the function

$$L_{class} = \sum_{i=0}^{S^2} 1_{ij}^{obj} \sum_{c \in classes} [\, p_i(c) - \hat{p}_i(c)]^2 \quad (8)$$

It helps to assess the accuracy of the predicted class probabilities $p_i(c)$ in comparison to the actual class probabilities $\hat{p}_i(c)$ class. This loss is computed only for cells that contain objects, and it is summed across all classes.

Finally, Confidence Loss for Non-Object ($L_{noobj}$) is represented by the formula

$$L_{noobj} = \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{noobj} \left( C_i - \hat{C}_i \right)^2 \quad (9)$$

The function penalizes the model for making confident predictions in cells where there is no object.

The entire formula of YOLO loss function is represented by combining (6), (7), (8), (9).

$$Loss = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2]$$

$$+ \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left[ \left( \sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left( \sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} \left( C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{noobj} \left( C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{S^2} 1_{ij}^{obj} \sum_{c \in classes} [p_i(c) - \hat{p}_i(c)]^2 \quad (10)$$

where:

$S$: the grid size.

$B$: the number of bounding box predictors per cell.

$\lambda_{coord}$: the coefficient of the bounding box coordinates.

$\lambda_{noobj}$: the coefficient of the no-object confidence.

$1_{ij}^{obj}$: the indicator function, it is 1 if there is an object in cell $i$ associated with bounding box $j$, and 0 otherwise.

$1_{ij}^{noobj}$: the indicator function, it is 1 if there is no object in cell $i$ associated with bounding box $j$, and 0 otherwise

#### 3.5.2.3. YOLOv8 Overview

The YOLOv8 model is the most recent version in the YOLO architecture series development Ultralytics, it has demonstrated better performance than the previous models. YOLOv8 come in five different sizes, YOLOv8n (nano), YOLOv8s (small), YOLOv8m (medium), YOLOv8l (large), and YOLOv8x (extra-large).

| Model | size (pixels) | mAP$^{val}$ 50-95 | Speed CPU ONNX (ms) | Speed A100 TensorRT (ms) | params (M) | FLOPs (B) |
|---|---|---|---|---|---|---|
| YOLOv8n | 640 | 37.3 | 80.4 | 0.99 | 3.2 | 8.7 |
| YOLOv8s | 640 | 44.9 | 128.4 | 1.20 | 11.2 | 28.6 |
| YOLOv8m | 640 | 50.2 | 234.7 | 1.83 | 25.9 | 78.9 |
| YOLOv8l | 640 | 52.9 | 375.2 | 2.39 | 43.7 | 165.2 |
| YOLOv8x | 640 | 53.9 | 479.1 | 3.53 | 68.2 | 257.8 |

Figure 10 – YOLOv8 architectures. [6]

Smaller models like YOLOv8n and YOLOv8s, having fewer layers and parameters, are better for real-time applications with limited resources because they reduce computational demands while allowing faster inference. However, these models may sacrifice some accuracy, especially when detecting smaller objects or complex scenes. On the other hand, larger YOLO models, with more layers and parameters, excel in achieving higher accuracy and better performance on challenging tasks. They are well-suited for scenarios where accuracy is the main focus but come with the drawback of having higher computational requirements, which makes them less efficient for real-time processing or environments with limited resources.

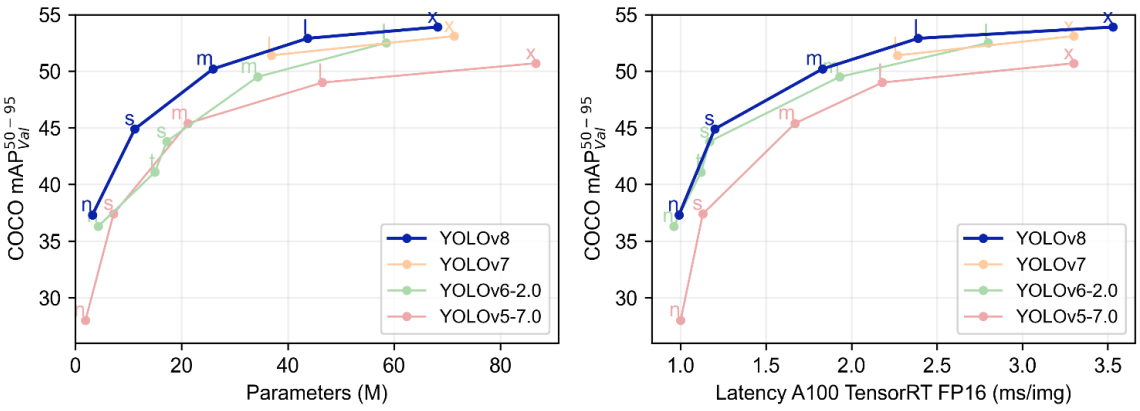


Figure 11 – YOLOv8 versions comparison. [6]

YOLOv8 supports multiple vision tasks, including object detection, segmentation, pose estimation, tracking, and classification. It employs an anchor-free model with a decoupled head for independent processing of objectness, classification, and regression tasks, enhancing overall accuracy. The output layer uses a sigmoid function for objectness scores and softmax for class probabilities. YOLOv8 used a deep convolutional neural network (CNN) architecture with

significant enhancements, including the efficient CSPNet backbone, the FPN+PAN neck for improved feature aggregation, and the PANet head for increased robustness to occlusion and scale variations. YOLOv8 operates by dividing input images into a grid of cells, predicting bounding boxes and class probabilities for each cell, and using non-maxima suppression for final object selection.

To improve performance, YOLOv8 utilizes CIoU and DFL loss functions for bounding box loss and binary cross-entropy for classification loss, particularly benefiting smaller objects [5]. Additionally, YOLOv8 introduces YOLOv8-Seg for semantic segmentation, featuring a CSPDarknet53 backbone, a C2f module, and two segmentation heads [5]. This model achieves state-of-the-art results on various benchmarks while maintaining high speed and efficiency.

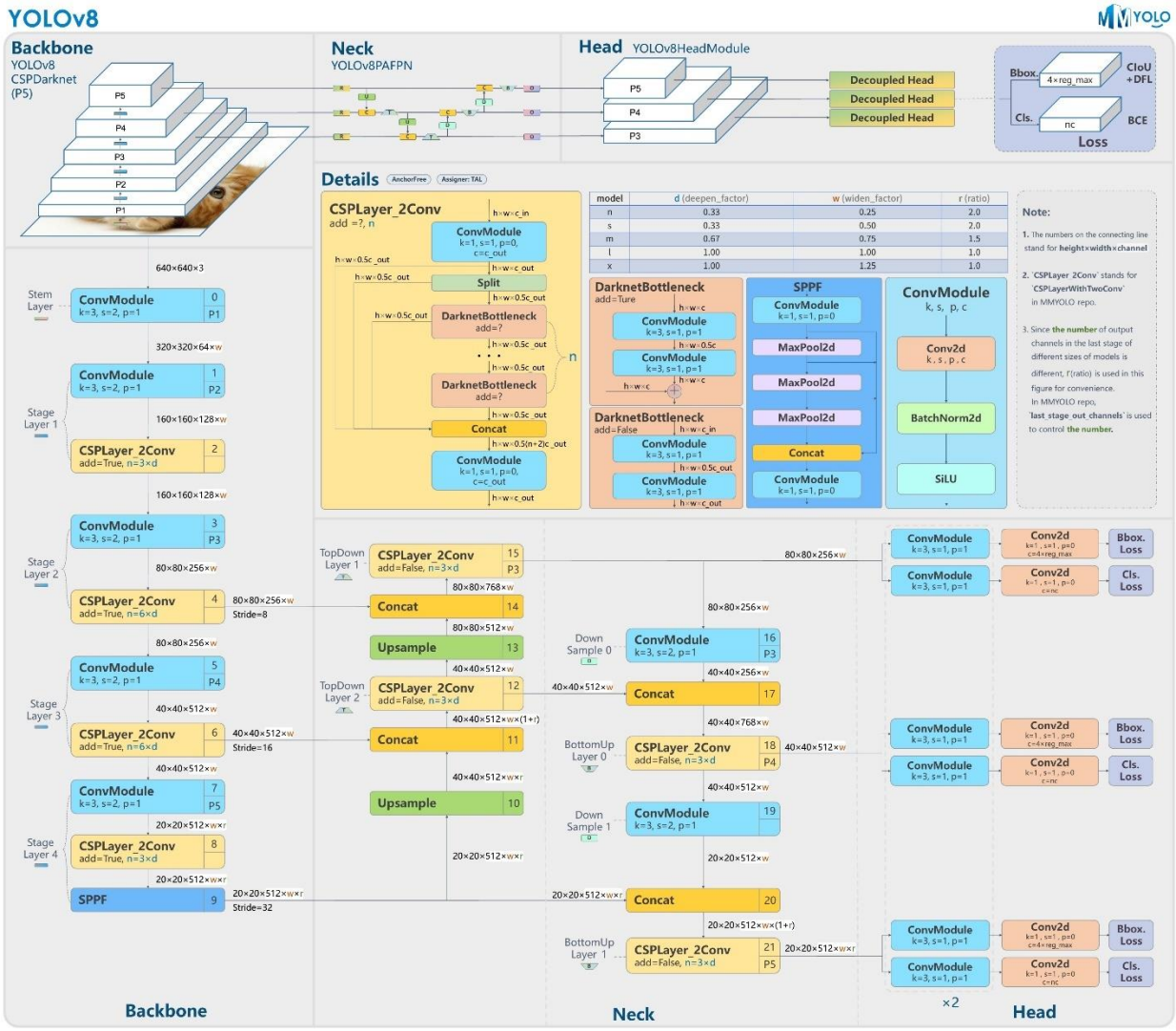


**Figure 12 - YOLOv8 architecture [32]**

Base on **Figure 12**, the loss function of YOLOv8 is composed of two primary components: the classification branch and the regression branch. The classification branch employs Binary Cross Entropy (BCE) Loss and is responsible for determining the correct object class. This loss

is weighted by the hyperparameter "cls," which controls its relative contribution to the overall loss. On the other hand, the regression branch handles bounding box prediction and utilizes a combination of two independent losses: Distribution Focal Loss (DFL) and Complete Intersection over Union (CIoU) Loss. DFL determines class imbalance in object detection tasks, especially benefiting objects with unclear boundaries. CIoU Loss considers both the overlap between predicted and ground truth boxes and the aspect ratio difference, enhancing the accuracy of bounding box regression. The hyperparameter "bbox" controls the weighting of this combined regression loss in the total loss calculation.

### 3.5.3. Proposed Tracking Algorithms

#### 3.5.3.1. DeepSORT: A Deep Learning-Based Tracking Algorithm

DeepSORT, short for "Deep Simple Online and Realtime Tracking," is an advanced tracking algorithm created for multi-object tracking in videos. To achieve accurate and real-time object tracking, it combines deep learning methods with conventional computer vision techniques. The algorithm establishes upon the existing SORT (Simple Online and Realtime Tracking) algorithm by including deep appearance descriptors, enhancing its ability to handle object identity across frames more effectively.

DeepSORT uses a two-step process: detection and association. In the detection phase, a pre-trained object detection model determines objects in each frame of a video. Following detection, the objects move on to the association phase, when DeepSORT focuses on maintaining object identities between frames. In order to do this, the algorithm collects each object's deep appearance features and creates a signature which makes it easier to tell one object apart from another.

In the association stage, DeepSORT applies the Kalman filter for state prediction and a data association algorithm, often based on the Hungarian algorithm, to assign detections to existing tracks. To create precise correlations, the system takes into account both the appearance attributes and the motion predictions.

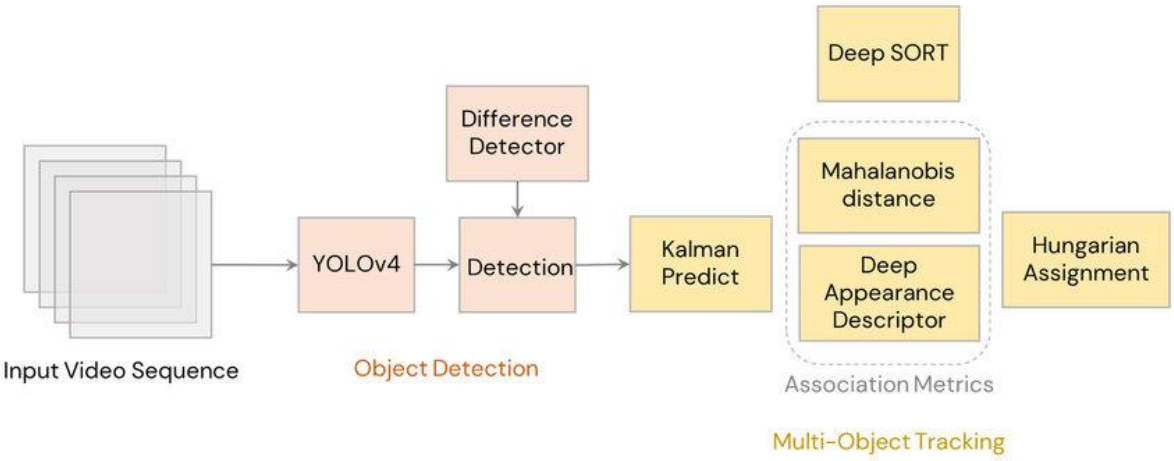


**Figure 13 – DeepSORT architecture [33]**

# CHAPTER 4

# IMPLEMENTATION AND RESULTS

## 4.1. Implementation

### 4.1.1. Dataset

The dataset use in this study includes 3635 total images, of which 3035 were used in the training phase, 368 in the evaluation phase, and 232 in the testing phase. The dataset containing image of 8 object classes (5 vehicle types and 3 traffic light types): car, motorbike, bike, bus, truck, red-light, green-light, and yellow-light.

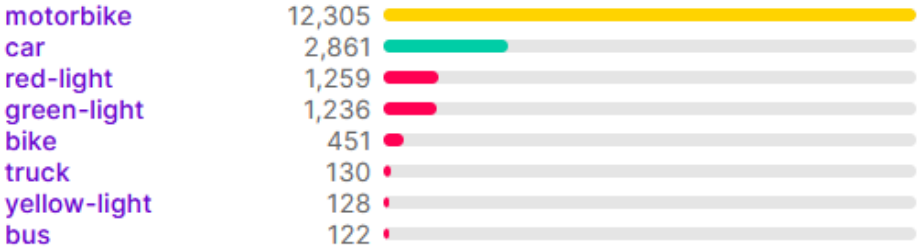


**Figure 14 - Number of instances of every classes**

The dataset prepared for this research is created from many different sources. The primary source is footage obtained from public cameras and then spliced into frames in Adobe Photoshop. In order to improve the dataset's variety and complexity, data augmentation techniques were applied to modify existing images by altering their rotation, contrast level, and brightness.

Figure 15 - Dataset augmentation

### 4.1.2. Training Model

The model was trained using a computer with the following specification:

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

Table 1 - Computer Specification

The proposed YOLOv8l was choose for the training, and was trained with 120 epochs at a batch size of 64.

### 4.1.3. Technical Implementation

To identify instances of RLR violations, the YOLOv8 model, after being trained, is used to perform inference on public road footage. This consists of detecting various objects (vehicles and traffic lights) through different frame of the video. The confidence threshold for this implementation was set at 0.5 or higher to ensure accurate and reliable detections. In each frame of the footage, the YOLOv8 model generates a list containing the object's positional coordinates (x, y, w, h) to form bounding boxes along with their confidence score.

Figure 16 - A batch size of the dataset

Next, this list of objects undergoes processing through the DeepSORT tracker, assigning a unique ID to each element in the list. The coordinates return by the tracker are then mapped using an algorithm with the coordinates provided by the model which allows the display of these coordinated objects inside the frames. This integration of tracking and coordination algorithms ensures the intuitive visualization and monitoring of detected objects across each frame in the video footage.

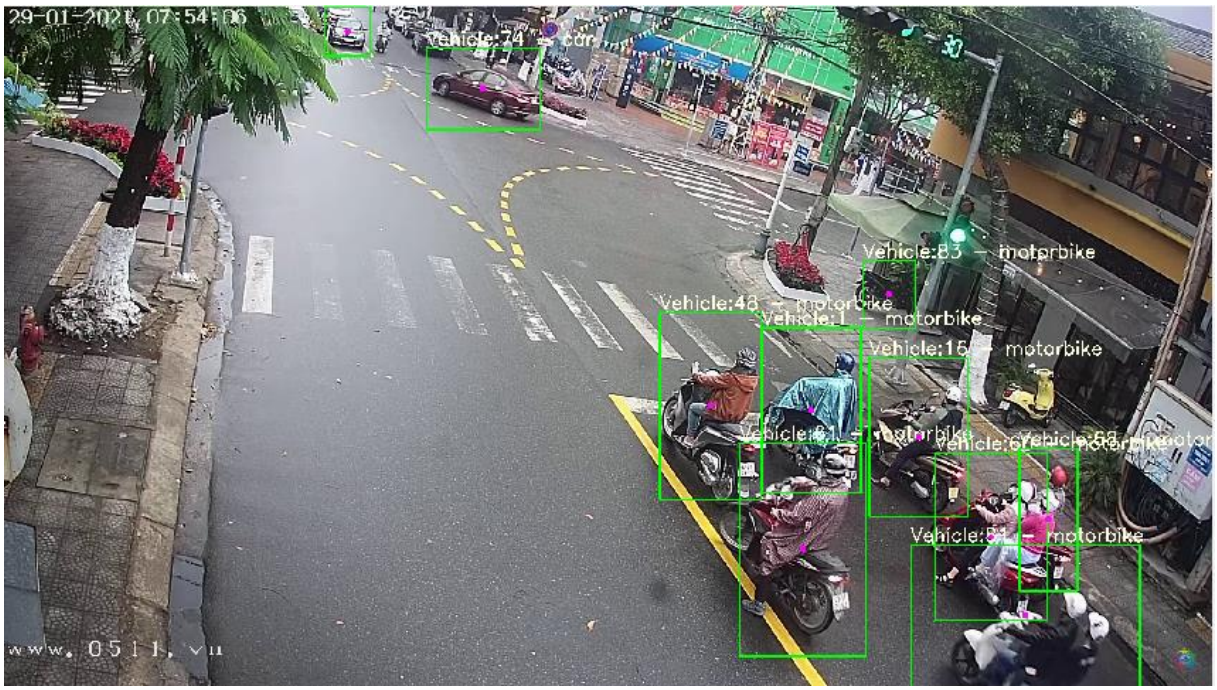


Figure 17 - DeepSORT tracking

To identify instances of vehicles crossing at a red traffic light, it is essential to manually set coordinates for the specific area where violations are recorded. Given the unique characteristics of every intersection out there, this procedure is necessary to define the region of interest. The system can concentrate its detection efforts on those regions where red-light violations are most likely to happen. Once these regions of interest are established, a mechanism for turning on and off the detection process needs to be implemented. This can be done by detecting whether different kinds of traffic lights are present in each frame. It means that when the red light is on, the system will only activate the violation detection process; when the yellow or green light is on, the system will deactivate detection.

**Figure 18 - Red-light running detection**

To determine whether a vehicle engaged in RLR, an examination of the vehicle's position relative to the predefined restricted region is needed. To do this, mark the center of each vehicle's bounding box with a point. The following formula can be used to determine the bounding boxes' center point coordinates.

$$C_x = \frac{x_0 + (x_1 - x_0)}{2} \quad (11)$$

$$C_y = \frac{y_0 + (y_1 - y_0)}{2} \quad (12)$$

where:

$x_0$ and $y_0$: the coordinates of the top-left corner of the bounding box

$x_1$ and $y_1$: the coordinates of the bottom-right corner of the bounding box

After calculating the x and y coordinates of the center point, these values become important for tracking the positions of vehicles. By tracking the fluctuations in $C_x$ and $C_y$ then comparing

them with predefined region of interest throughout consecutive frames, the location and movement of vehicles can be monitored. This approach enables the detection and flagging of potential RLR violations.

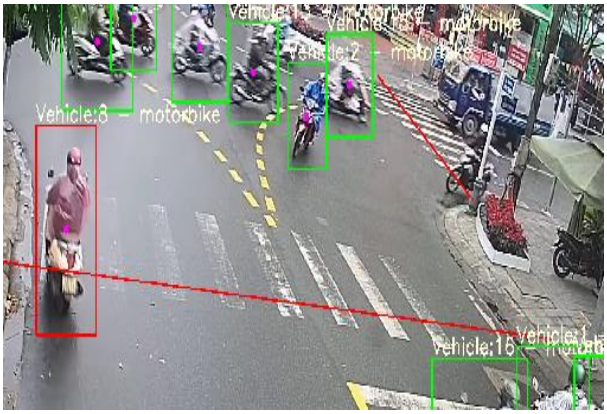


**Figure 19 - Vehicle flagging**

However, there is a problem with the approach. In many cases, Vietnamese traffic laws usually do not penalize vehicles for making right turns at red lights. In order to remove marked vehicles from the list of offenders, an additional region of interest must be established. When a vehicle turns right to cross the region, the traffic violation record for that vehicle will be cleared if it was flagged as an offender when it went through the first line at a red signal.

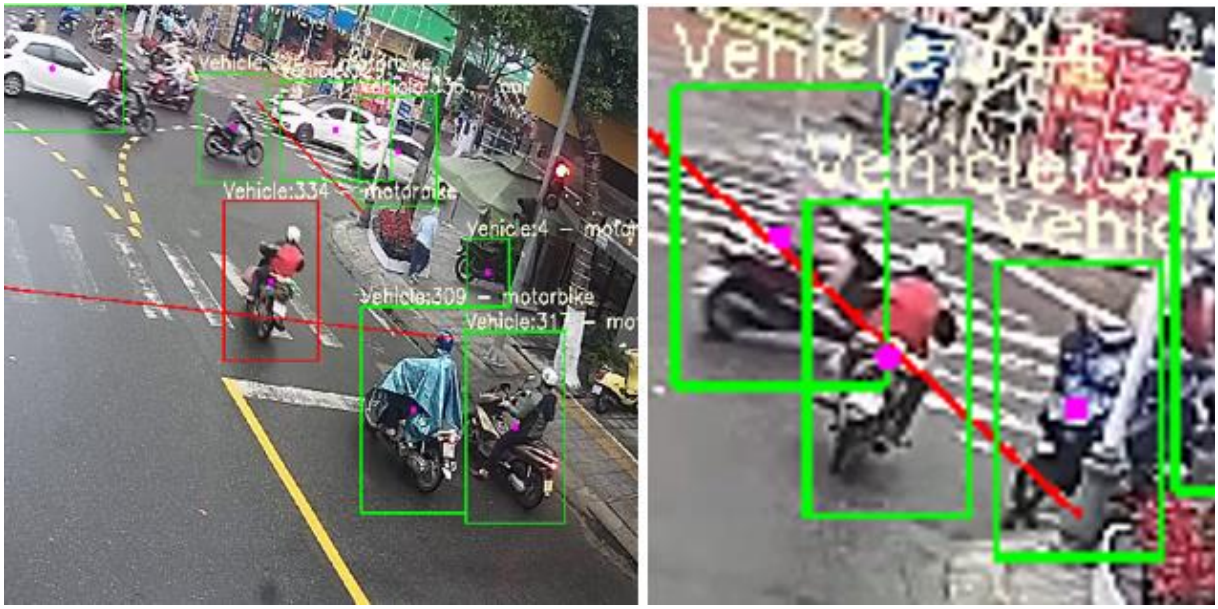


**Figure 20 - Right turner filtering**

### 4.1.4. Graphic User Interface (GUI)

The graphical user interface (GUI) operates as the main interface point for users to navigate and control the RLR detection system. With a focus on simplicity, the user interface (UI) is

designed to allow users with various levels of technical proficiency to easily use the system without requiring any coding knowledge. The interface was created with PyQt5, a powerful Python UI-building package. The UI was designed as follow:

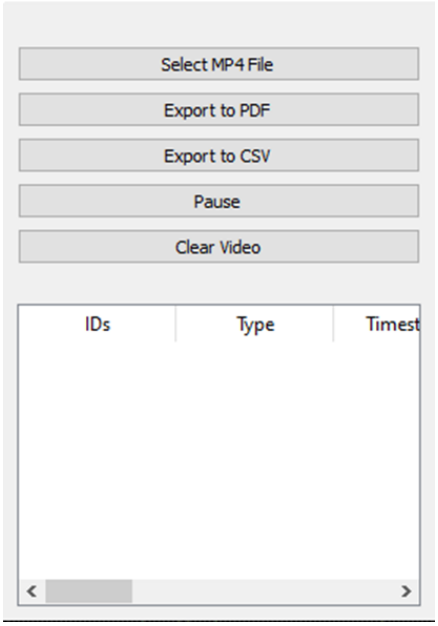


**Figure 21 - System UI**

The “**Select MP4 File**” button is used to open the video footages. When a video is selected it will be displayed in the UI. After loading the video, the user can manage the detection process by using the "Pause" button to halt the video or the "Clear Video" button to reset everything.

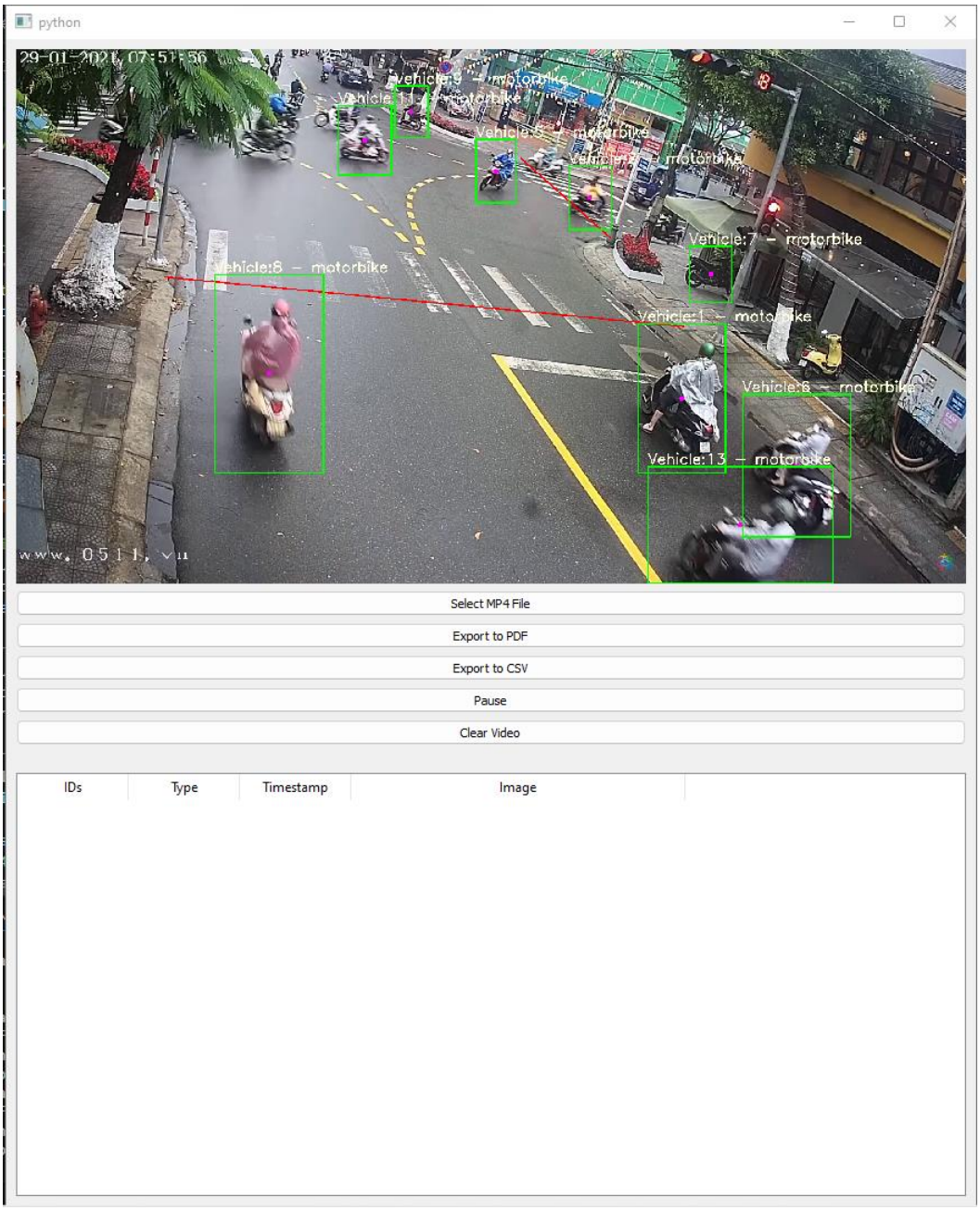


**Figure 22 - System performing detection**

The system systematically analyzes the video frame by frame to identify potential red-light violations. Upon detecting a violation, the system will display it in the table below.

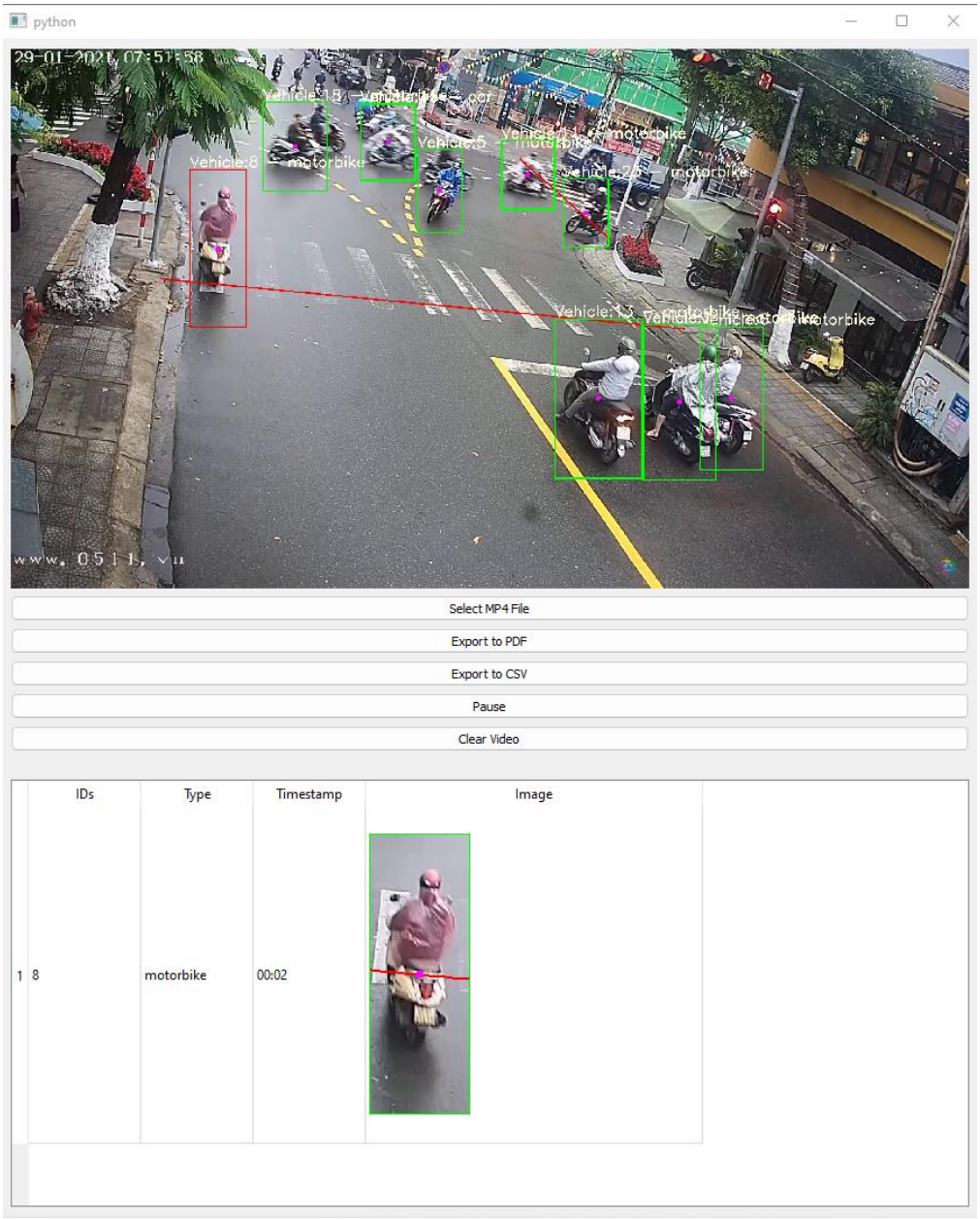


**Figure 23 - System recorded violation**

When violations are recorded, the user has the option to export the list to either a PDF or CSV format using the export buttons.

## 4.2. Training Results

The trained model reached 92.4% precision, 91.2% recall and 94.0% mAP50. These results are better illustrated in these following training graphs:

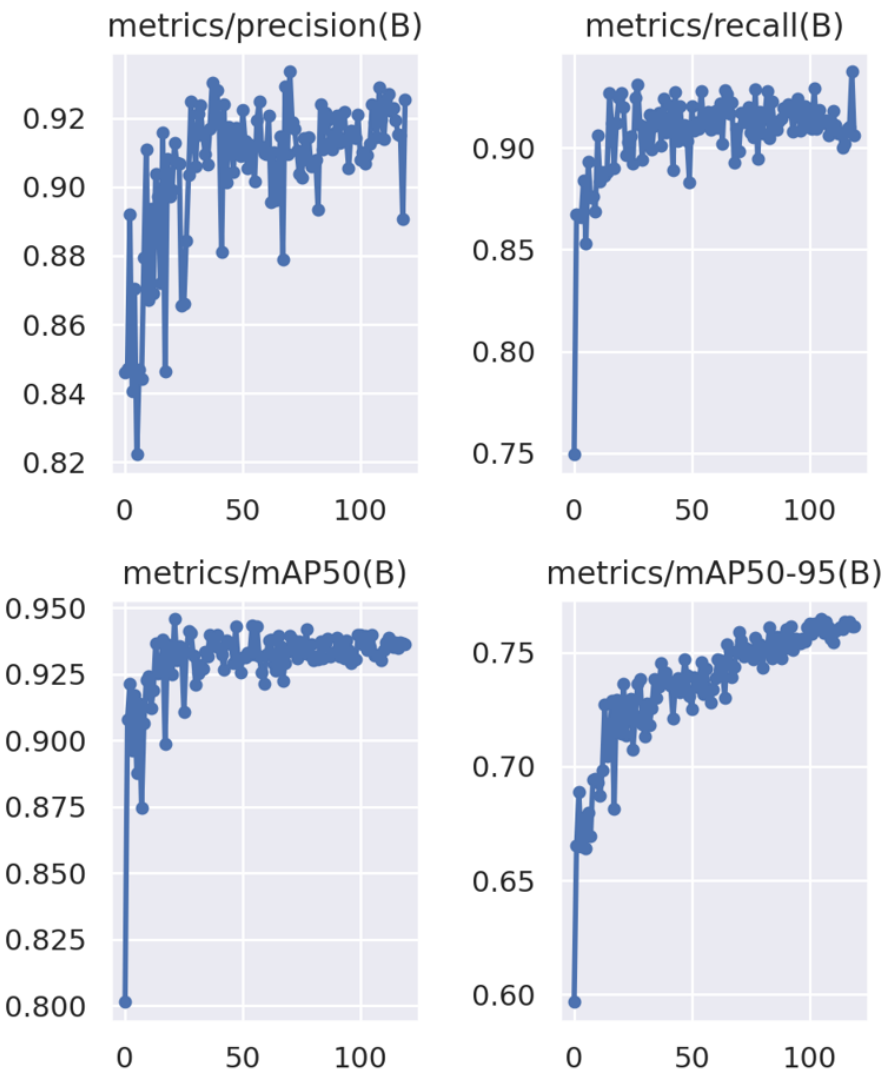


Figure 24 - Validation metrics

In the training, the model box loss is 0.5, the classification loss is 0.25 and the dfl loss is 0.9, which can be seen in Figure X.


Figure 25 - Training Loss

The loss results of the validation phase are shown in the Figure 25.

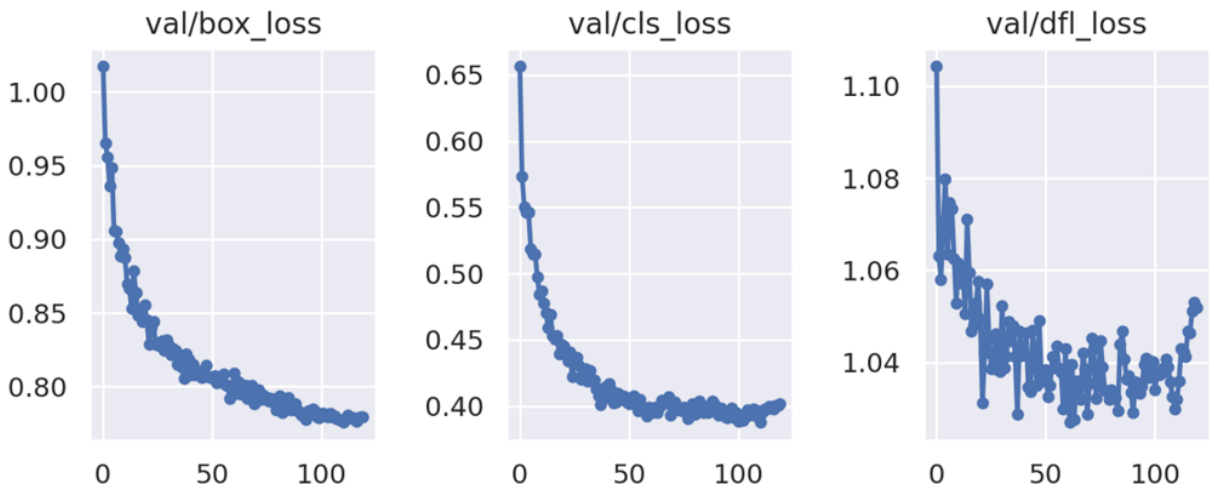


**Figure 26 - Validation Loss**

To evaluate the model performance in every class, a confusion matrix was created to provide a comprehensive breakdown of the model's predictions across multiple classes. This matrix serves as a tool for understanding how well the classification algorithm performs for each individual class within the dataset.

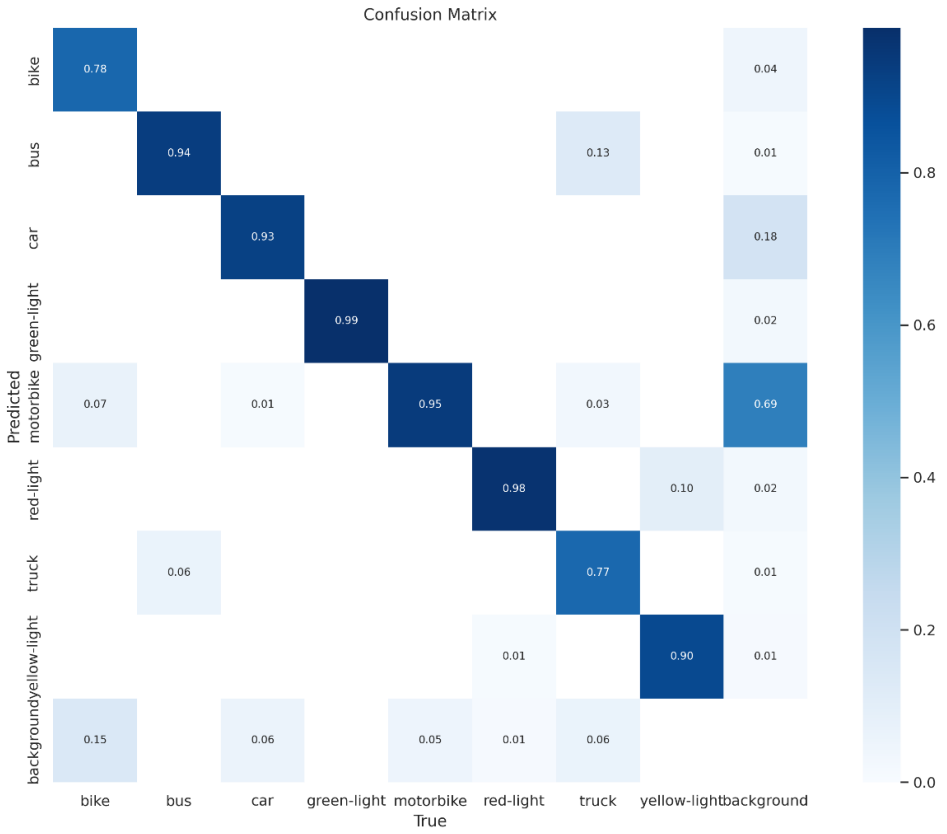


**Figure 27 - Confusion Matrix**

## 4.3. Test Results

When running the model on the testing dataset with 232 images, the model achieved 90.1% precision, 90.2% recall, and 93.1% mAP50. The testing results are slightly lower than the evaluation results, but they are considered negligible.

# CHAPTER 5

# DISCUSSION AND EVALUATION

## 5.1. Discussion and Evaluation

Based on the provided figures, the model demonstrates relatively high performance, with all evaluation metrics at around 90%. However, a look of the validation graph for Distribution Focal Loss (DFL) shows noticeable fluctuations and inconsistencies. These results suggest the possibility of overfitting, particularly in certain classes.

This is proved to be true in **Figure 14**, the instances of motorbike are many times higher than the other classes. This is due to the fact that the dataset is collected mostly using CCTV footages in Vietnam where motorbike is the most common type of transportation. The pronounced overrepresentation of motorbike instances highlights a dataset bias toward this specific class. Such uneven distribution may lead to a model that is overly focused on recognizing and categorizing motorbikes, potentially compromising its ability to generalize well to other classes.

In the **Figure 27**, due to overfitting, the class "motorbike" can sometimes be mistakenly detected in the background where there are no motorbikes. Nevertheless, the model should still be able to correctly detect motorbikes 95% of the time. The model will also be able to detect all traffic lights and vehicle classes except for bikes and trucks, with an accuracy level of around 90%.

Another problem with the system during testing is that it requires a lot of GPU power. This is mainly caused by DeepSORT, which demands significant resources to run smoothly. When running the system using a GTX 1660, it was able to operate at 10 FPS under low traffic conditions, but only achieved 3 FPS when there were a lot of vehicles in the frames. Therefore, it is recommended to use a GPU such as GTX 1080 or higher for optimal performance.

In addition, since the system uses camera videos as input data, a major disadvantage of dependent on a system like this is its strongly reliance on the quality of the footage and the positioning of CCTV cameras. The model will have trouble identifying objects or may produce false positives if the video quality is too low. Similarly, if the camera is positioned in a way that does not cover the entire street view, there is a likelihood of missing certain violations.

# CHAPTER 6

# CONCLUSION AND FUTURE WORK

The study presents an unconventional approach for RLR detection by applying object detection alongside a tracking algorithm to help alleviate accidents related to RLR in Vietnam. The proposed YOLOv8 model was able to perform real-time detection with a fairly accurate rate of more than 90% in almost all metrics. The results demonstrate that, although the model achieved high accuracy in detection, there are still limitations. One limitation is dataset overfitting in the motorbike classes, leading to inconsistent validation of the model. This can be fixed by simply modifying the data in feature implementation. However, certain limitations, such as computational resources and inconsistent input data, require both financial investment and more careful selection of data during the implementation phase. Therefore, the current plan for future system implementation involves improving the dataset's quality to enhance the system's performance in identifying and addressing RLR violations. This includes refining data collection methods and ensuring a more representative and diverse set of examples to better train the model for real-world scenarios, while also expanding the types of vehicles the system can detect, including ambulance, police bikes, and fire trucks. This expansion aims to enhance the functionality of the system, making it more adept at recognizing a broader range of emergency and law enforcement vehicles. This step is crucial for improving the system's overall performance and responsiveness in diverse traffic situations, ultimately contributing to its effectiveness in promoting road safety.

# REFERENCES


1. Campbell, B. N., Smith, J. D., & Najm, W. (2004). Analysis of fatal crashes due to signal and stop sign violations (No. FHWA-JPO-05-050). United States. National Highway Traffic Safety Administration.
2. World Health Organization. (2019). Global status report on road safety 2018. World Health Organization.
3. VietNamNet News (April 15, 2024), Ministry proposes installing cameras to detect traffic violations, https://vietnamnet.vn/en/ministry-proposes-installing-cameras-to-detect-traffic-violations-703422.html.
4. Tuoi Tre News (2023, July 25), Vietnam's Da Lat City starts traffic surveillance camera system, https://tuoitrenews.vn/news/society/20230725/vietnams-da-lat-city-starts-traffic-surveillance-camera-system/74587.html
5. Terven, J., & Cordova-Esparza, D. (2023). A comprehensive review of YOLO: From YOLOv1 to YOLOv8 and beyond. arXiv preprint arXiv:2304.00501.
6. Jocher, G., Chaurasia, A., & Qiu, J. (2023). YOLO by Ultralytics (Version 8.0.0) [Computer software]. https://github.com/ultralytics/ultralytics
7. 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. In 2020 IEEE 7th international conference on industrial engineering and applications (ICIEA) (pp. 774-778). IEEE.
8. Momin, B. F., & Mujawar, T. M. (2015, March). Vehicle detection and attribute based search of vehicles in video surveillance system. In 2015 international conference on circuits, power and computing technologies [ICCPCT-2015] (pp. 1-4). IEEE.
9. Yung, N. H. C., & Lai, A. H. (2001). An effective video analysis method for detecting red light runners. IEEE Transactions on Vehicular Technology, 50(4), 1074-1084.
10. Zaheri, D., & Abbas, M. (2015, September). An algorithm for identifying red light runners from radar trajectory data. In 2015 IEEE 18th International Conference on Intelligent Transportation Systems (pp. 2683-2687). IEEE.
11. Luo, D., Huang, X., & Qin, L. (2008, August). The research of red light runners video detection based on analysis of tracks of vehicles. In 2008 International Conference on Computer Science and Information Technology (pp. 734-738). IEEE.
12. 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.
13. Li, M., Chen, X., Lin, X., Xu, D., & Wang, Y. (2018). Connected vehicle-based RLR prediction for adaptive signalized intersections. Journal of Intelligent Transportation Systems, 22(3), 229-243.
14. 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.
15. Budiardjo, B., & Ramli, K. (2013). A DISCRETE TRACKING BASED-ON REGION FOR RLR DETECTION. International Journal of Engineering Science and Technology, 5(4), 772.
16. Hussain, Q., Alhajyaseen, W. K., Brijs, K., Pirdavani, A., & Brijs, T. (2020). Innovative countermeasures for red light running prevention at signalized intersections: A driving simulator study. Accident Analysis & Prevention, 134, 105349.
17. Baratian-Ghorghi, F., Zhou, H., & Zech, W. C. (2016). RLR traffic violations: A novel time-based method for determining a fine structure. Transportation research part A: policy and practice, 93, 55-65.

18. Jahangiri, A., Rakha, H. A., & Dingus, T. A. (2015, September). Adopting machine learning methods to predict RLR violations. In 2015 IEEE 18th international conference on intelligent transportation systems (pp. 650-655). IEEE.
19. Gonzalez Viejo, C., Torrico, D. D., Dunshea, F. R., & Fuentes, S. (2019). Emerging technologies based on artificial intelligence to assess the quality and consumer preference of beverages. Beverages, 5(4), 62.
20. Serrano, W. (2017). Smart internet search with random neural networks. European Review, 25(2), 260-272.
21. Hassan, E., Khalil, Y., & Ahmad, I. (2020). Learning feature fusion in deep learning-based object detector. Journal of Engineering, 2020, 1-11.
22. Saadat, M. N., & Shuaib, M. (2020). Advancements in deep learning theory and applications: Perspective in 2020 and beyond. Advances and Applications in Deep Learning, 3.
23. Phung, V. H., & Rhee, E. J. (2019). A high-accuracy model average ensemble of convolutional neural networks for classification of cloud image patches on small datasets. Applied Sciences, 9(21), 4500.
24. Azlah, M. A. F., Chua, L. S., Rahmad, F. R., Abdullah, F. I., & Wan Alwi, S. R. (2019). Review on techniques for plant leaf classification and recognition. Computers, 8(4), 77.
25. Cakır, S., Aytaç, T., Yıldırım, A., & Gerek, Ö. N. (2011). Classifier-based offline feature selection and evaluation for visual tracking of sea-surface and aerial targets. Optical Engineering, 50(10), 107205-107205.
26. Ahmad, J., Muhammad, K., & Baik, S. W. (2017). Data augmentation-assisted deep learning of hand-drawn partially colored sketches for visual search. PloS one, 12(8), e0183838.
27. Roboflow: Go from Raw Images to a Trained Computer Vision Model in Minutes. (n.d.), Roboflow.ai, https://roboflow.com/
28. 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).
29. Rahman, Z., Ami, A. M., & Ullah, M. A. (2020, June). A real-time wrong-way vehicle detection based on YOLO and centroid tracking. In 2020 IEEE Region 10 Symposium (TENSYMP) (pp. 916-920). IEEE.
30. Pham, M. T., Courtrai, L., Friguet, C., Lefèvre, S., & Baussard, A. (2020). YOLO-Fine: One-stage detector of small objects under various backgrounds in remote sensing images. Remote Sensing, 12(15), 2501.
31. Terven, J., Córdova-Esparza, D. M., & Romero-González, J. A. (2023). A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS. Machine Learning and Knowledge Extraction, 5(4), 1680-1716.
32. M. Contributors, (May 13, 2023) "YOLOv8 by MMYOLO." https://github.com/open-mmlab/mmyolo/tree/main/configs/yolov8
33. 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.