A Deep Learning Model for Helmet Detection and Automatic License Plate Recognition

Final Year Project Final Report

Pham Trung Hieu

Vu Van Nghiep

Instructor

Dr. Bui Van Hieu

Bachelor of Artificial Intelligence

Hoa Lac campus - FPT University

– Hanoi, 20 December 2023 –

ACKNOWLEDGEMENT

First and foremost, we would like to express my deep gratitude to my supervisor, Dr Bui Van Hieu, for the dedication and extensive knowledge guiding and supporting us throughout the implementation of this thesis. His valuable opinions and detailed instructions greatly contributed to the success of the project.

We also want to send special thanks to FPT University for its wonderful learning and research environment. The support and modern facilities at the school have facilitated my research process.

Finally, we are grateful to our friends and family, who have always been by our side, encouraging us and providing constant emotional support. Difficult moments become light thanks to their care and love.

ABSTRACT

Not wearing a helmet among motorbike drivers is one of the leading causes of fatal accidents in developing countries. Detecting individuals not wearing helmets through license plates plays an important role in monitoring, reminding and punishing violators to help reduce accidents. Current models for detecting traffic violators through license plates are facing many limitations, such as difficulty detecting multiple vehicles in one frame, and ineffective methods for identifying license plates and people in the same vehicle and cannot simultaneously perform both parts: detecting people not wearing helmets and license plate recognition. Our proposed pipeline consists of two steps. Step 1 is to identify violating vehicles using YOLOv8m. Step 2 is to extract license plate information using the PaddleOCR library. To resolve confusion between objects, we use Bytetrack in first step to track and analyze. In both steps, post-processing techniques are developed to avoid errors in the identification process. This technique will take the class that appears most frequently in the motorcyclist's box and assign them together. Through testing, the model achieved high accuracy with mAP (mean Average Precision) is 97.9% and an accurate license plate recognition rate of 90.4%. Research results show that the proposed model achieves impressive efficiency in both tasks, helping to improve traffic safety and traffic management effectively.

Keywords: *YOLOv8m, ByteTrack, PaddleOCR, helmet detection, license plate recognition.*

Table of Contents

List of abbreviations

Abbreviation	Definition
AIP	Asia Injury Prevention Foundation
ALPR	Automatic License Plate Recognition
AP	Average Precision
BB	Bounding Box
BCE	Binary Cross Entropy
BLT	Belong to
CCTV	Closed Circuit Television
CIP	Calculate intersection percentage
CIoU	Complete Intersection over Union
CNN	Convolutional Neural Network
CSV	Comma Separated Values
DB	Differentiable Binarization
DFL	Distribution Focal Loss
FID	Frame ID
FN	False negative
FP	False positive
FPN	Feature Pyramid Network
FPS	Frames per second
FV	Filtering Vehicles
GPS	Global Positioning System
IOU	Intersection over Union
MT	Metadata
OCR	Optical Character Recognition
PAN	Path Aggregation Network
PLI	Plate Information
SCP	Score Plate
SiLU	Sigmoid Linear Units

SPP	Spatial pyramid pooling
SPPF	Spatial pyramid pooling fast
SSD	Single Shot MultiBox Detector
TP	True positive
WHO	World Health Organization
YOLO	You Only Look Once

List of tables

List of figures

List of appendix

1. Introduction

1.1. Problem & Motivation

Road traffic safety is a serious issue that is posing a major challenge globally. Figures from World Health Organization (WHO) indicate that in 2018 [1], about 1.35 million people died and more than 50 million were injured in road traffic accidents worldwide. In Vietnam, in 2022, there were more than 3.2 million newly registered motorbikes, bringing the total number of motorbikes to 72.1 million [2]. Wearing a helmet properly when participating in traffic is always necessary to ensure that unfortunate incidents that occur during traffic are limited. According to the Asia Injury Prevention Foundation (AIP), wearing a helmet proper insurance helps reduce the risk of traumatic brain injury and death. This ratio is true for all ages, including children. The casualty rate can be reduced by up to 69% and the risk of death by 42% if you wear a helmet properly when driving on the road [3].

In recent years, the integration of deep learning techniques in computer vision applications has significantly advanced the field of intelligent transportation systems and security. This paper presents a comprehensive study on the development of a deep learning model designed for Helmet Detection and Automatic License Plate Recognition (ALPR). The fusion of these two crucial functionalities aims to enhance overall safety and security in various scenarios, particularly in the context of traffic monitoring and law enforcement. The primary objective of our research is to create a robust system that can simultaneously detect helmets on individuals and recognize license plates on vehicles within a given environment. Breaking down this overarching goal, we identify two key components: object detection [4] for helmet identification and Optical Character Recognition (OCR) [5] for license plate extraction. Each of these tasks poses its own set of challenges and intricacies, demanding specialized attention for effective implementation.

Object detection involves identifying and locating objects of interest within an image or a video frame. This technology plays a pivotal role in security and surveillance applications, enabling the automated analysis of visual data. The choice of an appropriate object detection method is pivotal, as it directly influences the model's accuracy, speed, and applicability to real-world scenarios. Among the array of object detection methods, the R-CNN family [6,7,8], Single Shot MultiBox Detector (SSD) [9], RetinaNet [10], and EfficientDet [11], the You Only Look Once (YOLO) architecture has outstanding advantages compared to others. It is renowned for its real-time processing capabilities. By framing object detection as a regression problem and directly predicting bounding boxes and class probabilities, YOLO achieves remarkable speed, making it suitable for applications where quick and accurate responses are crucial. YOLO excels in handling objects at various scales within an image. Its grid-based approach and the ability to predict bounding boxes across different scales make it particularly effective in scenarios where objects may vary significantly in size.

On the other hand, OCR focuses on extracting text information from images, converting it into machine-readable text. These two technologies individually

contribute significantly to the automation of various processes, but their combination introduces complexities that require careful consideration. Currently, PaddleOCR [12] and Tesseract libraries are high-precision libraries that can take advantage of the power of GPU to improve processing speed. PaddleOCR's unique feature lies in its ability to utilize GPUs to speed up the recognition process, outperforming Tesseract by 46% on a standard GPU. This not only improves processing speed but also opens the door to applications that require low latency. Another strength of PaddleOCR is its compact model size (2MB), which is only about 10% of Tesseract's English training data. This reduces the load on model deployment and storage. Combining object detection and OCR presents a unique set of challenges. The coexistence of pedestrians wearing helmets and vehicles with license plates necessitates a sophisticated model that can accurately distinguish between these distinct elements. Furthermore, the varying scales, orientations, and lighting conditions inherent in real-world scenarios amplify the difficulty of this integration. Addressing these challenges requires a holistic approach that considers the intricacies of both object detection and OCR.

One notable limitation in existing systems is the potential for misclassification or confusion between pedestrians and vehicles, leading to inaccurate results in both helmet detection and ALPR. Most studies only do helmet detection or license plate recognition separately. This paper introduces a deep learning model specifically tailored to overcome these challenges. This article proposes a system that uses a combination of YOLO and ByteTrack [13] models to detect people not wearing helmets, then uses the PaddleOCR library to extract license plate information. In the subsequent sections of this paper, we delve into the technical details of our proposed model, experimental results, and a thorough analysis of its performance in real-world scenarios. Through this research, we aim to contribute to the advancement of intelligent systems that play a crucial role in ensuring public safety and security.

1.2. Related Works

Research that has been done in the past has mostly involved detecting motorcyclists not wearing helmets. Only a few studies have incorporated license plate recognition.

Authors Rohith CA, Shilpa A Nair, Parvathi Sanil Nair et al. [14] presented a system using the Caffe model for detecting motorbikes and people, then detecting and classifying helmets, the InceptionV3 model was used. However, the results are not really good with accuracy of 86% and 74% respectively. These models were utilized in conjunction with their pre-trained weights and underwent some minor fine-tuning for optimization. The system can be confused with other objects such as a cyclist's helmet.

System proposed by authors Aditya Mandeep Vakani, Ashwin Kumar Singh, Shrey Saksena, and Vanamala H.R. et al [15] used YOLOv3 trained on the COCO dataset to recognize the Bike+Person class, head region class and license plate. Initially, they manually took the top 25% of the Bike+Person boxes and labeled this as the top box. This caused a lot of problems because there were many boxes on top of each other, especially when the bikes were very close together. They solved this problem by

creating a separate class for the head. Accuracy has increased by 25% compared to the original. Then use the ResNet [16] model to detect whether the person is wearing a helmet or not. The model recognized the bike + person class at 83%, the license plate at 90% and the person not wearing a helmet at 83.75%.

In this paper [17], YOLOv2 is used for the real-time detection of license plate problems. They drew a large frame containing the coordinates of the classes from top to bottom including helmet, person, plate. A horizontal line that acts as a reference line is drawn at 3/10 of the frame. This ensures that the system will not recognize the license plate when the driver's head exceeds the frame. Accuracy when the system identifies people not wearing helmets is 95.07%. The overall mAP of the 3 classes is 97.9%.

One of the problems that needs to be solved is not to mistakenly recognize a pedestrian or the wrong license plate number of another vehicle. Chenyang Wei, Zhao Tan, Qixiang Qing, Rong Zeng and Guilin Wen et al [18] have a new solution. They proposed a lightweight YOLOv5 model (SG-YOLOv5) to recognize hats and license plates. Once the rider's frame is determined, the algorithm will retain classes that satisfy the following condition: the center point of the helmet or no-helmet class is above 1/2 of the rider target box, the center point of the class. The license plate is located below 1/3 of the rider target box. Model SG-Yolov5 has mAP relatively similar to YOLOv5s but the number of parameters is reduced by 90.8%, Flops is reduced by 80.5%, model file size is reduced by 88.8% and FPS is 2.7 times that of YOLOv5s. However, the limitation of this problem is that the system's accuracy will decrease in complex traffic environments. The system cannot OCR the license plate.

With the hope that the model can be implemented in real-time using a Webcam or a CCTV as input, Lokesh Allamki et al [19] tested using YOLOV3 tiny with mAP of 75%. After cutting off the license plate of the violating vehicle, the OCR model will process it. The results are saved to a text file and corresponding confidence. The output is the text with the highest confidence. The accuracy of this method reaches 85%. However, this method will fail if another license plate appears in the driver's frame.

A lightweight and highly efficient multi-angle license plate recognition model was proposed by Cheng-Hung Lin and Chen-Hao Wu [20]. This paper collects a large number of license plate images from different environments, angles and sizes as training data. Angles range from 0~75 degrees. Tiny-YOLOv2 optimized the network architecture, including changing the number of filters in the 9th convolutional layer to 128, and changing the number of filters in the 11th convolutional layer to 256, and removing the 13th convolutional layer. Characters are identified separately and then combined into a complete number plate. Experimental results show that the proposed model can recognize license plates with tilts of 0~60 degrees. The recall rate is 84.5%.

With the advantage of speed and the ability to detect multi object at the same time, YOLO is used in most research instead of conventional machine learning models. So based on those advantages, we decided to use it as the main algorithm for the article.

1.3. Contribution

This paper contributes significantly to the existing body of knowledge include:

- Firstly, a novel public dataset, comprising a total of 6562 images categorized into four distinct classes-motorcyclist, helmet, no_helmet, and license plate-has been meticulously curated, with each image annotated with precise bounding boxes.

- Secondly, in addressing the inherent limitations within the realm of head area and license plate detection as identified in extant research, our approach is grounded in and enhances the detection methodology initially proposed by Chenyang Wei et al [18].

- Furthermore, a post-processing method was developed, which facilitates the discrimination and tracking of multiple objects within a single frame. This technique also increases the accuracy of extracting license plate information.

- Finally, we create a website with a friendly and easy-to-use interface so users can upload videos, run model and adjust parameters for an intuitive look.

The subsequent sections of this paper unfold in a methodical sequence. Section 2 provides a comprehensive introduction to the methodology employed in this research. Section 3 meticulously details the experimental procedures undertaken, offering insights into the outcomes. In Section 4, we expound upon the deployment of the web application derived from our findings. Finally, Section 5 encapsulates the culmination of our study, drawing essential conclusions from the overarching research framework. This organized structure ensures a systematic presentation of our contributions and aids in the assimilation of the advancements we proffer in the domain.

2. Methodology

2.1. Overview pipeline

In this section, we delineate the comprehensive workflow employed throughout the project illustrated in Figure 1. Commencing with the collection of data, a myriad of sources, including personal cameras and traffic surveillance systems, is harnessed to amass a diverse dataset. This reservoir of data is then meticulously stored on individual computing systems for subsequent processing. The subsequent phase involves the judicious application of Roboflow, a versatile platform esteemed for its expediency and efficiency in annotation, preprocessing, and augmentation tasks. Leveraging Roboflow streamlines these critical preparatory stages, ensuring the dataset is suitably refined for subsequent model training. The platform's agility in handling diverse data sources aligns seamlessly with the project's overarching objectives. For object detection, the YOLOv8 [21] architecture is implemented, capitalizing on its prowess in accurately identifying and delineating various entities within the collected images. Concurrently, PaddleOCR is enlisted for the intricate task of character recognition. This dynamic tandem of YOLOv8 and PaddleOCR embodies a synergistic approach to address the dual facets of detection and recognition, amplifying the overall efficacy of the model. To enhance the interpretability and usability of the outcomes, a post-processing stage intervenes, fine-tuning the results before they are presented to end-users. This meticulous post-processing ensures that the information relayed to users is refined, coherent, and aligned with the overarching objectives of the project. Finally, the synthesized information is presented to end-users through the incorporation of Streamlit, a platform known for its efficacy in building interactive and user-friendly data applications [22]. Streamlit acts as the conduit through which the inferences derived from the model are made accessible, thereby bridging the gap between the complex computational processes and the intuitive understanding of the results for the end-user.

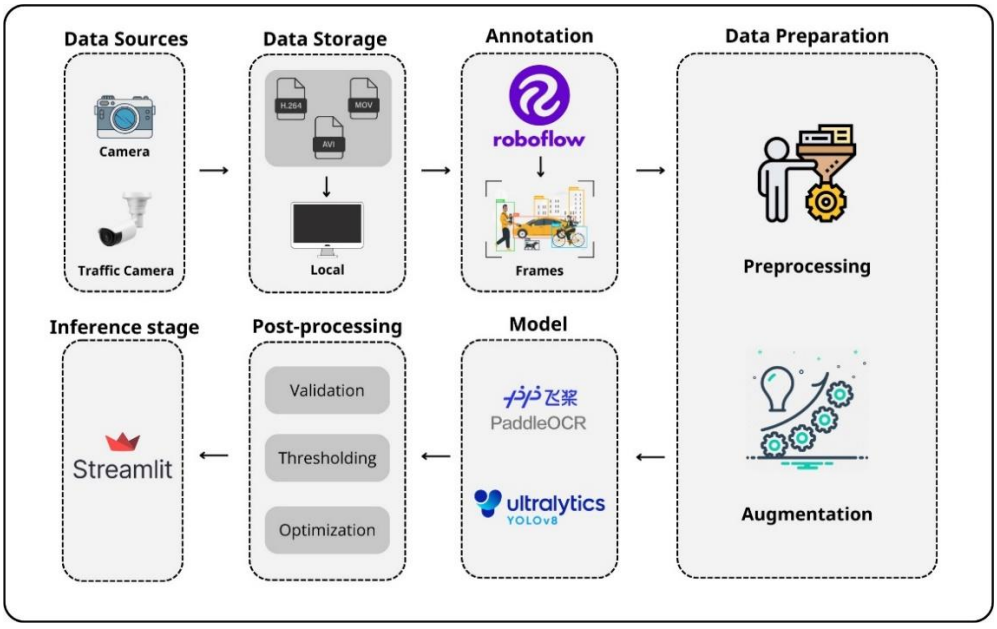


Figure 1. Workflow composer overview

Figure 2 is the proposed method to identify motorcyclists without helmets and recognize license plates in each stage. First, the input is fed into YOLOv8m to detect all trained layers. When an object is detected that is a motorbike, the system will determine whether the motorcyclist is wearing a helmet or not. If the motorcyclist is not wearing a helmet, the system will record information about the coordinates of that motorbike's license plate. In the above stage, one of the objects is not detected, the process will move to the next Frame. Next, the license plate is cut out and fed into the PaddleOCR library for character extraction. Post-processing techniques are added to improve the model's accuracy. Information and license plate images will be saved in the database for future use.

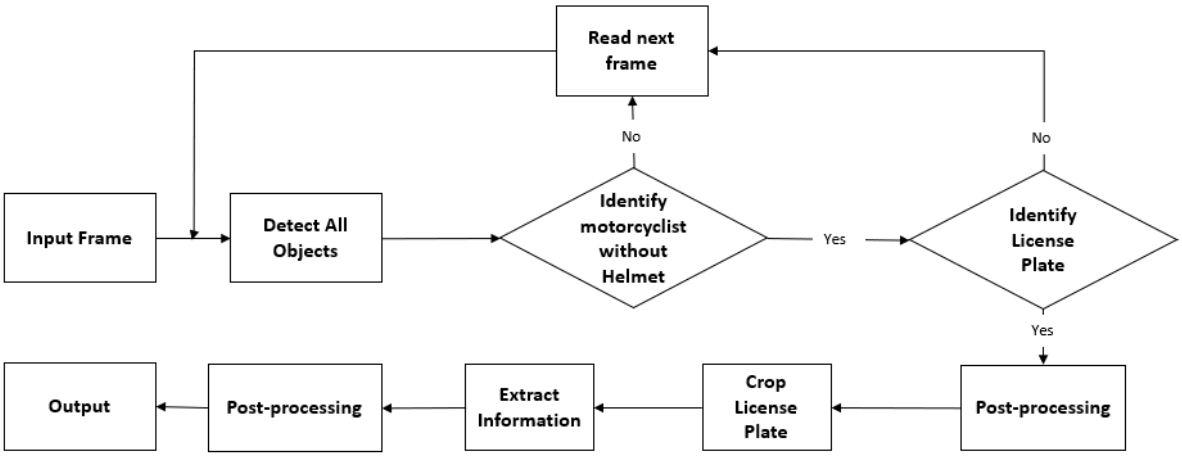


Figure 2. System architecture

2.2. Object Detection

2.2.1. YOLOv8 Architecture

YOLOv8 is a practical object detection model developed by Ultralytics. It was released in January 2023 and inherits the architecture of YOLOv5. It has five types of models, named YOLOv8n, YOLOv8s, YOLOv8m, YOLOv8l, YOLOv8x. In this article, we choose the YOLOv8m model because it balances accuracy and processing speed. YOLOv8's architecture includes three main parts: Backbone, Neck, Heads as illustrated in Figure 3.

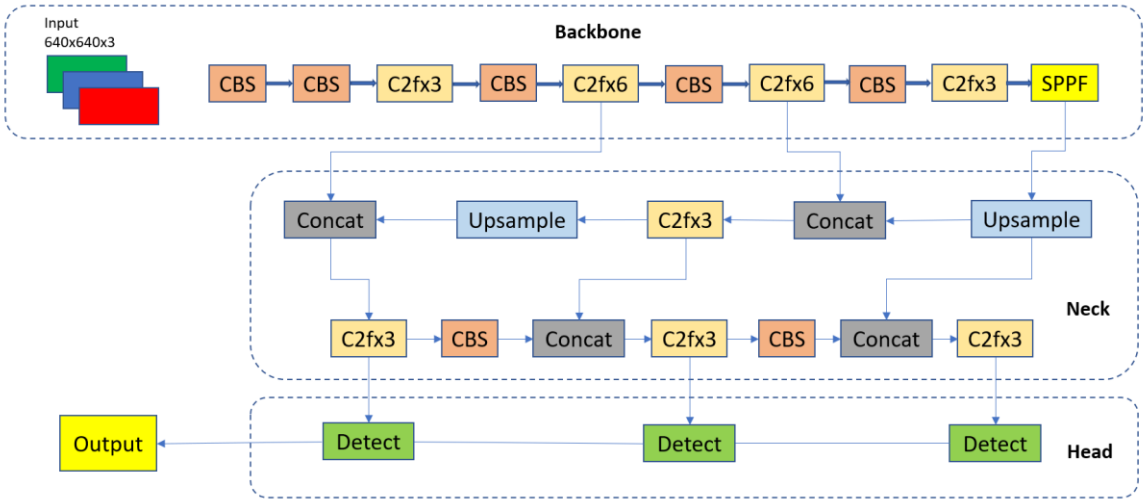


Figure 3. Network structure of YOLOv8

The Backbone is the initial image converter, responsible for converting the input image into a representation that can be used by object detection models. YOLOv8 uses the CSPDarknet53 [23] architecture as its backbone, which is a variant of the Darknet53 architecture with improved accuracy and speed. The CBS module carries out a convolution operation on the input data, then applies batch normalization, and finally activates the data stream using SiLU to obtain the resulting output. The C3 module has been substituted with the C2f module based on the CSP idea. In the C2f module, the outputs from all Bottleneck units are concatenated, diverging from C3 where only the output of the final Bottleneck was utilized. To enhance the efficiency and reduce latency, the backbone network effectively utilizes the spatial pyramid pooling fast (SPPF) module. This module facilitates pooling of input feature maps into a fixed-size map for adaptive size output. By incorporating SPPF, computational effort is significantly reduced compared to the traditional spatial pyramid pooling (SPP) structure [24]. Moreover, SPPF achieves lower latency by sequentially connecting three maximum pooling layers.

The next part, the Neck, is the connection between the backbone and the prediction outputs. YOLOv8 adopts the PAN-FPN feature fusion method, reinforcing the amalgamation and exploitation of feature layer information across different scales. Leveraging two Upsampling processes and incorporating multiple C2f modules, YOLOv8 introduces a decoupled head structure inspired by the YOLOX [25] paradigm. This innovative approach fuses confidence and regression boxes, elevating the model's accuracy to unprecedented levels.

The Heads are prediction outputs, including predictions about the location, size and class of objects in the image. YOLOv8 uses the same FPN (Feature Pyramid Network) architecture [26] as previous YOLO versions to generate prediction outputs at different levels of the image. The decoupled head structure utilizes two distinct branches to handle object classification and predicted bounding box regression. The YOLOv8 model calculates its loss using two main functions: one for classification loss and one for bounding box loss.

For classification loss, YOLOv8 employs the Binary Cross Entropy (BCE) loss, as expressed in Equation (1). The BCE loss is a standard choice for binary classification tasks and is utilized to measure the dissimilarity between predicted class probabilities and ground-truth class labels.

$$BCE = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log\big(p(y_i)\big) + (1 - y_i) \cdot \log\big(1 - p(y_i)\big) \quad (1)$$

Where:

N is the number of samples.
y_i is the ground-truth class label for sample i.
$p(y_i)$ is the predicted probability of the positive class for sample i.

For bounding box loss, YOLOv8 combines the Complete Intersection over Union (CIoU) and Distribution Focal Loss (DFL).

IoU is calculated between the predicted bounding box (N) and the ground-truth box (B^{gt}), as shown in Equation (2). It measures the overlap between the two boxes.

$$IoU = \frac{B \cap B^{gt}}{B \cup B^{gt}} \quad (2)$$

Complete Intersection over Union (CIoU):

$$CIoU = 1 - IoU + \left|\frac{\rho^2(b, b^{gt})}{c^2}\right| + \beta\upsilon \quad (3)$$

Where:

b and b^{gt} denotes the central point of the predicted bounding box (B) and the ground-truth box (B^{gt}).
$\rho(\cdot)$ denotes the Euclidean distance.
c is the diagonal length of the smallest enclosing box covering the two boxes.
β is the trade-off parameter and υ measures the consistency of the aspect ratio.
β and υ are defined as shown in Equation (5) and Equation (4) respectively.

$$\upsilon = \frac{4}{\pi^2}\left(\arctan\frac{\omega^{gt}}{h^{gt}} - \arctan\frac{\omega}{h}\right)^2 \quad (4)$$

$$\beta = \frac{\upsilon}{1 - IoU + \upsilon} \quad (5)$$

Where:

ω and h is the width and height of the bounding box.

Traditional loss functions like Cross-Entropy Loss may not be ideal for object detection tasks due to the inherent class imbalance problem, where the majority of image regions do not contain objects. Focal Loss addresses this issue by down-weighting the loss assigned to well-classified examples, focusing more on challenging instances. Distribution Focal Loss (DFL), as an extension of Focal Loss, further incorporates class distribution information into the loss function.

DFL aims to learn a dynamic weighting scheme for the loss based on the distribution of classes in the training data. The key idea is to assign more weight to under-represented classes and less weight to over-represented classes. This adaptive weighting contributes to more precise bounding box estimations.

To implement DFL, the continuous distribution of the regression value $\hat{y}$ is converted to the discrete domain using a softmax layer with $n + 1$ units, denoted as $P(\hat{y})$. The estimated regression value $\hat{y}$ is expressed using Equation (6):

$$\hat{y} = \sum_{i=0}^{n} P(y_i)\, y_i \tag{6}$$

Here, $P(y_i)$ represents the probability of the i-th unit in the softmax layer. The Distribution Focal Loss is then given by Equation (7):

$$DFL(S_i, S_{i+1}) = \left((y_{i+1} - y)\log(S_i) + (y - y_i)\log(S_{i+1})\right) \tag{7}$$

In this equation, S_i and S_{i+1} are the softmax values corresponding to the i-th and $(i + 1)$-th units, and y_i and y_{i+1} are the ground-truth values for the same units. DFL effectively leverages class distribution information to enhance the loss function, contributing to improved object detection performance.

Thus, it can be observed that the loss functions used in YOLOv8 are designed to enhance the model's performance on various datasets.

Compared to YOLOv5 versions, YOLOv8 has a number of significant improvements. First, the developers replaced the C3 module with the C2f module and the first 6x6 Conv with 3x3 Conv in the Backbone. Next, it replaces the first 1x1 Conv with 3x3 Conv in the Bottleneck. Finally, it uses a decoupled head and deletes the objectness branch.

During online training, YOLOv8 utilizes various image augmentations to enhance its performance. One such augmentation technique is known as mosaic augmentation. This technique involves combining four different images together, which helps the model learn to detect objects in new positions, even when partially occluded and against varying background pixels. However, it has been empirically demonstrated that the performance of the model can be negatively affected if mosaic augmentation is applied throughout the entire training process. As a result, YOLOv8 disabled this augmentation technique for the final ten epochs of training.

2.2.2. Helmet and License Plate Detection

Upon receiving a frame as input in the helmet detection phase, the intricacies of the deep learning model's operations unfold with meticulous precision. The initial step in this computational dance involves resizing the input frame to a standardized format of 800x800 pixels. This deliberate choice in dimensions serves not only to optimize computational efficiency but also to maintain a consistent and manageable scale for

subsequent processing. Following the resizing operation, the model adeptly undertakes the task of detecting all discernible classes (no_helmet, motorcyclist, license plate) within the frame. As the model identifies a class within the frame, the outcome is presented in the form of bounding boxes encapsulating the detected objects. Each bounding box encompasses the spatial information necessary to precisely delineate the location of the identified object. In the subsequent stage, a meticulous transformation occurs to revert these bounding box coordinates to the scale of the original input image. This conversion ensures that the spatial information accurately corresponds to the dimensions of the initial frame, facilitating a seamless integration of the detection outcomes into the context of the entire scene. The ultimate output of this intricate process is a comprehensive list encapsulating the coordinates of each detected class. The license plate will be cropped into an image to serve as input for the next part.

2.3. Optical Character Recognition

2.3.1. PaddleOCR

PaddleOCR, an impressive open source library developed by PaddlePaddle, offers the perfect solution for Optical Character Recognition (OCR). By seamlessly integrating into the PaddlePaddle ecosystem, PaddleOCR not only achieves outstanding performance but also grants users access to a wide range of advanced machine learning tools and resources. This powerful library offers a diverse range of OCR models, catering to various languages and character sets, including Latin and Chinese. It effectively addresses the challenges posed by multilingual and multi-codeset scenarios. What sets PaddleOCR apart is its adaptability, allowing it to be deployed on multiple platforms like CPUs and GPUs. This ensures consistent and reliable performance across different systems. The version used in this article is PP-OCRv4. Figure 4 depicts its working procedure.

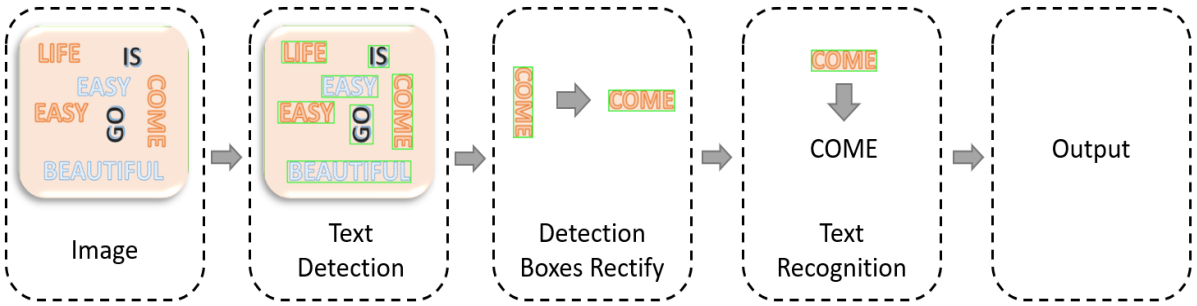


Figure 4. PaddleOCR Working Procedure

The structure of PaddleOCR includes three parts text detection, detected boxes rectification and text recognition. The function of text detection is to focus on identifying and detecting the location of areas containing text in images. The goal is to create bounding boxes around the text as a background for the next steps of the OCR process. PP-OCRv4 uses variations of the DB model, including DB-ResNet50 and DB-MobileNetV3, to optimize object detection. Pyramid Attention Network (PANet) is used to enhance attention during text container recognition. This improves the model's classification and positioning capabilities. Detection boxes rectify is

responsible for adjusting the bounding boxes of text areas after being detected. It uses Green's theory to evaluate the image as clockwise or counterclockwise and then rotates in the correct direction of the text. PP-OCRv4 recognizer is optimized based on text recognition algorithm SVTR_LCNet. It is a newly designed lightweight text recognition model. It combines two algorithms: SVTR [29] (a transformer-based algorithm) and PP-LCNet [30] (a convolution-based algorithm). The goal is to combine the advantages of accuracy and speed from both algorithms. The text recognition part is divided into two main stages. Stage 1 is detecting text areas and dividing text areas into individual characters. Stage 2 is recognizing individual characters.

2.3.2. License Plate Recognition

Through Figure 5 we can see the processing steps of the information extraction process. The journey commences with the input - an image that encapsulates the pre-cropped license plate. The next step in this part involves the astute determination of text regions within each line of the license plate. Through a judicious interplay of neural networks and convolutional layers, the model scrutinizes the segmented license plate, isolating distinct textual regions. This methodical approach ensures that the subsequent information extraction is performed with surgical precision, capturing the essence of each alphanumeric character imprinted on the plate. An inherent checkpoint emerges at this juncture: if the model, in its discernment, identifies a solitary line, the extraction process is tactfully suspended. This prudent decision-making mechanism, rooted in the model's contextual understanding, serves as a safeguard against potential inaccuracies in cases where the complete license plate information is not discernible. In instances where dual lines are detected, the model orchestrates a seamless fusion, culminating in the amalgamation of the two lines to form the complete license plate. It is then saved as a csv file for post-processing. Information will be saved as string, ID, score, date and time of violation.

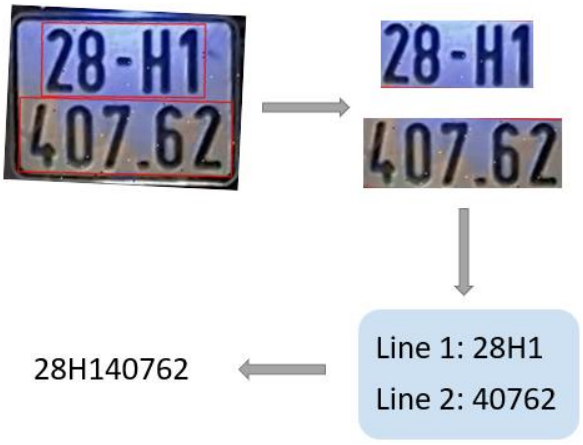


Figure 5. Extract License Plate Information

The subsequent phase unveils a meticulous character-by-character scrutiny, wherein each alphanumeric entity undergoes stringent conditions to validate its authenticity. The license plate format has four forms: NNAN-NNNNN, NNAA-NNNNN, NNAA-NNNN, NNAA-NNNN where N represents a number and A represents an alphabet

character. The third character is always a letter in the English alphabet. In case the third character is recognized as a number, we will convert it back to a letter based on similarity, for example "8" to "B", "4" to "A". The remaining characters when recognized as letters (except the letters "J" and "I"), they will also be converted into numbers based on Table 1. This detailed examination ensures that only valid characters are considered, contributing to the formation of a comprehensive and accurate license plate representation.

Table 1. Highly similar characters

Letter	Number
I	1
Z	2
J	3
A	4
S	5
G	6
B	8
D	0

2.4. Post-processing Technique

This section addresses two primary tasks: firstly, identifying license plates for each helmetless motorcyclist, ensuring a unique identification for each participant not wearing a helmet; secondly, determining the most accurate license plate for a vehicle across various frames. Based on the results obtained from the Object Detection and Optical Character Recognition phases, we stored the bounding box coordinates of three objects (no_helmet, motorcyclist, and licenseplate) representing the coordinates of the helmetless participant's head, the motorcycle's coordinates, and the license plate's coordinates of that helmetless individual, respectively.

(To better understand our working process, you can refer to the 6 algorithms in the appendix below)

2.4.1. Filtering Helmetless Motorcyclists

To address the first task, we examine a specific frame within a video. For each no_helmet object present in the current frame, we determine which motorcyclist object contains it. This is achieved using two methods: CIP (Calculate the percentage of intersection area between two bounding boxes) and HHB (Histogram of Height Box).

CIP method is explained in Figure 6. This method calculates the percentage of the intersection area between the no_helmet and the motorcyclist object by dividing the intersection area by the area of the no_helmet object.

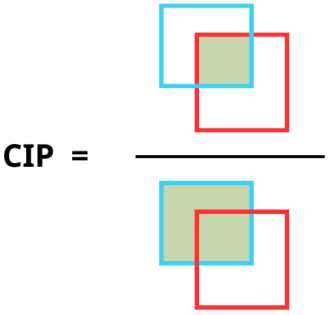


Figure 6. CIP method

HHB method is explained in Figure 7. This method assesses whether the no_helmet object lies within a reasonable height range concerning the height of the motorcyclist object.

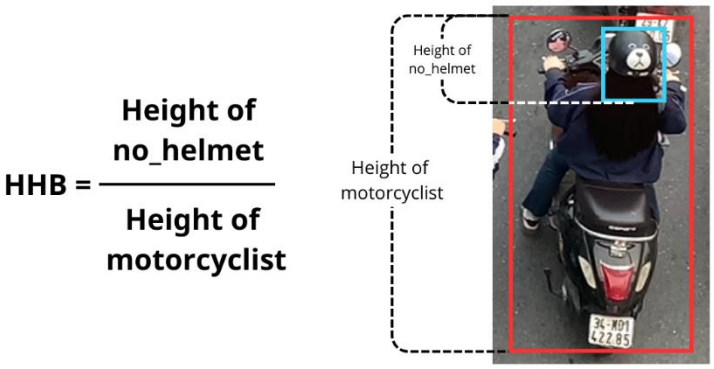


Figure 7. HHB method

In Figure 8 below, we present a histogram distribution based on 63,402 objects, depicting the relationship between the position of the no_helmet object relative to the height of the containing motorcyclist object. The histogram illustrates that 95% of the no_helmet objects fall within the range of 0.11 to 0.29 times the height of the motorcyclist object. Similarly, 95% of the licenseplate objects fall within the range of 0.64 to 0.82 times the height of the motorcyclist object.

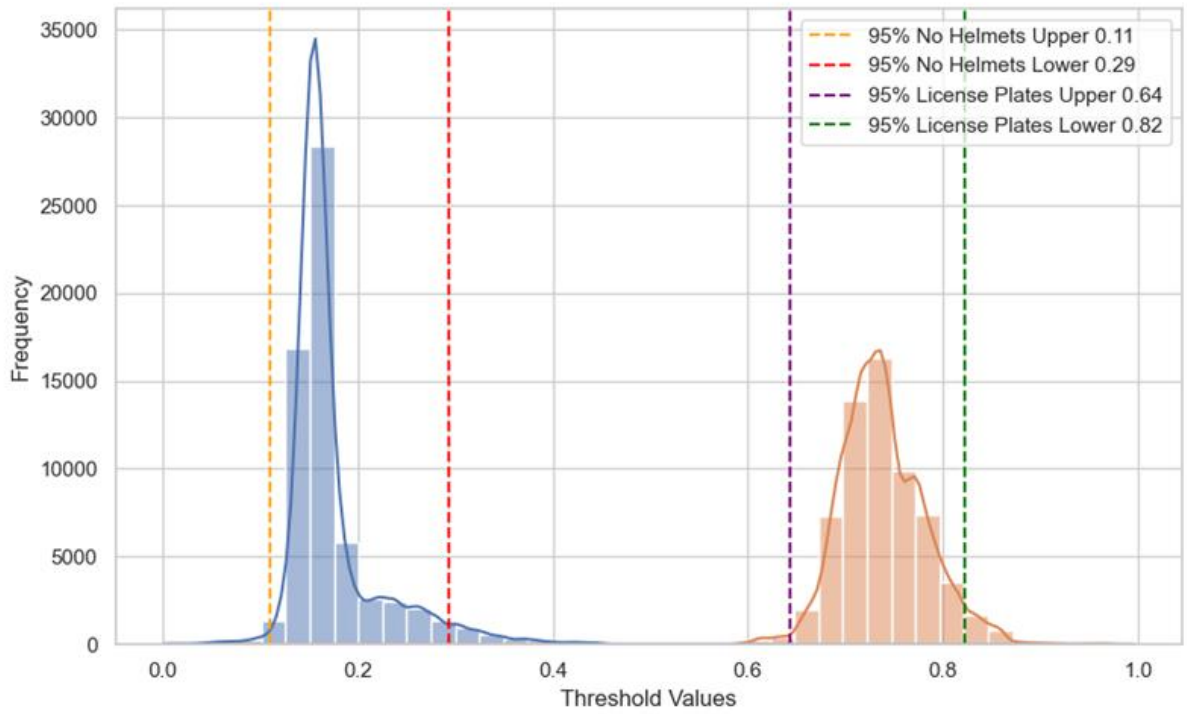


Figure 8. Histogram of threshold values for no_helmet and license plate

Consequently, we define that a no_helmet object is considered part of a motorcyclist object if it satisfies two conditions: first, CIP must be greater than 0.947, and second, the HHB of the no_helmet object must be within the range of 0.11 to 0.29 times the height of the motorcyclist object, and the license plate object must be within the range of 0.64 to 0.82 times the height of the motorcyclist object.

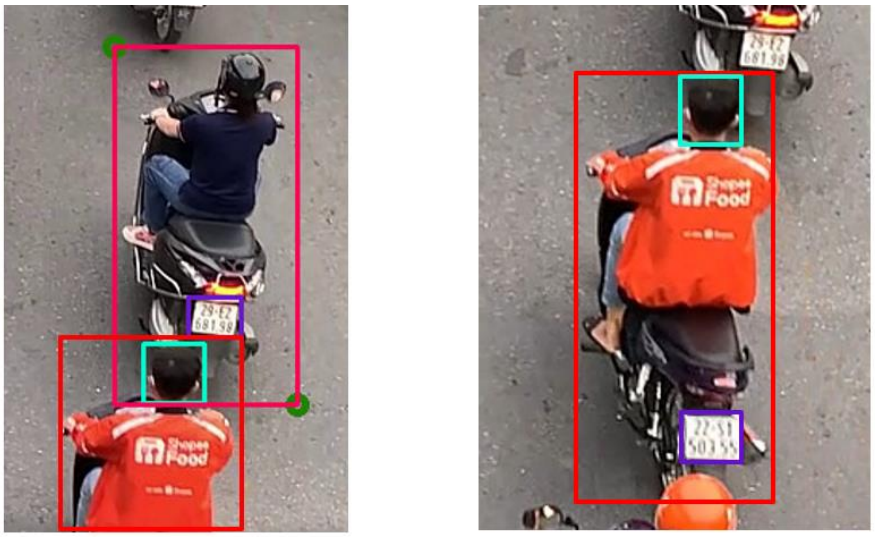


Figure 9. (a) Before applying partition. (b) After applying partition

In Figure 9, the effectiveness of applying both CIP and HHB methods is demonstrated. In Figure 9a (before processing), the no_helmet object is erroneously identified within the regions of two motorcyclist objects. After processing, as shown in Figure 9b, the no_helmet object is correctly associated with the appropriate motorcyclist object.

In practice, exceptions may still arise when a no_helmet object satisfies both conditions of the CIP and HHB methods, potentially associating with multiple motorcyclist objects, as illustrated in Figure 10 below.

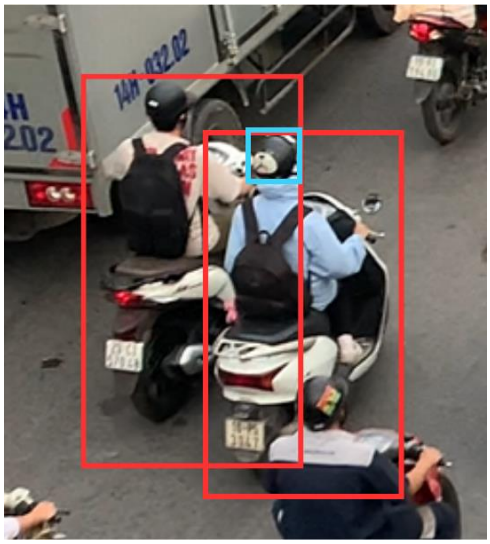


Figure 10. The head area is located in both boxes of the two motorbikes

To address this issue, we propose a method named Filtering Vehicles (FV). For each frame in the video, if a no_helmet object is associated with multiple motorcyclist objects, the frequency of each motorcyclist containing the no_helmet object under consideration is incremented by 1.

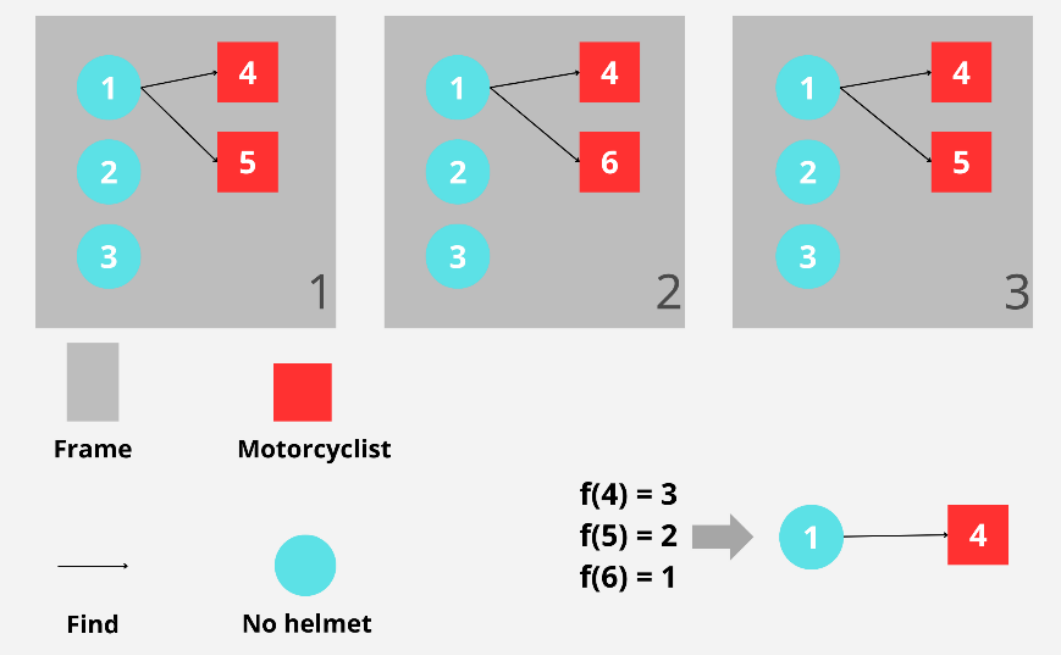


Figure 11. Post-processing technique

In Figure 11, assuming the total number of frames in the video is 3, taking no_helmet with ID 1 as an example, if no_helmet 1 appears in all 3 frames, we evaluate the associations in each frame. In the first frame, no_helmet 1 is found with motorcyclist 4 and motorcyclist 5. After the first frame, the frequency of motorcyclist 4 containing

no_helmet 1 increases to 1, and the frequency of motorcyclist 5 containing no_helmet 1 also increases to 1. In the second frame, no_helmet 1 is found with motorcyclist 4 and motorcyclist 6, resulting in an increase in frequency for motorcyclist 4 to 2 and motorcyclist 6 to 1. In the third frame, no_helmet 1 is found with motorcyclist 4 and motorcyclist 5. The frequency for motorcyclist 4 increases to 3, and the frequency for motorcyclist 5 increases to 2. After analyzing all frames, the highest frequency is with motorcyclist 4. At this point, we associate no_helmet 1 with motorcyclist 4 and eliminate all other motorcyclist associations. A similar process is applied to find the license plate from the motorcyclist.

2.4.2. Determining the Most Accurate License Plate Information

The second task in post-processing is to find the most accurate license plate information across frames. As shown in Figure 12, after extracting license plate information, it becomes apparent that for a given license plate, different results can be obtained for each frame.

Frame	License plate line top	License plate line bottom	Score line top	Score line bottom
1	35B2	**47480**	0.82	**0.87**
2	**35B2**	47430	**0.90**	0.73
3	35BZ	47480	0.67	0.81

Final output: 35B2-47480

Figure 12. Select the line with highest score

Because PaddleOCR recognizes one line at a time, we evaluate all frames where this license plate appears, select the frame where each line has the highest OCR reliability, and combine them together. Then, assign that license plate information to all other frames.

2.5. Evaluation metric

In scientific articles about object detection, evaluating model performance is an important part of ensuring the reliability of the research. The count for correct predictions on positive samples is referred to as TP, while the count for incorrect predictions on positive samples is known as FP. Similarly, the count for incorrect predictions on negative samples is referred to as FN. Popular evaluation methods include mAP, Precision (P) and Recall (R).

Precision measures the proportion of correct predictions compared to the proportion of all positive predictions. It helps evaluate the model's accuracy in predicting objects.

$$P = \frac{TP}{TP + FP} \tag{8}$$

Recall measures the ratio of correct predictions to the ratio of all positive actuals. It evaluates the model's ability to miss objects.

$$R = \frac{TP}{TP + FN} \tag{9}$$

mAP is an overall assessment method, calculated by drawing a Precision-Recall curve for each class of objects, then calculating the area under the curve (AP) for each class and averaging them. A high mAP refers to the model's ability to detect and classify objects. mAP0.5 represents the mAP when the IOU threshold is 0.5, and mAP0.5:0.95 represents the average mAP at different IOU thresholds (from 0.5 to 0.95, step 0.05).

$$AP = \int_0^1 P(R) \tag{10}$$

$$mAP = \frac{1}{n}\sum_{i=1}^{n} AP_i \tag{11}$$

3. Experiments

3.1. Data collection

The dataset has 6562 images, most were filmed on cameras with a resolution of 2,560 x 1,440. Because the public dataset is not available, our dataset was collected on many streets in Hanoi, Vietnam. To be able to extract license plate information, the viewing angle is from behind the vehicle and the standing position is on an overpass with the same height as traffic cameras. In addition, data on motorcyclists not wearing helmets is collected from many different sources. Some original photos are shown in Figure 13. Data is collected in many different traffic situations such as high traffic density, deserted roads, camera angles from left to right and right to left. By using OpenCV the video frames are converted into frames using video capture with fps = 1s. The collected raw images are carefully processed such as removing duplicate images, cleaning data, removing images without vehicles, resizing to 800x800 etc.

Figure 13. Examples in the database

The images are labeled with 4 classes: motorcyclist, helmet, no_helmet, license plate using a labeling tool. Specifically, motorcyclist refers to both the driver and the motorbike, helmet refers to the head of a person wearing a helmet, no_helmet refers to the head of a person without a helmet, license plate refers to the variable number of the motorbike. Through Figure 14, the distribution of motorcyclist and no_helmet is more even than the other two classes. Objects will be concentrated in green areas and gradually sparse in blue areas. The augmentation methods are Brightness: between -10% and +10%, Blur: up to 0.25px and Rotation: between -5° and +5°. The dataset is divided in the ratio as shown in table 2.

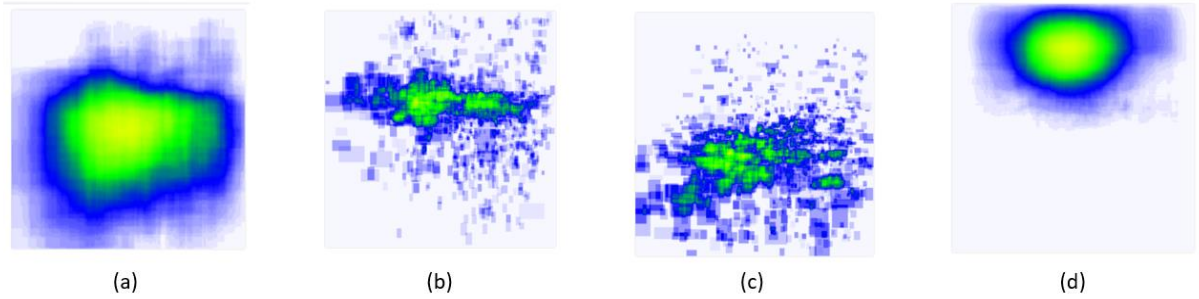


Figure 14. Annotation Heatmap. (a) motorcyclist (b) helmet (c) license plate (d) no_helmet

Table 2. Data splitting

	Image	**Ratio**
Train	5248	80%
Validation	984	15%
Test	330	5%
Total	**6562**	**100%**

3.2. Implementation

The dataset was trained on Kaggle Notebook. Kaggle Notebook is an online GPU providing Tesla P100-PCIE-16GB, CPU is Intel Xeon E-2300 series with 29 GB Ram. The specific experimental configuration is shown in Table 3. During training of YOLOv8m the model was iterated for 120 epochs, batch size was 16, input image shape was (1, 3, 800, 800), learning rate was 0.01 and weight decay was 0.001.

Table 3. Parameter of Environment

Parameter	**Configuration**
GPU	Tesla P100-PCIE-16GB
CPU	Intel Xeon E-2300
Framework	Kaggle Notebooks
Language	Python 3.10

3.3. Result & Analysis

3.3.1. Helmet and License Plate Detection

Initially, when the dataset included images we collected ourselves using the camera, the results were not very positive, mAP only reached about 78% due to data imbalance. The data was then consolidated with a data set containing only the no_helmet class and the results were significantly improved. Specifically, Precision reached 94.9%, Recall reached 95.3% and mAP was 97.9% shown in Figure 15.

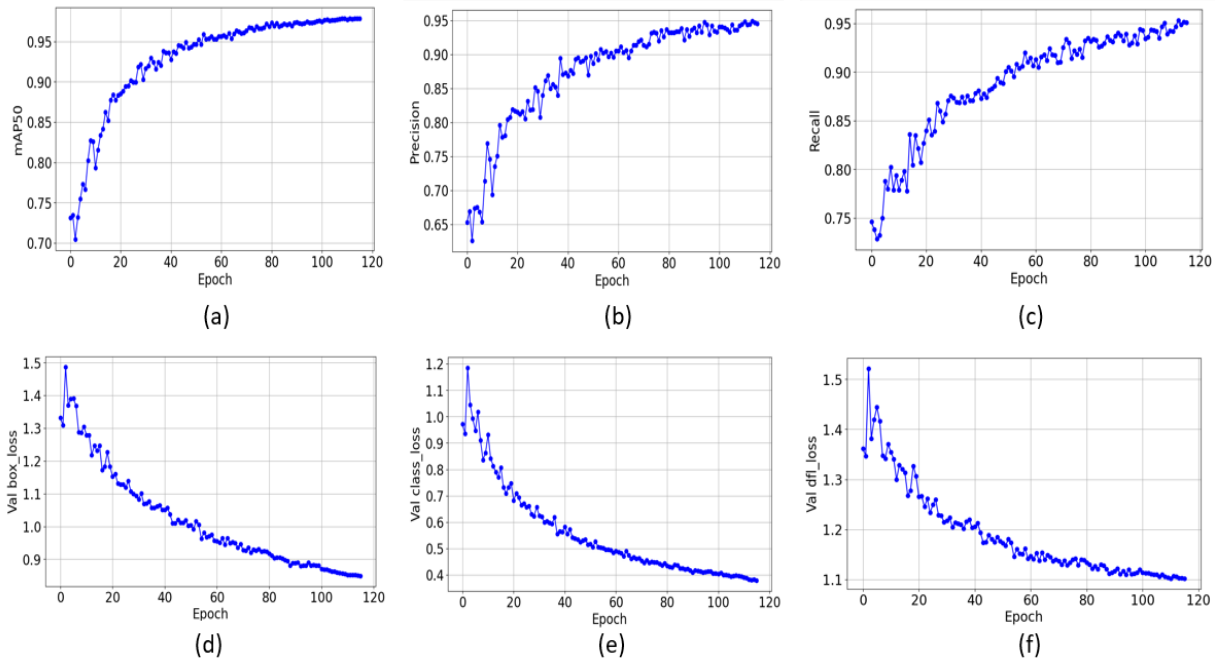


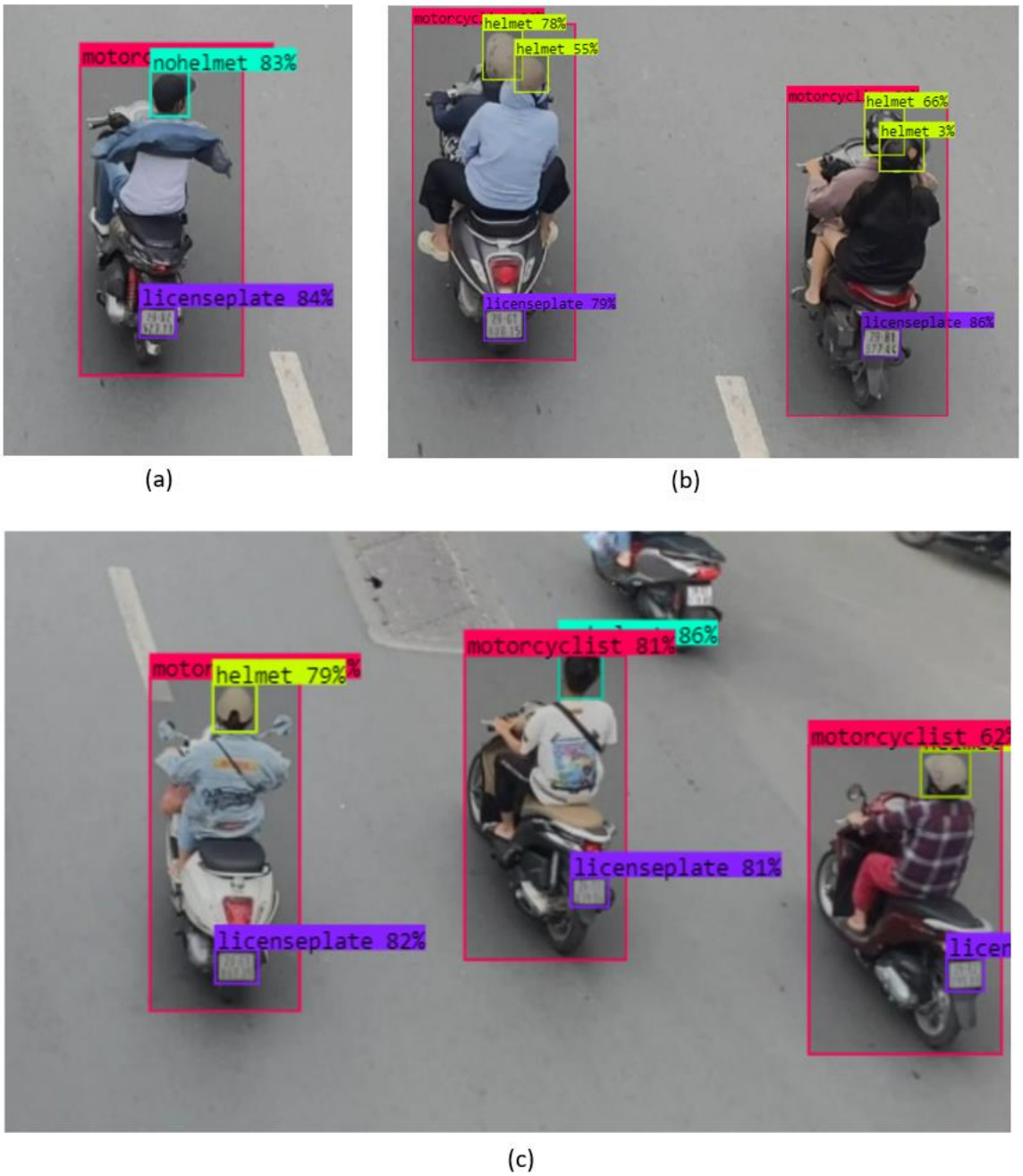
Figure 15. Training result (a). mAP (b). Precision (c). Recall (d). Val box_loss (e). Val class_loss (f). Val dfl_loss

The training results of the model have improved significantly over 120 epochs. mAP at an Intersection Over Union (IoU) threshold of 50% is an important metric, especially in object detection tasks. The increase from 0.7 to 0.97 shows that the model has become more proficient at accurately identifying and delineating objects in images. Precision increased from 0.6 to 0.95, which is a significant improvement and indicates that the model is capable of reducing false positives. The increase in recall from 0.7 to 0.95 is another positive sign. Recall measures the model's ability to capture all instances of a class, and this increase indicates that the model has become more effective at detecting all instances of no helmets, helmets, license plates and motorcyclists. Table 4 shows detailed performance of models in each class.

Table 4. Result of each class

Class	Precision	Recall	mAP50	mAP50-95
helmet	0.942	0.93	0.969	0.716
no_helmet	0.946	0.932	0.973	0.694
motorcyclist	0.959	0.993	0.991	0.938
license plate	0.936	0.949	0.978	0.744

Some test images with bounding boxes are shown in Figure 16, the red box represents motorcyclist, the purple box represents license plate, the yellow box represents helmet and the green box represents no_helmet. The model can recognize multiple vehicles simultaneously as shown in Figure 16c or 16b. The model can identify people wearing a flat cap even though it looks very similar to a helmet with the naked eye as shown in Figure 16a. However, if Confidence Threshold and Overlap Threshold are both set to 0.5, then in Figure 11b there will be 1 helmet class lost because the rate is only 3%. This is also one of the limitations that may appear in some frames that require multiple object detection. Overall, as seen in table 5, the model has high performance with notable accuracy for all object types with an accuracy above 97%.



(a) (b)

(c)

Figure 16. Predicted image in the testset

To demonstrate the effectiveness of our new method, we tested on 300 separate objects on roads with high traffic density. With the old method, there were only 246 objects that correctly linked the license plate and head area, achieving 82% accuracy. When applying post-processing techniques, the number of correctly linked objects increased to 295, reaching an accuracy of 98.3%. Thus, the accuracy has improved up to 16.3%.

Table 5. Detection rate of each class

Class	Total	Detected	Accuracy
helmet	247	241	97,6%
no_helmet	362	359	99,2%
motorcyclist	289	283	97,9%
license plate	273	268	98,2%

3.3.2. License Plate Recognition

After having the photo of the license plate cut out, PaddleOCR was used to extract the information. This library provides a powerful and flexible set of OCR models that can recognize low-resolution characters. In Figure 17a, the letter G was misidentified as the number 6 and has been corrected. However, in Figure 17b the character "2" is mistakenly recognized as "3" and "6" as "0" due to the image being flashed. This is also one of the problems that are easily encountered causing errors in the OCR process, in addition to models that recognize missing characters, blurred license plates or completely peeling paint.

Figure 17. OCR result

Figure 18 illustrates the character accuracy results using PaddleOCR and our proposed method. Most of the characters have been improved significantly. Characters like “R”, “U”, “Z” appear very rarely, so currently we do not see a significant distribution. This will be work for us to improve in the future. The empirical validation of our proposed method involved a meticulous experimentation process, incorporating a dataset comprising 2383 images. PaddleOCR, serving as the baseline, exhibited a commendable accuracy of 69.8%. However, the implementation of our devised post-processing technique resulted in a noteworthy performance elevation, yielding an accuracy of 90.4%. This signifies a substantial improvement of 20.6%, affirming the efficacy of the proposed method in enhancing the precision and discriminatory power of character recognition within the experimental scope.

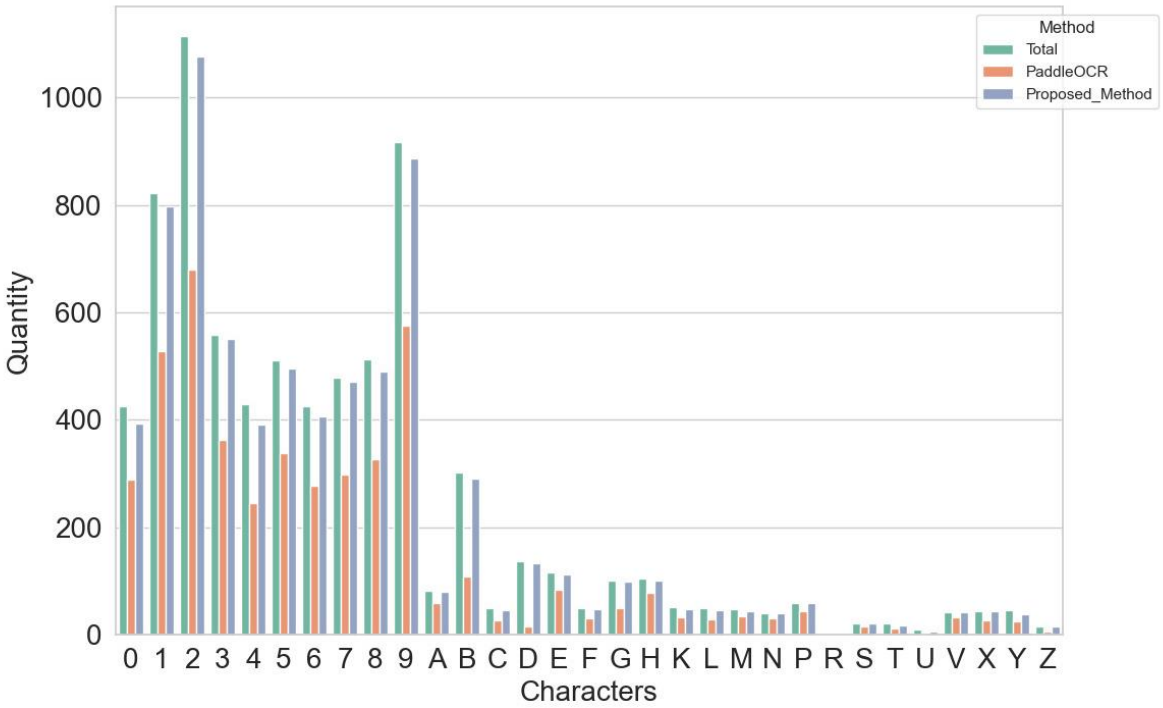


Figure 18. OCR accuracy comparison graph

In the pursuit of evaluating the robustness and efficacy of our deep learning model a rigorous experimentation phase was conducted on a dataset comprising 220 dynamically moving motorcycles traversing urban roadways. The outcomes of this empirical investigation offer valuable insights into the model's performance under real-world conditions. Our model showcased a commendable level of accuracy, successfully identifying 208 out of the 220 observed two-wheeled vehicles. This equates to an overall precision rate of approximately 94.55%, signifying a notable capability in discerning and categorizing the target objects amidst dynamic and complex traffic scenarios. However, it is imperative to acknowledge the instances where the model exhibited misclassifications. Noteworthy cases of misclassification encompassed the erroneous categorization hat of jackets as helmet. Additionally, misrecognitions of characters on license plates were observed, contributing to inaccuracies in the license plate recognition component. Further challenges were encountered when portions of license plates were obscured, resulting in misidentifications due to partial information. Instances where the license plate visibility was compromised, leading to the model's inability to extract the alphanumeric information accurately, were also noted. Notably, blurriness in some images emerged as a hindrance, impeding the successful extraction of license plate details.

4. Web Application Deployment

4.1. Introduction

In the context of escalating concerns regarding traffic security, developing an efficient system for monitoring and managing road users has become a critical need. The project "A Deep Learning Model for Helmet Detection and Automatic License Plate Recognition" addresses this need by employing Deep Learning applications to detect the absence of helmets and extract license plate information from traffic video data. To optimize the application of the project, we have decided to deploy the model through a web application, featuring an intuitive and user-friendly interface.

The primary goal of the Web Application Deployment section is to create a simple yet powerful web application, providing a convenient experience for traffic law enforcement officers to receive video data from various traffic cameras. This facilitates the easy management, analysis, and storage of information about road users, making traffic security management more efficient.

4.2. Web Application Deployment

4.2.1. System Architecture

Firstly, we constructed a flexible and scalable system architecture, enabling the application to process multiple video streams simultaneously from diverse sources. Our system is described below Figure 19. It comprises key components such as Frontend, Backend, and a crucial part, Model Inference Engine.

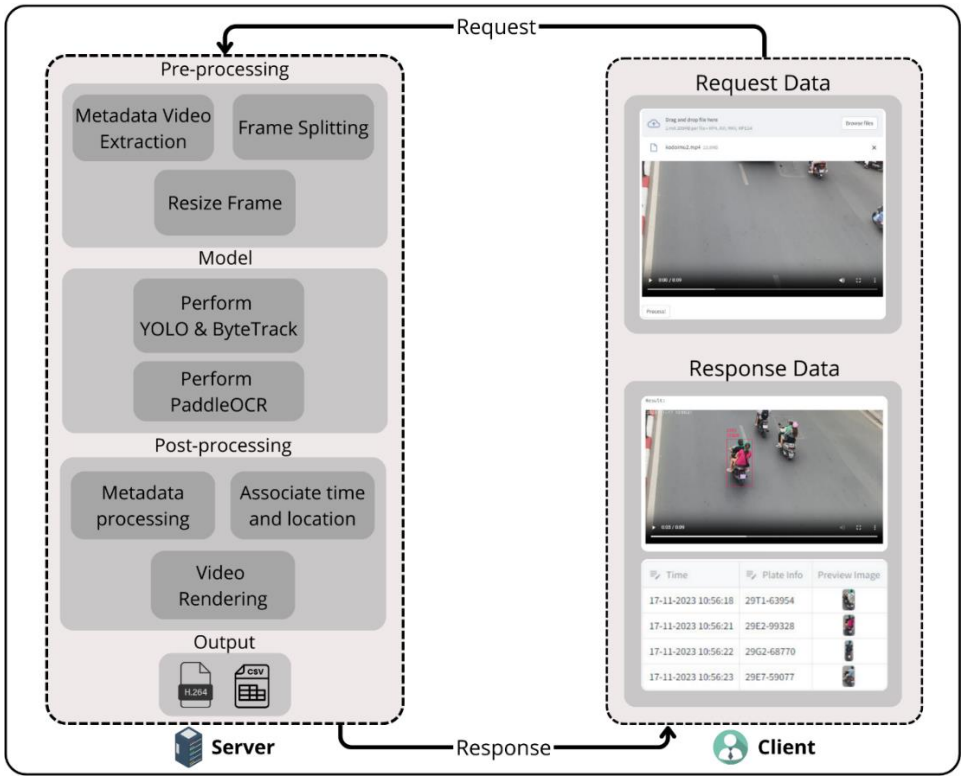


Figure 19. Request and Response Architecture

4.2.2 Frontend

We utilized the Streamlit library to develop the user interface. Streamlit not only delivers a visual experience but also aids in quickly deploying and experimenting with new features. The interface is straightforward yet powerful, facilitating efficient user interaction.

4.2.3 Backend

The backend is implemented in Python without the use of specific libraries, focusing on simplicity and efficiency. It handles requests from the frontend, interacts with the Model Inference Engine, and manages the storage of data from traffic videos.

4.2.4 Model Inference Engine

We integrated the Deep Learning model developed from the core project into the system. This model is responsible for analyzing videos to detect individuals without helmets and extracting license plate information.

When a user initiates a request to process a video, the video is securely uploaded to the server for pre-processing. On the server side, upon receiving the video, various steps are executed as part of the pre-processing phase:

Pre-processing:

- Metadata Video Extraction: This involves extracting essential information from the video such as recording time, location, frames per second (FPS), and more.
- Frame Splitting: The video is split into individual frames in order to prepare them for further processing by the model.
- Resize Frame: The size of each frame is adjusted to an appropriate dimension that is suitable for accurate prediction.

Model:

- Perform YOLO & ByteTrack: Predict and track objects through each frame.
- Perform PaddleOCR: Extract valuable information pertaining to vehicle license plates.

Post-processing:

- Metadata processing: Identify individuals not wearing helmets by associating metadata with traffic participants and identifying license plates attached to corresponding vehicles.
- Associate time and location: Assign time and location data to traffic participants who violate regulations.
- Video Rendering: Create a new video with bounding boxes drawn.

Output:

- The server returns 2 files to the user: a video with additional bounding boxes and a CSV file containing information about the violation including time of violation, license plate number, and image of the violation.

4.2.5 User Interface

The web application is designed to meet the needs of traffic law enforcement officers and traffic management units. Users can directly interact with the system, view video analysis results, and conveniently manage data.

4.3. HDALPR System

The web application is designed with an intuitive and user-friendly interface using Streamlit, providing seamless navigation through four main tabs: Home, Upload, Tasks, and Settings. Each tab serves a distinct purpose in facilitating the user's interaction with the system.

4.3.1 Home Tab

The Home tab serves (Figure 20) as the default landing page, featuring a project introduction. Users are welcomed with an overview of the project's mission and goals upon visiting the website.

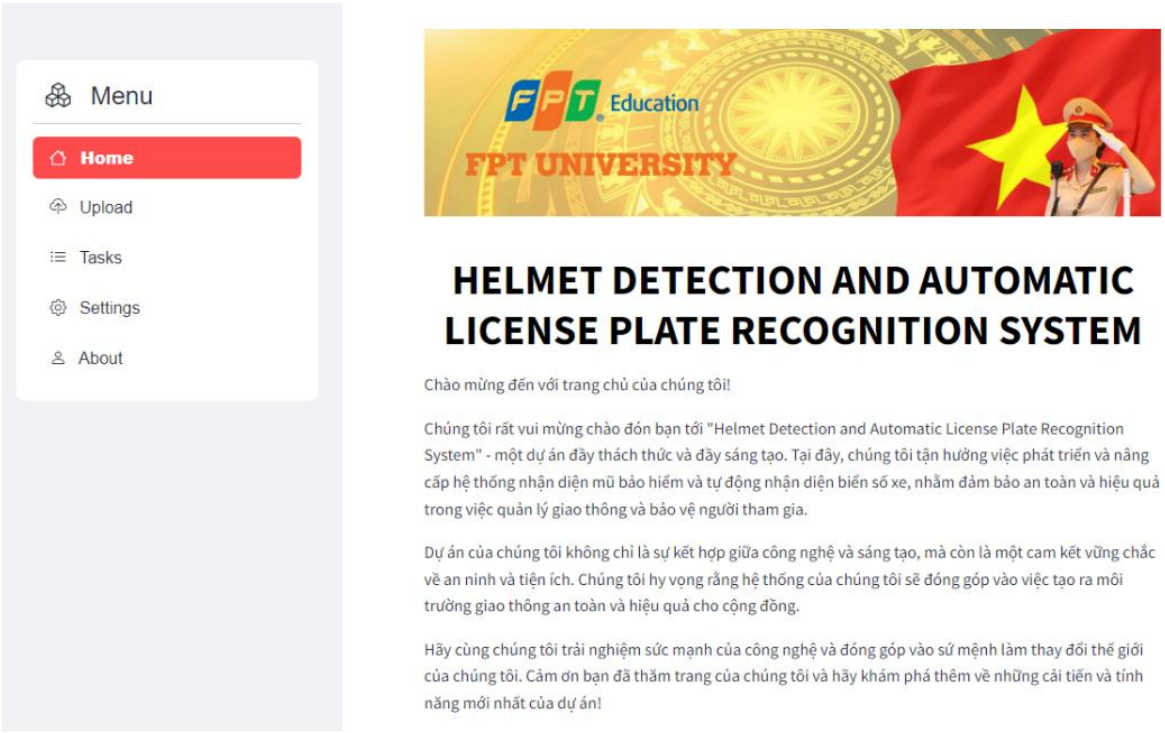


Figure 20. Home Page

4.3.2 Upload Tab

Video Processing:

In the Upload tab, users can upload videos from various sources (Figure 21), such as cameras, via the "Browse files" button. After uploading, the "Process!" button initiates video processing.

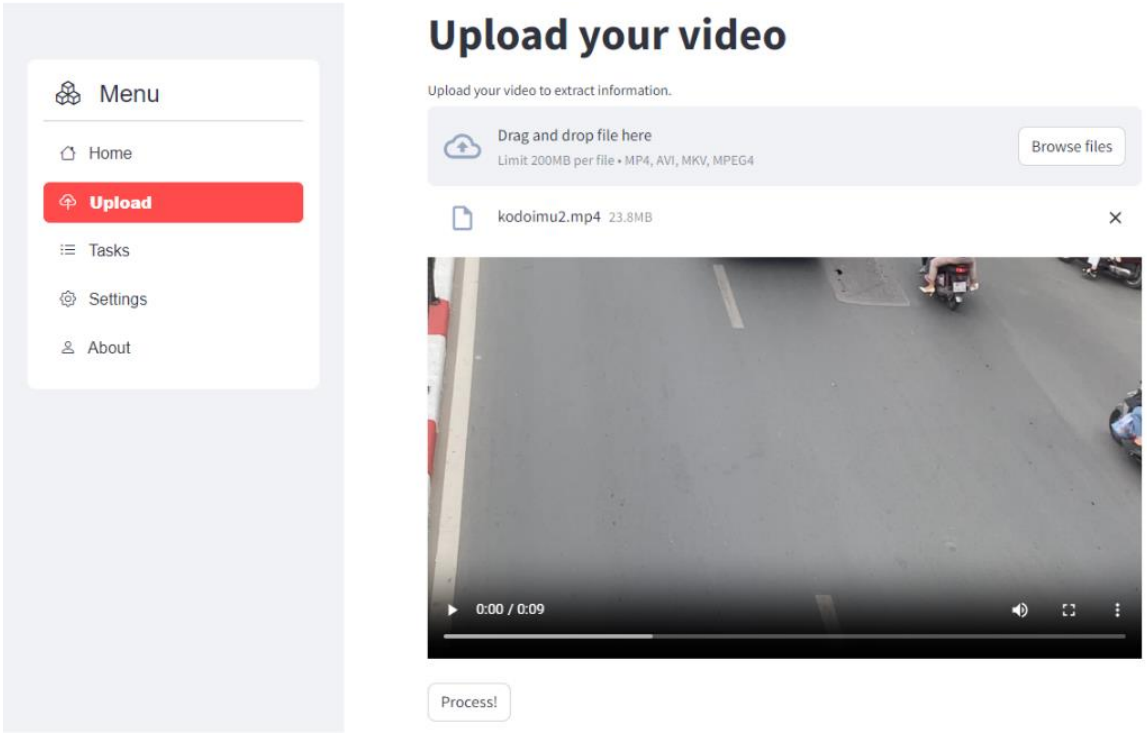


Figure 21. Upload Page

Result Presentation:

Once completed, a new video displays with bounding boxes around individuals not wearing helmets, along with their extracted license plate information. Below the processed video, two tables provide detailed information (Figure 22). The "Moto with License" table includes details of individuals without helmets and successfully extracted license plates. The "Moto without License" table lists individuals without helmets, where license plate extraction was unsuccessful. Both tables contain three fields: Time (time of the violation), Plate Info (license plate), and Preview Image (image of the violator).

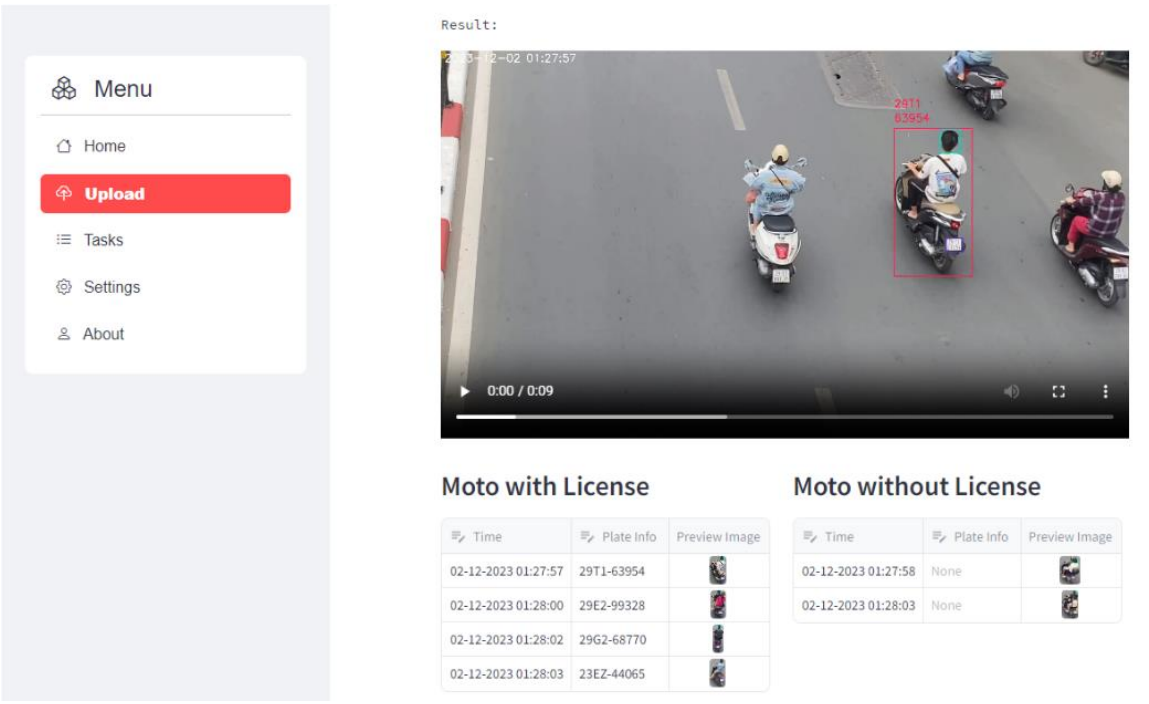


Figure 22. Results after video processing

4.3.3 Tasks Tab

Video History:

The Tasks tab, allows users to review processed videos and stored information in the database (Figure 23). The initial table displays five fields: Preview Image (preview image of the processed video), File Name (name of the processed file), Time (time when the video was created), Number of Violators (total violators in the video), and GPS (location information if available).

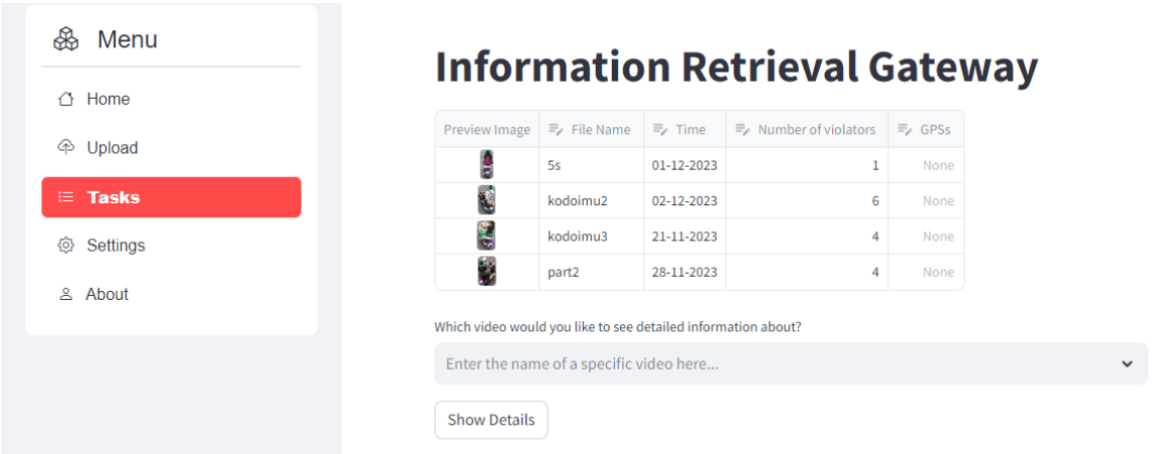


Figure 23. Tasks Page

Search Functionality:

A search bar beneath the table enables users to search for previously processed videos by their "File Name." Upon searching, the screen displays the video with bounding boxes and violator information, similar to the Upload Tab after processing (Figure 24).

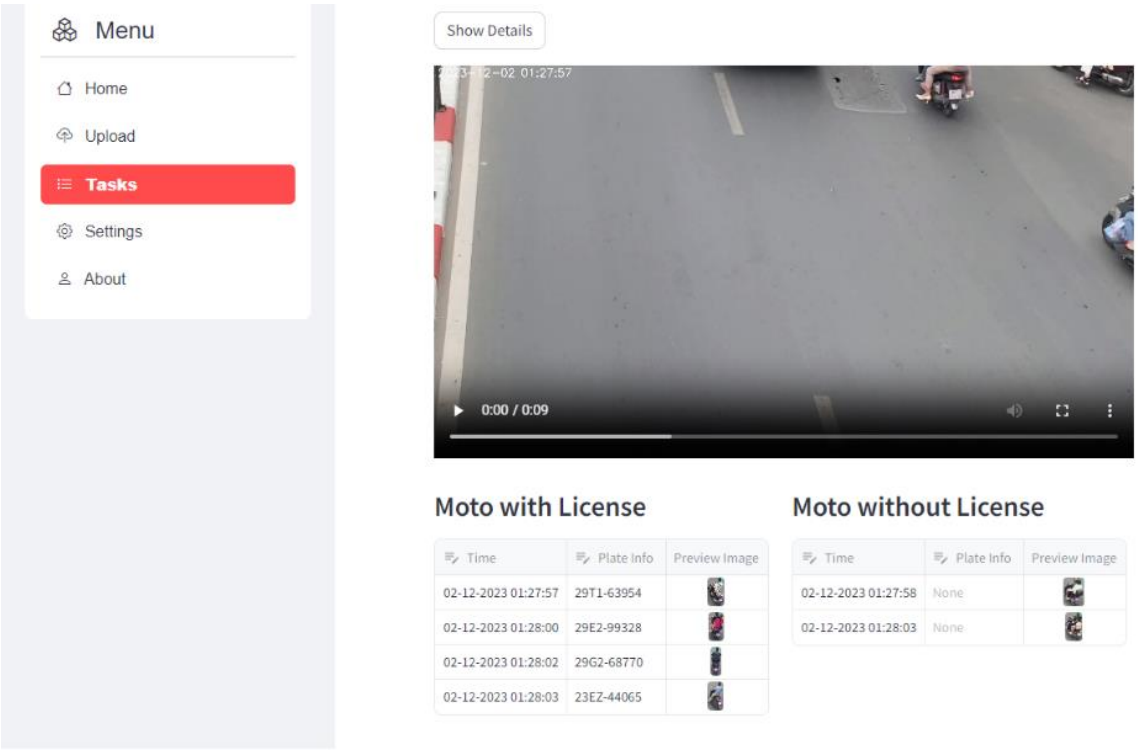


Figure 24. Data after being retrieved from the database

4.3.4 Settings Tab

Model Inference Configuration:

In the Settings tab, users can customize model inference parameters (Figure 25). Three adjustable parameters include Confidence (confidence threshold), Overlap (bounding box overlap threshold), and Device (choose between CPU or GPU for inference).

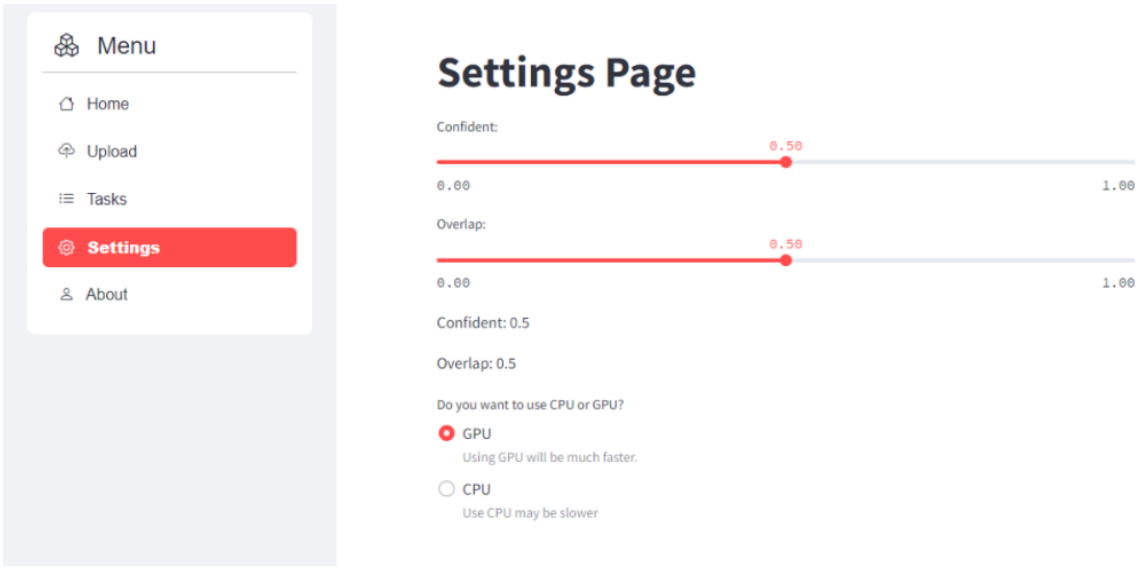


Figure 25. Settings Page

5. Conclusion and Future Work

The significance of addressing the dual challenge of helmet detection and automatic license plate recognition through a deep learning model cannot be overstated. The pervasive use of motorcycles as a primary mode of transportation underscores the urgent need for robust safety measures. In the pursuit of developing an effective deep learning model, our comprehensive approach involved a series of meticulous tasks that have significantly contributed to the success of our endeavor. One notable achievement was the creation of a robust dataset comprising 6562 images, each meticulously annotated with bounding boxes. This dataset served as the cornerstone of our model training, ensuring its capacity to generalize and perform reliably across diverse real-world scenarios. Furthermore, our dedication to meticulous research and analysis led us to carefully select a model that is a combination of YOLOv8 and PaddleOCR for good recognition results. This involved an in-depth exploration of various architectures, training techniques, and fine-tuning strategies to tailor the model to the nuances of helmet detection and license plate recognition. Recognizing the critical importance of post-processing techniques, we applied advanced methods to enhance the accuracy of vehicle and license plate identification, particularly in scenarios involving potential violations. This strategic application of post-processing not only refined the model's performance but also played a pivotal role in streamlining the identification process, reducing the likelihood of misclassification. In addition, we also created a friendly and easy-to-use website so users can upload files and adjust model parameters. In the initial phase of our study, the deployed model exhibits commendable proficiency in the recognition of both helmets and license plates, achieving a remarkable mAP score of 97.9%. Subsequently, our investigation delves into the utilization of the PaddleOCR optical character recognition tool in the latter segment of this scholarly discourse. This tool proves instrumental in the extraction of alphanumeric characters imprinted on license plates. Empirical findings substantiate the model's adeptness in extracting license plate information, manifesting an impressive accuracy rate of 90.4%.

As we chart the trajectory for future research endeavors, our aspirations extend towards the broadening of the problem scope. Specifically, we aim to enhance the model's capacity to discern helmets of suboptimal quality, thereby fortifying its utility in addressing safety concerns comprehensively. Additionally, we envision augmenting the model's detection capabilities across diverse meteorological conditions, ensuring its robust performance under varying weather scenarios.

References

1. WHO, Global Status Report on Road Safety 2018;[Online], https://www.who.int/publications/i/item/9789241565684
2. https://vov.vn/xa-hoi/xu-phat-gan-4200-ty-dong-vi-pham-trat-tu-an-toan-giao-thong-trong-nam-2022-post992295.vov
3. https://www.aip-foundation.org/what-we-do/where-we-work/
4. Felzenszwalb, Pedro F., et al. "Object detection with discriminatively trained part-based models." IEEE transactions on pattern analysis and machine intelligence 32.9 (2009): 1627-1645.
5. Jana, Ranjan, Amrita Roy Chowdhury, and Mazharul Islam. "Optical character recognition from text image." International Journal of Computer Applications Technology and Research 3.4 (2014): 240-244.
6. Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.
7. R. Girshick, "Fast R-CNN," 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 2015, pp. 1440-1448, doi: 10.1109/ICCV.2015.169.
8. Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks. arXiv 2015." arXiv preprint arXiv:1506.01497 (2015).
9. Liu, Wei, et al. "Ssd: Single shot multibox detector." Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14. Springer International Publishing, 2016.
10. Lin, Tsung-Yi, et al. "Focal loss for dense object detection." Proceedings of the IEEE international conference on computer vision. 2017.
11. Tan, Mingxing, Ruoming Pang, and Quoc V. Le. "Efficientdet: Scalable and efficient object detection." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020.
12. https://github.com/PaddlePaddle/PaddleOCR
13. Zhang, Yifu, et al. "Bytetrack: Multi-object tracking by associating every detection box." European Conference on Computer Vision. Cham: Springer Nature Switzerland, 2022.
14. C. A. Rohith, S. A. Nair, P. S. Nair, S. Alphonsa and N. P. John, "An Efficient Helmet Detection for MVD using Deep learning," 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 2019, pp. 282-286, doi: 10.1109/ICOEI.2019.8862543.
15. A. M. Vakani, A. Kumar Singh, S. Saksena and V. H. R., "Automatic License Plate Recognition of Bikers with No Helmets," 2020 IEEE 17th India Council International Conference (INDICON), New Delhi, India, 2020, pp. 1-5, doi: 10.1109/INDICON49873.2020.9342598.
16. He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
17. Jamtsho, Yonten, Panomkhawn Riyamongkol, and Rattapoom Waranusast. "Real-time license plate detection for non-helmeted motorcyclist using YOLO."

Ict Express 7.1 (2021): 104-109.
18. Wei, Chenyang, Zhao Tan, Qixiang Qing, Rong Zeng, and Guilin Wen. 2023. "Fast Helmet and License Plate Detection Based on Lightweight YOLOv5" Sensors 23, no. 9: 4335. https://doi.org/10.3390/s23094335.
19. Allamki, Lokesh, et al. "Helmet detection using machine learning and automatic License Plate Recognition." Int. Res. J. Eng. Technol.(IRJET) 6 (2019): 80-84.
20. Lin, Cheng-Hung, and Chen-Hao Wu. "A lightweight, high-performance multi-angle license plate recognition model." 2019 international conference on advanced mechatronic systems (ICAMechS). IEEE, 2019.
21. https://github.com/ultralytics/ultralytics
22. https://streamlit.io/
23. Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." arXiv preprint arXiv:1804.02767 (2018).
24. He, Kaiming, et al. "Spatial pyramid pooling in deep convolutional networks for visual recognition." IEEE transactions on pattern analysis and machine intelligence 37.9 (2015): 1904-1916.
25. Ge, Zheng, et al. "Yolox: Exceeding yolo series in 2021." arXiv preprint arXiv:2107.08430 (2021).
26. X. -T. Vo and K. -H. Jo, "Enhanced Feature Pyramid Networks by Feature Aggregation Module and Refinement Module," 2020 13th International Conference on Human System Interaction (HSI), Tokyo, Japan, 2020, pp. 63-67, doi: 10.1109/HSI49210.2020.9142674.
27. Zheng, Zhaohui, et al. "Distance-IoU loss: Faster and better learning for bounding box regression." Proceedings of the AAAI conference on artificial intelligence. Vol. 34. No. 07. 2020.
28. Li, Xiang, et al. "Generalized focal loss: Learning qualified and distributed bounding boxes for dense object detection." Advances in Neural Information Processing Systems 33 (2020): 21002-21012.
29. Du, Yongkun, et al. "Svtr: Scene text recognition with a single visual model." arXiv preprint arXiv:2205.00159 (2022).
30. Cui, Cheng, et al. "PP-LCNet: A lightweight CPU convolutional neural network." arXiv preprint arXiv:2109.15099 (2021).

Appendix

BB is a 1D-array of the form [x0, y0, x1, y1] containing bounding box information of the object.

N (nohelmet), ***M*** **(**motorcyclist) and ***L*** (license plate) are 2D-arrays containing [ID, BB].

N (Nohelmet class) contains the ID provided by Bytetrack, BB of nohelmet and BLT(Belong to) is a 1D-array containing M objects.

M (Motorcyclist class) contains the ID provided by Bytetrack, BB of motorcyclist and BLT(Belong to) is a 1D-array containing L objects.

L (Licenseplate class) contains the ID provided by Bytetrack, BB of the license plate, PLI (Plate_info) is the information extracted from the number plate, SCP (Score_plate) is the reliability of PLI.

Appendix 1. Identify vehicles without helmet

Algorithm 1: Determination of **M** of **N**

```
M_tmp ← []
for i ← 0 to Length (N) do
    N = N[i]
    for j ← 0 to Length (M) do
        M = M[j]
        if CIP(N.BB, M.BB) ≥ TS & N.BB > N_lower & N.BB < N_upper then
            if N.BLT is None then
                N.BLT ← [M]
            else
                N.BLT[N.BLT.length + 1] ← M
            if M not in M_tmp then
                M_tmp[M_tmp.length + 1] ← M
            End If
        End If
    End For
End For
M ← M_tmp
```

After obtaining information about **N**, the corresponding **M** for each **N**, **P** for each **M**, and Frame_id while traversing the video, we will initialize an object **O** with the respective information and pass it to the MT.

Suppose in a certain frame, an **N** can belong to 2 **M**. Therefore, we need to determine which **M** is correct. Similarly, in one **M**, there can be 2 **L**, so we need to identify the correct **L**.

Let **A** be a 1D-array containing **N**.ID in MT, corresponding to column 2.

Let **B** be a 1D-array containing **M**.ID in MT, corresponding to column 3.

Let **C** be a 1D-array containing **L**.ID in MT, corresponding to column 4.

Appendix 2. Assign no_helmet class to its vehicle

Algorithm 2: Filtering vehicles to ensure that one **N** corresponds to only one **M**.

```
D ← empty dictionary

for each n in unique_values (A) do
    D[n] ← empty dictionary
    for 0 ← i to Length (B) do
        if A[i] = n then
            D[n][B[i]] ← 0
        End If
    End For
End For

for 0 ← i to Length (B) do
    D[A[i]][B[i]] ← D[A[i]][B[i]] + 1
End For

N_M ← empty dictionary

for each n in keys(D) do
    max_key ← key_with_max_value(D[n])
    N_M[n] ← max_key
End For
```

'N_M' is a dictionary containing keys as **N**.ID and corresponding values as **M**.ID.

We will filter the rows in **DT**, meaning the **O** in **DT**, such that (**N**.ID in keys(*N_M*) & *N_M*[**N**.ID] == **M**.ID) & (**M**.ID in keys(*M_L*) & *M_L*[**M**.ID] == **L**.ID). For each vehicle, we need to have a unique license plate.
Considering a vehicle in **DT**, similar to the remaining vehicles.
(The relevant algorithms will be detailed in the Appendix section)

Appendix 3. Identify of plates of vehicles without helmet

Algorithm 3: Determination of **L** of **M** (Identification of plates of vehicles without helmet)

```
L_tmp = []
for i ← 0 to Length (M) do
    M = M[i]
    for j ← 0 to Length (L) do
        L = L[j]
        if CIP(L.BB, M.BB) ≥ TS & L.BB > L_lower & L.BB < L_upper then
            if M.BLT is None then
                M.BLT ← [L]
            else
```

```
                    M.BLT[M.BLT.length + 1] ← L
                if L not in L_tmp then
                    L_tmp[L_tmp.length + 1] ← M
                End If
            End If
        End For
    End For
L ← L_tmp
```

Appendix 4. Assign license plate to its vehicle

Algorithm 4: Filtering license plates to ensure that one **M** corresponds to only one **P**.

```
E ← empty dictionary

for each n in unique_values(B) do
    E[n] ← empty dictionary
    for i ← 0 to length(C) do
        if B[i] = n then
            E[n][C[i]] ← 0
        End If
    End For
End For

for i ← 0 to length(C) do
    E[B[i]][C[i]] ← E[B[i]][C[i]] + 1
End For

M_L ← empty dictionary
for each n in keys(E) do
   max_key ← key_with_max_value(E[n])
   M_L[n] ← max_key
End For
```

'M_L' is a dictionary containing keys as **N**.ID and corresponding values as **M**.ID.
Let **F** be a 1D array containing FID of the currently examined **M**.
Let **S** be a 1D array containing the SCP values of the **P** within the currently examined **M**.

Appendix 5. Choose the best frame contain license plate

Algorithm 5: Select the FID position with the highest SCP for the currently examined **M**.

```
max_SCP ← max(S)
max_index ← S.index(max_SCP)
FID_highest_SCP ← F[max_index]
```

'FID_highest_SCP' is the Frame_id where **L**.PLI has the highest **L**.SCP.

Appendix 6. Method of calculating interference

Algorithm 6: Calculate the percentage of intersection area between two bounding

boxes (BB).

```
procedure CIP(box1, box2):

    area_box1 = (box1[2] - box1[0]) * (box1[3] - box1[1])
    area_box2 = (box2[2] - box2[0]) * (box2[3] - box2[1])

    intersection_x1 = max(box1[0], box2[0])
    intersection_y1 = max(box1[1], box2[1])
    intersection_x2 = min(box1[2], box2[2])
    intersection_y2 = min(box1[3], box2[3])

    if intersection_x2 < intersection_x1 or intersection_y2 < intersection_y1:
        return 0
    area_intersection = (intersection_x2 - intersection_x1) * (intersection_y2 - intersection_y1)

    percentage_intersection = area_intersection / min(area_box1, area_box2)

    return percentage_intersection
end procedure
```

O (Object class) contains FID (Frame_id) which is the order of frames in the video**,** **N**, M**,** and **L**

MT (Metadata) is a Dataframe that stores information of **O** in frames including: **O**.FID**,** **N**.ID**,** **M**.ID, **L**.ID, **N.**BB, **M**.BB, **L**.BB, **L**.PLI, **L**.SCP
None is of no value.
N_lower, N_upper, L_lower, L_upper are the thresholds that we set to determine whether **N** or **L** belongs to **M**.

TS is the threshold that determines whether 2 BB intersect or not.