MOTORBIKE IMAGE RECOGNITION USING YOLO

MR.PHASIN PLOYPICHA
MR. PIYAKORN RODTHANONG
MR.SOPON PLEANGNOI

A PROJECT REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE BACHELOR OF ENGINEERING IN COMPUTER ENGINEERING

FACULTY OF ENGINEERING & INTERNATIONAL COLLEGE
MAHIDOL UNIVERSITY

2025

MOTORBIKE IMAGE RECOGNITION USING YOLO

MR.PHASIN PLOYPICHA
MR. PIYAKORN RODTHANONG
MR.SOPON PLEANGNOI

A PROJECT REPORT SUBMITTED IN PARTIAL
FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE
BACHELOR OF ENGINEERING IN COMPUTER ENGINEERING

FACULTY OF ENGINEERING & INTERNATIONAL COLLEGE
MAHIDOL UNIVERSITY
2025

Computer Engineering Project
entitled

MOTORBIKE IMAGE RECOGNITION USING YOLO

Mr.Phasin Ploypicha
Researcher

Mr. Piyakorn Rodthanong
Researcher

Mr.Sopon Pleangnoi
Researcher

Dr. Mingmanas Sivaraksa , Ph.D. (Information Engineering),
Ph.D. (Computer Innovation Engineering)
Advisor

Thesis
entitled
MOTORBIKE IMAGE RECOGNITION USING YOLO

was submitted to the Faculty of Engineering & International College,
Mahidiol University
for the degree of Bachelor of Engineering (Computer Engineering)
on
July 25, 2025

_____
Dr. Mingmanas Sivaraksa , Ph.D. (Information Engineering),
Ph.D. (Computer Innovation Engineering)
Chair Committee

_____
Committee 1, Ph.D. (Computer Engineering)
Committee

_____
Committee 2, Ph.D. (Computer Engineering)
Comittee

BIOGRAPHY

NAME	Mr.Phasin Ploypicha
DATE OF BIRTH	28 March 2003
BIRTHPLACE	Bangkok, Thailand
EDUCATION BACKGROUND	Mahidol International College
MOBILE PHONE	111 111 1111
E-MAIL	phasin.plo@student.mahidol.edu

NAME	Mr. Piyakorn Rodthanong
DATE OF BIRTH	15 May 2003
BIRTHPLACE	Bangkok, Thailand
EDUCATION BACKGROUND	Mahidol International College
MOBILE PHONE	222 222 2222
E-MAIL	piyakor.rod@stdent.mahidol.edu

BIOGRAPHY

NAME	Mr.Sopon Pleangnoi
DATE OF BIRTH	03 March 2002
BIRTHPLACE	Bangkok, Thailand
EDUCATION BACKGROUND	Mahidol International College
MOBILE PHONE	333 333 333
E-MAIL	Sopon.pln@student.mahidol.edu

ACKNOWLEDGEMENT

First and foremost, we would like to express our special thanks and sincere of gratitude to our advisor and co-advisors, Asst. Prof. XXX XXXX, Asst. Prof. YYY YYYY, Asst. Prof. ZZZ ZZZZ, who insight, advise and knowledge us a lot in finalizing this project within the limited time frame.

Beside our advisors, we would like to thank to the committees, Asst. Prof. XXX XXXX, Asst. Prof. YYY YYYY, and Asst. Prof. ZZZ ZZZZ, for their insightful comment and encouragement. Also, the questions which encourage us to rethink and research more about some factors which we missed. Therefore, this project would not be completed without comments, questions, and support from our committees.

Finally, we would like to special thanks to our university, Mahidol University International College, and Faculty of Engineering, Mahidol University, which provided us a lot of resources to study, financial means for researching this project.

Mr.Phasin Ploypicha
Mr. Piyakorn Rodthanong
Mr.Sopon Pleangnoi

ชื่อโครงงาน	ชื่อวิทยานิพนธ์
ผู้วิจัย	นายชื่อผู้แต่ง นามสกุลผู้แต่ง ICCI
	นางสาว ชื่อผู้แต่ง นามสกุลผู้แต่ง ICCI
	นายชื่อผู้แต่ง นามสกุลผู้แต่ง ICCI
	ปริญญา วิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษา	ดร. ชื่อที่ปรึกษา นามสกุลที่ปรึกษา
สำเร็จการศึกษา	15 กรกฎาคม 2567

บทคัดย่อ

บทคัดย่อภาษาไทย (not required for your project proposal, but it is mandatory for the black book)

คำสำคัญ : ลาเท็ก / วิทยานิพนธ์ (~5 คำ)

34 หน้า

MOTORBIKE IMAGE RECOGNITION USING YOLO

STUDENTS	Mr. Phasin Ploypicha ICCI
	Mr. Piyakorn Rodthanong ICCI
	Mr. Sopon Pleangnoi ICCI
DEGREE	Bachelor of Engineering (Computer Engineering)
PROJECT ADVISOR	Dr. Mingmanas Sivaraksa , Ph.D. (Information Engineering),
	Ph.D. (Computer Innovation Engineering)
DATE OF GRADUATION	July 25, 2025

ABSTRACT

The paper describes an implemation of system to help in detection of motorbike, person using YOLO in combination with a custom model to detect head and helmet. In order to count these classes and document the class entering University gates through surveillance camera. The data is to be recorded throgh google sheet API, taking snapshot of a specific scenario where a civillians using motorbike are not wearing helmet. YOLOv8 is uses as the main framework for it performances and continuous supports from developers as well as its accessibility. The custom model for helmet and head detection are trained, annotated through ROBOFLOW program, and the datasets are from the university's cctv camera. Five model has been trained and, one of the five has been choosesn for their perfromace to be implemented for the detections. The product is a system that detects motorbike, person, helmet and head. Using BotSort tracker in conjunction to aid in the counting algorithm.

KEYWORDS : LaTeX / Thesis (~5 words)

34 Pages

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure **Page**

CHAPTER 1
INTRODUCTION

1.1 Background

Urban traffic congestion and road safety, particularly for motorbikes, remain major issues. Motorcycles, due to their mobility, contribute greatly to traffic flow. However, they are also prone to violations, such as not wearing helmets or using the wrong lane, which increase the chance of an accident. Traditional traffic management systems lack the ability to detect and address such specific breaches in real time.

AI technologies, like as YOLO, a deep learning model for object detection, have showed potential in traffic surveillance but have not yet been routinely used to identify helmet use or lane breaches. Previous research has shown that YOLO is good for vehicle detection, but there is a gap in its use for targeted motorbike detection and safety compliance.

This research seeks to close this gap by utilising YOLO to detect motorcycles, check for helmet use, and identify lane violations. By focussing on these critical aspects, the system will improve road safety and aid in more effective traffic enforcement.

1.2 Objective

1. To detect motorbike, person, helmet and non-helmet user.
2. Count all the essentails object efficiently
3. upload system up to university's server.

1.3 Scope

1. Develop system to detect motorbike, person, hemet and non-helmet user.
2. Display annotated video.
3. Count all the essentails and display real time.
4. University server running on system efficiently.

1.4 Expected Results

1. To reduce illegal activities of riders.
2. To ensure both safety of rider and pedestrian.
3. To spraed safety awareness.
4. Has more than 80 percent accuracy .

1.5 Timeline

Table 1.1 Project Timeline

Plan	Timeline											
	2024									2025		
	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May	Jun	July	Aug	Sep
Wrting Proposal												
Review Paper and Methodology												
Data Gathering												
Start Training Model												
Finalize Training Model												
Prepare University Server Specs												
Deploy and connect to database												
Presentation												

CHAPTER 2
LITERATURE REVIEW

The papers in the literature review are used as a foundation to guide, the thesis, this included inspiration, the methodoly, and expected results.

2.1 Video Image Recognition of Car Track Characteristics at Intersections [6]

The system relies on machine learning algorithms to interpret video data in real-time, helping to optimize traffic signal control, reduce congestion, and enhance safety. The study emphasizes its application in intelligent transportation systems and urban traffic flow optimization.

The use of YOLOv8, a popular deep learning model for object detection, because it offers a balance of speed and accuracy. YOLOv8 is efficient in processing real-time video streams, which is crucial for tracking vehicles at intersections. It excels in recognizing objects (like cars) in complex environments, and its architecture allows it to detect vehicles quickly while maintaining high precision.

2.2 A Lightweight Remote Sensing Small Target Image Detection Algorithm Based on Improved YOLOv8 [3]

The authors optimize the YOLOv8 architecture by refining the feature process and introducing techniques like multi-scale detection, anchor box adjustments, and a better feature fusion method. This helps the algorithm perform better with small targets. ensuring more accurate detection without significantly increasing computational cost.

they conclude that, this makes it suitable for real-time applications in remote sensing, where detecting small objects like vehicles, ships, or buildings in satellite imagery is critical. The enhancements lead to better precision and recall.

2.3 Traffic Detection and Enhancing Traffic Safety: YOLO V8 Framework and OCR for Violation Detection Using Deep Learning Techniques [5]

The YOLOv8 algorithm is a deep learning model that uses a convolutional neural network (CNN) to detect and classify objects in input images. It uses bounding box regression and class prediction to identify objects and assign class probabilities.

The model is pre-processed by converting the input video into frames, extracting features, and detecting objects. The YOLOv8 model then uses a grid to track objects, predicting bounding boxes and class probabilities. The model is fine-tuned on specific object classes and implemented for tracking, counting, and speed estimation.

This project consists of two main modules: one for vehicle count and speed detection, which uses a dataset of 800 images and video clips, and the other for number plate recognition. The first module uses computer vision techniques to accurately count vehicles and calculate speeds. The second module uses a Roboflow website dataset to extract license plate information, useful for applications like parking management and traffic monitoring.

This project aims to create two modules with distinct functions in traffic monitoring. The number plate recognition module enhances functionality for recognizing and tracking vehicles based on their number plates, while the vehicle count and speed detection module provides insights into traffic flow and speed trends.

The study on traffic detection using the YOLOv8 framework and Optical Character Recognition (OCR) has shown promising results. The system accurately detects traffic violations in real-time, while OCR enhances its ability to identify and classify violations like illegal parking or speeding.

YOLOv8 and OCR provides a robust solution for automating violation detection processes, demonstrating the effectiveness of deep learning methodologies in enhancing traffic safety and enforcing traffic regulations.

2.4 Smart Traffic Management System for Efficient Mobility and Emergency Response [1]

The proposed solution combines AI with real-time data analytics to improve traffic light management. It incorporates real-time congestion response, algorithmic optimization, data-driven decision making, and feedback loop integration.

The system uses a vast dataset of lane-specific traffic information to inform real-time adjustments. The Traffic Light Control System (OF) is activated, and the dataset undergoes preprocessing to ensure compatibility with the algorithms. A predictive model is trained using machine learning techniques, and a feedback loop is implemented to continuously evaluate the effectiveness of the decisions.

The AI system also optimizes traffic flow during emergencies, adjusting timings to prioritize passage and ensure smooth traffic flow. This innovative approach aims to alleviate congestion and improve overall traffic flow, ultimately leading to sustainable and safer cities.

2.5 Utilizing Image Processing and the YOLOv3 Network for Real-Time Traffic Light Control [2]

The paper presents the creation of a traffic light control system which is aimed at relieving congestion in urban areas. The system is designed based on the YOLOv3 object detection algorithm and is capable of counting both vehicles and pedestrians at intersections in real time using a Jetson Nano board for data processing. The research further shows that traffic lights can be controlled by taking into consideration factors such as the number of vehicles that have stopped, the number of vehicles that are crossing over and the number of pedestrians thereby enhancing the efficiency of the system which in turn reduces negative impacts on the environment.

The authors note the cost entailed in relief from congestion in UK being in the region of £37.7 billion per year and admonish that such inefficient delay is caused by the inadequate control of traffic lights that energy loss is also the outcome of that inefficient management of the traffic signal is responsible. The developed system loks set to achieve average precision with average vehicle counting for 95 percent as supported by this research.

2.6 Research on Vehicle Lane Change Warning Method Based on Deep Learning Image Processing [4]

The paper presents the progress in work which is falling into the category of the development of the low-cost lane-changing warning system which works in the efforts to improve driving safety through monocular camera and deep learning algorithms. The main aim of the study is to detect when there is a pole vehicle-user behavior who would be changing the existing lane as this is paramount to mitigate risk of collision, consequently enhance safety of roads.

As a result, the authors enhanced the Mask Region-based Convolutional Neural Network (Mask R-CNN) aiming towards a vehicle target seeker. They used the K-means clustering technique to identify appropriate proportions of anchor frames for vehicle targets, which facilitated more precise candidate box generation without degrading the performance of the network. This led to introduction of a fresh anchor set and mAP increased by about 2.6 percent thanks to about 26 percent in the mean average precision improvement.

The system was trained in a vernacular fashion with a full with 66,389 vehicle targets which were adequately annotated for target annotation assuring their appropriate performance. The authors performed various evaluations and recorded an accuracy of 94.5 percent for detection of lane changing after identifying and validating marked images of lane changing sequences. This high accuracy demonstrates the effectiveness of the proposed system in real time applications.

2.7 Summary

The reviewed studies highlight innovative uses of AI and deep learning for traffic management and safety. One study employs the YOLOv8 framework and Optical Character Recognition (OCR) to detect traffic violations in real-time, using modules for vehicle counting, speed estimation, and license plate recognition, achieving effective enforcement of traffic regulations. Another study integrates AI with real-time analytics to optimize traffic light control, dynamically responding to congestion and prioritizing emergency vehicles, thereby improving urban mobility and sustainability. Similarly, a traffic light control system based on YOLOv3 counts vehicles and pedestrians at intersections, achieving 95 percent accuracy in optimizing traffic flow and reducing urban congestion. Lastly, a lane-change warning system utilizing Mask R-CNN and K-means++ clustering achieves 94.5 percent accuracy in detecting lane changes, offering a cost-effective solution for collision risk mitigation and enhanced road safety. These approaches collectively demonstrate the transformative potential of AI in traffic management.

CHAPTER 3
METHODOLOGY

3.1 System Design

The purpose of the helmet detection system is to track helmet use from CCTV footage on a college campus. Data preparation, model training, and counting implementation are the three primary phases of the system. The first step involves converting campus CCTV video footage into image frames, which are subsequently labelled with helmet and unhelmeted head information using Roboflow. A customised YOLOv8 object detection model that is tuned for real-time performance is trained using the annotated dataset. After training, the model is incorporated into a real-time system that recognises and categorises heads and helmets by processing incoming video frames. In order to provide real-time statistics that facilitate safety compliance monitoring, a counting mechanism is integrated to monitor the quantity of helmets and heads seen within a designated area of interest.

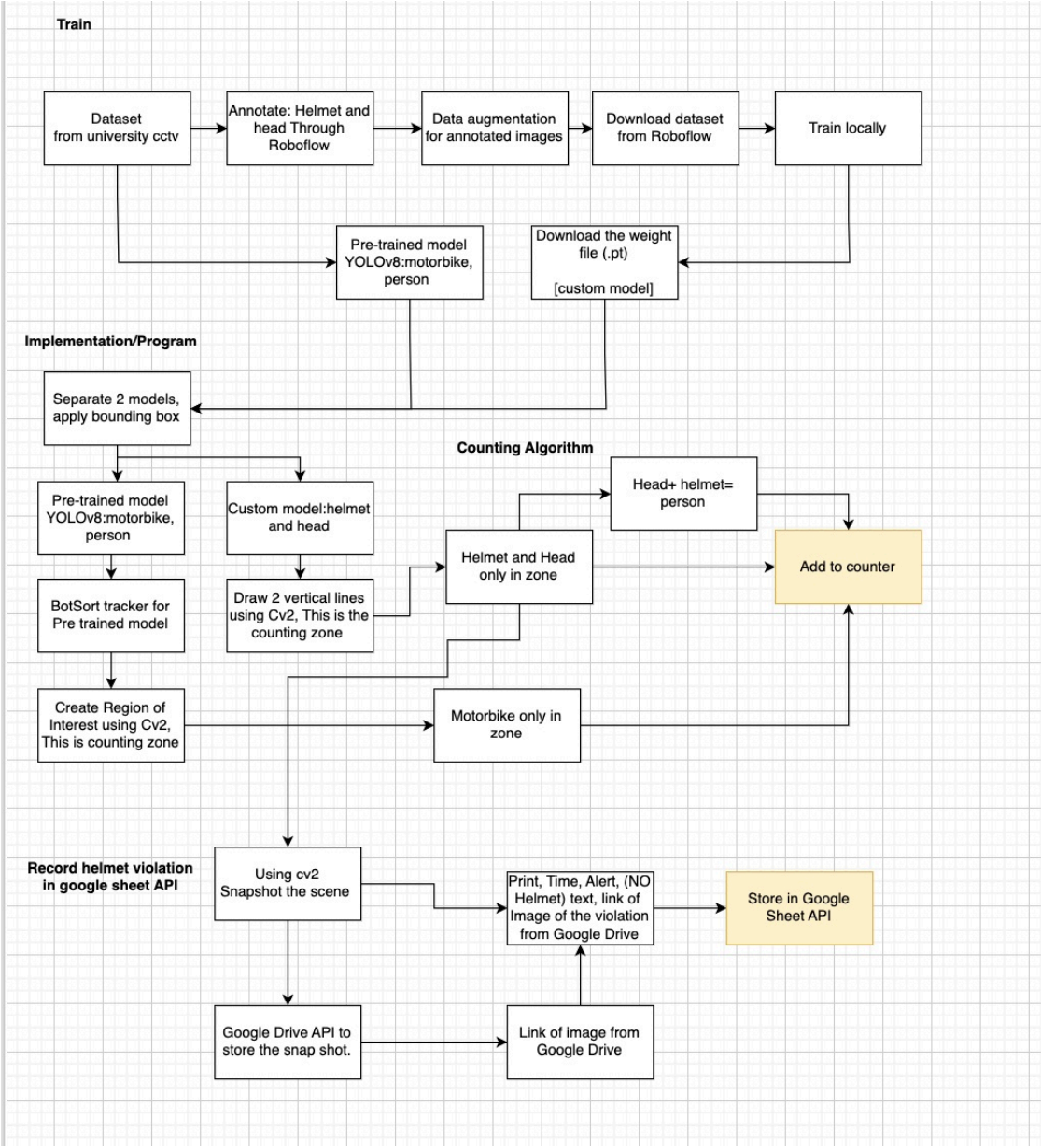

Figure 3.1

The flowchart illustrates the overall workflow of the system. It begins first obtaining the dataset, we obtain them through the cctv footage of the university camera. Specifically spliting the video footages into 3 frames per second using Roboflow. Then annotate each frames splited, the annoated items are helmet and head, then using data augmentations from Roboflow. Then we can downlaod the datasets. Finally, using dataset downloaded we can train locally through data.yaml, the data augmentations

details are to be specified in Chapter 3. After the training we will receive the weighted .pt file, the file is the model that will be used in the system in conjunction with the pre-trained model from YOLOv8. The Implementation phase is to apply the models, work with the video stream input. The bounding box and region of interest made from cv2. Using these zone we speparate the counting zone for helmet, head, and another zone to count person and helmet. As for the person we decided to count the head and the helmet to total to person as there are issues about the camera angel limiting the accuracy of the person detection.

After implementing the program with the model, we use cv2 to take snapshot of the frame as when helmet violation occur. The system links and uses Google API, to store the detection evidence of the scenario. The Google Sheet API will append new rows each time a head detection occur. The image are visible, and stored by Google Drive API, to check the instances where it happens.

3.2 Data

Dataset: videos of riders wearing and not wearing helmets, collected from local sources which is gate six Mahidol University CCTV camera. Videos will then be split in Roboflow which allow custom annotation and training.

Annotations: Create two classes. first class containing the bouding box of helmet user only and the second class containing head(which is non-helmet user.)
Preprocessing: Includes resizing images to 640x640 pixels, normalizing pixel values, and applying augmentations (e.g., rotation, scaling, and brightness adjustments).

3.3 Custom Model

YOLOv8 Architecture: The YOLOv8 architecture was selected for this project due to its state-of-the-art performance in object detection, offering a balance between real-time inference speed and high accuracy. YOLOv8 (You Only Look Once version 8), developed by Ultralytics, is known for its streamlined architecture and improved detection performance over its predecessors (e.g., YOLOv5, YOLOv7), particularly in low-light or occluded environments—conditions often encountered in real-world surveillance footage. Specifically for this project, YOLOv8 is supported by Roboflow, a tool important for training and annotating helmet and head classes easily. Additionally YOLOv8 was easier to set up and supported by developers and communities in a larger scale than other YOLO versions. This will be essentials in debugging and implementing additional tools like trackers.

Despite the availability of newer object detection models such as YOLO-NAS or YOLOv9, YOLOv8 remains a highly reliable and widely supported choice. In this project, YOLOv8 was used to create a custom-trained model that focuses on five specific object classes: motorcycles, helmets, crosswalks, pedestrians, and lanes. However, the two most critical objects for our safety analysis—helmet and head—were prioritized in both annotation and evaluation.

Figure 3.3.1: Roboflow Image

Figure 3.3.2: YOLO Image

To obtain the custom model for "head", and "helmet" detection we must first upload the datasets into Roboflow website in figure 3.1. Roboflow will allow users to specifically annotate desired classes, which after annotating said datasets, we can download directly from Roboflow the annotated datasets which we can train locally. Training can be done localy or through Google Collab. However due to limitations we used Vistual Studio as a medium to train the datasets from Roboflow.

Training Process: As for our goals the training processes for our custom YOLOv8 model consisted of several key stages, summarized below:

- **Data Collection:** The dataset was compiled from real CCTV footage recorded on campus at Mahidol University's Faculty of Engineering. The footage was collected over a span of seven days during the morning rush hour (8:00–10:00 AM), totaling approximately two hours. This time window was chosen to capture peak motorcycle and pedestrian traffic for accurate helmet usage monitoring. The image below is an example of a snapshot from the dataset, the actually dataset is a video as mention.

Figure 3.3.3: Dataset Image

- **Frame extracgtion:** Using the cv2 module from OpenCV, video footage was processed and extracted into still frames. To ensure computational efficiency while retaining sufficient information, frames were sampled at a rate of 3 frames per second (FPS). The extracted images were then uploaded to Roboflow, a computer vision dataset platform, for annotation and augmentation. However for Roboflow they provide a function to slpit the frames within the website and we will be extracting them with Roboflow instead. In the figure below is an example of frame extraction function after uploading the datasets into Roboflow. Extracting the video datasets into frames, means that we can annotate each frames specifically we have three frames across one second. This is to also ensure that we do not have to much datasets, and it also depends on how much the desire classes appears on the videos. For our case there are enough objects across three frames per second to annotate.

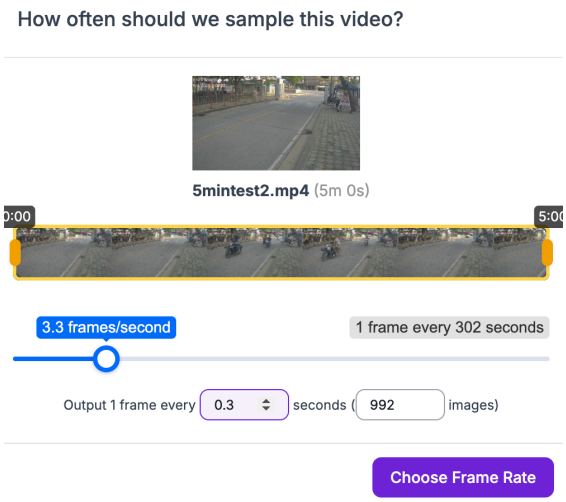

Figure 3.3.3: Frame Extraction Image

- **Annotation:** Within Roboflow, bounding boxes were manually drawn around the target objects—specifically helmets and heads—to create accurate ground truth data for supervised learning. Annotations were carefully reviewed to ensure consistency, especially in challenging cases such as partial occlusions or non-standard helmet designs. The images belows show an example how we draw the bounding boxes for the classes. Drawing the bouding box this way minimizes detection errors and allows the YOLOv8 custom model to focus on the objects more precisely. We encourage to draw the bounding box as close to the objects as possible.

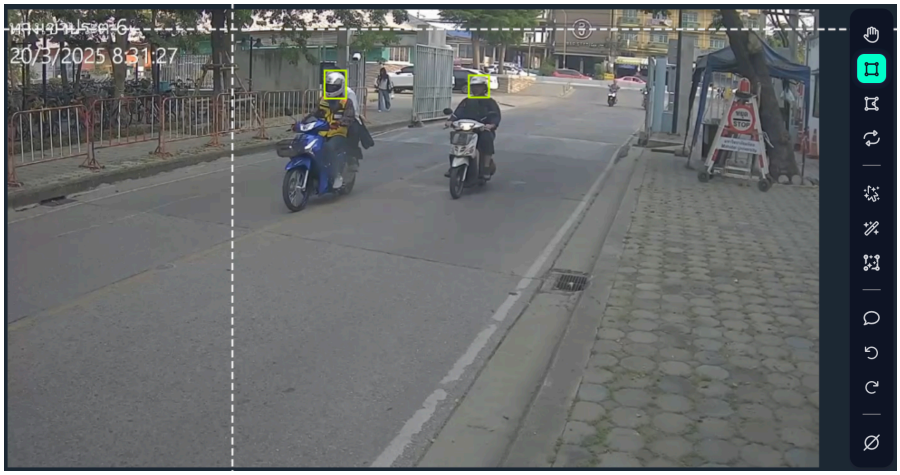

Figure 3.3.4: Annotation Image

- **Dataset Splitting:** The annotated dataset was divided into training, validation, and testing sets with a ratio of 70% training, 20% validation, and 10]% testing. This split was chosen to ensure sufficient training samples while preserving a robust validation set to monitor overfitting and a separate test set for unbiased evaluation. Using this data split is widely uses apporach in machine learning as it offers balnce between training, validation, and testing. 70% of the data use to train allows it to learn patterns from the annotation effectively. It is effieicent and enough for pattern recognition. The 20% is very important as becuase as this portion is used to check how well the model is learning. Additionally to help tune with the hyperparameters and prevent overfitting. Overfiting occurs when a model memorizes the training data too well and fails to perform on new, unseen data. Lastly the 10% is reserved for testing, for unbaised evaluation of the model. The split ensures thatt the model has enough data to learn, and adjust. This process will occur after completing the annotation process.

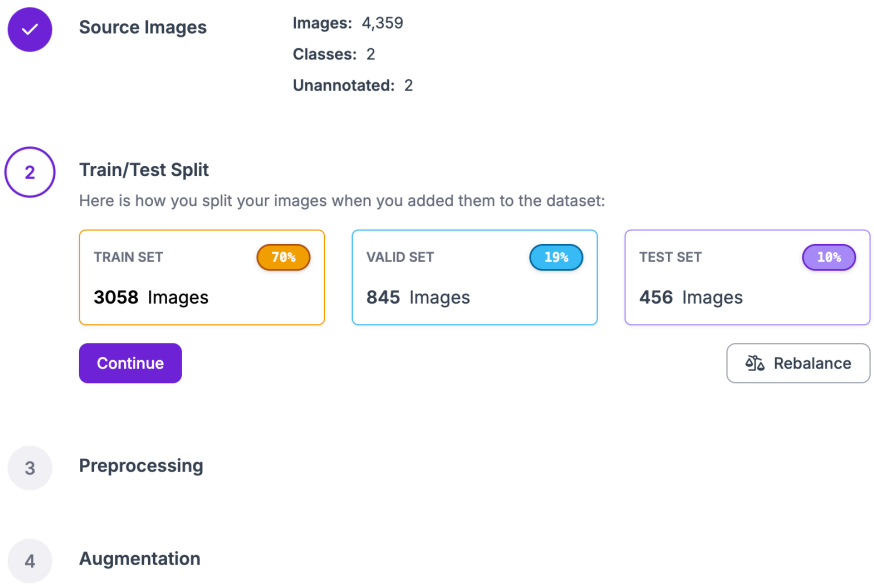

Figure 3.3.5: Spliting Datasets Image

At this step we can also augmentate our data through data augmentation from Roboflow seen in Figure 3.3.5. Data augmentation techniques applied are horizontal flipping, contrast enhancement, and brightness variation were applied to

improve the model's robustness to environmental variations such as lighting conditions or camera angles.

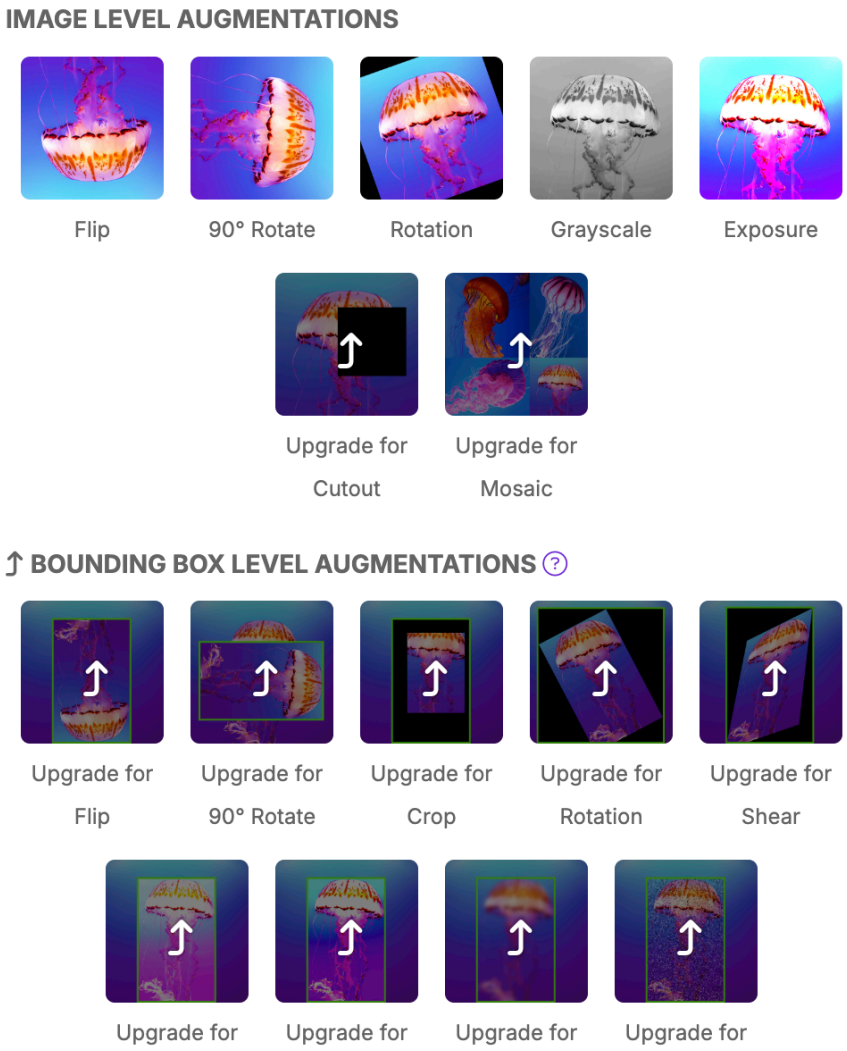

Figure 3.3.6: Augmentation Image

- **Data Configuration and Training:** Aftering Spliting the data we can download the dataset files from Roboflow. The file will be splited to 70/20/10, train, validate, test. A data.yaml file will have to be configured to specify class names, dataset paths, and other training parameters. The file is then imported imported into Visual Studio Code (VS Code) for model development and training using the Ultralytics YOLOv8 framework. The images below shows how data.yamlfile is set up.

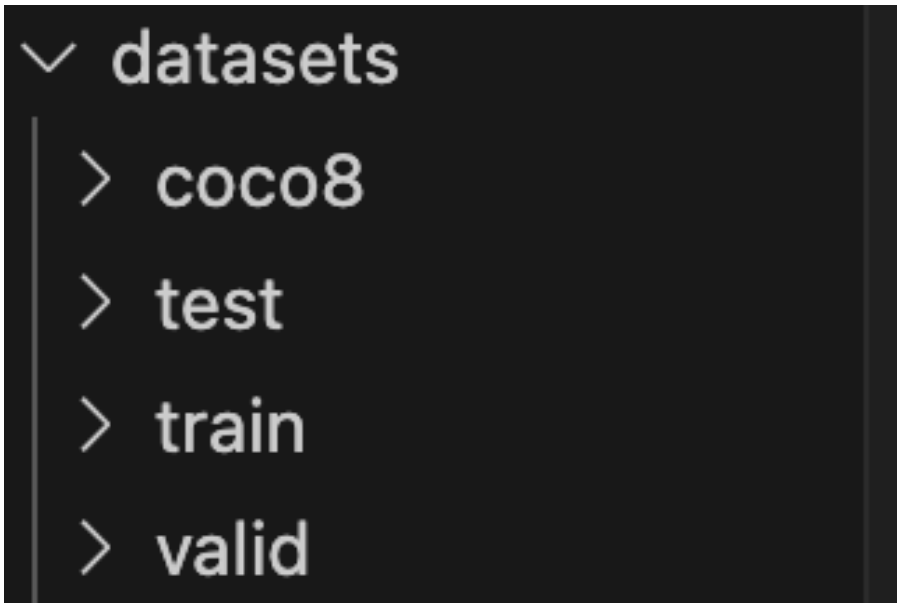

Figure 3.3.7: Files from downloaded Dataset

The detailedconfigurations needed used inside data.yaml is located in the Hyperparameter section. Hyperparameters, such as learning rate and non-maximum suppression (NMS) thresholds, were tuned for optimal performance. After applying the hyparameters, we can start the training through the data.yaml file. After the training has been finish, the wieghted model files will be created in runs/detect. Select the .pt files as a yolo model this will be the model that we will use after all the steps above are complete. For our project be have a total of 4,359 images that are annotated for custom model. The image belows shows the data.yaml content. We will use this to obtain a custom weighted .pt file.

```python
results = model.train(
    data='data.yaml',
    epochs=500,                # Updated
    imgsz=640,                 # Image resolution
    batch=32,                  # Updated batch size
    patience=50,
    save=True,
    save_period=10,
    device='cpu',              # Use 'cuda' if you have GPU
    workers=4,
    resume=True,
    exist_ok=True,
    name='helmet_continued',
    pretrained=True,
    optimizer='Adam',
    lr0=0.005,                 # Updated learning rate
    project='runs/detect',
    save_dir=output_dir
)

# === Save the final combined model ===
output_path = os.path.join(output_dir, 'weights', 'final_model.pt')
os.makedirs(os.path.dirname(output_path), exist_ok=True)
model.export(format='pt', filename=output_path)

print(f"Training completed. Model saved to: {output_path}")
```

Figure 3.3.8: Data.yaml file in VS

3.4 Counting System

3.4.1 Helmet and Head

After customizing our YOLOv8 model, we implemented a counting system to track the number of helmets, unhelmeted heads, persons, and motorcycles detected in the video.

For helmet and head counting, we applied a double-line counting method. This defines a "counting zone" using two vertical lines, and only counts detections whose bounding box centers fall within the zone. To prevent duplicate counting across frames, we used a rolling memory of the last ten frames. If a new detection is within a set distance (e.g., 30 pixels) of a previously counted object of the same type, it is ignored. This method improves reliability by reducing overcounting and stabilizing results, especially for slow-moving or flickering detections. The final counts are updated in real time and displayed on the video, offering a clear summary of helmet usage and safety compliance.

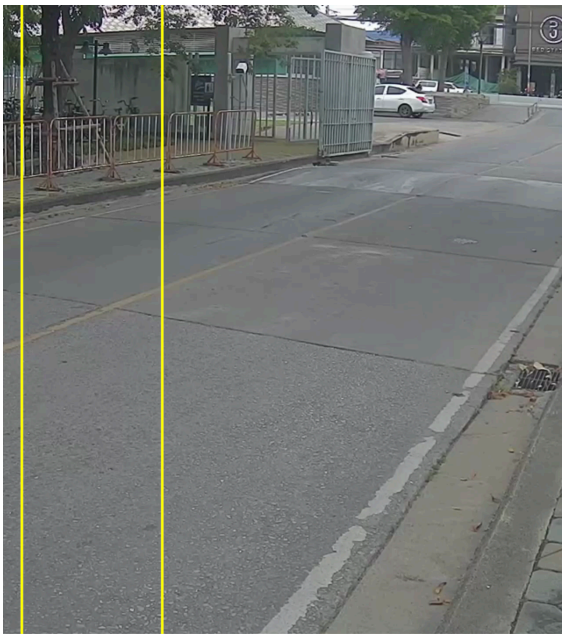

Figure 3.2

Figure 3.2 illustrates the designated counting zone used for helmet and head detection. The image shows two vertical yellow lines placed 150 pixels apart, which define the area for double-line counting. When an object passes through both lines in sequence, it is registered as a valid count. This method helps reduce duplicate or false counts caused by temporary detection overlaps or brief tracking loss.

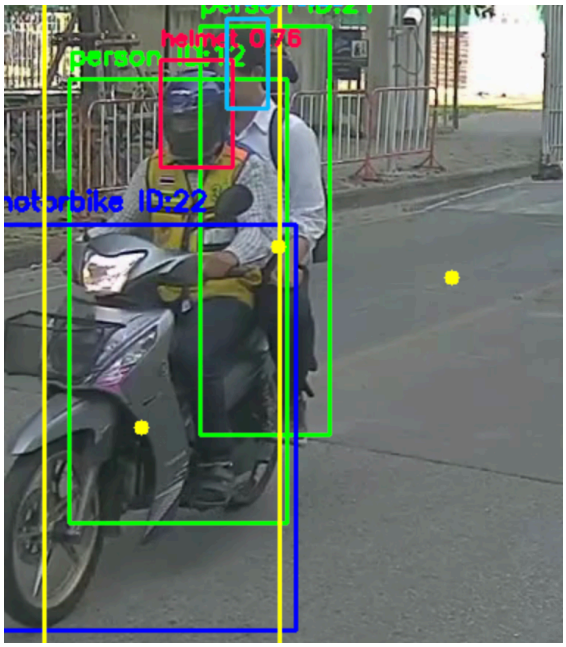

Figure 3.3

Figure 3.3 shows an example of the counting system in action when objects, such as a helmet and/or a head, enter the counting zone. As seen in the image, the objects are detected and labeled while passing through the two vertical lines, triggering the counting logic. This illustrates how the system identifies and differentiates helmeted and non-helmeted riders within the defined zone.

Figure 3.4

Figure 3.4 displays the result of the helmet and head counting process, shown at the top-right corner of the screen. These values are updated in real-time after the system detects and counts objects passing through the counting zone. This visual feedback confirms the detection and classification outcome for each rider, distinguishing between helmeted and non-helmeted individuals.

3.4.2 Person and Motorbike

However as for the Person, and Motorbike detection we can use the Yolov8s.pt model, class 0(person), and class 3(motorbike), as their model are well polished for tracking. For the tracking use Bot-sort tracker, obtain through Roboflow ultralytics/trackers/bot_sort.py. The tracker will be use to create specific id number for each class that has been detected. Using these id we will create center point of each classes which will be important in the counting method.

For the counting method, we can create a region of interest, through using open CV. From there we create a condition, to store the track id with track history. track_history = function will store the position cx, cy of the tracked object. We will use this value to identify its current and previous position to ensure that when object inside the Region of Interest, position compare is is greater than the current position objected will not be used to count, while if the current position of the tracked id compared to stored id is less than it, it can be counted inside the ROI. This is to ensure that the person, and motorbike can be counted by by what it can be tracked. The detection should look similiar to figure1.

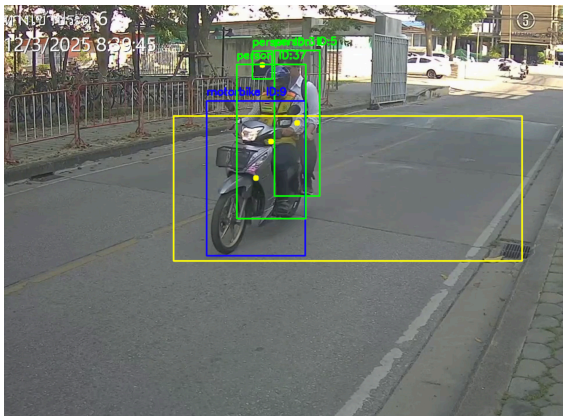

Figure 3.5

3.4.3 Output Generation

The output generation phase transforms the detection and counting results into a visual and interpretable format. For each processed video frame, the system overlays bounding boxes and labels indicating the object class (helmet, head, person, or motorcycle), along with live counts displayed on the top-left corner using OpenCV. These real-time visual indicators allow users to monitor helmet usage compliance at a glance.

Additionally, the system can produce an annotated video file as output, which includes all detections and updated counts across the entire duration. This output is valuable for both immediate observation and later review or reporting, providing a clear and documented summary of safety compliance in the monitored area. The images below shows the output of the system which after we run the program, will show live dection of the video input, and the save video file after the program completed the detection, we save it as a way to easier analyse and observe the detection through video playback, as analysing it live while running maybe not be as efficient.

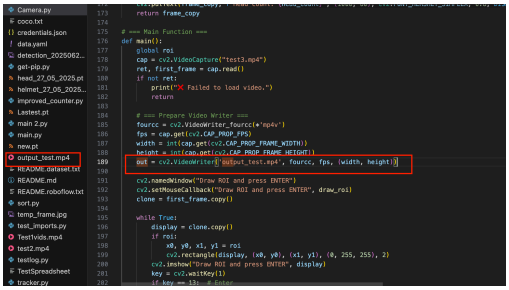

Figure 3.4.1: Saved output Video

Figure 3.4.2: Live Output Generation

3.4.4 Storing Output in Google Sheet API

After the output has been generated, we store the detection into a log using Google API from the video steam. The Google API integration is used for logging and evidence storage for helmet violation cases. Firstly as mention the two model, YOLOv8 pre-trained model, and custom model for head/helmet detection. How the system is works is, when a head is detected within the zone. The frames will be stored, using cv2 to snap-shot the specific frame of head detection. We log the detections events to Google Sheet, it records, the timestamp, head detection, alert, and link to Google Drive file to the captured image.

The credentials.json file can be obtained through Google API, the system will connect to pre-defined Google Sheet as we name them(Data log) and appneds the new rows with each detection events that occurs, the image are uploaded to Google Drive using the Drive API, and public access links are generated per log. This implementation allows users to store datas of the detection, this is to allow scalable and automated solution for

monitoring helmet violations. The result is as seen in the figure below.

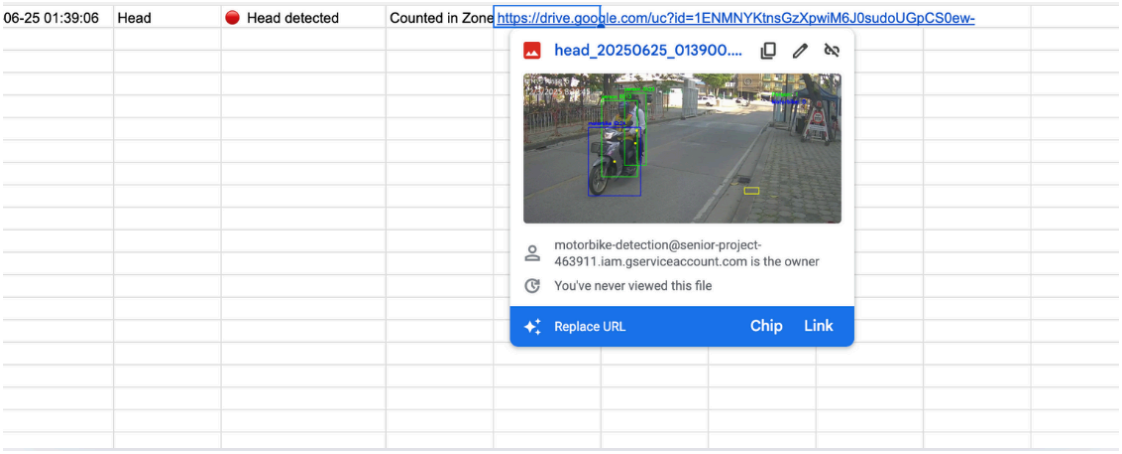

Figure 3.6

3.5 Tools and Frameworks

- **YOLOv8 Framework**: For object detection.
- **Libraries**: Python libraries such as ultralytics, opencv-python, numpy, collections.
- **Annotation Tools**: Roboflow.

3.6 Hyperparameter Configurations

During the training process, serveral key herparameters were carefufully selected to balace the performce, the model accuracy and accordingly to our local computation resrouces.

Epochs = 500, this is to ensure suffieicent exposeure to the training datasets. It will allow the model to learn patterns, and improve generalization overtime. However we can stop the early depending on our resources and depending on overfitting.

Image Size imgsz = 640 (image resolution) 640x640 pixel is most commonly use parameter. This will also help decrease the traning time. However for smaller objects like head and helmet detection. Training them on 640x640 is enough, however in some circumstances when training very far, helmet and head that are annotated are risk to losing fine-grained details. This is important as it is very essential for detecting small overlapping objects. For our case we annotate them after they pass the bumper, we have

checked that at this area before hand that when resolution was 640x640, it was not too grainy for traing. The image belows shows the annotated image that was resized to 640x640 for training.

Figure 3.4.1: Resized Image

Figure 3.4.2: Resized Image

Batch Size =32, this is due to primarily contrained by the GPU, as larger batches can lead larger samples, and faster convergency. Nonetheless, batch size 32istill alowws for reasonable effieicnt traning and consistent model updates. This value can be changed depending on the GPU memory. While a larger batch size could have potentially accelerated training and stabilized gradient updates, our hardware limitations required a compromise at 32 images per batch.

Learning Rate = 0.005. Setting it to this value helps to cacilitate steady learning without causing instability. If it was too high, it could result in model to overshoot the optimal solution during traning. This value is also recommended by YOLOv8.

3.7 Performace Evaluation

The trained YOLOv8 model was evaluated using: mAP: To quantify detection accuracy. Confusion matrix: To identify true positives, false positives, and negatives. The purpose of the performance evaluation is to help our team, look at the development of the different models. The benefits of finding the best model is to use them for the best accuracy for the counting systems.

Metric	all	Helmet	head
Images	814	622	321
Instances	1084	715	369
Precision (P)	**0.901**	0.934	0.869
Recall (R)	**0.862**	0.913	0.811
mAP@0.50	**0.924**	0.959	0.889
mAP@0.50:0.95	**0.497**	0.512	0.482

Figure 3.7

The evaluation results in figure 3.7 demonstrate that the best custom YOLOv8 model performs strongly in detecting both helmet and heads(non-helmet), with particularly high accuracy in helmet detection. Overall, the model achieved a precision of 0.901 and recall of 0.862 across all classes, indicating reliable and consistent predictions. For the helemt class specifically, the model attained satisfying performance with a precision of 0.934, recall of 0.913, and a high mAP@0.50 of 0.959. This suggests that the model can effectively identify helmets with minimal false positives and false negatives. In comparison, the head class yielded slightly lower results, with a precision of 0.869, recall of 0.811, and mAP@0.50 of 0.889, reflecting more variation or complexity in detecting uncovered heads. The overall mAP@0.50:0.95 score of 0.497, while moderate, still indicates acceptable performance under stricter localization conditions. These results affirm the model's suitability for real-time helmet usage monitoring, particularly in scenarios where accurate helmet detection is critical.

Metric	Head
Precision (all)	0.871
Recall (all)	0.859
mAP50 (all)	0.891
Helmet Precision	0.867
Helmet Recall	0.885
Helmet mAP50	0.907
Helmet mAP50-95	0.486
Head Precision	0.876
Head Recall	0.833
Head mAP50	0.875

Figure 3.8

Figure 3.8 shows the previous model performace evaluation, which is call Head_model. In comparing the performance of the updated helmet and head detection model, which is figure 3.7, to the previous version, several improvements are evident.

The current model achieves an overall precision of 0.901 and recall of 0.862, both higher than the previous model's precision of 0.871 and recall of 0.859, suggesting better accuracy and detection coverage. Helmet detection has particularly improved, with the current model reaching a helmet precision of 0.934, recall of 0.913, and mAP@0.50 of 0.959, outperforming the previous model's respective values of 0.867, 0.885, and 0.907. Head detection has also seen slight gains, with the current model achieving 0.869 precision, 0.811 recall, and 0.889 mAP@0.50, compared to 0.876, 0.833, and 0.875 in the earlier version. Additionally, the current model maintains a solid mAP@0.50:0.95 of 0.497 overall, indicating better localization performance under stricter IoU thresholds. These enhancements reflect a more robust and reliable detection system, making the current model better suited for real-world deployment in helmet compliance monitoring scenarios.

CHAPTER 4
RESULT

We have used pre-recorded footage from CCTV cameras to analyze and compare the performance and accuracy of the detection and counting system.

To evaluate the accuracy of the model, we compared detection counts with manual counting over a ten-minute span of one video with every model we have. The aim of this is to see the accuracy and improvement of our model. The results in Table 4.1 shows the accuracy results of different models. Person and motorbike used YOLO pretrain model to detect these two classes, so the accuracy and considered constant throughout different models. Helmet detection rises from 42% to 96%, which is 54% improvement from the first model to latest model. The calculation method of finding helemt detection accuracy across different models is to find the total ground truth and detection of the train model, then divide it to find the accuracy. We find the best confident for each model by comparing confident of 0.15 and 0.3 to find the best accuracy. Best17 model have the ground truth helmet of 60, we have the confident set to 0.15 and detected the detection number to be 55, which is 90%. On the other hand, we change the confident to 0.3 and the results of the detection is 58, which is 95% resulting in a better accuracy.

After evaluating the model's detection accuracy, we proceeded to assess the performance of the counting system. This analysis was conducted using two pre-recorded CCTV footage videos. Tables 4.2 and 4.3 present the counting accuracy for helmeted and non-helmeted (head) riders, derived from Video 1 and Video 2 respectively. These tables utilize a double-line counting method.

For the helmet and head detection specifically, we experimented with adjusting the length of the double-counting lines to evaluate their impact on counting accuracy. For instance, we tested line lengths such as 300 pixels for Line X and 450 pixels for Line Y, along with several other variations. The results indicated that optimal performance oc-

curs when the length of the double lines is kept within approximately 150 pixels. Longer lines tended to increase the likelihood of multiple or false counts due to prolonged object overlap with the counting zone.

In contrast, Tables 4.4 and 4.5 focus on motorbike and person counting accuracy, also based on video 1 and video 2. However, these use the BotSort tracking and counting method instead of the double-line approach. This separation allows us to compare the performance of different counting techniques across differents object categories and scenarios.

To evaluate counting accuracy, we use a relative error formula to calculate the percentage difference between the counted value and the ground truth. This approach provides a clear measure of overcounting or undercounting, allowing us to identify and analyze overflow or missed counts in the system.

4.1 Model Result

Table 4.1

Model Name	Person	Motorbike	Helmet
BestV8	87%	96%	42%
Best21			72%
Helmet_model			82%
Head_model			86%
Best17			95%

The figure presents the detection accuracy of various models across three object categories: person, motorbike, and helmet. The accuracy values for person and motorbike detection remain constant across models—87% and 96% respectively—because these classes were detected using the pretrained YOLO model without additional fine-tuning. In contrast, the helmet detection results vary significantly between models, as each represents a different version of a custom-trained helmet detection model.

4.2 Counting Result

Table 4.2

	Double Line Counting		Ground Truth	
Video 1	Head	Helmet	Head	Helmet
	25	26	25	25

Table 4.2 shows the counting results for Video 1 using the double-line method compared to the ground truth. The system accurately counted heads (25 vs. 25) but slightly overcounted helmets (26 vs. 25), likely due to duplicate detections or crossing artifacts.

Table 4.3

	Double Line Counting		Ground Truth	
Video 2	Head	Helmet	Head	Helmet
	29	24	32	29

Table 4.3 shows the double-line counting results for Video 2. The system counted 29 heads and 24 helmets, while the ground truth recorded 32 heads and 29 helmets. This indicates undercounting in both categories, which could be due to missed detections or rapid movements causing objects to skip the counting lines.

Table 4.4

	BotSort Counting		Ground Truth	
Video 1	Head	Helmet	Head	Helmet
	25	26	25	25

Table 4.4 presents BotSort counting results for Video 1. The system detected 52 persons and 32 motorbikes, compared to the ground truth values of 51 persons and 30 motorbikes. This reflects a small overcount in both categories, likely due to brief tracking ID switches or overlapping detections in crowded scenes.

Table 4.5

Video 2	BotSort Counting		Ground Truth	
	Head	Helmet	Head	Helmet
	29	24	32	29

Table 4.5 summarizes the BotSort results for Video 2. The model counted 56 persons and 44 motorbikes, while the ground truth showed 60 persons and 34 motorbikes. This indicates slight undercounting of persons and significant overcounting of motorbikes, which may result from false positives or tracking drift over longer video durations.

4.3 Discussions

4.3.1 Model Improvement

The comparison table highlights the performance improvement of various helmet detection models, while person and motorbike detection accuracies remain constant across all models at 87% and 96%, respectively. This consistency is expected, as all models use the original YOLOv8 architecture for detecting persons and motorbikes. The main focus of model optimization lies in helmet detection, which has undergone several iterations of training using custom datasets. The earliest version, BestV8, achieved only 42% accuracy in helmet detection, likely due to limited training data or suboptimal labeling. This was significantly improved in Best21, which reached 72% accuracy. Subsequent models, such as Helmet_model and Head_model, showed further improvements to 82% and 86%, respectively, indicating better dataset quality and fine-tuning strategies. The most refined version, Best17, achieved a 95% helmet detection rate, demonstrating the effectiveness of progressive model training, class balancing, and more consistent annotation.

4.3.2 Helmet and Head Double Line Counting

In the first video which is table 4.2, the accuracy of our helmet and head detection are very precise, since the video displays a clear bounding box of head and helmet detection and no overlapping with other head and helmet. Table 4.2 showing the accuracy of

non-helmet to be at 100%, but there are minor flaws in helmet accuracy as it exceed the ground truth. The reason there are 26 helmets being displayed from the counting system is because the person who is in the front often blocks out the person who is in the back and this can cause false detection, classifying people who are not wearing a helmet as wearing it.

In the second video which is table 4.3, the accuracy went down compared to the first video. The reason is that there are many occasions where multiple motorbikes come through at the same time, causing confusion of our model and causing many overlapping bounding boxes of head and helmet. This makes the accuracy of helmet detection to 82% and non-helemt to 90%.

4.3.3 Person and Motorbike BotSort Counting

Table 4.4 shows the accuracy of person detection to be 98% and motorbike accuracy to be 93%. this result is being tested on video 1, which is the video where overlapping hardly appear, making the result satisfying.

Table 4.5 which is being tested on the second video where this video overlapping appears often more than the first video resulting in lower accuracy comparing to the first video. As person accuracy drop down to 93% and motorbike drop to 77%. Occasionally motorbike with two person on it can cause confusion in bounding boxes and it can appear to detect one person instead of two person, since the people who is driving and riding is really close to each other. Our BotSort method works well with less overlapping of bounding boxes, and as the more overlapping occur, it causes a confusion of unique ID tracking system when it is in the counting zone. Causing a new unique ID to be create eventhough it is consider the same person or motorbike proceeding throughout the frame.

CHAPTER 5
CONCLUSION AND DISCUSSION

This project presents a helmet and head detection system trained on a custom dataset using the YOLOv8 architecture, combined with a counting mechanism applied to university CCTV footage. The final model achieved significant accuracy improvements in helmet detection, reaching up to 95% in our latest version. The double-line counting method effectively reduced duplicate counts and stabilized results, especially for slow-moving or occluded objects. For person and motorbike tracking, the system performed well in low-overlap conditions, providing strong results that demonstrate the system's potential for practical monitoring and analysis of helmet usage compliance. As for the counting for motorbike and person class, we can implement Bot_sort tracker as their Yolo model are much for refine allowing us to count regularly through trackers.

5.1 Obstacles

During system development and testing, the main challenge involved dealing with overlapping bounding boxes, particularly when multiple motorcycles and riders appeared close together. This sometimes led to incorrect object counts or detection errors, such as merging two people into one or misidentifying a detection. The tracking system also struggled when many objects entered the counting zone at once, occasionally creating new IDs for the same object. These issues were more pronounced in crowded scenes and highlight the limitations of relying on bounding box-based tracking in complex real-world environments.

5.2 Future Work

In future iterations, the focus will be on improving both the detection model and the tracking accuracy. Training with more diverse and balanced data will help the model generalize better across various environments and video conditions. Further enhancement

of the counting and tracking logic is also planned to address issues with ID switching and occlusion. Additionally, while deployment to the university server is not yet complete, it remains a key objective for real-time monitoring in a practical setting. These improvements will support the goal of developing a more accurate, stable, and scalable helmet compliance monitoring system.

REFERENCES

[1] F. S. R. B. R. N. K. R. G. S. S. A. Bharadwaj. Smart traffic management system for efficient mobility and emergency response. *2024 International Conference on Knowledge Engineering and Communication Systems (ICKECS)*, page 3, 2024. doi: 10.1109/ICKECS61492.2024.10616800.

[2] S. F. S. A. et al. Utilizing image processing and the yolov3 network for real-time traffic light control. *Journal of Engineering (ISSN: 2314-4912)*, pages 1--6, 2023. doi: -.

[3] H. Nie, H. Pang, M. Ma, and R. Zheng. A lightweight remote sensing small target image detection algorithm based on improved yolov8. *School of Computer Science and Engineering, Changchun University of Technology*, pages 1--3, 9, 2024.

[4] H. S. Qiang Zhang, Ziming Sun. Research on vehicle lane change warning method based on deep learning image processing. *Sensors, published by MDPI*, pages 1,5,9,11--13, 2022. doi: -.

[5] N. P. M. S. K. A. M. G. B. S. V. V. S. S. S. P. G. K. Reddy. Traffic detection and enhancing traffic safety: Yolo v8 framework and ocr for violation detection using deep learning techniques. *2024 International Conference on Science Technology Engineering and Management (ICSTEM)*, page 3–5, 2024. doi: 10.1109/ICSTEM61137.2024.10560904.

[6] M. V. Yashina, Y. A. Akilin, V. V. Osada, and P. G. Serov. Video image recognition of car track characteristics at intersections. pages 1--4, 2024. doi: 10.1109/IEEECONF60226.2024.10496754.