IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2022.Doi Number

TRAFFIC VIOLATION DETECTION USING DRONE

Hadid Candra Diputra[1], Muhammad Iqbal Dharmawan[2], Zalva Ihilani Pasha[3], Casi Setianingsih[4], and Anggunmeka Luhur Prasasti[5]
School of Electrical Engineering, Telkom University, Bandung, Indonesia

hadidcandra@student.telkomuniversity.ac.id[1], iqbalqibal@student.telkomuniversity.ac.id[2], zalvapsh@student.telkomuniversity.ac.id[3], setiacasie@telkomuniversity.ac.id[4], anggunmeka@telkomuniversity.ac.id[5]

ABSTRACT The rising complexity of traffic conditions in Indonesia has led to increased safety risks due to traffic violations. Traditional methods of monitoring, such as stationary CCTV cameras, face limitations in coverage and mobility, resulting in undetected violations. The study proposes a drone-based traffic violation detection system that combines visual sensing technology and artificial intelligence (AI) to address this. By integrating drones into traffic monitoring, the system aims to overcome the limitations of stationary cameras. The system employs object detection and optical character recognition (OCR) technologies to identify violations like not wearing helmets and stopping at zebra crossings. AI algorithms, trained to recognize patterns of traffic violations from image data, support the decision-making process. The system's testing phase demonstrated promising results. The combined technologies successfully identified various violations with satisfactory accuracy. The model training involved a dataset split of 88% for training and 12% for validation/testing. The model achieved a precision of 0.742, recall of 0.798, F1-score of 0.768, mAP@[0.5] of 0.81, and mAP@[0.5:0.95] of 0.592. Application-wise, the system achieved accuracy rates of 90.51% for no-helmet violations, 69.1% for detecting brief stops on zebra crossings, and 11.92% for OCR-based license plate recognition. Field testing demonstrated the system's efficacy in capturing previously challenging violations to monitor, showcasing the potential of drones in enhancing traffic safety and law enforcement. The proposed drone-based traffic violation detection system offers an innovative approach to address the limitations of conventional methods. It holds promise for improving future traffic safety and law enforcement efforts.

INDEX TERMS Drone, Object Detection, Optical Character Recognition, Traffic Violation Detection

I. INTRODUCTION

Amid the rapid advancement of technology, various aspects of life are also evolving. One of these aspects pertains to the realm of law enforcement. Recently, Indonesian law enforcement agencies have begun incorporating technological advancements, utilizing CCTV cameras extensively, particularly at traffic lights. However, new technology has recently been explored involving drones as assistants or even substitutes for CCTV cameras [1].

Simultaneously, the current systems heavily rely on human labor to operate and manage them. Capturing images and videos using CCTV cameras still needs to be solved, such as unclear license plate visibility due to low camera resolutions. While CCTV cameras are used for image and video capture, the identification process for traffic violators remains manual, involving human personnel behind the screens.

Drones, or uncrewed aerial vehicles (UAVs), are pilotless aircraft. They are controlled either automatically through programmed computer systems or remotely by pilots on the ground or in other vehicles. The pilot controls the drone remotely using a connected remote control. Equipped with cameras, drones allow users (pilots) to monitor and record from aerial viewpoints. These drones operate on battery power [2].

Our research collected data on riders and motor vehicles for our dataset. Our developed program utilizes deep learning technology to detect various forms of violations. The program is trained using training datasets. This program aims to streamline the identification process of violations committed

by riders, as the current process is manual and susceptible to intentional or unintentional errors during identification.

The primary objective of using drones as image and video capture devices in our project is to capture areas that are not accessible or viewable by CCTV cameras. CCTV cameras suffer from mobility limitations, restricting their movement to a fixed location with only the capability to adjust camera angles. Many riders try to evade detection by CCTV cameras, attempting to outsmart the system by staying behind other vehicles or covering parts of their license plates to avoid electronic fines. Being mobile and capable of detecting the rear lines of queued vehicles at traffic lights, drones can minimize attempts to deceive the system by hiding and covering license plates.

Drones are expected to address the challenges faced when using CCTV cameras. Drones have advanced significantly, boasting high-resolution cameras that enhance the accuracy of deep learning-based violation detection. Drones can aid law enforcement agencies in identifying individuals attempting to outwit the system. Detection using drones presents various challenges, such as video quality, drone movement, captured subjects, flight location, and altitude [3].

II. RELATED WORKS

According to a survey conducted by Fact Checker, traffic violations account for 80% (117,914) of all road accident fatalities, totaling around 323 deaths per day. Among these cases, nearly 36% are delayed in the trial phase due to insufficient evidence, arguments, and assessments, while approximately 61% await trials because of a lack of substantial evidence [4]. Enhancing safety necessitates improvements in driver behavior, achievable through monitoring devices. Research indicates that drivers frequently involved in serious accidents often have clear traffic violations [5]. Drone-based electronic traffic law enforcement (ETLE) trials have commenced in the Central Java Regional Police. The mechanism captures riders deemed to have committed traffic violations, followed by validation before issuing fines [6].

Tracking targets with small drones has been undertaken using various methods, categorized into visual and non-visual trackers. Visual tracking by small drones involves video, where various deep learning-based trackers are compared with camera motion models. Real-world testing using UAV-based videos reveals that small objects, many targets, and camera motion can degrade tracking performance, even when using high-end CPUs. Deep learning-based object detectors and trackers require significant computation with massive training data, often posing real-time processing challenges to be addressed [7].

In an experiment, a drone flew at a constant speed of 5.1 m/s at an altitude of 150 m while recording video clips of nine moving targets. Detection rates, false alarm rates, and receiver operating characteristic curves were obtained, and displacement vectors estimated the drone's speed in the x and y directions. Average detection rates ranged from 90% to 97%, while false alarm rates ranged from 0.06 to 0.5. The root mean square error of speed was 0.07 m/s when the reference frame was fixed, indicating the robustness of the proposed method [8].

Most training samples are derived from 578 drone videos from popular video services. The complete training dataset consists of 51,446 images downsampled from various resolutions (ranging from 640x480 to 4K) to 640x480 resolution, where 51,445 images contain a total of 52,676 drone bounding boxes and one negative image (containing no UAVs). For object detection tasks, most input images are negative examples (not containing the intended class), so adding negative data would not provide added value to the model. In the training dataset, there are more small objects than large objects. Specifically, around 40.8% are small objects (area < 1024), 35.8% are medium-sized (1024 < area < 9216), and 23.4% are large-sized (area > 9216), as defined by the COCO challenge. Overall, this diverse large dataset aims to present drones of various types, sizes, scales, positions, environments, times of the day, etc., allowing for varied representations to train object detection models [9].

UAVs possess the advantages of high maneuverability, small size, and sensitive operation. They are extensively used in transportation, military, scientific research, and other fields. The development of computer and image recognition technology has matured moving target detection technology, and using unmanned aerial vehicles can elevate the capabilities of moving target detection technology to a higher level [10]. UAVs will fly over the area, endeavoring to collect as much data as possible, evaluating the coverage rate of vehicles, the percentage of abnormal events (congestion and violations), and the duration of detected events [11].

III. SOLUTION DESIGN

A. YOU ONLY LOOK ONCE (YOLO)

The YOLO (You Only Look Once) methodology, based on Convolutional Neural Networks (CNNs) as illustrated by Redmon et al. (2016), represents an object recognition approach. YOLO stands out for its simplicity, as depicted in Figure 3.1. With a single convolutional network, YOLO simultaneously predicts multiple bounding boxes and class probabilities for each box. It employs entire images during training, thereby directly enhancing object detection performance. Thus, this integrated model offers several advantages over traditional object detection methods. Notably, YOLO's strength lies in its speed. The basic version can achieve 45 frames per second without requiring batch processing on specific hardware like a Titan X GPU. A faster version can exceed 150 frames per second, enabling real-time video processing with less than 25 milliseconds of latency. More importantly, YOLO achieves over twice the average precision of other real-time systems [12].

The fundamental principle of YOLO involves dividing an image into an s x s grid for object detection. In each grid, a bounding box anticipates object presence, computing the confidence value indicating how well the box contains an object according to planning and projection. The equation expresses this value in Figure 3.2. The confidence score is crucial for accurate object detection. This approach facilitates considering all objects in the image rather than concentrating on specific areas. This enables YOLO to achieve real-time object detection with impressive precision. Moreover, this method is responsive to object scale and orientation variations, making it effective for various object detection applications [13].

YOLO considers the Intersection of Union (IOU) value to enhance object detection accuracy. IOU is the overlap ratio between predicted and ground truth boxes. Higher IOU values correspond to higher object detection accuracy, as the equation diagram illustrates. Each YOLO bounding box consists of five variables: x, y, w, h, and c. x and y represent the central coordinates of the detected object's bounding box. The width (w) and height (h) of the bounding box are depicted by w and h variables. The last variable, c, represents the confidence of the bounding box, indicating the system's confidence that the box contains the sought-after object. Thus, the IOU concept and bounding box structure enhance YOLO's object detection effectiveness and reliability [13].

In YOLOv7, the technique of bag of freebies is employed for training. This technique encompasses modules and optimization methods aimed at simultaneously improving object detection accuracy, training cost, and inference cost. YOLOv7 utilizes a Rough to Fine Label Assignment instead of a conventional Dynamic Label Assignment, enhancing model training with multiple output layers. Efficiency in the convolutional layers of the YOLO network is vital for efficient inference speed. Cross Stage Partial Networks are introduced by WongKinYiu to maximize the efficiency path, considering memory requirements and gradient propagation distances. A shorter gradient result in a stronger learning network [14].

The YOLOv7 architecture comprises the Backbone, Neck, and Head. The ELAN and E-ELAN models are employed in the Backbone as convolutional neural networks that aggregate image pixels to form features at various granularity levels. Backbone models are typically pre-trained on classification datasets. Incorporating additional layers in the architecture produces different features with higher semantics. The Neck employs the CSPSPP+(ELAN, E-ELAN)PAN model as an additional layer to extract feature maps from various stages in the Backbone before they proceed to the Head. The Head employs the YOLOR model for predicting bounding boxes and classifications [15]. Performance parameters in YOLO are divided into four (4) categories.

1) PRECISION

Precision represents the fraction of true positive detections among all positive detections. It quantifies the quality of detection, that is, how accurately predicted objects are. Precision is calculated using the following formula.

$$Precision = \frac{TP}{TP + FP}$$

Explanation:
TP: True Positive
FP: Fake Positive

2) RECALL

Recall signifies the fraction of true positive detections among all ground truth objects. It measures the completeness of detection, i.e., how many actual objects are detected. Recall is computed using the following formula.

$$Recall = \frac{TP}{TP + FN}$$

Explanation:
TP: True Positive
FP: Fake Positive

3) F1-SCORE

The F1-Score is the harmonic mean of precision and recall. It provides a balance between precision and recall and is a useful metric when classes are imbalanced. The F1-score has the following formula:

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

4) MEAN AVERAGE PRECISION (MAP)

mAP is a metric utilized to evaluate object detection models like YOLO. It represents the average of the Average Precision (AP) for each class, where AP is the area under the precision-recall curve. mAP gauges the overall performance of the model across all classes. The mAP formula is as follows:

$$\boldsymbol{mAP} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{N}} \frac{\boldsymbol{APi}}{\boldsymbol{N}} \times \mathbf{100\%}$$

Explanation:
Σ (AP): total of *Average Precision*
N: total of classes

B. LIBRARY

The frontend components of the system employ React JS, a JavaScript library known for its component-based architecture, efficient Virtual DOM utilization, and strong community support. Tailwind CSS, a versatile framework, facilitates the creation of custom designs through utility-first classes. React Router DOM streamlines web application navigation.

On the backend, argparse handles command-line arguments, while the io library manages I/O operations. PIL or Pillow is employed for image manipulation, datetime for time-related operations, and torch for optimized neural network development. cv2 supports image and video processing, numpy handles numerical operations, and Flask enables web app creation. Additional functionalities include regular expression handling via re, time management using time, JSON operations through JSON, math calculations using math, and OCR with by tesseract. Watchdog manages file and directory monitoring, and threading allows multithreading operations. The library supports queue data structures, while Flask-Cors manages CORS in Flask web apps. These libraries collectively contribute to the efficient implementation of the system's functionalities.

C. DRONE CONNECTION WITH HARDWARE

MonaServer is a multimedia server designed to provide easy and flexible real-time solutions. It supports protocols such as RTMFP, RTMP, RTMPE, WebSocket, and HTTP. OBS (Open Broadcaster Software) is open-source and free software for recording videos and live streaming. OBS has extensive support for various plugins and scripts and can be used for live broadcasting, screen recording, and more. One of these plugins is OBS Virtual Camera, which allows the OBS output to be used as a virtual webcam in other applications.

The design of using MonaServer and OBS starts by setting up MonaServer on the server side as the RTMP connection hub. MonaServer will function as an RTMP server that receives drone connections for streaming from OBS and broadcasts them to the web application. You must configure MonaServer settings, such as IP addresses and ports, to suit your needs. OBS will capture video from the drone's RTMP stream, process it, and send it to the web application. This means that whatever is displayed in OBS, including overlays, graphics, and more, will be shown in the web application.

FIGURE 1. MonaServer Illustration

D. MAIN ARCHITECTURE SYSTEM

The system will utilize input to retrieve data in the form of live video, which will be transmitted through communication devices and subsequently processed. The processing will be carried out by a Deep Learning program to classify the type of vehicle and categorize it, whether it is a car or a motorcycle. Subsequently, the classification of the violation type will be performed, and based on this data, it will be regrouped to generate a transcript report as the process output. The output results will then be directly sent back to the application used by the user.

FIGURE 2. Main Architecture System

E. WEB APPLICATION DESIGN

This product employs a simple design to ensure users can easily utilize this violation detection application. The web application is designed using Figma as the prototype tool. The user flow, or the sequence of user actions when using this product, is not complex. Users only need to use one feature that can be clicked to view the results of violations captured using the drone.

FIGURE 3. Website Interface

Furthermore, this application also provides various information such as the number of detected vehicles, drone flight location, drone system, and a simultaneous view of the drone, along with bounding boxes.

IV. IMPLEMENTATION

A. DATASET SUBSYSTEM

Figure 4 depicts the block diagram of the system, wherein the Roboflow web application plays a pivotal role in dataset management as showcased in the diagram. Roboflow processes and organizes image datasets, offering functionalities for automated annotation, data preprocessing, and augmentation. Specifically tailored for object detection with YOLO (You Only Look Once), the collected image dataset needs to be annotated to mark the locations of specific objects within the images. This annotation is crucial as it aids the model in learning to recognize the desired objects.

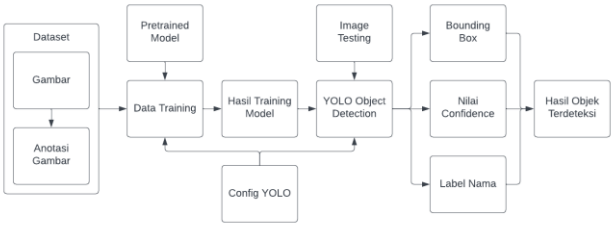

FIGURE 4. Block Diagram

In Roboflow, annotation involves labeling objects in images with bounding boxes and assigning labels to each bounding box. The resultant annotations are then stored in .txt files, containing the coordinates of each bounding box and the object label. These .txt files are used in the YOLO model training process. Leveraging Roboflow streamlines and enhances the data annotation process. Moreover, the application enables users to apply data augmentation techniques, diversifying the dataset and contributing to model performance improvement.

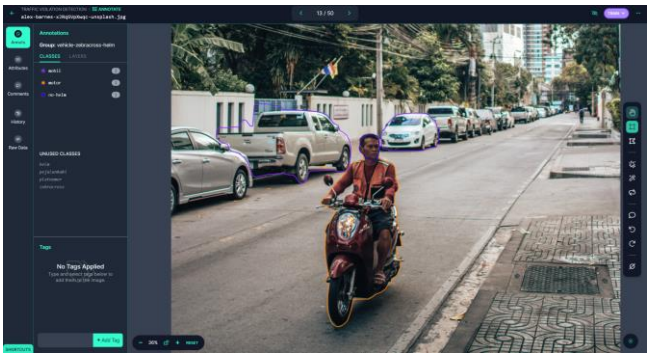

FIGURE 5. Image Annotation Process for Training

The annotated dataset is furnished with .txt files containing annotation coordinates for each image, annotated using the Roboflow web application. This dataset then proceeds to the training phase, utilizing pretrained YOLO models and configuration files. Upon completing the training process, YOLO generates a trained model file in the .pt format, serving as a custom model. When employing this model, YOLO detects objects according to the dataset, displaying bounding boxes, labels, and confidence values for the identified objects.

After the implementation, the dataset consists of 1000 photos with 7 classes encompassing various road scenarios. Data annotation was performed balanced, with annotation counts ranging from 150 to 200 in each set of 1000 photos. The dataset encompasses diverse object types such as zebra cross, helmets, no-helmet instances, cars, motorcycles, license plates, and pedestrians. Although challenges were encountered in classifying motorcycles and pedestrians, the dataset yielded a representative variety.

B. DETECTING VIOLATIONS

Figure 6 illustrates the flowchart of the stages to be executed within the system. Once the system is initiated, it detects potential violations. The outcome of this detection is a photograph capturing the violation.

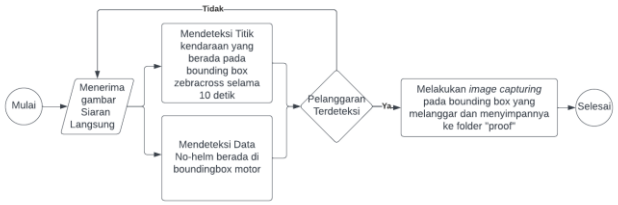

FIGURE 6. Detection Flow Diagram

The implementation phase of the violation subsystem involves extracting bounding box coordinates for the "no-helmet," "motorcycle," "car," and "zebracross" classes. Collected video data is employed for testing purposes, a step that validates the employed source code.

```
if class_name in ['no-helm']:
    center_x, center_y = (x1 + x2) // 2, (y1 + y2) // 2
    for motor_box in motor_boxes:
        zx1, zy1, zx2, zy2 = motor_box
        if zx1 <= center_x <= zx2 and zy1 <= center_y <= zy2:
            timestamp = int(time.time())
            image_filename = f"proof/{class_name}_{timestamp}.jpg"
            cv2.imwrite(image_filename, resized_img)

if class_name in ['motor']:
    print(f'Processing class: {class_name}')
    center_x, center_y = (x1 + x2) // 2, (y1 + y2) // 2
    offset_y = 30
    center_y += offset_y
    for zebracross_box in zebracross_boxes:
        zx1, zy1, zx2, zy2 = zebracross_box
        if zx1 <= center_x <= zx2 and zy1 <= center_y <= zy2:
            current_time = time.time()
            print(f'Object in zebracross at time {current_time}')
            if class_name not in timestamps_in_zebracross:
                timestamps_in_zebracross[class_name] = current_time
                print(f"perhitungan {current_time} - {timestamps_in_zebracross[class_name]}")
            elif current_time - timestamps_in_zebracross[class_name] >= 5:
                print('Object has been in zebracross for 3 seconds')
                image_filename = f"proof/{class_name}_{int(current_time)}.jpg"
                cv2.imwrite(image_filename, resized_img)
                del timestamps_in_zebracross[class_name]
```

FIGURE 7. Detecting Violations Source Code

The provided source code, depicted in Figure 7, serves this purpose. The code excerpt's objective is to gather information about riders not wearing helmets and stopping on the zebra cross line for a duration of 3 seconds. Following information processing, the experiment's results are presented in the form of testing analysis.

The code excerpt's objective is to gather information about riders not wearing helmets and stopping on the zebra cross line for a duration of 3 seconds. Following information processing, the experiment's results are presented in the form of testing analysis.

Then, testing is performed directly using the separate source code within the main application. This commences with the collection of traffic violation videos. These videos are individually assessed within the software code editor, monitoring the resulting image captures generated by the code.

Testing result shows that the system successfully detects violations at zebra cross crossings and instances of motorcyclists not wearing helmets. Zebracross violations are detected based on stopping for more than 5 seconds within the zebra cross area, while violations of motorcyclists without

helmets are detected by examining the intersection of motorcycle and no-helmet bounding boxes. The system's ability to detect and capture violations in real time facilitates swifter and more efficient law enforcement.

C. WEB-APP

Figure 8 showcases a code segment from a web-based application that utilizes machine learning and camera technology for traffic monitoring. This application uses React JS and Python, two powerful and popular programming languages. React JS, a JavaScript library developed by Facebook, is employed to construct the user interface. With React JS, the application delivers a dynamic and responsive user experience, enabling real-time traffic information updates on the application interface. On the other hand, Python serves as the application's backend, particularly for implementing machine learning algorithms. Python is a fitting choice due to its abundant libraries, including TensorFlow and PyTorch, which facilitate implementing and enhancing machine learning models.

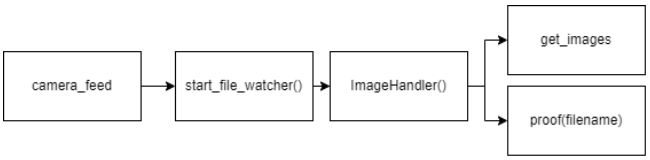

FIGURE 8. WEB-APP Diagram

For the implementation, In the "camera_feed" section, the system captures each frame from the video stream, performs object detection, and annotates the frame based on detection results. Following detection, the application initiates various actions depending on the detected objects. This function targets specific object categories such as 'license_plate,' 'no-helmet,' 'motorcycle,' 'car,' and 'zebracross.' Within the "route" segment, direct violation detection occurs. If a 'no-helmet' object is detected within a 'motorcycle' object, this function stores the image as evidence of the violation. Additionally, if a 'motorcycle' is detected within a 'zebracross' object for more than 5 seconds, the corresponding image is saved.

This code also initializes monitoring using the 'start_file_watcher()' function within the "image_folder" directory, activating the 'ImageHandler()' action upon photo additions in that directory. The 'start_file_watcher()' function creates an instance of the Observer class, which commences the monitoring process. Furthermore, the 'ImageHandler()' function serves as the file system event manager, encompassing the 'process_and_rename_number_plate()' function for license plate letter or digit detection. Upon ImageHandler instantiation, the system also initializes a queue and a new thread. This queue holds paths for each newly generated image, while the created thread processes the images asynchronously in a separate thread.

Two additional API paths are designated within the main code to communicate information to the frontend. These routes are /proof/path:filename and /get_images. The 'proof()' function retrieves and sends back files stored in the "proof" directory based on the provided filename parameter. Meanwhile, the 'get_images()' function generates a list of images stored on the server. This function searches for all .jpg files within the specified directory (image_folder), constructs complete URLs for each image based on the server address, and sends all this data back as a JSON response.

For testing, the testing involves detecting the position of points within vehicle bounding boxes. If a vehicle is positioned over a zebra cross for more than the specified time parameter, it undergoes analysis using YOLOv7 to determine if any violations are detected.

The testing result shows that the web application designs an intuitive and efficient user interface. The system architecture is designed to ensure robust and reliable performance. The process from inputting videos to generating notification outputs runs smoothly without interruption due to errors.

D. PLATE NUMBER DETECTION ON MOTOR VEHICLES

Figure 9 presents a flowchart detailing the stages of the system's operation. When the system is initiated, it attempts to detect objects already present in the dataset, and the outcome involves analyzing the results obtained from these objects.

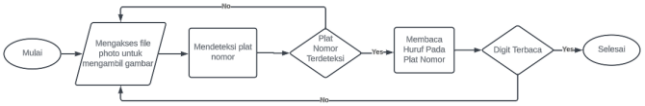

FIGURE 9. Plate Number Detection Diagram

The implementation of this sub-system commences with collecting several sample photos. The program will subsequently crop These photos with bounding boxes placed on them. Afterward, digits within the cropped images are read using the Pytesseract library to generate results. These results are used as file names for the output produced.

For the testing, license plate detection system is performed by inputting sample photos used to detect license plates. The system then detects and reads the license plate from the input photo.

The testing result shows that the system successfully detects license plates using YOLOv7 and reads digits from license plates with a reasonably high accuracy using OCR technology, particularly by Tesseract. The steps of cropping, color inversion, and contour identification assist in obtaining license plate digits.

E. FINAL SYSTEM

FIGURE 10. Final System

Figure 10 shows that a system that combines drone technology, web-based applications, and machine learning algorithms has been successfully implemented with satisfactory results. Core functionalities such as drone monitoring, image analysis, and real-time information dissemination operate efficiently and effectively. Despite challenges posed by weather and lighting conditions, the system maintains stable performance and high detection accuracy. The responsive and intuitive user interface facilitates effective traffic monitoring, while integration with databases and automatic reporting enhance process efficiency. The system has been developed and tested by technology and security standards, confirming the success and security of this implementation. Despite its success, the system still holds potential for improvement and development in the future.

V. TESTING

The YOLO (You Only Look Once) system parameter testing process will involve experimenting with various parameters and scenarios. Starting with determining the training and validation data ratios, we will explore ratios of 58%:42%, 68%:32%, 78%:22%, 88%:12%, and 98%:2%. For each ratio, the larger portion of data will be used for training the model and the smaller portion for validation. After data division, we will test the model using different learning rates: 0.001, 0.0001, 0.00001, and 0.000001. These learning rates will control how fast or slow the model learns from data. Next, the model will be tested with batch sizes: 2, 4, and 8. The batch size determines the number of samples processed before the model updates its internal parameters. Lastly, the model will be tested with varying epochs: 100, 200, and 300. Epochs determine how many times the model will see the entire training dataset. All these tests will be conducted in a stable computational environment, ideally on a computer with sufficient specifications to run deep learning models. Throughout these tests, performance metrics such as precision, recall, and F1-score will be monitored, aiming to achieve parameter combinations that yield the best results based on these metrics.

Then, for the system testing, a controlled and secure environment needs to be established to test the system. This can be an enclosed room or an open area with sufficient space for the drone to move at different heights, free from unwanted obstacles and interference. Subsequently, a series of testing scenarios will be designed to cover all variables of interest, including heights of 3 meters, 5 meters, and 7 meters, camera angles of 45, 60, and 90 degrees, and varying drone speeds.

A. DATA SCENARIO TESTING

In the first testing phase, we examined data ratios within the dataset. After performing the tests, we analyzed the results and obtained the following outcomes:

a. For the data ratio of 58%:42%, the average parameter value obtained was 0.573.
b. For the data ratio of 68%:32%, the average parameter value obtained was 0.5926.
c. For the data ratio of 78%:22%, the average parameter value obtained was 0.6032.
d. For the data ratio of 88%:12%, the average parameter value obtained was 0.7394.
e. For the data ratio of 98%:2%, the average parameter value obtained was 0.7086.

Based on these test results, it can be concluded that the highest average accuracy value was achieved with the data ratio of 88%:12% at a value of 0.7394. This observation is further supported by the bar chart in Figure 7, illustrating the data ratio testing outcomes.

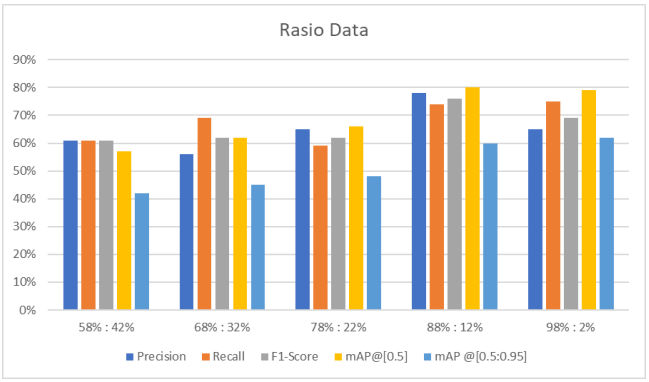

FIGURE 11. Bar Chart on Data Ratio Testing

B. LEARNING RATE SCENARIO TESTING

In the second testing phase, we examined different learning rates. After performing the tests, we analyzed the results and obtained the following outcomes:

a. For the learning rate of 0.001, the average parameter value obtained was 0.727.
b. For the learning rate of 0.0001, the average parameter value obtained was 0.742.
c. For the learning rate of 0.00001, the average parameter value obtained was 0.7394.
d. For the learning rate of 0.000001, the average parameter value obtained was 0.7182.

From these test results, it can be concluded that the best average accuracy value was achieved with a learning rate of 0.0001, yielding a value of 0.742. The bar chart in Figure 8

supports this observation, illustrating the learning rate testing outcomes.

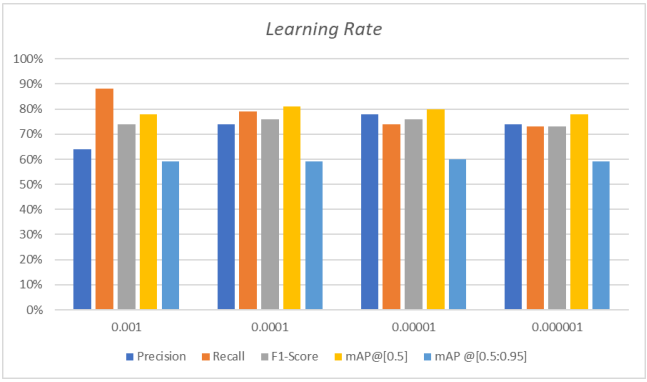

FIGURE 12. Bar Chart on Learning Rate Testing

C. BATCH SIZE SCENARIO TESTING

In the third testing phase, we conducted evaluations on different batch sizes. Following the completion of these tests, we proceeded to analyze the outcomes, yielding the following results:

a. The average parameter value obtained for the batch size of 2 was 0.704.
b. The average parameter value obtained for the batch size of 4 was 0.739.
c. The average parameter value obtained for the batch size of 8 was 0.742.

From the results of this testing, it can be deduced that the most favorable average accuracy value was achieved with a batch size of 8, resulting in a value of 0.742. This conclusion is depicted in the bar chart labeled Figure 9, illustrating the batch size testing results.

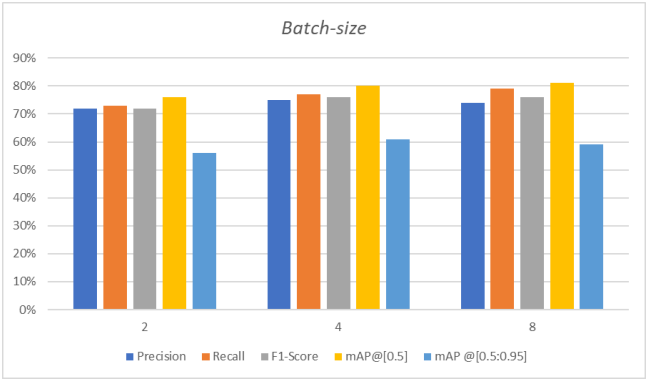

FIGURE 13. Bar Chart on Batch-Size Testing

D. EPOCH SCENARIO TESTING

In the fourth testing phase, we conducted evaluations on different epoch values. After performing these tests, we proceeded to analyze the obtained results, leading to the following observations:

a. For the epoch value of 100, the average parameter value obtained was 0.742.
b. For the epoch value of 200, the average parameter value obtained was 0.689.
c. For the epoch value of 300, the average parameter value obtained was 0.705.

Based on the test results, it can be concluded that the optimal average accuracy value was achieved with an epoch value of 100, resulting in a value of 0.742. This conclusion is visually represented in the bar chart labeled Figure 10, depicting the outcomes of epoch testing.

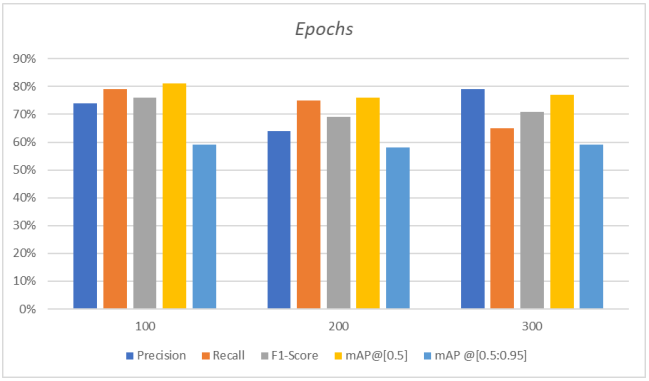

FIGURE 14. Bar Chart on Epoch Testing

E. OPTICAL CHARACTER RECOGNITION (OCR) TESTING

In the fifth testing phase, we conducted tests on OCR models, the first utilizing easyocr and the second employing tesseract. Based on the conducted tests, it can be concluded that digit recognition on license plates is more accurate and precise when using by tesseract. The testing was carried out by comparing the two OCR types using identical photos, allowing for more straightforward result comparison.

1) OCR TESTING USING EASYOCR

Based on the OCR testing using easyocr, as depicted in Table 1, several conclusions can be drawn. In the first trial, the OCR achieved a 100% accuracy rate, successfully recognizing all digits. However, in the second trial, the accuracy dropped to 50%, indicating that it correctly identified only half of the digits. The most notable observation is the third trial, where the OCR failed to detect any digits, resulting in a 0% accuracy rate. This performance degradation from the first trial to the subsequent ones suggests that the OCR's accuracy is inconsistent, and its reliability can be compromised in certain conditions.

Table 1. OCR Testing Using easyocr

	Total Testing	First Testing	Second Testing	Third Testing
Accuracy Score	3	100%	50%	0%
Error Score	3	0%	50%	100%

2) OCR TESTING USING PYTESSERACT

Based on the OCR testing using Pytesseract, as presented in Table 2, several conclusions can be derived. In the first trial, ytesseract demonstrated a perfect % accuracy rate of 100%, accurately identifying all digits. Although in the second trial, the accuracy slightly decreased to 87.5%, indicating successful identification of 7 out of 8 digits. Even in the third trial, where the accuracy was further reduced to 85.7%, Pytesseract recognized 6 out of 7 digits. These consistent and relatively high accuracy rates suggest that Pytesseract performs consistently well across different trials, making it a more reliable option than easyocr for digit recognition in license plates.

Table 2. OCR Testing using Pytesseract

	Total Testing	First Testing	Second Testing	Third Testing
Accuracy Score	3	100%	87,5%	85,7%
Error Score	3	0%	12,5%	14,3%

F. System Testing

In the comprehensive system testing, various parameters were examined across different conditions, yielding valuable insights as follows:

1) HEIGHT 3M

The system's performance was evaluated at a height of 3 meters with different camera angles. The analysis presented in Table 3 showcased varying accuracy rates. Notably, at a camera angle of 90 degrees, the detection system struggled with object recognition, yielding low accuracy rates. Moreover, the OCR for license plates faced challenges due to image clarity issues.

Table 3. Accuracy Score 3m Height

Height	Camera Angle	Accuracy Score		
		Helm	Zebracross	License Plate
3	45 °	100%	66.7%	16.7%
3	60 °	100%	83.4%	16.7%
3	90 °	33.4%	16.7%	0%

2) HEIGHT 5M

Testing at a height of 5 meters, as shown in Table 4, highlighted the system's difficulty in recognizing objects, particularly at a camera angle of 90 degrees. The OCR for license plates also encountered difficulties due to image clarity and height-related challenges, leading to an accuracy of 0% for license plate recognition.

Table 4. Accuracy Score 5m Height

Height	Camera Angle	Accuracy Score		
		Helm	Zebracross	License Plate
5	45 °	83.4%	83.4%	0%
5	60 °	83.4%	66.7%	0%
5	90 °	33.4%	50%	0%

3) HEIGHT 7M

At a height of 7 meters, detailed in Table 5, similar challenges persisted in object recognition, especially at a camera angle of 90 degrees. The OCR for license plates faced image clarity and height-related limitations, resulting in zero license plate recognition accuracy.

Table 5. Accuracy Score 5m Height

Height	Camera Angle	Accuracy Score		
		Helm	Zebracross	License Plate
7	45 °	66.7%	50%	0%
7	60 °	33.4%	50%	0%
7	90 °	16.7%	16.7%	0%

4) DRONE SPEED

The testing at different drone speeds, as shown in Table 6, demonstrated accuracy fluctuations. However, the overall accuracy rates were affected by the system's live streaming frame rate limitations of 30-60fps, leading to less accurate data.

Table 6. Drone Speed Accuracy Score

Camera Angle	Accuracy Score		
	Helm	Zebracross	License Plate
0.7-1.1	83.4%	33.4%	16.7%
1.4-1.8	66.7%	16.7%	0%
3.2-4	16.7%	16.7%	0%

5) LIGHT INTENSITY

The analysis of light intensity testing results, outlined in Table 7, indicated that higher lux values are conducive to better object classification accuracy. Nonetheless, high light intensity could result in shadows obstructing objects, leading to challenges in detection.

Table 7. Light Intensity Accuracy Score

Camera Angle	Accuracy Score		
	Helm	Zebracross	License Plate
7536	83.4%	66.7%	16.7%
17644	100%	83.4%	16.7%
2251	33.4%	50%	0%

In conclusion, the comprehensive system testing revealed that the system's performance is influenced by height, camera angle, drone speed, and light intensity. While the system achieved acceptable accuracy rates in certain scenarios, such as object recognition difficulties and image clarity issues were particularly prominent at higher heights and varying light

conditions. These findings underscore the need for further optimization to enhance the system's accuracy and robustness under diverse operational conditions.

VI. CONCLUSION

In summary, the comprehensive testing conducted on the proposed drone-based traffic violation detection system yielded valuable insights into its performance under diverse operational conditions. The evaluations were carried out considering variables such as height, camera angle, drone speed, and light intensity. The results, as presented in Tables 3 to 7, revealed significant findings. At a height of 3 meters, the system showcased satisfactory accuracy for helmet and zebra crossing detection, with decreasing accuracy as the camera angle increased. However, challenges were more pronounced at heights of 5 and 7 meters, particularly in recognizing objects and license plate characters, resulting in lower or zero accuracy scores, especially at a 90-degree camera angle. Additionally, the influence of drone speed on accuracy was evident, with the optimal range of 0.7 to 1.1 m/s yielding the highest accuracy rates. Light intensity also played a crucial role, with higher lux values contributing to better object classification accuracy, albeit excessive light potentially causing shadow-related issues.

The outcomes of this research underline both the potential and limitations of the proposed system. While achieving commendable accuracy rates in certain scenarios, the system faced challenges under varying conditions, highlighting the importance of ongoing optimization to enhance its robustness and precision. This study's significance lies in its capacity to revolutionize traffic violation detection methods through the integration of drone technology and AI algorithms. By addressing the shortcomings of traditional traffic monitoring approaches, this system offers broader coverage, real-time data processing, and increased mobility, ultimately contributing to heightened traffic safety and more effective law enforcement practices. The insights garnered from this research can provide law enforcement agencies with valuable tools to better detect and manage traffic violations, thus fostering safer road environments and greater compliance with traffic regulations.

REFERENCE

[1] Ridwan Arifin, (Oktober 2022). “Polisi kini bisa tilang dari udara pakai drone pelanggaran apa yang diincar”. https://www.oto.detik.com/
[2] Surti Risanti, (Agustus 2022). “Drone: Pengertian, jenis, dan fungsinya”. https://www.fortuneidn.com/.
[3] Shanthi.K.G, Sesha Vidhya S, Vishakha K, Subiksha S, Srija.K.K, Srinee Mamtha.R. Algorithms for Face Recognition Drones, June 2021
[4] Sugam Dembla, Niyati Dolas, Ashish Karigar, Dr. Santosh Sonavane. Machine Learning based Object Detection and Classification using Drone. International Research Journal of Engineering and Technology (IRJET), Volume: 08 Issue: 06, Juni 2021.
[5] Nourdine Aliane, Javier Fernandez, Mario Mata, Sergio Bemposta. A System for Traffic Violation Detection. sensors : www.mdpi.com/journal/sensors .
[6] Serafina Ophelia, (November 2022). “Polisi mulai uji coba tilang elektronik pakai drone”. https://www.kompas.com/.
[7] Seokwon Yeom, Don-Ho Nam. Moving Vehicle Tracking with a Moving Drone Based on Track Association. Applied Sciences: April 2021.
[8] Donho Nam, Seokwon Yeom. Moving Vehicle Detection and Drone Velocity Estimation with a Moving Drone. International Journal of Fuzzy Logic and Intelligent Systems: March 2020.
[9] Maciej Ł. Pawelczyk, Marek Wojtyra. Real World Object Detection Dataset for Quadcopter Unmanned Aerial Vehicle Detection. Institute of Electrical and Electronics Engineers (IEEE): August, 2020.
[10] Sining Cheng, Jiaxian Qin, Yuanyuan Chen, Mingzhu Li. “Moving Target Detection Technology Based on UAV Vision”. Juli 2022,https://www.hindawi.com/.
[11] Mouna Elloumi, Riadh Dhaou, Benoît Escrig, Hanen Idoudi, Leila Saidane. Monitoring road traffic with a UAV-based system.HAL Open Science : Mei 2019.
[12] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015). You Only Look Once: Unified, Real-Time Object Detection. http://arxiv.org/abs/1506.02640
[13] Sarosa, M., & Muna, N. (2021). IMPLEMENTASI ALGORITMA YOU ONLY LOOK ONCE (YOLO) UNTUK DETEKSI KORBAN BENCANA ALAM. 8(4). https://doi.org/10.25126/jtiik.202184407
[14] Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. http://arxiv.org/abs/2207.02696
[15] Jacob Solawetz. (2022, July 17). What is YOLOv7? A Complete Guide. https://blog.roboflow.com/yolov7-breakdown/