Journal of Information Systems and Informatics
Vol. 6, No. 4, December 2024 e-ISSN: 2656-4882 p-ISSN: 2656-5935
DOI: 10.51519/journalisi.v6i4.912 Published By DRPM-UBD

Development of a Real-Time Traffic Density Detection Website Using YOLOv8-Based Digital Image Processing with OpenCV

Rizki Juliansyah[1], Muhammad Aqil Musthafa Ar Rachman[2], Muhammad Al Amin[3], Aisya Tyanafisya[4], Nurrizkyta Aulia Hanifah[5], Endang Purnama Giri[6], Gema Parasti Mindara[7]

[1,2,3,4,5,7]College of Vocational School IPB University, Indonesia
[6]School of Data Science, Mathematics and Informatics IPB University, Indonesia
Email: [1]riskijhsijuliansyah@apps.ip.ac.id, [2]agilarrachman@apps.ipb.ac.id,
[3]muhammadalamin@apps.ipb.ac.id, [4]aisyatyanafisya@apps.ipb.ac.id,
[5]nurrizkytaaulia@apps.ipb.ac.id, [6]endang_pg@apps.ipb.ac.id, [7]gemaparasti@apps.ipb.ac.id

Abstract

This study introduces a real-time traffic density monitoring system utilizing YOLOv8-based digital image processing to improve traffic management efficiency. By leveraging YOLOv8's enhanced speed and precision, the system detects and classifies five types of vehicles and displays traffic data through a web interface developed with OpenCV and Flask. Key implementation features include real-time video streaming and accurate detection metrics, with the system achieving 96% Precision, 84% Recall, and an F1 Score of 90% during field testing in Bogor. This indicates the system's potential for minimizing manual traffic monitoring and aiding traffic authorities in making data-driven decisions. The research also discusses the system's integration into urban traffic management and its scalability for diverse environments.

Keywords: Traffic Density Monitoring, YOLOv8, Digital Image Processing, Real-time processing, Object Detection

1. INTRODUCTION

The advancement of digital technology and the increasing number of motor vehicles have introduced new challenges in urban traffic management. With the number of vehicles reaching approximately 133.6 million units in 2019, uncontrolled traffic density can lead to congestion, accidents, and reduced transportation efficiency [1]. Traffic congestion not only wastes valuable time but also has significant economic repercussions, including increased fuel consumption, delayed goods delivery, and productivity losses[2]. Additionally, high traffic density contributes to environmental pollution and raises public health concerns, exacerbating respiratory conditions and reducing the overall well-being of urban populations[3]. To address these issues, real-time traffic density detection systems are essential. Digital image processing technology enables automated

visual analysis to detect and count the number of vehicles on roads with high accuracy.

Digital image processing is a technology that enables the analysis and interpretation of images with the help of computers. This technology transforms images into data that can be processed automatically, such as detecting and counting objects within an image[4]. In the context of traffic monitoring, object detection refers to the identification of vehicles such as cars, motorcycles, or trucks captured in images or videos [4]. The system operates by analyzing each video frame to determine the location and type of detected objects, enabling real-time traffic monitoring with high accuracy[5]. This technology forms the basis of traffic density detection systems, allowing for the rapid and efficient identification of traffic issues.

Various research efforts have explored YOLO-based models to monitor traffic, each version offering distinct improvements in speed, accuracy, and computational efficiency. For instance, YOLOv3 has been widely used for vehicle detection and classification via CCTV feeds, demonstrating high precision. However, it often encounters delays in processing large datasets, which hinders real-time performance [6][7]. Tiny-YOLO, on the other hand, is favored for lightweight systems due to its compatibility with low-end devices, although this comes at the expense of accuracy [8]. Similarly, YOLOv4 provides a better balance between speed and precision, making it suitable for real-time traffic monitoring, but still faces limitations in highly dynamic environments [9].

One of the rapidly evolving solutions in this field is the YOLO (You Only Look Once) algorithm, renowned for its real-time object detection capabilities, high classification performance, and efficient computational parameters [10]. YOLOv8, the latest iteration, offers improved detection precision of up to 88.27%, especially in applications like tea bud detection, with more complex parameters and computations compared to earlier YOLO models [2]. Additionally, platforms such as OpenCV support the development of computer vision and machine learning applications, providing both low-level image processing functions and high-level object detection algorithms for effective vehicle identification [8].

The YOLO (You Only Look Once) object detection models have evolved significantly across versions, with each new iteration offering improvements in speed, accuracy, and architecture. YOLOv1, introduced in 2016, was a breakthrough as it treated object detection as a regression problem, predicting bounding boxes and class probabilities in a single evaluation. However, it struggled with small objects and overlapping targets in complex scenes [11]. YOLOv2 and YOLOv3 enhanced the framework by introducing anchor boxes and multi-scale

detection, resulting in higher accuracy without sacrificing speed. YOLOv4 further improved the model's performance by integrating cross-stage partial networks (CSPNet) and self-adversarial training, achieving an optimal balance between precision and speed[12].

With the release of YOLOv5, developed by Ultralytics, the model became popular for its usability, incorporating model scaling and tools for easy customization. Although it is widely adopted, it is not officially part of the original YOLO series [6]. YOLOv6 took this further by introducing efficient backbones and anchor-free detection, improving performance in industrial applications with faster inference and greater accuracy [13]. YOLOv7, officially recognized as part of the original YOLO lineage, focused on high-speed detection with enhanced architecture through extended efficient layer aggregation (E-ELAN). It has been successfully applied in real-time environments such as autonomous driving and smart city infrastructures [14].

Despite these advancements, several challenges remain. Many of the previous studies highlight the trade-off between speed and accuracy. While models like Tiny-YOLO offer faster processing, they lack the precision required for complex traffic scenarios [8]. Additionally, the reliance on high-end hardware for better performance restricts the deployment of these models on lower-spec devices [9]. Furthermore, most prior research has been conducted using relatively small or controlled datasets, which limits the scalability and applicability of these models in real-world traffic conditions [6][7]. These issues indicate the need for a more robust, scalable, and efficient solution that can function reliably across diverse environments.

To ensure the successful development of AI projects like this, the AI Project Cycle serves as a comprehensive framework, covering phases such as planning, data acquisition, modeling, and evaluation [12]. Building on these technologies and methodologies, this study aims to develop a real-time traffic density detection website based on YOLOv8, providing a fast, accurate, and scalable solution to address the challenges of urban traffic.

This study seeks to address the identified limitations in previous traffic monitoring approaches by leveraging the advanced capabilities of YOLOv8, chosen for its unique combination of high precision and fast processing, both critical for real-time traffic density detection [15]. Unlike its predecessors, YOLOv8 achieves a balance between speed and accuracy, which is essential for handling the high-density and dynamic conditions often encountered in urban traffic environments. The integration of a web-based platform in this research further enhances accessibility, enabling continuous monitoring and management from any location without dependency on specific hardware configurations [15].

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

Given the increasing number of vehicles and the resulting traffic congestion, an efficient and accurate traffic density monitoring system is essential for effective transportation management. The novelty of this research lies in the development of a real-time traffic density detection website utilizing YOLOv8-based digital image processing, specifically optimized to address the challenges of vehicle monitoring in complex, real-world environments [15][6]. This approach enhances the capability to monitor, detect, and classify vehicles with high precision, enabling authorities to respond to traffic conditions in real-time. Furthermore, this research seeks to leverage YOLOv8's strengths in balancing speed and accuracy, effectively overcoming the limitations faced by previous models in handling dynamic and high-density traffic [7] [15] .By integrating a robust and scalable web interface with the advanced detection capabilities of YOLOv8, this study aims to contribute to the field of intelligent transportation systems, providing traffic management authorities with an accessible and effective solution for real-time monitoring and decision-making.

This research aims to develop a real-time traffic density detection website utilizing the advanced YOLOv8-based digital image processing capabilities, chosen specifically for its high precision and rapid object detection, which addresses key challenges in real-time traffic monitoring. The system is designed to enhance traffic monitoring accuracy and efficiency by overcoming limitations found in previous YOLO versions, such as the trade-off between processing speed and detection accuracy, making it particularly suited for high-density and dynamic urban environments. By integrating YOLOv8 with a web-based interface, this study seeks to provide a scalable and accessible solution that can function effectively across diverse traffic scenarios. The proposed system aims to reduce the need for manual monitoring, minimize delays in traffic condition detection, and support better decision-making for traffic management authorities, thus contributing to the advancement of intelligent transportation systems.

The research is guided by the following questions:

a. How can the use of YOLOv8 enhance real-time traffic density detection through a web interface?
b. What specific improvements does YOLOv8 offer in terms of accuracy, speed, and usability compared to its predecessors?
c. How can the proposed system address the limitations found in previous YOLO implementations, such as balancing speed and precision in real-world applications?

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

2. METHODS

The AI Project Cycle is a structured process for developing AI projects [16]. The implementation of the AI Project Cycle involves several stages, as shown in Figure 1.

Figure 1. Research Stage

2.1. Problem Scoping

The process of identifying or mapping the limitations of the problem to be solved helps clarify and focus the objectives or targets, making it easier to find solutions [17]. The problem scope used is the 4W method, which includes Who, What, Where, and Why. This process aids in identifying and addressing the problem with more effective and efficient efforts.

2.2. Data Acquisition

The data source is taken from the official website of the Bogor City government, https://jembatanotista.kotabogor.go.id, which provides videos of vehicles passing at specific locations. Image Acquisition Method: The video is downloaded and processed using a frame-by-frame method, where each frame containing passing vehicles is extracted and saved as a single image. Selected frames are taken at specific intervals (e.g., every second or every few seconds) to avoid redundancy and ensure that the captured data has a variety of positions and conditions. Object Categories include various types of vehicles, such as cars, motorcycles, buses, and trucks. Each captured image is expected to contain one or more vehicles from these categories.

The collected data must undergo several processing stages to be ready for model training. Data Labeling: The collected image data is uploaded to a labeling platform such as Roboflow. Here, each object in the images (car, motorcycle, bus, truck) is annotated with a bounding box and labeled according to its category. The bounding box is manually drawn around each visible vehicle in the image, allowing the model to identify vehicle locations in each frame. The labels are re-verified to ensure no objects are missed and there are no labeling errors.

To increase data variety and make the model more resilient to various environmental conditions, data augmentation is applied with the following techniques:

1. Rotation and Cropping: Rotating and cropping images to create variations in viewpoint.
2. Lighting Adjustments: Modifying contrast, brightness, or adding shadow effects to create different lighting conditions.
3. Horizontal or Vertical Flip: Flipping images to account for vehicles from various directions.
4. Scaling and Zooming: Applying zoom in or zoom out to add variations in object size within the images.

2.3. Data Exploration

The exploration stage involves examining the dataset obtained from the previous stage. Image data annotation is performed by marking objects using bounding boxes and assigning appropriate class labels to each image [18]. The annotated dataset will then be divided into testing data, validation data, and training data. Training, Validation, and Testing Sets: The dataset is divided into three parts:

1. Training Set: About 70% of the data is used to train the model.
2. Validation Set: About 20% of the data is used to test model performance during training.
3. Testing Set: Around 10% of the data is used to evaluate the model's final performance. Class Stratification: The dataset is split while considering class proportions (cars, motorcycles, trucks, etc.) so that each set has a balanced class distribution.

2.4 Modelling

The process of creating algorithms in a programming language serves as a machine learning method (training phase) used to identify patterns in the data, forming the foundational knowledge for the system to make decisions or predictions [19]. The program configuration process for training the model is conducted using the Python programming language, with DeepSORT added as part of the tracking algorithm. This enables each object to be identified and tracked consistently, even with changes in position or appearance. Additionally, the training algorithm used for detecting and recognizing objects is YOLOv8, a highly efficient and accurate deep learning algorithm for detecting objects in images or videos. By using YOLOv8 for object detection and DeepSORT for tracking, the resulting model is capable of recognizing and tracking multiple objects simultaneously with optimal performance. DeepSORT (Deep Simple Online and Realtime Tracking) can be integrated with YOLOv8 for object tracking by utilizing YOLOv8 as the object detector and DeepSORT as the tracker. This combination is often chosen because YOLOv8 can detect objects quickly and accurately, while DeepSORT efficiently maintains the identities of detected objects across frames.

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

To train the YOLOv8 model for vehicle detection, various parameters have been selected to optimize the model's performance in detecting objects accurately and efficiently. The choice of these parameters is based on the need to balance detection accuracy, computational efficiency, and inference speed, which are essential for real-time applications. Below is an explanation of each parameter used:

1. Model = yolov8s.pt, Refers to the use of the small version of the YOLOv8 model (YOLOv8s), which has been pre-trained and saved in a .pt (PyTorch) file. YOLOv8s is smaller and lighter compared to the larger versions, making it more computationally efficient, especially if GPU resources are limited. YOLOv8s can produce good accuracy while maintaining fast detection speed, making it suitable for real-time applications like vehicle tracking.
2. Epochs = 100, the parameter epochs=100 sets the number of full iterations (epochs) to train the model on the entire dataset. Training for 100 epochs allows the model sufficient time to learn the data patterns well without risking overfitting. This number also considers the model's need to learn details from various environmental conditions and vehicle types found in traffic data.
3. imgsz=640
   The parameter imgsz=640 specifies the image size that will be used as input for training, i.e., 640x640 pixels. The 640x640-pixel size is large enough to capture essential details of vehicles yet efficient for processing. This size allows the model to have sufficient resolution to detect small objects while maintaining good speed during training and inference.
4. Optimizer: Stochastic Gradient Descent (SGD)
   SGD is chosen for its efficiency in handling large datasets, providing more frequent parameter updates, and aiding in the model's good generalization. It is well-suited for models like YOLOv8 that are trained with substantial traffic data.
5. Learning Rate: 0.001
   A learning rate of 0.001 is chosen to ensure stable parameter updates, avoid overshooting, and allow for smoother convergence. This is a commonly effective value for training object detection models like YOLOv8.

The pixel values in the images are normalized to a range of [-1,1] to make the training process more stable. Images are resized to a specific resolution 640x640 pixels that aligns with the input expected by the YOLOv8 model. Bounding box annotations or labels are stored in a format that can be read by the model, such as YOLO format (.txt files with bounding box coordinates) or COCO format (.json files). Each image has a label file containing object category information and bounding box coordinates, allowing the model to process it directly during

training. Each image and annotation is reviewed to ensure bounding box accuracy and correct labeling. Blurry, irrelevant, or mislabeled images are removed or corrected. Sampling: Sampling is conducted to confirm that the dataset adequately represents real-life road situations (e.g., types of vehicles and positional variations).

2.5 Evaluation

The evaluation stage is conducted as part of the assessment process to select the most suitable model for use in the AI project. At this stage, evaluation is carried out using a Confusion Matrix, which helps measure the model's performance based on the combination of various outcomes from actual values and model predictions [20]. With the Confusion Matrix, we can better understand how the model handles both correct and incorrect classifications.

Subsequently, further testing is performed using video. In this testing, a Region of Interest (ROI) line is defined to identify specific areas within the video, allowing the model to be optimized for detecting, counting, and classifying vehicles in real-time [21]. The goal is to determine whether the generated pre-trained model functions according to requirements or if there are discrepancies between the system's output and the expected results. This process is crucial for identifying errors or weaknesses in the model, enabling corrective actions to enhance the performance and accuracy of the AI model in real-world situations. The model selection process considers several components, including [14]:

1. **Accuracy:** the percentage of correct predictions from the total observations. Formula → (TP + TN) / (TP + FP + TN + FN)
2. **Precision:** the percentage of cases predicted by the AI that actually occurred based on reality. Formula → (TP) / (TP + FP)
3. **Recall:** measures the fraction of cases that occurred and were accurately predicted by the AI. Formula → (TP) / (TP + FN)

3. RESULTS AND DISCUSSION

3.1 Problem Scoping

This research develops a system capable of detecting traffic density in real-time through recordings of roadways. The main focus is to identify vehicle types such as motorcycles, cars, trucks, buses, and public transportation, as well as to analyze traffic density based on the number of detected vehicles. In the problem formulation stage, it is important to clearly identify the problem boundaries so that the research can be directed with the right focus. The problem scoping process

maps these boundaries to make the objectives more targeted and the solutions easier to find.

In this study, the primary challenge identified is the manual monitoring of traffic, which requires significant time and resources. Therefore, automated technology based on object recognition, such as the YOLOv8 algorithm, has been chosen as a solution to monitor traffic more efficiently and quickly. This algorithm is expected to detect various types of vehicles in real-time, provide accurate results, and minimize the need for manual intervention. To facilitate the problem scoping process, the 4W method is used to outline the issues more structurally:

1. **Who** (Who is involved in this problem?): In this case, the parties involved include traffic managers, road users, and the automated monitoring system.
2. **What** (What is the main problem?): The problem lies in the difficulty of efficient and accurate manual traffic monitoring, especially in congested urban roads.
3. **Where** (Where does the problem occur?): This problem occurs on congested urban roads, where traffic density needs to be monitored in real-time to avoid congestion or accidents.
4. **Why** (Why does this problem need to be solved?): Due to the rapid increase in the number of vehicles, significant traffic issues have arisen, and efficient automated solutions are needed to address these challenges.

Thus, this problem scoping helps ensure that the research focuses on developing a system capable of efficiently detecting and analyzing traffic density, providing practical solutions to urban traffic problems.

3.2 Data Acquisition

The data acquisition process in this study begins with obtaining traffic video recordings from the official website of the Bogor City government, which is https://jembatanotista.kotabogor.go.id. The videos collected are live recordings of roadway conditions at several strategic points in Bogor City, representing various levels of traffic density at different times. After obtaining the videos, the next step is to convert the videos into a series of screenshots or frames. This process is carried out by extracting images from the video using software such as OpenCV, as shown in Figure 1.

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

Figure 2. Image Extraction Process

The image extraction process in this research is performed by processing the video taken from each location using OpenCV software to break it down into frames or images. These images are then further analyzed to identify and classify vehicles on each road, thereby providing accurate data for monitoring traffic flow in Bogor City.

3.3 Data Exploration

After the image upload process to the Roboflow platform is complete, the next step is to perform labeling on the vehicle objects present in the images extracted from the traffic videos in Bogor City. At this stage, the labeled objects include various types of vehicles commonly seen on the roads, such as motorcycles, cars, trucks, buses, and angkot (public minivans). One challenge is the impact of poor lighting conditions, which can reduce image clarity and detection accuracy, especially at night or in dim environments, to address this, additional image pre-processing steps, such as contrast enhancement or brightness adjustments, can be applied to improve visibility. Furthermore, training the model on a diverse dataset with varying lighting conditions can help increase its robustness for reliable detection across different times and weather conditions. Each object is manually labeled in each image to ensure that the YOLOv8 model to be used can recognize and detect these types of vehicles with high accuracy during the training and implementation process, as shown in Figure 3.

Journal of Information Systems and Informatics
Vol. 6, No. 4, December 2024
p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

Figure 3. Proses Labelling

The labeling process is a crucial stage, as the quality and consistency of the labeling will affect the model's ability to accurately recognize vehicle objects. Each image must be labeled precisely and accurately for the model to learn the visual patterns of each type of vehicle. Additionally, the differences between one type of vehicle and another, such as size and shape, must also be considered so that the model can effectively distinguish between various objects.
Once all objects have been labeled, the dataset is then divided into several subsets consisting of:

1. **Training Set (70 images):** This dataset is used to train the model to recognize the patterns and characteristics of each vehicle object.
2. **Testing Set (20 images):** This set is used to test the model's performance after training, with the goal of evaluating how well the model can detect vehicle objects in data it has not seen before.
3. **Validation Set (10 images):** This dataset serves to validate the training results during the training process, allowing the model to be optimized without the risk of overfitting.

All labeled images, which have been divided into these subsets, are then resized to 640x640 pixels. This size is chosen because it matches the input dimensions required by the YOLOv8 model, while also aiming to accelerate the training and inference processes without compromising detection accuracy. With a smaller size, the computational resource requirements, such as memory and processing power, can be reduced, yet the model can still perform detection with high precision. This data exploration phase ensures that the dataset is ready for use in the model training stage, with a balanced distribution between the training, testing, and validation sets, and ensures that the model will be trained with representative data from real traffic conditions.

3.4 Modelling

3.4.1 Model Training

After the dataset has been processed and divided into training, testing, and validation sets, the next step is to train the model using the YOLOv8 algorithm. This algorithm was chosen for its ability to perform real-time object detection with high speed and accuracy, which is crucial for traffic detection applications. YOLOv8 is the latest development in the YOLO (You Only Look Once) family, known for its efficiency in detecting various types of objects, including vehicles, in images captured from videos. The capability of YOLOv8 to work quickly and effectively in processing large images, such as those from traffic videos, is the main reason for its selection in this research.

During the model training process, YOLOv8 is trained to recognize vehicle objects such as motorcycles, cars, trucks, buses, and minibuses based on the previously labeled dataset. The goal of this training is for the model to accurately identify and detect these vehicles in real traffic situations with optimal performance.

3.4.2 Model Configuration

The model training is conducted using the Google Colab platform, which provides a GPU (Graphics Processing Unit)-based computing environment. By leveraging the power of GPUs, the model training process can be carried out more quickly and efficiently, especially when working with a relatively large dataset as in this research. Google Colab was chosen for its accessibility and availability to integrate hardware with the computational capacity needed to train the YOLOv8 model.

The Ultralytics library, an advanced and user-friendly implementation of the YOLO (You Only Look Once) object detection algorithm, was used for training the YOLOv8 model. This library is known for its simplicity in setting up and

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

running training sessions while providing comprehensive support for customization. By utilizing the Ultralytics library, the model training process could be seamlessly implemented, from data preprocessing to evaluation.

The dataset that has been processed and labeled through Roboflow, containing vehicle objects such as motorcycles, cars, trucks, buses, and minibuses, is used as input for model training. The model configuration during training includes several key parameters:

1. Model = yolov8s.pt
2. Batch Size: 16
3. Learning Rate: 0.001
4. Epochs: 100
5. Optimizer: Stochastic Gradient Descent (SGD)
6. Image Size: 640x640 pixels

These parameters are adjusted to achieve a balance between training speed and detection accuracy. A batch size of 16 is chosen to ensure that memory usage during training remains efficient without compromising model performance. The learning rate is set at 0.001 to control how quickly the model updates its weights at each epoch, allowing the model to learn patterns steadily without overfitting.

The training process is conducted over 20 epochs to give the model enough time to learn from the dataset, while the use of Stochastic Gradient Descent (SGD) as the optimizer is chosen for its advantages in performing parameter updates quickly and efficiently. Furthermore, all images are resized to a size of 640x640 pixels, which is the standard input dimension for YOLOv8, to ensure that the training and detection processes run smoothly and align with the available processing capacity. Below is the process of training the model using YOLOv8, as shown in Figure 4.

Figure 4. Proses Training Model menggunakan YOLOv8

With this configuration and parameters, it is expected that YOLOv8 can achieve an optimal detection accuracy level and be capable of detecting various types of

vehicles in real-time traffic environments, in line with the objectives of this research.

3.5 Evaluation

3.5.1 Evaluation Metrics

After the training is complete, the model is then evaluated using the validation set to monitor its performance on unseen data. The evaluation is conducted using metrics such as Precision, Recall, F1 Score, and Mean Average Precision (mAP). The training results are presented in the form of several evaluation graphs as follows:

1. Loss Curve

The Loss Curve graph shows the changes in loss values as the number of epochs increases. The loss value indicates how far the model's predictions are from the actual values. A smaller loss value indicates that the model is performing more accurately.

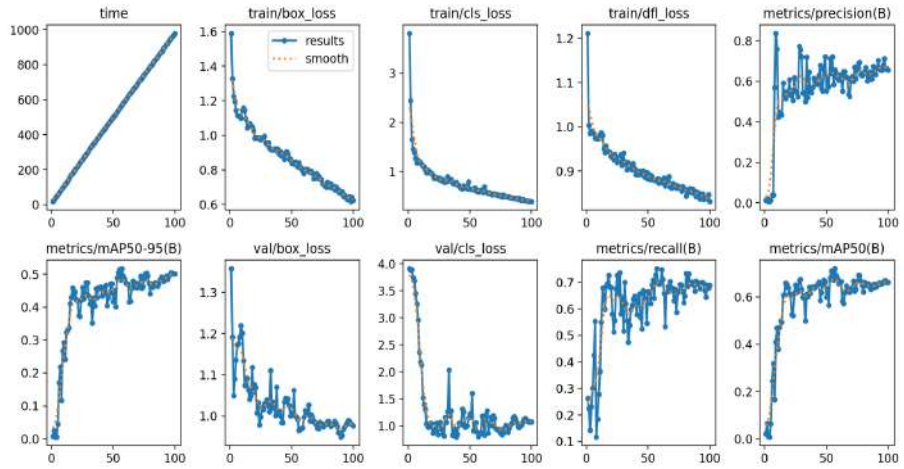

Figure 5. Loss Curve Graphic

In the train box loss graph, a significant decrease is observed from around 1.5 at the beginning of training to nearly 0.5 by the 100th epoch. This indicates an improvement in the model's ability to predict bounding boxes. Additionally, the train classification loss and train distribution focal loss (DFL) show consistent decreases, suggesting that the model is becoming better at classifying objects and understanding their positions more accurately.

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

In the validation data, a similar trend is observed in the validation box loss and validation classification loss. Both metrics demonstrate a stable decrease during training. Although there are slightly larger fluctuations compared to the training data, the model maintains good performance on the validation data. Overall, the results from these graphs indicate that the model has significantly improved over the 100 epochs of training. Despite some fluctuations in certain metrics, the model effectively enhances its object detection performance.

2. F1-confidence Curve

The F1-confidence Curve graph illustrates the relationship between the F1 score and the confidence threshold for various object classes. The F1 score is a metric that combines precision and recall, providing a balanced measure of the model's classification performance. The confidence threshold refers to the level of certainty the model has when making positive predictions. A higher confidence value indicates greater certainty in the model's predictions, with higher F1 scores reflecting better performance.

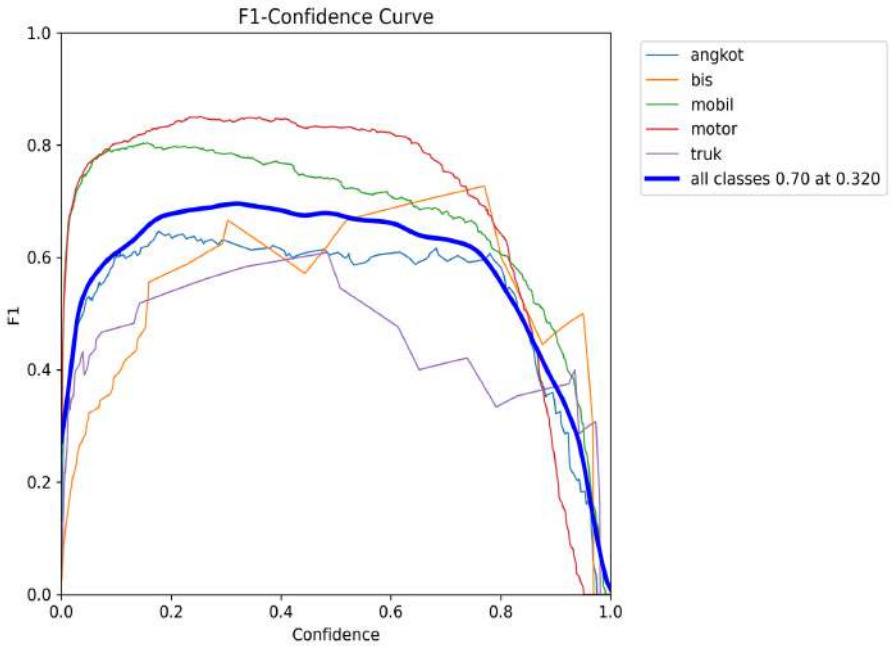

Figure 6. Grafik F1-confidence Curve

Each line in the graph represents the model's performance for a specific object class. The motorcycle and car classes demonstrate the best performance, with stable F1 scores around 0.9 at confidence levels between 0.4 and 0.6. The bus class

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

also performs well, achieving an F1 score of approximately 0.75 at a confidence level around 0.75. In contrast, the truck class shows the lowest performance compared to the others, with an F1 score reaching only about 0.5 at confidence levels between 0.3 and 0.5. The angkot class (light blue line) occupies a middle position, with a stable F1 score of around 0.6 at a confidence level of approximately 0.6. The blue line represents the overall performance across classes. At a confidence level of 0.320, the F1 score reaches its highest value of 0.70, indicating that at this confidence level, the model achieves the best balance between precision and recall. At a confidence value of 0.320 and an F1 score of 0.70, the model performs optimally in handling class variations while avoiding bias toward specific classes. From the graph above, it can be concluded that the model performs very well for certain classes such as motorcycles and cars but struggles to detect classes with limited data, like trucks. The decline in the F1 score at very high confidence levels (above 0.7) suggests that when the model becomes overly confident in its predictions, the error rates increase.

3. Confusion Matrix

The confusion matrix illustrates how the model classifies objects correctly and incorrectly. There are four main categories: True Positive, False Positive, True Negative, and False Negative. This matrix provides an overview of correct classifications and detection errors made by the model.

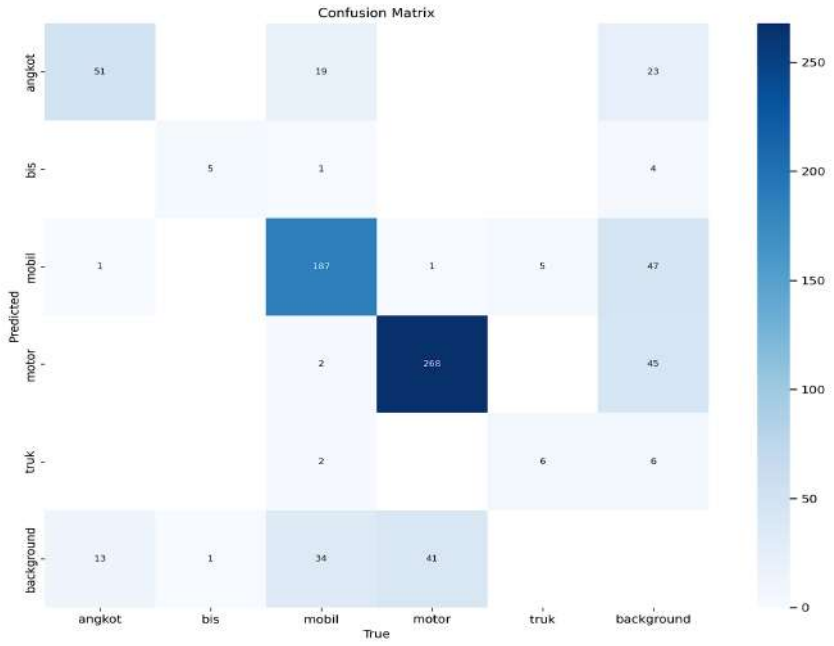

Figure 7. Confusion Matrix

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

The diagonal elements in the matrix indicate the predicted labels that match the actual labels. For instance, there are 51 angkots classified correctly as angkots, 5 buses correctly classified as buses, 6 trucks correctly classified as trucks, 187 cars accurately identified as cars, and 268 motorcycles accurately predicted as motorcycles. Conversely, the off-diagonal elements represent classification errors. Errors occur when the model predicts a class different from the actual one. For example, 19 cars were incorrectly classified as angkots. This indicates that the model sometimes experiences confusion between certain classes.

4. Confusion Matrix Normalized

The normalized version of this confusion matrix provides a clearer comparison of how often the model correctly or incorrectly detects each class, measured as a proportion of the total classifications.

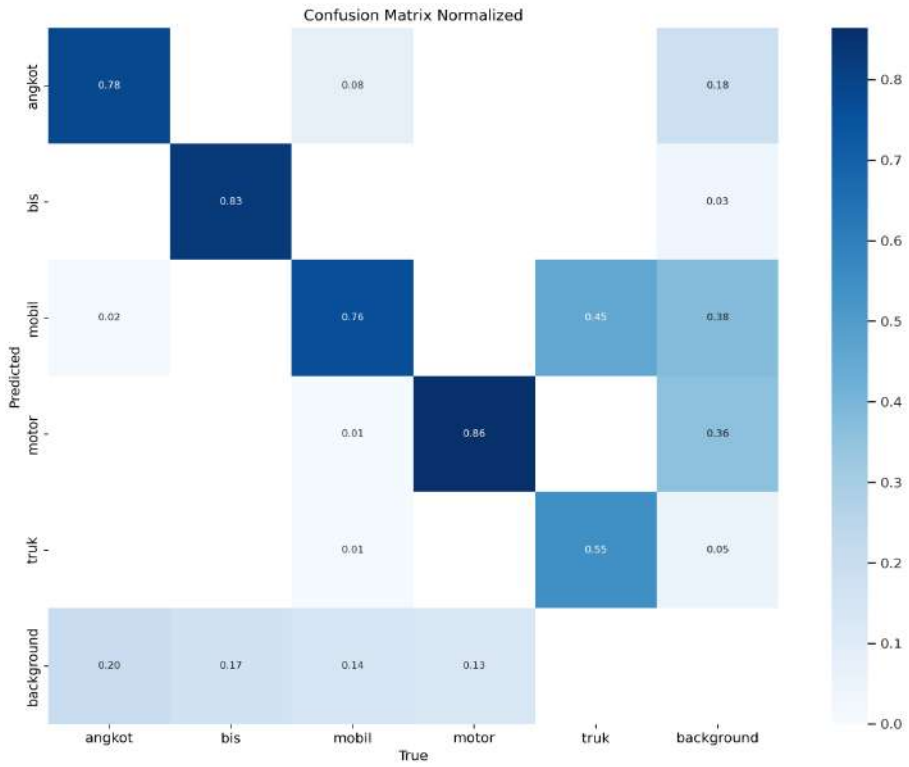

Figure 8. Normalized Confusion Matrix

From Figure 8, it can be seen that the motor and bus classes show relatively good prediction performance with accuracies of 0.86 and 0.83, respectively. Most of the angkot and bus objects detected by the model were successfully classified

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

correctly. However, there were significant prediction errors, such as 45% of truck objects being classified as cars.

5. P Curve (Precision Curve)

The Precision Curve graph illustrates the relationship between confidence levels and precision values. Precision measures the model's ability to correctly identify vehicles (true positives) compared to the total predicted positives. The higher the precision value, the fewer prediction errors the model produces.

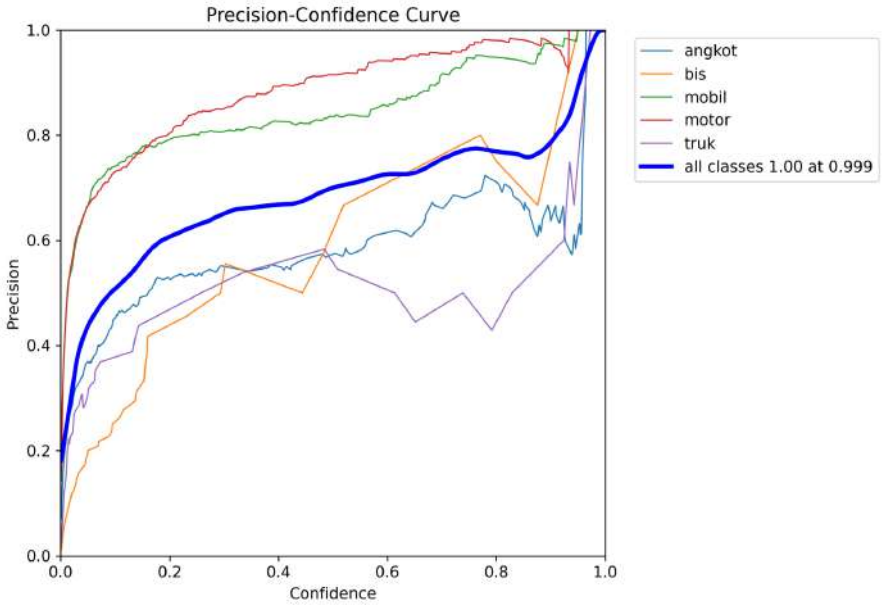

Figure 9. Precision Curve

The blue line in the graph represents the overall accuracy across all classes. At low confidence levels, precision is relatively low. As the confidence approaches one, precision also approaches one. This indicates that the model is very accurate in predictions when the confidence level is high.

6. R Curve (Recall Curve)

The recall curve shows the relationship between recall (the model's sensitivity in detecting specific categories) and the confidence level of predictions across various classes. Recall measures how many actual positive cases were correctly detected by the model.

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

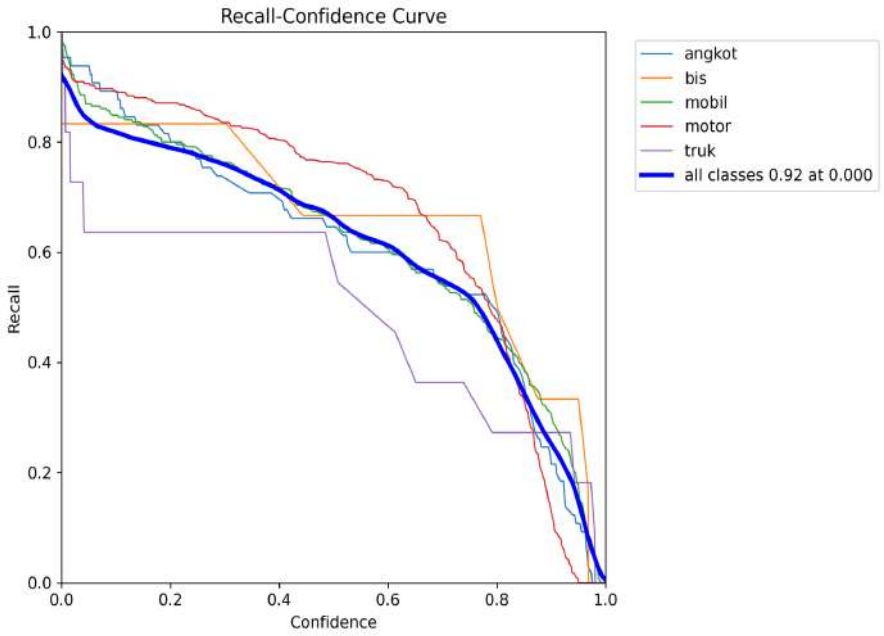

Figure 10. Recall Curve Graph

There are five curves representing each class: angkot, bus, car, motorcycle, and truck. Additionally, there is a blue curve representing the aggregate performance of the model across all classes. This curve shows how recall decreases as confidence increases. Generally, when the confidence threshold is raised (e.g., only predictions with high probabilities are accepted), the model tends to make fewer true positive predictions, resulting in a decrease in recall.

7. Precision-Recall Curve

The Precision-Recall Curve graph shows the relationship between precision and recall at various detection thresholds. This graph is useful for understanding the trade-off between accurate detection (precision) and broad detection coverage (recall) at different sensitivity levels of the model. Overall, the motorcycle class performs best with the highest precision and recall (mAP = 0.890), indicating that the model is very effective in identifying motorcycles with few errors. Meanwhile, the truck class has the lowest performance (mAP = 0.525), suggesting that the model struggles to recognize trucks correctly. The poor detection performance may be due to a lack of truck image data in the training set. With an mAP of 0.720 for all classes, the model is quite good at balancing precision and recall.

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

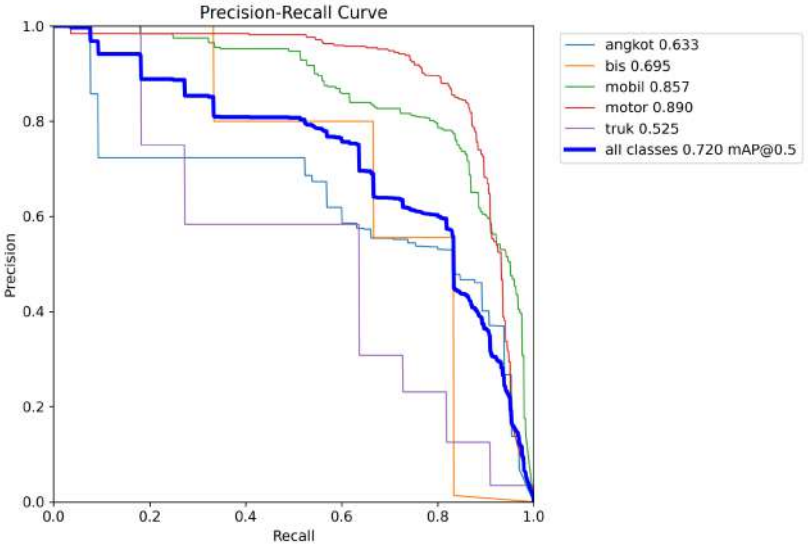

Figure 11. Precision-Recall Curve Graph

8. Labels

The label distribution graph shows how many examples of each object class were used in training and evaluating the model. This is important to ensure that the dataset has a balanced distribution and does not affect the model's performance.

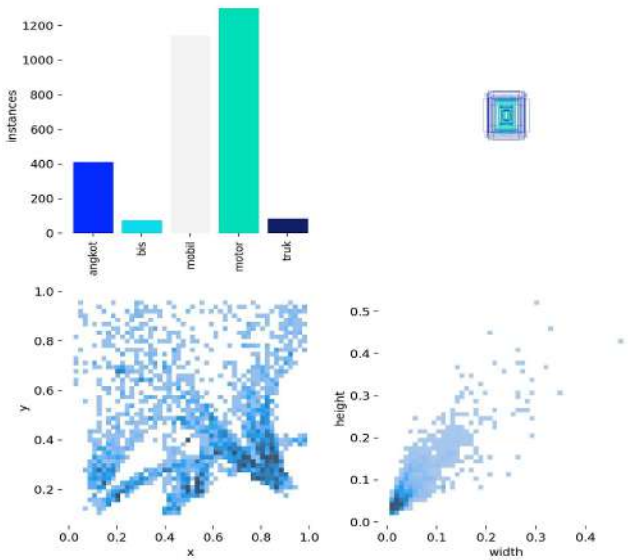

Figure 12. Label Distribution Graph

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

In the Figure 12, it is evident that the most abundant labels are for cars and motorcycles. Meanwhile, the bus and truck labels are very few.

3.5.2 Training Performance

At this stage, evaluation was conducted to measure the performance of the YOLOv8 model in detecting vehicles in a traffic environment. The model was tested for 1 minute on four main roads in the Bogor area: BTM, Kapten Muslihat, Sukasari, and Tugu Kujang, detecting five types of vehicles: motorcycle, car, angkot, truck, and bus. During testing, True Positives (TP), False Positives (FP), and False Negatives (FN) were calculated for each vehicle type at each location, which were then used to calculate evaluation metrics such as Precision, Recall, F1 Score, and Accuracy. Here are the TP, FP, and FN calculations for the YOLOv8 model during testing at the four locations.

Table 1. Training and Evaluation Table

Location	Car Type	TP	FP	FN
Jalan Depan BTM	Car	42	5	0
	Motorcycle	91	1	0
	Angkot	6	0	1
	Truck	1	0	3
	Bus	0	1	0
Jalan Kapten Muslihat	Car	19	0	1
	Motorcycle	21	0	19
	Angkot	2	0	0
	Truck	0	0	0
	Bus	0	0	0
Jalan Lawanggintung	Car	6	0	5
	Motorcycle	11	0	31
	Angkot	4	0	0

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

Location	Car Type	TP	FP	FN
	Truck	1	3	0
	Bus	0	0	0
Jalan Tugu Kujang	Car	46	0	0
	Motorcycle	58	0	0
	Angkot	6	0	0
	Truck	0	2	0
	Bus	1	1	0

The table above shows the results of detecting each vehicle at various locations, with the True Positive (TP) count indicating the vehicles that were correctly detected, False Positives (FP) indicating the number of incorrectly detected vehicles, and False Negatives (FN) indicating vehicles that were not detected by the model. To get an overall picture of the model's performance, we can sum the TP, FP, and FN values from the table above and then calculate Precision, Recall, F1 Score, and Accuracy as follows:

1. Total TP = 315
2. Total FP = 13
3. Total FN = 60

Using these values, the evaluation metrics are calculated as follows:

1. Precision = TP / (TP + FP) = 315 / (315 + 13) ≈ 0.96
2. Recall = TP / (TP + FN) = 315 / (315 + 60) ≈ 0.84
3. F1 Score = 2 × ((Precision × Recall) / (Precision + Recall)) ≈ 0.90
4. Accuracy = TP / (TP + FP + FN) ≈ 0.81

From these calculations, it can be concluded that the YOLOv8 model performs well in detecting five types of vehicles in four different locations. With a Precision of 96%, the model is quite accurate in identifying the correct vehicles, and with a Recall of 84%, it shows high sensitivity in capturing vehicles present on the road. An F1 Score of 90% indicates a balance between precision and recall, while an Accuracy of 81% reflects the overall accuracy of the model in detecting vehicles in traffic environments. Overall, the YOLOv8 model demonstrates excellent performance in detecting vehicles under varying traffic conditions. With

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

evaluation metrics showing consistent and accurate results, this model can be relied upon for detecting various types of vehicles in different locations with a low error rate. This indicates that the YOLOv8 model can be effectively implemented for applications requiring real-time vehicle detection in the traffic of Bogor City.

3.6 Implementation

After the vehicle detection process using the YOLOv8 model is completed, the next step is to count the number of detected vehicles and analyze traffic density. This process is conducted using OpenCV for image processing and Flask as the backend framework to display the detection results on the website. Additionally, Chart.js is utilized to create dynamic and easily understandable real-time traffic density visualizations.

3.6.1 Vehicle Detection and Counting

After YOLOv8 detects vehicles in the video frames, each vehicle will be given a bounding box and label based on its type, such as car, truck, bus, motorcycle, and angkot using OpenCV. OpenCV iterates through each video frame to count the number of detected vehicles. By leveraging data from YOLOv8, the number of vehicles by type (e.g., cars or trucks) and the total count is calculated automatically each time the model detects a vehicle object. The output is as shown in the following image.

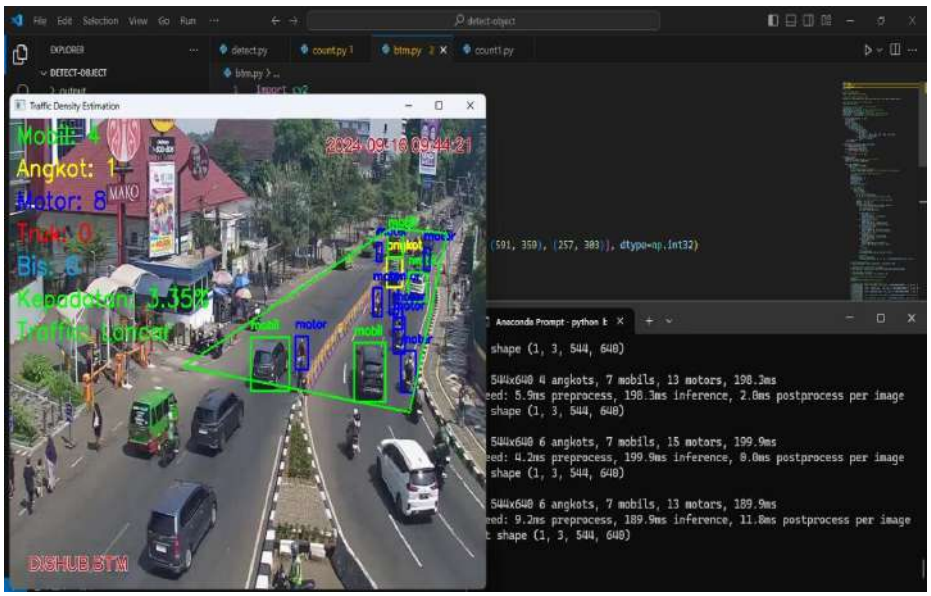

Figure 12. Vehicle Detection with OpenCV Bounding Box

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

3.6.2 Density Analysis and Real-Time Visualization

Traffic density is analyzed based on the number of vehicles detected in the observation area over a specific period. The number of vehicles successfully detected by the YOLOv8 model is compared to the area of the observed road to calculate the density level. Google Earth is used to measures the road area. The traffic density formula is calculated by dividing the area of each vehicle passing through the observation area during a specific time frame by the road area., yielding density values that can be classified into three main categories, free flow in 0% - 9% density percentage, Slightly Congested Flow in 10% - 15% density percentage, and Congested Flow for more than 15% density percentage:

1. **Free Flow:** Low vehicle count, traffic moves without obstruction.
2. **Slightly Congested Flow:** Traffic is becoming slightly congested, but vehicles can still move at a reasonable speed.
3. **Congested Flow:** High vehicle count, traffic moves slowly or is at a standstill.

This density level is crucial for understanding traffic conditions at various times and locations. The following visualization uses Chart.js to create a line graph representing traffic density based on detection data.

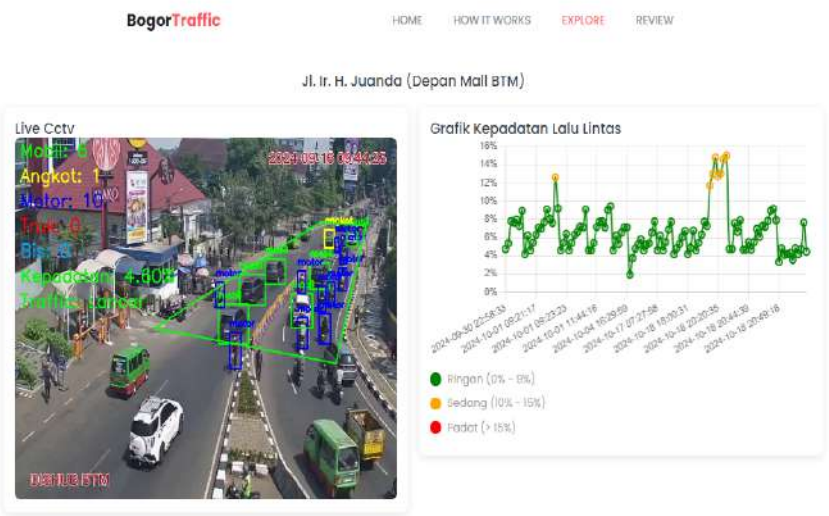

Figure 13. Traffic Density Real Time Visualization

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

In Figure 13, an example of a real-time traffic density graph generated using Chart.js is displayed. This graph shows how the density levels change over time, providing in-depth insights into traffic patterns at the analyzed location.

3.6.3 Visualization on the Website with Flask

After the vehicle data is counted and the density is analyzed, the results are displayed on the website using Flask. The detection results are shown in real-time on the web page with the help of OpenCV and Flask. Each processed video frame is sent to the frontend via a Flask endpoint, allowing users to see the detection results directly in their browser.

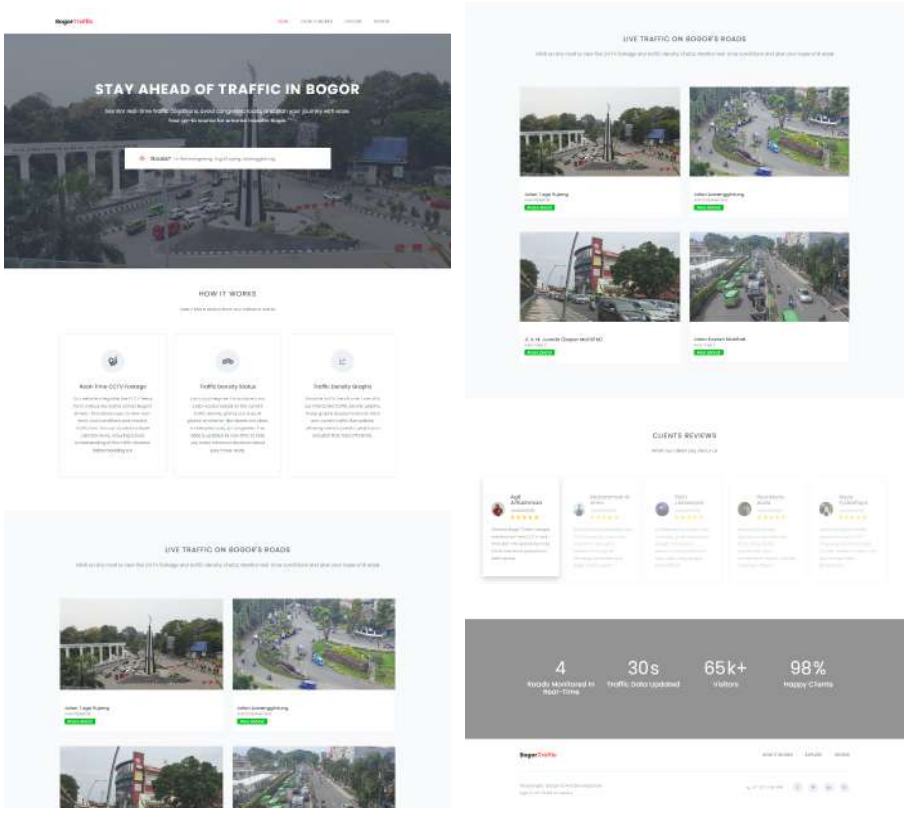

Figure 14. BogorTraffic Website Display

BogorTraffic is a website designed to monitor traffic density in real-time at various strategic locations in Bogor City. At each monitoring point, the website displays up-to-date information regarding traffic conditions, with status categories such as Smooth Flow, Slightly Congested, and Congested. Users can easily access this data

to plan their journeys, helping them avoid traffic jams and choose more efficient routes. Thus, BogorTraffic plays an essential role in assisting the community and drivers to better understand the traffic situation in Bogor City.

Overall, the YOLOv8-based traffic monitoring system demonstrates advantages over traditional monitoring solutions, such as inductive sensors or standard cameras. This system effectively detects various types of vehicles in real-time with high accuracy, without requiring additional road infrastructure, thus offering greater flexibility and cost efficiency. In urban traffic management, the YOLOv8 model can be applied to provide instant density information to traffic authorities, enabling dynamic adjustments like traffic light timing or suggesting alternative routes to reduce congestion. Expected benefits of implementing this system include smoother traffic flow, reduced travel times, and improved road safety. More broadly, this system has the potential to support environmental management by lowering vehicle emissions through congestion reduction and to assist authorities in gathering long-term traffic data that can inform infrastructure planning and transportation policy more effectively.

3.7 Discussion

This research highlights the transformative potential of real-time traffic density detection using the YOLOv8 algorithm, addressing critical challenges in urban traffic management. By automating vehicle detection and traffic density analysis, the system overcomes limitations associated with traditional manual traffic monitoring methods, which are time-intensive and error-prone. The focus on leveraging advanced object detection technology enables precise and efficient traffic monitoring, offering valuable insights for improving urban mobility. The problem scoping phase provided a strong foundation for the study by defining the research boundaries and identifying key challenges. Through a structured 4W analysis, the study established the involvement of stakeholders, the main problem areas, and the importance of addressing traffic congestion. This structured approach ensured the research remained focused on its primary objective: developing a practical solution to improve traffic monitoring in urban settings. The selection of YOLOv8, known for its speed and accuracy in real-time object detection, reflects the study's emphasis on deploying state-of-the-art technology for meaningful applications.

One of the main contributions of this study is the robust data preparation process, which involved acquiring traffic video recordings from real-world urban environments and converting them into labeled datasets. Despite challenges such as poor lighting conditions, the research mitigated these issues through image preprocessing and dataset diversification. These steps ensured that the YOLOv8 model was trained on a dataset representative of real-world conditions, enabling

reliable performance across varying scenarios. However, the imbalance in the dataset, particularly the underrepresentation of certain vehicle classes like trucks, highlighted the need for more comprehensive data collection to further improve model accuracy. The integration of YOLOv8 into a real-time traffic monitoring system showcased its effectiveness in both detection and usability. By incorporating the model with tools such as OpenCV, Flask, and Chart.js, the system was able to visualize traffic density in real-time, providing actionable insights for traffic management. The BogorTraffic website exemplifies how such a system can directly benefit users by displaying live traffic conditions, enabling informed decision-making, and reducing congestion through better route planning.

The study's findings also underscore the broader implications of adopting real-time traffic monitoring systems. Beyond immediate benefits like improved traffic flow and reduced congestion, the system contributes to environmental goals by potentially lowering vehicle emissions through congestion management. Additionally, the long-term traffic data collected can serve as a valuable resource for urban planners and policymakers in designing sustainable infrastructure and transportation systems. However, the research also identified areas for improvement and future exploration. Addressing dataset imbalances, enhancing model robustness under challenging conditions (e.g., poor lighting or extreme weather), and integrating additional features such as speed estimation or accident detection could significantly expand the system's capabilities. Furthermore, leveraging edge computing to process data on-site could enhance the system's scalability and responsiveness. This study demonstrates how advanced object detection algorithms like YOLOv8 can be effectively utilized to tackle pressing urban traffic challenges. The development of a real-time, automated traffic monitoring system not only improves the efficiency and accuracy of traffic management but also sets the stage for broader applications in smart city initiatives. By addressing limitations and building on the demonstrated strengths, this research paves the way for further innovations in traffic management and urban mobility solutions.

4. CONCLUSION

In conclusion, this study demonstrates that the YOLOv8-based real-time traffic density detection system significantly enhances traffic monitoring with high Precision (96%), Recall (84%), and an F1 Score of 90% in urban settings. Despite these achievements, the system faces limitations, such as reduced performance in poor lighting conditions and high computational demands. Future research should focus on improving robustness under diverse lighting through advanced preprocessing, optimizing performance for lower-spec devices, and integrating with IoT frameworks to enable adaptive traffic control. Testing across various

cities and environments will further validate scalability and efficacy, contributing to more intelligent urban traffic management solutions.

REFERENCES

[1] Tirtowahjono, S. C., and Purwanto, D. G., "Penentuan Distribusi Arus Lalu Lintas Pada Persimpangan Berbasis Teknologi Computer Vision & Deep Learning," *Jurnal Dimensi Pratama Teknik Sipil*, vol. 11, no. 1, 2022.

[2] S. Xie dan H. Sun, "Tea-YOLOv8s: A Tea Bud Detection Model Based on Deep Learning and Computer Vision," *Sensors*, vol. 23, no. 14, Jul 2023, doi: 10.3390/s23146576.

[3] C. Reche, A. Tobias, dan M. Viana, "Vehicular Traffic in Urban Areas: Health Burden and Influence of Sustainable Urban Planning and Mobility," *Atmosphere (Basel)*, vol. 13, no. 4, Apr 2022, doi: 10.3390/atmos13040598.

[4] T. A. Dompeipen dan S. R. U. A. Sompie, "Penerapan Computer Vision Untuk Pendeteksian Dan Penghitung Jumlah Manusia," *Jurnal Teknik Informatika*, vol. 15, no. 4, pp. 1–12, 2021.

[5] J. Azimjonov dan A. Özmen, "A real-time vehicle detection and a novel vehicle tracking systems for estimating and monitoring traffic flow on highways," *Advanced Engineering Informatics*, vol. 50, pp. 101393, Okt 2021, doi: 10.1016/J.AEI.2021.101393.

[6] D. A. Abdurrafi, M. Taqijjuddin Alawiy, dan B. M. Basuki, "Deteksi Klasifikasi Dan Menghitung Kendaraan Berbasis Algoritma You Only Look Once (YOLO) Menggunakan Kamera CCTV," *SCIENCE ELECTRO*, vol. nn, no. 9, 2023.

[7] S. W. J. Hutauruk, T. Matulatan, dan N. Hayaty, "Deteksi Kendaraan Secara Real Time Menggunakan Metode YOLO Berbasis Android," *Jurnal Sustainable: Jurnal Hasil Penelitian Industri Terapan*, vol. 09, no. 01, pp. 8–14, 2020.

[8] P. Y. Putra *at all*., "Deteksi Kendaraan Truk Pada Video Menggunakan Metode Tiny-Yolo V4," *JIP (Jurnal Informatika Polinema)*, vol. 9, no. 2, pp. 215–2023, 2023.

[9] I. Surya Dharma, "Sistem Rekognisi Jenis Kendaraan Dengan Analisis Video Menggunakan Metode Yolov4," *Kumpulan jurnaL Ilmu Komputer (KLIK)*, vol. 11, no. 1, 2024.

[10] C. M. Badgujar, A. Poulose, dan H. Gan, "Agricultural object detection with You Only Look Once (YOLO) Algorithm: A bibliometric and systematic literature review," *Comput Electron Agric*, vol. 223, pp. 109090, 2024, doi: 10.1016/j.compag.2024.109090.

[11] J. Redmon, S. Divvala, R. Girshick, dan A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," Jun 2015.

[12] J. R. Terven, D. M. C. Esparza, dan J.-A. Romero-González, "A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS," *Mach. Learn. Knowl. Extr.*, vol. 5, pp. 1680–1716, 2023.

[13] C. Li *at all.*, "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications," Sep 2022, doi: 10.48550/arXiv.2209.02976.

[14] C.-Y. Wang, A. Bochkovskiy, dan H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," Jul 2022.

[15] N. J. Hayati, D. Singasatia, M. R. Muttaqin, T. Informatika, S. Tinggi, dan T. Wastukancana, "Object Tracking Menggunakan Algoritma You Only Look Once (Yolo)V8 Untuk Menghitung Kendaraan," *KOMPUTA : Jurnal Ilmiah Komputer dan Informatika*, vol. 12, no. 2, 2023.

[16] F. Azimah dan K. Wardani, "Sistem Pendeteksi Gejala Awal Covid-19 dengan Penggunaan Metode Al Project Cycle," *Journal Locus Penelitian dan Pengabdian*, vol. 1, pp. 405–418, Sep 2022, doi: 10.58344/locus.v1i6.135.

[17] M. R. S. Putri, S. Hermuningsih, dan G. Wiyono, "Pengaruh Profitabilitas, Leverage, Ukuran Perusahaan, dan Pertumbuhan Aset Terhadap Nilai Perusahaan," *Owner*, 2024.

[18] A. Faqih, K. Mutmainnah, dan M. R. Afifah, "Seperation Deteksi Kendaraan Pada Citra Digital Dengan Menggunakan Algoritma YOLO (You Only Look Once)," *Jurnal Teknik Informatika dan Elektro*, vol. 3, no. 2, pp. 5–11, 2021.

[19] Mulajati, M., "Implementasi Teknik Web Scraping dan Klasifikasi Sentimen Menggunakan Metode Naïve Bayes Classifier dan Asosiasi Teks (Studi Kasus: Data Ulasan Penumpang Maskapai Penerbangan Garuda Indonesia pada Situs TripAdvisor)," 2017.

[20] M. K. Suryadewiansyah dan T. E. E. Tju, "Naïve Bayes dan Confusion Matrix untuk Efisiensi Analisa Intrusion Detection System Alert," *Jurnal Nasional Teknologi dan Sistem Informasi*, 2022, [Daring]. Tersedia pada: https://api.semanticscholar.org/CorpusID:252085210

p-ISSN: **2656-5935** http://journal-isi.org/index.php/isi e-ISSN: **2656-4882**

[21] A. H. Pratomo, W. Kaswidjanti, dan S. Mu'arifah, "Implementation of Region Of Interest (ROI) Algorithm To Improve Car Detection And Classification Algorithm," *Jurnal Teknologi Informasi*, vol. 7, no. 1, pp. 155–162, 2020, doi: 10.25126/jtiik.202071718.