INTERNATIONAL JOURNAL
ON INFORMATICS VISUALIZATION

journal homepage : www.joiv.org/index.php/joiv

Enhancing Vision-Based Vehicle Detection and Counting Systems with the Darknet Algorithm and CNN Model

Abdul Haris Rangkuti [a,*], Varyl Hasbi Athala [a]

[a] *Computer Science Department, School of Computer Science, Bina Nusantara University, Bandung Campus, Jakarta, Indonesia*
Corresponding author: [*]rangku2000@binus.ac.id

***Abstract*—This study focuses on developing an algorithm that accurately calculates the volume of vehicles passing through a busy crossroads in Indonesia using object recognition. The high density of vehicles and their proximity often pose a challenge when distinguishing between vehicle types using a camera. Therefore, the proposed algorithm is designed to assign a unique identity (ID) to each vehicle and other objects, such as pedestrians, ensuring that volume calculations are not repeated. The objective is to provide an equitable comparison of road density and the total number of detected vehicles, enabling the determination of whether the road is crowded. To accomplish this, the algorithm incorporates the Non-Max Suppression function, which displays bounding boxes around objects with confidence values and counts the objects within each box. Even when objects are nearby, the algorithm tracks them effectively, thanks to the support of the Darknet Algorithm. The main capabilities of this algorithm for improving vehicle detection include enhanced accuracy, speed, and generalization ability. Typically, it is used in conjunction with the You Only Look Once (YOLO) object detection framework. Five convolutional neural network models are tested to assess the algorithm's accuracy: YOLOv3, YOLOv4, CrResNext50, DenseNet201-YOLOv4, and YOLOv7-tiny. The training process utilizes the Darknet Algorithm. The best-performing models, YOLOv3 and YOLOv4, achieve exceptional accuracy and F1 scores of up to 99%. They are followed by CrResNext50 and DenseNet201-YOLOv4, which achieve accuracy rates of 92% and 98% and F1 scores of 94% and 98%, respectively. The YOLOv7-tiny model achieves an accuracy rate and F1 score of 86% and 88%, respectively. Overall, the results demonstrate the algorithm's success in accurately detecting and calculating the volume of vehicles and other objects in a busy intersection. This makes it a valuable tool for regional government decision-making.**

***Keywords*—Volume; vehicle; object recognition; crossroads; CNN; accuracy; F1 Score.**

Manuscript received 12 Feb. 2024; revised 6 Apr. 2024; accepted 3 Jun. 2024. Date of publication 31 Jan. 2025.

I. Introduction

The government has demonstrated its commitment to developing infrastructure for road construction throughout Indonesia. By providing transportation facilities, the government aims to facilitate interaction between local communities and their surrounding environment, encompassing social, economic, and cultural aspects [1]. Roads are crucial in accommodating various vehicles and pedestrians, including cars, public transport, trucks, bicycles, motorbikes, and pedestrians. They have become an indispensable component of transportation systems [2]. Modern society faces serious problems with transportation systems, including but not limited to traffic congestion, safety, and pollution. Information communication technologies have gained increasing attention and importance in modern transportation systems [3]. To address this, transport authorities have increasingly turned to CCTV cameras to monitor traffic flows and gather valuable data for various applications. One such application is automatic vehicle classification, which involves specialized software identifying different types of vehicles (small, medium, and large) in recorded footage. This technology offers numerous benefits, from optimizing traffic management to informing infrastructure planning [4].

Different current road loads can cause inefficiency in using lanes at the intersection. Traffic regulation at the intersection regulates the movement of each group of vehicle movements so that they can move alternately and do not interfere with each other or disrupt existing flows [5]. However, traffic lights in urban areas are still less effective due to the unbalanced volume of vehicles. All traffic flow values (per direction and total) are converted into passenger car units (pcu) using the car ferry equivalent, which is derived from each type of vehicle as follows [6].

a. Light vehicles (LV), namely two-axle, 4-wheeled motorized vehicles with 2.0-3.0 m (including passenger cars, microbuses, pick-ups, and small trucks)
b. Heavy vehicles (HV), namely motorized vehicles with more than 3.5 m and typically with more than four wheels (including buses, two axles' trucks, three trucks, and combination trucks).
c. Motorcycles (MC), namely two or three-wheeled motorized vehicles.

Traffic volume is the number of vehicles that pass a certain point or line. Vehicles are typically classified into several types, including heavy vehicles, light vehicles, motorcycles, and non-motorized vehicles [4]. The traffic volume on a road will vary, forming a traffic flow pattern. Traffic flow patterns indicate changes in traffic volume over a given period [7]. Basically, traffic flow patterns help us know peak and non-peak hours and their intervals. Density is the number of vehicles per unit length of the road (vehicles/km). Density can be observed from aerial photos and Closed-Circuit Television (CCTV) installed at several intersection points. Describing the short-term traffic flow is essential for studying intelligent transportation systems [8]. Knowing beforehand the real-time density of a road or an intersection could make the road less crowded due to drivers avoiding potentially high traffic [9].

Detecting vehicle objects is the first step in obtaining traffic flow information at the intersection. Object detection aims to get the location and classification of objects from an image. The goal is to acquire the features of the object. In this study, observations will be made of six class objects around the intersection. Class objects include cars, trucks, public transportation, bicycles, motorcycles, and people who are at the intersection location. Five convolutional neural network models will test the objects during the testing process. However, the models undergo a training process using Darknet before detecting the targeted objects. This paper aims to use object detection on traffic in urban areas and to experiment with which convolutional neural network models are best suited for this case.

The crossroads used as the experimental site of this research is in the Bandung area, West Java province. Fig 1. shows four intersections with dense characteristics, such as the intersection of the *Buah Batu* and *Batu Nunggal* highways, during a test experiment using a darknet framework. At the depicted intersection, the volume of each object class will be automatically calculated. Knowing the volume of objects for each class can also improve the supervisory function of vehicle objects while supporting local government decision-making.

Implementing an idea of fast and timely traffic flow that can effectively reduce traffic jams, reduce accidents, and prepare a comfortable traffic environment. Traffic conditions at the intersection are data on traffic volume taken during peak hours in the Bandung city area. The description for the traffic volume data that became the research material was taken from several intersection roads in the city of Bandung. At this intersection, every day, there is a tremendous amount of traffic. For this reason, regulating traffic lights at crossroads is needed to run vehicles and accommodate every road user. The problem with regulating the traffic system using a fixed time model is it can cause changes in traffic density to be unpredictable because of the traffic lights.

Fig. 1 Shows intersections area with four locations during an experiment using a darknet framework

II. Materials and Methods

A. Related Works

An intelligent traffic light control system must be implemented dynamically with real-time traffic. Studies are using deep reinforcement learning techniques for traffic light control, showing reasonably good results for control [10]. Ultimately, using smart transportation (e.g., smart traffic lights) will make our trips more comfortable and efficient and help avoid congestion on one side of the road [11]. In general, the Intelligent Transportation System (ITS) application has become an essential component and has been widely implemented for smart cities to overcome the limitations of traditional transportation systems. The existing traffic light control system divides the traffic light signals into fixed durations and operates inefficiently [12]. The description of the need for Intelligent Transportation Systems (ITS) has become a concern in recent years. In addition, with the rapid development of vehicle computing hardware, vehicle sensor systems, and city-wide infrastructure, many of these applications continue to be developed, such as Vehicular Cloud (VC), intelligent traffic control, etc [13].

Traffic demand forecasting is essential for transport management and public safety. Still, it is very challenging because of the complex spatial-temporal dependence and consequent uncertainties created by the road network and traffic conditions [14]. Traffic flow prediction is the central part of ITS research. Road traffic data shows the same trend on successive days. Accurate traffic flow prediction ensures public safety and solves traffic jams. The increasing demand for faster travel, severe traffic congestion, and its adverse impact on traffic safety and environmental conditions have attracted significant attention from countries worldwide [15] due to the limited land resources, construction costs, and time-consuming processes. Furthermore, the highway expansion project cannot wholly and effectively solve this problem. In addition, potential traffic demand is also generated due to increased vehicle traffic capacity [16]. For this reason, the research is focused on knowing the volume of vehicles in an area so that it will be an input for local governments to find

appropriate and efficient solutions in dealing with increasingly severe traffic jams. The description for detecting vehicle volume in this study is to use Artificial intelligence technology through machine learning.

The problem that necessitates the development of object detection applications is the need to accurately determine road density, particularly at intersections, in real-time. By addressing the density issue, drivers can avoid road congestion and reduce the likelihood of encountering heavy traffic. Several additional benefits are associated with an application capable of detecting the density of vehicle objects. Firstly, drivers can actively seek alternative routes to avoid being trapped in time-consuming traffic jams, thus saving valuable travel time. Moreover, this technology empowers drivers to proactively circumvent potential high-traffic areas, further enhancing their ability to avoid congestion. Target detection technology, as one of the core technologies in computer vision, provides basic technical support for many aspects, such as target tracking, semantic segmentation of vehicles under heavy traffic conditions, discovering in-vehicle alcohol to prevent road accidents, and detecting cyberattacks on autonomous vehicles [17].

The typical approach for determining the number of vehicles traversing a highway involves employing detection and tracking techniques. By analyzing the tracking trajectories of vehicles, it becomes possible to calculate the total count of vehicles passing through a specific area. The vehicle calculation process based on detection and tracking methods can be subdivided into background reduction and DNN-based methods [18]. Background reduction technology is used to design the background model and extract the existing moving vehicles in the videos. Several morphological operations are usually applied to the vehicle segment to count the traffic vehicles [19]. The background model is specifically used for a limited region within the video frame. Subsequently, morphological processing is applied to the extracted target to amplify its features and mitigate the impact of obstructing vehicles [20]. Various techniques are employed for detecting moving objects. The process involves several post-processing steps, which are crucial in establishing optimal thresholds for distinguishing between foreground and background. These steps significantly enhance the detection rate accuracy achieved through this technique. However, identifying and adapting a suitable threshold, particularly in environments with limited visibility, has proven unsuccessful thus far [19].

The preparation phase uses “ffmpeg” for video cutting and labeling images (Fig 2). Once the preparation has been completed, the training phase uses the Darknet algorithm to train the prepared data. Lastly, the testing phase measures the performance and outputs the results using OpenCV. OpenCV is an open-source library of programming functions mainly for image processing. OpenCV was chosen in favor of Darknet for the testing due to its ability to write the appropriate algorithm for the experiment [20].

This comprehensive paper delves into deep learning-based object detection frameworks, thoroughly reviewing their capabilities and advancements [21]. Recognizing the diverse nature of specific detection tasks, we extend our exploration to encompass a brief survey of notable tasks such as salient object detection, face detection, and pedestrian detection [22]. By analyzing the unique characteristics of each task, we aim to provide valuable insights into the evolving landscape of object detection within the realm of deep learning [23]. The utilization of deep learning object detection algorithms, specifically designed for analyzing 2D images, has emerged as a formidable force in road object detection within autonomous driving [24]. The remarkable success achieved by deep learning methods in the context of road vehicle detection is indisputable [25]. These advancements have solidified the critical role of deep learning in enhancing the accuracy and efficiency of road object detection and paved the way for unprecedented progress in autonomous driving [26].

Nevertheless, rapidly and accurately detecting and classifying vehicles faces challenges arising from the limited spacing between vehicles on the road and interference features in photos or video frames containing vehicle images. A novel vehicle detection and classification model has been developed by optimizing the YOLOv4 model to address this issue. This model incorporates an attention mechanism that effectively suppresses image interference features by considering both channel and spatial dimensions [27]. The CNN model detects moving vehicles using various techniques. One common approach is frame difference, where the model compares consecutive frames in a video to identify the differences in object positions. When an object, such as a vehicle, moves in a video, its location changes from frame to frame. By detecting these changes, the model can identify the presence of moving vehicles [28]. Another method involves training a deep learning model specifically for object detection. This approach requires labeled data to train the model by collecting and annotating a dataset or fine-tuning a pre-trained model on specific data. The trained model can then detect moving vehicles in videos [29].

This research focuses on detecting street objects using camera surveillance. There are six objects in focus: cars, motorcycles, trucks, bicycles, humans, and public transportation. These objects are likely to be seen throughout the city streets of Indonesia. In this experiment, video footage of the road using camera surveillance in Bandung City is used on the CCTV. The video quality and lighting of the footage are not the focus. Therefore, the footage is obtained from the source. There are three phases of the experiment: the preparation, training, and testing phase. Each phase uses a different method of processing. This research also compares five different convolutional neural networks or CNN models. The five different CNNs, namely use You Only Look Once Version 3 (YOLOv3), Version 4 (YOLOv4), Version 7 tiny (YOLOv7-tiny), CSResNext50-Panet-SPP, and DenseNet201-YoloV4. These CNNs were chosen to differentiate from previously published work. In some instances, the method for using the CNN model for object detection and comparing the models to determine the best performance has been experimented with. Therefore, the main topic of this research is to compare the performance of the five different CNN models for each street location. A more thorough explanation can be seen in the following section.

B. Preparation Phase

In Fig 2, the preparation phase diagram is shown. In the first step, a system was created to detect the objects within a CCTV frame, using images obtained from a road CCTV video. These videos were mainly 44 seconds long, and five public

CCTV videos were collected from the Area Traffic Control System of Bandung *DISHUB* website. Once collected, frames from every few seconds of the video get extracted and become new images using a software called FFMPEG. These images are used to train the machine. Each image requires labeling to make the machine understand the image. Image labeling is constructing a map of visual features with semantic and spatial labels that describe the objects in the image [30]. Image labeling has the function of teaching the machine to understand the given image. This process outputs a class definition text file and an image label text file for each image. Therefore, these files are the references for the machine during training to differentiate between objects and non-objects [31].

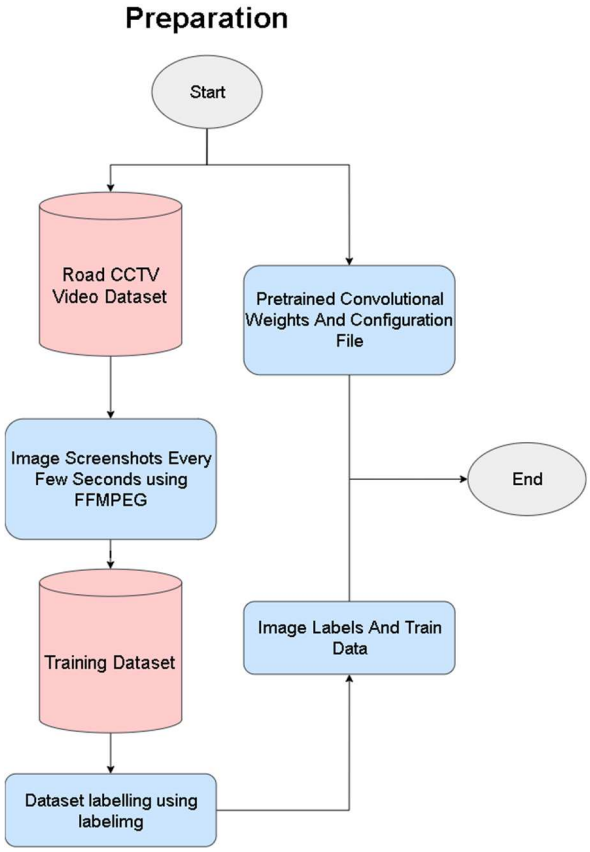

Fig. 2 Preparation Phase Diagram in detecting the vehicle and People

Image labeling has the function of teaching the machine to understand the given image. As stated before, the image labeling process uses labeling as software. This software reads all the images in the given folder and manually draws the bounding box and the class name within the image. This process outputs a class definition text file and an image label text file for each image. Therefore, these files are the references for the machine during training to differentiate between objects and non-objects. There are six types of objects that the CNN models must detect. These objects are cars, motorcycles, trucks, people, bicycles, and Indonesian public transportation called *angkot*. Each CNN model must correctly detect and identify as many of these objects as an object detection system to have a high value. Table I shows an example of precisely labeled objects with their correct object category.

TABLE I
IMAGE LABEL SAMPLE AND CATEGORIES IN ENGLISH AND INDONESIAN LANGUAGE

Image Sample	Category	
	English	**Indonesian**
	Car	*Mobil*
	Motorcycle	*Motor*
	Truck	*Truk*
	People	*Orang*
	Bicycle	*Sepeda*
	Public Transportation	*Angkot*

This experiment needs some pre-trained convolutional weights and configuration files data for each CNN model, in addition to the image labels and train data. After completing the preparation process, there will be four types of data.

C. Dataset Setup

There are 43 combined street images extracted from obtained road CCTV videos. These images have sizes ranging from 516 to 832 kilobytes each. These also have relatively the same size and aspect ratio. This research doesn't have any preprocessing methods for the images. However, all images have sufficient lighting and contrast for the experiment. Therefore, the machine would learn unprocessed image data to detect real-world objects. Each image has one labeling text that contains data on roadway image objects. The dataset also contains one class definition text. These data are created during the labeling process by a program called labeling. Therefore, in total, there are 87 files contained in a dataset for the training phase. These data are important to help the machine understand objects in the image. This research used CCTV video from one of the obtained video collections for the testing phase. An overview of the training in detecting the volume of vehicles at the intersection using the darknet algorithm can be seen in Fig 3.

Fig. 3 Diagram of Training Phase Using Darknet

The diagram in Fig. 4 presented below is a training process to describe an automatic system designed to detect traffic with darknet algorithms. The aim is to detect and inform the amount of traffic volume at road intersections. This research aims to improve the detection of darknet traffic by exploring a series of machine learning and deep learning techniques to classify such traffic and accurately show related application types.

D. Training Phase

In Fig 4, the data serves as the foundation for the training process. The training process was conducted five times as a part of this experiment, as five different CNN models were tested. Each of these CNN models possesses its own unique architecture, including Yolo V3, Yolo V4, CSRestNext50, Yolo V7 Tiny, and DenseNet201-YoloV4. Throughout this training process, only five CNN models were utilized optimally despite attempting several other models that did not yield satisfactory accuracy results. By employing a variety of CNN models, it becomes possible to identify the most suitable model to achieve optimal accuracy. Consequently, the number of extracted features and the output size may differ among the models.

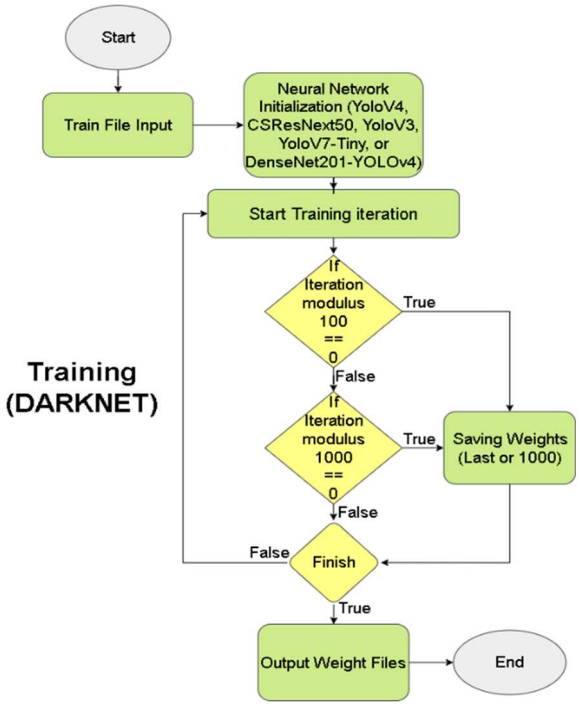

Fig. 4 Training Phase Diagram Using Darknet algorithm.

However, this discrepancy is not problematic, as the models still adhere to the fundamental CNN architecture. Specifically, CNNs, in this case, consist of three primary layers: Convolutional, Pooling, and The Fully Connected Layer [32]. Another study addresses these issues concerning accidents and aims to find solutions to reduce road accidents resulting from traffic-related incidents. The main challenge faced in computer vision lies in obtaining effective results when dealing with variations in data shapes and colors [33]. The training process saves the last weight every 100 iterations to prevent complete data loss if something happens. After 1000 iterations, the final weight files were created. There are five final weight files for each of the five CNN models. Therefore, these 100 and 1000 in iteration calculations are solely to save the training progress in those iteration numbers. An overview of the monitoring process stages for several vehicles at bustling crossroads was obtained using the darknet algorithm. Some peripherals support the training process. It can be seen in Table II.

TABLE II
DEVICE SPECIFICATION USED FOR TRAINING

No.	Component	Specification
1	Processor	AMD Ryzen™ 5 5600H Up To 4.2 GHz
2	RAM	16 GB DDR4
3	GPU	RTX 3060 Mobile 6GB
3	Disk Space	1.5 TB

Based on Table II, the heavy process of determining the difference factor between CNN models to carry out the training phase is done on the same device. Table II provides information about the six classes' device specifications for the training stage. This type of computer device is a personal computer with the latest generation of computer components that can carry out tasks specifically in detecting 6 classes of vehicles at road intersections, including counting the number of vehicles passing through road intersections. The need for GPU memory and processor specifications becomes dominant in processing class objects carried out in the training process.

E. Testing Phase

The next step after the training process is the testing process. Fig 5 explains the testing process stages, which start with inputting data on vehicle objects, people, and input frames. Next, predictions are made for object detection using one of the five CNN models used in the experiment. The prediction results are in the form of calculated bounding box coordinates. Non-Max suppression calculations are performed to display only the most optimal bounding boxes for objects to reduce the number of unimportant bounding boxes. The bounding boxes are then drawn and saved for later use by developers or users.

Furthermore, the retrieving vehicle process, the person class data, and calling the volume calculation function for each object are immediately processed. This step is crucial so that the vehicle calculation process can be carried out on the frame being processed at that time. Calculating the object or object volume begins by taking data on the number of object labels detected in the frame and then adding them up. This becomes the vehicle volume data detected in the frame. Then, the algorithm performs the object tracking process, which begins with giving an ID to the object in question. However, the object tracking process has two "if" cases. If two objects are adjacent to the main object and the main object is new, the algorithm gives the object a new ID.

Once object tracking is complete, the calculation of object volume is also performed for each label. This is done in 12 seconds. When the 12-second time limit is reached, the calculation algorithm outputs the model testing performance data, including the traffic volume, the number of objects detected in each category, and the inference time. In addition, the algorithm also generates images that contain information

about the traffic volume at that time, the number of objects that appear in each category, and the resulting bounding box.

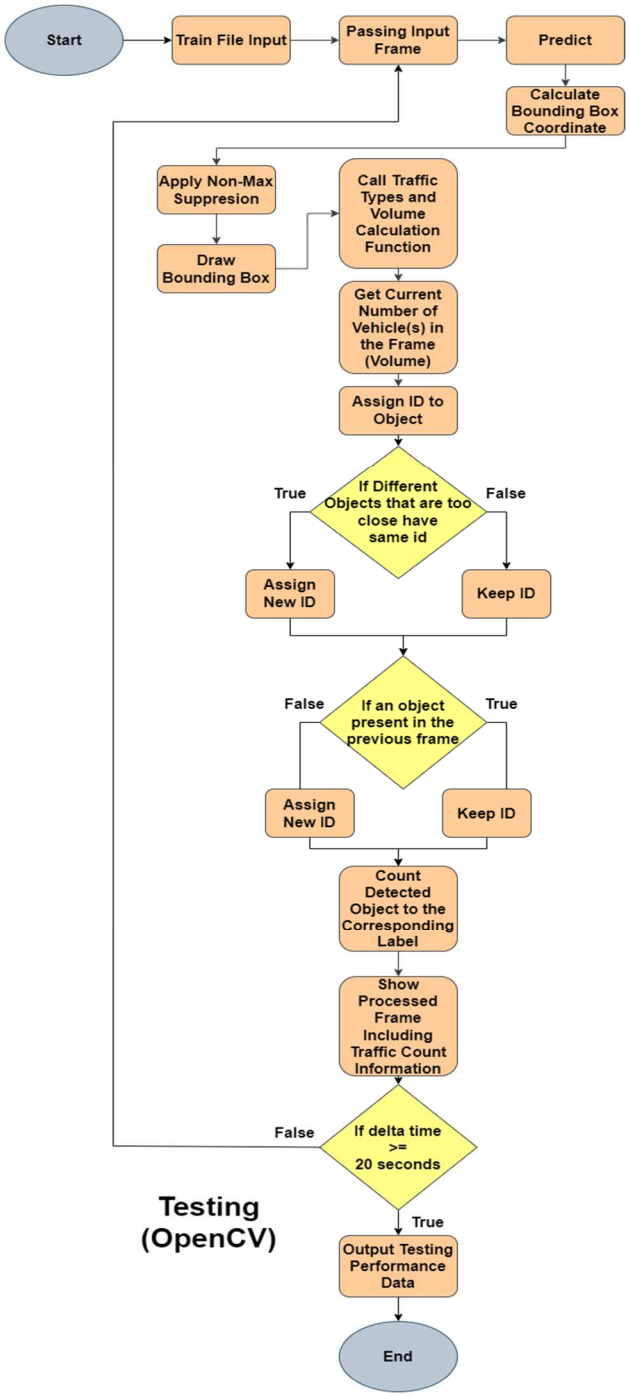

Fig. 5 Diagram of Testing Phase Using OpenCV

F. Measuring Model Performance

There are measurements to determine how well a model performs in a specific case. This experiment used five measurements: precision, recall, F1-Score, mean average precision, and traffic count. Collecting these variables, the extracted data will be used to compare the performance between the five chosen CNNs. Therefore, this comparison will determine which CNN has the most optimal performance in the case of CCTV object detection. A more detailed explanation of these calculations is given in the following section:

1) Precision and Recall:

Precision is used to determine the ability of a model to detect or identify the targeted objects. The recall is used to assess the ability of a model to find the targeted objects. Precision is the percentage of correct positive predictions, while recall is the percentage of correct positive predictions among all given ground truths (the number of total objects). To obtain the precision and recall values, each detected bounding box needs to be classified as:

a. True positive (TP): A correctly detected ground truths bounding box.
b. False positive (FP): An incorrectly detected non-existing or existing object with a misplaced detection bounding box.
c. False negative (FN): An undetected ground-truth bounding box.

Suppose a dataset with G ground-truths and a model that produces N detections and S of which are correct (S ≤ G) [25]. The concepts of precision and recall can be formally expressed as follows:

$$\mathrm{Pr} = \frac{\sum_{n=1}^{S} TP_n}{\sum_{n=1}^{S} TP_n + \sum_{n=1}^{S} FP_n} = \frac{\sum_{n=1}^{S} TP_n}{all\ detections} \quad (1)$$

$$\mathrm{Rc} = \frac{\sum_{n=1}^{S} TP_n}{\sum_{n=1}^{S} TP_n + \sum_{n=1}^{S} FN_n} = \frac{\sum_{n=1}^{S} TP_n}{all\ ground\ truths} \quad (2)$$

2) F1-Score and mean Average Precision:

F1-score is a calculation to produce a mean of precision and recall, which can be expressed as:

$$F_1 = 2\frac{Pr.Rc}{PR+Rc} = \frac{TP}{TP+\frac{FN+FP}{2}} \quad (3)$$

The values of the F1-score range from 0 to 1, where 1 means the highest accuracy when both precision and recall are 1 and 0 if precision or recall (or both) have the value of 0. Mean Average Precision or mAP is the average AP over all classes, which is expressed as the following formula [34]:

$$mAP = \frac{1}{C}\sum_{i=1}^{C} AP_i \quad (4)$$

where:

AP_i: The AP value for the *i*-th class
C: The total number of classes being evaluated.

3) Intersection over Union:

Intersection over Union or IoU is a metric used in object detection to compare the similarity between two bounding boxes: the predicted bounding box and the ground reference bounding box (the box the developer previously labeled). IoU encodes the shape properties of the objects under comparison into the region property, such as the widths, heights, and locations of two bounding boxes. Then, it computes a normalized measure focusing on their areas or volumes [37]. Fig 6 illustrates how IoU works.

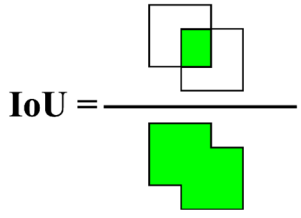

Fig. 6 Training Graph of Each CNN Model

4) Traffic Count (volume):

Traffic count is a crucial measurement used to assess the level of traffic in a video at a specific point in time. This measurement plays a vital role in evaluating the performance of a model in real-world scenarios. The traffic count measurement involves two calculations: the number of objects detected in a particular second and the number of objects in the last captured frame. In this experiment, 12 seconds were selected, and all objects were counted during this period. The machine must successfully detect the objects to calculate the total traffic for a given time. For each frame, the machine only counts new objects, assigning them a unique ID until they disappear. This process continues until the last frame at 12 seconds, accumulating objects for their respective categories. Once the total traffic volume is calculated, the next step is to count the number of objects in the last captured frame. This calculation identifies and evaluates flaws in the model's vehicle object detection performance by determining the number of objects in a single frame.

III. Results and Discussion

A. Results

This experiment will use five types of CNN models to detect cars, motorcycles, trucks, bicycles, public transportation, and people as their targeted objects. Each CNN model has a different performance result when used in that scenario. The performance of the models can be measured through five metrics. The metrics are precision, recall, F1-score, average IoU percentage, and inference time relative to mAP@0.50. These five metrics are automatically calculated by Darknet algorithm and generated as an output. Two additional metrics were also calculated during the training process: duration and the average model training loss.

Fig. 7 shows the training process of CSResnext50-Panet-SPP in a graph. The graph shows that the loss percentage began to fall after around the 300th iteration and continuously dropped until the 3000th iteration. From the 3000th iteration, the loss percentage maintained a steady reduction, staying between 0% and 2% until the last iteration. The next CNN model is the DenseNet201-YOLOv4.

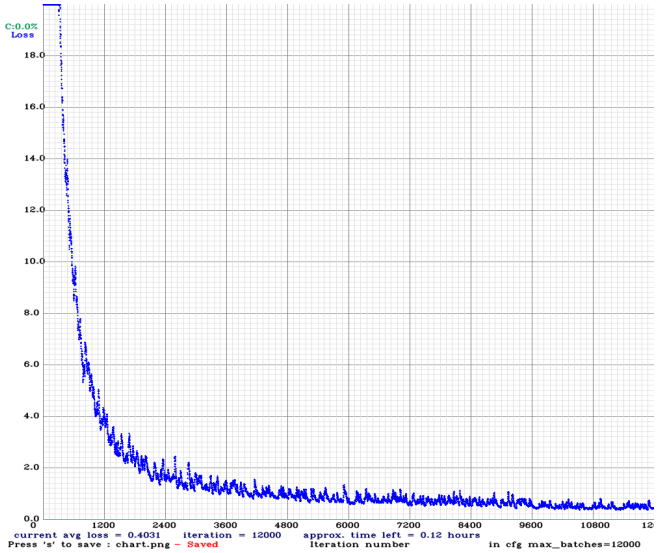

Fig. 7 Training Graph of CSResnext50-Panet-SPP

Fig 8 illustrates the model's training graph. This graph shows the most unsteady training process. Unlike other training graphs that have a steadier line, this graph shows that the loss percentage tends to increase and decrease every 100 iterations. However, it still resulted in a lower loss percentage overall for every 3000 iterations after the 2400th iteration.

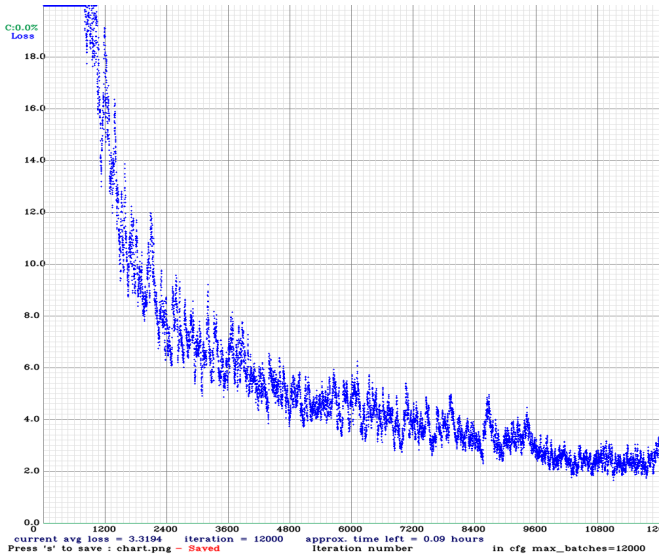

Fig. 8 Training Graph of DenseNet201-YOLOv4

Fig 9 shows the training graph of YOLOv3. The training process of YOLOv3 has many similarities with the training process of CSResnext50-Panet-SPP with their steady loss percentage. The last iteration resulted in between 0.2% and 0.6% loss percentage.

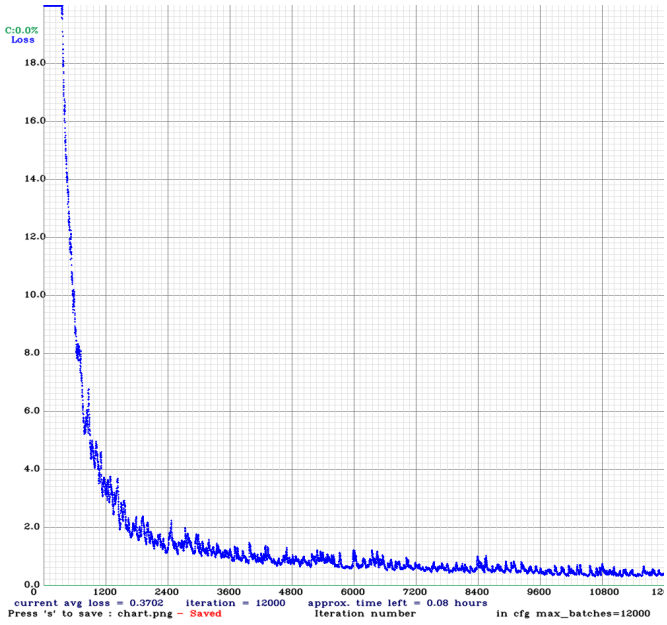

Fig. 9 Training Graph of YOLOv3

Fig. 10 presents the next CNN model for training six classes of objects: YOLOv4. Fig. 10 illustrates the training process in a graph. After the 2400th iteration, the loss percentage kept increasing and decreasing by 1% for every 100 iterations. Eventually, the last iteration stopped at between 1% and 2%.

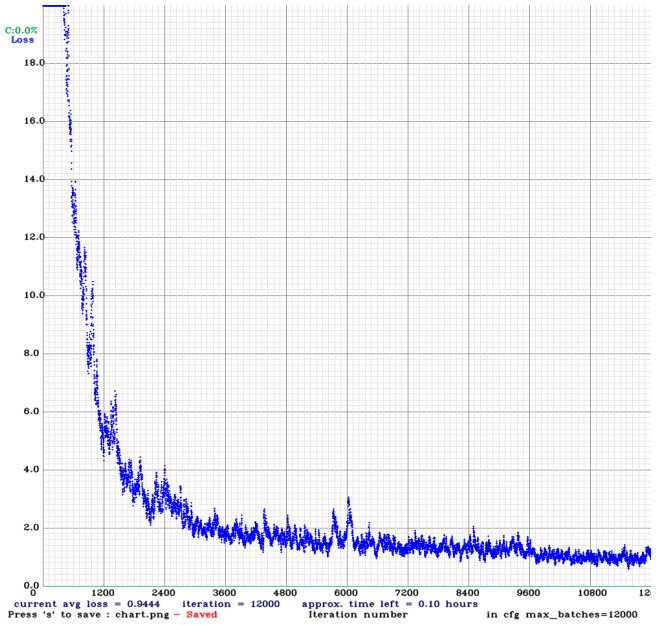

Fig. 10 Training Graph of YOLOv4

Fig. 11 presents the result of the experiment using YOLOv7-tiny. Fig 11 shows the training process graph of YOLOv7-tiny. Unlike other models, which have a steady line of loss percentage with the only difference of how constant the loss percentage occurs, this training process has a particular case. After around the 300th iteration, the loss percentage increases again until the 700th iteration. This case only takes place in this training process. Then, the loss percentage kept decreasing steadily like the training process of CSResnext50-Panet-SPP until the 9600th iteration. From there on, the loss percentage stays in the same line of loss percentage until the very end of the iteration.

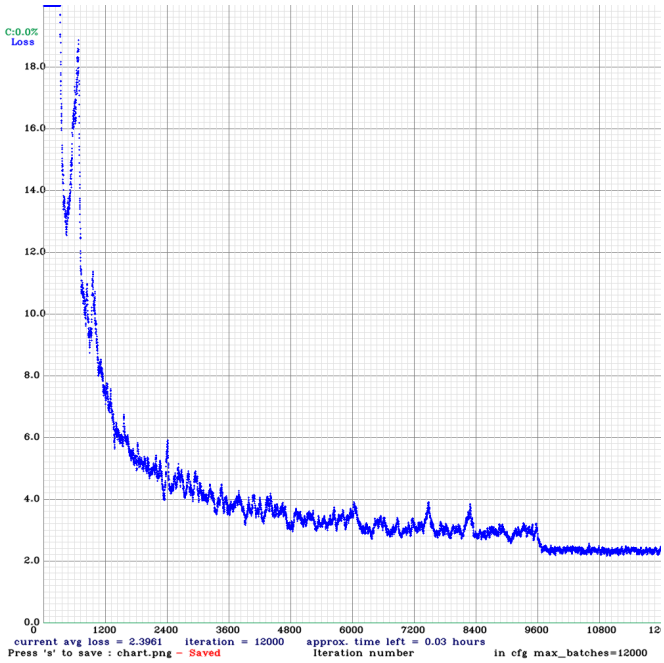

Fig. 11 Training Graph of YOLOv7-tiny

Every CNN model resulted in a different training graph, with a different training duration and training loss percentage. The only similarity is the total number of iterations, which is 12000.

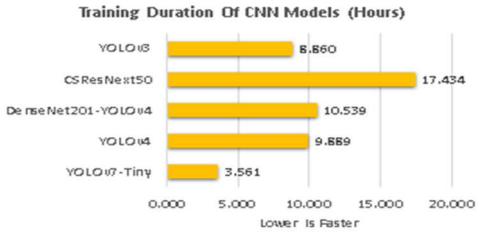

Fig. 12 Training Duration of CNN Models

The training graph shows the training duration of each CNN model. Fig. 12 presents the training duration of every CNN model. In this case, the lower the duration, the better the CNN models because of the low wait time. YOLOv7-Tiny has the best training duration, with only 3.561 hours. The worst training duration came from CSResNext50, with 17.434 hours. The rest of the models have almost the same duration, ranging from 8.5 to 10.5 hours.

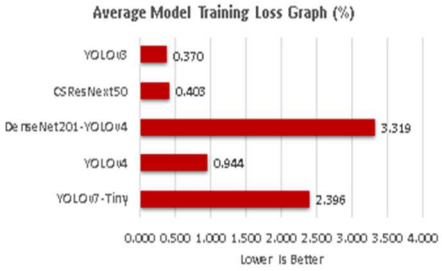

Fig. 13 Average Model Training Loss Graph

The training graph also shows the training loss of each CNN model. Fig 13 depicts the average training loss of every CNN model. The lower the value of the training loss, the better the quality of the model. Three models have a score under 1.0: YOLOv3, CSResNext50, and YOLOv4. The lowest training loss score is YOLOv3, 0.37, while DenseNet201-YOLOv4 has the highest score, with a value of 3.319.

The first three metrics that could affect the model's performance are precision, recall, and F1-score. Fig 14 shows each CNN model's precision, recall, and F1-score metrics. Compared to other models, YOLOv4 has the highest value in all three metrics, with a score of 0.99 for precision, 1.0 for recall, and 1.0 for F1-score. All other CNN models have decent values, with an over 0.9 score in all metrics, except for YOLOv7-Tiny, which has a value of less than 0.9 for all metrics.

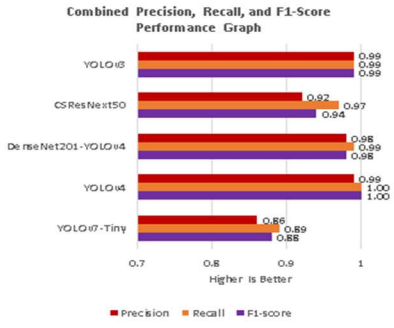

Fig. 14 Combined Precision, Recall, and F1-Score Performance Graph

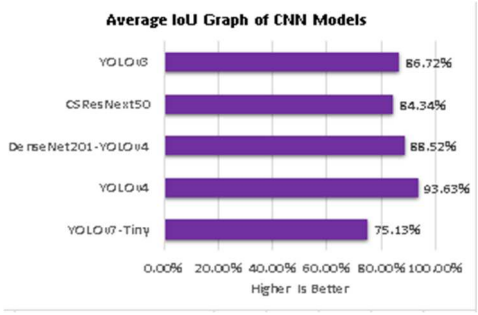

Fig. 15 Average IoU Graph of CNN Models

The next metric that could affect the performance of a CNN model is the average Intersection of Union or IoU. For this metric, the higher the percentage value, the higher the performance of a CNN model. Fig 15 shows the average IoU of CNN models. All models have a high IoU percentage of over 75%. The highest average IoU value of all five CNN models is YOLOv4, with a score of 93.63%. In comparison, the lowest average IoU is YOLOv7-Tiny, with a score of 75.13%.

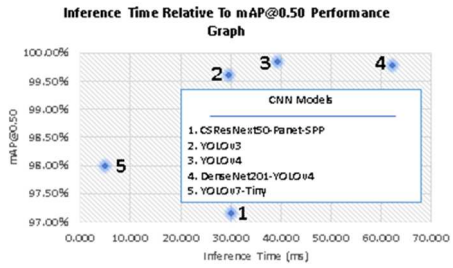

Fig. 16 Inference Time, Relative to mAP@0.50 Performance Graph

The last metric is the inference time relative to mAP@0.50. Fig. 16 illustrates the inference time relative to the mAP@0.50 performance graph of every CNN model. The inference time represents how fast a model detects the targeted object. The lower the inference time, the faster the detection becomes. However, mAP also needs to be considered to know more about compatibility on such devices. The CNN model with the fastest inference time and high compatibility is DenseNet201-YOLOv4. YOLOv3 and YOLOv4 have a high mAP but have a lower inference time than DenseNet201-YOLOv4. YOLOv4-Tiny has the lowest inference time with a mAP of around 98%, while CSResNext50-Panet SPP has the lowest mAP percentage with a value of around 97%.

B. Discussion

During the experiment, a total of five different CNN models were tested to evaluate their performance in detecting cars (*mobil*) and motorcycles (*motor*) as the targeted objects. Upon completion of the experiment, it was observed that each CNN model produced varying predictions when presented with the same test video.The CNN model exhibits precise predictions for objects falling within three distinct categories: cars (*Mobil*), motorcycles (*motor*), and people (*orang*). An illustration of these predictions is presented in Fig 17.

In this frame, the model detects a total of five cars (*Mobil*), seven motorcycles (*motor*), and four people (*orang*). Compared to CSResNext50, DenseNet201 detects one person more but one motorcycle (*motor*) and one car (*Mobil*) less. DenseNet201 could detect a person who was a lot further from the road. In this frame, the other motorcycle (*motor*) was a lot more unclear than in Fig. 17 due to the possibility that the vehicle was moving too fast. The car (*mobile) that was on* the left side of the road and was the furthest from the camera could not be detected. The problem could be with the tree that blocks some parts of the car (*mobil*).

Fig. 17 CSResNext50 Prediction Results to detect the object

Showcasing the results obtained through the employment of the CSResNext50 CNN model. In this particular frame, the model successfully recognizes a total of six cars (*Mobil*), eight motorcycles (*motor*), and three individuals (*orang)*. Its primary focus lies in identifying moving vehicles, disregarding stationary parked ones. Furthermore, the model targets explicitly individuals situated close to the road. Within the context of this figure, three classes of objects are observable, namely people (*orang*), cars (*mobil*), and motorbikes (*motor*). However, there exist three other classes of objects that remain unseen, including trucks (*truk*), bicycles (*sepeda*), and public transportation (*angkot*). The detection of public transportation *(angkot)* and cars (*mobil)* proves to be a challenging task for this model, resulting in a somewhat similar visual appearance between the two classes.

A model could encounter problems detecting objects if the target object is blocked by another object in the frame or if the target object moves too fast for the model to detect. Fig. 18 shows the prediction when using the DenseNet201 CNN model.

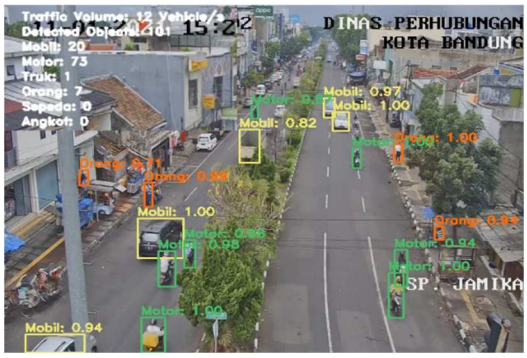

Fig. 18 DenseNet201 Prediction Results to detect the object

In recent object detection research, a CNN model was used to process faces when two target objects were too close to each other. One of the advantages of this model is its ability to detect multiple objects in close proximity. For example, in Fig. 19, the YOLOv3 CNN model was used for prediction. In this frame, the model successfully detected a total of five cars

(*mobil*), seven motorcycles (*motor*), and four people (*orang*). It was able to detect a person who was far from the road and all seven visible motorcycles (*motor*). However, it failed to detect the car (*mobil*) on the right side of the road that was trying to overtake the car (*mobil*) in front. object detection research using a CNN model has shown promising results in detecting multiple objects, even when they are close to each other. For example, the YOLOv3 CNN model could detect various objects in a given frame. However, there are still challenges in accurately detecting all objects, especially in complex scenarios or occluded by other objects. Further research and advancements in object detection models are continuously being made to improve their performance and accuracy.

Fig. 19 YOLOv3 Prediction Results to detect the object.

In addition to the challenge of detecting target objects, CNN models can also make incorrect predictions. An example of this can be seen in Fig. 20, where the YOLOv4 CNN model was used for prediction. In this frame, the model detected a total of five cars (*mobil*), eight motorcycles (*motor*), one public vehicle (*angkot*), and five people (*orang*). However, unlike the previous model that struggled with object detection, YOLOv4 misinterpreted the car (*mobil*) on the left road, furthest from the camera, as a person instead of a car. CNN models, like YOLOv4, are designed to learn and recognize image patterns through extensive training on large datasets.

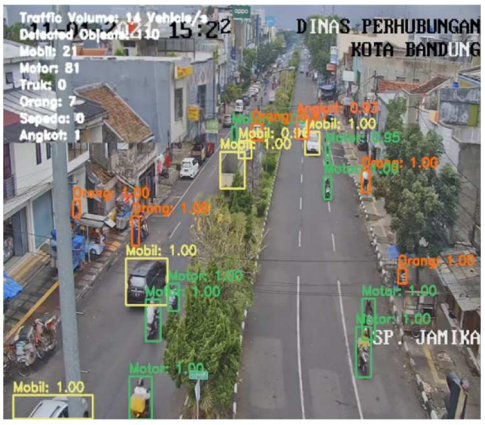

Fig. 20 YOLOv4 Prediction

Fig. 21 showcases the prediction results obtained using the YOLOv7-tiny CNN model. In this frame, the model successfully detects four cars (*mobil*), three motorcycles (*motor*), one public vehicle (*angkot*), and four people (*orang*). However, similar to YOLOv3, YOLOv7-tiny encounters a specific issue. It fails to detect the motorcycle (*motor*) located next to the car (*mobil*) on the left side, closest to the camera. YOLOv7-tiny is a variant of the YOLO object detection model. It is optimized for edge GPU devices and is designed to be lightweight, making it suitable for real-world computer vision applications and distributed systems. While YOLOv7-tiny offers faster inference times, it may struggle with detecting certain objects that are small or far away.

Fig. 21 YOLOv7-tiny Prediction results to detect

TABLE III
TOTAL OBJECTS DETECTED IN THE LAST FRAME (VOLUME)

Class	YOLO v7-Tiny	YOLO v4	DenseNet 201-YOLOv4	CSResNext50	YOLO v3
Car	4	5	5	6	5
Motorcycle	3	8	7	8	7
Truck	0	0	0	0	0
People	0	5	4	3	4
Bicycle	0	0	0	0	0
Public Transportation	1	1	0	0	0
SUM	8	19	16	17	16

Table III shows the total number of objects detected in the last frame of the video. The last frame of the video does not show either trucks or bicycles. Therefore, there is not a single model that detects those vehicles. In Table III, YOLOv4 resulted with the most objects detected out of all five CNN models, detecting five cars, eight motorcycles, five people, and one public transportation. The lowest total object detected was when using YOLOv7-tiny with four cars, three motorcycles, and one public transit.

TABLE IV
TOTAL OBJECTS DETECTED IN THE LAST 12 SECONDS

Class	YOLO v7-Tiny	YOLO v4	DenseNet 201-YOLOv4	CSResNext50	YOLO v3
Car	22	21	20	19	22
Motorcycle	47	81	73	43	65
Truck	0	0	1	0	0
People	0	7	7	8	6
Bicycle	0	0	0	0	0
Public Transportation	0	1	0	0	0
SUM	69	110	101	70	93

Table IV informs the total objects based on the experiment that were detected in the last 12 seconds of the test video. The last 12 seconds of the test video without any bicycle in it. Therefore, all of the models resulted in zero results of detecting a bicycle. The Table shows that YOLOv4 detects the most vehicles, with 21 cars, 81 motorcycles, 7 people, and 1 public transportation. The lowest detection rate is the YOLOv7-tiny, with only 22 cars and 47 motorcycles detected.

Table III and Table IV inform that YOLOv4 is the most accurate in detecting targeted objects compared to other CNN models. However, the number of total objects that YOLOv4 detected is not correctly predicted. For example, the model incorrectly predicts a car into a person. Therefore, other metrics such as precision, recall, and f1-score, which calculate the model's accuracy, are needed to determine their overall performance.

Besides how many objects a model could detect in a specific time length, inference time relative to the mAP of a CNN model and the average IoU could also affect their performance. In Fig 16, all CNN models are highly compatible with a mAP of above 97%. YOLOv4 has the highest compatibility but has the second-highest inference time and the highest average IoU, according to Fig 15. In conclusion, among five CNN models, YOLOv4 performs better when detecting objects and has high compatibility and a high inference time that could slow down the detection process.

IV. Conclusion

An experiment was conducted to determine the most effective method or algorithm for detecting objects at intersections of highways and calculating their volume. The study focused on six objects: cars, motorbikes, bicycles, trucks, people, and public transportation. These objects were detected in the same testing environment. The training process utilized darknet algorithms to detect these six classes of objects at the intersection. On the other hand, the testing process employed five CNN models: YOLOv7-Tiny, YOLOv4, DenseNet201-YOLOv4, CSResNext50, and YOLOv3. The purpose was to identify which CNN model produced the most accurate results, including determining the volume of each object passing through the intersection.

The experiment took place at a busy intersection in Bandung, West Java. Five CNN models were used: YOLOv3, YOLOv4, CSResNext50, DenseNet201-YOLOv4, and YOLOv7-Tiny. The experiment results showed that each model achieved an accuracy rate of over 87%. Notably, the YOLOv3 and YOLOv4 models achieved an accuracy rate of over 98%. This indicates that each CNN model had a sufficiently high accuracy to detect objects correctly. However, it should be noted that particular objects may not be detected if they are blocked or overlapped by other objects or if they are non-targeted objects, such as trees. Additionally, fast-moving objects may challenge the CNN models, resulting in detection errors. To address this issue, it is necessary to optimize the CNN models or explore alternative models better suited for this purpose. Further research should focus on testing the capabilities of these models. Based on the experiment's findings, it can be concluded that the chosen CNN models demonstrated high accuracy in detecting objects at the intersection. However, improvements are needed to address certain limitations and optimize the models for future research.

References

[1] A. Desmi, L. A. Widari, and R. Yanti, "Efektifitas Model Karakteristik Arus Lalu Lintas pada Ruas Jalan Simpang 4 Bireun (Perbandingan dengan Metode Greenshield, Greenberg, Underwood)," *Teras Jurnal : Jurnal Teknik Sipil*, vol. 9, no. 1, p. 19, Apr. 2019, doi:10.29103/tj.v9i1.178.

[2] G. Prati, V. Marín Puchades, M. De Angelis, F. Fraboni, and L. Pietrantoni, "Factors contributing to bicycle–motorised vehicle collisions: a systematic literature review," *Transport Reviews*, vol. 38, no. 2, pp. 184–208, Apr. 2017, doi: 10.1080/01441647.2017.1314391.

[3] J. Guerrero-Ibáñez, S. Zeadally, and J. Contreras-Castillo, "Sensor Technologies for Intelligent Transportation Systems," *Sensors*, vol. 18, no. 4, p. 1212, Apr. 2018, doi: 10.3390/s18041212.

[4] M. Won, "Intelligent Traffic Monitoring Systems for Vehicle Classification: A Survey," *IEEE Access*, vol. 8, pp. 73340–73358, 2020, doi: 10.1109/access.2020.2987634.

[5] J. Bai and Y. Chen, "A Deep Neural Network Based on Classification of Traffic Volume for Short‐Term Forecasting," *Mathematical Problems in Engineering*, vol. 2019, no. 1, Jan. 2019, doi:10.1155/2019/6318094.

[6] J. Liu et al., "Secure intelligent traffic light control using fog computing," *Future Generation Computer Systems*, vol. 78, pp. 817–824, Jan. 2018, doi: 10.1016/j.future.2017.02.017.

[7] N. Kumar, S. S. Rahman, and N. Dhakad, "Fuzzy Inference Enabled Deep Reinforcement Learning-Based Traffic Light Control for Intelligent Transportation System," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 4919–4928, Aug. 2021, doi:10.1109/tits.2020.2984033.

[8] A. Boukerche and J. Wang, "Machine Learning-based traffic prediction models for Intelligent Transportation Systems," *Computer Networks*, vol. 181, p. 107530, Nov. 2020, doi:10.1016/j.comnet.2020.107530.

[9] J. Wang, X. Guo, and X. Yang, "Efficient and Safe Strategies for Intersection Management: A Review," *Sensors*, vol. 21, no. 9, p. 3096, Apr. 2021, doi: 10.3390/s21093096.

[10] H. Wei, G. Zheng, H. Yao, and Z. Li, "IntelliLight: A reinforcement learning approach for intelligent traffic light control," *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2496–2505, Jul. 2018, doi:10.1145/3219819.3220096.

[11] R. I. Meneguette, R. E. De Grande, and A. A. F. Loureiro, Intelligent Transport System in Smart Cities. Springer International Publishing, 2018. doi: 10.1007/978-3-319-93332-0.

[12] A. T. Wecksler, L. Veale, M. Basanta-Sanchez, and M. Bern, "Development of Software Workflow for the Rapid Detection of Cross-Linked Dipeptides," *Journal of the American Society for Mass Spectrometry*, vol. 33, no. 3, pp. 598–602, Feb. 2022, doi:10.1021/jasms.1c00312.

[13] S. Pradhan and S. Tripathy, "FRAC: a flexible resource allocation for vehicular cloud system," *IET Intelligent Transport Systems*, vol. 14, no. 14, pp. 2141–2150, Dec. 2020, doi: 10.1049/iet-its.2020.0390.

[14] J. Tang, F. Gao, F. Liu, and X. Chen, "A Denoising Scheme-Based Traffic Flow Prediction Model: Combination of Ensemble Empirical Mode Decomposition and Fuzzy C-Means Neural Network," *IEEE Access*, vol. 8, pp. 11546–11559, 2020, doi:10.1109/access.2020.2964070.

[15] Y. Chen and Z. Li, "An Effective Approach of Vehicle Detection Using Deep Learning," *Computational Intelligence and Neuroscience*, vol. 2022, pp. 1–9, Jul. 2022, doi: 10.1155/2022/2019257.

[16] Q. Abu Al-Haija and M. Krichen, "A Lightweight In-Vehicle Alcohol Detection Using Smart Sensing and Supervised Learning," *Computers*, vol. 11, no. 8, p. 121, Aug. 2022, doi: 10.3390/computers11080121.

[17] A. A. Alsulami, Q. Abu Al-Haija, A. Alqahtani, and R. Alsini, "Symmetrical Simulation Scheme for Anomaly Detection in Autonomous Vehicles Based on LSTM Model," *Symmetry*, vol. 14, no. 7, p. 1450, Jul. 2022, doi: 10.3390/sym14071450.

[18] S. Chen, M. Klemp, J. Taghia, U. Kühnau, N. Pohl, and R. Martin, "Improved Target Detection Through DNN-Based Multi-Channel Interference Mitigation in Automotive Radar," *IEEE Transactions on Radar Systems*, vol. 1, pp. 75–89, 2023, doi:10.1109/trs.2023.3279013.

[19] B. G. Rajagopal, N. Vishakraj, N. U. Kumar, and P. Jothivenkatesh, "Vision-based system for counting of moving vehicles in different

weather conditions," *2017 International conference of Electronics, Communication and Aerospace Technology (ICECA)*, pp. 86–91, Apr. 2017, doi: 10.1109/iceca.2017.8203649.

[20] M. A. El-Khoreby and S. Abd Rahman Abu-Bakar, "Vehicle detection and counting for complex weather conditions," *2017 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, pp. 425–428, Sep. 2017, doi:10.1109/icsipa.2017.8120648.

[21] S. Sengupta et al., "A review of deep learning with special emphasis on architectures, applications and recent trends," *Knowledge-Based Systems*, vol. 194, p. 105596, Apr. 2020, doi:10.1016/j.knosys.2020.105596.

[22] V. K. Sharma and R. N. Mir, "A comprehensive and systematic look up into deep learning based object detection techniques: A review," *Computer Science Review*, vol. 38, p. 100301, Nov. 2020, doi:10.1016/j.cosrev.2020.100301.

[23] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object Detection With Deep Learning: A Review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212–3232, Nov. 2019, doi:10.1109/tnnls.2018.2876865.

[24] J. Zhao et al., "Autonomous driving system: A comprehensive survey," *Expert Systems with Applications*, vol. 242, p. 122836, May 2024, doi:10.1016/j.eswa.2023.122836.

[25] A. Theissler, J. Pérez-Velázquez, M. Kettelgerdes, and G. Elger, "Predictive maintenance enabled by machine learning: Use cases and challenges in the automotive industry," *Reliability Engineering & System Safety*, vol. 215, p. 107864, Nov. 2021, doi:10.1016/j.ress.2021.107864.

[26] J. Zhao et al., "Improved Vision-Based Vehicle Detection and Classification by Optimized YOLOv4," *IEEE Access*, vol. 10, pp. 8590–8603, 2022, doi: 10.1109/access.2022.3143365.

[27] Q. A. Al-Haija, M. Gharaibeh, and A. Odeh, "Detection in Adverse Weather Conditions for Autonomous Vehicles via Deep Learning," *AI*, vol. 3, no. 2, pp. 303–317, Apr. 2022, doi: 10.3390/ai3020019.

[28] M. I. Chacon-Murguia and A. Guzman-Pando, "Moving Object Detection in Video Sequences Based on a Two-Frame Temporal Information CNN," *Neural Processing Letters*, vol. 55, no. 5, pp. 5425–5449, Nov. 2022, doi: 10.1007/s11063-022-11092-1.

[29] C. Amisse, M. E. Jijón-Palma, and J. A. S. Centeno, "Fine-Tuning Deep Learning Models for Pedestrian Detection," *Boletim de Ciências Geodésicas*, vol. 27, no. 2, 2021, doi: 10.1590/s1982-21702021000200013.

[30] T. Chen, M. Xu, X. Hui, H. Wu, and L. Lin, "Learning Semantic-Specific Graph Representation for Multi-Label Image Recognition," *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 522–531, Oct. 2019, doi: 10.1109/iccv.2019.00061.

[31] G. J. Ansari, J. H. Shah, M. Yasmin, M. Sharif, and S. L. Fernandes, "A novel machine learning approach for scene text extraction," *Future Generation Computer Systems*, vol. 87, pp. 328–340, Oct. 2018, doi:10.1016/j.future.2018.04.074.

[32] N. Jahan, S. Islam, and Md. F. A. Foysal, "Real-Time Vehicle Classification Using CNN," *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–6, Jul. 2020, doi: 10.1109/icccnt49239.2020.9225623.

[33] K. Khan, S. B. Zaidi, and A. Ali, "Evaluating the Nature of Distractive Driving Factors towards Road Traffic Accident," *Civil Engineering Journal*, vol. 6, no. 8, pp. 1555–1580, Aug. 2020, doi: 10.28991/cej-2020-03091567.

[34] P. Tassinari et al., "A computer vision approach based on deep learning for the detection of dairy cows in free stall barn," *Computers and Electronics in Agriculture*, vol. 182, p. 106030, Mar. 2021, doi:10.1016/j.compag.2021.106030.