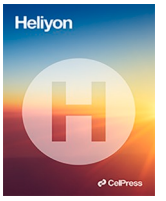
Heliyon 10 (2024) e35247

Contents lists available at ScienceDirect

Heliyon

journal homepage: www.cell.com/heliyon



Research article

A new CNN-BASED object detection system for autonomous mobile robots based on real-world vehicle datasets

Udink Aulia [a,b], Iskandar Hasanuddin [b], Muhammad Dirhamsyah [b], Nasaruddin Nasaruddin [c,*]

[a] Doctoral Program, School of Engineering, Post Graduate Program, Universitas Syiah Kuala, Banda Aceh, 23111, Indonesia
[b] Dept. of Mechanical and Industrial Engineering, Universitas Syiah Kuala, Banda Aceh, 23111, Indonesia
[c] Dept. of Electrical and Computer Engineering, Universitas Syiah Kuala, Banda Aceh, 23111, Indonesia

ARTICLE INFO

Keywords:
Autonomous mobile robot (AMR)
Transfer learning
CNN
Real-world datasets
Object detection

ABSTRACT

Recently, autonomous mobile robots (AMRs) have begun to be used in the delivery of goods, but one of the biggest challenges faced in this field is the navigation system that guides a robot to its destination. The navigation system must be able to identify objects in the robot's path and take evasive actions to avoid them. Developing an object detection system for an AMR requires a deep learning model that is able to achieve a high level of accuracy, with fast inference times, and a model with a compact size that can be run on embedded control systems. Consequently, object recognition requires a convolutional neural network (CNN)-based model that can yield high object classification accuracy and process data quickly. This paper introduces a new CNN-based object detection system for an AMR that employs real-world vehicle datasets. First, we create original real-world datasets of images from Banda Aceh city. We then develop a new CNN-based object identification system that is capable of identifying cars, motorcycles, people, and rickshaws under morning, afternoon, and evening lighting conditions. An SSD Mobilenetv2 FPN Lite 320 × 320 architecture is employed for retraining using these real-world datasets. Quantitative and qualitative performance indicators are then applied to evaluate the CNN model. Training the pre-trained SSD Mobilenetv2 FPN Lite 320 × 320 model improves its classification and detection accuracy, as indicated by its performance results. We conclude that the proposed CNN-based object detection system has the potential for use in an AMR.

1. Introduction

In recent years, rapid advancements have been made in terms of transporting materials. The transformation of automated guided vehicles (AGVs) into autonomous mobile robots (AMRs) is among the most significant of these advances. Since 1955, when the first AGVs were introduced, they have served as guidance systems for material-handling systems. AGVs have evolved into vision-based systems incorporating mechanical, optical, inductive, inertial, and laser technologies. A vision-based system such as this employs sensors, a powerful onboard computer, and artificial intelligence (AI), which allow the device to recognize and navigate its surroundings without the need to first define and implement reference points [1]. Autonomous vehicles include self-driving cars, robots, and other platforms that can sense their environment, interact with it, and navigate without human intervention. Nevertheless,

* Corresponding author.
E-mail address: nasaruddin@usk.ac.id (N. Nasaruddin).

https://doi.org/10.1016/j.heliyon.2024.e35247
Received 11 November 2022; Received in revised form 21 July 2024; Accepted 25 July 2024
Available online 26 July 2024

autonomous vehicles face several problems, including unknown environments, blind spots (invisible areas), non-line-of-sight scenarios, poor sensor performance due to weather conditions, sensor faults, false alarms, limited energy, limited computing resources, algorithm complexity, human-machine communication, size, and weight limitations. Several algorithmic approaches, including those for sensor design, processing, control, and navigation, have been developed to address this issue [2].

An AMR is a system that can operate in an uncertain and partially unknown environment. This means that the robot must be able to navigate its environment without interruption and avoid any obstacles. As a result, an AMR requires little or no human intervention to move and is programmed to follow a predetermined path in both indoor and outdoor environments [3]. It does not require external guidance devices such as wires, magnetic strips, or sensors, and can travel in dynamic situations without much assistance from outside sources. It also has cutting-edge sensors that can detect nearby objects. An intelligent system that can analyze the onboard image dataset of such a robot would allow for exploration of an area by loading an existing map or by creating one for use. As a result, AMRs can operate in almost any industrial setting [4]. Collisions between the mobile robot and obstacles must be avoided as the robot travels from its starting point to its final destination. The term 'robot navigation' refers to the ability of the robot to move through its environment to reach its destination without colliding with obstacles. A trajectory must be calculated to move the robot from its current location to its destination; this procedure requires a map, the location of the destination, and the robot's current location, as determined by sensors or other location systems. In addition, a suitable algorithm must be used to detect collisions between robots and obstacles in the working environment, thereby allowing the robot to alter its trajectory or stop before a collision. The obstacle avoidance algorithm therefore aids in preventing collisions [5].

Hospitality is an important sector in which service robots are starting to be adopted. Positive online reviews of hotel services are correlated with the use of service robots, and guests are encouraged to provide evidence of the services provided by robots in their reviews. The catering and delivery industry is a further example of the early commercialization of this technology; for example, malls and university cafeterias utilize delivery robots to reduce lines, thereby decreasing average delivery times. According to a study, implementing a contactless meal order and takeout service (CMOTS) can increase a company's profits by 95.4 % [6]. Several studies of the interaction between an AMR and the surrounding environment involve the use of sensory equipment for human detection and action recognition in industrial environments. Together with a human operator, collaborative behavior can be applied to AMRs carrying out a given task in an industrial space [7]. One study evaluated human perceptions of AMR courtesy behavior in industrial facilities, with a specific focus on crossing areas [8]. Research needs to be conducted on testing technologies for AMR, and to explore how digital twin technology can be used to test the operating environment of an AMR, robot navigation testing, and localization algorithms [9].

These robots are well-known for their intelligence and their ability to move independently, which enable them to react and make decisions in response to their environment. Therefore, they are employed in a range of industrial, transportation, and rescue applications. The planning of a trajectory is one of the most crucial aspects of the navigation of AMRs: researchers have worked on this problem over the past two decades, and numerous solutions have been developed. Path planning aims to determine a collision-free route between two points while minimizing the associated costs. This task can be divided into static and dynamic environments based on the characteristics of the environment: if the position of an obstacle does not change over time, it is referred to as static path planning, whereas if the position and/or orientation of the obstacle change over time, it is referred to as dynamic path planning [10]. In robotics and other associated fields, navigation is an important issue. For a mobile robot to navigate autonomously, it must be aware of its current location, its intended destination, and how to get there, meaning that navigation is the most important aspect to consider when designing a mobile robot. The objective is for the robot to travel from one location to another in a controlled or uncontrolled environment; however, in the majority of cases, a mobile robot cannot travel directly from its starting point to its destination, meaning that motion planning techniques must be implemented. The robot must therefore rely on processes such as perception (valuable data gathered by the mobile robot's sensors), localization (the robot's position and configuration), and cognition (decisions made by the mobile robot). In addition, goal attainment and motion control must be considered (i.e., the robot must estimate the input force to apply to the actuator to achieve the anticipated trajectory) [3].

Extensive datasets and a substantial amount of time are often needed to train a deep neural network (DNN) with a new model, and transfer learning is often used to reduce these requirements. In many instances, transfer learning begins with a network that has already been trained on a sizable dataset, such as ImageNet, and the network is then retrained (especially the classification layers) on the domain-specific dataset. Lower training times and significantly smaller datasets are needed for this. Various networks may be used for transfer learning, and a range of pretraining datasets can also be employed [11].

Real-world images can be gathered by the developer or obtained from external sources, with the two main types of images obtained from third parties being raw images and pre-existing datasets. Although search engines can be used to gather raw images, the developer will then need to label them. While search engines can offer sizable collections of images, there is a dearth of available metadata, and it may be that these photos were not taken under the best or most appropriate conditions, for example with suitable lighting. As a result, these images might make for a biased test dataset as well as a sub-optimal training dataset for the computer vision model, since they may only be available under a partial range of conditions, which will affect how well it performs. Due to the potential for producing inaccurate results, bias in test datasets must be considered. For instance, when a DNN was tested using publicly available datasets for vehicle detection, the results showed that the computer vision model performed well on images taken in comparable circumstances, such as in the same city; however, the computer vision model could not be generalized to cases outside the training dataset. The initial test dataset was therefore biased towards conditions similar to the training set when the performance was tested on datasets from alternative urban or rural settings. This may be acceptable for applications specific to the particular area in which the training images were collected, but is not suitable for more widespread use [11]. In addition to obtaining raw photos, several publicly accessible datasets created by prior researchers are available that cover a variety of disciplines, such as the common objects dataset

(COCO) [12], PASCAL [13], ImageNet [14], and the vehicle detection datasets CityScapes [15] and CamVid [16]. If real-world images are scarce or unavailable, synthetic images can be utilized. The images are computer-generated and produced from real-world images, proprietary 3D simulations, or a combination of real-world images and 3D computer models. Although synthetic images are typically easier to create and label than real datasets, the domain gap between the simulated and real environments often results in poor performance [11].

Research on deep learning has proliferated in recent years and has been used in many areas of application. Deep learning approaches have also been adopted for the tasks of localization and positioning; for example, a vision system based on the YOLO algorithm identifies static objects that can become obstacles in the path of a moving robot. To identify objects and their distances, objects must be determined in advance, such as chairs, wheelchairs, boxes, and tables [17]. A moving object detection system then uses images obtained from cameras mounted on objects, such as cars and mobile robots [18]. With the existing form of mobile robots, these problems are challenging to solve, due to the difficulties arising in various environments such as agricultural fields. It is, therefore, necessary to design a robot that can load and transport sufficient quantities of crops and that can move freely, even in tight spaces [19].

Developing an object detection system for an AMR requires a deep-learning architectural model with adequate accuracy, high inference speed, and a model that is small enough to be run on an embedded control system. One deep learning architecture model that meets the above requirements is the single-shot multi-box detector (SSD) and its variants. An SSD object detection model with a small number of parameters allows it to be used in real-time applications involving the detection of the human lower body using an AGV [20]. Vehicle detection has been carried out using improved SSD models [21], and one study of the detection of urban panel advertising compared the results from SSD and YOLO [22]. Studies of the use of SSD for Trident Feature and Squeeze have demonstrated the high accuracy of this approach for small object detection [23]. Furthermore, a CNN has been used to detect landmarks to determine the location of a mobile robot [24]; this was applied with object recognition in complex rural areas via an embedded graphics processing unit (Jetson Nano), which allows multiple neural networks to run simultaneously. Computer vision algorithms have also been applied to image recognition [25].

Many different types of objects may be present in a real-world environment, and the identification of such objects with the help of machines is a complex task. Computer vision is a branch of computer science that allows machines to see, identify, and process objects from still images and videos of their surroundings. A video is a continuous sequence of images (called video frames) that are displayed at a certain frame rate. Humans can immediately detect or recognize objects from images or videos and can determine their locations, as the neurons of the brain are interconnected. Video object detection is an AI task that uses the same process to detect objects. Object detection is an elementary step, and computer vision has been a major area of active research for decades, with various applications such as face detection, facial recognition, pedestrian counting, security systems, vehicle detection, self-driving cars, etc. [26]. Positioning can be carried out using several ultrasonic sensors for autonomously moving robots, where the coordinates are calculated according to the time ratio [27].

The feasibility and performance of AMRs are controlled completely from the cloud via a wireless connection [28], using a low-overhead real-time ego-motion estimation (visual odometry) system based on either a stereo or an RGB-D sensor [29], a prototype mining explorer robot has also been created with SLAM software and LIDAR sensor hardware [4]. Since many robots cannot independently diagnose whether localization results based on LIDAR are reliable, this may cause serious problems with autonomous navigation. A self-diagnosis scheme in which SVM is used to determine the localization status [30], develop a method of navigation for autonomous robots in dense environments for the data-driven prediction of robot-human interactions, and carry out experimental tests that can be reproduced under controlled conditions. This approach is called navigation with the local human-interaction prediction (NLHP) [31].

Sensors that detect and monitor vehicles contain three components: a transducer, a signal processing unit, and a data processing device. In an AMR, all three are very important to ensure that correct information about the surrounding environment is received. The sensors that are currently used in vehicles include monocular cameras, binocular cameras, light detection and ranging (LiDAR), global navigation satellite systems (GNSS), global positioning systems (GPS), radio detection and ranging (radar), ultrasonic sensors, odometers, and many more. However, each of these sensors has certain advantages and disadvantages: for example, LiDAR operates on the principle of radar but emits infrared light waves rather than radio waves. As a result, this sensor has much higher accuracy than radar below 200 m, although weather conditions such as fog or snow can negatively impact its performance [32].

Several previous authors have used a deep learning model with an AMR to detect objects, for example by developing a CNN-based method to detect distracted drivers and to identify the causes of distraction (talking, sleeping, or eating, via face and hand localization) [33]. A MobileNet-SSD architecture has been applied to detect damage to pallet racks [34]. An SSD with MobileDet Edge TPU and Mobilenet Edge TPU as the backbone was used to detect vine trunks, and the performance of this model was compared with that of MobileNet-V1 and MobileNet-V2 on agricultural datasets [35]. A system that allowed mobile robots to localize and navigate inaccessible areas of an indoor environment independently was developed in which the advantages of the convolutional neural network (CNN) model were exploited to extract data feature maps for image classification and visual localization. The CNN used a suitable cost function to overcome the limited generalizability of existing CNN classifier models due to the limited size of the dataset [36]. An efficient object navigation strategy was proposed in this paper, and a semantic association model was obtained using Mask R–CNN. An effective object navigation strategy was designed in another study to enable a robot to find a given target efficiently [37]. An open-source ground robot called SROBO was designed to accurately identify its position and navigate specific areas using deep CNNs and transfer learning. A pre-trained SSD-Mobilenet-v2 model was retrained on the SROBO control system using 100,000 images of indoor objects [38].

In the present paper, we propose a new CNN-based object detection system for an AMR using real-world vehicle datasets. The AMR has stringent requirements in terms of detecting road targets in real-time. This study aims to construct an object detection system for

the city of Banda Aceh using a deep learning model and to evaluate the post-training capabilities of this model, including its capacity for location prediction and object classification. Real-world datasets and an SSD Mobilenetv2 FPN Lite 320 × 320 CNN model are built for object detection. To the best of our knowledge, no prior researchers have examined the use of real-world datasets with several small images for object detection when applied to traffic conditions in Banda Aceh city. A previously-trained CNN model is therefore retrained using a real-world dataset to detect occluded and truncated objects. As a result of this research, it is envisaged that the accuracy of object localization, categorization, and detection can be enhanced, with rapid processing. We also present a CNN SSD Mobilenetv2 FPN Lite 320 × 320 architectural model that is trained on a real-world dataset of vehicle images, captured in Banda Aceh under morning, afternoon, and evening lighting conditions, and which contain four categories of objects: cars, motorcycles, people, and goods rickshaws. The dataset contains fewer than 500 images, and an augmentation approach is used to increase both the number of images and the number of objects per image, in order to prevent overfitting. The SSD Mobilenetv2 FPN Lite 320 × 320 model is pre-trained on the ImageNet dataset containing three of the four objects (excluding goods rickshaws). We then examine the effect of training the SSD Mobilenetv2 FPN Lite 320 × 320 CNN model on the dataset to detect all four types of objects on the streets of Banda Aceh. Finally, we evaluate the object detection and classification performance of the proposed CNN-based model. The contributions of our work can be summarized as follows:

1) We create a real-world dataset made up of a small number of images based on four object categories: cars, motorcycles, people, and goods rickshaws. The images were taken from various angles, at various shooting scales, and under various lighting conditions (morning, afternoon, and evening). The dataset also contains truncated and occluded objects.
2) We propose a new CNN-based SSD Mobilenetv2 FPN Lite 320 × 320 object detection model that is retrained on real-world datasets.
3) We compare the detection capabilities of the model before and after training on images taken under morning, afternoon, and evening ambient lighting. We also assess the object classification and detection performance of the proposed CNN-based system using both qualitative and quantitative techniques.

2. Method

The methodology used in this paper is shown in Fig. 1, where a real-world dataset is formed from images taken in several locations across the city of Banda Aceh. There are three main steps in this methodology. The first is the creation of a dataset containing image frames at several locations. This dataset is then subjected to initial processing to create images with a size of 800 × 600 pixels. Next, the SSD Mobilenetv2 FPN Lite 320 × 320 CNN model is trained, and the performance of the model is evaluated. In the last step, the final results of the model are produced, consisting of an image with the name of the object, a bounding box, and an estimate of the classification accuracy.

The steps in the research method used in this study are shown in more detail in Fig. 2. First, pictures are taken for the dataset using a camera under three lighting conditions (morning, afternoon, and evening). The next stage is the training process of the SSD Mobilenetv2 FPN Lite 320 × 320 CNN model. The result of this process is a checkpoint file that can be used to evaluate the model. Following training, the model is assessed both quantitatively and qualitatively. We evaluated the training results by looking at precision and recall values, measured as mean average precision (mAP), across 10 IoU thresholds from 0.05 to 0.95. We also examined the curves for

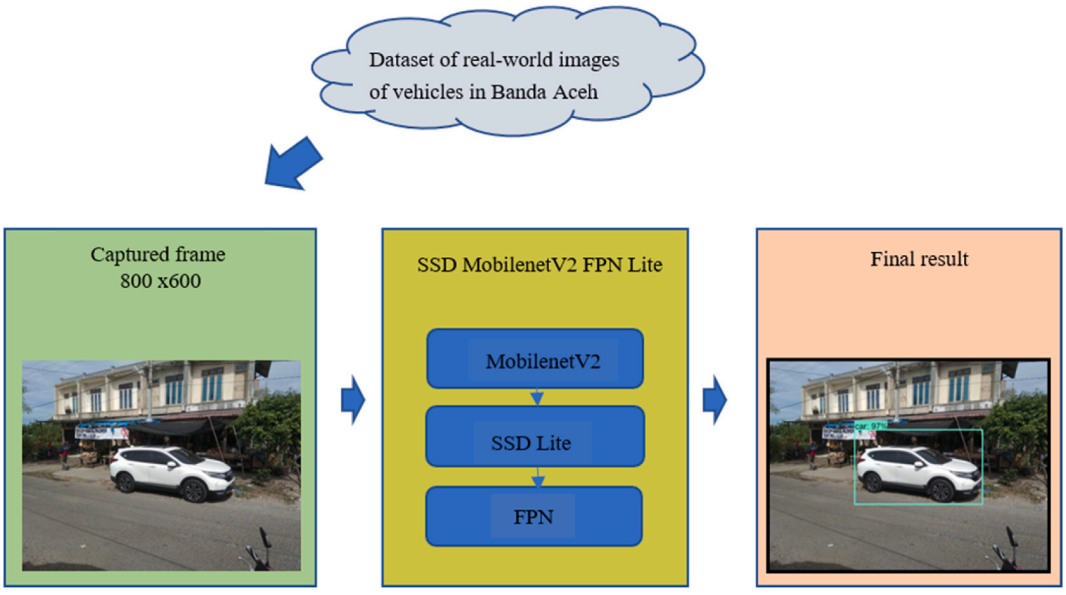


Fig. 1. Proposed methodology.

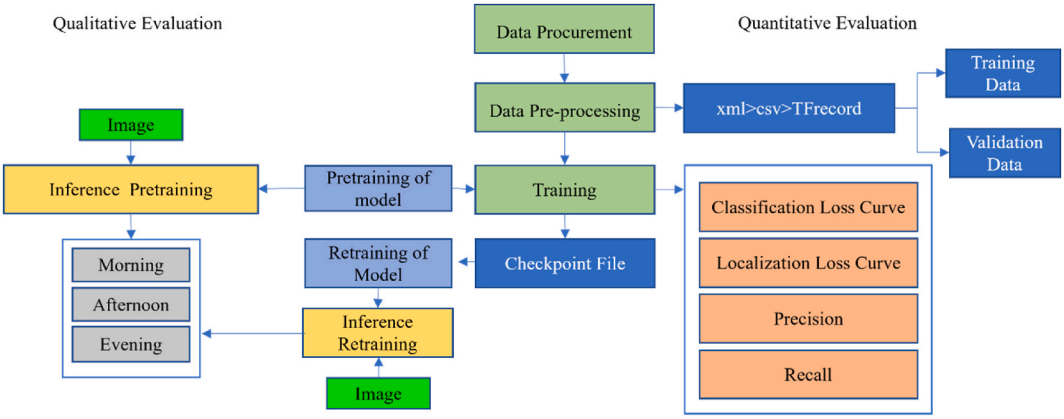


Fig. 2. Stages of the research methodology.

classification, localization, and regularization loss.

We assessed the model's quality by testing it with images before and after training. We compared how well it performs in three lighting conditions: morning, afternoon, and evening. An assessment was also carried out on the accuracy of the bounding boxes before and after the training process, using a real-world dataset consisting of images of vehicles in Banda Aceh.

2.1. Data procurement

To the best of our knowledge, no open-source images of representative road objects are available for Banda Aceh. We addressed this key need by collecting images of vehicles and creating the first road object dataset for Banda Aceh, which can be used to train object detection models. The vehicle dataset consisted of images taken with a camera to create a representative dataset. Images were taken while paying attention to the position of each vehicle and the size of the object, and were divided into three groups based on the lighting conditions (morning, afternoon, and evening). The total number of images in the datasets was 325, each with a size of 4160 × 3120 pixels.

2.2. Data pre-processing

The static dataset of images taken with the camera was pre-processed and reduced to a size of 800 × 600 pixels. Images were also taken from the internet, with sizes greater than 320 pixels for both the length and width. Image dimensions were reduced to enhance training efficiency and accelerate image processing during testing, leveraging the computational capabilities of a Google Colab GPU. The SSD Mobilenetv2 FPN Lite 320 × 320 model accepted an input image of size 320 × 320 × 3, thus requiring the sizes of all of the static images to be reduced. Next, the collected images were annotated with the LabelImage application, in order to apply bounding boxes to the dataset. Determining the bounding boxes is particularly important, as these will serve as a ground truth when the accuracy of the bounding box predictions is assessed during the training phase. The implementation of the bounding box was determined based on the nature of our dataset, and needed to be "tightly tied," focusing only on vehicles and people, in order to provide the training phase with sufficient data for generalization. An annotated dataset with bounding boxes for object detection consists of an image file with the appropriate bounding box annotation files. In this study, we used the PASCAL VOC format, and all datasets were converted accordingly. In this format, each image has a corresponding xml file containing all the annotation data; this makes it more convenient, as all the information is contained and labeled in the xml file. The next step was image pre-processing. When each image had been labeled, further processing was required, as each xml file needed to be converted to csv and finally to the TFRecords format for training of the model. The last step was to create a label_map.pbtxt file, to define the mapping of category ids to category names.

The dataset is divided into two main parts: training data for training the model and validation data for evaluating its performance. This was because the learning process was based on a small batch size (minibatch) for training and validation, meaning that each batch of training results could be validated to determine the training accuracy. The training data consisted of a jpg image file and an xml annotation file. For data testing, only jpg image files were needed. The training and validation datasets represented 80 % (261 images)

Table 1
Number of objects per dataset category.

Category	Number of objects per category
Cars	297
Motorcycles	330
People	370
Goods rickshaws	121

and 20 % (64 images) of the total, respectively. Table 1 shows the number of objects per dataset category used in the training process.

2.3. Training

2.3.1. Data augmentation

In the training process, an augmentation process was applied to the dataset, which consisted of a set of techniques to increase the size and quality of the training data. The goal was to increase the generalization capability of the model without changing the categories of images. Basic augmentation operations such as random cropping and horizontal flipping were performed. Random cropping of the image was carried out to create a random subset of the original sample. Random cropping in data augmentation randomly extracts different regions from images during training. This enhances a model's ability to recognize objects in various locations, sizes, and backgrounds, and offers better generalization and robustness.

Horizontal flipping was applied to the dataset to create mirror images, enhancing the dataset's diversity and promoting the model's ability to recognize objects from different perspectives. For example, the vehicles were mostly on the left side of the road, and this augmentation increased the generalization of objects to the right side of the road, and reversed the direction of the vehicle when parked (facing left or right).

The testing dataset was used to test the resulting model, after the training process was complete. The training process involved learning new data, thus allowing the model to carry out the vehicle object detection process. The model parsed the training and validation data stored in tfrecord format, subsequently leveraging a pre-trained model for training and validation. The objective was to derive a model characterized by minimized losses in both localization and classification through rigorous evaluation of the dataset.

The result of the training process was a checkpoint file, which was converted into a model for retraining. To assess the performance, the retrained and pre-trained models were used to apply an inferencing process to the image to produce bounding boxes, object names, and detection accuracy for images taken under three lighting conditions (morning, afternoon, and evening).

2.3.2. Architecture selection

When the geometric transformation of the dataset was complete, the next step was to select the architecture that would be used for training on the dataset. The pre-trained model was initially trained on a large ImageNet dataset, and was then trained to produce a checkpoint file. The checkpoint file was then converted into a model for retraining. The vehicle detection model was an SSD Mobilenetv2 FPN Lite 320 × 320, a variant of the single-stage detection algorithm.

Object detection algorithms can be divided into two categories, single-stage and two-stage methods, with the main difference being whether they include a region proposal step. Two-stage algorithms use a region proposal network (RPN) to generate ROIs in the first phase, and then send the region proposals to the pipeline for object classification and bounding-box regression, meaning that they have an advantage in terms of accuracy, although they are usually slower. In contrast, a single-stage algorithm only performs a single detection; the absence of a region proposal step is advantageous in terms of speed, but these models achieve a lower level of accuracy. Numerous object detection architectures are available for object detection, but most require a dedicated GPU, and are usually hosted in the cloud due to their high computing requirements. This is because they typically apply a two-stage detection methodology, where the architecture first generates region proposals and then performs object classification on each proposal.

Two-stage detection methods, including R–CNN [39], R–FCN [40], and the region pyramid network (RPN) [41], can be used to generate the desired regions, and classification is then performed on this region using a fully connected or convolution layer. While a two-stage model provides a higher level of accuracy, the increased computational load makes it unusable at the edge. In this research, we aimed to train a lightweight architecture that could perform inference in a closed environment (Jetson Nano) with high accuracy and in real-time. We therefore chose a one-shot detector, in which bounding box classification and regression are performed in a single step. SSD, a single-shot multi-box detector model, is a popular single-stage architecture for high-speed inference in cases where the highest level of accuracy is not the primary requirement [42].

2.4. SSD Mobilenetv2 FPN lite 320 × 320 single-shot multibox architecture

The chosen architecture was tested to investigate its advantages and disadvantages. The main drawbacks of the RCNN algorithm [43] are the difficulty of detecting objects on a small scale and incorrect localization. An SSD can overcome these limitations to some extent, as it performs both operations in one shot. This offers two significant benefits. Firstly, feature maps of different sizes (10 × 10, 5 × 5, 3 × 3, etc.) can be extracted from the scene, where small-scale feature maps are generally used to detect large objects, and large-scale maps to detect small objects (multi-reference detection). Secondly, anchor boxes with different scales and aspect ratios can be generated (multi-resolution detection). Another significant advantage of SSDs is their increased performance on low-resolution images, which helps to avoid the need for expensive sensors.

For real-time processing, SSD adopts a one-shot approach by simultaneously predicting category scores and bounding box coordinates across multiple default boxes. This integrated prediction mechanism eliminates the need for a separate RPN. The multi-scale functionality allows SSD to analyze and detect objects across multiple resolutions, enhancing its adaptability to diverse object scales and improving overall precision in object localization and classification. The integrations of default boxes and multi-scale functionality in SSD serve as critical enhancements, yielding a substantial acceleration in processing speed. Simultaneously, the incorporation of multi-scale functionality empowers SSD to effectively address variations in object sizes, contributing to its resilience in the face of lower-quality images.

An SSD network architecture was implemented with the VGG16 network as the backbone, which generated six feature maps with

various dimensions and aspect ratios for object detection on the back. A non-maximum suppression (NMS) mechanism was also applied to the detection output. The output with the highest confidence score was proposed in the case of multiple overlapping detection outputs. Although using VGG16 as the backbone provides useful feature extraction capabilities, its internal architecture is so deep, with more than 130 million parameters, that it is difficult to achieve real-time inference on edge devices with limited computing capacity. The base MobileNetV2 network included an SSD layer to deal with the problem of computational complexity as a result of using VGG16 as the backbone of the SSD detector. Google replaced the VGG16 architecture with the Mobilenet network [44], which led to improved real-time inference for SSD architectures. SSDLite, which was introduced in Mobilenetv2 [45], has a dramatically reduced number of parameters and computational cost. SSDs are very sensitive to the size of the bounding box, and perform much worse on smaller objects than on larger ones. However, they perform well on large objects, and are very powerful for different object aspect ratios. Classic single-stage detection methods, such as enhanced detectors, DPM, and newer methods such as SSD, evaluate almost 10^4 to 10^5 candidate locations per image; however, only some of these locations contain objects (i.e., the foreground), and the rest are just background objects. This gives rise to the issue of class imbalance [46], which causes two problems:

1. The training process is not effective because detectors can quickly and easily label most locations as unimportant background (easily negative). Unfortunately, this doesn't help the model learn valuable information.
2. The difficult negative samples can easily be classified into the wrong category. Although they generate small loss values individually, these can collectively overwhelm the calculated losses and gradients and lead to degenerate models.

RetinaNet is one of the best single-stage object detection models, and has been proven to work well with both compact and small-scale objects. SSD Mobilenetv2 FPN Lite 320 × 320 is a variant of RetinaNet, whose architecture, as shown in Fig. 3, was created by adding two improvements to the existing one-stage object detection model: feature pyramid networks (FPNs) and the focal loss (FL). FL is an improvement on the cross-entropy loss (CE), and was introduced to deal with class imbalance problems with single-stage object detection models. A background class imbalance issue with extreme foreground-background disparity arises in a single-stage model, attributed to the dense sampling of anchor boxes, which represent potential object locations. In RetinaNet, there may be thousands of anchor boxes at each layer of the pyramid, meaning that few will be assigned to ground-truth objects, while most will be background classes. These easy examples (with a high probability of detection) can collectively overwhelm the model, even though they produce small loss values. The use of FL reduces the loss of contributions from easy examples and increases the importance of correcting misclassified examples. This approach naturally solves the class imbalance problem because examples from the majority class are usually easy to predict, while those from the minority class are difficult because a lack of data or examples from the majority class dominates the loss and gradient process. Due to this similarity, the use of FL may solve both problems.

2.5. Experimental setup

In our experiments, we used a deep learning framework consisting of TFOD 2 (Tensor Flow Object Detection 2), Google Collaboratory, an Intel(R) Xeon(R) CPU @ 2.20 GHz, an Nvidia Tesla T4 GPU, 12.68 GB RAM, a 78.19 GB disk, libcudnn8 = 8.1.0.77, CUDA Version: 11.2, OpenCV v4.6.0, Tensorflow = 2.9.0.

Fig. 4 shows the experimental setup for the model training process. The experiment began with the data acquisition stage, in which we collected images to create a dataset. This was followed by pre-processing and augmentation of the images. The SSD Mobilenetv2 FPN Lite 320 × 320 single-stage object detection model architecture was then selected. The process of network training and validation was carried out, and the final stage involved testing the model on the images.

2.6. Performance evaluation

The performance of the proposed object detection system was evaluated both qualitatively and quantitatively. In the qualitative

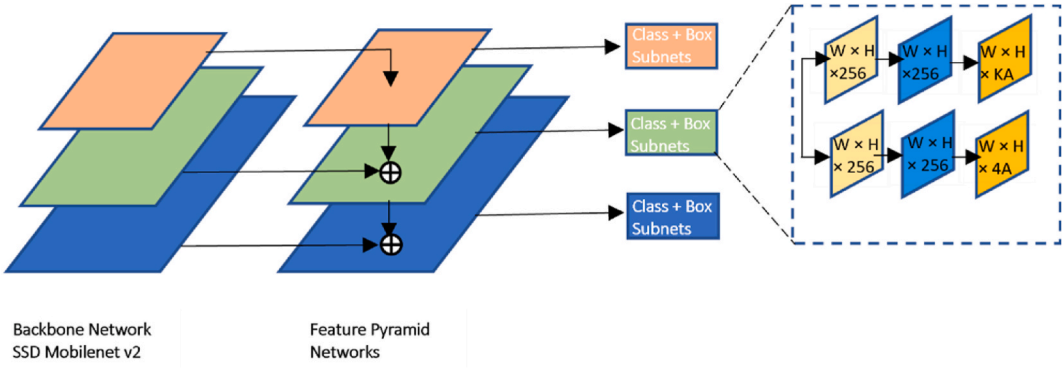


Fig. 3. Architecture of SSD Mobilenetv2 FPN lite 320 × 320.

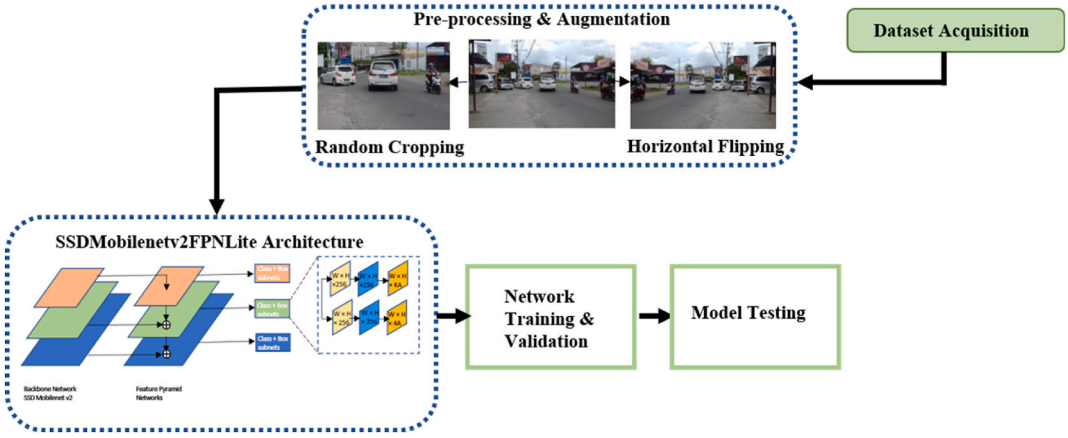


Fig. 4. Experimental setup used for training.

evaluation, we assessed the images resulting from the inference process in the model, whereas the quantitative evaluation is explained in the following. This evaluation was based on the localization losses for the bounding box offset predictions and the classification losses for the conditional class probabilities [42]. The loss L is calculated as the sum of the squared errors and is given in Equation (1).

$$L = \frac{1}{N}(L_{cls} + \alpha L_{loc}), \tag{1}$$

where N is the number of matching bounding boxes, L_{cls} is the classification loss, α is the balancing parameter for the weights between the two losses L_{cls} and L_{loc}, and L_{loc} is the localization loss. If $N = 0$, we set the loss L to zero. The localization loss is a smooth loss $L1$ that is applied in Fast RCNN [47] between the predicted box (p) and the ground truth box (g) parameters. It is also the loss between the predicted bounding box correction and the true value. The coordinate correction transformation is the same as that applied by R–CNN in the bounding box regression process. A robust $L1$ is less sensitive to outliers than the loss used in R–CNN and SPPnet. The smooth loss $L1$ is shown in Equation (2).

$$L_1^{smooth} = \begin{cases} 0.5x^2 & if\ |x| < 1 \\ |x| - 0.5 & otherwise \end{cases}, \tag{2}$$

where x is the predicted value minus the ground truth value. When the regression targets are unbounded, training with this loss may require careful tuning of the learning rates in order to prevent exploding gradients. Using Equation (3) can eliminate this sensitivity. L_{loc} is calculated in Equation (3) [47].

$$L_{loc} = \sum_{i,j} \sum_{m \in \{x,y,w,h\}} l_{ij}^{match} L_1^{smooth} \left(d_m^i - d_m^j \right)^2. \tag{3}$$

To perform bounding box regression, we use four coordinate parameters in Equations (4)–(7) [43].

$$t_x^j = (g_x^j - p_x^i)/p_w^i, \tag{4}$$

$$t_y^j = (g_y^j - p_y^i)/p_h^i, \tag{5}$$

$$t_w^j = \log(g_w^j / p_w^i), \tag{6}$$

$$t_h^j = \log(g_h^j / p_h^i), \tag{7}$$

where l_{ij}^{match} indicates whether the i-th bounding box with coordinates $(p_x^i, p_y^i, p_w^i, p_h^i)$ corresponds to the j-th ground truth box with coordinates $(g_x^j, g_y^j, g_w^j, g_h^j)$ for each object. $d_m^i, m \in \{x, y, w, h\}$, are the predicted correction terms for the center coordinates of the box and its width and height. The classification loss in Equation (8) is defined as the softmax loss applied to multiple classes [43]:

$$L_{cls} = -\sum_{i \in pos} l_{ij}^k \log(\hat{c}_i^k) - \sum_{i \in neg} \log(\hat{c}_i^0), \text{where } \hat{c}_i^k = \text{softmax}\ (\hat{c}_i^k), \tag{8}$$

where l_{ij}^k indicates whether the bounding box to i-th and the ground truth box to j-th corresponds to an object in class k, "pos" is the set of matching bounding boxes (N items in total), and "neg" is the set of negative examples. SSD uses hard negative mining to select easily

misclassified negative examples to create the neg set. Once all the anchor boxes have been sorted based on the objectiveness confidence score, the model selects the top candidates for training so that the ratio between negative and positive is at most 3:1.

A metric is used to measure the performance of the detection algorithm. In the case of object detection, the evaluation metric used here measures how close the detected bounding box is to the ground truth bounding box. This measurement is carried out independently for each object class by assessing the amount of overlap between the prediction area and the ground truth. Each detected bounding box must first be classified in order to calculate the precision and recall values, as follows:

a. A true positive (TP) is a correct detection of the ground-truth bounding box.
b. A false positive (FP) is a false detection of a missing object or misplaced object detection.
c. A false negative (FN) is an undetected ground truth bounding box.

Precision is the ability of a model to identify only relevant objects [48]. This is the percentage of correct positive predictions, and is calculated as Equation (9).

$$Precision = \frac{TP}{TP + FP}. \tag{9}$$

Recall is the ability of a model to find all relevant cases (all ground truth bounding boxes). It is the percentage of correct positive predictions out of the total number of ground truths. It is calculated as the ratio between the correctly classified positive samples and the total number of true positive samples: the more positive samples are detected, the higher the recall. It is expressed in Equation (10).

$$Recall = \frac{TP}{TP + FN}. \tag{10}$$

In addition to the precision and recall metrics, we also calculated the IoU and FL. IoU is defined as the intersection between a detected bounding box and the ground truth bounding box. The area of overlap between the predicted and ground truth boxes is divided by the union area, resulting in the IoU [48]. An IoU value of >0.5 is usually considered a good prediction. This metric determines how many objects were correctly detected and how many false positives were generated, and is defined in Equation (11).

$$IoU = \frac{Area\ of\ Overlap}{Area\ of\ Union}. \tag{11}$$

FL is an improved version of CE, which attempts to address the problem of class imbalance by assigning more weight to examples that are difficult or easily misclassified (i.e., backgrounds with noisy textures or partial objects, or objects of interest), and lowering the weight of easy examples (i.e., background objects) [46]. It can be calculated as shown in Equation (12).

$$FL(p_t) = -\alpha_t(1 - p_t)^{\gamma} \log p_t, \tag{12}$$

where p_t is the predicted confidence score for an object candidate, α is the weighting factor, and γ is the focusing parameter. $(1 - p_t)^{\gamma}$ is a modulating factor, with an optimum value for $\alpha = 0.25$ and $\gamma = 2$.

The F1 score is mathematically expressed as the harmonic mean of precision and recall, providing a symmetrical representation of both metrics [48]. It is presented as Equation (13).

$$F1\ score = \frac{2 * Precision * Recall}{Precision + Recall}. \tag{13}$$

Fig. 5. Results of image annotation with the LabelImage application.

3. Results

3.1. Model evaluation

In the model evaluation process, the first step is to perform an annotation process to ensure that the model does not contain errors. The output data consists of an image dataset and an xml file, which is the result of the process of labeling with the LabelImage application. In this way, the relationship between the image, the bounding box, and the label of each image is obtained. The xml file is subsequently transformed into a csv file and further processed into a tfrecord file for use in training and validation data. These data are then passed as input to the training process, in addition to the pbtxt file, to determine the number of classes, and the config file, to determine the hyperparameters during the training process. Finally, an annotation process (naming the bounding boxes for each object) is carried out for all four object classes (cars, motorcycles, people, and goods rickshaws). Fig. 5 shows the result of an image annotation process, where the bounding box coordinates and object names are stored in an xml file.

After the training process, the SSD Mobilenetv2 FPN Lite 320 × 320 deep learning model is evaluated based on a loss classification and localization loss graph, in which the loss curve produces a value of close to zero with an increase in the number of training steps. We then carried out experimental testing of the model's ability, before and after training, to detect the location, name, and accuracy of objects in images that were not included in the training process, under three different lighting conditions (morning, afternoon, and evening). Performance testing yields values for the average precision and average recall.

The results of the simulation are the elapsed time and the training loss. Elapsed time refers to the total amount of time it takes for the training process to complete. This includes the time taken for processing each epoch (complete pass through the entire training dataset), updating the model parameters, and iteratively improving the model's performance. The training loss, in the context of the SSD MobileNetV2 FPN Lite 320 × 320 deep learning model, refers to the quantitative measure of error incurred during the training process on the specified dataset. The objective of this training is to iteratively adjust the model's parameters to minimize this loss value, indicative of the dissimilarity between the model's predictions and the ground truth labels within the training dataset. The training phase yields distinct loss curves, delineating the classification and localization losses. As illustrated in Fig. 6, the classification loss converges to a final value of 0.0413515. This denotation of a notably low training classification loss suggests the model's proficiency, thereby signifying its potential for effective generalization in object classification tasks. The smaller the classification loss, the better the model's performance in terms of detecting objects. Fig. 7 shows the localization loss, which has a final value of 0.01855353. This value indicates the level of accuracy of the bounding box; the smaller the loss, the higher the IoU, and the more precise the object detection capability of the proposed model.

The optimal mapping from the inputs to outputs is determined by the weights learned by the neural network. Large weights may be indicative of an unstable network, in which minute changes in the input can result in significant changes in the output. This may indicate that the network has been overfitted to the training set, and is likely to perform poorly when applied to new data. This issue can be resolved by modifying the learning algorithm to encourage the network to keep the weights small. This process is shown in Fig. 8, and is referred to as weight regularization; it can be used as a general strategy to reduce the overfitting of the training dataset and to increase the generalizability of the model.

In addition to the classification and localization losses, Table 2 also shows low regularization losses, meaning that overfitting can be avoided in the training process. The iteration number is 16,000 steps. We conduct a comparative analysis of our training loss in comparison to that reported in prior research. In the previous study, a real-world object detection system was trained exclusively on synthetic data using YOLOv5, resulting in a training loss exceeding 0.01 [49]. The training losses of the D-CNN network model in the flow statistics module (STM) exceed 1.0 [50]. The loss function reflects the ability of a model to identify the input data items in a dataset correctly: the lower the loss, the better the classifier predicts the relationship between the input data and the output targets. It represents how accurately the algorithm can guess the correct class of a given object in a scenario involving classification loss [51]. The training losses of the YOLOv3 network model for detecting road hazards are below 0.04 [52]. The elapsed time for the proposed model

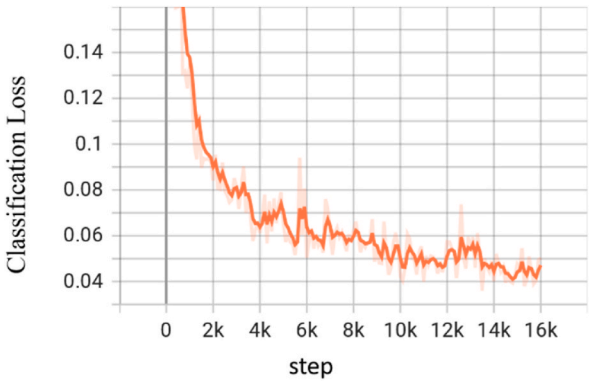


Fig. 6. Classification loss, calculated as a softmax loss over multiple classes for each example in the training set.

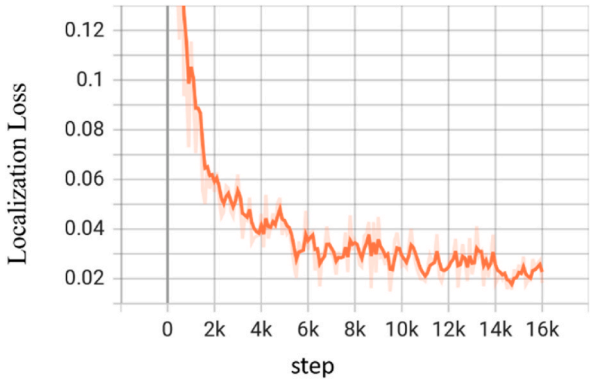


Fig. 7. Localization loss, calculated as a smooth L1 loss between the predicted bounding box correction and the true value.

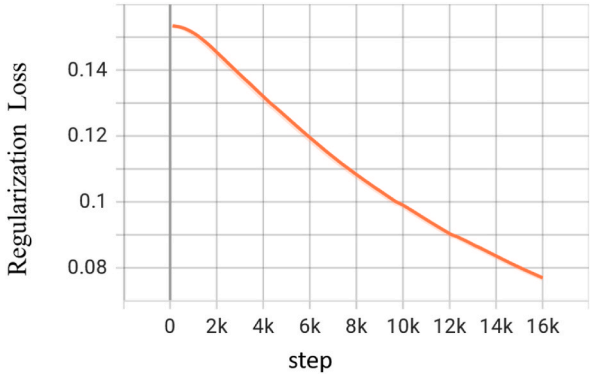


Fig. 8. Regularization loss based on weight regularization.

Table 2
Final loss values for the fine-tuned SSD Mobilenetv2 FPN Lite 320 × 320.

Classification loss	0.0413515
Localization loss	0.01855353
Regularization loss	0.07530993
Total loss	0.13521497
Elapsed time	2 h 9 min

was low, as the Google Collaboratory GPU was used, and the dataset considered in this study was not large. As a result, the elapsed time for the training process for the proposed model was lower than for comparable models in other studies [53].

3.2. Experimental results

The experiments were carried out on three types of images, which were taken in the morning, afternoon, and evening. This was done to explore the changes in the detection accuracy when there is a change in lighting. The test results for the morning, afternoon, and evening conditions are shown in Fig. 9(a and b), 10(a and b), and 11(a and b), respectively.

Fig. 9(and b) shows the experimental findings for the object classification accuracy, before and after the proposed deep learning model was trained on images taken in the morning. It can be seen that following training, the classification accuracy for cars increased from 73 % to 96 %, while for motorcycles, the classification accuracy increased from 75 % to 93 %. For people, the classification accuracy was 77 % prior to training, and 51 % after training.

The reason is that there are fewer pictures of people standing sideways in the datasets compared to images of people doing activities, like riding motorcycles. Before training, the accuracy for classifying motorcycles, bicycles, and benches was below 47 %, but the accuracy for goods rickshaws was unknown. After training, the accuracy for classifying goods rickshaws reached 100 %. The results of this experiment show that the classification accuracy of the proposed model is very high.

Fig. 10(a and b) compares the object classification accuracy before and after training of the proposed model on images taken in the

afternoon. Before training, the classification accuracy for cars ranged from 74 % to 75 %, but it was a perfect 100 % after completing the training. For motorcycles, the accuracy increased from 49 % to 97 % after training, whereas for people, the classification accuracy increased from 33 % to 75 %. Before training, the classification accuracy for goods rickshaws was at or below 45 %, while after training, it reached 100 %.

Fig. 11(a and b) compares the accuracy of object classification before and after training on images taken under evening conditions. The classification accuracy for cars was 70 % prior to training and 99 % afterward, while for motorcycle objects, it increased from 68 % to 100 %. For people, the classification accuracy was 73 % before training but approached 97 % after training. The classification accuracy for goods rickshaws reached 100 % after training; previously, these had been classified as motorcycles. The qualitative classification accuracy results for cars, motorcycles, people, and goods rickshaws, under the three conditions of morning, afternoon, and evening lighting, are shown quantitatively in Table 3.

The lighting condition was one of the factors taken into account to determine the classification accuracy. Fig. 9(a and b), 10(a and b), and 11(a and b) illustrate the effects of morning, afternoon, and evening illumination, respectively, on the classification accuracy of the CNN model. In the morning, the accuracy for identifying cars was 96 %, motorcycles 93 %, people 51 %, and rickshaws 68 % and 100 %. In the afternoon conditions, five object cars, three of which attain 100 %, one object 99 %, and one object 77 %. The object classification accuracy for motorcycles was 97 % and 57 %, whereas for people it was 75 %, and for goods rickshaws, it was 100 %.

During the evening evaluation as shown in Fig. 11(b), the classification accuracy for cars exhibited values of 99 %, 100 %, and 81 % across distinct scenarios. Accuracy in identifying people reached 97 %, while motorcycles and rickshaws achieved perfect accuracy at 100 %. Notably, the presence of shadows cast by roadside objects did not yield any discernible impact on classification accuracy across varying illumination conditions. Upon visual inspection, Fig. 10(b) displayed classification accuracies of 97 % for motorcycles and 75 % for people. In contrast, Fig. 11(b) showcased enhanced performance with 100 % accuracy for motorcycles and 97 % for people. These findings underscore the model's adaptability to diverse environmental conditions, affirming its efficacy in maintaining accurate classifications amid varying lighting and contextual scenarios.

The difference in precision is due to the fact that the motorcycle and people in Fig. 11(b) are shown on a large scale, whereas in Fig. 9(b) they are shown on a medium scale. The results in Table 4 indicate that the mAP value for large objects is 0.374, whereas, for medium-sized objects, it is 0.206. There is a linear relationship between the classification accuracy value and the mAP value, as the mAP for large objects is greater than for medium-sized objects. The higher the mAP, the greater the accuracy of classification. In the truncation condition depicted in Fig. 10(b), there were three car objects, with two achieving a perfect accuracy of 100 %, and one reaching 99 %, respectively. The high classification accuracy can be attributed to several images in the dataset depicting objects in truncation conditions at the image boundary, and the cropping process used to generate additional images for training and validation.

In terms of the detection accuracy for cars, we note that the proposed SSD Mobilenetv2 FPN Lite 320 × 320 was comparable to the CNN DP-SSD architecture [54]. The classification accuracy for vehicles with the DERD model, based on the YOLOv3 network, was reported to reach 75 % [50]. A vehicle detection model was implemented using 11 different extraction network models. However, their highest classification accuracy of 86 % was still inferior to that of the proposed SSD Mobilenetv2 FPN Lite 320 × 320, which achieved a perfect score of 100 % [55]. The average classification accuracy of our Mobilenetv2 FPN Lite 320 × 320 SSD reached 84 % for all objects (people objects were detected with the lowest value of 51 % in the morning conditions, while for the other 3 objects, the average accuracy value was higher), making it less accurate than Resnet 50 (86 %) and Darknet53 (85 %).

Lastly, another study compared various object models with five CNN models to classify cars and people [56]. All objects were classified by size, truncation, and occlusion, based on the KITTI criteria, into three levels of difficulty: easy, medium, and difficult. The highest classification accuracy for object classification was 82 %, i.e., lower than the proposed SSD Mobilenetv2 FPN Lite 320 × 320. The classification of people gave the highest classification accuracy of 56 %, as compared to a maximum of 97 % for the proposed SSD Mobilenetv2 FPN Lite 320 × 320.

Table 3
Object classification accuracy under different lighting conditions.

Condition	Object category	Accuracy (%)	
		Before training	After training
A. Morning	Car	73	96
	Motorcycle	75	93
	People	77	51
	Goods rickshaw	–	100
B. Afternoon	Car	74	100
	Motorcycle	49	97
	People	33	75
	Goods rickshaw	–	100
C. Evening	Car	70	99
	Motorcycle	68	100
	People	73	97
	Goods rickshaw	–	100

Fig. 9. Experimental object classification under morning illumination. (a) before training. (b) after training.

3.3. Performance of the proposed model

In this study, we created a real-world dataset and a new CNN model for use with AMRs. Previous studies have provided results only for the COCO dataset [49,52,53]. The mAP is a measurable parameter in COCO, and a 101-point interpolated definition of average precision (AP) is utilized in the calculation. The mAP is used to evaluate object detection models by comparing each ground-truth box to the detected box and delivering a score. The greater this score, the more precise the detection. Object detection models are typically tested using a variety of IoU thresholds, each of which may produce predictions that differ from the others. For COCO, the AP is the weighted average of the minimum IoU required to consider a match positive. AP@[0.50:0.05:0.95] represents the average AP for an IoU of between 0.5 and 0.95, with a step size of 0.05. In the COCO competition, the AP is the average of 10 IoU levels over 80 categories

(a) (b)

Fig. 10. Experimental object classification under afternoon illumination. (a) before training. (b) after training.

(AP@[0.50:0.05:0.95], i.e., ranging from 0.5 to 0.95 in increments of 0.05).

The precision reflects the ability of the model to identify only pertinent objects, whereas recall is the capacity of the model to locate all pertinent cases (all ground truth bounding boxes). It is the proportion of accurate positive predictions to the total number of ground truth boxes. In addition, IoU is defined as the intersection between the detection and ground truth bounding boxes: it is calculated by dividing the area of overlap between the predicted bounding box and the ground truth by the union area.

The precision and recall metrics were used to evaluate the performance of the proposed CNN model by the COCO guidelines for evaluation. The AP and average recall (AR) were computed for each object class, and the results are shown in Tables 4 and 5. There are four values each for AP and AR, spanning IoU thresholds from 0.50 to 0.95, evaluated across small, medium, and large areas.

Table 4 shows the performance results for the CNN model using a COCO-based evaluation. Each AP value is further subdivided into four areas: all, large, medium, and small. Our results demonstrate that the greater the area, the higher the average precision and the stronger the object detection capability of the proposed model. The overall detection capability of our model is shown to be excellent.

Table 5 shows the performance results for the average recall of the CNN model, which are also divided into four categories: all, large, medium, and small. There are 10 values of IoU, varying from 0.05 to 0.95 in increments of 0.05. The findings for the AR are also good, with the object detection capacity increasing as the area increases. The vehicle detection employed a CNN YOLOv5 model, and yielded a mAP of 0.258 at IoU(0.5), whereas with our SSD Mobilenetv2 FPN Lite 320 $\times$ 320, the mAP reaches 0.273, and the recall is 0.411 mean average recall (mAR) at IoU (0.50: 0.95) and 0.48 mAP at IoU (0.5) [51].

Precision and recall form the two components of the F1 score, which combines these metrics into a single statistic. The F1 score was also designed to work effectively with unbalanced data. Table 6 displays the average precision and recall evaluation ratings for the SSD Mobilenetv2 FPN Lite 320 $\times$ 320 following fine-tuning.

To observe the impact of the hyperparameters on the effectiveness of our CNN model, the number of iterations must be modified. Tables 7 and 8 show the changes in the AR and AP values when the original 16,000 iterations were increased to 50,000, for IoU values

Fig. 11. Experimental object classification under evening illumination. (a) before training. (b) after training.

Table 4
Results for average precision after fine-tuning the SSD Mobilenetv2 FPN Lite 320 × 320.

No.	IoU	Area	maxDets	Average precision (mAP)
1	0.50:0.95	All	100	0.273
2	0.50:0.95	Small	100	0.169
3	0.50:0.95	Medium	100	0.206
4	0.50:0.95	Large	100	0.374

Table 5
Results for average recall after fine-tuning the SSD Mobilenetv2 FPN Lite 320 × 320.

No.	IoU	Area	maxDets	Value of average recall (mAR)
1	0.50:0.95	All	100	0.411
2	0.50:0.95	small	100	0.293
3	0.50:0.95	Medium	100	0.348
4	0.50:0.95	Large	100	0.515

in the range 0.50:0.95 and four area conditions (all, small, medium, and large), with a max dets value of 100. The AP value for small areas increased from 0.169 to 0.171, while for medium areas it increased from 0.206 to 0.388, and for large areas, it increased from 0.374 to 0.477. The AR value for all regions increased from 0.411 to 0.497, while for the small condition, it decreased from 0.293 to

Table 6
F1-scores based on average precision and average recall results after fine-tuning the SSD Mobilenetv2 FPN Lite 320 × 320.

No.	Area	IoU	maxDets	Average precision	Average recall	F1-score
1	All	0.50:0.95	100	0.273	0.411	0.328
2	Small	0.50:0.95	100	0.169	0.293	0.214
3	Medium	0.50:0.95	100	0.206	0.348	0.258
4	Large	0.50:0.95	100	0.374	0.515	0.433

Table 7
Average precision results after fine-tuning the SSD Mobilenetv2 FPN Lite 320 × 320, for several iterations varying between 16,000 and 50,000.

No.	IoU	Area	maxDets	Value of average precision (mAP)	
				16,000 steps	50,000 steps
1	0.50:0.95	All	100	0.273	0.384
2	0.50:0.95	Small	100	0.169	0.171
3	0.50:0.95	Medium	100	0.206	0.388
4	0.50:0.95	Large	100	0.374	0.477

Table 8
Average recall results after fine-tuning the SSD Mobilenetv2 FPN Lite 320 × 320, for a number of iterations varying between 16,000 and 50,000.

No	IoU	Area	maxDets	Value of average recall (mAR)	
				16,000 step	50,000 step
1	0.50:0.95	All	100	0.411	0.497
2	0.50:0.95	Small	100	0.293	0.281
3	0.50:0.95	Medium	100	0.348	0.532
4	0.50:0.95	Large	100	0.515	0.569

0.281, for the medium condition it increased from 0.348 to 0.532, and for the large condition it increased from 0.515 to 0.569.

The number of iterations, or epochs, is a hyperparameter that determines how many times the learning algorithm will work through the entire training dataset. Increasing the number of epochs generally leads to a longer training time and higher computational cost. Training for too many epochs can be computationally expensive, especially for large datasets and complex architectures. In practice, determining the optimal number of iterations often involves monitoring the model's performance on a validation set during training and selecting the point where further training doesn't significantly improve mAP or may even lead to overfitting. It is a balance between ensuring the model has learned the relevant features from the data and preventing it from memorizing the training set.

The challenge with a small dataset is that the model may overfit to the limited examples. Transfer learning aims to mitigate this issue by leveraging knowledge from the source domain to provide a better initialization for the target domain. Transfer learning is influenced by the ability of the pre-trained model to leverage knowledge from a source domain (usually a larger dataset) and adapt it to a target domain (small dataset). A pre-trained model from a source domain may have achieved high classification accuracy on a larger dataset. The hope is that the learned features are transferable and can provide a good starting point for the target task.

By modifying the hyperparameters of the CNN model during training, a trade-off can be achieved between the accuracy of the proposed system and its computational requirements. Our SSD Mobilenetv2 FPN Lite 320 × 320, the mAP reaches 0.384, and the recall

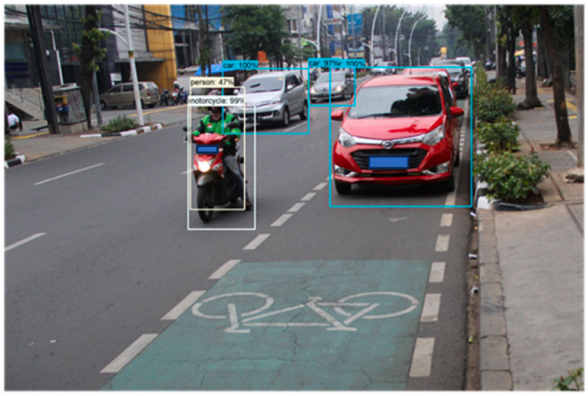


Fig. 12. Object classification experiment with images of Jakarta.

is 0.497 mean average recall (mAR) at IoU (0.50: 0.95) and the precision is 0.62 mAP at IoU (0.5).

In this paper, we considered a dataset for the city of Banda Aceh, and our approach required validation before it could be applied to other cities. To do this, two experiments were conducted, one based on images of the city of Jakarta, Indonesia, and the other based on images taken in European countries. Figs. 12 and 13 are from id.quora.com and Shutterstock, respectively. During the evaluation of our convolutional neural network (CNN) model on images of Jakarta, as illustrated in Fig. 12, the identification of three car objects yielded a classification accuracy exceeding 97 %. Simultaneously, the accuracy for recognizing motorcycles reached 97 %, while the classification accuracy for people was observed to be 47 %. This discrepancy in accuracy may be attributed to variations in the dataset distribution and the inherent challenges associated with the diverse visual characteristics of people. Upon applying the CNN model to images of European scenes, as exemplified in Fig. 13, the model demonstrated increased performance. Specifically, the model successfully recognized and classified two car objects with a classification accuracy achieving 100 %. This outcome suggests a notable proficiency of the model in adapting to different geographical contexts and datasets, potentially influenced by variations in environmental conditions, object appearances, and dataset characteristics between the two regions.

Although our CNN model cannot detect small objects, it yields acceptable performance for AMR applications, as the closest objects can still be recognized. Numerous scholarly works have presented neural network architectures that cannot be applied to mobile devices and embedded systems. In general, a large model such as VGGNet is used as a feature extractor, and novel functionality is added on top. The main issue with architectures such as VGGNet, ResNet50, and Inception is that a single pass through the network requires billions of calculations and tens of millions of parameters. To train the architectures on clusters, extremely powerful GPUs are required.

The proposed SSD is a sufficiently fast object detector to be used with real-time video. Our SSD is designed to be independent of the underlying network, allowing it to operate on top of almost any structure, including MobileNet. The object detection module of MobileNet + SSD relies on a variant called SSDLite, in which depthwise separable layers are used in place of regular convolutions. When SSDLite is used on top of MobileNet, it is easy to obtain genuinely real-time results (30 FPS or more). We not only feed the output of the final base network layer into the SSD layers but also feed the outputs of several previous layers. The MobileNet layers are responsible for converting the pixels from the input image into features that describe the content of the image and passing these to the other layers. Consequently, MobileNet is employed as a feature extractor in the second neural network. The MobileNet architecture is optimized for operation with mobile devices, and requires only up to 4 million parameters, significantly fewer than VGG (130 million) and ResNet50 (25 million). It also performs significantly fewer calculations: 300 MFLOPs as opposed to 4 GFLOPs or more for these larger variants. MobileNet is a suitable replacement for VGGNet due to its equivalent accuracy. Since object detection is a more complex process than classification, we add numerous additional convolutional layers to the SSD network. It is therefore essential to use a fast feature extractor, and MobileNet V2 provides precisely this.

A common method for determining the computational cost is to count FLOPs (the number of multiply-accumulates); however, a speed test on real devices also needs to be considered, as many other factors such as caching, I/O, hardware optimization, etc. can affect the actual time cost. SSDLite + MobileNetV2 320 × 320 has a model size (number of parameters) of 4.3M and a computational cost of 805 MFLOPs. A low-power embedded GPU (Jetson Nano) is used, as it enables the simultaneous operation of multiple neural networks and the application of a computer vision algorithm for image recognition. When models are trained on large-scale datasets, algorithms that use deep learning techniques incur high computational costs; however, this compact, powerful device enables the parallel execution of numerous neural network algorithms. It can assist in the development of applications such as audio detection, image classification, object detection, segmentation, and audio or speech processing.

With a quad-core 64-bit ARM CPU and a 128-core integrated NVIDIA GPU, the Jetson Nano delivers 472 GFLOPS of computing performance. Jetson is a new integrated accelerator hardware that is extensively used with machine learning algorithms. Its capabilities mean that it can be used in the development of AMRs, intelligent Internet of Things devices, and advanced AI systems, among other applications. In addition, a technique known as learning transfer can be implemented with Jetson Nano pre-trained networks that utilize machine learning. The main characteristics of this device are its portability and low power consumption. However, to maximize its capabilities and attain real-time performance, an optimization phase is required for both the hardware and the various algorithms used.

4. Discussion

We aimed to develop a model for a specific dataset that could accept numerous inputs and produce outputs using a deep learning method and employed a metric known as the loss to evaluate the performance of our model. The loss quantifies the error produced by the model: a high loss value usually implies that the model produces inaccurate output, whereas a low value suggests that the model produces less error.

The training loss measures how well the predictions of a deep learning model match the training data and is used to assess the performance of the model on the training set, i.e., the subset of the data used to train the model initially. The loss is broken down into two parts: the classification loss (for predicting conditional class probabilities) and the localization loss (for predicting bounding box offsets). In this case, our SSD Mobilenetv2 FPN Lite 320 × 320 deep learning model had a classification loss of 0.0413515, a localization loss of 0.01855353, a regularization loss of 0.07530993, and a total loss of 0.13521497 when evaluated. This loss value is very low, meaning that the model can classify and localize objects based on limited datasets.

To determine the ability of our model to produce object names, we assessed the classification accuracy for each type of object (cars, motorcycles, people, and goods rickshaws). A test of the classification accuracy before and after training of the model showed an increase in the classification accuracy for cars from 70 % to above 96 % for the morning, afternoon, and evening conditions. For

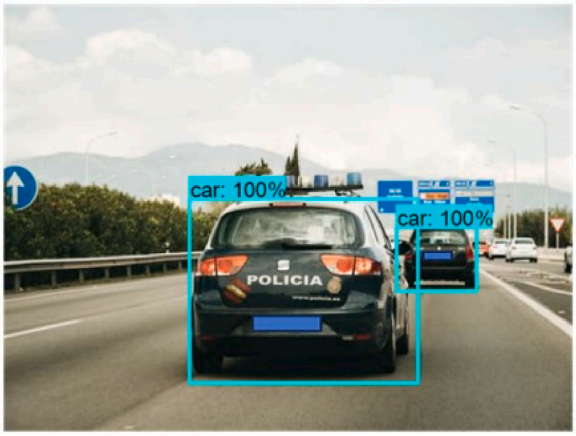


Fig. 13. Object classification experiment on images taken in Europe.

motorcycles, the accuracy increased from 49 % under afternoon lighting to 97 % after being trained. For people, the accuracy decreased from 77 % to 51 % after training, under morning lighting.

In the assessment of images acquired during the afternoon, the preliminary classification accuracy stood at 33 %, revealing a subsequent enhancement to 75 % following the completion of the training process. Similarly, for images captured in the evening, the initial accuracy was measured at 73 %, experiencing an appreciable augmentation to 97 % post-training. Notably, the classification accuracy values for goods rickshaws across the three lighting conditions (morning, afternoon, and evening) were initially unavailable due to the absence of specific training for this category within the model. However, after targeted training, the model exhibited a noteworthy proficiency, achieving a perfect 100 % classification accuracy for goods rickshaws. This underscores the adaptability and capacity of the model to generalize and accurately identify objects for which it was not initially trained.

A real-time road traffic management system was based on an upgraded YOLOv5 model. Experimental findings showed that the YOLOv5 algorithm has an overall accuracy of 78 % for cars and 77 % for people for mAP (0.5) [51]. YOLOv4 is the main structure to detect vehicles in tough weather like haze, sandstorms, snowstorm, fog, dust, and rain, during both day and night. With some improvements called CSPDarknet53 with SPP-NET added to YOLOv4, it can accurately identify cars with an 81 % overall accuracy [57]. The classification accuracy of the multi-scale hybrid attention module (MHAM-YOLO) algorithm on the task of car detection in an occluded environment is 97 %. It accurately determines whether the vehicle is occluded by another vehicle or by irrelevant background information for cars [58].

An algorithm that fuses LiDAR with machine vision sensors was based on an improved aggregate view object detection network (AVOD) scheme in a prior study. In this case, the AVOD fusion algorithm was taken as the benchmark framework. The results showed that the improved AVOD fusion algorithm improved the detection accuracy and speed compared with the original AVOD fusion algorithm in terms of car objects and pedestrians in urban road scenarios. This model had a classification accuracy of 78 % for cars and 77 % for people [56].

We compared the object detection capabilities of our system under three different lighting conditions (morning, afternoon, and evening), and images with varied object scales were considered to detect four classes of objects (cars, motorcycles, people, and goods rickshaws). Tables 6 and 7 show the metrics used to evaluate how well our CNN model performed the task of object detection. In the detection process, a location and a confidence rating are assigned to each object. Precision measures the accuracy of the positive detections made by the model, specifically the ratio of correctly identified objects to the total objects predicted. Recall measures the ability of the model to identify and capture all relevant objects within an image. It is the ratio of correctly identified objects to all actual objects.

The first real-time technique to attain mAP exceeding 70 % is SSD300. For a mAP of 74.3 %, the speed of inference is 59 FPS, which is very high. The frame rate for Faster R–CNN is only 7 FPS, although its mAP is extremely high. YOLOv1 has a frame rate of 45 FPS and a mAP of 63.4 %; despite the low resolution of the input images, it can still produce excellent results. Based on the results of experiments, the SSD300 model does not lose accuracy when it is sped up; however, it does have some drawbacks. For example, the default box needs to be explicitly set, and the detection performance for small objects is inferior to that of Faster RCNN. The disadvantages of SSD include the requirement to explicitly specify the minimum, maximum, and aspect ratio parameters for the previous box and its inability to detect small objects [42].

The YOLOv4 method is based on the original YOLO architecture, and employs the leading optimization technique that is currently available. It also includes enhancements in terms of data processing, network training, activation function, loss function, and other aspects. Despite a lack of theoretical novelty, many engineers value this approach. It has three main advantages: (i) it is a highly efficient and accurate object identification model that can be trained on a standard GPU for consumer use; (ii) it evaluates the effectiveness of several state-of-the-art target detector training techniques; and (iii) the target detection module offers an excellent level of performance by achieving a balance between FPS and precision. However, several variants of the YOLO series have drawbacks: for example, the detection of small targets is ineffective, each cell can only generate two boxes and one class, and the recall rate is low.

Despite the use of numerous techniques to improve performance, the detection of small objects has not improved substantially. The precision of YOLOv3 remains subpar compared to the R–CNN series. The deficiency arises from the practice of overwriting data initially populated at identical output layers and anchors with subsequently filled data. For instance, in scenarios where a dog and a cat of similar dimensions coexist in the same spatial location, the model may only identify one of the animals due to the data overwrite mechanism. In YOLOv4, a variety of techniques are employed to enhance the performance.

RetinaNet gives the same high accuracy as Faster R–CNN. Since the formula for the FL is flexible and can be expressed in a variety of ways while maintaining the same level of performance, it is not important how the FL is defined. In the domain of computer vision, a streamlined single-stage detector, RetinaNet, was devised to substantiate and address issues pertaining to the loss of focus in object detection. This model closely resembles a combination of Resnet and FPN. ResNet is adopted as the basis for feature extraction, while the purpose of FPN is to enhance the use of Resnet's multi-scale features in order to generate more evocative feature maps using multi-scale target region data. Although the FL considerably reduces the disparity between positive and negative samples, several aspects could be improved: for example, it is susceptible to noise interference, making accurate sample labeling extremely challenging.

CenterNet, a novel anchor-free object detection method based on the CornerNet foundation and published in April 2019, outperforms all other one-stage object detection algorithms on the MS COCO dataset by constructing object detection triples. In terms of AP, it outperforms the majority of extant one-stage and two-stage methods on the COCO dataset. Several enhancements have been made to the architecture of CenterNet, with positive results. However, CenterNet also has some disadvantages: for instance, in the training and prediction procedure, it can detect only a single center point, and classify two objects as a single object if the centers of the two objects overlap after downsampling [59].

A high recall value with a low precision means that although all of the ground truth objects can be discovered, the majority are inaccurate (many false positives). In contrast, a low recall value with high precision suggests that although all of the predicted boxes are correct, the majority of the ground truth objects are missed (many false positives and negatives). The mAP and mAR were measured for 10 different IoU thresholds in steps of 0.05 (0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9), which can also be expressed as mAP@[0.5,0.95]. The object size was measured in pixels, with large objects defined as those with more than 96 pixels, medium-sized ones as between 32 and 96 pixels, and small ones as below 32 pixels. The model seeks to strike a balance between minimizing false positives (improving precision) and capturing as many true positives as possible (improving recall). The optimization process involves considering various IoU values, reflecting different levels of overlap between predicted and ground truth locations. This approach aims to enhance the model's ability to accurately determine locations across a range of scenarios.

To optimize the model for high recall values and precise object position identification, we must consider the AR value. The AR for large objects in this study was 0.51, while the values were 0.348 for medium-sized objects and 0.293 for small objects. The average rating for all sizes was 0.411. The test results also demonstrate the capacity of the detection model to accurately recognize each object's location and name under different lighting conditions (morning, afternoon, and evening). The AP and AR values were used to calculate the performance of the model: larger objects generated the highest AP value of 0.374, while small objects generated the lowest value of 0.169; larger objects also had the highest AR value of 0.515, while small objects produced the lowest value of 0.293. The capacity of the model to detect large objects therefore exceeds its ability to detect small objects. A high recall value means that an object has been detected, but its name does not match the ground truth. A high recall value is extremely desirable when determining the location of an object.

It was discovered in this study that low values of the classification and localization losses affected the AP and AR. The capacity of the CNN model to recognize an object is therefore strongly influenced by the size of the object. After assessing the classification accuracy, it was discovered that large and medium-sized objects were detected more accurately than small ones. There was also an effect from the illumination of the images (captured in the morning, afternoon, and evening) on the classification accuracy following training. Another effect was the difference between the detection abilities before and after training, with a final classification accuracy of up to 100 %. A further discovery was that our CNN model could yield excellent classification accuracy even when trained on a small dataset. Finally, as indicated in Table 1, new objects can be trained on the SSD Mobilenetv2 FPN Lite 320 × 320 model using a dataset of 121 goods rickshaws.

The AP is computed differently for the VOC and COCO datasets. The VOC dataset includes 20 object categories, and the AP for each category is calculated using an interpolated 11-point sampling of the precision-recall curve. The final AP is computed by taking the mean of the APs across all 20 categories. In contrast, the Microsoft COCO dataset includes 80 object categories and uses a more

Table 9
Comparison of object detection models.

Model	AP	AP50
Faster RCNN [59]	36.8 %	57.7 %
SSD [59]	31.2 %	50.5 %
YOLOv4 [60]	43.5 %	65.7 %
YOLOv5x [60]	55.8 %	–
YOLOR [60]	55.4 %	73.3 %
YOLOX [60]	50.1 %	–
YOLOv6-L6 [60]	57.2 %	–
YOLOv7-E6 [60]	55.9 %	73.5 %
YOLOv8x [60]	53.9 %	–
Proposed model	–	**62.1 %**

complex method for calculating the AP; rather than using an 11-point interpolation, it uses a 101-point interpolation, i.e., it computes the precision for 101 recall thresholds from zero to one in increments of 0.01. The AP is also obtained by averaging over multiple IoU values rather than just one; the exception is a common AP metric called AP50, which is the AP for a single IoU threshold of 0.5.

We compared our object detection model with the Faster RCNN, SSD, and YOLO models, as shown in Table 9. The YOLO model was evaluated based on the MS COCO dataset test-dev 2017 on four training machines (NVIDIA V100, NVIDIA A100, Tesla V100, and Tesla T4). When applied to the MS COCO dataset test-dev 2017, YOLOv4 achieved an AP of 43.5 % and AP50 of 65.7 %, with a speed of more than 50 FPS on an NVIDIA V100. YOLOv5x achieved an AP of 50.7 % on images with a size of 640 pixels. Using a batch size of 32, it could achieve a speed of 200 FPS on an NVIDIA V100. With a larger input size of 1536 pixels and test-time augmentation (TTA), YOLOv5 achieved an AP of 55.8 %. YOLOR achieved an AP of 55.4 % and AP50 of 73.3 % with a speed of 30 FPS on an NVIDIA V100, whereas YOLOX achieved state-of-the-art results in 2021, with an optimal balance between speed and accuracy, with 50.1 % AP at a speed of 68.9 FPS on a Tesla V100.

YOLOv6 consists of eight scaled models, called YOLOv6-N to YOLOv6-L6. When evaluated on the MS COCO dataset test-dev 2017, the largest model achieved an AP of 57.2 % at a speed of around 29 FPS on an NVIDIA Tesla T4. YOLOv7-E6 achieved an AP of 55.9 % and AP50 of 73.5 % with an input size of 1280 pixels, at a speed of 50 FPS, on an NVIDIA V100. YOLOv8x achieved an AP of 53.9 % with an image size of 640 pixels (compared to 50.7 % for YOLOv5 with the same input size) with a speed of 280 FPS on an NVIDIA A100 and TensorRT. As shown in Table 9, the 62.1 % AP50 of the overall classes can be achieved on the three lighting conditions (morning, afternoon, and evening). For AP50 values, our presented framework is 4.4 % and 11.6 % higher than those achieved by Faster RCNN and SSD, while our presented framework is 3.6 %, 11.2 %, and 11.4 % lower than those achieved by YOLOv4, YOLOR, and YOLOv7-E6 respectively.

The SSD Mobilenetv2 FPN Lite 320 $\times$ 320 CNN model proposed in this paper has limitations in terms of detecting objects under nighttime, rain, fog, and snowy conditions. In addition, the model can only detect cars, motorcycles, people, and goods rickshaws. The classification accuracy for small objects is low, although a moderate degree of classification precision was seen for images of people standing upright.

In further work, we will consider a dataset containing images taken under nighttime and inclement conditions and will employ an augmentation process with two variations, namely mosaic augmentation and geometric transformation augmentation, which will include image displacement, random scaling, motion blurring, and the addition of random noise.

The proposed CNN model is especially suitable for use with AMRs in environments that require the detection of goods rickshaws as well as in general scenarios with cars, motorcycles, and people, to avoid collisions. The possible applications for this CNN model include object detection for AMRs as well as integration with path planning modules and road detection modules, which will allow robots to make decisions when turning and overtaking other objects on the road.

5. Conclusion

This paper has proposed a new CNN-based object detection system based on images of real-world vehicles, for use with AMRs. When developing the proposed system, we employed the concept of transfer learning by fine-tuning a CNN model that had been trained previously. First, we created an original dataset of vehicle images taken in Banda Aceh. An SSD Mobilenetv2 FPN Lite 320 $\times$ 320 model was then developed, which was fine-tuned to detect four classes of objects: cars, motorcycles, people, and goods rickshaws. Our results demonstrated that the CNN model could determine the position of the object and its name, with good classification accuracy. Afternoon lighting conditions provided the best detection capacity. The trained CNN model could identify cars, motorcycles, people, and goods rickshaws with medium to high accuracy (50–100 %), even when applied to images of occluded and truncated objects. The classification accuracy of the CNN model in identifying cars, motorbikes, and goods rickshaws approached 100 % after training and reached 97 % for people. Training the CNN model boosted its classification accuracy and reduced errors in determining object names and locations. The results presented here indicate that the proposed CNN-based object detection system shows promise for future implementation in AMRs.

Funding statement

This work was supported by Doctoral dissertation research, the Ministry of Education, Culture, Research, and Technology of the Republic of Indonesia, under Grant No. 67/UN11.2.1/PT.01.03/DPRM/2022.

Data availability statement

Data will be made available on request.

Additional information

No additional information is available for this paper.

CRediT authorship contribution statement

Udink Aulia: Writing – original draft, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization.

Iskandar Hasanuddin: Validation, Resources, Formal analysis, Data curation. **Muhammad Dirhamsyah:** Visualization, Validation, Supervision, Resources, Project administration. **Nasaruddin Nasaruddin:** Writing – review & editing, Validation, Supervision, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] G. Fragapane, R. de Koster, F. Sgarbossa, J.O. Strandhagen, Planning and control of autonomous mobile robots for intralogistics: literature review and research agenda, Eur. J. Oper. Res. 294 (2) (2021) 405–426, https://doi.org/10.1016/j.ejor.2021.01.019.

[2] T.A.Q. Tawiah, A review of algorithms and techniques for image-based recognition and inference in mobile robotic systems, Int. J. Adv. Rob. Syst. 17 (6) (2020) 1–25, https://doi.org/10.1177/1729881420972278.

[3] M.B. Alatise, G.P. Hancke, A review on challenges of autonomous mobile robot and sensor fusion methods, IEEE Access 8 (2020) 39830–39846, https://doi.org/10.1109/ACCESS.2020.2975643.

[4] D. Topolsky, et al., Development of a mobile robot for mine exploration, Processes 10 (5) (2022) 1–17, https://doi.org/10.3390/pr10050865.

[5] F. Rubio, F. Valero, C. Llopis-Albert, A review of mobile robots: concepts, methods, theoretical framework, and applications, Int. J. Adv. Rob. Syst. 16 (2) (2019) 1–22, https://doi.org/10.1177/1729881419839596.

[6] J.A. Gonzalez-Aguirre, et al., Service robots: trends and technology, Appl. Sci. 11 (22) (2021), https://doi.org/10.3390/app112210702.

[7] A. Bonci, P.D.C. Cheng, M. Indri, G. Nabissi, F. Sibona, Human-robot perception in industrial environments: a survey, Sensors 21 (5) (2021) 1–29, https://doi.org/10.3390/s21051571.

[8] C. Faria, A. Rocha, Human – robot interaction in industrial settings : perception of different courtesy cues, Robotics 11 (59) (2022), https://doi.org/10.3390/robotics11030059.

[9] P. Stączek, J. Pizoń, W. Danilczuk, A. Gola, A digital twin approach for the improvement of an autonomous mobile robots (AMR's) operating environment—a case study, Sensors 21 (23) (2021), https://doi.org/10.3390/s21237830.

[10] F. Gul, W. Rahiman, S.S. Nazli Alhady, A comprehensive study for robot navigation techniques, Cogent Eng. 6 (1) (2019), https://doi.org/10.1080/23311916.2019.1632046.

[11] C. Newman, J. Petzing, Y.M. Goh, L. Justham, Investigating the optimisation of real-world and synthetic object detection training datasets through the consideration of environmental and simulation factors, Intell. Syst. with Appl. 14 (2022), https://doi.org/10.1016/j.iswa.2022.200079.

[12] T.Y. Lin, et al., Microsoft COCO: common objects in context, Lect. Notes Comput. Sci. 8693 LNCS (PART 5) (2014) 740–755, https://doi.org/10.1007/978-3-319-10602-1_48.

[13] M. Everingham, V-G L, W C K I, W J, Z A, The pascal visual object classes (VOC) challenge, Int. J. Comput. Vis. 88 (2) (2015) 303–338.

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, ImageNet: a large-scale hierarchical image databas, 2009 IEEE Conf. Comput. Vis. Pattern Recognit. 20 (11) (2009) 1221–1227.

[15] M. Cordts, et al., The cityscapes dataset for semantic urban scene understanding, IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recogn. 2016-Decem (2016) 3213–3223, https://doi.org/10.1109/CVPR.2016.350.

[16] G.J. Brostow, J. Fauqueur, R. Cipolla, Semantic object classes in video: a high-definition ground truth database, Pattern Recogn. Lett. 30 (2) (2009) 88–97, https://doi.org/10.1016/j.patrec.2008.04.005.

[17] D.H. Dos Reis, D. Welfer, M.A. De Souza Leite Cuadros, D.F.T. Gamarra, Mobile robot navigation using an object recognition software with RGBD images and the YOLO algorithm, Appl. Artif. Intell. 33 (14) (2019) 1290–1305, https://doi.org/10.1080/08839514.2019.1684778.

[18] T. Toda, G. Masuyama, K. Umeda, Moving object detection using a stereo camera mounted on a moving platform, SICE J. Control. Meas. Syst. Integr. 10 (5) (2017) 344–349, https://doi.org/10.9746/jcmsi.10.344.

[19] E.T. Baek, D.Y. Im, ROS-based unmanned mobile robot platform for agriculture, Appl. Sci. 12 (9) (2022), https://doi.org/10.3390/app12094335.

[20] X. Gao, J. Xu, C. Luo, J. Zhou, P. Huang, J. Deng, Detection of lower body for AGV based on SSD algorithm with ResNet, Sensors 22 (5) (2022), https://doi.org/10.3390/s22052008.

[21] J. Cao, et al., Front vehicle detection algorithm for smart car based on improved SSD model, Sensors 20 (16) (2020) 1–21, https://doi.org/10.3390/s20164646.

[22] Á. Morera, Á. Sánchez, A.B. Moreno, Á.D. Sappa, J.F. Vélez, Ssd vs. Yolo for detection of outdoor urban advertising panels under multiple variabilities, Sensors 20 (16) (2020) 1–23, https://doi.org/10.3390/s20164587.

[23] U.-C. Hwang, Y-J, J.-G. Lee, Moon and Park, Squeeze and extraction feature fusion, Sensors 20 (13) (2020), https://doi.org/10.3390/s20133630.

[24] S. Nilwong, D. Hossain, S.I. Kaneko, G. Capi, Deep learning-based landmark detection for mobile robot outdoor localization, Machines 7 (2) (2019), https://doi.org/10.3390/machines7020025.

[25] L. Barba-Guaman, J.E. Naranjo, A. Ortiz, Deep learning framework for vehicle and pedestrian detection in rural roads on an embedded GPU, Electron 9 (4) (2020) 1–17, https://doi.org/10.3390/electronics9040589.

[26] J. Kaur, W. Singh, Tools, techniques, datasets and application areas for object detection in an image: a review, Multimed. Tool. Appl. (2022), https://doi.org/10.1007/s11042-022-13153-y.

[27] M. Shen, Y. Wang, Y. Jiang, H. Ji, B. Wang, Z. Huang, A new positioning method based on multiple ultrasonic sensors for autonomous mobile robot, Sensors 20 (1) (2020), https://doi.org/10.3390/s20010017.

[28] M. Balogh, et al., Cloud-controlled autonomous mobile robot platform, IEEE Int. Symp. Pers. Indoor Mob. Radio Commun. PIMRC 2021-Septe (March 2022), https://doi.org/10.1109/PIMRC50174.2021.9569730, 2021.

[29] M. Aladem, S.A. Rawashdeh, Lightweight visual odometry for autonomous mobile robots, Sensors 18 (9) (2018) 1–14, https://doi.org/10.3390/s18092837.

[30] J. Kim, J. Park, W. Chung, Self-diagnosis of localization status for autonomous mobile robots, Sensors 18 (9) (2018), https://doi.org/10.3390/s18093168.

[31] Y. Kobayashi, et al., Robot navigation based on predicting of human interaction and its reproducible evaluation in a densely crowded environment, Int. J. Soc. Robot. 14 (2) (2022) 373–387, https://doi.org/10.1007/s12369-021-00791-9.

[32] F. Khan, S. Hussain, S. Basak, M. Moustafa, P. Corcoran, A review of benchmark datasets and training loss functions in neural depth estimation, IEEE Access 9 (2021) 148479–148503, https://doi.org/10.1109/ACCESS.2021.3124978.

[33] M.U. Hossain, M.A. Rahman, M.M. Islam, A. Akhter, M.A. Uddin, B.K. Paul, Automatic driver distraction detection using deep convolutional neural networks, Intell. Syst. with Appl. 14 (2022) 200075, https://doi.org/10.1016/j.iswa.2022.200075.

[34] M. Hussain, T. Chen, R. Hill, Moving toward Smart Manufacturing with an Autonomous Pallet Racking Inspection System Based on MobileNetV2, 2022.

[35] K. Alibabaei, E. Assunção, P.D. Gaspar, V.N.G.J. Soares, J.M.L.P. Caldeira, Real-Time Detection of Vine Trunk for Robot Localization Using Deep Learning Models Developed for Edge TPU Devices, 2022, pp. 1–16.

[36] F. Foroughi, Z. Chen, J. Wang, A CNN-based system for mobile robot navigation in indoor environments via visual localization with a small dataset, World Electr. Veh. J. 12 (3) (2021), https://doi.org/10.3390/wevj12030134.

[37] Y. Guo, Y. Xie, Y. Chen, X. Ban, B. Sadoun, M.S. Obaidat, An efficient object navigation strategy for mobile robots based on semantic information, Electron. 11 (7) (2022), https://doi.org/10.3390/electronics11071136.

[38] S.S. Esfahlani, The deep convolutional neural network role in the autonomous navigation of mobile robots (SROBO), Rem. Sens. 14 (14) (2022), https://doi.org/10.3390/rs14143324.
[39] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: towards real-time object detection with region proposal networks, IEEE Trans. Pattern Anal. Mach. Intell. 39 (6) (Jun. 2015) 1137–1149, https://doi.org/10.1109/TPAMI.2016.2577031.
[40] J. Dai, Y. Li, K. He, J. Sun, R-FCN: object detection via region-based fully convolutional networks, Adv. Neural Inf. Process. Syst. (2016), https://doi.org/10.48550/arXiv.1605.06409.
[41] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recogn. (2014) 580–587, https://doi.org/10.1109/CVPR.2014.81.
[42] W. et al Liu, SSD: Single Shot MultiBox Detector Wei 794 (2015) 185–192 [Online]. Available: http://arxiv.org/abs/1512.02325.
[43] R. Girshick, J. Donahue, T. Darrell, J. Malik, U.C. Berkeley, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recogn. 1 (2014) 5000, https://doi.org/10.1109/CVPR.2014.81.
[44] A.G. Howard, et al.. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, 2017, pp. 1–9 [Online]. Available: http://arxiv.org/abs/1704.04861.
[45] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.C. Chen, MobileNetV2: inverted residuals and linear bottlenecks, IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recogn. (2018) 4510–4520, https://doi.org/10.1109/CVPR.2018.00474.
[46] T.Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollar, Focal loss for dense object detection, IEEE Trans. Pattern Anal. Mach. Intell. 42 (2) (2020) 318–327, https://doi.org/10.1109/TPAMI.2018.2858826.
[47] R. Girshick, Fast R-CNN, Proc. IEEE Int. Conf. Comput. Vis. 2015 (2015) 1440–1448, https://doi.org/10.1109/ICCV.2015.169. Inter.
[48] R. Padilla, W.L. Passos, T.L.B. Dias, S.L. Netto, E.A.B. Da Silva, A comparative analysis of object detection metrics with a companion open-source toolkit, Electron 10 (3) (2021) 1–28, https://doi.org/10.3390/electronics10030279.
[49] I. Rasmussen, S. Kvalsvik, P.A. Andersen, T.N. Aune, D. Hagen, Development of a novel object detection system based on synthetic data generated from unreal game engine, Appl. Sci. 12 (17) (2022), https://doi.org/10.3390/app12178534.
[50] S. Zhao, C. Wang, P. Wei, Q. Zhao, Research on the deep recognition of urban road vehicle flow based on deep learning, Sustain. Times 12 (17) (2020), https://doi.org/10.3390/su12177094.
[51] T. Sharma, B. Debaque, N. Duclos, A. Chehri, B. Kinder, P. Fortier, Deep learning-based object detection and scene perception under bad weather conditions, Electron 11 (4) (2022) 1–11, https://doi.org/10.3390/electronics11040563.
[52] C. Neelam Jaikishore, et al., Implementation of deep learning algorithm on a custom dataset for advanced driver assistance systems applications, Appl. Sci. 12 (18) (2022), https://doi.org/10.3390/app12188927.
[53] M. Carranza-García, J. Torres-Mateo, P. Lara-Benítez, J. García-Gutiérrez, On the performance of one-stage and two-stage object detectors in autonomous vehicles using camera data, Remote Sens 13 (1) (2021) 1–23, https://doi.org/10.3390/rs13010089.
[54] F. Zhang, C. Li, F. Yang, Vehicle detection in urban traffic surveillance images based on convolutional neural networks with feature concatenation, Sensors 19 (3) (2019), https://doi.org/10.3390/s19030594.
[55] Y. Li, J. Wang, J. Huang, Y. Li, Research on deep learning automatic vehicle recognition algorithm based on RES-YOLO model, Sensors 22 (10) (2022), https://doi.org/10.3390/s22103783.
[56] Z. Bai, D. Bi, J. Wu, M. Wang, Q. Zheng, L. Chen, Intelligent driving vehicle object detection based on improved AVOD algorithm for the fusion of LiDAR and visual information, Actuators 11 (10) (2022) 272, https://doi.org/10.3390/act11100272.
[57] M. Humayun, F. Ashfaq, N.Z. Jhanjhi, M.K. Alsadun, Traffic management: multi-scale vehicle detection in varying weather conditions using YOLOv4 and spatial pyramid pooling network, Electron 11 (17) (2022), https://doi.org/10.3390/electronics11172748.
[58] T. Deng, X. Liu, L. Wang, Occluded vehicle detection via multi-scale hybrid attention mechanism in the road scene, Electron. 11 (17) (2022), https://doi.org/10.3390/electronics11172709.
[59] K.S. Chahal, K. Dey, A Survey of Modern Object Detection Literature using Deep Learning, 2018, pp. 1–15 [Online]. Available: http://arxiv.org/abs/1808.07256.
[60] J. Terven, D. Cordova-Esparza, A Comprehensive Review of YOLO: From YOLOv1 to YOLOv8 and Beyond (2023) 1–33 [Online]. Available: http://arxiv.org/abs/2304.00501.