



Advancing Vehicle Logo Detection with DETR to Handle Small Logos and Low-Quality Images

Rifky Fahrizal Ubaidillah¹, Mahmud Dwi Sulistiyo^{2*}, Gamma Kosala³, Ema Rachmawati⁴, Deny Haryadi⁵
^{1,2,3,4,5}School of Computing, Telkom University, Bandung, Indonesia

¹rifkyfahrizal@student.telkomuniversity.ac.id, ²mahmuddwis@telkomuniversity.ac.id,

³gammakosala@telkomuniversity.ac.id, ⁴emarachmawati@telkomuniversity.ac.id, ⁵denyharyadi@telkomuniversity.ac.id

Abstract

Image-based vehicle logo detection is an important component in the implementation of vehicle information recognition technology, which supports the development of intelligent transportation systems. Vehicle logos, as elements that represent the identities of vehicle brands and models, play a significant role in completing vehicle identity data. The information obtained from this logo can be utilized to solve various traffic problems, such as vehicle document counterfeiting and theft, and for better traffic planning and management purposes. However, the main challenge in developing an accurate logo detection system lies in the wide variety of shapes, sizes, and positions of logos in different types of vehicles. In addition, the generally small size of logos, especially on certain vehicles, often makes it difficult for computer-based detection systems to recognize logos consistently, thus affecting the overall performance of the detection model. In this research, the Detection Transformers (DETR) method is used to build a vehicle logo detection system that focuses on small-scale logo. The testing process was conducted using the VL-10 dataset, which was specifically designed for vehicle logo detection evaluation. The results show that the DETR model can detect vehicle logos very well, even for small-scale logos. The model achieved an AP50 value of 0.952, which indicates a high level of accuracy and reliability in detecting the vehicle logo in the dataset used.

Keywords: detection transformers; logo; object detection; vehicle

How to Cite: R. F. Ubaidillah, M. D. Sulistiyo, G. Kosala, E. Rachmawati, and D. Haryadi, "Advancing Vehicle Logo Detection with DETR to Handle Small Logos and Low-Quality Images", J. RESTI (Rekayasa Sist. Teknol. Inf.), vol. 9, no. 4, pp. 796 - 804, Aug. 2025.

Permalink/DOI: <https://doi.org/10.29207/resti.v9i4.6236>

Received: December 13, 2024

Accepted: July 22, 2025

Available Online: August 17, 2025

*This is an open-access article under the CC BY 4.0 License
Published by Ikatan Ahli Informatika Indonesia*

1. Introduction

A logo is a visual identity that represents the values, philosophy, and image of a company [1]. In an era where information technology and the automotive industry are developing quickly, intelligent systems in transportation are becoming a valuable aspect of the utilization of systems in various fields. Many companies are interested in exploring logo detection as a strategic area, to support decisions related to brand development and attracting customer attention [1], [2]. In order to evaluate how well their brand is accepted by the public, industry players and marketing agencies require precise market data and to measure the branding done in a certain area. By enhancing system/user capabilities, it is useful to identify an item's identity through its logo [3]. In vehicles, especially cars, the position of the brand logo is usually located in the middle of the car body and between the two headlights.

Each logo on the car has its own characteristics, so people can know a brand just by looking at the logo. The shape of each logo is quite complicated and diverse, also has a very small size [4].

The image captured by the camera contains vehicle information, which we can see through the license plate and also the vehicle logo. As vehicle information, logo can help in complete vehicle identity and dealing with traffic issues such as counterfeiting [5], theft [6], and cross-border cases [7]. This issue is one of the detection system implementations conducted by government in traffic law enforcement. The system, which is carried out automatically, can be a solution to the problems encountered, for instance by detecting various types of traffic violations with high accuracy, no violations are missed, and lower costs are spent [8].

In detecting vehicle logo, of course, there are some problems that will occur that may cause the resulting

accuracy to be not optimal. There are several factors including blurry images, location/position, size, viewing angle, weather, and lighting. These challenges arise from the diversity and complexity of real-world scenarios, where conditions are often far from ideal. There are several research focuses that have focused on addressing these problems, such as small objects [2], complex scenes [9], and so on.

In previous research, the majority of the models used YOLO and CNN which were quite effective. YOLO accomplish this by handling object detection as a single stage paradigm, predicting bounding boxes and class probabilities directly from full images in one evaluation [10]. Additionally, they are highly efficient because simultaneously forecast bounding boxes and class probabilities while dividing the image into a grid [11]. On the other hand, by allowing models to automatically learn hierarchical feature representations from data, CNNs improved the detection of objects. The ability of CNN-based models, such as Faster R-CNN, to learn hierarchical representations has allowed them to outperform conventional techniques in object localization and feature extraction, particularly in complex environments with varying object scales and occlusions [11]. Some research has been done by combining the strengths of YOLO and CNNs such as crowd object detection [11], fire and smoke detection [12], potato disease identification [13], and many more. These results demonstrate its adaptability and effectiveness in a wide array of real-world scenarios.

This trend highlights the potential of leveraging both speed and accuracy for specific applications. For instance, research [2] proposed a method of detecting vehicle logo using convolutional neural networks (CNN). The authors introduce a lightweight network structure or framework with separable convolutions to improve real time performance and enable implementation on embedded systems. To improve small-scale objects, the authors change the training policy. The authors discuss the challenges in vehicle logo detection and evaluate existing detectors focusing on small objects. This study resulted in a mAP of 88.1%. The experimental results show that their model effectively improves the vehicle logo detection accuracy and its ability to detect small-scale objects.

In research [9], a new approach introduced to quickly and precisely detect vehicle logo under complex circumstances. This approach involves modifying the YOLOv3 model and solving the challenge of detecting small objects. In addition, a new dataset, VLD-30, was created to enhance the training method and improve accuracy consisting of 2010 images with 30 categories. From this research, the maximum accuracy result is 98.9% outperforming other models such as Faster-RCNN, SSD-512, and YOLOv2. The experimental results show that this approach is effective for fast and accurate vehicle logo detection under complex circumstances.

Research [14] discussed about pedestrian detection in the middle of a crowd of people. The authors compared the performance of end-to-end detectors with traditional detectors such as Faster-RCNN with FPN. Additionally, this study also proposes new decoders and mechanisms to improve the performance of these detectors. Furthermore, it describes the challenges of detecting pedestrians in crowded scene and overlapping images. The results of the author's research resulted in an average precision score of 89.54% and recall of 94.00. From these results, DETR outperforms other models in terms of detection.

In research [15], vehicle logo detection was tested in various traffic situations. The author uses a custom dataset VL-10 as a test material for testing. Then the system used is YOLOv5 and improvises on IWPOD-NET. According to research findings, the system showed an improvement in each test case. The system is able to detect logo within an angle range 0-30° and achieved mAP of 79%.

A modified DETR architecture has been utilized in several preliminary studies to investigate real-time vehicle logo detection [16], however these efforts mostly concentrate on enhancing the baseline RT-DETR model with improvements such as ResNet-34, SENetV2, and Context Guided methods. Experimental obtained a mAP@50:95 of 0.698, indicating a 1.5% improvement over the baseline RT-DETR model. These studies, however, frequently rely on specialized datasets such as VLD-Micro and do not adequately address the more general difficulties involved in identifying small-scale vehicle logos in a variety of real-world situations. There are several object detection studies that conduct experiments related to small objects [16] - [18], and others. Research in this area is still limited, especially when facing the challenge of detecting small-scale objects, which are often small and located in areas that are difficult to be detected.

DETR [19] offers a distinct advantage over traditional CNN-based models like YOLO and Faster R-CNN by leveraging self-attention mechanisms to capture global dependencies. Unlike Faster R-CNN, which relies on region proposal networks, DETR directly predicts object locations and classes in an end-to-end manner, eliminating hand-crafted components. While YOLO excels in real-time detection, it struggles with small objects due to its reliance on grid-based detection and anchor boxes, which can lead to poor localization accuracy. DETR, on the other hand, avoids these limitations by using a bipartite matching strategy, which is effective to solve small logo detection issues.

The key contributions of this paper include the implementation of DETR for small logo detection, a comparative analysis with traditional models like YOLO and Faster R-CNN, and an exploration of the impact of hyperparameters, particularly AGB values, on detection accuracy and computational efficiency. This research provides valuable insights into

Transformer-based detection techniques and their application to real-world logo detection challenges.

2. Methods

Figure 1 shows the process chain to build the model in this study. The process begins with the dataset collection phase from Kaggle. Since the dataset annotations are originally in YOLO format, they need to be converted into the COCO annotation format to be compatible with the DETR model. Next, the DETR model is trained using the prepared dataset, with necessary parameter tuning performed to optimize performance. After training, the model performance evaluation phase is conducted to assess the effectiveness of the trained DETR model. Various evaluation metrics are analyzed to determine the model's performance. The process concludes with a fully trained and evaluated DETR model, ready for deployment or further improvements.

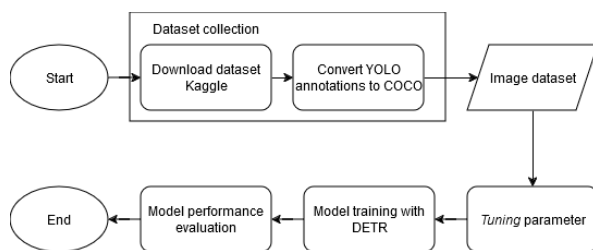


Figure 1. Flowchart of the modelling process.

2.1 Dataset

In this research, the dataset used is secondary data, specifically “Vehicle Logo (VL-10)” [20] which contains images of 10 vehicle brands. The dataset is sourced from Kaggle contains training data and validation data. The images in the dataset are captured from various locations and the types of vehicles are quite varied including city cars, MPVs, SUVs, and heavy vehicles. This variety ensures that the dataset is representative of real-world scenarios, making it suitable for developing robust and scalable detection models. The total dataset consists of 15,000 training images, 1,000 validation images, and 500 test images.

The detection process is further complicated by the dataset's includes process of images of varying resolutions. This variation simulates the difficulties encountered in real world applications, such as traffic surveillance or roadside monitoring, by requiring the detection model to handle both high resolution and low-resolution images efficiently. An example image of the dataset used can be seen in Figure 2.

2.2 Detection Transformers (DETR)

Detection Transformer is a deep learning object detection model developed by Facebook Research Team in 2020 that utilizes Transformer architecture. The DETR model architecture consists of a CNN backbone, an encoder-decoder transformer, and a feed-forward network (FFN). Each of these components

have their own tasks and functions. The first is CNN, which serve as a feature extractor, producing a set of features and activation maps derived from input. Next, the extracted features are processed by a transformer, consisting of an encoder-decoder that outputs a set of predictions. The process performed by DETR for object detection can be seen in Figure 3.



Original image



Blurry image



Noisy image

Figure 2. Examples of the vehicle image from the dataset with different conditions: normal, blurry, and noisy.

Each encoder-decoder layer has a standard architecture consisting of multi-head self-attention mechanism and FFN modules. In this process, each object query searches for a specific object in the image and predict the coordinates of the bounding box. FFN plays a role to improve the model's ability to learn patterns and relationships between objects in the image. The results

of the decoder are then passed through into prediction heads, which are used for calculations using bipartite matching loss. The illustration of DETR architecture can be seen in Figure 4.

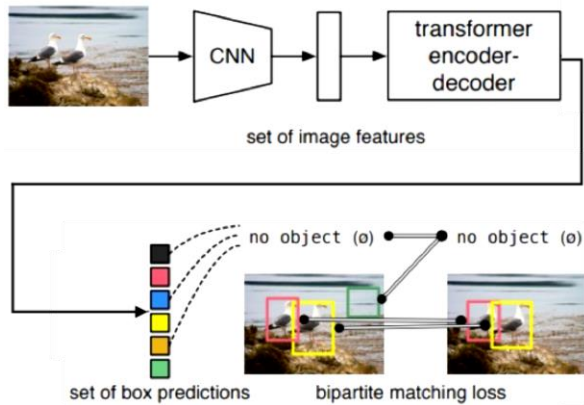


Figure 3. Illustration of the object detection using DETR [19].

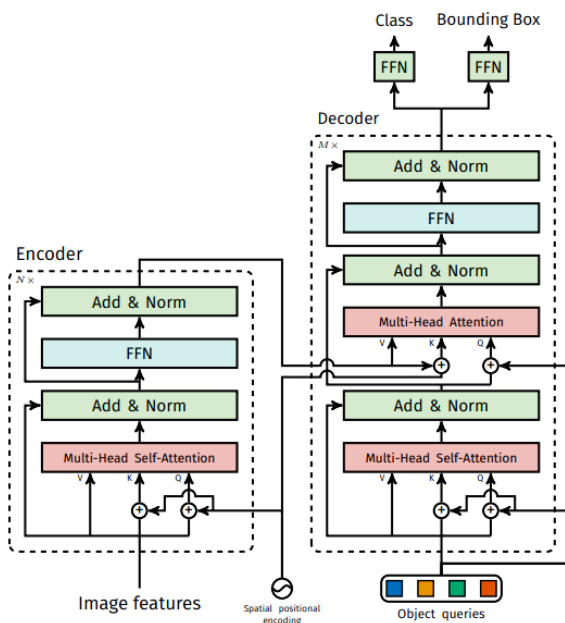


Figure 4. DETR model architecture [19].

2.3. Tuning Parameter

This step aims to optimize the performance of the model to achieve accurate and efficient detection results. There are two techniques implemented; Gradient clip validation (GCV) is used to limit the magnitude of the gradient during training to prevent explosive growth. This help avoid training instability, enabling the model to learn more effectively and reducing the risk of it getting stuck at a certain value. The model can fail and not converge if the gradient value is too large, which occurs when network weights are updated excessively. Implementing this method can speed up convergence during the training process [21]. In this test, the type of gradient clipping used is norm clipping, which limit the norm of the gradient vector to a specified threshold. The algorithm of gradient clip by norm is presented in Equation 1.

$$\begin{aligned} & \text{if } \|g\| > v \\ & g \leftarrow \frac{gv}{\|g\|} \end{aligned} \quad (1)$$

v is the hyperparameter, g is the gradient, and $\|g\|$ is the norm of g [22].

Accumulation gradient batches (AGB) is a process that virtually increases the batch size during the training process, which is useful when the available GPU memory is insufficient to accommodate the required batch size [23]. This method reduces the number of parameter updates performed during training while enabling effective utilization of batch size. This process allows for batch upgrades without memory loss by processing micro-batches sequentially [24]. After the micro batch process completes backward propagation, the system does not immediately update the model, but instead accumulates the micro batches to obtain the global batch gradient [25]. In this method, before performing the backward pass, the accumulated gradient is run on K small batches of size N . This has the effect of obtaining a batch size of size $K \times N$, where N is the previous batch size [26].

2.4 Evaluation

IoU is one of the most common methods for comparing similarities between data samples in machine learning [27]. This method is a metric used to measure the area of overlap in the bounding box. The bounding box is a marker that surrounds the object. The measurement is done by comparing between the predicted bounding box and the ground truth.

IoU is used for measuring how accurate the bounding box prediction is in object detection. We can identify the detection as true or false by comparing the IoU at a certain threshold [28]. The following is the formula for calculating IoU in Equation 2.

$$IoU = \frac{\text{area}(B_p \cap B_{gt})}{\text{area}(B_p \cup B_{gt})} \quad (2)$$

In Equation 2, B_p is the prediction bounding box and B_{gt} is the ground truth bounding box. The larger the IoU value, the higher the accuracy of the model in detection. In other words, IoU is used as a metric that differentiates between true and false objects. In this research evaluation, we use thresholds of 0.5, 0.75, and 0.95 to measure the accuracy of object detection.

Then implement Average Precision (AP) and Average Recall (AR) as another evaluation. These methods are metrics used to measure the accuracy of the model in performing the prediction process. Both AP and AR are commonly used in evaluating model performance in the field of object detection. Precision calculates how accurate the model is in detecting the correct objects, while recall calculates the number of objects that actually exist that can be detected by the model. In other words, recall indicates how completely the model can identify objects that should be detected as positive. The

following is the calculation of precision in Equation 3 and recall in Equation 4.

$$Precision = \frac{TP}{TP+FN} \quad (3)$$

$$Recall = \frac{TP}{TP+FP} \quad (4)$$

True Positive (TP) is defined as when the model successfully recognizes the object and it is detected correctly; True Negative (TN) is defined as when the model does not recognize the object but it is considered correct; False Positive (FP) is defined as when the model incorrectly detects an object that does not exist but is recognized by the model; and False Negative (FN) is defined as when the model cannot find an object that actually exists.

AP in the model is used as a performance indicator for object detection. It measures the area under the precision and recall curve (PR curve) [28]. The PR curve is a graph that shows a plot of values in precision and recall at a threshold. AP summarizes the PR curve into one scalar value. The larger the AP value, the better the performance of the object detection system.

The PR curve visualizes the performance level of the object detection system in maintaining a balance between precision and recall at a certain threshold. The axis of the PR curve is made with recall on the x -axis and precision on the y -axis, calculated at various thresholds, which shows how the precision and recall values change when the threshold changes. The following is the calculation in Equation 5.

$$AP = \frac{1}{11} \sum_{(0,0.1,0.2,\dots,0.9,1)} P_{interp}(r) \quad (5)$$

$$P_{interp} = \frac{\max}{\tilde{r}: \tilde{r} \geq r} p(\tilde{r})$$

In calculating AP, an 11-point interpolation technique approach is used to overcome small variations in precision. The above equation explains that, $p(\tilde{r})$ is the precision at recall point r . Then $P_{interp}(r)$ is the AP value on the interpolated PR curve.

In evaluating objects using the `cocoevaluator`, there are three size categories used, namely (according to the COCO usage guidelines [29]): small (less than 32^2), medium (between 32^2 and 96^2), and large (more than 96^2).

This study aims to evaluate the performance of the object detection model on vehicle logo using two evaluation categories, namely “all” and “small”. The “small” category is used to test the model's performance in handling small objects, which is a challenge in object detection tasks. By comparing the evaluation results in these two categories, it is hoped that a better understanding of the strengths and weaknesses of the model in detecting small objects in vehicle logo can be achieved.

3. Results and Discussions

3.1 Training Configuration

In this study, the dataset used is a secondary dataset obtained from Kaggle. The image data used is split into 91% training data, 6% validation data, and 3% test data. There is an adjustment of the label which previously was in YOLO format and then will now be converted to COCO in order to evaluate using `cocoevaluator`. Then the DETR model training was implemented from the Hugging Face library and PyTorch. The training scheme uses two ways, namely local and Kaggle Notebook. The local device used is the NVIDIA RTX 4060 Laptop GPU, then for the Kaggle Notebook device is the P100 GPU. In configuring the DETR model, several parameters are needed for support during training. The parameters used in this training are shown in Table 1.

Table 1. Training Parameter

Parameter	Value
Backbone	ResNet-50
Learning rate transformer	0.0001
Learning rate backbone	0.00001
Weight decay	0.0001
Optimizer	AdamW
Epoch	25
Batch Size	4

ResNet-50 was chosen as the backbone due to its proven effectiveness in image recognition and object detection tasks, as demonstrated by its widespread use in object detection models [30], [31]. Following the methodology of the original DETR model research, the initial values for the learning rates (transformer and backbone) and weight decay were adopted for this study [19]. In order to improve generalization and convergence when training deep neural networks, the AdamW optimizer is utilized due to its variable learning rate capabilities and decoupled weight decay [32]. The decision to train for 25 epochs with a batch size of 4 was influenced by considerations, such as the dataset size and available computational resources.

There are two parameters that will be tested, with the aim of identifying the optimal values, namely gradient clip validation and accumulate gradient batches. Both parameters have a total of four values, so this study will get a total of 16 trials. Each value will be tested from various aspects, such as log, AR, and AP. The parameters to be tested are shown in Table 2.

Table 2. Tuning Parameter

Gradient Clip Val	Accumulate Gradient Batches
0.05	4
0.1	8
0.3	12
0.5	16

3.2 Performance Analysis Based on IoUs

This section discusses the evaluation of the DETR model that has been trained for vehicle logo detection. For both AP and AR, the evaluation is based on a specific IoU with thresholds of 0.5, 0.75, and 0.95. The

tests are used to determine the optimal IoU value and measure the performance generated by each model. Shown below is one of the test configurations that have been trained. The evaluation results on the experiments that have been conducted are shown in Table 3.

Table 3. IoU Evaluation Results

Evaluation Method	GCV	AGB	@IoU 0.5	@IoU 0.75	@IoU 0.95
AP	0.05	4	0.955	0.459	0.000
AR	0.05	4	0.965	0.602	0.002

Based on the results in Table 3, the best AP value obtained at IoU 0.5 is 0.955 and AR at IoU 0.5 with a value of 0.966. The evaluation results show that IoU 0.5 is better when compared to other IoUs. According to the results obtained, in the future testing, only IoU 0.5 parameters are used in further evaluation.

3.3 Evaluation Results

In the training that has been conducted, several outputs were generated for identification, including models and logs. The model will be evaluated using the `cocoevaluator` library to produce the required values. The vehicle image dataset used here displays a varied and diverse logo size, from small to large objects represented in the area parameter.

Regarding logo objects that are small in scale, the size standard includes areas that are smaller than 32×32 pixels, which refers to COCO [29]. In the testing dataset used, there are 169 images in the small area category, 312 images in the medium category, and 19 images in the large scale. This small variation in logo size becomes a challenge in the logo detection process. The evaluation results of the conducted experiments are shown in Tables 4 and 5.

Table 4. AP Evaluation Results

GCV	AGB	IoU 0.5 (All)	IoU 0.5 (Small)
0.05	4	0.955	0.909
	8	0.966	0.934
	12	0.965	0.936
	16	0.964	0.917
0.1	4	0.946	0.902
	8	0.956	0.914
	12	0.963	0.903
	16	0.955	0.894
0.3	4	0.937	0.889
	8	0.953	0.885
	12	0.966	0.930
	16	0.962	0.932
0.5	4	0.968	0.952
	8	0.954	0.894
	12	0.956	0.920
	16	0.956	0.923

Table 4 shows that the GCV parameter (0.5) outperforms others, achieving an AP50 of 0.968 for all-area and 0.952 for small-area when combined with AGB set to 4.

Table 5. AR Evaluation Results

GCV	AGB	IoU 0.5 (All)	IoU 0.5 (Small)
0.05	4	0.965	0.928
	8	0.975	0.956
	12	0.975	0.956
	16	0.970	0.936
0.1	4	0.959	0.922
	8	0.967	0.933
	12	0.970	0.939
	16	0.961	0.914
0.3	4	0.950	0.901
	8	0.963	0.917
	12	0.972	0.945
	16	0.971	0.953
0.5	4	0.975	0.961
	8	0.961	0.914
	12	0.968	0.934
	16	0.968	0.942

Based on the results in Table 5, the GCV parameter with a value of 0.5 again demonstrates the highest performance compared to other values. Similarly, the AGB parameter set to 4 achieves an AR50 value of 0.975 for the all-area category and 0.961 for the small area category. From the evaluation results of AP and AR, the highest values at IoU 50 are achieved with the GCV configuration set to 0.5 and AGB set to 4. This parameter configuration consistently gets the highest values among others in both areas. To see more in the GCV 0.5 configuration, you can see the log in Table 6.

Table 6. Test Results at GCV 0.5

GCV	AGB	Training Loss	Steps	Training Time	Inference Time
0.5	4	0.4524	23.449	29.69 h	0.151 s
	8	0.5935	11.724	23.88 h	0.128 s
	12	0.4611	7.824	33.10 h	0.111 s
	16	0.5510	5.874	48.76 h	0.113 s

The results in Table 6 show that as the value of AGB increases, the number of steps executed decreases. This proves that the way AGB works is to reduce the update frequency of the model. In general, as the value of AGB increases, the time taken tends to be longer. In the configuration AGB of 8, there is a trade-off between time and training loss. It reflects that the training time run is shorter than others, but it increases the training loss value to be very high at a value of 0.5935. Therefore, the best configuration is obtained based on the results of the above analysis, that is GCV with a value of 0.5 and AGB of 4. This configuration strikes a balance between model update frequency and training efficiency.

Table 7. Comparison Results of the Two Backbones

Backbone	GCV	AGB	Training Loss	Training Time	AP50 All	AP50 Small	AR50 All	AR50 Small
R50	0.5	4	0.4524	29.69 h	0.968	0.952	0.975	0.961
R101			0.3888	61.93 h	0.961	0.958	0.974	0.967

Furthermore, a comparison was made with different backbones, ResNet-50 and ResNet-101, for vehicle

logo detection. The purpose of this comparison of different backbones is to see how a more complex and

deep architecture can affect the accuracy and efficiency of vehicle logo detection, especially for small and hard to recognize logo.

In Table 7, the comparison results between the two backbones with the same GCV and AGB configurations are shown. The results show that both backbones are able to perform vehicle logo detection quite well. ResNet-101 tends to produce higher AP and AR Small values compared to ResNet-50. This shows that ResNet-101 detects small vehicle logo better. However, ResNet-50 shows a much faster training time of 29.69 hours while ResNet-101 takes 61.93 hours. Thus, from the research results in the table, ResNet-101 produces higher AP and AR Small values, but there is a trade off in the time taken which is quite long.

This shows that ResNet-101 is better at detecting small objects such as vehicle logo more accurately. However, due to the more complex architecture of ResNet-101, it requires longer training time and a larger number of parameters. In contrast, ResNet-50 has a simpler architecture and requires less training time.

3.4 Log Model Results

In this section, we will analyse the resulting logs. These logs allow us to review the performance process of the model during the training period. The results are interpreted through a visual representation including training loss and validation loss.

In this analysis, a comparison will be made between two models with different configurations. Model 1 employs a configuration without involving GCV and AGB parameters, while Model 2 applies a configuration with GCV parameters set to 0.5 and AGB set to 4. The analysis of the two models is used to identify the influence of GCV and AGB parameters on the training

process. The experimental results are shown in Table 8. Based on the results of Table 8, it can be seen that Model 2 achieved better results than Model 1. The differences between the two models are quite significant from various aspects. For visualization, the result of the table results is shown in Figures 5 and 6.

Table 8. Comparison Results of the Two Models

Model	Steps	Training Loss	Validation Loss	Time
Model 1	93.724	1.3016	6.1433	37.27 h
Model 2	23.449	0.4524	0.6152	29.69 h

Figure 5 illustrates the training results for both models, providing a detailed comparison of their performance during the training process. In this visualization, version 25 corresponds to Model 1, while versions 33 and 34 correspond to Model 2. The results reveal a clear distinction in the behavior of the two models throughout the training period. The results show that Model 1 is unable to reduce the value of training loss. Although it initially shows a downward trend, training loss increases at step 6,309 (epoch 1). In contrast, Model 2 demonstrates good results, so that it converges. It can be seen that the difference in steps between the two models is very significant, this is because Model 2 implements gradient clipping. The parameter limits the gradient value, so the process of updating the model parameters becomes smaller.

The results in Figure 6 show that Model 1 graph has increased quite high. This indicates that the model is difficult to recognize the logo. The difference between the two models is significant. It is shown that the Model 1 tends to fluctuate so that to simplify visualization, scalar smoothing with a value of 0.99 is used in order to see the trend of the graph clearly.

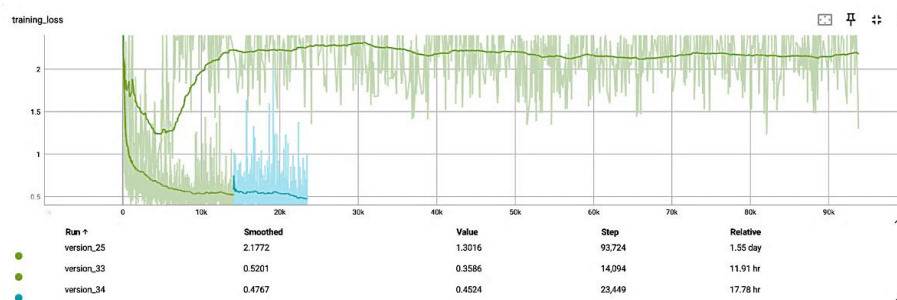


Figure 5. Visualization of the training loss results.

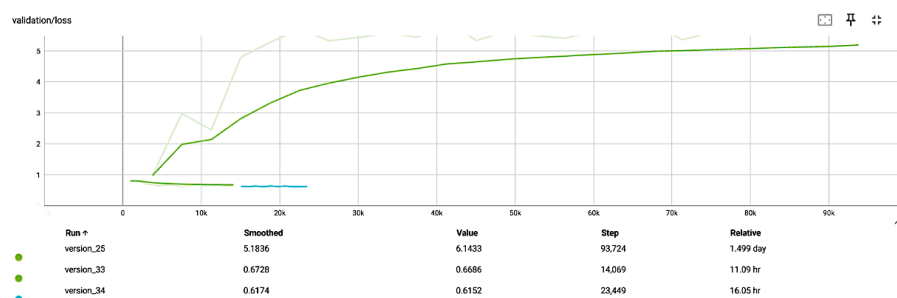


Figure 6. Visualization of the validation loss results.

3.5 Detection Results

The next process involves performing inference on the image to detect the vehicle logo using the DETR model. The analysis is done by observing the model's ability to detect logo based on the predictions generated. The inference results are visualized using the bounding boxes on each image. By examining these predictions, it is possible to gauge how well the model has learned to distinguish logo from other features within vehicle images. The inference results are visualized with bounding boxes superimposed on each image, providing a clear representation of the model's detections. Figure 7 provides an example of an image successfully detected by the model.

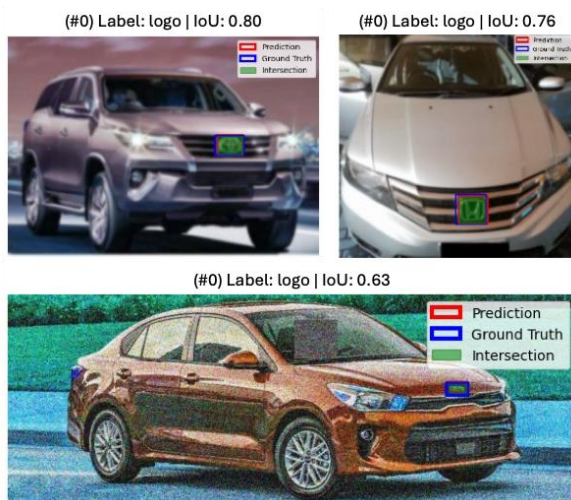


Figure 5. DETR model successfully detects vehicle logos.

Figure 7 shows that the model successfully performs vehicle logo detection with a relatively high accuracy, achieving IoU scores of 0.80 and 0.76, respectively. These values indicate that the model can correctly localize vehicle logos in most cases, demonstrating its effectiveness in detecting brand emblems under standard conditions. The consistent detection performance suggests that the model has learned to recognize key features of the logos, such as their shape, contrast, and placement on the vehicle body.

However, as shown in Figure 8, the model fails to detect vehicle logos in some images and instead misidentifies other car parts, such as headlights and door handles. These misdetections suggest that the model may struggle to differentiate logo-specific features from general vehicle design elements, particularly in challenging conditions such as poor lighting, occlusions, or low-resolution images. Additionally, it sometimes performs multiple detections where only one should exist. This issue may be caused by image noise, variations in logo size and orientation, or inconsistent training annotations. These findings indicate the need for further improvements, such as data augmentation, model fine-tuning, or additional contextual cues, to enhance detection robustness and accuracy.

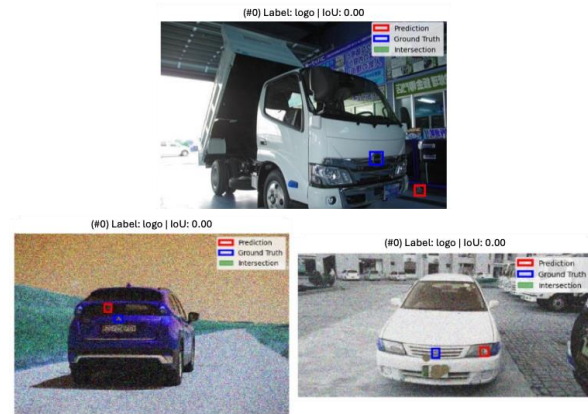


Figure 6. DETR model is failed to detect vehicle logos.

4. Conclusions

The test results show that the DETR model is capable of performing vehicle logo detection very well. The main focus of this test is to assess the performance of the DETR model in identifying small vehicle logo and to understand how tuning certain parameters impact the performance of the model. The test results indicate that using the parameters Gradient Clip Val set to 0.5 and Accumulate Gradient Batches set to 4 the best evaluation values. The configuration gets AP and AR evaluation results with values of 0.952 and 0.961 respectively at IoU 50 for Small data.

Although the model successfully detects vehicle logo, there are still instance where it produces multiple detections for a single image. It was found that some shapes such as car headlights are mistakenly identified by the model as vehicle logo. To maximize the potential of the model, further research is needed so that the application of the model on various objects is more optimal.

Acknowledgment

The authors would like to thank the Directorate of Research and Community Service, Telkom University for the financial support under the TUKJ-Collaborative Strengthening Grant Scheme in 2024 with an agreement number 887/PNLT3/PPM/2023.

References

- [1] A. Eldho, T. Francis, and C. V. Hari, "YOLO based Logo detection," in *2019 9th International Conference on Advances in Computing and Communication (ICACC)*, IEEE, Nov. 2019, pp. 283–289. doi: 10.1109/ICACC48162.2019.8986207.
- [2] J. Zhang, S. Yang, C. Bo, and Z. Zhang, "Vehicle logo detection based on deep convolutional networks," *Computers & Electrical Engineering*, vol. 90, p. 107004, Mar. 2021, doi: 10.1016/j.compeleceng.2021.107004.
- [3] J. Sahel, M. Alsahafi, M. Alghamdi, and T. Alsubait, "Logo Detection Using Deep Learning with Pretrained CNN Models," *Engineering, Technology & Applied Science Research*, vol. 11, no. 1, pp. 6724–6729, Feb. 2021, doi: 10.48084/etasr.3919.
- [4] J. Zhang, L. Chen, C. Bo, and S. Yang, "Multi-Scale Vehicle Logo Detector," *Mobile Networks and Applications*, vol. 26,

- no. 1, pp. 67–76, Feb. 2021, doi: 10.1007/s11036-020-01722-0.
- [5] F. C. Soon, H. Y. Khaw, J. H. Chuah, and J. Kanesan, “Hyper-parameters optimisation of deep CNN architecture for vehicle logo recognition,” *IET Intelligent Transport Systems*, vol. 12, no. 8, pp. 939–946, Oct. 2018, doi: 10.1049/iet-its.2018.5127.
- [6] Y. Huang, R. Wu, Y. Sun, W. Wang, and X. Ding, “Vehicle Logo Recognition System Based on Convolutional Neural Networks with a Pretraining Strategy,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 1951–1960, Aug. 2015, doi: 10.1109/TITS.2014.2387069.
- [7] F. C. Soon, H. Y. Khaw, J. H. Chuah, and J. Kanesan, “Vehicle logo recognition using whitening transformation and deep learning,” *Signal Image Video Process*, vol. 13, no. 1, pp. 111–119, Feb. 2019, doi: 10.1007/s11760-018-1335-4.
- [8] A. Omid, A. Heydarian, A. Mohammadshahi, B. A. Beirami, and F. Haddadi, “An Embedded Deep Learning-based Package for Traffic Law Enforcement,” in *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, IEEE, Oct. 2021, pp. 262–271. doi: 10.1109/ICCVW54120.2021.00034.
- [9] S. Yang, J. Zhang, C. Bo, M. Wang, and L. Chen, “Fast vehicle logo detection in complex scenes,” *Opt Laser Technol*, vol. 110, pp. 196–201, Feb. 2019, doi: 10.1016/j.optlastec.2018.08.007.
- [10] T. Diwan, G. Anirudh, and J. V. Tembhurne, “Object detection using YOLO: challenges, architectural successors, datasets and applications,” *Multimed Tools Appl*, vol. 82, no. 6, pp. 9243–9275, Mar. 2023, doi: 10.1007/s11042-022-13644-y.
- [11] M. A. M. Ali, T. Aly, A. T. Raslan, M. Gheith, and E. A. Amin, “Advancing Crowd Object Detection: A Review of YOLO, CNN and ViTs Hybrid Approach,” *Journal of Intelligent Learning Systems and Applications*, vol. 16, no. 03, pp. 175–221, 2024, doi: 10.4236/jilsa.2024.163011.
- [12] S. Dalal, U. K. Lilhore, M. Radulescu, S. Simaiya, V. Jaglan, and A. Sharma, “A hybrid LBP-CNN with YOLO-v5-based fire and smoke detection model in various environmental conditions for environmental sustainability in smart city,” *Environmental Science and Pollution Research*, Jan. 2024, doi: 10.1007/s11356-024-32023-8.
- [13] G. Dai, L. Hu, and J. Fan, “DA-ActNN-YOLOV5: Hybrid YOLO v5 Model with Data Augmentation and Activation of Compression Mechanism for Potato Disease Identification,” *Comput Intell Neurosci*, vol. 2022, p. 6114061, 2022, doi: 10.1155/2022/6114061.
- [14] M. Lin et al., “DETR for Crowd Pedestrian Detection,” Dec. 2020.
- [15] Z. Sultan, M. U. Farooq, and R. H. Raza, “Improved Vehicle Logo Detection and Recognition for Complex Traffic Environments Using Deep Learning Based Unwarping of Extracted Logo Regions in Varying Angles,” 2023, pp. 12–25. doi: 10.1007/978-3-031-37649-8_2.
- [16] M. Jin and J. Zhang, “Research on Microscale Vehicle Logo Detection Based on Real-Time DETection TRansformer (RT-DETR),” *Sensors (Basel)*, vol. 24, no. 21, Oct. 2024, doi: 10.3390/s24216987.
- [17] Y. Kong, X. Shang, and S. Jia, “Drone-DETR: Efficient Small Object Detection for Remote Sensing Image Using Enhanced RT-DETR Model,” *Sensors*, vol. 24, no. 17, p. 5496, Aug. 2024, doi: 10.3390/s24175496.
- [18] J. Huang and H. Wang, “Small Object Detection by DETR via Information Augmentation and Adaptive Feature Fusion,” Jan. 2024.
- [19] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, “End-to-End Object Detection with Transformers,” May 2020.
- [20] U. Farooq, “Vehicle Logos (VL) - 10.” Accessed: Dec. 07, 2023. [Online]. Available: <https://www.kaggle.com/datasets/mufarooqq/vl-10/data>
- [21] J. Zhang, T. He, S. Sra, and A. Jadbabaie, “Why gradient clipping accelerates training: A theoretical justification for adaptivity,” May 2019.
- [22] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning*. London: The MIT Press, 2016.
- [23] Sebastian Raschka, “Finetuning LLMs on a Single GPU Using Gradient Accumulation.” Accessed: Jul. 11, 2024. [Online]. Available: <https://lightning.ai/blog/gradient-accumulation/>
- [24] J. Lamy-Poirier, “Layered gradient accumulation and modular pipeline parallelism: fast and efficient training of large language models,” Jun. 2021.
- [25] Z. Huang, B. Jiang, T. Guo, and Y. Liu, “Measuring the Impact of Gradient Accumulation on Cloud-based Distributed Training,” in *2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, IEEE, May 2023, pp. 344–354. doi: 10.1109/CCGrid57682.2023.00040.
- [26] Pytorch Lightning, “Effective Training Techniques.” Accessed: Jul. 08, 2024. [Online]. Available: https://lightning.ai/docs/pytorch/stable/advanced/training_tricks.html#accumulate-gradients
- [27] Jiaqian Yu, Jingtao Xu, Yiwei Chen, and Weiming Li, “Learning Generalized Intersection Over Union for Dense Pixelwise Prediction,” *Proceedings of the 38th International Conference on Machine Learning*, 2021.
- [28] R. Padilla, S. L. Netto, and E. A. B. da Silva, “A Survey on Performance Metrics for Object-Detection Algorithms,” in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, IEEE, Jul. 2020, pp. 237–242. doi: 10.1109/IWSSIP48289.2020.9145130.
- [29] COCO - Common Objects in Context, “Detection Evaluation.” Accessed: Sep. 08, 2024. [Online]. Available: <https://cocodataset.org/#detection-eval>
- [30] A. Benali Amjoud and M. Amrouch, “Convolutional Neural Networks Backbones for Object Detection,” 2020, pp. 282–289. doi: 10.1007/978-3-030-51935-3_30.
- [31] E. Suherman, B. Rahman, D. Hindarto, and H. Santoso, “Implementation of ResNet-50 on End-to-End Object Detection (DETR) on Objects,” *Sinkron*, vol. 8, no. 2, pp. 1085–1096, Apr. 2023, doi: 10.33395/sinkron.v8i2.12378.
- [32] I. Loshchilov and F. Hutter, “Decoupled Weight Decay Regularization,” Nov. 2017.