

Detecting road damage utilizing retinanet and mobilenet models on edge devices

Haniah Mahmudah^{1,2}, Aulia Siti Aisjah², Syamsul Arifin¹, Catur Arif Prastyanto³

¹Department of Electrical Engineering, Politeknik Elektronika Negeri Surabaya, Surabaya, Indonesia

²Department of Physics Engineering, Faculty of Industrial and Systems Engineering, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia

³Department of Civil Engineering, Faculty of Civil, Planning, and Geo Engineering, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia

Article Info

Article history:

Received Mar 26, 2024

Revised Oct 25, 2024

Accepted Nov 14, 2024

Keywords:

Edge device

Inference time

MobileNet model

RetinaNet model

Tuning hyperparameters

ABSTRACT

A particular form of road digitalization produces a system that detects road damage automatically and in real time, employing the device to detect road damage as an edge device. The application of RetinaNet152 and MobileNetV2 models for road damage detection on edge devices necessitates a trade-off between high system performance and efficiency. Currently, edge devices have limited storage. In this paper, we explore how tuning hyperparameters with batch size and several optimizers improves system performance on RetinaNet152 and MobileNet models, as well as how they are implemented on edge devices. After tuning hyperparameters in the batch size of the optimizer, the Adam optimizer displayed enhanced performance with mean average precision (mAP), average recall (AR), and F1-score. This implies a positive impact on overall model performance. The MobileNetV2 model's hyperparameter tuning technique significantly improves performance, resulting in faster inference times and overall system performance. This demonstrates that the MobileNetV2 model could be used directly on edge devices to identify road damage. However, the RetinaNet152 model has a lower inference time, which cannot be deployed directly to edge devices. The RetinaNet152 model can be deployed on edge devices; however, a technique for speeding up inference time is essential.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Aulia Siti Aisjah

Department of Physics Engineering, Faculty of Industrial and Systems Engineering

Institut Teknologi Sepuluh Nopember

ITS Campus, Sukolilo, Surabaya, 60111, Indonesia

Email: auliasa@its.ac.id

1. INTRODUCTION

A particular form of road digitalization produces a system that detects road damage automatically and in real time, employing the device to detect road damage as an edge device. The implementation of digitalization on roads through automation, artificial intelligence (AI), and digital information flow results in enhanced efficiency, lower operating costs, and improved road services. A particular application of digitalization on roads is in road maintenance programs that support smart cities [1]–[3]. Several studies have been conducted to detect road damage utilizing AI technologies, camera sensors and various image data processing approaches. Some of the research with approach methodologies include image-based image processing techniques [4]–[9], machine learning (ML) model [10]–[14], and deep learning method [15]–[27].

This investigation employs image processing techniques for a variety of applications, including the detection of road defects. Potholes are a type of road defect that can be detected using classic image

processing techniques. This technique needs numerous stages to achieve high accuracy, including manually extracting image features and tweaking image processing parameters. The speed of image frames varies depending on road conditions [4]. The ML model includes a data processing stage that manually pulls features to increase accuracy, which requires a significant number of computational resources. Several studies have used deep convolutional neural network (CNN) approaches to automate the process of feature extraction and categorization at the same time [18], [26].

A particular application of the CNN model for road problem identification on edge devices necessitates a trade-off between high accuracy and high efficiency [13]. The CNN models are highly accurate in terms of system performance recall, precision, high accuracy, and low loss. This necessitates CNN models with massive architecture and competitive computing costs that are inefficient. As a result, it requires a simpler CNN model configuration setting while maintaining good accuracy. The CNN models are highly efficient in that they can detect road faults on edge devices with fast inference times. Because the CNN model on the edge device has limited storage, inference time must be taken into consideration. The CNN models have fast inference times, perform well on simpler model configurations, but are difficult to obtain high accuracy. To achieve excellent system performance, CNN model parameters must be carefully selected. When selecting detection models, there is no conclusive answer as to which model has the highest performance, but one must make decisions based on demands [13], [14].

In order to produce a system that detects road damage automatically and in real time, CNN model research must be developed. The development study in road damage detection enhances system performance by utilizing one- and two-stage CNN models. Some research on road damage detection using CNN includes detection techniques using CNN models with semi-supervised learning using pseudo-labels [15], CNN one-stage detector model architecture namely InceptionV2 and MobileNet on personal computer (PC) [16], YOLO namely Tiny-YOLOv2, darknet neural networks, YOLOv3, Tiny-YOLOv3 and YOLOv4 [20], [21], YOLOv5 on smartphones [22], YOLOv3, YOLOv2 and TinyV3 models [23], YOLOv5, two variations of YOLOR, and faster R-CNN with five different backbones namely ResNet50, VGG16, MobileNetV2, InceptionV3, and proposed modified VGG16 (MVGG16) [24]. The results of road damage detection research implementing the CNN model include system performance features such as accuracy, mean average precision (mAP), average recall, and the F1-score.

One effort to increase accuracy in CNN models is tuning hyperparameters relating to network structure and training. The tuning hyperparameters consider network structure, namely kernel size, width, and depth. Researchers examined three network designs of a CNN one-stage detector model (small, medium, and large), as well as a combination of hyperparameter tuning and activation function adjustments. System performance can be improved by implementing a CNN model network architecture and adjusting hyperparameters [17], [18], [26], [27]. Several research have achieved high accuracy by employing an optimizer to change hyperparameters on CNN models. To obtain good performance, several researchers on object identification employ one or two optimizers as tuning hyperparameters on pretrained CNN models [28]–[31]. However, none of the CNN model research has been deployed on edge devices for detecting road damage.

Several road defect investigations were conducted utilizing CNN models on edge devices, resulting in systems that detect road faults automatically and in real time. Maeda *et al.* [16] employed MobileNet on a smartphone, achieving a system performance of 71% and an inference time of 1.5 seconds. Other research applies the single shot multi-box detector (SSD)-MobileNet model with a batch size of 64 to NVIDIA Jetson Nano devices with accelerators [32] and unmanned aerial vehicles (UAV) using Raspberry Pi [33]. Other researchers created MobileNetV2 with hyperparameter learning rate and batch size settings using NVIDIA Jetson Nano, which has a higher mAP of 0.0869 and a lower total loss training of 0.6028 compared to SSD Resnet50V1 [34], [35]. Based on the results of that investigation, the CNN model is merged into hardware devices that have been developed but still require additional research to increase system performance and accelerate inference time, as shown in Table 1.

The challenge of detecting road damage on edge devices is to do it automatically and in real time as the edge device moves at a specific speed. This necessitates the deployment of a CNN mode on an edge device capable of overcoming this limitation. The RetinaNet152 model detects objects in images by combining anchor boxes with feature pyramid networks. RetinaNet152 refers to the number of layers in the backbone network, which in this case is ResNet-152. ResNet-152 is well-suited for road damage detection since it can recognize and categorize objects with high accuracy and speed, which is critical for real-time applications. Based on the advantages of RetinaNet's research on road damage by modifying the backbone [36], it employs RetinaNet152 with hyperparameter optimizer adjustment [37]. The RetinaNet152 model has advantages; however, it has not been deployed on edge devices for road damage detection equipment. According to Table 1, the MobileNetV2 model [34] must increase system performance by modifying hyperparameters using an optimizer.

Table 1. The CNN models on road damage

Previous studies	Model CNN	Object detection	Performance system	Edge device
[16] [32]	MobileNet MobileNetV2	Eight types of road damage Potholes, longitudinal cracks, alligator cracks	Recall, precision, inference time mAP	Smartphone NVIDIA Jetson Nano
[33] [34]	MobileNet V1 MobileNetV2	Pavement Potholes, longitudinal cracks, alligator cracks	Accuracy, FPS mAP, AR	Raspberry Pi NVIDIA Jetson Nano
Proposed system	RetinaNet152 MobileNetV2	Potholes, longitudinal cracks, alligator cracks	mAP, AR, F1-score, size model, inference time, FPS	NVIDIA Jetson Nano

To address a research gap in this study, we present a system that improves the system performance of the RetinaNet152 and MobilenetV2 models in a road damage detection system on edge devices, thereby improving system performance and inference time. The research results with optimization produced good system performance. Road damage detection systems for edge devices with limited memory and mobile data retrieval use the RetinaNet152 and MobilenetV2 models. Larger batch sizes can help speed up the training process by allowing the model to process more data concurrently. This can lead to faster convergence and improved performance. A larger batch size can help reduce gradient variance, hence improving model stability and accuracy [37]. However, it is particularly important to note that raising the batch size and optimizer can result in higher memory needs and the need for more processing resources. To increase system performance and implementation on edge devices with environmental variations, the road damage detection system needs to simulate batch sizes and optimizer parameters on RetinaNet152 and MobileNet models.

The primary contributions of this research to the development of road defect detection systems are described as follows:

- The development of RetinaNet152 [38] and MobileNetV2 [34] models for road damage is based on tuning hyperparameters: batch size and optimizers to improve performance system mAP, AR, and F1-score.
- Edge-device deployment of the RetinaNet152 and MobileNetV2 models for real-time road damage detection. This research uses a road damage system and a road damage detection system.
- Comparing the analysis systems performance of the Retinanet152 and MobileNetV2 models, inference times, frames per second, and size model when deployed on edge devices.

The paper has four parts. Section 1 introduction focuses on the topic and research goal. Section 2 covers methods. Section 3 contains results and discussions: i) the system's performance tuning hyperparameter, ii) deploying the RetinaNet152 and MobileNetV2 models on edge device, and iii) comparison system performance model CNN. Finally, section 4 presents ideas for future research.

2. METHOD

Several issues arise when gathering and identifying road defects when moving with edge devices. Potholes, alligator cracks, and longitudinal fractures are produced by capturing road damage statistics on edge devices in real time and while in motion. Under certain conditions, the recoverable cracks are very small and unclear. This is due to the tiny size of the cracks and the fluctuation of road damage object detection in the surrounding environment, which makes it difficult to discern road damage in the form of cracks. Furthermore, road damage datasets in the form of fractures result in nearly identical color shifts between the detected items and the surrounding surroundings. This is the challenge of gathering moving data on road damage on an edge device. A CNN model is required for good performance and can be deployed on edge devices.

The problem concerning road damage detection devices in the form of mobile edge devices exists because they have limited computation and storage capabilities. Thus, the CNN model deployed on edge devices must be considered. Some studies employ the CNN one-stage detector MobileNetV2 [16], [32], [34] and RetinaNet152 [38] models installed on edge devices. The RetinaNet152 model has been selected for current research to provide real-time detection and classification with hyperparameter tuning to increase system performance, which includes mAP, recall, and F1-score. This research employs the RetinaNet152 model, with further advancements in road damage identification systems through batch size and real-time video testing.

The RetinaNet152 and MobileNetV2 models are employed in a variety of steps, including dataset retrieval, pre-processing, annotation, batch size and optimizer modeling, and system performance evaluation. The data analysis parameters are mAP, AR, and F1-score. The RetinaNet152 and MobileNetV2 models were converted to TFlite and deployed on an edge device using the NVIDIA Jetson Nano, allowing the inference time to be measured. Validation testing of the RetinaNet152 and MobileNetV2 road damage categorization

models involved validating video and then evaluating system performance. Figure 1 shows the modeling procedure for RetinaNet152 and MobileNetV2 models, including hyperparameter tuning, performance analysis, and deployment on edge devices.

This dataset retrieval combines a collection of road damage images from the East Java Public Works Office's Directorate of Highways survey findings with data obtained directly. Datasets were collected on provincial highways in East Java utilizing the road damage tool system [15]. The dataset retrieval images returned 11,176 images.

Pre-processing the dataset is required to equalize the image's size and format, allowing for faster performance and improved model performance. The Windows utility converts image data into 640×640-pixel images. The dataset is annotated to identify and describe characteristics based on item classification. The labelling tool annotates the entire dataset. This annotation process selects the type of road damage classification. The current research utilizes four labeling datasets for road damage classification: potholes (L00), longitudinal cracks (R02), alligator cracks (R03), and shadows (00). The dataset labeling method addresses the demand for road damage data based on the asphalt road condition survey form provided by the Department of Public Works, Directorate General of Highways in Indonesia.

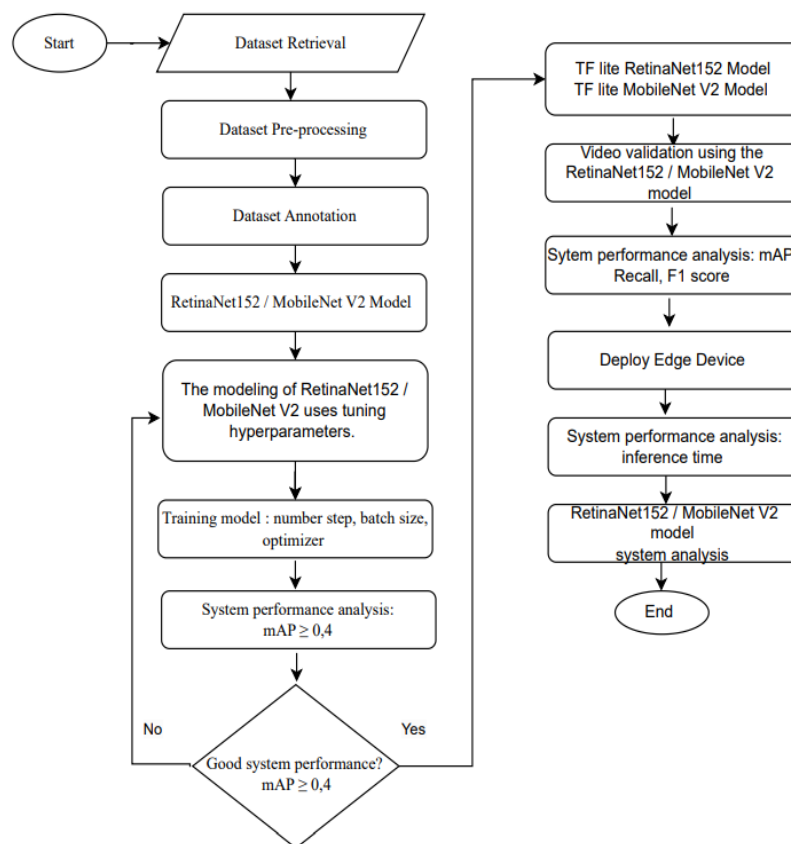


Figure 1. The RetinaNet152 and MobileNetV2 models use hyperparameter tuning

The road damage detection procedure makes use of deep learning technology, specifically the RetinaNet152 and MobileNetV2 models, which have already been trained. This decision is taken to obtain peak performance while drastically reducing training time compared to starting from scratch. The RetinaNet152 and MobileNetV2 models are developed using the TensorFlow object detection API, an open-source framework based on TensorFlow. This API simplifies the development, training, and deployment of object detection models. The TensorFlow object detection API includes the model Zoo, a collection of pre-trained models with various architectures. These models were trained on many datasets, including COCO, KITTI, and Open Image. Using these pre-trained models promotes transfer learning and allows customization to meet a variety of operational objectives. This approach provides a solid foundation for the road damage classification system.

This research analyzes the use of optimizers like momentum, RMSprop, and Adam on the RetinaNet152 and MobileNetV2 models. The momentum optimizer can help to overcome the issue of delayed convergence and accelerate model training. The momentum optimizer assists in filling local gaps and enhances converge in the right direction. momentum optimizer parameter equations as in (1) and (2) [38].

$$v_{(t)} = \beta \cdot v_{(t-1)} + (1 - \beta) \cdot \nabla F_{\cdot}(W_t) \quad (1)$$

$$W_{(t-1)} = W_t - \alpha \cdot v_{(t)} \quad (2)$$

Where t is the parameter at the t -iteration, α is the learning rate, $\nabla F(W_t)$ is the gradient of the cost function with respect to the parameter at iteration t . RMSprop is one of the optimization algorithms that maintains the squared gradient average for each weight. The equation RMSprop is shown in (3) [38].

$$MeanSquare_{(w,t)} = \rho \times MeanSquare_{(w,t-1)} + 0.1(\partial E \partial w_{(t)})^2 \quad (3)$$

Adam optimizer in (4) [38].

$$W_t = W_{t-1} - \alpha \left(\frac{m_t}{\sqrt{v_t}} + \varepsilon \right) \quad (4)$$

Where g is gradient, m is first moment, v is second moment, β_1, β_2 are exponential decay rates, α is learning rate, W the parameter is the weight. During the optimization phase for the RetinaNet152 and MobileNetV2 models, trials are carried out with various learning rates and optimizer settings. To determine the test outcomes, use the evaluation matrix defined by the two variables in (5) and (6) [19], [38].

$$Precision = \frac{TP}{TP+FP} \quad (5)$$

$$Recall = \frac{TP}{TP+FN} \quad (6)$$

The total loss is the result of an incorrect forecast, while the loss is a numerical value that measures the degree of inaccuracy in the model's prediction. Precision is defined as the ratio of correct positive predictions to the total number of positive instances predicted; it is simply the true positives (TP) divided by total detections that are true positives (TP) and false positives (FP). Recall quantifies the ratio of successfully predicted actual positives, specifically the true positives (TP), to all ground truth values that are true positives (TP) and false negative (FN). The F1-score is an evaluation metric that determines the accuracy of the RetinaNet152 and MobileNetV2 models. It brings together a model's precision and recall scores as shown in (7) [19]. The accuracy statistic calculates the number of times the RetinaNet152 and MobileNetV2 models predicted correctly over the full dataset.

$$F1 \text{ score} = 2 \times \frac{(Precision \times Recall)}{Precision+Recall} \quad (7)$$

Validation testing of the road damage classification model was performed using video samples lasting around 40 seconds. Real-time video capture, which is based on the road damage detection system, records the road segment in motion using a camera. During video capture, the camera is designed to be mounted on the car's front bonnet at an angle of around 28 degrees to the road. The weather conditions during data collection were described as partially cloudy. Video validation testing of the RetinaNet152 and MobileNetV2 models was carried out on the Jupyter Notebook server.

3. RESULTS AND DISCUSSION

This section presents the research achievements of the proposed system from the RetinaNet152 and MobileNetV2 models. The system performance analysis results of RetinaNet152 and MobileNetV2 models include hyperparameter tuning system performance analysis, video validation, and deployment results on edge devices, specifically NVIDIA Jetson 4 GB. In addition, comparative analysis of mAP, AR, and F1-score system performance, inference time, and previous work and the proposed system. Specifically, the stages are stated as follows.

3.1. The system's performance tuning hyperparameter

The influence of batch size and optimizer could affect total loss and system performance. Table 2 shows the relationship between batch size and optimizer for the proposed RetinaNet152 and MobileNetV2 models. The simulation results of the RetinaNet152 model show that increasing the batch size could develop a more stable gradient estimate, allowing the momentum optimizer to produce a more consistent gradient descent direction, resulting in less loss. The RetinaNet152 model uses the momentum optimizer, and batch size 4 loses 0.573, whereas batch size 8 loses 0.225. The RMSProp optimizer adjusts the learning rate for each parameter based on the most recent observation and the average gradient. However, the RMSProp optimizer has a method for managing important gradient variance, resulting in a loss of 0.802 for batch size 4 to 1.088 for batch size 8, as compared to momentum.

Table 2. The system performance of RetinaNet152 and MobileNetV2 models

Proposed Sytem	Parameter (Optimizer, Batch size)	Loss	mAP (@IoU 0.5-0.95)	mAP (@IoU 0.5)	(AR)	F1-score (@IoU 0.5-0.95)	F1-score (@IoU 0.5)
RetinaNet152 Model	Momentum, 4	0.573	0.277	0.406	0.449	0.343	0.426
	RMSprop, 4	0.802	0.235	0.454	0.457	0.310	0.455
	Adam, 4	0.264	0.351	0.457	0.499	0.412	0.477
	Momentum, 8	0.225	0.334	0.455	0.423	0.373	0.438
	RMSprop, 8	1.088	0.256	0.477	0.446	0.325	0.461
	Adam, 8	0.220	0.349	0.469	0.410	0.377	0.438
MobileNetV2 Model	Momentum, 4	0.326	0.292	0.563	0.467	0.349	0.451
	RMSprop, 4	1.970	0	0	0	0	0
	Adam, 4	0.656	0.208	0.578	0.408	0.338	0.462
	Momentum, 8	0.668	0.201	0.561	0.368	0.342	0.450
	RMSprop, 8	2.010	0	0	0	0	0
	Adam, 8	0.320	0.298	0.606	0.476	0.322	0.481

Based on the second estimate of the gradient moments, Adam's optimizer adjusts the learning rate adaptively to modify the batch size. Simulation results for the RetinaNet152 model show that a very small batch size of 4 could produce an unstable estimate of the gradient moment, resulting in a loss of 0.264, but a batch size of 8 makes the Adam optimizer more stable, resulting in a loss of just 0.220. The batch size 8 and Adam optimizer scenario had a higher performance system with mAP 0.469 and AR 0.410, resulting in the highest F1-score value of 0.438. The scenario had a less overall loss than the other possibilities, which is advantageous.

Simulation results for the MobileNetV2 model demonstrate that raising the batch size can lead to more stable gradient estimation, allowing the momentum optimizer to lower the loss of 0.326 at batch size 4 and 0.668 at batch size 8. However, the RMSProp optimizer adapts the learning rate to the average gradient. However, the RMSProp optimizer generates a loss higher than one; therefore, it cannot attain system performance. Adam's optimizer with adaptive produces a more constant gradient descent direction, which leads to less loss. Adam's optimizer has a lower loss of 0.656 at batch size 4 and 0.320 at batch size 8 than momentum and RMSprop. The results of system performance according to the MobileNetV2 model, batch size 8, and Adam optimizer scenario produced a higher performance system with mAP 0.606 and AR 0.476, resulting in the highest F1-score value of 0.481.

The batch size indicates the number of samples utilized in a single training iteration before the Retinanet152 and MobileNetV2 model weights are adjusted. The RetinaNet152 model illustrates that raising the batch size reduces loss in the momentum and Adam optimizers since the resulting gradient is an average of more data, except in RMSprop, where increasing the batch size increases loss. The MobileNetV2 model reveals that increasing the batch size on the momentum optimizer and RMSprop increases the loss. The MobileNetV2 model minimizes loss on the Adam optimizer while maintaining acceptable system performance (mAP, AR, and F1-score). Overall, the RetinaNet152 and MobileNetV2 models have the lowest loss and the best system performance (mAP, AR, and F1-score) in batch size 8 and Adam optimizer.

When implementing the Retinanet152 and MobileNetV2 models on edge devices with limited memory, it is important to keep in mind that if the batch size exceeds the available memory capacity, memory overflow and performance challenges may occur. In road defect detection systems, the Retinanet152 and MobileNetV2 models are used on edge devices with limited memory. This is an important issue to consider while selecting the RetinaNet152 and MobileNetV2 models on settings such as batch size and optimizer. It is critical to undertake procedures and cross-validation to determine the best combination of the RetinaNet152 and MobileNetV2 models based on parameters such as batch size and optimizer.

Our findings suggest that the proposed systems use RetinaNet152 and MobileNetV2, and that adding an adaptive optimizer, specifically Adam, can result in improved system performance. Some of the

issues faced by this system are road damage datasets with almost equal variance between detection objects and the surrounding environment, small road damage detection objects, and object retrieval while in motion. This results in inadequate system performance, enhancing this issue of real-time road damage detection research using the device. As a result, more investigation is required to increase the performance of a more optimal system by including many approaches into the one-stage CNN model used on edge devices.

3.2. Video validation and deploying RetinaNet152 and MobileNetV2 models on edge device

The evaluation of the RetinaNet152 and MobileNetV2 models is based on classification results, which have a considerable impact on road damage calculations. The test seeks to differentiate the model's predictions when running on video samples to see if they match the intended objects. Confusion evaluators use metrics such as TP, FP, and FN. TP indicates that objects within the bounding box were accurately identified based on their anticipated label. FP implies that objects were mistakenly identified within the bounding box, whereas FN shows that things should have been discovered but were not. Precision, recall, and F1-score are among the measures obtained from confusion metrics.

Precision assesses the ability to make accurate positive predictions. Recall (sensitivity or true positive rate) measures the model's ability to detect all cases that should be positive. The F1-score illustrates the trade-off between precision and recall, demonstrating the balance of these two criteria. Table 3 shows the results of the validation testing of the RetinaNet152 and MobileNetV2 model's detection with video samples.

Table 3 presents the detection results for the RetinaNet152 and MobileNetV2 models based on video samples. The average values for the system performance parameters precision, recall, and F1 score are the same for the RetinaNet152 and MobileNetV2 models. Based on the video validation results, Figures 2 exhibit the classification and detection capabilities of the RetinaNet152 and MobileNetV2 models to detect road damage systems.

Table 3. The performance of video validation

Proposed sytem	Type	TP	FP	FN	Ground truth	Precision	Recall	F1-score
RetinaNet152 model	L00	29	14	1	30	0.67	0.97	0.79
	R02	3	2	1	4	0.60	0.75	0.67
	R03	12	7	1	13	0.63	0.92	0.75
	00	3	1	4	7	0.75	0.43	0.55
					Average	0.66	0.77	0.67
MobileNetV2 model	L00	32	10	2	28	0.76	0.94	0.84
	R02	5	2	1	3	0.71	0.83	0.76
	R03	13	5	2	12	0.72	0.86	78
	00	2	2	4	7	0.5	0.33	0.4
					Average	0.67	0.74	0.69



Figure 2. Prediction results RetinaNet152 and MobileNetV2 models, parameter batch size 8, and Adam optimizer

The implementation testing of the road fault detection system focuses on analyzing the CNN model's performance on NVIDIA Jetson Nano 4 GB used as an edge device [32]. In this research, utilizing the RetinaNet152 [38] and MobileNetV2 models, adjusting hyperparameters such as batch size and an optimizer are built to increase system performance. Performance parameters include average frame per second (FPS) and average inference time. Table 4 shows the results of deploying the RetinaNet-152 and MobileNetV2 models on the edge device with modifications in hyperparameter tuning parameters such as batch size and optimizer.

Table 4. Comparison of inference time RetinaNet152 and MobileNetV2 models

Model	FPS	Inference Time (s)	Model (Mb)
RetinaNet152	0.05	19.48	70
MobileNetV2	3.50	0.28	3.7

Table 4 shows that the performance of the RetinaNet-152 model on the NVIDIA Jetson Nano 4 GB has direct field-testing results with a very low FPS of 0.05. This demonstrates that adjusting the batch size and optimizer hyperparameters can increase system performance (mAP, recall, and F1-score) but also speed up inference time. Inference time with smaller values and model sizes demonstrates that the RetinaNet 152 model performs better at detecting road damage. However, deployment findings on the edge device model RetinaNet152 showed a lower FPS on the Adam optimizer with a batch size of 8 versus 4. Larger batch sizes have the potential to accelerate the training process, resulting in faster inference time. However, big batches require resources, particularly when implemented on edge devices with limited memory. Among the cases, Adam optimizer with batch size 8 produced better parameter values: FPS 0.05, inference time 19.4780 seconds, model size 70 MB.

Based on research conducted with the MobileNetV2 model [34], this research developed the MobileNetV2 model, which was deployed utilizing an NVIDIA Jetson 4 GB. The MobileNetV2 model has an inference time 0.28 seconds faster than RetinaNet152 and an FPS of 3.5, giving it better performance than RetinaNet152. Table 4 demonstrates how the MobileNetV2 model, which has faster inference, may be utilized to detect road damage on edge devices.

The general RetinaNet152 model cannot be employed directly on edge devices; however, methods are required to reduce inference time. This is due to the RetinaNet152 model's far more sophisticated architecture than MobileNetV2. RetinaNet152 contains more layers and parameters to accommodate the need for high precision in object detection tasks involving multiple objects at varying scales and features. This means that RetinaNet152 needs more processing resources to perform this inference. If the RetinaNet152 model is used in research on edge devices, methods such as quantization, pruning, and knowledge distillation must be used to reduce inference time.

3.3. Comparison system performance model CNN

Table 2 shows the system performance results of the RetinaNet152 and MobileNetV2 models with respect to all the minimum loss parameters. The results of mAP of 0.469; AR of 0.410 and F1-score of 0.377 for the RetinaNet152 model parameters batch size 8 and Adam optimizer are obtained. Table 4 shows the efficiency of the RetinaNet152 model with batch size 8 and Adam optimizer in terms of inference time of 19.48 seconds, FPS of 0.05, and model size of 70 MB. The MobileNetV2 model parameters batch size 8 and Adam optimizer yielded the highest F1-score value of 0.481, with mAP 0.606 and AR 0.476, respectively. The MobileNetV2 model with batch size 8 and Adam optimizer is efficient in terms of inference time of 0.28 seconds, FPS of 3.50, and model size of 3.7 MB. The RetinaNet152 and MobileNetV2 models with test results have the best system performance when compared to the findings of another investigation, as shown in Table 5.

Table 5. The system performance on edge device

Previous studies	Model CNN	System performance	Edge device
[16]	MobileNet	Recall 0.71; precision 0.77; inference time 1.5 s.	Smartphone
[32]	MobileNetV2	mAP 0.22	NVIDIA Jetson Nano
[33]	MobileNet V1	Accuracy 0.60; frames per second 1.2 s	Raspberry Pi
[34]	MobileNetV2	mAP 0.086; AR 0.241	NVIDIA Jetson Nano
Proposed system	RetinaNet152	mAP 0.469; AR 0.410; F1-score 0.377; size model 70 MB; inference time 19,4780 second; FPS 0.05	NVIDIA Jetson Nano
Proposed system	MobileNetV2	mAP 0.606; AR 0.476; F1-score 0.481; size model 3.7 MB; inference time 0.28 second; FPS 3.5	NVIDIA Jetson Nano

This study investigates the effects of batch size and the optimizer on CNN models, specifically the RetinaNet152 and MobileNetV2 models, which are deployed on edge devices to detect road defects. Although earlier research investigated the influence of hyperparameter CNN models on system performance, little has been performed to specifically examine the effect of batch size and the optimizer. This result study is compared to the MobileNetV2 model [32], [34] research using learning rate and batchsize hyperparameter tuning. In this study, based on Table 5, the RetinaNet152 model using batchsize and optimizer hyperparameter tuning methods proposed in this study show a proportional improvement in system performance results for mAP and AR. Based on the system performance comparison in Table 5, the

hyperparameter tuning strategy in the MobileNetV2 model improves performance significantly, resulting in good system performance and rapid inference time. This demonstrates that the MobileNetV2 model could potentially be implemented directly on edge devices for road damage detection. The RetinaNet152 model, which employs the batchsize hyperparameter tuning method and the optimizer developed in this paper, achieves a proportional improvement in system performance results for mAP and AR. However, the RetinaNet152 model cannot be implemented directly on edge devices; it requires post-training procedures like quantization and pruning, among others.

However, according to the results of this study, RetinaNet152 models need to improve system performance in several respects when compared to the MobileNetV2 model [16]. The first method is to improve system performance by expanding the dataset by boosting image capture resolution with a high-resolution camera. There is no comparable dataset for road damage research [16]; hence, it presents unique issues. Furthermore, adding many augmenting scenarios will increase system performance on the RetinaNet152 and MobileNetV2 models. In addition, to improve system performance, the RetinaNet152 and MobileNetV2 models should be optimized for hyperparameter selection utilizing grid search, random search, bayesian optimization, and so on. Further research into the deployment of CNN models on edge devices should focus on the structure and size of the models. This is a crucial issue to consider because CNN models with large model structures and sizes demand more processing resources during the inference phase to detect objects. To reduce inference time, a CNN model with a large model structure and size should be employed in edge device research, along with methods such as quantization, pruning, and knowledge distillation [39], [40].

4. CONCLUSION

There is a need for a device as an edge device that recognizes road damage using technology and AI, as a sort of road digitalization creates a system to identify road damage automatically, efficiently, and in real time. The use of CNN models for road damage detection on edge devices requires a trade-off between high accuracy and high efficiency. Currently, edge devices have limited storage. Results comparing the effects of increasing batch size and different optimizers on RetinaNet152 and MobileNetV2 models: i) the Adam optimizer performed better, with a mAP, AR, and F1-score for batch size 8. This means that tweaking the optimizer improves overall model performance; ii) the MobileNetV2 model's hyperparameter tuning technique considerably enhances performance, resulting in fast inference time and good system performance overall. This shows that the MobileNetV2 model might be deployed directly on edge devices to detect road damage; and iii) the RetinaNet152 model, which uses the batchsize hyperparameter tuning approach and the optimizer described in this research, achieves a proportionate gain in system performance for mAP and AR. However, the RetinaNet152 model cannot be deployed directly to edge devices using the Jetson Nano 4 GB. The RetinaNet152 model can be deployed on edge devices; however, a technique for speeding up inference time is essential, as are post-training procedures such as quantization and pruning. Several recommendations are made for further research or implementation. First, consider augmenting and improving the dataset's quality to attain better CNN model training outcomes. Second, system implementation testing should be carried out on a high-end computer server, such as a NVIDIA RTX 3080 with 32 GB of RAM. In addition, consider using the RetinaNet50 model, which has a lighter design and model size than the RetinaNet152. Finally, investigate speeding up the inference time exploring methods like post-training quantization and converting Tensorflow Lite models to TensorRT to improve implementation efficiency.

ACKNOWLEDGEMENTS

The authors would like to thank Politeknik Elektronika Negeri Surabaya and Institut Teknologi Sepuluh Nopember for financial support, laboratory equipment and facilities for this work.

REFERENCES




- [1] I. Widyatmoko, "Digital transformation to improve quality, efficiency and safety in construction of roads incorporating recycled materials," *IOP Conference Series: Earth and Environmental Science*, vol. 599, no. 1, Nov. 2020, doi: 10.1088/1755-1315/599/1/012093.
- [2] V. Hegde, D. Trivedi, A. Alfarrarjeh, A. Deepak, S. Ho Kim, and C. Shahabi, "Yet another deep learning approach for road damage detection using ensemble learning," in *2020 IEEE International Conference on Big Data (Big Data)*, IEEE, Dec. 2020, pp. 5553–5558, doi: 10.1109/BigData50022.2020.9377833.
- [3] A. Pernestål, A. Engholm, M. Bemler, and G. Gidofalvi, "How will digitalization change road freight transport? scenarios tested in sweden," *Sustainability*, vol. 13, no. 1, Dec. 2020, doi: 10.3390/su13010304.
- [4] G. M. Jog, C. Koch, M. Golparvar-Fard, and I. Brilakis, "Pothole properties measurement through visual 2d recognition and 3d reconstruction," in *Computing in Civil Engineering (2012)*, Reston, VA: American Society of Civil Engineers, Jun. 2012, pp. 553–560, doi: 10.1061/9780784412343.0070.

- [5] E. Buza, S. Omanovic, and A. Huseinovi, "Pothole detection with image processing and spectral clustering," in *2nd International Conference on Information Technology and Computer Networks (ITCN '13)*, 2013, pp. 48–53.
- [6] L. Huidrom, L. K. Das, and S. K. Sud, "Method for automated assessment of potholes, cracks and patches from road surface video clips," *Procedia - Social and Behavioral Sciences*, vol. 104, pp. 312–321, Dec. 2013, doi: 10.1016/j.sbspro.2013.11.124.
- [7] M. B. S. G. Naik and V. Nirmalrani, "Detecting potholes using image processing techniques and real-world footage," in *Cognitive Informatics and Soft Computing-Advances in Intelligent Systems and Computing*, Springer, Singapore, 2021, pp. 893–902, doi: 10.1007/978-981-16-1056-1_72.
- [8] R. Sharma, K. Singh, and L. Chand, "Analysis of image processing techniques for road anomalies detection," *International Journal of Emerging Research in Management & Technology*, vol. 5, no. 1, 2016, doi: 10.13140/RG.2.2.15513.54886.
- [9] M. Gao, X. Wang, S. Zhu, and P. Guan, "Detection and segmentation of cement concrete pavement pothole based on image processing technology," *Mathematical Problems in Engineering*, vol. 2020, pp. 1–13, Jan. 2020, doi: 10.1155/2020/1360832.
- [10] M. H. Yousaf, K. Azhar, F. Murtaza, and F. Hussain, "Visual analysis of asphalt pavement for detection and localization of potholes," *Advanced Engineering Informatics*, vol. 38, pp. 527–537, Oct. 2018, doi: 10.1016/j.aei.2018.09.002.
- [11] N.-D. Hoang, "An artificial intelligence method for asphalt pavement pothole detection using least squares support vector machine and neural network with steerable filter-based feature extraction," *Advances in Civil Engineering*, vol. 2018, no. 1, Jan. 2018, doi: 10.1155/2018/7419058.
- [12] H. Song, K. Baek, and Y. Byun, "Pothole detection using machine learning," in *Advanced Science and Technology Letters, SERSC*, Feb. 2018, pp. 151–155, doi: 10.14257/astl.2018.150.35.
- [13] L. Liu *et al.*, "Deep learning for generic object detection: a survey," *International Journal of Computer Vision*, vol. 128, no. 2, pp. 261–318, Feb. 2020, doi: 10.1007/s11263-019-01247-4.
- [14] N.-D. Hoang, T.-C. Huynh, and V.-D. Tran, "Computer vision-based patched and unpatched pothole classification using machine learning approach optimized by forensic-based investigation metaheuristic," *Complexity*, vol. 2021, no. 1, Jan. 2021, doi: 10.1155/2021/3511375.
- [15] C. Chun and S.-K. Ryu, "Road surface damage detection using fully convolutional neural networks and semi-supervised learning," *Sensors*, vol. 19, no. 24, Dec. 2019, doi: 10.3390/s19245501.
- [16] H. Maeda, Y. Sekimoto, T. Seto, T. Kashiyama, and H. Omata, "Road damage detection and classification using deep neural networks with smartphone images," *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, no. 12, pp. 1127–1141, Dec. 2018, doi: 10.1111/mice.12387.
- [17] S. Zhou and W. Song, "Deep learning-based roadway crack classification using laser-scanned range images: a comparative study on hyperparameter selection," *Automation in Construction*, vol. 114, Jun. 2020, doi: 10.1016/j.autcon.2020.103171.
- [18] S. Faghieh-Roohi, S. Hajizadeh, A. Nunez, R. Babuska, and B. De Schutter, "Deep convolutional neural networks for detection of rail surface defects," in *2016 International Joint Conference on Neural Networks (IJCNN)*, IEEE, Jul. 2016, pp. 2584–2589, doi: 10.1109/IJCNN.2016.7727522.
- [19] L. Pauly, H. Peel, S. Luo, D. Hogg, and R. Fuentes, "Deeper networks for pavement crack detection," in *34th International Symposium on Automation and Robotics in Construction*, International Association for Automation and Robotics in Construction, Jul. 2017, pp. 479–485, doi: 10.22260/ISARC2017/0066.
- [20] A. Zhang *et al.*, "Deep learning-based fully automated pavement crack detection on 3d asphalt surfaces with an improved cracknet," *Journal of Computing in Civil Engineering*, vol. 32, no. 5, Sep. 2018, doi: 10.1061/(ASCE)CP.1943-5487.0000775.
- [21] Y. Cha, W. Choi, and O. Büyüköztürk, "Deep learning-based crack damage detection using convolutional neural networks," *Computer-Aided Civil and Infrastructure Engineering*, vol. 32, no. 5, pp. 361–378, May 2017, doi: 10.1111/mice.12263.
- [22] D. Wang, Z. Liu, X. Gu, W. Wu, Y. Chen, and L. Wang, "Automatic detection of pothole distress in asphalt pavement using improved convolutional neural networks," *Remote Sensing*, vol. 14, no. 16, Aug. 2022, doi: 10.3390/rs14163892.
- [23] M. Faramarzi, "Road damage detection and classification using deep neural networks (yolov4) with smartphone images," *SSRN Electronic Journal*, 2020, doi: 10.2139/ssrn.3627382.
- [24] M. A. Benallal and M. S. Tayeb, "An image-based convolutional neural network system for road defects detection," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 12, no. 2, Jun. 2023, doi: 10.11591/ijai.v12.i2.pp577-584.
- [25] R. Ishimwe, P. Iradukunda, and J. B. Kwizera, "Real-time road damage detection using deep convolutional neural networks and a smartphone: project report," *Carnegie Mellon University*, pp. 1-5, 2021, doi: 10.13140/RG.2.2.26696.44801.
- [26] J. Dhameeshkar, V. S. Dhakshana, S. A. Aniruthan, R. Karthika, and L. Parameswaran, "Deep learning based detection of potholes in indian roads using yolo," in *2020 International Conference on Inventive Computation Technologies (ICICT)*, IEEE, Feb. 2020, pp. 381–385, doi: 10.1109/ICICT48043.2020.9112424.
- [27] K. R. Ahmed, "Smart pothole detection using deep learning based on dilated convolution," *Sensors*, vol. 21, no. 24, Dec. 2021, doi: 10.3390/s21248406.
- [28] E. M. Dogo, O. J. Afolabi, and B. Twala, "On the relative impact of optimizers on convolutional neural networks with varying depth and width for image classification," *Applied Sciences*, vol. 12, no. 23, Nov. 2022, doi: 10.3390/app122311976.
- [29] H. Samma, S. A. Suandi, N. A. Ismail, S. Sulaiman, and L. L. Ping, "Evolving pre-trained cnn using two-layers optimizer for road damage detection from drone images," *IEEE Access*, vol. 9, pp. 158215–158226, 2021, doi: 10.1109/ACCESS.2021.3131231.
- [30] Y. Wang, Z. Xiao, and G. Cao, "A convolutional neural network method based on adam optimizer with power-exponential learning rate for bearing fault diagnosis," *Journal of Vibroengineering*, vol. 24, no. 4, pp. 666–678, Jun. 2022, doi: 10.21595/jve.2022.22271.
- [31] Vidushi, M. Agarwal, A. Rajak, and A. K. Shrivastava, "Assessment of optimizers impact on image recognition with convolutional neural network to adversarial datasets," *Journal of Physics: Conference Series*, vol. 1998, no. 1, Aug. 2021, doi: 10.1088/1742-6596/1998/1/012008.
- [32] I. D. Pratama, H. Mahmudah, and R. W. Sudiby, "Design and implementation of real-time pothole detection using convolutional neural network for iot smart environment," in *2021 International Electronics Symposium (IES)*, IEEE, Sep. 2021, pp. 675–679, doi: 10.1109/IES53407.2021.9594038.
- [33] M. Al Qurishee, "Low-cost deep learning uav and raspberry pi solution to real time pavement condition assessment," *M.Sc. Thesis*, Department of Science and Engineering, University of Tennessee, Chattanooga, US, 2019.
- [34] Z. S. Hernanda, H. Mahmudah, and R. W. Sudiby, "CNN-based hyperparameter optimization approach for road pothole and crack detection systems," in *2022 IEEE World AI IoT Congress (AIoT)*, IEEE, Jun. 2022, pp. 538–543, doi: 10.1109/AIoT54504.2022.9817316.
- [35] A. C. Aqsa, H. Mahmudah, and R. W. Sudiby, "Detection and classification of road damage using cnn with hyperparameter optimization," in *2022 6th International Conference on Informatics and Computational Sciences (ICICoS)*, IEEE, Sep. 2022, pp. 101–104, doi: 10.1109/ICICoS56336.2022.9930607.
- [36] L. Ale, N. Zhang, and L. Li, "Road damage detection using retinanet," in *2018 IEEE International Conference on Big Data (Big Data)*, IEEE, Dec. 2018, pp. 5197–5200, doi: 10.1109/BigData.2018.8622025.




- [37] Z. Xue *et al.*, "Large-batch optimization for dense visual predictions," *arXiv-Computer Science*, pp. 1-23, Oct. 2022.
- [38] H. Mahmudah, S. Arifin, A. S. Aisjah, and C. A. Prastyanto, "Optimizers impact on retinanet model for detecting road damage on edge device," in *2024 IEEE International Conference on Artificial Intelligence and Mechatronics Systems (AIMS)*, IEEE, Feb. 2024, pp. 1–6, doi: 10.1109/AIMS61812.2024.10512864.
- [39] H. Mahmudah, S. Arifin, and A. S. Aisyah, "A survey of the optimization system road damage detection on cnn model for edge device," in *2023 International Conference on Information Technology and Computing (ICITCOM)*, IEEE, Dec. 2023, pp. 323–328, doi: 10.1109/ICITCOM60176.2023.10442409.
- [40] M. M. H. Shuvo, S. K. Islam, J. Cheng, and B. I. Morshed, "Efficient acceleration of deep learning inference on resource-constrained edge devices: a review," *Proceedings of the IEEE*, vol. 111, no. 1, pp. 42–91, Jan. 2023, doi: 10.1109/JPROC.2022.3226481.

BIOGRAPHIES OF AUTHORS






Haniah Mahmudah    completed bachelor's degree in Department of Engineering Physics, Institut Teknologi Sepuluh Nopember and master's degree in Department of Electrical Engineering, Institut Teknologi Sepuluh Nopember. She is currently working as a lecturer in Department of Electrical Engineering, Politeknik Elektronika Negeri Surabaya. She is also active in conducting research especially in signal processing, wire and wireless communication, internet of things, applied of machine learning. She can be contacted at email: haniah@pens.ac.id.






Aulia Siti Aisjah    received doctoral degree Institut Teknologi Sepuluh Nopember. She is currently working as a lecturer in Department of Engineering Physics, Institut Teknologi Sepuluh Nopember. Her research interests are in applied of machine learning in industrial, marine engineering and instrumentation. She can be contacted at email: auliasa@its.ac.id.



Syamsul Arifin    is currently working as a lecturer in Department of Engineering Physics, Institut Teknologi Sepuluh Nopember. His research interests are in instrumentation and applied of machine learning. He can be contacted at email: syamsul.arifin@its.ac.id.



Catur Arif Prastyanto    completed bachelor's degree, master's degree and doctor's degree in Department of Civil Engineering, Institut Teknologi Sepuluh Nopember. He is currently working as a lecturer in Department of Civil Engineering, Institut Teknologi Sepuluh Nopember. Her research interests are highways and transportation. He can be contacted at email: catur_ap@its.ac.id.