IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 15, No. 3, June 2026, pp. 2811~2825
ISSN: 2252-8938, DOI: 10.11591/ijai.v15.i3.pp2811-2825

Performance comparison of deep learning models for concrete crack detection on mobile devices

Sarapee Chunkaew[1], Somporn Ruang-On[1], Prawit Nuengmatcha[1], Kritaphat Songsri-in[2]
[1]Department of Creative Innovation in Science and Technology, Faculty of Science and Technology, Nakhon Si Thammarat Rajabhat University, Mueang Nakhon Si Thammarat, Thailand
[2]Department of Computer Science, Faculty of Science and Technology, Nakhon Si Thammarat Rajabhat University, Mueang Nakhon Si Thammarat, Thailand

Article Info

Article history:

Received Feb 22, 2025
Revised Feb 24, 2026
Accepted May 12, 2026

Keywords:

Concrete crack detection
Convolutional neural network
MobileNet
Structural damage assessment
Transfer learning

ABSTRACT

Concrete crack detection is essential for structural maintenance, yet traditional manual inspection methods are time-consuming and require specialized expertise. While deep learning offers promising solutions, existing models often demand high computational resources unsuitable for mobile deployment. This research evaluates three convolutional neural network (CNN) architectures, namely mobile network (MobileNet), visual geometry group-16 (VGG-16), and residual network-50 (ResNet-50), to identify an optimal model for practical mobile-based crack detection. A dataset of 1,634 images was collected from online databases and field documentation, categorized into 10 classes across three severity levels: i) severe cracks requiring urgent repair (30%); ii) cracks requiring monitoring (40%); and iii) minor cracks (30%). The models were trained using standardized parameters with 224×224-pixel RGB input, rectified linear unit (ReLU) activation, and softmax classification. Systematic parameter optimization was conducted across epochs, learning rate, dropout rate, and optimizer selection, with stochastic gradient descent (SGD) identified as the optimal optimizer. Experimental results demonstrate that MobileNet achieves the best performance with 80% accuracy and a compact model size of 13.1 megabytes. This study concludes that MobileNet provides an optimal balance between detection accuracy and computational efficiency, enabling practical field deployment for automated concrete crack detection, with expert verification recommended for critical structural assessments.

Corresponding Author:

Somporn Ruang-On
Department of Creative Innovation in Science and Technology, Faculty of Science and Technology
Nakhon Si Thammarat Rajabhat University
1, Tambon Tha Ngio, Mueang Nakhon Si Thammarat, Nakhon Si Thammarat 80280, Thailand
Email: somporn_rua@nstru.ac.th

1. INTRODUCTION

Concrete structures are essential elements critical to the safety of building occupants and construction. However, these structures tend to deteriorate over time, with cracking, which is one of the most common forms of damage, varying in severity. These include severe cracks requiring urgent repair, e.g., diagonal wall cracks indicating structural problems, vertical beam cracks affecting building stability, and rust-exposed cracks showing reinforcement deterioration. Cracks requiring monitoring include column cracks, roof cracks that may cause leakage, and road cracks that could develop into potholes. Minor cracks refer to crazing and slight material degradation. Recent advances in automated detection systems have enabled more efficient identification and classification of these various crack types [1], [2].

In general, structural cracks wider than 0.3 millimeters can lead to reinforcement deterioration, strength loss, and potentially structural collapse. Deep learning-based crack detection systems have demonstrated effectiveness in identifying cracks and predicting damage severity in reinforced concrete structures [3], [4]. Early and regular detection and assessment of concrete cracks are significant for maintaining structural integrity, allowing for identification and repair before cracks progress to severe structural damage while reducing long-term maintenance costs.

Traditional concrete crack inspection methods include visual inspection [5], crack width gauge measurements, microscopic examination, ultrasonic flaw detection, and high-resolution imaging techniques. However, these approaches have several limitations, such as being time-consuming, requiring expert involvement, and accuracy depending on inspector experience, along with efficient but cost-prohibitive equipment [5], [6]. Automated crack-detecting systems have been made possible by progress in image processing and artificial intelligence. In particular, deep learning [7]–[9] has demonstrated remarkable potential in detecting, classifying, and assessing the severity level of cracks from images with high accuracy. Convolutional neural networks (CNNs) [10], [11] are popular deep learning architectures for image analysis, enhanced by AI-assisted automation systems for practical deployment. These networks are capable of automatically learning crack characteristics through transfer learning approaches [12], [13], which leverage pre-trained models to improve performance even with limited training data. Currently, there is significant interest in developing high-performance, lightweight models for mobile devices. Models such as mobile network (MobileNet), visual geometry group-16 (VGG-16), and residual network-50 (ResNet-50) [14], [15] are designed to operate efficiently on resource-constrained devices.

However, deploying these models on mobile devices remains challenging due to processing resources, memory, and power limitations [16]. While deep learning has shown effectiveness in various concrete crack detection applications [17], [18], comparative studies evaluating model performance on mobile devices remain limited [19], particularly regarding the trade-offs between detection accuracy, resource utilization, and processing speed [20]. Recent comprehensive reviews of crack detection technologies have highlighted the growing importance of automated systems for infrastructure inspection [21].

Advanced techniques combining U-Net and fully convolutional networks have shown enhanced performance in classifying concrete damage under various environmental conditions [22]. Therefore, this research focuses on comparing the performance of three deep learning models, namely MobileNet, VGG-16, and ResNet-50, for concrete crack detection on mobile devices [23], [24]. These CNN architectures have demonstrated effectiveness in automated crack detection for structural health monitoring applications [3].

The novelty of this study lies in three key contributions:

i) A fine-grained classification scheme comprising 10 crack categories across three severity levels, enabling more precise structural damage assessment than conventional binary detection.
ii) A systematic hyperparameter optimization framework covering epoch selection, learning rate, dropout rate, and optimizer selection, tailored specifically for mobile-constrained deployment.
iii) End-to-end mobile deployment via TensorFlow Lite conversion and Flutter-based application development, demonstrating real-world feasibility for field inspection without specialized equipment.

The study considers not only detection accuracy but also crucial factors for practical implementation, such as model size [25] and processing speed. Additionally, the research includes hyperparameter optimization to find the optimal balance between performance and resource utilization [26]. This comparative study of deep learning models for mobile-based concrete crack detection has significant practical implications for civil engineering and structural maintenance. By identifying the most suitable CNN architecture for mobile applications [27], this research contributes to the development of more accessible, efficient, and reliable crack detection tools. These tools can be widely utilized by construction professionals, building inspectors, and maintenance personnel [28], as well as homeowners interested in examining cracks in their own properties.

2. METHOD

This research employs an experimental approach focused on comparing deep learning model performance for concrete crack detection on mobile devices. Quantitative research methodology was used for analysis and evaluation. The research process consists of nine main stages: i) concrete crack image dataset collection and preparation, ii) dataset partitioning, iii) deep learning model preparation, iv) model adaptation for crack detection, v) model training, vi) testing and performance evaluation, vii) result analysis and comparison, viii) model deployment testing on mobile devices, and ix) performance and resource utilization analysis. The details of each stage are as follows.

2.1. Dataset collection and preparation

In the data collection and preparation phase, 1,634 crack images were gathered from two sources: online databases and researcher-captured photographs. The images were classified according to crack location and cause into 10 classes: i) diagonal wall center cracks, ii) material deterioration cracks, iii) column cracks, iv) road cracks, v) roof cracks, vi) rust-tinted reinforcement cracks, vii) hairline cracks, viii) vertical beam center cracks, ix) general cracks, and x) non-crack images. This detailed image categorization enables the model to effectively learn the distinctive characteristics of each crack type. Although 1,634 images are relatively small compared to large-scale datasets, the use of transfer learning addresses this constraint. All three models were initialized with weights pre-trained on ImageNet, which contains over 1.2 million images across 1,000 categories. As a result, the models could apply previously acquired visual knowledge to the crack classification task, reducing the need for a large domain-specific training set.

2.2. Dataset partitioning and organization

To ensure efficient model training and evaluation, the dataset management was systematically organized. All collected images were first uploaded and structured in category-specific folders on Google Drive facilitates streamlined data access and management. Following standard machine learning practices, the complete dataset was partitioned into three distinct sets using a conventional split ratio: 60% of the images were allocated to the training set for model learning, 20% to the validation set for parameter tuning and optimization, and the remaining 20% to the test set for final performance evaluation. Each set contained a balanced distribution of images from all ten crack categories to ensure representative sampling. The 60/20/20 split ratio was selected following standard practice in deep learning research, ensuring sufficient training data while keeping the validation and test sets independent. Although k-fold cross-validation could improve result reliability, it was not applied here due to the high computational cost of training three large pre-trained models across multiple folds. Figure 1 illustrates this organizational structure, showing both the folder hierarchy and representative sample images from all ten crack categories used in this study, including eight distinct crack types, general objects, and crack-free surfaces.

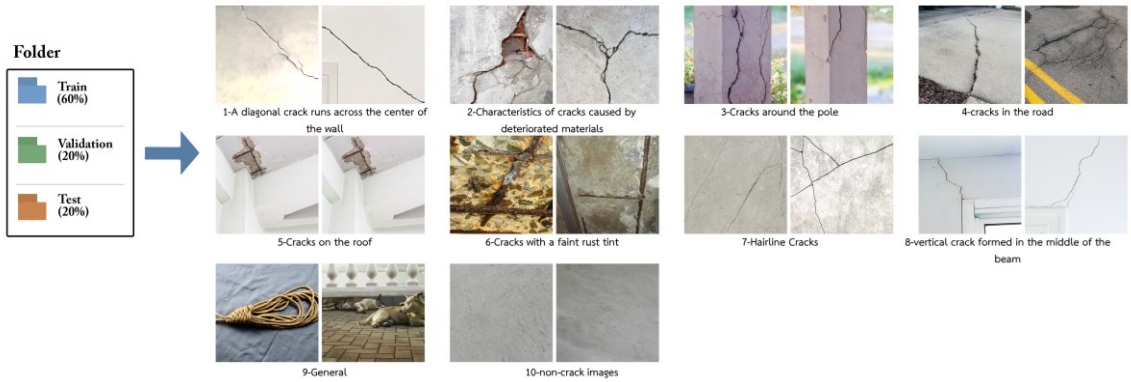

Figure 1. Dataset organization and folder hierarchy by crack categories

2.3. Deep learning model preparation

To compare model performance and size, three high-performance deep learning models were selected: MobileNet, VGG-16, and ResNet-50. Initial hyperparameters were standardized across models as follows: input layer =224×224 (RGB), convolutional 2D layers with rectified linear unit (ReLU) activation function, dropout rate =0.3, dense layer =10 with softmax activation function, Adam optimizer, learning rate =0.01, categorical cross-entropy loss function, and epochs =15. This selection of models allows detailed evaluation of different architectural performances in concrete crack detection.

2.4. Model optimization for crack detection

This process involves finding optimal parameter configurations for each model to maximize performance and prevent overfitting. Hyperparameter optimization is particularly critical for deep learning applications on mobile devices, as suboptimal configurations may result in either underfitting, where the model fails to capture essential crack features, or overfitting, where the model becomes overly specialized to training data and loses generalization capability. The following parameters were systematically tuned through controlled experiments to identify the configuration that achieves the best balance between detection accuracy and computational efficiency on resource-constrained mobile devices.

2.4.1. Optimizer comparison

The selection of an appropriate optimizer is crucial for deep learning model performance. Optimizers are algorithms that adjust neural network parameters during training to minimize the loss function. This study compared three widely used optimizers: stochastic gradient descent (SGD), Adam, and AdamW, each with distinct characteristics affecting convergence speed and final model performance.

SGD updates parameters using random subsets of training data in each iteration. Though its convergence is slower than adaptive methods, it tends to generalize well due to the natural noise in its gradient estimates. Adam computes individual learning rates for each parameter based on first and second moment estimates, which works well for data with sparse gradients or noisy data. AdamW builds on Adam by separating weight decay from the parameter update step, which helps reduce overfitting more effectively than standard Adam.

For this comparative study, all three optimizers were evaluated under identical conditions with a learning rate of 0.03, a dropout rate of 0.3, and 20 training epochs. The performance evaluation focused on training stability, convergence speed, and final accuracy on the validation set. This systematic comparison aimed to identify the optimizer that provides the optimal balance between training efficiency and model performance for mobile-based concrete crack detection.

2.4.2. Learning rate

The learning rate is a critical hyperparameter that controls the magnitude of parameter updates during model training. It determines how quickly the model adapts its weights to minimize the loss function, directly influencing both the convergence speed and the stability of the training process. In this study, learning rates ranging from 0.01 to 0.1 with an interval of 0.01 were systematically evaluated to determine the optimal value for each deep learning model.

Higher learning rates usually achieve faster convergence. However, they risk overshooting the optimal solution, potentially causing training instability and degraded model performance. Conversely, an excessively low learning rate leads to slow convergence and may become trapped in local minima, requiring significantly more epochs to achieve satisfactory results. Therefore, selecting an appropriate learning rate is essential for achieving reliable model performance in crack detection tasks.

This study conducted learning rate optimization experiments while keeping all other hyperparameters constant, including an input layer size of 224×224 (RGB), a dropout rate of 0.3, the Adam optimizer, a dense layer with 10 nodes using softmax activation, categorical cross-entropy loss, and 20 training epochs. This controlled setup ensured that observed variations in accuracy and loss metrics were directly attributable to changes in the learning rate. All other factors remained unchanged to isolate the learning rate's effect.

2.4.3. Dropout rate

The dropout rate is a regularization technique applied during training to prevent overfitting by randomly deactivating a proportion of neural network nodes in each training iteration. This mechanism encourages the network to learn more general features rather than memorizing specific patterns in the training data. In this study, dropout rates ranging from 0.1 to 0.9 were tested to find the value that best balances model complexity and generalization performance.

A dropout rate that is too low allows the model to memorize training samples, resulting in poor performance on unseen data. On the other hand, setting the dropout rate too high limits the model's ability to learn, causing underfitting and reduced accuracy. Selecting an appropriate dropout value, therefore, requires careful tuning to ensure the model learns meaningful crack features without becoming too specialized to the training set.

The dropout rate optimization was performed with all other hyperparameters held constant, including an input layer size of 224×224 (RGB), Conv2D layers with ReLU activation, a dense layer with 10 nodes and softmax activation, the Adam optimizer, a learning rate of 0.03, categorical cross-entropy loss, and 20 training epochs. This systematic approach allowed precise identification of the dropout rate that yields the best trade-off between training accuracy and generalization performance. The evaluation focused on performance across the ten crack categories.

2.4.4. Epoch selection

The number of training epochs determines how many complete passes the model makes through the entire training dataset during the learning phase. Each epoch allows the model to update its internal parameters based on the accumulated gradient information, progressively improving its ability to classify concrete crack images. In this study, training epochs of 5, 10, 15, 20, 30, and 50 were evaluated to identify the optimal number of iterations for each model. An insufficient number of training epochs may result in an

undertrained model that has not yet converged to a stable solution, leading to lower accuracy on test data. On the other hand, excessive training epochs can lead to overfitting, where the model becomes highly specialized to the training data patterns and loses its ability to generalize to new, unseen images. Monitoring both training and validation performance across epochs is essential to identify the point at which further training no longer improves or begins to degrade generalization performance.

The epoch optimization experiments were conducted under identical conditions for all three models, with fixed hyperparameters including an input layer size of 224×224 (RGB), Conv2D layers with ReLU activation, a dropout rate of 0.3, a dense layer with 10 nodes and softmax activation, the Adam optimizer, a learning rate of 0.01, and categorical cross-entropy loss. This standardized protocol enabled direct comparison of convergence behavior across different training durations. The task focused on concrete crack detection.

2.5. Model training

The three models, namely MobileNet, VGG-16, and ResNet-50, were trained on the prepared dataset. The data was split into training (60%), testing (20%), and validation (20%) sets, containing 980, 327, and 327 images, respectively. Each set maintained a balanced distribution across all ten crack categories to ensure representative model training. During the training phase, each model progressively learned discriminative features from the training set while the validation set was used to monitor generalization performance and tune hyperparameters. The training process utilized a categorical cross-entropy loss function to measure the discrepancy between predicted and actual crack classifications, with the Adam optimizer employed to efficiently update model weights. All three models were trained under identical conditions to facilitate a fair and systematic comparison of their respective performance characteristics.

The training procedure was repeated for multiple configurations as part of the hyperparameter optimization process described in sub-section 2.4. Performance metrics, including accuracy and loss values, were recorded at each epoch to enable detailed analysis of training dynamics, convergence behavior, and potential overfitting patterns for each of the three deep learning architectures. This systematic evaluation approach ensured reliable identification of the optimal model configuration for mobile-based crack detection.

2.6. Testing and performance evaluation

After the training and validation phases were completed, the final performance of each deep learning architecture was evaluated using an independent held-out test set consisting of 327 images. This test set was strictly excluded from all stages of model development, including training, validation, and hyperparameter optimization, in order to provide an unbiased estimate of each model's generalization capability. By evaluating the trained models on previously unseen data, the testing procedure assessed their practical reliability for mobile-based concrete crack classification. Model performance was assessed using classification accuracy and categorical cross-entropy loss. Classification accuracy measured the proportion of correctly predicted samples across the ten crack categories, providing a direct indication of overall recognition performance. Categorical cross-entropy loss evaluated the discrepancy between the predicted probability distributions and the corresponding ground-truth labels, reflecting the quality and confidence of the model predictions. Together, these metrics provide a clear basis for comparing the predictive effectiveness of the evaluated architectures.

In addition to classification performance, model file size was recorded as a deployment-oriented criterion. Since the intended application involves mobile-based concrete crack detection, model compactness is an important practical consideration. A model must not only achieve reliable classification performance but also remain suitable for storage and deployment on resource-constrained mobile devices. Therefore, the evaluation considered accuracy, loss, and model size to support a balanced comparison among the three deep learning architectures. While additional diagnostic analyses may be useful in future extended studies, the present evaluation focused on these criteria because they directly reflect the study's objective of comparing predictive performance and deployment suitability for mobile-based crack detection.

2.7. Result analysis and comparison

This research conducted the analysis and comparison of results in two main aspects to provide a complete assessment of model suitability for mobile-based concrete crack detection. The first aspect focused on comparing the initial performance of the three pre-trained models, namely MobileNet, VGG-16, and ResNet-50, under identical experimental conditions. The second aspect involved detailed fine-tuning optimization of the selected best-performing model to maximize its detection capabilities. The comparative analysis framework evaluated multiple performance dimensions, including classification accuracy, loss convergence behavior, model file size, and training efficiency. These metrics were selected to capture both the predictive capability and the practical deployment requirements of each model architecture. Statistical analysis of the results across different experimental configurations provided reliable evidence for model selection decisions.

The final analysis synthesized findings from both the initial model comparison and the hyperparameter optimization experiments to identify the optimal configuration for concrete crack detection on mobile devices. This integrated approach ensured the selected model achieved superior classification performance while meeting practical mobile deployment constraints. Those constraints included acceptable file size, processing speed, and resource utilization.

2.7.1. Transfer learning-based model comparison

The performance of the three models was compared considering key factors: accuracy, loss values, model size, and training time. Transfer learning was used instead of training complex deep learning models from scratch with randomly initialized weights, which would require extensive data and computational resources. This approach uses pre-trained models' knowledge from similar tasks, reducing training time for large models while maintaining effectiveness.

2.7.2. MobileNet fine-tuning optimization

The optimization process involved systematic adjustment of four key parameters. The number of epochs was varied across six levels, 5, 10, 15, 20, 30, and 50, to determine the optimal training duration. The learning rate was tested within a range of 0.01 to 0.1, while the dropout rate was explored from 0.1 to 0.9 to evaluate regularization effectiveness. Additionally, three optimizers, namely SGD, Adam, and AdamW, were compared to identify the most suitable algorithm for the MobileNet model.

2.8. Mobile device implementation testing

MobileNet model was converted to TensorFlow Lite (.tflite) format and deployed on an Android mobile device for field testing. The mobile application was developed using Flutter framework, integrating the converted model with a camera interface that captures real-time images for crack classification. Test results revealed performance variations compared to laboratory testing due to external factors such as camera quality, lighting conditions, and color variations. To enhance real-world accuracy, the dataset was expanded, particularly in the general crack category, and additional augmentation techniques were applied during training to improve model robustness under diverse environmental conditions.

2.9. Performance and resource analysis

Based on comprehensive model testing and hyperparameter optimization results, MobileNet was selected as the final model due to its superior balance between classification accuracy and computational efficiency. The final configuration incorporated the optimized parameters identified through systematic experimentation: input layer =224×224 (RGB), Conv2D (activation = ReLU), Dense (10, activation = softmax), SGD optimizer, learning rate =0.03, dropout rate =0.3, categorical cross-entropy loss, and 20 epochs. The optimized model achieved 80% accuracy with a training loss of 0.73 and a compact size of 13.1 MB, and when integrated into the Flutter-based classification application, the total application size was 110 MB, demonstrating practical feasibility for mobile-based concrete crack detection.

3. RESULTS AND DISCUSSION

This study compares deep learning model performance for crack detection on mobile devices. The research findings comprise four main sections: i) pre-trained models' comparison, ii) hyperparameter optimization, iii) mobile device performance evaluation, and iv) overall model performance comparison. The details are as follows.

3.1. Pre-trained models' comparison

This research employed transfer learning techniques [12], [13] using ImageNet weights as initialization points to reduce model training time. MobileNet, VGG-16, and ResNet-50 models were compared using identical initial parameters: input layer size of 224×224 (RGB), Conv2D layers (activation = ReLU), dropout rate =0.3, dense layer (10, activation = softmax), Adam optimizer, learning rate =0.01, cross-entropy loss function, and 15 epochs. The model adaptation process for all three models, namely MobileNet, VGG-16, and ResNet-50, began with weights pre-trained on the ImageNet dataset. These parameters were then fine-tuned for the specific task of concrete crack detection. After optimization, each model's performance was evaluated using the test set. The performance results of the optimized MobileNet, VGG-16, and ResNet-50 models are illustrated in Figure 2, which presents four performance metric graphs comparing the three models across training epochs. Figure 2(a) and Figure 2(b) display the training accuracy and training loss curves, while Figure 2(c) and Figure 2(d) present the validation accuracy and validation loss curves to evaluate optimization effectiveness and potential overfitting for each model architecture.

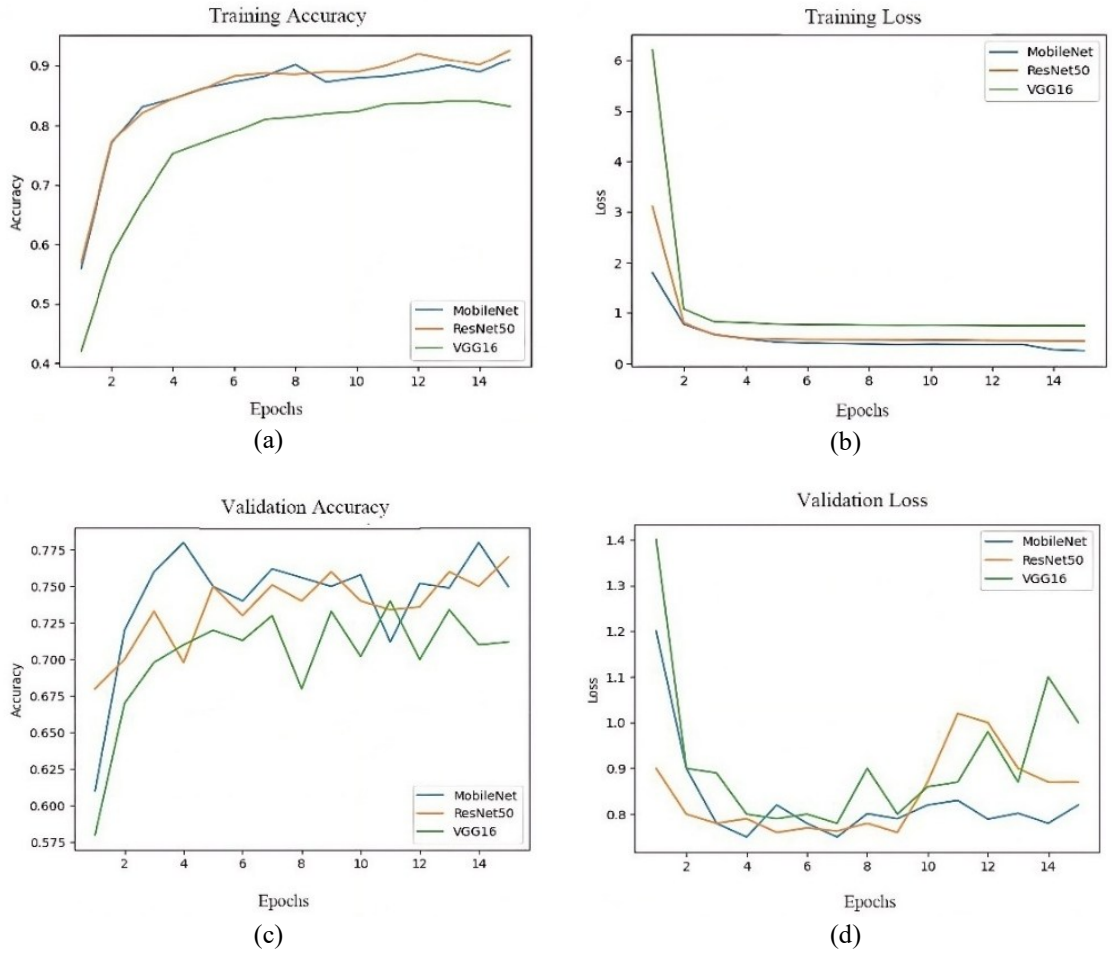

Figure 2. Model performance metrics of (a) training accuracy, (b) training loss, (c) validation accuracy, and (d) validation loss

The performance results of the three models are presented as follows. Figure 2(a) presents the training accuracy curves of MobileNet, VGG-16, and ResNet-50 over 15 epochs, in which MobileNet demonstrates the steepest initial convergence and reaches the highest training accuracy at approximately 0.81. Figure 2(b) shows the corresponding training loss curves, where MobileNet and VGG-16 maintain comparable loss values around 0.89, while ResNet-50 exhibits a slightly higher loss of 0.91. Figure 2(c) illustrates the validation accuracy trends, revealing that MobileNet achieves the most stable validation accuracy at approximately 0.77 throughout training, consistently outperforming the other two models. Finally, Figure 2(d) displays the validation loss progression, in which MobileNet maintains the lowest and most stable validation loss at approximately 0.80, indicating effective learning without overfitting compared to VGG-16 and ResNet-50, as summarized in Table 1.

Table 1. Performance comparison of MobileNet, VGG-16, and ResNet-50 models using test set data

Models	Accuracy	Loss	Epochs	Size (MB)
MobileNet	0.81	0.89	15	13.1
VGG-16	0.78	0.89	15	57.9
ResNet-50	0.80	0.91	15	97.1

Table 1 shows that MobileNet achieved the highest accuracy at 81%, followed by ResNet-50 at 80% and VGG-16 at 78%. MobileNet is the smallest model at 13.1 MB, compared to VGG-16 at 57.9 MB and ResNet-50 at 97.1 MB. Considering both accuracy and model file size, MobileNet was selected as the choice for mobile device implementation, offering the best balance between performance and resource efficiency.

3.2. MobileNet fine-tuning results

The fine-tuning process adapted the pre-trained model for crack detection. This study experimented with adjusting epochs, learning rate, dropout rate, and optimizer parameters. The results are summarized as follows.

3.2.1. Epoch optimization results

Training iterations were varied from 5 to 50 epochs while keeping other parameters constant: input layer =224×224×3, Conv2D with ReLU activation, dropout rate =0.3, dense layer with 10 nodes and softmax activation, Adam optimizer with learning rate =0.01, and categorical cross-entropy loss. Figure 3 shows epoch optimization results with four sub-graphs: training accuracy (Figure 3(a)), training loss (Figure 3(b)), validation accuracy (Figure 3(c)), and validation loss (Figure 3(d)). These plots reveal convergence patterns and help identify the epoch count that yields optimal generalization performance.

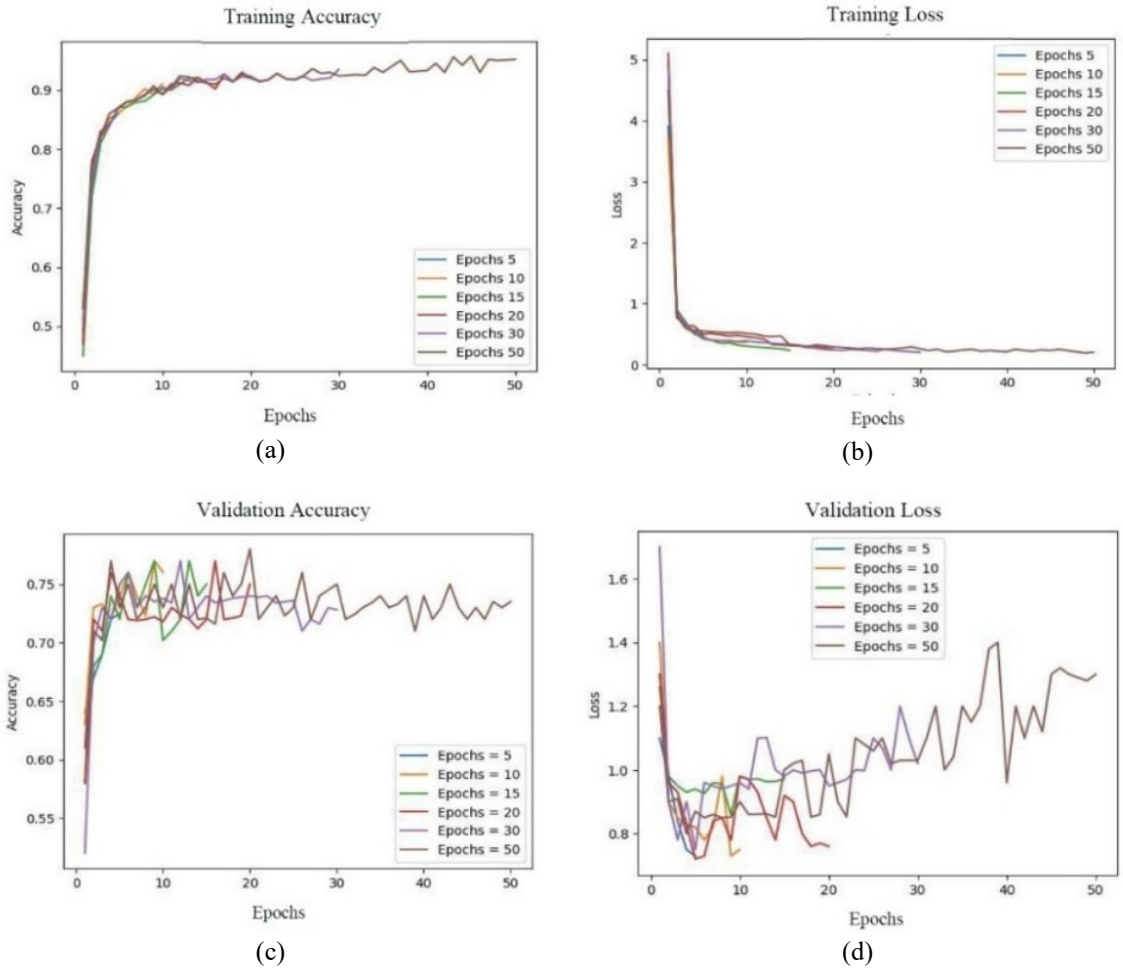

Figure 3. MobileNet epoch optimization results of (a) training accuracy, (b) training loss, (c) validation accuracy, and (d) validation loss

The epoch optimization results for MobileNet are presented in Figure 3. Figure 3(a) shows the training accuracy at different epoch settings (5, 10, 15, 20, 30, and 50), in which the highest training accuracy of 0.81 is achieved at 20 epochs. Figure 3(b) displays the training loss values across the same epoch configurations, where the loss increases from 0.75 at 5 epochs to 0.79 at 20 epochs, before rising sharply to 1.29 at 50 epochs. Figure 3(c) illustrates the validation accuracy trends, revealing that performance stabilizes at approximately 0.73 around 20 epochs and begins to fluctuate at higher epoch counts. Figure 3(d) presents the validation loss progression, demonstrating that the lowest validation loss of approximately 0.80 is maintained at 20 epochs, with a notable increase beyond 30 epochs indicating the onset of overfitting, as summarized in Table 2.

Table 2. Epoch optimization results using test set data

Round	Epochs	Accuracy	Loss
1	5	0.77	0.75
2	10	0.78	0.83
3	15	0.77	0.93
4	20	0.81	0.79
5	30	0.80	1.03
6	50	0.79	1.29

The experimental results demonstrate that while increasing training epochs can improve model accuracy, excessive iterations eventually lead to overfitting. The optimal configuration was identified at 20 epochs, achieving the highest accuracy of 0.81 with a stable validation loss of 0.79. Consequently, this study indicates that model optimization requires a balanced consideration of multiple factors beyond simply increasing training iterations, including computational efficiency, dataset characteristics, and the inherent trade-off between training duration and generalization performance.

3.2.2. Learning rate optimization results

The learning rate was adjusted while maintaining other parameters constant: input layer =224×224×3, Conv2D (activation = ReLU), dense layer (10, activation = softmax), Adam optimizer, dropout rate =0.3, categorical cross-entropy loss, and 20 epochs. Figure 4 presents the MobileNet performance results under varying learning rates, with each sub-graph (Figures 4(a)–4(d)) capturing a distinct performance dimension. The plots reveal how different learning rate values influence convergence stability, training speed, and the trade-off between model complexity and generalization capability.

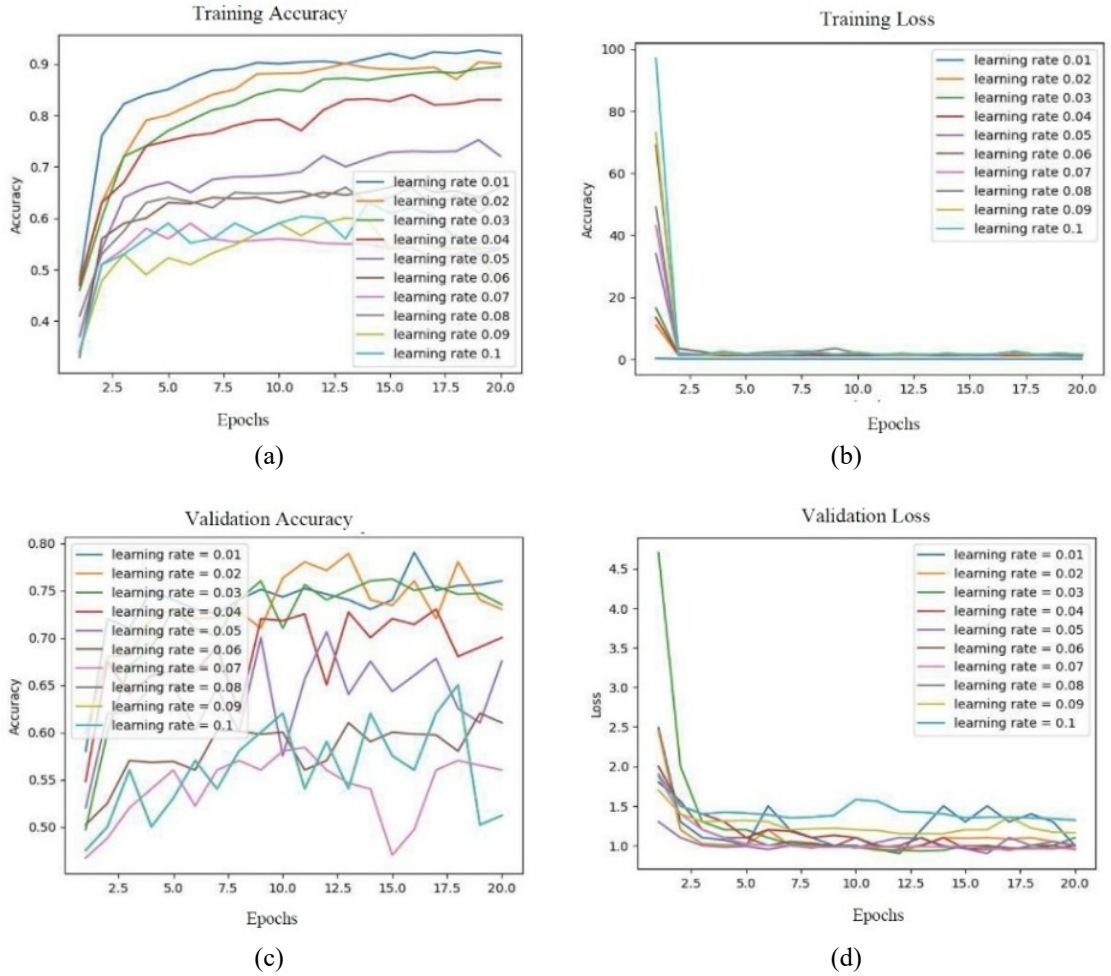

Figure 4. MobileNet performance comparison with learning rate optimization of (a) training accuracy, (b) training loss, (c) validation accuracy, and (d) validation loss

The learning rate optimization results are presented as follows. Figure 4(a) presents the training accuracy across learning rates ranging from 0.01 to 0.10, in which learning rates of 0.01 to 0.03 achieve a high training accuracy of approximately 0.90 to 0.91 at epoch 20, while learning rates above 0.05 result in significantly lower accuracy. Figure 4(b) displays the corresponding training loss values, where learning rates of 0.01 to 0.04 maintain a low training loss near zero after convergence. Figure 4(c) illustrates the validation accuracy trends, revealing that learning rates of 0.01 to 0.03 achieve a validation accuracy of approximately 0.75 to 0.78 with moderate fluctuations. Figure 4(d) presents the validation loss progression, demonstrating stable values around 1.0 to 1.5. Finally, the test set evaluation in Table 3 confirms that learning rates of 0.02 and 0.03 achieve the best performance, with an accuracy of 0.78 to 0.79 and a loss of 0.87 to 0.91; meanwhile, a learning rate of 0.04 shows decreased performance, confirming that the optimal learning rate range is between 0.02 and 0.03 for balanced model performance.

Table 3. Learning rate optimization results using test set data

Round	Learning rate	Accuracy	Loss
1	0.01	0.79	1.29
2	0.02	0.78	0.87
3	0.03	0.79	0.91
4	0.04	0.74	1.38
5	0.05	0.70	1.26
6	0.06	0.64	1.31
7	0.07	0.58	1.16
8	0.08	0.58	1.30
9	0.09	0.56	1.18
10	0.10	0.56	1.36

From Table 3, experiments with increasing learning rate revealed that accuracy remained stable at 0.78 to 0.79 across learning rates of 0.01 to 0.03, while the lowest loss values of 0.87 to 0.91 were achieved at learning rates of 0.02 to 0.03. Accuracy demonstrated a consistent declining trend when the learning rate was increased beyond this optimal threshold, dropping significantly from 0.74 (at learning rate =0.04) to 0.56 (at learning rate =0.10). Consequently, both accuracy and loss patterns confirm that the range of 0.02 to 0.03 provides the most balanced performance, effectively maintaining high predictive accuracy while ensuring model stability.

3.2.3. Dropout rate optimization results

The dropout rate was adjusted while maintaining other parameters constant: input layer =224×224×3, Conv2D (activation = ReLU), dense layer (10, activation = softmax), Adam optimizer, learning rate =0.03, categorical cross-entropy loss, and 20 epochs. Figure 5 displays the performance impact of varying dropout rates on MobileNet training dynamics through four sub-graphs (Figures 5(a)-5(d)), illustrating how increasing dropout rates affect model learning capacity, regularization effectiveness, and the balance between training accuracy and generalization performance on the validation set. Specifically, Figure 5(a) presents training accuracy, Figure 5(b) presents training loss, Figure 5(c) presents validation accuracy, and Figure 5(d) presents validation loss.

The dropout rate optimization results are presented in Figure 5(a) shows the training accuracy across dropout rates from 0.1 to 0.9, in which accuracy remains stable at approximately 0.76–0.79 for dropout rates between 0.1 and 0.6, before declining sharply to 0.56 at a dropout rate of 0.9. Figure 5(b) displays the training loss values, where loss fluctuates within a moderate range for lower dropout rates but increases notably when the dropout rate exceeds 0.7. Figure 5(c) illustrates the validation accuracy trends, revealing that optimal validation accuracy of approximately 0.70–0.75 is maintained at dropout rates of 0.1 to 0.3, with a gradual decline at higher values. Similarly, Figure 5(d) presents the validation loss progression, demonstrating that the most stable and lowest validation loss values are observed in the 0.1–0.3 dropout range, confirming this as the optimal regularization configuration for the MobileNet model, as summarized in Table 4.

Based on the dropout rate optimization results shown in Table 4, the accuracy and loss metrics indicate minimal variation in accuracy during initial increases in dropout rate. However, loss values showed a notable decrease in the range of 0.1 to 0.3, confirming this as the optimal dropout range for the MobileNet model. The accuracy demonstrated a consistent declining trend with continuous increases in dropout rate beyond this optimal range, indicating excessive information loss during training.

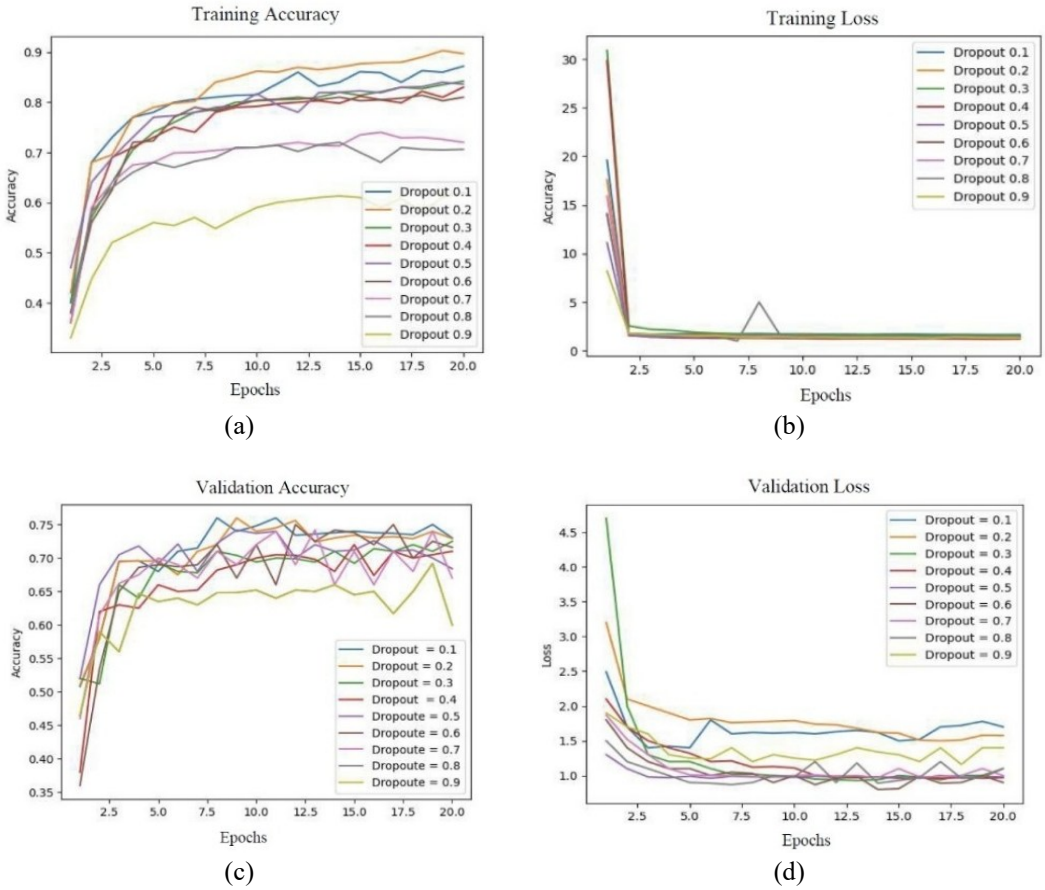

Figure 5. MobileNet performance comparison with dropout rate optimization of (a) training accuracy, (b) training loss, (c) validation accuracy, and (d) validation loss

Table 4. Dropout rate optimization results using test set data

Round	Dropout	Accuracy	Loss
1	0.1	0.79	1.34
2	0.2	0.77	0.94
3	0.3	0.77	0.97
4	0.4	0.76	0.91
5	0.5	0.76	0.84
6	0.6	0.77	0.7
7	0.7	0.70	0.90
8	0.8	0.61	1.06
9	0.9	0.56	1.18

3.2.4. Study results of optimizer adjustment

The experiment was conducted by maintaining constant values for other hyperparameters as follows: input layer =224×224×3, Conv2D (activation = ReLU), Dense (10, activation = softmax), loss = categorical_crossentropy, epochs =20, dropout =0.3, and learning rate =0.03. Figure 6 presents the comparative performance of SGD, Adam, and AdamW optimizers applied to MobileNet training through four sub-graphs (Figures 6(a)–6(d)). These plots enable direct visual comparison of training stability, convergence speed, and final performance levels achieved by each optimizer under identical experimental conditions.

The comparative optimizer results are presented as follows. Figure 6(a) presents the training accuracy curves of SGD, Adam, and AdamW, in which SGD achieves the highest training accuracy of 0.80, followed by AdamW at 0.78 and Adam at 0.75. Figure 6(b) displays the training loss for each optimizer, where SGD maintains the lowest loss value of 0.73, while Adam exhibits the highest loss at 0.93, indicating less effective convergence. Figure 6(c) illustrates the validation accuracy trends, revealing that SGD maintains the most stable validation accuracy at approximately 0.75 throughout training, while Adam and AdamW show greater fluctuation. Figure 6(d) presents the validation loss progression, demonstrating that SGD achieves the lowest and most stable validation loss at approximately 0.80, confirming it as the optimal optimizer for this concrete crack detection task, as summarized in Table 5, which shows the performance

comparison of different optimizers, algorithms used for model parameter adjustment. The experimental results demonstrate that model accuracy across the three optimizers ranged from 0.75 to 0.80, with SGD achieving the highest accuracy of 0.80.

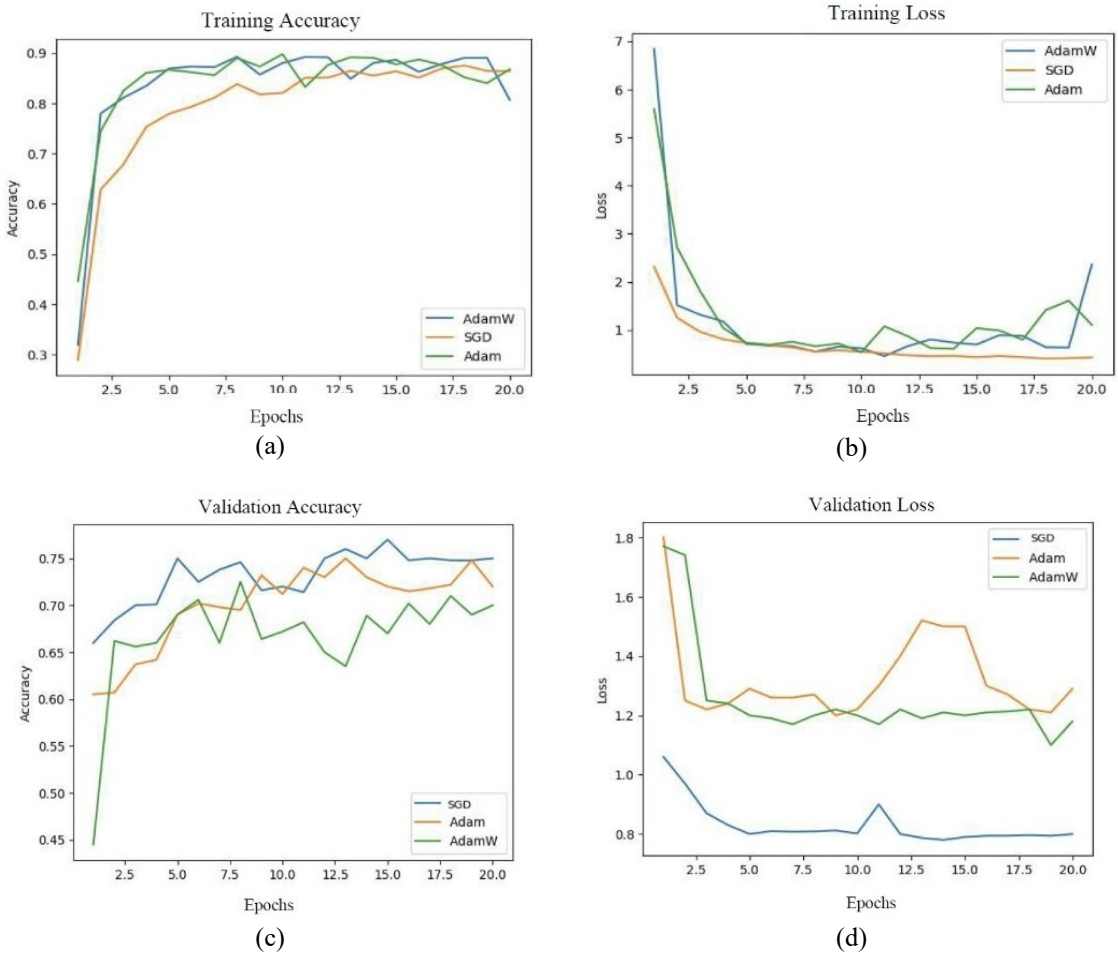

Figure 6. MobileNet performance comparison with different optimizers of (a) training accuracy, (b) training loss, (c) validation accuracy, and (d) validation loss

Table 5. Optimizer performance comparison (SGD, Adam, and AdamW) using test set data

Round	Optimizer	Accuracy	Loss
1	SGD	0.80	0.73
2	Adam	0.75	0.93
3	AdamW	0.78	0.76

The performance of MobileNet achieved in this study can be contextualized within the broader landscape of deep learning-based crack detection research. A comparative analysis of multiple deep learning architectures, including Inception-v3, VGG-16, and ResNet-50, for crack detection in buildings using 24,000 images found that Inception-v3 achieved the highest accuracy at 99.98%, while different models excelled across precision and recall metrics [28]. A study comparing CNN and MobileNetV2 for concrete crack identification using 40,000 images found that both models achieved comparable detection accuracy; however, MobileNetV2's significantly smaller model size made it a more efficient selection for mobile device deployment [18].

In this study, the MobileNet model, achieving 80% accuracy with a compact size of 13.1 MB, demonstrates competitive performance compared to these existing approaches [18], [28] while maintaining a significantly smaller model footprint suitable for mobile deployment. Furthermore, a comprehensive evaluation of 14 lightweight deep learning models, comprising CNN, vision transformer, and hybrid architectures, for real-time crack detection [27] found that the optimal model selection depends on the

trade-off between detection accuracy and inference time, with different architectures performing best on computers versus mobile devices. The findings of the present study align with this perspective, as MobileNet's favorable balance between detection accuracy and model size positions it as an effective solution for resource-constrained mobile environments.

Compared to existing research evaluating CNN pre-trained models for concrete crack classification under standard conditions [17], this study extends the comparative framework by incorporating mobile device deployment constraints and systematic hyperparameter optimization. The inclusion of optimizer selection analysis and epoch tuning provides additional practical insights for field-based infrastructure inspection applications. By demonstrating that MobileNet can achieve 0.80 accuracy with a compact model size of 13.1 MB, this research contributes to the growing body of knowledge on efficient deep learning solutions for resource-constrained structural health monitoring. However, misclassification may occur under poor lighting conditions or when crack characteristics are visually similar across categories; thus, expert verification is recommended for critical structural assessments.

4. CONCLUSION

This research compared MobileNet, VGG-16, and ResNet-50 for concrete crack detection on mobile devices using 1,634 images across 10 categories, partitioned at a 60:20:20 ratio. Among the three models, MobileNet achieved the best performance with 80% accuracy and a training loss of 0.73, using the SGD optimizer, a learning rate of 0.03, a dropout rate of 0.3, and 20 epochs, while maintaining a compact model size of 13.1 MB. The optimized model was successfully deployed as a Flutter-based mobile application totaling 110 MB, confirming its practical feasibility for field-based concrete crack inspection. Future work should expand the dataset to cover more crack types and environments, include per-class metrics such as precision, recall, and F1-score, apply k-fold cross-validation, and report mobile benchmarks including inference latency and frames per second.

ACKNOWLEDGMENTS

The authors would like to express their sincere gratitude to the Graduate School and Creative Innovation in Science and Technology Program, Faculty of Science and Technology, Nakhon Si Thammarat Rajabhat University, for providing the academic support and research facilities throughout this study. The generous support from the Faculty of Science and Technology, Songkhla Rajabhat University, has also been invaluable to the successful completion of this research. Special appreciation goes to the experts who participated in the validation process. Their constructive feedback and expertise have significantly enhanced the quality of this research. Lastly, we thank the Center of Excellence in Nanomaterials Chemistry for their technical support and research infrastructure that made this work possible.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Sarapee Chunkaew	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
Somporn Ruang-On	✓	✓	✓	✓	✓		✓			✓		✓	✓	
Prawit Nuengmatcha	✓			✓	✓					✓		✓	✓	
Kritaphat Songsri-in				✓	✓					✓		✓		

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

INFORMED CONSENT

Not applicable as this study did not involve human subjects or personal data collection.

ETHICAL APPROVAL

Not applicable as this research did not involve human subjects or animal experiments.

DATA AVAILABILITY

The dataset that supports the findings of this study is available from the corresponding author, [SR], upon request. The dataset consists of 1,634 crack images including online sources and researcher-captured photos, which have been categorized into 10 classes based on crack location and cause.

REFERENCES

[1] W. Xiong, Y. He, Y. Zhang, W. Luo, L. Ma, and J. Luo, "Fine-grained image-to-image transformation towards visual recognition," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020, pp. 5839–5848, doi: 10.1109/CVPR42600.2020.00588.

[2] M. AL-Qadri *et al.*, "Comparison of U-Net and fully convolutional networks (FCN) for concrete cracks detection using raw images under various conditions," *Journal of Intelligent & Fuzzy Systems: Applications in Engineering and Technology*, vol. 49, no. 2, pp. 527–539, Aug. 2025, doi: 10.3233/JIFS-239709.

[3] S. Roy, B. Yogi, R. Majumdar, P. Ghosh, and S. K. Das, "Deep learning-based crack detection and prediction for structural health monitoring," *Discover Applied Sciences*, vol. 7, no. 7, Jun. 2025, doi: 10.1007/s42452-025-07272-y.

[4] K. C. Laxman, N. Tabassum, L. Ai, C. Cole, and P. Ziehl, "Automated crack detection and crack depth prediction for reinforced concrete structures using deep learning," *Construction and Building Materials*, vol. 370, Mar. 2023, doi: 10.1016/j.conbuildmat.2023.130709.

[5] Y.-A. Hsieh and Y. J. Tsai, "Machine learning for crack detection: review and model performance comparison," *Journal of Computing in Civil Engineering*, vol. 34, no. 5, Sep. 2020, doi: 10.1061/(ASCE)CP.1943-5487.0000918.

[6] Q. Yuan, Y. Shi, and M. Li, "A review of computer vision-based crack detection methods in civil infrastructure: progress and challenges," *Remote Sensing*, vol. 16, no. 16, Aug. 2024, doi: 10.3390/rs16162910.

[7] G. Xu, Q. Yue, and X. Liu, "Deep learning algorithm for real-time automatic crack detection, segmentation, qualification," *Engineering Applications of Artificial Intelligence*, vol. 126, Nov. 2023, doi: 10.1016/j.engappai.2023.107085.

[8] A. Saberironaghi and J. Ren, "DepthCrackNet: a deep learning model for automatic pavement crack detection," *Journal of Imaging*, vol. 10, no. 5, Apr. 2024, doi: 10.3390/jimaging10050100.

[9] H. Zhang, J. Liu, and G. Hu, "FCN attention enhancing asphalt pavement crack detection through attention mechanisms and fully convolutional networks," *Scientific Reports*, vol. 15, no. 1, Jul. 2025, doi: 10.1038/s41598-025-92971-0.

[10] J. Wang, X. Li, Y. Xu, Z. Zhou, W. Wu, and Z. Li, "Optimizing CNN for pavement distress detection via edge-enhanced multi-scale feature fusion," *PLOS ONE*, vol. 20, no. 4, Apr. 2025, doi: 10.1371/journal.pone.0319299.

[11] T. Yamaguchi and T. Mizutani, "Road crack detection interpreting background images by convolutional neural networks and a self-organizing map," *Computer-Aided Civil and Infrastructure Engineering*, vol. 39, no. 11, pp. 1616–1640, Jun. 2024, doi: 10.1111/mice.13132.

[12] X. Wu, W. Li, D. Hong, R. Tao, and Q. Du, "Deep learning for unmanned aerial vehicle-based object detection and tracking: a survey," *IEEE Geoscience and Remote Sensing Magazine*, vol. 10, no. 1, pp. 91–124, Mar. 2022, doi: 10.1109/MGRS.2021.3115137.

[13] N. Li, Y. Zhang, Q. Zhang, and S. Zhu, "Transfer learning model for crack detection in side SlopesBased on Crack-Net," *Applied Sciences*, vol. 15, no. 13, Jun. 2025, doi: 10.3390/app15136951.

[14] S. Qian, C. Ning, and Y. Hu, "MobileNetV3 for image classification," in *2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*, Mar. 2021, pp. 490–497, doi: 10.1109/ICBAIE52039.2021.9389905.

[15] C. Lin, D. Tian, X. Duan, and J. Zhou, "TransCrack: revisiting fine-grained road crack detection with a transformer design," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 381, no. 2254, Sep. 2023, doi: 10.1098/rsta.2022.0172.

[16] S. Alshawabkeh, L. Wu, D. Dong, Y. Cheng, L. Li, and M. Alanaqreh, "Automated pavement crack detection using deep feature selection and whale optimization algorithm," *Computers, Materials & Continua*, vol. 77, no. 1, pp. 63–77, 2023, doi: 10.32604/cmc.2023.042183.

[17] M. Padsumbiya, V. Brahmbhatt, and S. P. Thakkar, "Automatic crack detection using convolutional neural network," *Journal of Soft Computing in Civil Engineering*, vol. 6, no. 3, pp. 1–17, 2022, doi: 10.22115/scce.2022.325596.1397.

[18] L. Hui, A. Ibrahim, and R. Hindi, "Computer vision-based concrete crack identification using MobileNetV2 neural network and adaptive thresholding," *Infrastructures*, vol. 10, no. 2, Feb. 2025, doi: 10.3390/infrastructures10020042.

[19] Y. Wu, S. Li, J. Li, Y. Yu, J. Li, and Y. Li, "Deep learning in crack detection: a comprehensive scientometric review," *Journal of Infrastructure Intelligence and Resilience*, vol. 4, no. 3, Sep. 2025, doi: 10.1016/j.iintel.2025.100144.

[20] G. X. Hu, B. L. Hu, Z. Yang, L. Huang, and P. Li, "Pavement crack detection method based on deep learning models," *Wireless Communications and Mobile Computing*, vol. 2021, no. 1, Jan. 2021, doi: 10.1155/2021/5573590.

[21] H. Kaveh and R. Alhajj, "Recent advances in crack detection technologies for structures: a survey of 2022-2023 literature," *Frontiers in Built Environment*, vol. 10, Jul. 2024, doi: 10.3389/fbuil.2024.1321634.

[22] L. Ali, F. Alnajjar, H. Al Jassmi, M. Gocho, W. Khan, and M. A. Serhani, "Performance evaluation of deep CNN-based crack detection and localization techniques for concrete structures," *Sensors*, vol. 21, no. 5, Mar. 2021, doi: 10.3390/s21051688.

[23] A. W. H, P. S. Ramachandraa, and S. Naveen, "Automatic detection of crack in concrete structure using deep learning: a study," in *2025 10th International Conference on Applying New Technology in Green Buildings (ATiGB)*, Jul. 2025, pp. 573–581, doi: 10.1109/ATiGB66719.2025.11142129.

[24] M. Impraimakis, "A convolutional neural network deep learning method for model class selection," *Earthquake Engineering & Structural Dynamics*, vol. 53, no. 2, pp. 784–814, Feb. 2024, doi: 10.1002/eqe.4045.
[25] K. S. B. Kharthik *et al.*, "Transfer learned deep feature based crack detection using support vector machine: a comparative study," *Scientific Reports*, vol. 14, no. 1, Jun. 2024, doi: 10.1038/s41598-024-63767-5.
[26] B. Guo, X. Li, and D. Li, "Crackwave R-convolutional neural network: a discrete wavelet transform and deep learning fusion model for underwater dam crack detection," *Structural Health Monitoring*, vol. 25, no. 2, pp. 1299–1315, Mar. 2026, doi: 10.1177/14759217241308132.
[27] G. Su, Y. Qin, H. Xu, and J. Liang, "Automatic real-time crack detection using lightweight deep learning models," *Engineering Applications of Artificial Intelligence*, vol. 138, Dec. 2024, doi: 10.1016/j.engappai.2024.109340.
[28] S. S. R. Krishnan *et al.*, "Comparative analysis of deep learning models for crack detection in buildings," *Scientific Reports*, vol. 15, no. 1, Jan. 2025, doi: 10.1038/s41598-025-85983-3.

BIOGRAPHIES OF AUTHORS

Sarapee Chunkaew received the M.S. degrees in Management Information System from Prince of Songkhla University, Thailand, in 2004. Currently, she is an assistant professor at the Faculty of Science and Technology, Songkhla Rajabhat University, Thailand. She is studying Ph.D. in Creative Innovation in Science and Technology, Nakhon Si Thammarat Rajabhat University, Thailand. She can be contacted at email: sarapee.ch@skru.ac.th.

Somporn Ruang-On received the B.Sc. degree in Computer Science from Rajabhat Phetchaburi Institute, Thailand, in 1995, the M.Sc. degrees in Information Technology from Sripatum University, Thailand, in 2003 and Ph.D. degree in Quality Information Technology from Phetchaburi Rajabhat University, in 2013. Currently, he is an assistant professor at the Faculty of Science and Technology, Nakhon Si Thammarat Rajabhat University, Thailand. He can be contacted at email: somporn_rua@nstru.ac.th.

Prawit Nuengmatcha is a chemist who has made significant contributions to the field of chemistry. He earned his Bachelor of Science degree in Chemistry in 1998 and continued his academic journey by obtaining a Master of Science degree in Chemistry in 2001. In the same year, he joined the faculty at the Department of Chemistry, Faculty of Science and Technology, Nakhon Si Thammarat Rajabhat University, where he began his career as a lecturer. His dedication to research and education led him to pursue a Doctorate in Chemistry, which he completed at Khon Kaen University, Thailand, in 2016. Over the years, his excellence in teaching and research earned him the title of assistant professor in 2009, followed by his promotion to associate professor in 2020. His research primarily focuses on the fabrication of visible light-active photocatalysts and sonocatalysts aimed at enhancing environmental protection. Currently, he serves as the head of the Creative Innovation in Science and Technology program and leads the Center of Excellence in Nanomaterials Chemistry, where he continues to advance the field through innovative research and mentorship. He can be contacted at email: prawit_nue@nstru.ac.th.

Kritaphat Songsri-in finished M.Eng. and Ph.D. in computing from Imperial College London in 2011 and 2020, respectively. Currently, he is an assistant professor in the Department of Computer Science at Nakhon Si Thammarat Rajabhat University, Thailand. His research interests include machine learning, deep learning, and computer vision. He has published in and is a reviewer for multiple international conferences and journals such as IEEE Transactions on Image Processing and IEEE Transactions on Information Forensics & Security. He was a recipient of the Royal Thai Government Scholarship covering his undergraduate and postgraduate degrees in 2010. He received the Best Student Paper Awards at the IEEE 13th International Conference for Automatic Face and Gesture Recognition (FG2018) and the 6th National Science and Technology Conference (NSCIC2021). In 2021, his Ph.D. thesis received an award from the National Research Council of Thailand (NRCT). He can be contacted at email: kritaphat_son@nstru.ac.th.