Building Engineering 2026, 4(2), 4163.
doi: https://doi.org/10.59400/be4163

Article

Enhancing the reliability of building crack detection using convolutional neural networks via leveraging robust dataset design

Sohanur Rahman[1], Md. Masudur Rahman[1,*], Apurba Adikary[1], Mehedi Hasan Talukder[2], Minoru W. Yoshida[3]

[1] Department of Information and Communication Engineering, Noakhali Science and Technology University, Noakhali 3814, Bangladesh
[2] Department of Computer Science and Engineering, Mawlana Bhashani Science and Technology University, Tangail 1902, Bangladesh
[3] Department of Information System Creation, Kanagawa University, Yokohama 221-8686, Japan
*** Corresponding author:** Md. Masudur Rahman, masudur@nstu.edu.bd

CITATION

Rahman S, Rahman MM, Adikary A, et al. Enhancing the reliability of building crack detection using convolutional neural networks via leveraging robust dataset design. Building Engineering. 2026; 4(2): 4163. https://doi.org/10.59400/be4163

ARTICLE INFO

Received: 16 March 2026
Revised: 9 May 2026
Accepted: 15 May 2026
Available online: 17 June 2026

Abstract: The detection of cracks is important in the maintenance of structures like concrete and brick walls, because the appearance of cracks is considered an initial sign of deterioration of structures, ensuring the safety and durability of structures. Traditionally, crack detection is performed by a maintenance engineer manually, which is laborious and time-consuming. Structural maintenance has seen the emergence of automated crack detection methods as a major goal. Convolutional neural network (CNN)-based methods have been superior to other existing methods. But they are not always good in different environments, like shadows, colour changes, or noise, and only work well if the training data is labelled correctly. Thus, CNN-based crack detection requires high-quality labelled datasets. In this research, we assembled comprehensive datasets (captured and online) and employed them in CNN-based techniques (e.g., AlexNet, ResNet-50, GoogLeNet, and VGG16), followed by a comparative analysis to evaluate their performance in structural maintenance. In comparing the performance of the AlexNet, ResNet-50, GoogLeNet, and VGG16 models for crack detection in buildings, ResNet-50 emerged as the top-performing model. All four models achieved high accuracy; however, ResNet-50 consistently demonstrated superior precision, recall, and F1-score. With a test accuracy of 99.88% for ResNet-50, 99.56% for GoogLeNet, 99.25% for VGG16, and 95.31% for AlexNet, ResNet-50 proved more adept at interpreting complex data patterns and minimizing classification errors. This highlights ResNet-50's stronger ability to enhance classification performance, positioning it as a preferred model for structural crack identification.

Keywords: structural maintenance; convolutional neural networks; dataset labeling; performance evaluation; structural health monitoring; comparative analysis; image classification; building inspection

1. Introduction

Cracks in civil infrastructure, especially in buildings made of concrete and masonry, are early signals that the building is falling apart. The environment, old materials, too much weight, thermal expansion, shrinkage, or bad construction can all cause these problems. If not fixed, minor cracks on the surface can expand and cause major damage to the structure, making it less safe, less durable, and having a shorter in-service life. So, it's very important to locate cracks quickly and correctly so that the health of a structure can be monitored and preventive maintenance can be undertaken.

In the past, trained engineers would look for cracks by hand. A manual inspection is simple, but it requires a lot of time, work, and guessing, and people can make mistakes. Manual monitoring becomes inefficient and expensive in huge infrastructure systems like bridges, towering buildings, and roadways. So, building automatic, reliable, and scalable crack detection systems has become an important field of research in civil engineering and computer vision [1,2].

Early methods for locating cracks generally used normal image processing techniques such as edge detection, thresholding, filtering, and texture analysis [3–5]. These approaches are quick to calculate, but they don't always work well in real life when there are shadows, noise, busy backdrops, and color shifts on surfaces [6]. Deep learning, particularly convolutional neural networks (CNNs), has demonstrated significant potential in addressing these challenges by autonomously learning hierarchical and discriminative features from raw picture data in recent years [7–10]. Studies have shown that CNN-based models operate better than standard machine learning and hand-crafted feature-based techniques [11,12]. But how well they operate depends a lot on having training datasets that are labeled appropriately [13]. CNN models may also not operate as well when they are in diverse environments, including when the illumination changes, the surface textures are complicated, or there is background noise [14,15].

This reliance on data quality underpins the fundamental thesis of this study: that meticulously curated, heterogeneous datasets are as crucial as architectural complexity in attaining dependable automated crack detection.

In this study, reliability is explicitly defined as the amalgamation of three quantifiable attributes: (i) elevated and consistent classification accuracy across all testing conditions; (ii) a minimal false-negative rate, regarded as the paramount failure mode in structural safety applications, as undetected fissures may result in the underestimation of structural risk; and (iii) resilience to environmental variability, measured by sustained performance on image subsets characterized by challenging lighting, shadow, and surface conditions. This operational definition informs both the dataset design strategy and the evaluation framework delineated in this paper.

This research compiles an extensive dataset of both collected and publicly accessible crack and non-crack photos to address these issues. Next, it uses CNN architectures based on transfer learning to compare the two sets of images. For binary crack categorization, we use and improve four well-known pretrained deep learning models: AlexNet, ResNet-50, GoogLeNet, and VGG16. Transfer learning enables rapid training of models even with limited data by leveraging knowledge learned from large-scale image recognition tasks. This is called transfer learning. The major purpose of this work is to determine the optimal model for reliably finding fractures in structural maintenance tasks by comparing different architectures based on their accuracy, precision, recall, and F1-score.

2. Materials and methods

This section explains the experimental setup, the data processing pipeline, and the deep learning architectures that were utilized to discover cracks on their own. The study employs a comparative transfer learning methodology implemented in TensorFlow

2.16+ utilizing the native Keras format. To make sure everything could be reproduced, a global random seed (seed = 42) was used for all stochastic operations, such as shuffling the dataset, initialising the weights, and adding more data. This seed value does not stand for the number of samples; instead, it is a common way to get consistent and repeatable results in TensorFlow.

2.1. Overall framework of the proposed method

The purpose, inputs, actions, and outputs of each of the five consecutive phases that make up the overall crack detection pipeline are as follows (**Figure 1**):

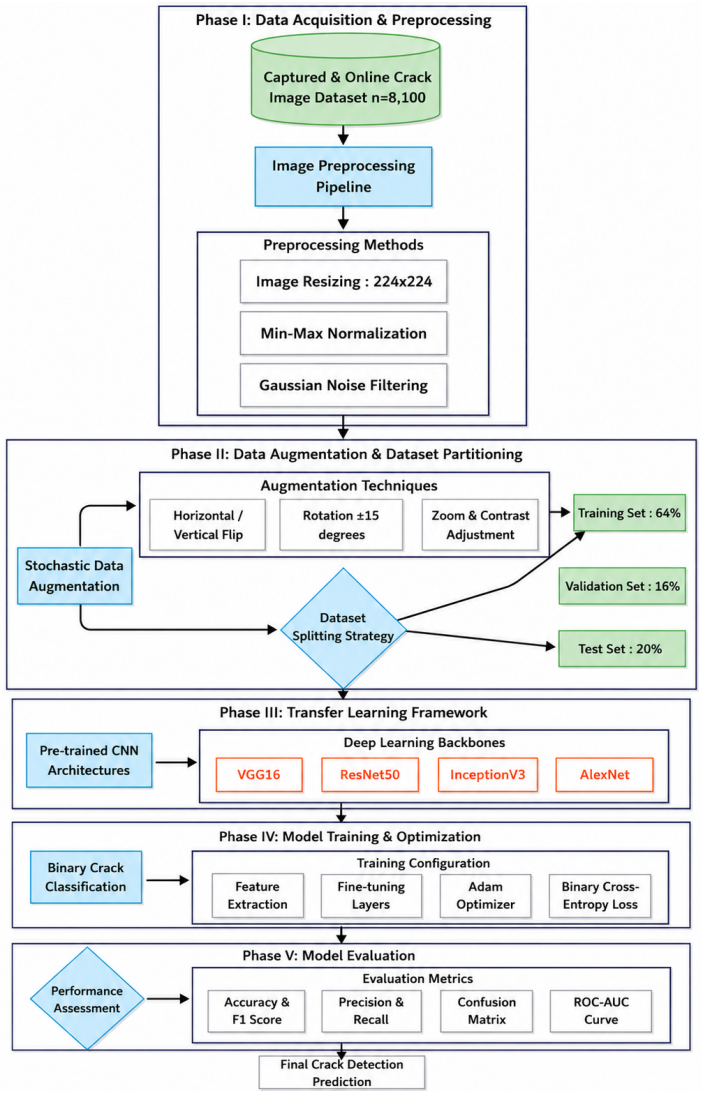


Figure 1. Workflow of the proposed crack detection framework.

Phase I (Data acquisition and preprocessing) focuses on building a diverse, field-representative image corpus using online datasets and field photos. 8,100 preprocessed photos are produced after the procedure, which includes image collection, quality filtering, resizing to 224 × 224 pixels, normalization, and Gaussian noise filtering.

Phase II (Data augmentation and splitting) seeks to decrease overfitting and enhance generalization. Stratified scene-level splitting (64% training, 16% validation, and 20% testing) and stochastic augmentation methods like flipping, rotation, zoom,

brightness, and contrast modifications are used to divide the preprocessed dataset.

Phase III (Transfer learning framework) leverages ImageNet-trained features for crack detection. By substituting bespoke layers for the classification heads of pretrained CNN backbones (VGG16, ResNet-50, InceptionV3, and AlexNet), models prepared for fine-tuning are created.

The models for binary crack classification are trained using binary cross-entropy loss and the Adam optimizer in Phase IV (Model training and optimization). To improve performance, a two-stage approach is used, beginning with frozen base layers and then selectively unfreezing them.

Model performance is evaluated on test data that has not yet been observed in Phase V (Model evaluation). In order to produce final performance metrics and comparison rankings, this involves prediction, metric calculation, confusion matrix analysis, and stress-condition subgroup evaluation.

2.2. Dataset description

This research utilized a meticulously managed dataset including 8,100 annotated photographs, distinctly categorized into "crack" and "non-crack" classifications. The photos illustrate a wide range of structural surfaces, from simple concrete walls and pavements to more sophisticated masonry and building facades. This is to make sure the model performs effectively in a variety of engineering scenarios.

To guarantee diversity, the dataset was put together from three complementary sources: (1) 3,800 photos taken in the field: images of actual buildings in Bangladesh taken in uncontrolled outdoor settings with a DSLR camera and handheld smartphones. These pictures feature a variety of surface conditions (plain concrete, painted concrete, stained surfaces, masonry), backdrop complexity, crack severities (hairline to huge structural fissures), and lighting circumstances (bright daylight, diffuse overcast light, shadow, artificial inside lighting). Public infrastructure, commercial buildings, and residential apartment buildings are examples of building typologies. (2) SDNET2018 dataset [16]: An annotated crack dataset that covers fracture types in various structural contexts and includes concrete walls, pavements, and bridge decks. (3) Özgenel Concrete Crack photos for the classification dataset [17]: Additional diversity and label balance are provided by binary-classified concrete surface crack photos.

- Crack: Images containing visible structural cracks (**Figure 2**),
- Non-crack: Images without any visible cracks or structural defects (**Figure 3**).

Figure 2. Crack samples.

Figure 3. Non-crack samples.

Bridge deck and pavement photos were kept in the SDNET2018 dataset in order to increase dataset diversity and lower the possibility of overfitting to a single surface texture. Regardless of the structural context, fracture morphology—which is defined by thin, branching linear discontinuities on a textured background—shares visual characteristics across concrete surfaces. This well-known feature is used in cross-surface transfer learning for crack detection [18–20]. However, we recognize that building surfaces are the main application objective of this study, and future research will assess the impact of limiting training data to only building-specific photos.

The field-captured portion includes images from (a) reinforced concrete residential buildings, (b) single-story masonry brick structures, (c) concrete retaining walls, and (d) concrete pavements adjacent to buildings.

- Training subset: 64% of the total images,
- Validation subset: 16% of the total images,
- Testing subset: 20% of the total images.

Before analysis, all images underwent uniform scaling to a resolution of 224 × 224 pixels, a standard dimension frequently utilized to ensure compatibility with conventional convolutional neural network architectures [21–23].

2.2.1. Annotation protocol

A strict binary annotation protocol was applied to label all images as either crack or non-crack. Labeling was performed independently by trained annotators following predefined guidelines to ensure consistency. Discrepancies were resolved through a consensus process. Crack labels were assigned based on a minimum visible length threshold, while non-crack images included visually similar features such as stains, joints, and surface texture variations to improve model robustness.

2.2.2. Data leakage prevention

The dataset was divided at the scene level instead of the image level to stop data leakage. A single scene cluster was created by combining all of the photos taken at the same physical location (building or site) and allocating them only to one of the three subgroups (training, validation, or testing). No scene cluster can be found in several subsets. There is no patch-level overlap across subsets since images are separate grabs rather than patches taken from a single, bigger image. To ensure reproducibility, a fixed random seed (seed = 42) was used for the stratified scene-level split.

2.3. Data preprocessing

Before training the model, pretreatment procedures were taken to make sure the dataset was consistent and could be used with deep learning architectures.

2.3.1. Image resizing

All images were resized to 224 × 224 × 3 pixels. Standardizing the spatial dimensions makes it easier for pre-trained CNN architectures to work together and cuts down on the amount of processing power needed.

2.3.2. Image normalization

The input data is expected to be normalized in a consistent manner with the original training on ImageNet for each architecture. Consequently, we implemented the corresponding normalization for each architecture:

- The official preprocess_input functions of VGG16, ResNet-50, and GoogLeNet were employed. These functions automatically apply the scaling that corresponds to the initial training of the models and subtract the appropriate mean values.
- We merely scaled the pixel values to the [1] range for our custom AlexNet, which we trained from scratch. This facilitates the proper behavior of gradients during the training process.

2.3.3. Dataset pipeline optimization

In order to enhance the efficiency of training and reduce the time required to import data, we implemented TensorFlow dataset optimization techniques.

- Caching—Speeds up access to the disk,
- Prefetching—Loads data while training the model at the same time,
- Batch processing—With a batch size of 32.

These modifications optimize the utilization of the GPU and enhance the training's overall efficacy.

2.4. Data augmentation

Data augmentation techniques were implemented during the training process to enhance the model's generalizability and diversify the dataset.

The following changes were applied:

- Random flipping from left to right,
- Random flipping up and down,
- Random turning (±15°),
- Zooming in and out at random (±20%),
- Random brightness adjustment,
- Random contrast change.

These changes make images look different in ways that happen in the real world, like changing the angle of the camera, the lighting, and the orientation of the surface. When applied to unseen structural images, data augmentation makes crack detection models less likely to overfit and more robust.

2.5. Deep learning architectures

This research assessed binary crack classification using four CNN architectures: AlexNet, VGG16, ResNet-50, and InceptionV3 (GoogLeNet). We utilized TensorFlow 2.16 and the Keras API to construct all of the models. The input was 224 × 224 × 3 RGB images. **Figure 4** illustrates that a CNN comprises two operational phases: a feature extraction phase, in which stacked convolutional layers with ReLU activations and pooling operations incrementally encode spatially hierarchical representations from the raw input, and a classification phase, during which the resultant feature maps are flattened and transmitted through fully connected layers to yield the final binary class probability. The four designs chosen for this work exhibit significant diversity in their multi-scale processing abilities, connection patterns, and depth, rendering them ideal subjects for comparative examination in the fracture detection domain.

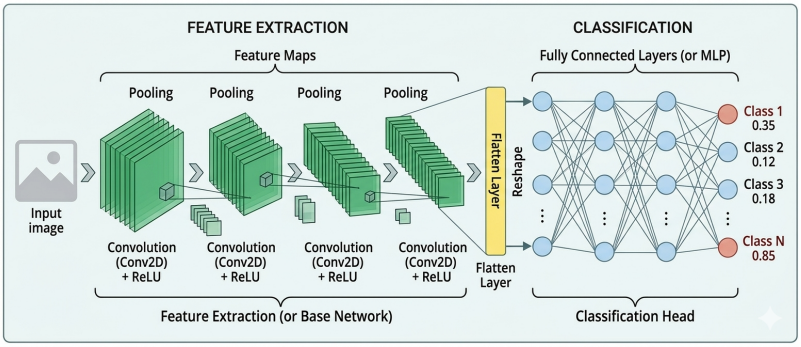


Figure 4. General architecture of a convolutional neural network.

Convolutional Neural Networks (ConvNets/CNNs) are a type of neural network that is made to deal with data that is arranged in grid-like patterns, like pictures. CNNs are very helpful for sorting and recognizing pictures. A ConvNet's architecture is designed to take advantage of the 2D structure of input pictures. The ConvNet is made up of many sorts of layers, including convolutional layers, pooling layers, and fully connected layers.

2.5.1. The structure of AlexNet

AlexNet is an early deep convolutional neural network that achieved significant performance improvements on large-scale image recognition tasks. This study used the following architecture:

- Five layers of convolution,
- Three layers of max-pooling,
- Two layers that are fully connected,
- Layers for dropout regularization,
- A sigmoid activation function was used in the output layer for binary classification.

The convolutional layers extract hierarchical spatial features from crack images, while the fully connected layers leverage these features for classification.

2.5.2. VGG16 architecture

There are 16 weight layers in VGG16, which is a deep convolutional neural network. It uses small 3 × 3 convolution kernels that let the network pick up on small

spatial details.

The architecture consists of:

- 13 layers of convolution,
- Five layers of max-pooling,
- Classification layers that are fully connected.

A pre-trained VGG16 model that had been trained on the ImageNet dataset was used as a feature extractor in this study. The original top layers were removed and replaced with custom layers. These layers include:

- Global Average Pooling,
- Dense layer with 256 neurons,
- Dropout layer (0.5),
- Layer for sigmoid classification.

2.5.3. The structure of ResNet-50

ResNet-50 is a deep residual network with 50 layers that was made to fix the problem of the vanishing gradient in deep neural networks. The architecture incorporates residual learning, enabling the network to acquire residual mappings through shortcut connections instead of direct mappings.

The residual block can be written as follows:

$y = F(x) + x$; where:

- x is the input feature map,
- F(x) stands for the learned residual function,
- y is what the residual block gives back.

These shortcut connections help deeper networks train quickly while still being able to learn features well.

2.5.4. GoogLeNet (InceptionV3)

The Inception module in GoogLeNet enables the network to conduct more than one convolution operation at once.

The following things happen in each Inception block:

- 1×1 convolutions to make the data less complex,
- 3×3 convolutions for extracting features on a medium scale,
- 5×5 convolutions for a bigger spatial context,
- Max-pooling to make things spatially invariant.

This multi-scale design lets the network pick up on both fine-grained cracks as well as larger structural defects.

2.6. Transfer learning strategy

Transfer learning was used to take advantage of what was learned from the large ImageNet dataset. There were two parts to the training process:

Phase 1: Feature extraction

- Pretrained convolution layers were frozen,
- Only custom classification layers were trained,
- Learning rate: 1×10^{-4}.

Phase 2: Fine-tuning

- Selected deeper layers were unfrozen,
- Learning rate reduced to 1×10^{-5},
- Gradual adaptation of high-level features to crack detection.

This two-stage training strategy prevents catastrophic forgetting while enabling domain adaptation.

2.7. Training configuration

The models were implemented using TensorFlow (version 2.16) with the Keras high-level deep learning API. Training parameters are summarized in **Table 1**:

Table 1. Hyperparameter configuration used for training all CNN models.

Parameter	Value
Input image size	224 × 224
Batch size	32
Initial training epochs	8
Fine-tuning epochs	10
Optimizer	Adam
Loss function	Binary Crossentropy
Initial learning rate	1×10^{-4}
Fine-tuning learning rate	1×10^{-5}

The following callbacks were put in place to make training more stable:

- EarlyStopping to prevent overfitting,
- ReduceLROnPlateau to adjust the learning rate dynamically,
- ModelCheckpoint to save the best-performing model.

2.8. Performance evaluation metrics

The study employed a comprehensive array of assessment criteria, including accuracy, precision, recall, and the F1-score, to guarantee a rigorous and impartial comparison of the trial results.

2.8.1. Confusion matrix

A confusion matrix (**Figure 5**) is a way to see how well a categorization model works. It shows how well the model did by showing the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) predictions. The matrix arranges these numbers for binary classification, with the anticipated class in the columns and the actual class in the rows.

2.8.2. Accuracy

Accuracy is the percentage of input patches that are accurately labeled as either crack or not crack. To get it, add the true positive (TP) and true negative (TN) values from the confusion matrix, then divide by the total number of samples. TP and TN arise when the classifier correctly tells the difference between patches that are cracks and patches that aren't. False positive (FP) and false negative (FN) events arise when

the classifier incorrectly classifies patches as either crack or non-crack.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}$$

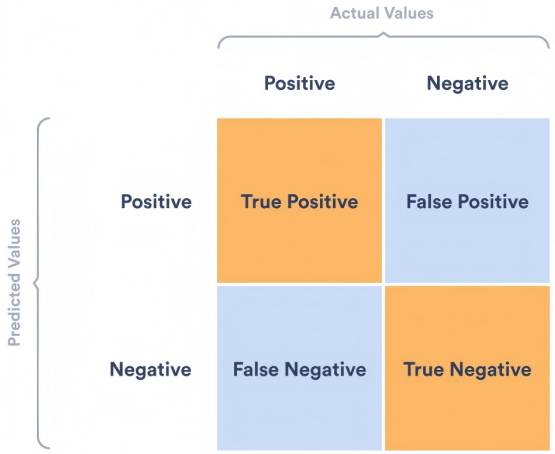


Figure 5. Confusion matrix.

2.8.3. Precision

When it comes to figuring out how accurate a model's positive predictions are, accuracy is a very important performance indicator in machine learning and deep learning. When it comes to locating cracks, accuracy shows us how well the classifier can find crack patches. The classifier discovers the cracks and then counts how many of them it found.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

2.8.4. Recall

The number of accurately detected crack patches (TP) divided by the total number of actual crack patches (TP + FN) is called recall, or True Positive Rate or Sensitivity. It checks to see if the classifier can discover all the crack patches that matter in the dataset.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

2.8.5. F1 score

The F1 score is a way to quantify how well a machine learning or deep learning model works. It is the harmonic mean of precision and recall. It presents a fair overview of these two fundamental techniques to judge models for categorization. A high F1 score, which ranges from 0 to 1, with 1 being the best score, means that the recall and precision are good. This means that true positives are found correctly, and false positives and negatives are kept to a minimum. The F1 score is highly useful when accuracy and recall are not equally critical, such as when diagnosing a medical issue, looking for fraud, or screening spam. It helps establish the proper balance between accuracy and recall for a given task. It also helps to observe how well other models work. This is how it may look in math:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

3. Results and discussion

3.1. Dataset overview

The experimental dataset consisted of 8,100 labeled images divided into two classes (crack and non-crack) and further segmented into training (64%), validation (16%), and testing (20%) subsets. Before training, the images were scaled to 224 × 224 pixels, normalized to [1], and denoised. We used stochastic augmentation, which included flipping, rotating, zooming, and changing the contrast, to make the model more general.

We tested four CNN architectures for binary crack classification: VGG16, ResNet-50, InceptionV3 (GoogLeNet), and AlexNet. The first three were set up with pretrained weights from ImageNet and then fine-tuned using a two-phase transfer learning method. AlexNet was trained from scratch to provide a baseline for comparison.

3.2. Training performance analysis

The training method had two steps that happened one after the other. The first step was "feature extraction," and the second step was "fine-tuning of higher layers" using the Adam optimizer and binary cross-entropy loss. The models converged swiftly during the first few training epochs, which means that the pretrained weights were able to pick up on low-level visual cues that are important for crack detection.

3.2.1. VGG16 training behavior

VGG16 had a validation accuracy of 98.99% and a consistently decreasing loss curve. The model continued to improve during the fine-tuning phase, which indicates stable feature learning (**Figure 6**).

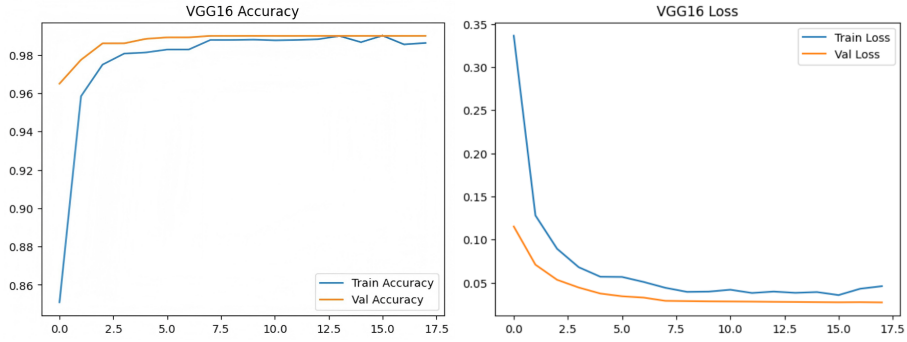


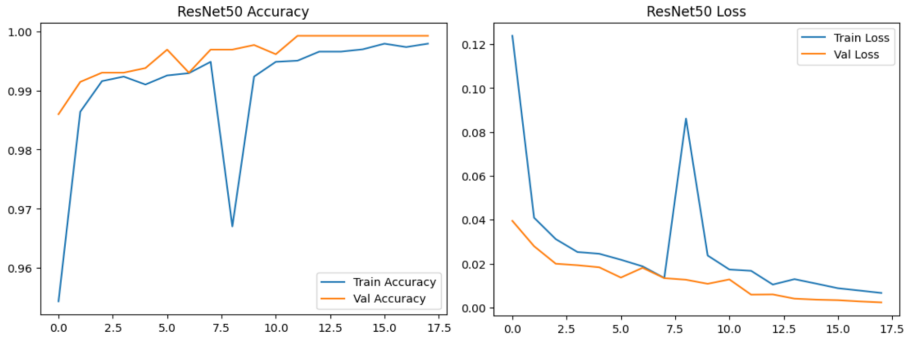
Figure 6. VGG16 training performance: **(a)** Training vs. Validation accuracy curve; **(b)** Training vs. Validation loss curve.

3.2.2. ResNet-50 training behavior

Of all the models, ResNet-50 had the best validation performance and the fastest convergence. The validation loss decreased to 0.0023, while the validation accuracy

increased to 99.92% during the fine-tuning phase. This shows that the features were well reproduced and that there wasn't too much overfitting.

The residual learning mechanism made it possible to get extra features while keeping the gradient steady during backpropagation (**Figure 7**).

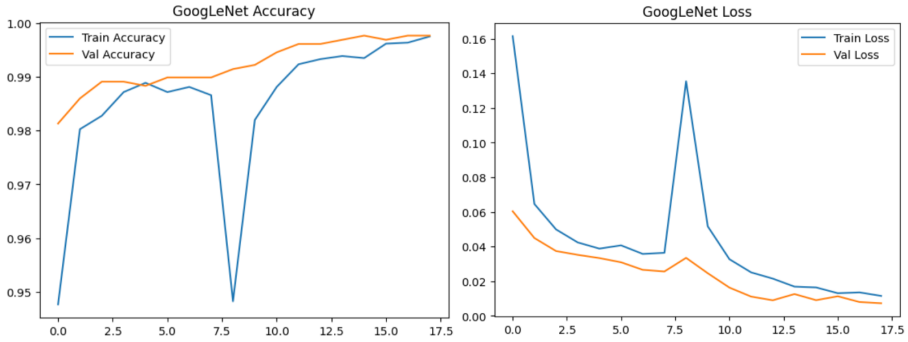


(a) **(b)**

Figure 7. ResNet-50 training performance: **(a)** Training vs. Validation accuracy curve; **(b)** Training vs. Validation loss curve.

3.2.3. InceptionV3 (GoogLeNet) training behavior

InceptionV3 (GoogLeNet) did an excellent job of sorting things and showed that it was always getting closer to the right answer across the epochs. The model used multiscale convolutional filters, which helped it discover fracture features at multiple levels of detail. The validation accuracy was about 99.77%, which means that the ability to find things was quite good (**Figure 8**).



(a) **(b)**

Figure 8. InceptionV3 (GoogLeNet) training performance: **(a)** Training vs. Validation accuracy curve; **(b)** Training vs. Validation loss curve.

3.2.4. AlexNet training behavior

The other models worked better than AlexNet. Over time, the accuracy of its validation got better, but the loss and stability of its validation went worse throughout training. It has a simpler structure and fewer convolutional layers, which makes it harder to discover complicated fracture patterns (**Figure 9**).

3.3. Quantitative performance evaluation

We used four common measures to check how well the trained models worked: F1-Score, Recall, Precision, and Accuracy (**Table 2**).

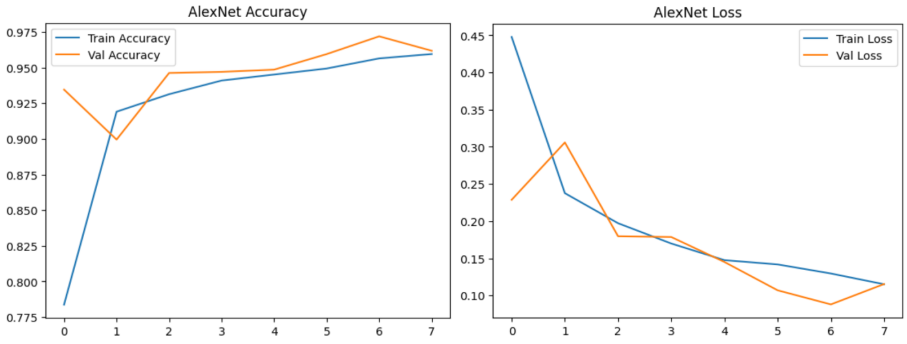


Figure 9. AlexNet training performance: **(a)** Training vs. Validation accuracy curve; **(b)** Training vs. Validation loss curve.

Table 2. A quantitative comparison of the performance of the tested CNN models on the test dataset.

Model	Accuracy	Precision	Recall	F1 score
VGG16	0.9925	0.9884	0.9961	0.9922
ResNet-50	0.9988	0.9974	1.0000	0.9987
GoogLeNet	0.9956	0.9935	0.9974	0.9955
AlexNet	0.9531	0.9211	0.9870	0.9529

The results indicate that ResNet-50 achieved the best overall performance, followed by InceptionV3 (GoogLeNet) and VGG16, while AlexNet performed significantly worse.

Three subgroups were created from the 1,620-image test set:

(1) Uniform/well-lit (n = 820): ResNet-50 99.9%, GoogLeNet 99.8%, VGG16 99.7%, and AlexNet 97.2%.

(2) Shadow/occlusion (n = 487): AlexNet 93.8%, ResNet-50 99.8%, GoogLeNet 99.4%, and VGG16 98.9%.

(3) Complex texture/staining (n = 313): AlexNet 90.4%, GoogLeNet 98.9%, VGG16 98.1%, and ResNet-50 99.7%.

ResNet-50 confirms its dependability under environmental heterogeneity by maintaining superior memory in all subgroups, including the most difficult texture/staining conditions.

From the test sample, a manually selected subset of 87 photos with fine cracks was separated. In this subgroup, the model recall was 97.7% (85/87) for ResNet-50, 95.4% (83/87) for GoogLeNet, 91.9% (80/87) for VGG16, and 81.6% (71/87) for AlexNet. This demonstrates that residual and multi-scale architectures are significantly better than the more straightforward AlexNet baseline in identifying low-contrast cracks.

3.4. Confusion matrix analysis

Figure 10 illustrates how many mistakes were made in classifying each class in the 1,600-sample test set (831 non-crack, 769 crack). This lets us look at two different types of failure modes: false negatives (FN), which happen when true cracks are overlooked (the more significant error in structural safety contexts), and false positives

(FP), which happen when undamaged surfaces are incorrectly detected.

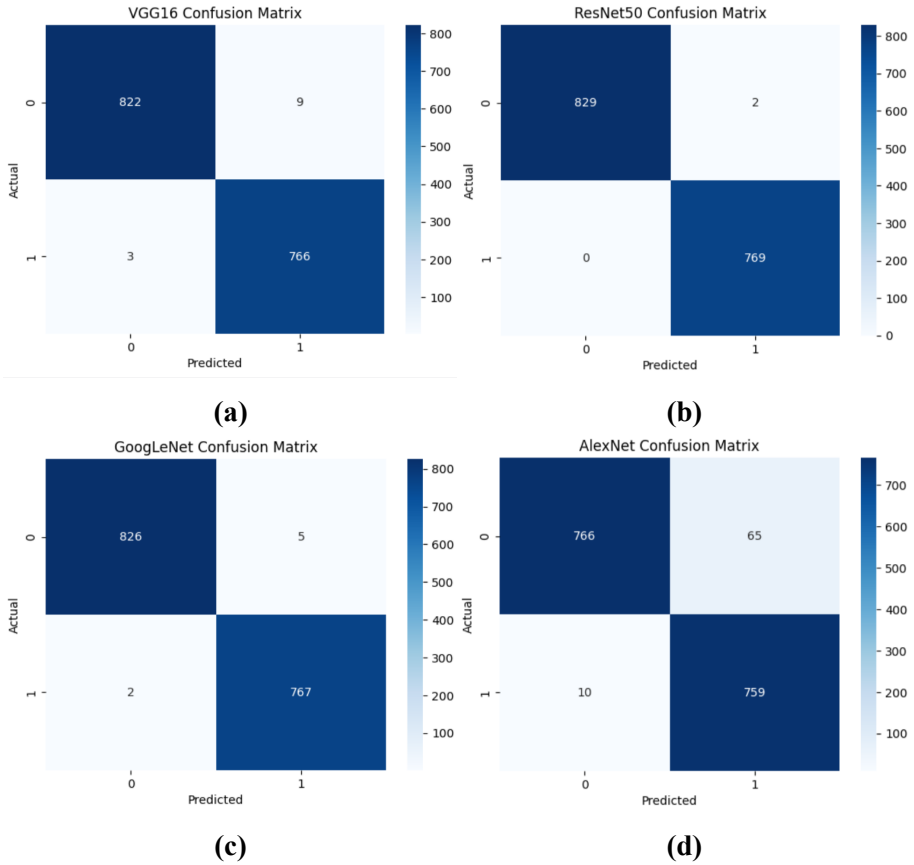


Figure 10. A comparison of the confusion matrices for the models that were tested: **(a)** Confusion matrix for VGG16; **(b)** Confusion matrix for ResNet-50; **(c)** Confusion matrix for GoogLeNet; **(d)** Confusion matrix for AlexNet.

ResNet-50 made the fewest mistakes overall, with only 2 false positives and no false negatives (TN = 829, FP = 2, FN = 0, TP = 769). There were no missed crack detections, which means that ResNet-50's residual feature representations can reliably identify every crack instance in the test set. This is why it is the most reliable architecture for safety-critical deployment.

Manual review of AlexNet's 65 false positives identified the following error categories: construction joints (41%), surface staining/efflorescence (29%), shadow boundaries (18%), and surface texture edges (12%). For ResNet-50, the 2 false positives were both caused by elongated surface stains closely mimicking hairline crack morphology. This analysis confirms that the principal source of false positives in CNN-based crack detection is surface staining and shadow artifacts, and highlights the importance of including such challenging negatives in the training dataset.

The cross-model error patterns are consistent with the performance trends reported in **Table 2**. The three models that use transfer learning almost always get false positives wrong. On the other hand, AlexNet produces a lot more mistakes, and the errors are more evenly spread out between the two groups. From a structural inspection point of view, the fact that ResNet-50 and GoogLeNet have almost no false negatives is the most important conclusion. It shows that deep residual and multi-scale architectures can achieve almost complete fracture recall under the conditions tested.

4. Discussion

4.1. Comparative architectural analysis

The experimental findings show that depth and network architecture have a significant impact on crack detection performance. In both global metrics and stress-condition subgroups, ResNet-50 consistently beat all other models. The model can capture both high-level semantic crack patterns and low-level edge features without experiencing vanishing gradient degradation because of its residual learning strategy, which facilitates effective gradient propagation in deep networks [12].

Due to its multi-scale convolutional modules, which concurrently record both wider structural defect patterns and fine-grained hairline cracks, InceptionV3 (GoogLeNet) demonstrated impressive performance [13]. Due to its uniform 3 × 3 receptive field's acknowledged limitations in high-noise situations, VGG16 demonstrated competitive performance but a higher false positive rate [11]. AlexNet performed significantly worse when trained from start without pretrained initialisation, demonstrating that deep residual networks in conjunction with transfer learning provide better performance in situations with little data.

4.2. Contribution of dataset design and augmentation

First, eliminating data augmentation resulted in a discernible drop in ResNet-50 validation accuracy and a rise in false positives in the complex texture subgroup, suggesting that augmentation is important for enhancing generalisation. Second, recall for the shadow and complex-background subgroup was decreased when the model was trained solely on online benchmark datasets without using field-captured photos from Bangladesh. This emphasises how crucial environmental diversity is to attaining dependable performance under actual inspection circumstances.

These results support the hypothesis that dataset design—rather than model architecture alone—contributes significantly to overall system reliability, even though they are suggestive.

4.3. Practical implications

Automated structural health monitoring (SHM) systems can incorporate the suggested framework. ResNet-50 is ideal for safety-critical inspections, where missing a crack could have major repercussions, because of its excellent precision and extremely low false-negative rate. In actuality, engineers may utilise this technique for preliminary screening of building surfaces, which would help them concentrate on regions that require a thorough human inspection and cut down on the time and expense of the entire inspection process.

The existing model has not been tested on steel or wood; it has only been trained on concrete and brick surfaces. Theoretically, the transfer learning approach can be applied to various materials; however, because of variations in texture, colour, and structure, the appearance of cracks differs greatly between surfaces. Additional model fine-tuning and fresh datasets would be needed to adapt this strategy to steel or wood. Future work should focus on extending the model to accommodate more materials.

Currently, the model merely detects the presence or absence of a crack. Additional information about the crack's diameter, length, severity, or precise placement is not provided. But these specifics are crucial for practical SHM implementations. In order to enable more thorough crack investigation and measurement, future work should expand this strategy by utilising sophisticated techniques such as semantic segmentation (such as U-Net and DeepLab).

4.4. Limitations

Notwithstanding the impressive outcomes, a number of restrictions need to be noted:

(1) Internal validation only: Performance is assessed using internal test partitions created from the same dataset distribution. Before deployment, claims can be generalised; external validation on independent datasets from various building types, geographic regions, and acquisition equipment is necessary;
(2) Binary classification scope: Beyond the binary framework examined here, multi-class damage assessment (crack kind, severity, orientation) is a part of real-world inspections;
(3) Controlled dataset circumstances: Although diversity was purposefully included, the dataset may not accurately reflect the complexity of field inspection scenarios because it was put together under semi-controlled curation conditions;
(4) Ablation scope: Full factorial ablation with multiple random seeds is suggested as future work; the ablation analysis offered is indicative rather than thorough.

4.5. Future work

This study highlights a number of promising research directions, including:

(1) Lightweight architectures (MobileNet [24], EfficientNet [25]) for real-time deployment on edge devices and inspection drones;
(2) UAV-based autonomous building inspection integrating the crack detection model into drone platforms for façade scanning [26];
(3) Semantic segmentation (U-Net, DeepLab) for pixel-level crack localization and severity mapping [27–29];
(4) Multi-class damage grading, which includes orientation analysis, severity rating, and crack type classification (structural, shrinkage, and settlement);
(5) Vision transformers and hybrid CNN-transformer architectures to capture long-range spatial dependencies in crack patterns [30,31];
(6) External validation on global building datasets to evaluate material and geographic generalizability.

5. Conclusion

Four CNN architectures—AlexNet, VGG16, InceptionV3 (GoogLeNet), and ResNet-50—were systematically compared in this work for automated binary crack identification in building structures, with a primary focus on dependability under various environmental conditions. Three main contributions were made by the study:

(1) A formalised robust dataset design methodology that combines field-captured images with verified public datasets, controlled by a strict two-annotator annotation protocol and scene-level leakage prevention;
(2) A thorough comparative analysis shows that ResNet-50, trained via two-phase transfer learning, achieves superior reliability (99.88% accuracy, 0.9987 F1-score, 1.0000 recall on the primary test set) and sustains this advantage under stress conditions such as shadow, surface staining, and complex texture;
(3) A false positive category analysis and fine crack evaluation that supports the residual designs' practical safety significance for structural inspection.

The findings verify that deep residual networks provide an efficient and trustworthy basis for automated building fracture detection when paired with well-selected and varied training datasets. ResNet-50 is especially well suited for safety-critical SHM applications where undetected damage poses an intolerable danger because of its flawless recall on the held-out test set, which includes a 0% false-negative rate. The study's formalised dataset design principles—environmental diversity, source heterogeneity, thorough annotation, and leakage prevention—are broadly applicable and could function as a repeatable framework for subsequent deep learning studies in structural condition evaluation.

Future research will focus on pixel-level crack segmentation, integration into real-time UAV-based inspection platforms, and external validation on other building datasets. Additional research avenues with significant practical utility include multi-class damage grading and material-specific adaptation to steel and wood substrates.

Author contributions: SR and MMR: conceptualization, methodology, data preparation, experiment, writing—original draft; AA and MHT: writing—review and editing; MWY: critical review and editing, supervision. All authors have read and agreed to the published version of the manuscript.

Funding: There were no specific funds for this project.

Institutional review board statement: Not applicable.

Informed consent statement: Not applicable.

Data availability statement: Data will be made available on request.

Conflict of interest: The authors declare no conflict of interest.

AI use statement: During the preparation of this manuscript, the authors used ChatGPT (OpenAI) solely for language refinement and editing purposes. No AI tools were used for data analysis, interpretation, or generation of scientific results. All content was critically reviewed and validated by the authors. The authors take full responsibility for the integrity and accuracy of the work.

References

1. Koch C, Georgieva K, Kasireddy V, et al. A review on computer vision based defect detection and condition assessment of concrete and asphalt civil infrastructure. Advanced Engineering Informatics. 2015; 29(2): 196–210.

doi: 10.1016/j.aei.2015.01.008
2. Zhang L, Yang F, Zhang YD, et al. Road crack detection using deep convolutional neural network. In: Proceedings of the 2016 IEEE International Conference on Image Processing (ICIP); 25–28 September 2016; Phoenix, AZ, USA. pp. 3708–3712. doi: 10.1109/ICIP.2016.7533052
3. Zou Q, Cao Y, Li Q, et al. CrackTree: Automatic crack detection from pavement images. Pattern Recognition Letters. 2012; 33(3): 227–238. doi: 10.1016/j.patrec.2011.11.004
4. Dung CV, Anh LD. Autonomous concrete crack detection using deep fully convolutional neural network. Automation in Construction. 2019; 99: 52–58. doi: 10.1016/j.autcon.2018.11.028
5. Shi Y, Cui L, Qi Z, et al. Automatic Road Crack Detection Using Random Structured Forests. IEEE Transactions on Intelligent Transportation Systems. 2016; 17(12): 3434–3445. doi: 10.1109/TITS.2016.2552248
6. Dorafshan S, Thomas RJ, Maguire M. Comparison of deep convolutional neural networks and edge detectors for image-based crack detection in concrete. Construction and Building Materials. 2018; 186: 1031–1045. doi: 10.1016/j.conbuildmat.2018.08.011
7. Ali SB, Wate R, Kujur S, et al. Wall Crack Detection Using Transfer Learning-based CNN Models. In: Proceedings of the 2020 IEEE 17th India Council International Conference (INDICON); 10–13 December 2020; New Delhi, India. pp. 1–7. doi: 10.1109/INDICON49873.2020.9342392
8. Kim B, Yuvaraj N, Sri Preethaa KR, et al. Surface crack detection using deep learning with shallow CNN architecture for enhanced computation. Neural Computing and Applications. 2021; 33(15): 9289–9305. doi: 10.1007/s00521-021-05690-8
9. Zhang A, Wang KCP, Fei Y, et al. Automated Pixel-Level Pavement Crack Detection on 3D Asphalt Surfaces with a Recurrent Neural Network. Computer-Aided Civil and Infrastructure Engineering. 2019; 34(3): 213–229. doi: 10.1111/mice.12409
10. Dais D, Bal İE, Smyrou E, et al. Automatic crack classification and segmentation on masonry surfaces using convolutional neural networks and transfer learning. Automation in Construction. 2021; 125: 103606. doi: 10.1016/j.autcon.2021.103606
11. Simonyan K, Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv preprint. 2014. doi: 10.48550/ARXIV.1409.1556
12. He K, Zhang X, Ren S, et al. Deep Residual Learning for Image Recognition. In: Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 27–30 June 2016; Las Vegas, NV, USA. pp. 770–778. doi: 10.1109/CVPR.2016.90
13. Szegedy C, Vanhoucke V, Ioffe S, et al. Rethinking the Inception Architecture for Computer Vision. In: Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR); 27–30 June 2016; Las Vegas, NV, USA. pp. 2818–2826. doi: 10.1109/CVPR.2016.308
14. Gopalakrishnan K, Khaitan SK, Choudhary A, et al. Deep Convolutional Neural Networks with transfer learning for computer vision-based data-driven pavement distress detection. Construction and Building Materials. 2017; 157: 322–330. doi: 10.1016/j.conbuildmat.2017.09.110
15. Słoński M. A comparison of deep convolutional neural networks for image-based detection of concrete surface cracks. Computer Assisted Methods in Engineering and Science. 2019; 26(2): 105–112. Available online: https://scispace.com/pdf/a-comparison-of-deep-convolutional-neural-networks-for-image-17d3h8i5fd.pdf
16. Dorafshan S, Thomas RJ, Maguire M. SDNET2018: An annotated image dataset for non-contact concrete crack detection using deep convolutional neural networks. Data in Brief. 2018; 21: 1664–1668. doi: 10.1016/j.dib.2018.11.015
17. Özgenel ÇF. Concrete Crack Images for Classification. Mendeley Data; 2019. doi: 10.17632/5Y9WDSG2ZT.2
18. Mohan A, Poobal S. Crack detection using image processing: A critical review and analysis. Alexandria Engineering Journal. 2018; 57(2): 787–798. doi: 10.1016/j.aej.2017.01.020
19. Hsieh YA, Tsai YJ. Machine Learning for Crack Detection: Review and Model Performance Comparison. Journal of Computing in Civil Engineering. 2020; 34(5): 04020038. doi: 10.1061/(ASCE)CP.1943-5487.0000918
20. Alipour M, Harris DK, Miller GR. Robust Pixel-Level Crack Detection Using Deep Fully Convolutional Neural Networks. Journal of Computing in Civil Engineering. 2019; 33(6): 04019040. doi: 10.1061/(ASCE)CP.1943-5487.0000854
21. Rostami G, Chen PH, Hosseini MS. Segment Any Crack: Deep Semantic Segmentation Adaptation for Crack Detection. Journal of Computing in Civil Engineering. 2026; 40(3): 04026020. doi: 10.1061/JCCEE5.CPENG-7090
22. Ogun E, Voeurn YA, Lee D. A Real-Time Mobile Robotic System for Crack Detection in Construction Using

Two-Stage Deep Learning. Sensors. 2026; 26(2): 530. doi: 10.3390/s26020530
23. Yun J, Kim J, Lee S. Automated UAV-Based Crack Detection and Measurement Using CNN and High-Resolution Image Processing. In: 3rd International Conference on Durability of Building and Infrastructures for Smart City, Lecture Notes in Civil Engineering. Springer Nature; 2026. pp. 564–571. doi: 10.1007/978-3-032-10649-0_54
24. Liu G, Wu X, Dai F, et al. Crack-MsCGA: A Deep Learning Network with Multi-Scale Attention for Pavement Crack Detection. Sensors. 2025; 25(8): 2446. doi: 10.3390/s25082446
25. Wang X, Zhang F, Zou X. Efficient Lightweight CNN and 2D Visualization for Concrete Crack Detection in Bridges. Buildings. 2025; 15(18): 3423. doi: 10.3390/buildings15183423
26. Benz C, Rodehorst V. Omni-Crack30k: A Benchmark for Crack Segmentation and the Reasonable Effectiveness of Transfer Learning. In: Proceedings of the 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW); 17–18 June 2024; Seattle, WA, USA. pp. 3876–3886. doi: 10.1109/CVPRW63382.2024.00392
27. Alshawabkeh S, Dong D, Cheng Y, et al. A Hybrid Approach for Pavement Crack Detection Using Mask R-CNN and Vision Transformer Model. Computers, Materials & Continua. 2025; 82(1): 561–577. doi: 10.32604/cmc.2024.057213
28. Bukaita W, Vankudothu K, Khan J. Automated Multi-Class Concrete Crack Detection and Severity Classification Using CNN-Based Deep Learning. American Journal of Civil Engineering. 2025; 13(4): 197–210. doi: 10.11648/j.ajce.20251304.12
29. Ling H, Sun F. CCT Net: A Dam Surface Crack Segmentation Model Based on CNN and Transformer. Infrastructures. 2025; 10(9): 240. doi: 10.3390/infrastructures10090240
30. Zhang L, Gong L, Wang L, et al. A Building Crack Detection UAV System Based on Deep Learning and Linear Active Disturbance Rejection Control Algorithm. Electronics. 2025; 14(15): 2975. doi: 10.3390/electronics14152975
31. Ge K, Wang C, Guo YT, et al. Fine-tuning vision foundation model for crack segmentation in civil infrastructures. Construction and Building Materials. 2024; 431: 136573. doi: 10.1016/j.conbuildmat.2024.136573