



Feasibility analysis of convolution neural network models for classification of concrete cracks in Smart City structures

Prashant Kumar^{1,2}  · Gaurav Purohit^{1,2} · Pramod Kumar Tanwar^{1,2} · Solomon Raju Kota^{2,3}

Received: 24 May 2021 / Revised: 3 November 2022 / Accepted: 13 March 2023 /

Published online: 23 March 2023

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract

Cracks are one of the forms of damage to concrete structures that debase the strength and durability of the building material and may pose a danger to the living being associated with it. Proper and regular diagnosis of concrete cracks is therefore necessary. Nowadays, for the more accurate identification and classification of cracks, various automated crack detection techniques are employed over a manual human inspection. Convolution Neural Network (CNN) has shown excellent performance in image processing. Thus, it is becoming the mainstream choice to replace the manual crack classification techniques, but this technique requires huge labeled data for training. Transfer learning is a strategy that tackles this issue by using pre-trained models. This work first time strives to classify concrete surface cracks by re-training of six pre-trained deep CNN models such as VGG-16, DenseNet-121, Inception-v3, ResNet-50, Xception, and InceptionResNet-v2 using transfer learning and comparing them with different metrics, such as Accuracy, Precision, Recall, F1-Score, Cohen Kappa, ROC AUC, and Error Rate in order to find the model with the best suitability. A dataset from two separate sources is considered for the re-training of pre-trained models, for the classification of cracks on concrete surfaces. Initially, the selective crack and non-crack images of the Mendeley dataset are considered, and later, a new dataset is used. As a result, the re-trained classifier of CNN models provides a consistent performance with an accuracy range of 0.95 to 0.99 on the first dataset and 0.85 to 0.98 on the new dataset. The results show that these CNN variants can produce the best outcome when finding cracks in the real situation and have strong generalization capabilities.

✉ Prashant Kumar
prashant.mnnit10@gmail.com

¹ CSIR-Central Electronics Engineering Research Institute, Pilani, Rajasthan 333031, India

² Academy of Scientific and Innovative Research (AcSIR), Ghaziabad, Uttar Pradesh 201002, India

³ CSIR - National Aerospace Laboratories, Bengaluru, Karnataka 560017, India

Keywords Convolution neural network model · Transfer learning · Image pre-processing · Global standardization · Clipping and rescaling pixel values · Confusion matrix · VGG-16 · Inception-v3 · ResNet-50 · DenseNet-121 · Xception · InceptionResNet-v2

1 Introduction

In today's world, concrete is the most frequently used human-made material for the construction of various smart city structures, such as buildings, bridges, dams, highways, nuclear facilities, and many more. It is ubiquitous because it has some positive characteristics, such as strength, durability, and a long-life cycle. Most living beings are somehow surrounded and dependent on concrete in order to ensure their protection and a substantial loss of property, the health of concrete structures should be monitored after a certain period of time. Concrete is known for its stability, but the cumulative impact of aging, the climate, human activities, and heavy loads have contributed to a degradation of concrete strength [15]. The concrete damages can be seen in the form of cracks and spalling. Little ignorance of these concrete damages can cause degradation in the quality of the infrastructure which can lead to significant damage to lives and economic losses. Even the slight findings of cracks on concrete are a major concern, indicating the early signs of infrastructure failure. Therefore, there is an urgent need for crack monitoring and detection by civil engineers.

The conventional approach for detecting concrete damage is to carry out a human examination by manually inspecting the presence of any damage with their naked eyes. Although, with this traditional approach, there are more chances of carelessness, which could lead to major losses. Automatic damage detection has now gained popularity through the use of Artificial Intelligence-based Machine Learning and Deep learning algorithms, which can detect even small defects with less Error Rate, minimizing time and cost consumption. In recent years, a variety of vision-based methods have been proposed for the concrete crack detection through Grey-Scale Histogram, Fuzzy C-Means Clustering, Cascade Features, V-Shaped Features, Spatial Tuned-Robust Multi-Feature (STRUM), and Spectral Analysis. Deep Convolutional Neural Networks (CNN) have recently been developed for the classification and detection of objects in Computer Vision [42]. In this research, CNN models based on transfer learning are developed to classifying the concrete surface cracks in two classes called crack and non-crack. The main contribution to the paper is as follows:

- **Dataset Formation:** All the CNN models are trained and validated separately on the standard dataset (Mendeley dataset) and the new dataset. Apart from the standard dataset, a new dataset is created by capturing images of various types of crack and non-crack areas on different campus buildings in CSIR-CEERI, Pilani. All of these datasets include various types of defects (Horizontal Cracks, Vertical Cracks, Diagonal Cracks, Shadows, Surface Roughness, Scaling, and Edges) on a concrete surface that helps to distinguish.
- **Image Pre-processing:** All images of both datasets before introducing them to the CNN models are pre-processed by standardization and rescaling of values in the ranges of 0 and 1. This reduces training time, promotes the convergence of optimization algorithms, and takes care of other modeling difficulties.
- **CNN Models:** This research refers to 6 deep CNN models such as VGG-16, Inception-v3, ResNet-50, Xception, InceptionResnet-v2, and DenseNet-121 with a transfer learning technique for crack classification of two distinct types of datasets. The classifier section of all 12 models (6*2) is modified to satisfy our problem requirements. Also, in the

classifier section, at least one Dense Layer is sandwiched so that a sufficient number of classifier parameters are available for learning features during training.

- **Tuning of Models:** During training, each model is validated on 20% images of the training set and after training, the history of the model is plotted to verify the correctness of the training. The hyperparameters of the model are tuned accordingly. This practice is followed with all models until the best curve fitting is achieved. Each model is executed at least 10–15 times in this way.
- **Metrics Calculation:** Apart from the accuracy that may be deceptive in evaluating the performance of the trained models, 6 other performance metrics, such as Precision, Recall, F1-Score, Cohen Kappa, ROC AUC, and Error Rate are measured on both datasets.

The structure for the paper is as follows. Section 2 offers a Literature Analysis of papers written in recent years. Shortly after a brief overview of the CNN models, is used for damage classification is given. Section 4 briefly outlines the methodology, including experimental set-up, dataset formation, pre-processing, and training & testing. Section 5 provides a thorough review of the results; Section 6 is for the conclusion.

2 Literature review

Over the last few years, progress has been made in the analysis and research work of the Structural Health Monitoring System to advance the processing and segmentation of concrete images based on Machine Learning. In the paper [10], the author proposed a model to detect various types of concrete damage and to classify them into five different groups, namely: longitudinal crack (LC), transverse crack (TC), diagonal crack (DC), spall damage (SD), and intact wall (IW). Steerable filters and projection integrals are used for image processing tasks to extract features from images along with machine learning algorithms; support vector machine (SVM) and least square support vector machine (LSSVM) to analyze and classify images into one of the five class labels. A satisfactory result with a classification accuracy of 85.33% has been achieved, which demonstrates that the combination of image processing and machine learning algorithms can achieve good accuracy. In the paper [8], the authors proposed a model for the identification of various cracks and the assessment of crack density using a deep, fully convolutional network (FCN). Three separate CNN models, i.e. VGG-16, Inception, and ResNet are employed for image classification, and their performance is compared, with VGG16 performing better than Inception-v3 and ResNet. The segmentation method, which achieved an approximate accuracy of 90%, was also used to classify cracks more precisely. The method of variation of pixel density ratio was used to measure crack density. In the paper [11], the authors proposed a model for detecting and segmenting crack and leakage defects that are typically located in the inner lining of the tunnel using FCN. In order to segment the defect images, two-stream algorithms are used to implement the corresponding FCN models, i.e. one stream is used to recognize crack by sliding-window-assembling process, and the other stream is adopted for the leakage by the resizing-interpolation operation. A deep learning algorithm was used in the paper [15] to detect cracks, and image processing techniques were used to analyze crack length and width. Crack segmentation was conducted using 2D CNN accompanied by the use of thinning, tracking, and profiling algorithms for crack characteristics. In the paper [18], the author proposed an AI-based approach for detecting low-level steel defects using the CNN algorithm. Different case studies have been performed and compared with each other. In the end, the proposed model was the one with the highest accuracy of about 95%. This paper [7] shows a comparative study of the six standard edge detection techniques

and the deep learning-based DCNN approaches used to detect concrete defects. The study shows that AlexNet DCNN used is much better than edge detection methods in many ways such as DCNN computational time is almost half as compared to the edge detection system, DCNN can be trained with a wide dataset that will help to improve the accuracy of the model, DCNN can detect finer cracks as narrow as 0.04 mm to 0.08 mm. On the other hand, edge detection methods are useful for detecting cracks greater than 0.1. These results demonstrate the importance of the adaptation of the DCNN system to edge detection methods. In addition, in order to minimize residual noise, the hybrid approach was proposed by integrating both edge detection methods and DCNN, which would result in a reduction in noise with a factor of 24. In the paper [33], horizontal and vertical cracks are detected using a statistical learning system, the whole process is divided into two phases, one of which is the training process and the other is the image detection process. The training process is completed by applying one of the Meta-algorithms, the Adaboost Learning Algorithm. Detected cracks are marked by red boxes. In the paper [9], VGG-16 was proposed with transfer learning to improve the efficiency of the model. In the paper [3], the author suggested a model using Mask R-CNN and image processing techniques. The validation of the trained model was carried out on 30 images, which were collected using drones, imaging cars, and road scanning cars. A total of 74 cracks were observed, of which 71 cracks are correctly identified by Mask R-CNN, and nine cracks are falsely detected, resulting in an accuracy of 88.7% and 95.5% of Recall. In the paper [40], the author proposed a method for dealing with the “All Black” problem, in which all the pixels of the image were considered as background and result in a significant loss. Crack-Patch-Only (CPO) supervision and generative adversarial learning have therefore been implemented to address the same problem. Semantic segmentation was introduced in the paper [41], in its entirety to the Fully Convolutional Networks. FCNN was proposed in this paper on the basis of dilated convolution for concrete crack detection over conventional segmentation algorithms, keeping in mind the lack of information when doing downsampling. At the encoding point, Resnet-18 was used as a simple network to extract features from the input image along with dilated convolutions at different dilation speeds. At the decoding stage, up-sampling is performed using deconvolutions until the size has become the same as the input image. The SoftMax function was then used to classify the up-sampling feature maps. The proposed model performance was then compared with three different DCNN models: FCN-Resnet-50, FCN-Resnet-18, and FCN-VGG, out of which the performance of the proposed model proved to be the most successful. In the paper [42], the author proposed a method for the identification of different types of defects using vision-based techniques. A CNN variant, Inception-v3 was used to train the model and then tested, which resulted in a test set accuracy of 97.8%. In the paper [1], a multifractal study of the features was carried out to identify variations in crack complexity. The level of crack damage was detected with an accuracy of 89.3%. In the paper [5], VGG16 with transfer learning was applied to the dataset of 3500 images, training and the testing division maintained a ratio of 80–20. The accuracy of 92.27% has been achieved. In the paper [12], pixel to pixel classification was performed using a deep convolutional neural network (DCNN) technique. One of its variant VGGNet was used for training purposes. The proposed model was compared to SVM and CNN with the best performance, with 92.8% accuracy and 92.7% F1-Score. The performance of both CNN with structured prediction and FCN were evaluated in paper [13] to detect the presence of cracks on pavements and their level of severity. Performance measurements parameters such as Precision, Recall and F1-Score for low, medium, and high severity are determined for both models. In paper [14], one of CNN’s versions, AlexNet was trained with five different classes to detect cracks automatically. The proposed model has been compared to other models and has been found to classify cracks more precisely. The proposed model was tested on 40 images with an average Precision of 92.35% and an average Recall of 89.28% at pixel level and in real-time video with a Recall of 81% and a Precision of 88%. In the paper [34], a

new multi-layer EML-based model was proposed for both learning and classification purposes instead of using other widely available crack detection techniques. This technique has shown a satisfactory result with effective training and testing speed. In the paper [4], the author addressed the value of time to time inspection of civil infrastructures such as bridges, tunnels, underground pipes and asphalt pavements. Several recent computer vision-based techniques have been reviewed. Image analysis methods for roof hole and wall crack detection have been used in paper [26]. SIFT and K-means clustering algorithm were used for roof hole detection and wall crack detection, and wall crack detection morphology, such as Dilation, Erosion, and Skeleton was applied. The CNN-based method and the SURF-based method were used in paper [16], after producing CCRs. The main aim of the paper was to distinguish crack and non-crack objects that could be misclassified as cracks using CCRs. The accuracy of the CNN-based approach was found to be more reliable and effective than the SURF-based crack identification method. In paper [31], the author addresses the forms of crack on concrete and the different method used to identify them. Morphological operation and the KD-tree were used to improve the image and to link the discontinuities present in the crack. Otsu's method of thresholding was used to identify cracks. The ANN algorithm was used in paper [19] to train and evaluate the model. Image processing was carried out using different filters and other morphological operations accompanied by an image classification using a backpropagation neural network. The accuracy of the crack image was 90% and the non-crack image was 92% with an overall accuracy of 90.25%. The CNN model with Atrous Spatial Pyramid Pooling (ASPP) module was proposed in paper [37]. The comparison of the proposed model was made with other versions of CNN, which resulted in the highest accuracy of 96.37%. Moreover, the proposed model can capture multi-scale image information at multiple sampling rates, taking the minimum computational time. One of the CNN variant AlexNet was introduced in paper [17] with few variations in it. Comparison of training outcomes by maintaining different base learning rates was made, and as a result, CNN obtained the highest validation accuracy of 99.06% at the base learning rate of 0.01. In paper [27], crack classification was performed by AE testing using AE parameters named RA and AF values. Classification of the model was achieved by the Gaussian Mixture Model, and the clustered data was separated by the SVM hyperplane. The combination of the two was used to classify various types of cracks. In paper [22], the CNN variant VGG-16 was used to model the CAM (Class Activation Mapping) method to locate the object. Followed by contrast with ResNet-50 and Inception models. In paper [32], the author compared four different CNN models according to different parameters, such as from scratch with and without DA, pre-trained VGG-16 with DA and pre-trained DA and FT. The results showed that the greatest accuracy of validation is the large pre-trained ConvNet with DA and FT. The genetic algorithm was used in paper [24] to detect defects on concrete. Genetic algorithm model based on genetic programming (GP) and percolation, not only has a high accuracy rate but also successfully produces performances that avoids additional interference factors such as stain, block, water leakage, etc. The threshold approach was used in paper [30] instead of the edge detection or machine learning algorithm dependent methods. The threshold method was used because of its ability to detect very thin cracks and gives a different colour to the crack that distinguishes it from the background. The clustering and thresholding of the input image were accomplished by the implementation of the geometric and contextual filters resulting in the output image in which the cracks were mapped in red.

This study employs a transfer learning technique to classify crack (Horizontal Cracks, Vertical Cracks, Diagonal Cracks, and Branch Cracks) and non-crack concrete surface images using CNN models such as VGG-16, ResNet-50, DenseNet-121, Inception-v3, Xception, and InceptionResNet-v2. Which are more sophisticated than typical image processing approaches like edge detection, fuzzy c-means clustering, and grey-scale histogram. Furthermore, their use

in crack classification with transfer learning is seldom seen in literature. In addition, the performance of CNN models is evaluated on two datasets so that the adaptability of CNN models for crack classification can be validated. The use of transfer learning allows the CNN models to train easily, rapidly converge and show improved generalization capabilities.

3 CNN models

CNN is a type of Neural Network that has shown its potential in the field of image recognition and classification. It is a multilayer model specially designed to identify the two-dimensional image. Like a neural network it has an input layer, an output layer and hidden layers consisting of several convolution layers with filters (Kernels), a non-linear activation layer (ReLU), a Pooling layer, a fully connected layer (FC) followed by a SoftMax function to classify an object with probabilistic values between 0 and 1.

The input can be in the form of an image, video, audio, or text with dimension, say $W \times H \times D$, when passed through the kernels or filters of any size, say $3 \times 3 \times D$ (with the same depth as input) gives the output the size of $W \times H \times N$ (where N is the number of filters). After each conv layer, the non-linear activation layer must be used to add non-linearity to the system since the associated functions are often usually non-linear. ReLUs are favored over other activation functions as they minimize computational time by speeding up training resulting in the improvement of the neural network. It has a simple calculation as it sets all the negative elements to 0. The pooling layer is a building block of CNN that is non-linear and used for downsampling to minimize image size and thus reduce computational and memory complexity. Mostly the max pool function is used, followed by a fully connected layer, which is connected to all the activation functions of the previous layers to define the final output category. Top categories are picked using SoftMax or SVM and display the associated probability. Here, the models of CNN are briefly described below. This research includes Inception-v3, DenseNet-121, Resnet-50, VGG-16, Xception and InceptionResnet-v2 (Tables 1 and 2).

Table 1 CNN Models Classifier Configuration for Training on DB1

CNN Models	Input Size	Layers in Classifier	Name of the Added Layers	No of Non-Trainable Parameters	No of Trainable Parameters
VGG-16	224*224	3	Flatten (25088), Dense (256), Dense (1)	14,714,688	6,423,041
DenseNet-121	224*224	5	Dense (1000), Activation (1000), Dense (500), Activation (500), Dense (1)	7,037,504	1,526,001
Inception-v3	224*224	3	GlobalAveragePooling2D (2048), Dense (1024), Dense (1)	21,802,784	2,099,201
ResNet-50	224*224	3	GlobalAveragePooling2D (2048), Dense (256), Dense (1)	23,587,712	524,801
Xception	224*224	4	GlobalAveragePooling2D (2048), Dropout (2048), Dense (256), Dense (1)	20,861,480	524,801
InceptionResNet-v2	224*224	3	GlobalAveragePooling2D (1536), Dense (1024), Dense (1)	54,336,736	1,574,913

Table 2 CNN Models Classifier Configuration for Training on DB2

CNN Models	Input Size	Layers in Classifier	Name of the Added Layers	No of Non-Trainable Parameters	No of Trainable Parameters
VGG-16	224*224	4	Flatten (25088), Dense (256), Dropout (256) Dense (1)	14,714,688	6,423,041
DenseNet-121	224*224	6	Dropout (1024), Dense (512), Activation (512), Dense (256), Activation (256), Dense (1)	7,037,504	656,385
Inception-v3	224*224	4	GlobalAveragePooling2D (2048), Dropout (2048), Dense (1024), Dense (1)	21,802,784	2,099,201
ResNet-50	224*224	4	GlobalAveragePooling2D (2048), Dropout (2048), Dense (256), Dense (1)	23,587,712	524,801
Xception	224*224	4	GlobalAveragePooling2D (2048), Dropout (2048), Dense (256), Dense (1)	20,861,480	524,801
InceptionResNet-v2	224*224	4	GlobalAveragePooling2D (1536), Dropout (1536), Dense (256), Dense (1)	54,336,736	393,729

3.1 VGG-16

VGG16 [38] is a type of convolution network given by K. Simonyan and A. Zisserman. The model obtains 92.7% test accuracy while being trained on the ImageNet dataset, which consists of over 14 million images of 1000 categories. It improves AlexNet by substituting large filters with several small 3*3 filters. It has a total of 21 layers, 13 Convolutional Layers with 3*3 filters and stride is fixed to 1 pixel. The 224*224 RGB image is given as an input to this layer. 5 Max Pooling Layers of 2*2 filters with stride 2 are also used with the same padding. The Convolution and Max Pooling Layers arrangement is being followed across the entire architecture. At last, there are three fully connected layers, followed by SoftMax. Out of these 21 layers, only 16 layers are weighted, therefore named VGG16.

3.2 RESNET-50

ResNet [35] stands for Residual Networks, which serve as a foundation for solving a variety of computer vision problems. This model won the first prize in ImageNet classification problem in 2015 with an error of 3.57% on the test data. It enables the training of a very deep neural network with 150+ layers without the gradient disappearing. It is form of convolution neural network with five stages. Each stage has different blocks; the convolution and identity block and each block comprise three layers of 1×1 , 3×3 , 1×1 convolution. Each ResNet architecture performs an initial convolution using a 7×7 size kernel and a max-pooling using a 3×3 size kernel. 50 in ResNet-50 signifies 50 layers in the network.

3.3 XCEPTION

Xception [29] is an add-on to the inception architecture that adapts the depth-wise separable convolutions instead of the traditional convolution. The depth-wise separable convolution is the depth-wise convolution preceded by a point-wise convolution. The Xception architecture consists of three blocks called entry flow, middle flow, and exit flow. It consists of 36 convolutional layers which create a feature extraction base of the network, divided into 14 modules with linear residual connections around them, except for the first and last modules.

3.4 INCEPTION-V3

Inception-v3 [23, 36] is one of the CNN's architectures that allows several enhancements over earlier versions such as Inception-v2, which includes factorized $7*7$ convolutions, label smoothing, and batch normalization in auxiliary classifiers. The Inception-v3 network consists of 11 modules of inception. Each module consists of pooling layer and convolutional filters with RLUs as an activation function. It consists of 42 convolutional layers using a stride of two convolutions instead of max-pooling between layers.

3.5 INCEPTIONRESNET-V2

Inception and ResNet alloyed Inception-ResNet, which blends the two architectures to boost the performance of both individuals. It is a mixture of the Inception structure and the Residual connection with identical computational costs as the inception-v3. In a single block of Inception-Resnet, the different dimensions of the convolutional filters are coupled with the residual connections. Residual connection prevents the loss of gradient problems and decreases training time. It is 164 layers deep and can classify images in 1000 object categories.

3.6 DENSENET-121

DenseNet [25] stands for Densely Connected Convolutional Networks, which have come into view due to its many advantages as it minimizes the core problem of gradient decay, makes full use of the network by feature reuse, reduces the number of parameters and strengthens the propagation of the features. Each layer in the Densenet is connected in a feed-forward way to the next layer and has direct access to the gradient from the loss function to the input image. 121 in DenseNet denotes the depth of the ImageNet models. Feature maps are connected and serve as input for the next consecutive layer. It consists of four dense blocks performing $1*1$ and $3*3$ convolution and pooling operations.

4 Methodology

The practice of this research consists of 5 main phases which comprise: 1) Transfer Learning 2) Database Formation 3) Image Pre-processing 4) Evaluation Metrics 5) Training and Testing 6) Result Analysis. Figure 1 shows that first of all CNN models are prepared by freezing the Feature Extractor section and attaching the modified Classifier section so that cracks can be classified. Images of the prepared datasets are pre-processed in the next step and 7 metrics are measured during and after training. The results of the training and testing are finally analyzed.

4.1 Transfer learning

Because it is difficult to obtain a dataset of sufficient size, most people do not train the entire convolutional network from scratch with random weights initialization. Instead, it is usual to employ a pre-trained convolution network that has been trained on large datasets such as ImageNet, and then reuse a convolutional network for another task, either as a starting point or as a fixed feature extractor. In our scenario, a convolutional network is used with the Fixed Feature Extractor mechanism. The Fixed Feature Extractor mechanism of transfer learning [21,

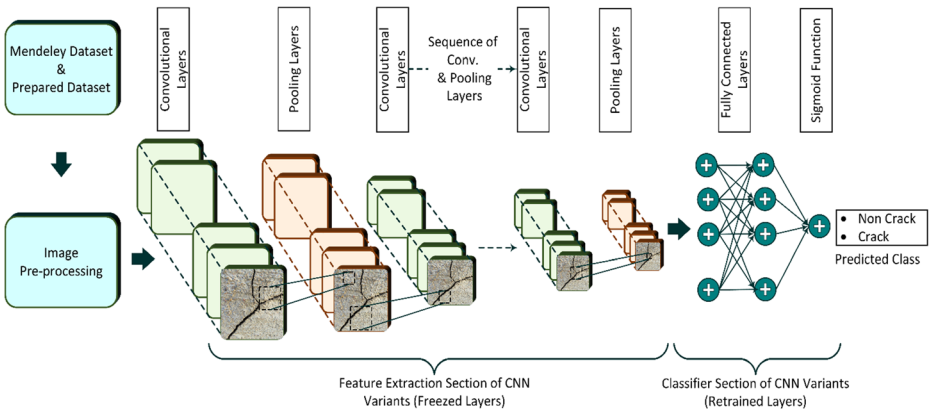


Fig. 1 Convolution Neural Network Architecture with Transfer Learning Technique

28], considers a convolutional network trained on a large dataset such as ImageNet, and removes the network’s top fully connected layer (classifier). It is used to generate 1000 class scores. The remainder of the convolutional network, in the absence of a classifier, is regarded as a Fixed Feature Extractor for the new dataset, i.e. the damages dataset. The improved classifier is then linked to the Fixed Feature Extractor to generate the appropriate classification result. Furthermore, throughout training, just the classifier portion is re-trained. Figure 1. depicts the notion of Fixed Feature Extractor-based Transfer Learning in an appealing manner.

4.2 Database formation

To classify the multiple types of defects on a concrete surface using CNN models, we need to train the model on the damage’s dataset. Here we are working on image processing, so the required dataset is in the form of images. Several images with different forms of defect are included in the dataset. In this research, we used two separate datasets to evaluate AI-based crack classification algorithms. Primarily, the selective images of Mendeley Crack Dataset [20, 39] were used in this analysis. The original dataset stores RGB images (227*227) in JPG format in their respective Positive (cracked) and Negative (non-cracked) directories. Each class comprises half of a total of 40,000 images. The dataset generated by the method proposed by Zhang et al. (2016) [39] consisted of 458 high-resolution images (4032*3024 pixel) of various buildings located in the Middle East Technical University. All images are taken at a distance of approximately 1 M from the target. A total of 5010 concrete surface images of the size of 227*227 pixels are considered for this test. Out of these images, 80% (3206 + 802 = 4008) of the randomly selected images were used for the train and validate the CNN models. The remaining 20% (1002) was used for the testing purposes. All images are pre-processed before

Table 3 Datasets Bifurcation for Training and Testing CNN Models

Database	Training	Validation	Test	Total
DB1	3206	802	1002	5010
DB2	1295	324	405	2024
Total	4501	1126	1407	7034

being used for the training, validation, and testing. A detailed overview of Image Pre-processing is given in the following section. In this article, it is referred to as DB1.

In the same way, another dataset of concrete surface images was constructed by capturing the different forms of cracks and non-crack surface images of various CSIR-CEERI, Pilani, laboratory buildings. The dataset includes various cracks types, such as Horizontal Cracks, Vertical Cracks, Diagonal Cracks and Scaling. Total 2050 images of different walls are captured and 2024 images of 480*480 pixels are selected from them. These images are then further divided into “crack” (1000) and “non-crack” (1024) classes for the task of classification. The bifurcation of the dataset for the training, validation and testing is shown in Table 3. The 16-MP Canon (DSLR) camera with a focal length of 4 mm was used to capture images without zooming. Images were stored in JPG format with an average file size of near 160 KB. In this article, it is referred to as DB2/Our Dataset/New Dataset. The main characteristics of our dataset are as follows:

- Surface images (crack & non-crack) of various buildings in CSIR-CEERI, Pilani, are captured.
- The images are taken from 10 cm to 40 cm away from the surface. Distance is determined by the Bosch Centimeter Tape.
- The camera was located parallel to the surface being captured.
- The width of 0.1 mm to 12 mm of different forms of surface cracks is considered. The crack width is measured using Carbon Fiber Composites Digital Caliper.
- Granular surface images of cracks and non-cracks are also included.
- A total of five types (Vertical Cracks, Horizontal Cracks, Diagonal Cracks, Branch Cracks, Scaling) of damage images are captured.
- Few surfaces are painted white or yellow.
- All the color images are clicked.
- All images are clicked in a similar light during the day.
- On few surfaces’ sunlight is incident.

The sample images of the two datasets are shown in Fig. 2. Moreover, before using the images of both datasets for training, validation, and testing, they both have to go through the Image Pre-processing phase. This is discussed in the following section.

4.3 Image pre-processing

The Images consist of an array of pixel values. Grayscale images are a 2D array of pixels, while color images have different pixel values for each color channel, for example, blue, red, green. The pixel values are generally unsigned integer and are between 0 and 255. While these pixel values can be introduced straightforwardly to the neural network in their raw format, this can bring about difficulties during modelling, for example, being slower than the predicted model training. Instead, there could be extraordinary advantages in setting image pixel values before the neural network is fed, for example, scaling, centering, and standardization of pixel values. Figure 3 indicates the steps that are taken before the images with CNN models are introduced:

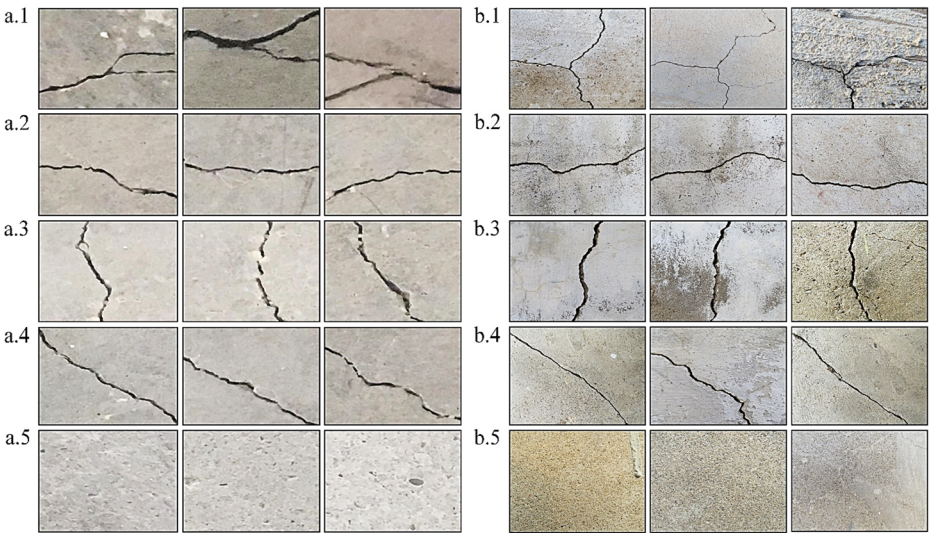


Fig. 2 Sample Images of Prepared Datasets (**a**: Selective Images of Mendeley Dataset; **b**: Our Dataset): (1) Branch Crack (2) Horizontal Crack (3) Vertical Crack (4) Diagonal Crack (5) Intact Surface

4.3.1 Image resizing

This is an important step in the field of Computer Vision. Primarily, the cost of computing can be minimized and the training of machine learning models could be improved if the model is trained on smaller images. Moreover, many machine learning models which require all images in the training dataset to be of the same size, while the captured images may be of different sizes. Image resizing is therefore required. In our case, all images are resized to 224 * 224 pixels.

4.3.2 Global standardization

In order to transform the distribution of pixel values to the Gaussian Distribution that is centering the pixel value on zero and the normalizing pixel values by dividing them with the Standard Deviation may benefit the training process. This distribution may be existing globally or per channel. In the case of Global Standardization, the mean and standard deviation is calculated across all color channels instead of a single channel. Standardization itself can fulfil the requirements of the normalization and centering the pixel values. It produces zero centered and small pixel values. For our case, global standardization per image is used for all training, validation and testing images [2].



Fig. 3 Image Pre-processing Steps

4.3.3 Clipping and rescaling pixel values

The pixel values after Global Standardization may be positive and negative, roughly lies between -3 to 3 , may vary according to the particulars of the dataset. It may be important to hold the pixel values in the positive domain so that the images can be visualized and the selected activation function of the machine learning algorithms can be facilitated. In our case, first of all, the output values of the Global Standardization are clipped to the range $[-1, 1]$ and then move the clipped values to $[0, 1]$ with a mean of 0.5 . Clipping and rescaling the performance values of Global Standardization is generally referred to as Positive Global Standardization [2].

4.4 Evaluation metrics

The performance of the CNN models can be evaluated using the Confusion Matrix. It returns the total number of false and accurate predictions made during classification. Various metrics for assessing the performance of CNN models can be defined by considering these values. In this analysis, 7 metrics are considered to evaluate and compare the damage classification performance of the CNN models, including Accuracy, Precision, Recall, F1-Score, Error Rate, ROC AUC, and Cohen Kappa scores. The value of said metrics shall be determined for each class in the models.

4.4.1 Accuracy

Classification accuracy refers to the ratio of correctly predicted crack and non-crack images to the total number of predictions made for the input images [6].

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

Where True Positive (TP) and True Negative (TN) mean correctly classified crack and non-crack images, respectively. Similarly, False Positive (FP) and False Negative (FN) are incorrectly labeled crack and non-crack images. Moreover, for the imbalanced dataset (where crack and non-crack classes have different number of image samples) the accuracy calculation might not be sufficient. The rationale behind this is that too many image samples of the majority class (or classes) would dominate the image samples of the minority class, and as a result, the poor model will achieve higher accuracy. Precision and Recall metrics are therefore required along with Accuracy.

4.4.2 Precision

In the case of an imbalanced binary classification problem, the Precision can be determined as the number of correctly predicted crack image samples divided by the total crack image samples predicted by the trained model. Similarly, it can be calculated for another class. The formula for the calculation of Precision for both classes is given as [6]:

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

The Precision value is lies between 0.0 and 1.0 . The value of 1.0 is known to be absolute Precision.

4.4.3 Recall

In the case of an imbalanced binary classification problem, the Recall can be calculated as the number of correctly predicted crack image samples divided by the total number of cracked image samples. It can also be measured for another class. The formula for the calculation of the Recall for both classes is given as [6]:

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

The value of the Recall is lies between 0.0 and 1.0. The value 1.0 is considered as the perfect Recall.

4.4.4 F1 score

It captures both Precision and Recall properties and generates a single measure to evaluate model performance. It is a better measure than the accuracy. Once the Precision and Recall are determined for the binary classification, the F1 Score can be computed as follows [6]:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

The value of F1 Score can only be 1 when Recall and Precision are 1. The high value of the F1 Score can only be reached both Recall and Precision remain high. The F1 Score is the harmonic mean of Recall and Precision.

4.4.5 Error rate

It can be easily determined by inverting the value of classification accuracy. It is often referred to as the rate of misclassification.

$$Error = 1 - \frac{TP + TN}{TP + FP + TN + FN}$$

4.4.6 Roc AUC

It uses prediction scores to compute Area Under the Receiver Operating Characteristics Curve (ROC AUC). It requires the true binary value and the positive class probability scores. Once the area has been computed under the ROC curve. It returns the single floating number that represents the area of the curve.

4.4.7 Cohen KAPPA score

It is a metric used to calculate the agreement between annotators. This function generates a single value for the classification problem that reflects the degree of agreement between the 2 annotators [6].

$$k = \frac{(p_0 - p_e)}{1 - p_e}$$

Where p_0 signifies the empirical likelihood of agreement of the label, that designates to any instance, and p_e is the anticipated agreement when the two annotators randomly assign the labels. The Cohen Kappa function returns the float value between the -1 and 1 . The value 1 is the complete agreement and the value below 1 or 0 is the chance agreement.

4.4.8 Confusion matrix

It helps to obtain the classification accuracy of the machine learning algorithm. It also briefs the performance of a classification algorithm. By calculating the confusion matrix, one can see how well the classification algorithm is doing. It displays the amount of incorrect and correct predictions made during classification by assigning to each class. Moreover, in the binary classification, the cells of confusion matrix CM are defined as follows: $CM_{0,0}$ is the count for true negative, $CM_{1,0}$ is the count for false negative, $CM_{1,1}$ serve as count for true positive and $CM_{0,1}$ is the count for False positive.

4.5 Training and testing

Training of all CNN models on both datasets is conducted on Google Colab with more than 12 GB of RAM, 32 GB of disk space and on a personal computer with the following configuration: CPU: Intel(R) Core (TM) i7-6700, RAM: 16 GB, GPU: NVIDIA Titan X (12 GB). Moreover, supporting libraries such as CUDA toolkit with the cuDNN, and OpenCV, are introduced on the Windows 10 (64 bit) operating system. Also, image pre-processing and CNN models were programmed and performed in Python3 using Colab Notebook and PyCharm IDE. The training set was designed by randomly selecting images from both datasets, holding an approximate equal number of images in both classes to train the model efficiently. The bifurcation of the datasets for training, testing and validation images is shown in Table 3.

Prior to starting the training of models through transfer learning, it is necessary to determine the value of the hyperparameters that includes: learning rate, momentum, batch size, number of epochs, and steps per epoch. These parameters have an effect on training outcomes and training time. The number of epochs is characterized as the number of times the dataset is fed to the training algorithm. If the value of this parameter is huge, it can lead to an overfitted model and high training period. Conversely, the small value of the epoch leaves the model untrained. Another essential parameter is the batch size that is characterized as the number of images given to the model at each step of the learning epoch. The value of batch size must be carefully chosen because the high value of the batch size requires high processing power. In this research, according to the volume of the datasets and specifications of Colab and personal computer, all models are trained at 50 epochs with 51 and 21 steps per epoch for DB1 and DB2 respectively, under the specified base learning rate (Tables 4 and 5). The value of the steps per epoch depends on the size of the batch and the large value of the steps may leave the model overfitted. The value of the batch size, i.e. 64, remains the same for both datasets. The value of the remaining parameters before training all models of CNN can be seen in Tables 4 and 5. Different cracked and non-cracked concrete images are provided as input for model training. Various deep learning layers, such as convolution, pooling, activation, and dropout, are applied in each model to improve the accuracy of the model; dense layers are updated according to the needs of the different model. The modified classifier of all CNN models can be seen in Tables 1, and 2. Also, during training, the model is validated for each epoch. The

Table 4 CNN Models Hyperparameters Setting for Training on DB1 Using Transfer Learning Technique

CNN Models	Train Batch Size	Number of Epochs	Steps Per Epoch	Optimizer	Learning Rate	Test Batch Size	Test Steps	Loss Function	Activation
VGG-16	64	50	51	Adadelata	–	64	16	Binary Cross-Entropy	Sigmoid
DenseNet-121	64	50	51	Adam	0.0001	64	16	Binary Cross Entropy	Sigmoid
Inception-v3	64	50	51	Adadelata	–	64	16	Binary Cross-Entropy	Sigmoid
ResNet-50	64	50	51	Adam	0.0001	64	16	Binary Cross Entropy	Sigmoid
Xception	64	50	51	Adam	0.00001	64	16	Binary Cross Entropy	Sigmoid
InceptionResNet-v2	64	50	51	Adadelata	–	64	16	Binary Cross-Entropy	Sigmoid

Table 5 CNN Models Hyperparameters Setting for Training on DB2 Using Transfer Learning Technique

CNN Models	Train Batch Size	Number of Epochs	Steps Per Epoch	Optimizer	Learning Rate	Test Batch Size	Test Steps	Loss Function	Activation
VGG-16	64	50	21	SGD	0.002	64	07	Binary Cross Entropy	Sigmoid
DenseNet-121	64	50	21	Adam	0.0001	64	07	Binary Cross Entropy	Sigmoid
Inception-v3	64	50	21	Adam	0.00001	64	07	Binary Cross Entropy	Sigmoid
ResNet-50	64	50	21	Adam	0.0001	64	07	Binary Cross Entropy	Sigmoid
Xception	64	50	21	Adam	0.00001	64	07	Binary Cross Entropy	Sigmoid
InceptionResNet-v2	64	50	21	Adam	0.0001	64	07	Binary Cross Entropy	Sigmoid

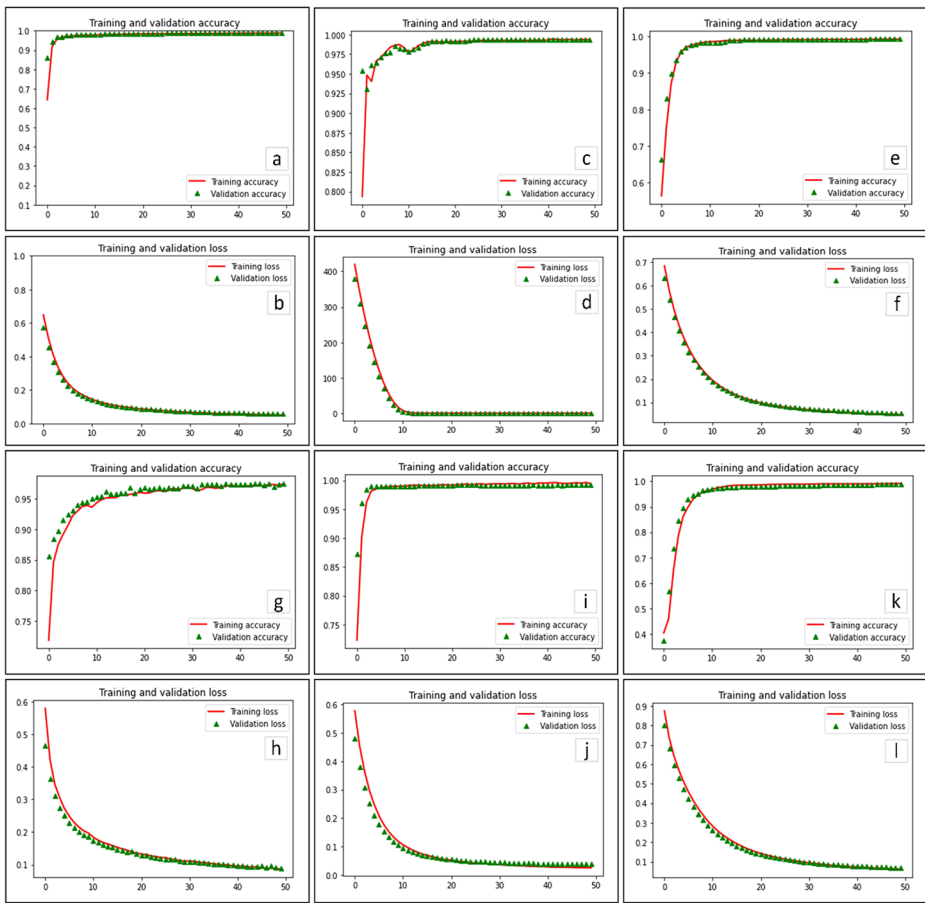


Fig. 4 Training Results of CNN Models on DB1: (a, b) Accuracy and Loss Curve of VGG-16 (c, d) Accuracy and Loss Curve of DenseNet-121 (e, f) Accuracy and Loss Curve of Inception-v3 (g, h) Accuracy and Loss Curve of ResNet-50 (i, j) Accuracy and Loss Curve of Xception (k, l) Accuracy and Loss Curve of InceptionResNet-v2

results of CNN models training on both datasets using Google Colab are shown in Figs. 4 and 5.

Soon after CNN models are validated on the test data. In the first case, the CNN models trained on the DB1 are checked on the 1002 new images taken from the Mendeley dataset (DB1). In the second case, CNN models trained on the DB2 are tested on the 405 new images from our dataset (DB2). The image from the test set is given as an input to the classifier, and after processing the input image, the model classifies the damage within a few seconds. Model has been trained to classify crack and non-crack images. During the testing, the models are evaluated against the following metrics: Accuracy Score, Precision, Recall, F1-Score, Cohen Kappa, ROC AUC, and Error Rate. The test results for both test sets are shown separately in Tables 6 and 7.

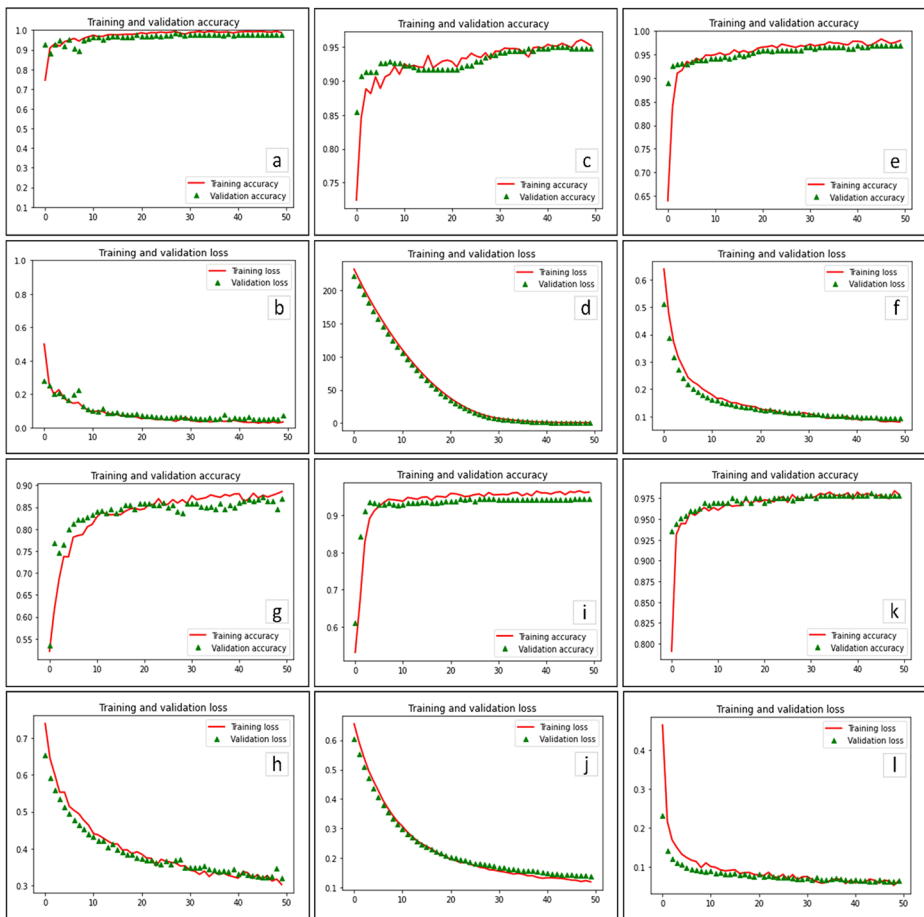


Fig. 5 Training Results of CNN Models on DB2: (a, b) Accuracy and Loss Curve of VGG-16 (c, d) Accuracy and Loss Curve of DenseNet-121 (e, f) Accuracy and Loss Curve of Inception-v3 (g, h) Accuracy and Loss Curve of ResNet-50 (i, j) Accuracy and Loss Curve of Xception (k, l) Accuracy and Loss Curve of InceptionResNet-v2

5 Result analysis

In this research, 6 CNN models are employed with transfer learning to develop classifier models based on the 2 prepared datasets. The Result Analysis section is divided into two parts. First of all, the evaluation result of CNN models after training and testing on DB1 obtained by picking selective images of the Mendeley dataset is presented. In the second part, CNN models are again assessed using DB2 which is obtained by capturing the surface images of the different buildings of CSIR-CEERI, Pilani and their results are discussed.

In addition to the performance evaluation metrics of the machine learning models, the training and testing speed of the model also plays an important role in choosing an effective pre-trained model for the concrete crack classification. The speed information for CNN models on DB1 is shown in Table 8. The comparison plot of training time could also be seen in Fig. 6.

Table 6 Performance Metrics of CNN Models Based on Indices Given by Confusion Matrix for Each Class in a Model for DB1

CNN Models	Class	Precision	Recall	F1	ROC AUC	KAPPA	Accuracy	Error Rate
VGG-16	0	0.952029	0.998065	0.974503	0.996945	0.945967	0.973053	0.026947
	1	0.997826	0.946391	0.971428				
DenseNet-121	0	0.969981	1.000000	0.984761	0.998098	0.967999	0.984031	0.015969
	1	1.000000	0.967010	0.983228				
Inception-v3	0	0.973584	0.998065	0.985672	0.998500	0.970004	0.985029	0.014971
	1	0.997881	0.971134	0.984325				
ResNet-50	0	0.917710	0.992263	0.953531	0.993113	0.899829	0.950099	0.049901
	1	0.990970	0.905154	0.946120				
Xception	0	0.986615	0.998065	0.992306	0.997595	0.984010	0.992015	0.007985
	1	0.997912	0.985567	0.991701				
InceptionResNet-v2	0	0.968164	1.000000	0.983824	0.997938	0.965996	0.983033	0.016967
	1	1.000000	0.964948	0.982161				

Table 7 Performance Metrics of CNN Models Based on Indices Given by Confusion Matrix for Each Class in a Model for DB2

CNN Models	Class	Precision	Recall	F1	ROC AUC	KAPPA	Accuracy	Error Rate
VGG-16	0	0.990000	0.980198	0.985074	0.998049	0.970369	0.985185	0.014815
	1	0.980487	0.990147	0.985293				
DenseNet-121	0	0.782945	1.000000	0.878260	0.998781	0.723644	0.861728	0.138272
	1	1.000000	0.724137	0.839999				
Inception-v3	0	0.926605	1.000000	0.961904	0.998390	0.921003	0.960493	0.039507
	1	1.000000	0.921182	0.958974				
ResNet-50	0	0.791164	0.975247	0.873613	0.969858	0.718678	0.859259	0.140741
	1	0.967948	0.743842	0.841225				
Xception	0	0.893805	1.000000	0.943925	0.994074	0.881515	0.940740	0.059260
	1	1.000000	0.881773	0.937172				
InceptionResNet-v2	0	0.975845	1.000000	0.987774	0.998610	0.975310	0.987654	0.012346
	1	1.000000	0.975369	0.987530				

Table 8 Training and Testing Time of CNN Models on DB1

CNN Models	Training Time	Training Time for Each Image (avg.)	Testing Time	Testing Time for Each Image (avg.)
VGG-16	844	0.2632	6	0.0059
DenseNet-121	558	0.1740	3	0.0029
Inception-v3	427	0.1331	2	0.0019
ResNet-50	600	0.1871	3	0.0029
Xception	884	0.2757	4	0.0039
InceptionResNet-v2	1093	0.3409	6	0.0059

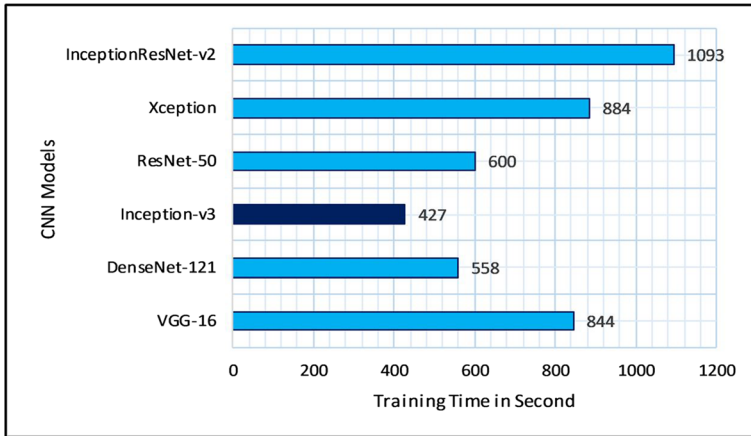


Fig. 6 Time Consumed by CNN Models During Training on DB1

Figure 6 clearly indicate that Inception-v3, DenseNet-121, and ResNet-50 are much faster than others. The probable reason for being faster is that these models are less complex than other models. As a result, Inception-v3 is the fastest model in this research and InceptionResNet-v2 is the slowest. It can also be seen in Table 8 that Inception-v3 takes the lowest time to produce output, i.e. 2 seconds. Furthermore, the performance metrics of the models mentioned in the Section 4 for the classification of concrete surface cracks are evaluated using a confusion matrix. First of all, the classification probabilities obtained from the test data of DB1 are labelled as crack if the value of the classification probabilities exceeds the set threshold, i.e. 0.8, and rest of the probabilities are marked as non-crack. Thereafter, confusion matrix presents the performance indices for each class (Fig. 7). In this analysis, 7 metrics (Accuracy, Precision, Recall, F1- Score, Cohen Kappa Score, ROC AUC, Error Rate) are calculated for each class in the models and shown in Table 6.

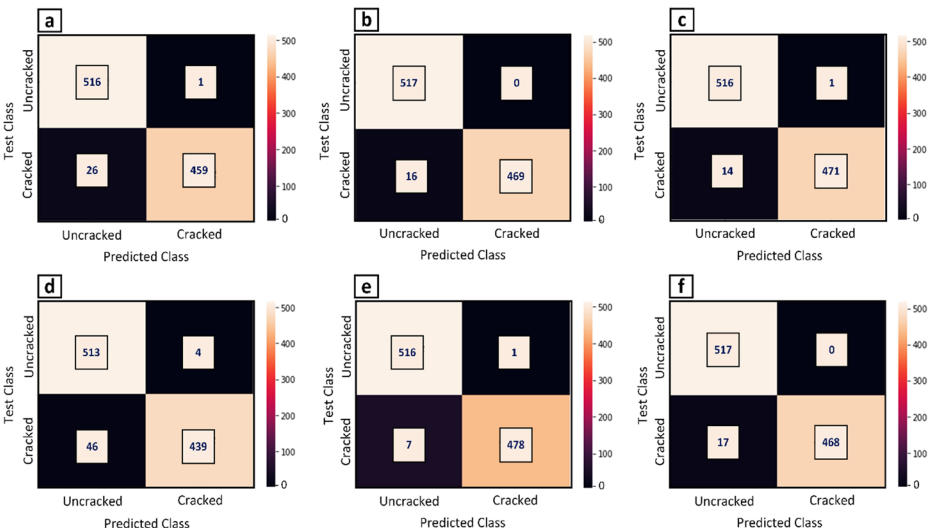


Fig. 7 Confusion Matrixes Obtained After Converting Predicted Probabilities on Testset of DB1 into Concrete Classes: (a) VGG-16 (b) DenseNet-121 (c) Inception-v3 (d) ResNet-50 (e) Xception (f) InceptionResNet-v2

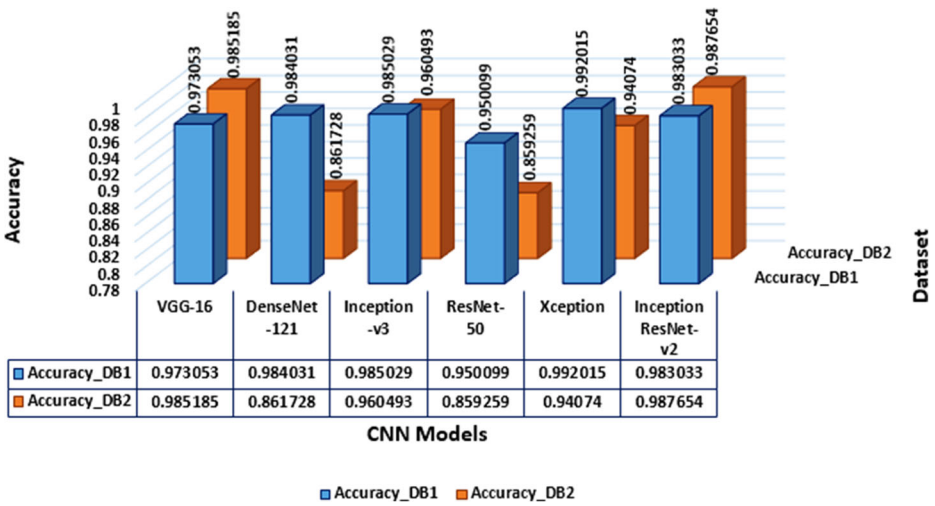


Fig. 8 Accuracy of CNN Models on Both Datasets

It is clearly depicted in Table 6 and Fig. 8; the overall accuracy score is between 0.95 and 1 for 6 CNN models. After comparing the accuracy score of all CNN models, it was determined that the Xception and Inception-v3 models demonstrated good performances compared to the rest of the models with 0.992015 and 0.985029 accuracy values, respectively. While ResNet-50 could have obtained the lowest accuracy value i.e. 0.950099.

The Recall value of the crack class in the trained models is between 0.90 and 0.99, and the non-crack class has the recall value across all models in the range of 0.99–1.0. The Xception and Inception-v3 have the highest Recall value for crack class i.e. 0.985567 and 0.971134, respectively. While DenseNet-121 and InceptionResnet-v2 have the same highest value for non-crack class i.e. 1.0. The ResNet-50 has received the lowest value of the Recall for both classes.

Table 6 also reveals that the precision of the crack class is higher than the non-crack class across all the models. It implies that, whenever the result of the classifier model is cracked, the outcome can be acknowledged with more confidence. As far as precision is concerned, DenseNet-121 and InceptionResNet-v2 have the best presentation for crack classification with 1.0 precision for both models.

The next metric, i.e. F1 generate a balance score by giving equal privilege to both Precision and Recall. Table 6 illustrate that the non-crack class has the most elevated F1-score in the entirety of the models with a range of 0.95 to 1.0. The best performance of the classification of concrete cracks is observed in the Xception model with an F1-Score estimate of 0.991701 and 0.992306 for crack and non-crack class respectively. The most remarkably awful performance is also associated with ResNet-50.

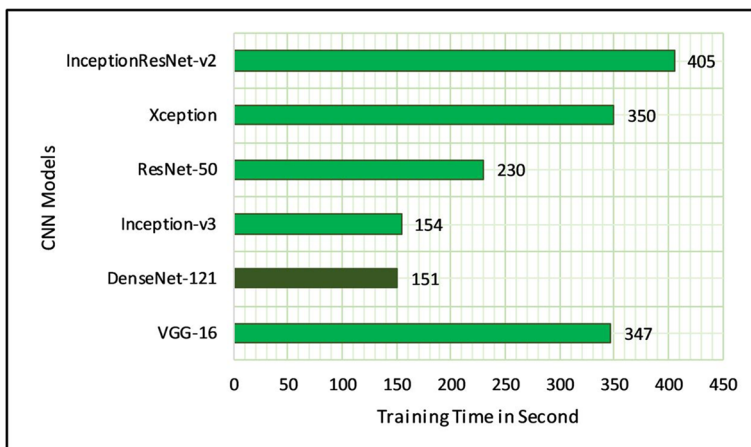
Two more metrics, i.e. Cohen Kappa and ROC AUC are determined on the basis of the complete predicted results. When calculating the ROC AUC, the ground truth test values and the predicted probabilities on the test data are passed to the function, whereas ground truth test values and the predicted concrete class values obtained after converting the predicted probabilities by setting probabilities values greater than 0.8 to 1 (crack) and rest to 0 (non-crack) are passed to the Cohen Kappa function. It is evident from Table 6 that the best ROC AUC and Cohen Kappa score are obtained by Inception-v3 and Xception, for example, 0.998500 and 0.984010 respectively.

Table 9 Training and Testing Time of CNN Models on DB2

CNN Models	Training Time	Training Time for Each Image (avg.)	Testing Time	Testing Time for Each Image (avg.)
VGG-16	347	0.2679	1	0.0024
DenseNet-121	151	0.1166	1	0.0024
Inception-v3	154	0.1189	1	0.0024
ResNet-50	230	0.1776	1	0.0024
Xception	350	0.2702	1	0.0024
InceptionResNet-v2	405	0.3127	2	0.0049

The performance assessment of the CNN models on DB2 that is built by collecting the concrete surface images (Crack & Non-Crack) of different campus buildings of CSIR-CEERI, Pilani, is presented below. The model's training and testing speed based on the DB2 can be seen in Table 9. The comparison plot of training time can also be seen in Fig. 9. Figure 9 clearly indicate that DenseNet-121, Inception-v3, and ResNet-50 are much faster than others. The training performance of CNN models on DB2 is almost same as DB1. It is also noticeable in Table 9, that all models have the same test time, i.e. 1 second except InceptionResNet-v2, which takes 2 seconds to produce the output.

Moreover, the classification probabilities obtained from the test data of DB2 are labelled as crack if the value of the classification probabilities exceeds the set threshold i.e. 0.8 and rest of the probabilities are marked as non-crack. Soon after the confusion matrix has been obtained and all the metrics discussed in Section 4 are calculated by the performance indices provided by the confusion matrix for each class in the models (Fig. 10). InceptionResNet-v2 outperforms the rest of the CNN variants with an accuracy of 0.987654. The accuracy value clearly depicts that InceptionResNet-v2 correctly classifies the damage images. F1 Score is also at the top in support of Accuracy score with 0.987774 for non-crack class and 0.987530 for crack class. Moreover, the ROC AUC and Cohen Kappa values of InceptionResNet-v2 lead the rest of the models except DenseNet-121 that showed slightly higher ROC AUC value. All of the evaluated metrics after training and testing models on DB2 could be seen in Table 7 and Fig. 8.

**Fig. 9** Time Consumed by CNN Models During Training on DB2

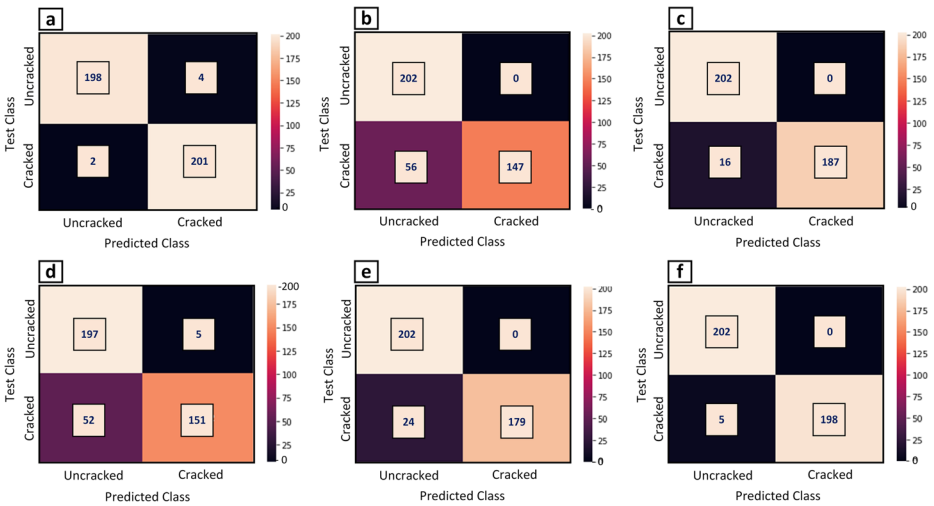


Fig. 10 Confusion Matrixes Obtained After Converting Predicted Probabilities on Testset of DB2 into Concrete Classes: (a) VGG-16 (b) DenseNet-121 (c) Inception-v3 (d) ResNet-50 (e) Xception (f) InceptionResNet-v2

In summary, after going through all of the performance metrics, it can be perceived that, apart from the number of trainable parameters, the training speed of the CNN models also depends on the different settings for hyperparameters such as learning rate, momentum, size of dataset, etc. If the learning rate value is high, it takes less time for the algorithm to converge, while in the case of low learning rate, the algorithm takes more time to converge. High momentum value helps speed up the training of CNN models. In addition, the large volume of data makes model training costly and time consuming. Moreover, it can be believed that even the problem is the same (cracks classification), but different models can be suited to different datasets. The most probable explanation may be that the applicability of the model depends on the characteristics of the dataset, such as the size of dataset, the distribution of the data, patterns in data, etc. The outcome of the models depends significantly on how well they learn the distribution of data. If the data feed to the models is changed, the results of the model will also be changed. Therefore, the performance comparison of models can only be made with the same dataset.

6 Conclusion

Classification and detection of concrete surface cracks is one of the most challenging tasks and core concern of Structural Health Monitoring. In this paper, a comparative study of Automated Structural Health Monitoring based on models of deep Convolution Neural Network (CNN) was proposed to preserve and update records of the strength and resilience of concrete structures. CNNs are a specific category of deep neural networks that have been applied for various tasks, such as extraction of features, prediction, classification, and so forth. The proposed study contributes to the automated classification of cracks on the concrete surface by training the deep learning models with cracked and non-cracked areas. Cracks of less than 12 mm and more than 0.1 mm are effectively classified by our trained models. For crack classification, different models of deep CNN, such as VGG-16, Inception-v3,

InceptionResNet-v2, ResNet-50, Xception, and DenseNet-121, have been re-trained by the transfer learning technique, which is tuned for two categories, i.e. cracked and non-cracked. Each model has been deployed on two separate image datasets. The first dataset is Mendeley dataset (this dataset is easily accessible on the internet consisting of different types of defect images) and the subsequent one is our dataset, which is created by collecting different types of cracks and non-cracks images by wandering within the CSIR-CEERI campus in Pilani. Before introducing the images of these datasets to the CNN models, all images are pre-processed, thereby minimizing training time and other modelling difficulties. The performance of CNN models was assessed by training time, testing time, and crack classification performance. In view of training and testing time, Inception-v3, and DenseNet-121 have showed decent performance over other models on both datasets. In this examination, seven performance metrics were used to evaluate and analyze the effectiveness of crack classification models, including Accuracy Score, Precision, Recall, F1-Score, Cohen Kappa, Error Rate, and ROC AUC. As indicated by performance metrics, Xception and InceptionResNet-v2 have, for the most part, improved performance compared to the others on DB1 and DB2 respectively.

Moreover, the accuracy of each model is compared on the same dataset, and the most accurate model of each dataset is compared to each other (Fig. 8). When different models have been validated and tested on the first dataset i.e. Mendeley dataset, the recorded accuracy of VGG-16 is 0.97, Inception-v3 is 0.98, InceptionResNet-v2 is 0.98, ResNet-50 is 0.95, Xception is 0.99, and DenseNet-121 is 0.98. The classifier yielding highest accuracy is Xception with an accuracy score of 99%. Similarly, when applied to a second dataset, i.e. our dataset VGG-16 and InceptionResnet-v2 shows the superiority at an accuracy level of 98%. When the overall results are compared with each other Xception is proved to be the best model with an accuracy of 99% for the concrete image classification in our research, which can classify cracks on the concrete surface precisely. However, the overall comparison cannot be made since the both problems are different in terms of the characteristics of dataset. In addition, the findings have shown that re-training CNN models using transfer learning is an effective strategy for concrete crack classification with an accuracy range of 0.85 to 0.99 in the overall models' performance on both the datasets. Future research should concentrate on checking the applicability of superior models to concrete wall images taken using drones instead of taking pictures manually. Moreover, the most reliable model can be programmed into portable hardware or as a mobile device application.

Acknowledgements The work of Prashant Kumar was supported in part by the All India Council of Technical Education, New Delhi, India, and in part by the Indian National Academy of Engineering (INAE), Gurgaon, India.

Funding This project does not have any funding.

Data availability DB1: <https://data.mendeley.com/datasets/5y9wdsg2zt/2> DOI: <https://doi.org/10.17632/5y9wdsg2zt.2>

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Athanasiou A, Ebrahimkhanlou A, Zaborac J, Hrynyk T, Salamone S (2020) A machine learning approach based on multifractal features for crack assessment of reinforced concrete shells. *Comput Civ Infrastruct Eng* 35(6):565–578. <https://doi.org/10.1111/micc.12509>
2. Brownlee J (2019) How to Manually Scale Image Pixel Data for Deep Learning, *Machine Learning Mastery* <https://machinelearningmastery.com/how-to-manually-scale-image-pixel-data-for-deep-learning/#:~:text=Pixel.values are often unsigned,expected training of the model>
3. Cho S, Kim B, Kim G (2019) Application of deep learning-based crack assessment technique to civil structures, Fifth conference on Smart Monitoring, Assessment and Rehabilitation of Civil Structures (SMAR-2019), pp. 1–8
4. Christian Koch PF, Georgieva K, Kasireddy V, Akinci B (2015) A review on computer vision based defect detection and condition assessment of concrete and asphalt civil infrastructure. *Adv Eng Inform* 29(2):196–210. <https://doi.org/10.1016/j.aei.2015.01.008>
5. de Lucena DS, da Silva WRL (2018) “Concrete cracks detection based on deep learning image classification,” *MDPI Proc*, <https://doi.org/10.3390/ICEM18-05387>
6. Developers S-L (2020) Metrics and scoring: quantifying the quality of predictions,” *Scikit Learn*, https://scikit-learn.org/stable/modules/model_evaluation.html#classification-metrics
7. Dorafshan S, Thomas RJ, Maguire M (2018) Comparison of deep convolutional neural networks and edge detectors for image-based crack detection in concrete. *Constr Build Mater* 186:1031–1045. <https://doi.org/10.1016/j.conbuildmat.2018.08.011>
8. Dung CV, Anh LD (2019) Autonomous concrete crack detection using deep fully convolutional neural network. *Autom Constr* 99:52–58. <https://doi.org/10.1016/j.autcon.2018.11.028>
9. Gao Y, Mosalam KM (2018) Deep transfer learning for image-based structural damage recognition. *Comput Civ Infrastruct Eng* 33(9):748–768. <https://doi.org/10.1111/micc.12363>
10. Hoang N-D (2018) Image processing-based recognition of wall defects using machine learning approaches and steerable filters. *Comput Intell Neurosci* 2018:1–18. <https://doi.org/10.1155/2018/7913952>
11. Huang H, Li Q, Zhang D (2018) Deep learning based image recognition for crack and leakage defects of metro shield tunnel. *Tunn Undergr Sp Technol* 77:166–176. <https://doi.org/10.1016/j.tust.2018.04.002>
12. Islam MMM, Kim J-M (2019) Vision-based autonomous crack detection of concrete structures using a fully convolutional encoder–decoder network. *Sensors* 19:1–12. <https://doi.org/10.3390/s19194251>
13. Jung NLWM, Naveed F, Hu B, Wang J (2019) Exploitation of deep learning in the automatic detection of cracks on paved roads, *Geomatica*
14. Kim B, Cho S (2018) Automated Vision-Based Detection of Cracks on Concrete Surfaces Using a Deep Learning Technique. *Sensors* 18(10):3452. <https://doi.org/10.3390/s18103452>
15. Kim A-R, Byun Y-S, Chun C, Kim D, Lee S-W (2019) Automated concrete crack detection and using deep learning and image processing method, *Adv Struct Eng Mech*
16. Kim H, Ahn E, Shin M, Sim S-H (2019) Crack and noncrack classification from concrete surface images using machine learning. *Struct Health Monit* 18(3):725–738. <https://doi.org/10.1177/1475921718768747>
17. Li S, Zhao X (2019) Image-based concrete crack detection using convolutional neural network and exhaustive search technique. *Adv Civ Eng* 2019:1–12. <https://doi.org/10.1155/2019/6520620>
18. Liu H, Zhang Y (2019) Image-driven structural steel damage condition assessment method using deep learning algorithm. *Measurement* 133:168–181. <https://doi.org/10.1016/j.measurement.2018.09.081>
19. Moon H-G, Kim J-H (2011) Intelligent Crack Detecting Algorithm on the Concrete Crack Image Using Neural Network, in 28th International Symposium on Automation and Robotics in Construction, pp. 1461–1467. <https://doi.org/10.22260/ISARC2011/0279>
20. Özgenel ÇF, Sorguç AG (2018) Performance comparison of pretrained convolutional neural networks on crack detection in buildings. In: *Proceedings of the International Symposium on Automation and Robotics in Construction (IAARC)*. <https://doi.org/10.22260/isarc2018/0094>
21. Pan SJ, Yang Q (2010) A survey on transfer learning. *IEEE Trans Knowl Data Eng* 22(10):1345–1359. https://doi.org/10.1007/978-981-15-5971-6_83
22. Perez H, Tah JHM, Mosavi A (2019) Deep Learning for Detecting Building Defects Using Convolutional Neural Networks. *Sensors* 19(16):3556. <https://doi.org/10.3390/s19163556>
23. Qian Y et al. (2019) Fresh Tea Leaves Classification Using Inception-V3, 2019 2nd IEEE Int. Conf Inf Commun Signal Process ICICSP 2019, pp. 415–419, <https://doi.org/10.1109/ICICSP48821.2019.8958529>.
24. Qu Z, Chen Y-X, Liu L, Xie Y, Zhou Q (2019) The algorithm of concrete surface crack detection based on the genetic programming and percolation model. *IEEE Access* 7:57592–57603. <https://doi.org/10.1109/ACCESS.2019.2914259>
25. Raj APSS, Vajravelu SK (2019) DDLA: dual deep learning architecture for classification of plant species. *IET Image Process* 13(12):2176–2182. <https://doi.org/10.1049/iet-ipr.2019.0346>

26. Rajeshwari M, Rathika K (2018) Detection of roof holes and wall crack using shape-based method. SSRG Int J Comput Sci Eng 5(5):6–10. <https://doi.org/10.14445/23488387/IJCSE-V5I5P102>
27. Sagar RV (2019) Support vector machine procedure and Gaussian mixture modelling of acoustic emission signals to study crack classification in reinforced concrete structures
28. Shao L, Zhu F, Li X (2015) Transfer learning for visual categorization: a survey. IEEE Trans Neural Networks Learn Syst 26(5):1019–1034. <https://doi.org/10.1109/TNNLS.2014.2330900>
29. Shi C, Xia R, Wang L (2020) A novel multi-Branch Channel expansion network for garbage image classification. IEEE Access 8:154436–154452. <https://doi.org/10.1109/ACCESS.2020.3016116>
30. Simler C, Trostmann E, Berndt D (2019) Automatic crack detection on concrete floor images, in Photonics and Education in Measurement Science 2019, p. 41. <https://doi.org/10.1117/12.2531951>.
31. Sitara RGS, Kavitha NS (2018) Review and analysis of crack detection and classification techniques based on crack types. Int J Appl Eng Res 13(8):6056–6062
32. Słoński M (2019) A comparison of deep convolutional neural networks for image-based detection of concrete surface cracks. Comput Assist Methods Eng Sci 26:105–112. <https://doi.org/10.24423/comes.267>
33. Wang S, Yang F, Cheng Y, Yang Y, Wang Y (2018) Adaboost-based Crack Detection Method for Pavement. IOP Conf Ser: Earth Environ Sci 189:022005. <https://doi.org/10.1088/1755-1315/189/2/022005>
34. Wang B, Li Y, Zhao W, Zhang Z, Zhang Y, Wang Z (2019) Effective Crack Damage Detection Using Multilayer Sparse Feature Representation and Incremental Extreme Learning Machine. Appl Sci 9(3):614. <https://doi.org/10.3390/app9030614>
35. Wu X, Xu H, Wei X, Wu Q, Zhang W, Han X (2020) Damage identification of low emissivity coating based on convolution neural network. IEEE Access 8:156792–156800. <https://doi.org/10.1109/ACCESS.2020.3019484>
36. Xia X, Xu C, Nan B (2017) “Inception-v3 for flower classification,” 2017 2nd Int. Conf. Image, Vis. Comput. ICIVC 2017, pp. 783–787, <https://doi.org/10.1109/ICIVC.2017.7984661>.
37. Xu H, Su X, Wang Y, Cai H, Cui K, Chen X (2019) Automatic Bridge Crack Detection Using a Convolutional Neural Network. Appl Sci 9(14):2867. <https://doi.org/10.3390/app9142867>
38. Xu G, Shen X, Chen S, Zong Y, Zhang C, Yue H, Liu M, Chen F, Che W (2019) A deep transfer convolutional neural network framework for EEG signal classification. IEEE Access 7:112767–112776. <https://doi.org/10.1109/access.2019.2930958>
39. Zhang L, Yang F, Daniel Zhang Y, Zhu YJ (2016) Road crack detection using deep convolutional neural network, in 2016 IEEE International Conference on Image Processing (ICIP), pp. 3708–3712. <https://doi.org/10.1109/ICIP.2016.7533052>.
40. Zhang K, Zhang Y, Cheng H-D (2019) CrackGAN: pavement crack detection using partially accurate ground truths based on generative adversarial learning, IEEE Trans Intell Transp Syst, <https://doi.org/10.1109/TITS.2020.2990703>
41. Zhang J, Lu C, Wang J, Wang L, Yue X-G (2019) Concrete Cracks Detection Based on FCN with Dilated Convolution. Appl Sci 9(13):2686. <https://doi.org/10.3390/app9132686>
42. Zhu J, Zhang C, Qi H, Lu Z (2020) Vision-based defects detection for bridges using transfer learning and convolutional neural networks. Struct Infrastruct Eng 16(7):1037–1049. <https://doi.org/10.1080/15732479.2019.1680709>

Publisher’s note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.