



Surface crack detection using deep learning with shallow CNN architecture for enhanced computation

Bubryur Kim¹ · N. Yuvaraj¹ · K. R. Sri Preethaa² · R. Arun Pandian³

Received: 24 February 2020 / Accepted: 5 January 2021 / Published online: 23 January 2021
© The Author(s), under exclusive licence to Springer-Verlag London Ltd. part of Springer Nature 2021

Abstract

Surface cracks on the concrete structures are a key indicator of structural safety and degradation. To ensure the structural health and reliability of the buildings, frequent structure inspection and monitoring for surface cracks is important. Surface inspection conducted by humans is time-consuming and may produce inconsistent results due to the inspectors' varied empirical knowledge. In the field of structural health monitoring, visual inspection of surface cracks on civil structures using deep learning algorithms has gained considerable attention. However, these vision-based techniques require high-quality images as inputs and depend on high computational power for image classification. Thus, in this study, shallow convolutional neural network (CNN)-based architecture for surface concrete crack detection is proposed. LeNet-5, a well-known CNN architecture, is optimized and trained for image classification using 40,000 images in the Middle East Technical University (METU) dataset. To achieve maximum accuracy for crack detection with minimum computation, the hyperparameters of the proposed model were optimized. The proposed model enables the employment of deep learning algorithms using low-power computational devices for a hassle-free monitoring of civil structures. The performance of the proposed model is compared with those of various pretrained deep learning models, such as VGG16, Inception, and ResNet. The proposed shallow CNN architecture was found to achieve a maximum accuracy of 99.8% in the minimum computation. Better hyperparameter optimization in CNN architecture results in higher accuracy even with a shallow layer stack for enhanced computation. The evaluation results confirm the incorporation of the proposed method with autonomous devices, such as unmanned aerial vehicle, for real-time inspection of surface crack with minimum computation.

Keywords Deep learning · Crack detection · Convolution neural network · Computer vision · Concrete structures

1 Introduction

Structures, such as bridges, tall buildings, and highways, deteriorate with time, which causes damage to the health of the structures [1]. Structural safety remains challenging in all infrastructures due to various internal and environmental parameters [2, 3]. Ensuring structural safety of the

infrastructure plays a significant role in structural maintenance, and it is exponentially related to the age of the infrastructure [3, 4]. Building inspection for damages is important to maintain the safety of its residents [6]. The lack of inspection guidance and the inadequate use of available technologies for structural health monitoring (SHM) reduce the safety and durability of the infrastructure [7]. However, to minimize the loss of the lives of people and property, the inspection methodologies for the conditional assessment of the infrastructure need to be improved [8]. Continuous structure monitoring for damages is important to improve the health and reliability of the structures.

Among the various parameters that ensure structural health, cracks remain as a key parameter directly indicating the condition of the structures [9]. Cracks exhibit the magnitude of the structural damage. Periodical monitoring

✉ N. Yuvaraj
yuvaraj.n@kpri.ac.in

¹ Department of Architectural Engineering, Dong-A University, Busan 49315, South Korea

² Department of Artificial Intelligence and Data Science, KPR Institute of Engineering and Technology, Coimbatore 641407, India

³ Artificial Intelligence and Data Science Lab, KPR Institute of Engineering and Technology, Coimbatore 641407, India

of building surfaces for cracks and taking measures for restoration are necessary to improve the durability of the building. Continuous and effective monitoring for cracks is greatly important to ensure structural safety. The analysis of a train accident in Japan in 1999, which resulted in the loss of the lives of people and property, reveals that shear cracks on concrete blocks were the root cause of the accident [10]. Statistical analysis shows that 46% of the structural weakness is only due to the failure to detect structural cracks [11]. To prevent such accidents, continuous autonomous inspection of civil infrastructures should be carried out [12].

Conventionally, the method of human visual inspections was employed to detect cracks. However, this method has limitations in terms of human experience, time consumption, and risk of inspection in inaccessible areas [13]. To enhance the accuracy of monitoring, the health vibration-based structural system identifications were applied [14]. In SHM, numerical method conjugations have been widely adopted [15, 16]. Monitoring of large-scale civil infrastructures, structural uncertainties, and environmental disturbances are the main challenges associated with the vibration-based approach [17, 18]. The computer vision and image processing techniques have become an integral part of SHM compared with the traditional manual inspection-based crack detection system [19, 20]. Computer vision-based approaches are employed to provide rich hidden insights out of the image and video data which facilitate in the automation of crack detection and identification systems [21–23]. The detection systems were mainly built using common image processing methods, such as segmentation, pattern recognition, fuzzy clustering [24], histogram analysis, Fourier transform [25], and image filtering [26] techniques. However, these traditional methods failed to accurately handle noisy image data.

To further improve the accuracy, researchers have initially used various edge detection algorithms [27]. The Canny edge detector, wavelet method, and Laplace of Gaussian [28, 29] were employed for edge detection in concrete crack identification. Among the various edge detection algorithms, wavelet method was mostly used in the literature [30]. Abdel-Qader et al. [31] considered and compared four edge detection algorithms, namely, the Haar transform, fast Fourier transform, Sobel, and Canny. A histogram-based method for feature extraction and fuzzy logic for crack classification were proposed. However, the outcome of the algorithm emphasized the need for accuracy improvement [32]. Researches using the Canny and k-means clustering techniques were conducted to detect surface cracks [33]. However, these traditional algorithms were unsuccessful in effectively processing a huge volume of concrete image data, which paved the way for machine learning-based approaches for crack detection [34].

Initially, support vector machine (SVM) [35] was employed for the binary classification of concrete images into to crack and non-crack via handcrafted manual feature extraction. Moreover, principal component analysis (PCA) [36] was used as a dimensionality reduction technique for extracting the healthy features out of a large set of features. SVM with PCA exhibited a significant improvement in the classification. To enhance the accuracy of crack identification, other classification algorithms, such as artificial neural network (ANN) [37], k-nearest neighbors (k-NN), fuzzy logic, and genetic [38], were implemented as a hybrid approach with SVM. Aside from classical machine learning algorithms, deep learning techniques were successfully implemented in computer vision, pattern recognition image classification, and object detection. The recent developments of computer vision techniques have resulted in end-to-end learning using ANNs [39] and convolutional neural networks (CNNs). ANNs and CNNs are capable of relating the complex input and output relationship of data using parameterized nonlinear activation functions. CNNs were employed to effectively classify the low-level features of biological microscopy images with high accuracy [40]. CNNs with machine learning methods were implemented to classify the crack and non-crack images from the video frames [41]. A CrackNet model without pooling layers was proposed to improve the accuracy of crack detection [32]. The combination of CNN and the sliding window technique was designed to classify the crack and non-crack images more accurately over IPMs [42]. To detect multiple crack damages, a faster region-based CNN (faster R-CNN) was adopted [42].

The ability of deep learning (DL) techniques to handle huge volume of data and to automate the feature extraction processes enables the DL techniques to develop solutions for crack detection problems on concrete structures [43]. CNN combined with Naïve Bayes was employed to analyze individual video frames for crack detection [44]. Moreover, CNN models were implemented for various crack detection applications, such as pavement crack detection [45], cracks on asphalt surfaces [46], automatic road crack detection [47], cracks on mechanical steel structures [48], damage detection, steel corrosion, bolt corrosion, and steel delamination [49]. Due to the advancements in hardware and computational software, deep convolutional neural networks (DCNNs) were widely used for concrete crack image classification and object detection [50]. In the literature, deeper CNN architectures were designed to handle images of sizes $99 \times 99 \times 3$ [19], $256 \times 256 \times 3$ [40], and $227 \times 227 \times 3$ [11].

Vision-based techniques depend on continuous motion videos, which exponentially increase the volume of data to be handled. Thus, these techniques remain a tough task. When the data volume increases, interpreting the inner

working of DCNN becomes difficult owing to its black box nature [51]. A building damage assessment model was established based on post-hurricane satellite images using the CNN architecture [52]. Researches have used many other CNN models, such as AlexNet [53], ResNet [22], VGG16 [54], and Inception [5], for crack identification. VGG16 is the most commonly used pretrained model deeply stacked with a 16-layer depth. ResNet is a residual architecture with a 152-layer depth, whereas Inception is a parallel stacked convolutional network with a 159-layer depth. To obtain accuracy, these pretrained models require a huge volume of high-quality data and high computational power.

Although the use of deep CNN layers can better extract more information from an image, overfitting to training data and vanishing gradients are the major challenges associated with deep CNN architectures. Due to these challenges, deep CNN architectures render the model void and unusable. More robust CNN architectures, such as VGG16, ResNet, and Inception, are computationally expensive to train and deploy during real-time analysis. Moreover, the use of very deep and advanced architectures is overrated for our surface crack binary classification as it they involve simple image features.

The enhancement of the accuracy of CNN models can be achieved by optimizing their hyperparameters [54]. The selection of hyperparameters for optimization in deep architectures becomes challenging due to the exponential increase in the number of parameters and the requirement of deep structures for complex tasks [55]. In addition, an efficient training of deep CNN architecture requires powerful hardware resources. Deep CNN models depend on Graphic Processing Unit (GPU), Tensor Processing Unit (TPU), Computer Unified Device Architecture (CUDA), and other servers to handle the large volume of data. A backpropagation learning algorithm using multiplexing layer scheme was attempted on a field-programmable gate array (FPGA) board to achieve more learning on low-performance devices [56]. The Internet of things (IoT) enabled devices, such as unmanned aerial vehicles (UAVs), which are used for autonomous monitoring, to handle only a minimum quantity of data due to their limited computational power [57]. Applying deep architecture in UAVs is practically impossible. Hence, the implementation of the deep CNN models in all applications involving UAV for live data monitoring is not feasible. Therefore, the deployment of such architectures in embedded and smart devices remains challenging.

The original LeNet-5 proposed by LeCun [4] in 1998 is capable of doing most of the image classifications that do not involve complex image features. Initially, LeNet-5 was used for handwritten digit recognition. Since the establishment of the LeNet-5 architecture, new functions and

design changes have been developed and have proven to be very robust inclusions in the newer architectures. To improve the accuracy of the LeNet-5 architecture for image classification, an Optimized LeNet (OLeNet) was proposed in this work. OLeNet infuses new functions, such as rectified linear unit (ReLU), dropout, and adaptive moment estimation (Adam) algorithm, to LeNet-5 to train the best accurate and possible hardware-efficient model. OLeNet can be trained using a small dataset and with relatively faster training time. It is capable of making fast predictions with high accuracy in a low-power computational device. The OLeNet architecture can also overcome the challenges associated with deep CNN architecture owing to its minimum layer stack [58].

The remainder of this paper is organized as follows. Section presents the overview of the proposed work. Section 3 describes the architecture of different pretrained models. Section 4 discusses the technical work of the proposed shallow OLeNet architecture. Finally, Sect. 5 provides the results and discussions.

2 Overview of the proposed model

In this section, the overview of the proposed OLeNet CNN model is summarized. Optimization of the hyperparameters with the shallow LeNet-5 architecture results in the establishment of OLeNet. Since OLeNet utilizes low-power computational resources, it is greatly suitable for IoT applications. LeNet-5 is the first core architecture of CNN with very simple and shallow layer arrangements. It provides the basic architecture over which the deep CNN models were developed. The LeNet-5 architecture exhibits 98.5% accuracy in the classification of the MNIST database with hand written digits [26]. Moreover, it utilizes minimum computational power, which enables its deployment on normal machines. The layer stack arrangement of LeNet-5 is loosely coupled, which enables the model to configure the hyperparameters easily according to the applications [55]. Since concrete crack images are not diversified images, depending on very deep architectures for crack classification is not necessary.

The present study proposes an OLeNet architecture by fine-tuning the hyperparameters of the LeNet-5 architecture for the detection of surface cracks on concrete structures. The proposed architecture is capable of processing even low-quality images of 50×50 pixels. This enables OLeNet to effectively work with IoT devices without depending on the GPU, unlike all the other existing deep architectures. First, the proposed model is trained for low-quality image classification on a benchmark crack dataset. Second, using a new set of images, the performance of the proposed OLeNet architecture is evaluated. Finally, the

performance is validated by comparing the proposed model (OLeNet) with other deep pretrained models, namely, VGG16 [25], Inception [55], and ResNet [56].

IoT devices employed for the SHM of the building are proposed to be used with the shallow OLeNet architecture (Fig. 1a). Owing to its shallow architecture, OLeNet is capable of running low-power computational devices. The configuration of the OLeNet architecture (Fig. 1b) is performed by fine-tuning the optimum hyperparameters of the core LeNet-5 architecture. The entire dataset is divided into training and testing data. To train and validate the proposed CNN model, the entire high-quality images are resized into minimum sizes of 50×50 pixels. Then, the images are converted into an array of numbers of the size 50×50 .

3 Pretrained convolutional neural networks for crack identification

A fully connected pretrained DCNN is employed to classify the input images into crack and non-crack [5–8]. In the literature [2–5], the performance of the pretrained CNN models was evaluated on the open-source dataset obtained from multiple buildings in the Middle East Technical University (METU). Table 2 presents the details of the METU dataset. In this section, the basic process of CNN and other pretrained models used for the validation of the proposed work is summarized.

3.1 Convolutional neural network architecture

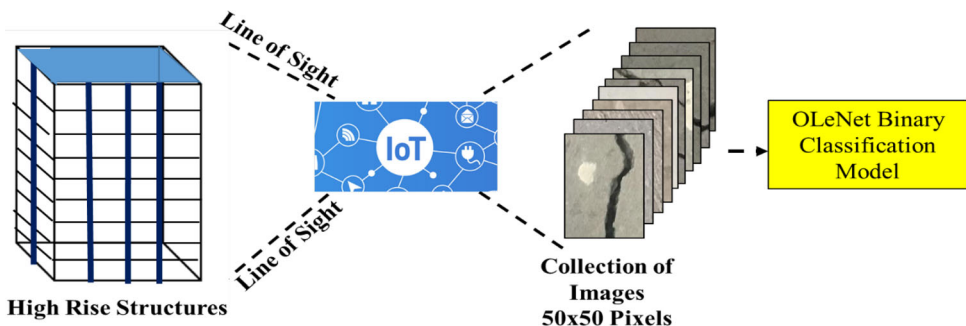
CNN is a deep neural architecture that is widely applied for image classification. It is made up of convolutional layers for analyzing the features out of the input and connected layers for the classification process. Each convolutional layer is configured with a convolutional layer, activation function, and a pooling layer [40, 41]. To capture the correlation between the input features, CNN employs a set of connected kernel filters. These filters form the core part of the CNN which performs the convolutional operation with the output of the preceding layers. Figure 2 demonstrates the operating principle of the CNN architecture.

The mathematical relationship for the convolution in each location of P_y of the output y is expressed as follows:

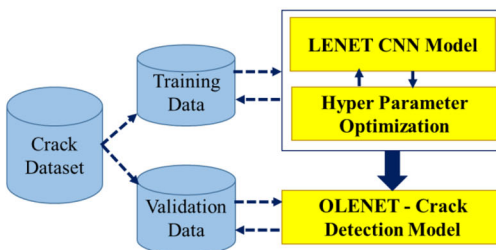
$$y(P_y) = \sum w(P_G).x(P_y + P_G) \tag{1}$$

where x denotes the input variable; w , the filter; G , the field in the convolutional layer; and P_G , the positions in the field G . The activation function improves the nonlinearity in the model. Some of the activation functions most commonly used by CNN are sigmoid, ReLU, and Tanh. The inputs of a 2D CNN layer can be visualized as multiple 2D matrices with different channels based on the image representations. The convolutional layer is enabled with multiple filters capable of scanning the inputs and producing the output mappings. If there are M inputs and N outputs, then it is important to have $M \times N$ filters to perform the convolutional operations.

Fig. 1 Overview of the proposed model



(a) The line of site of IoT devices and OLeNet process



(b) Proposed OLeNet process model

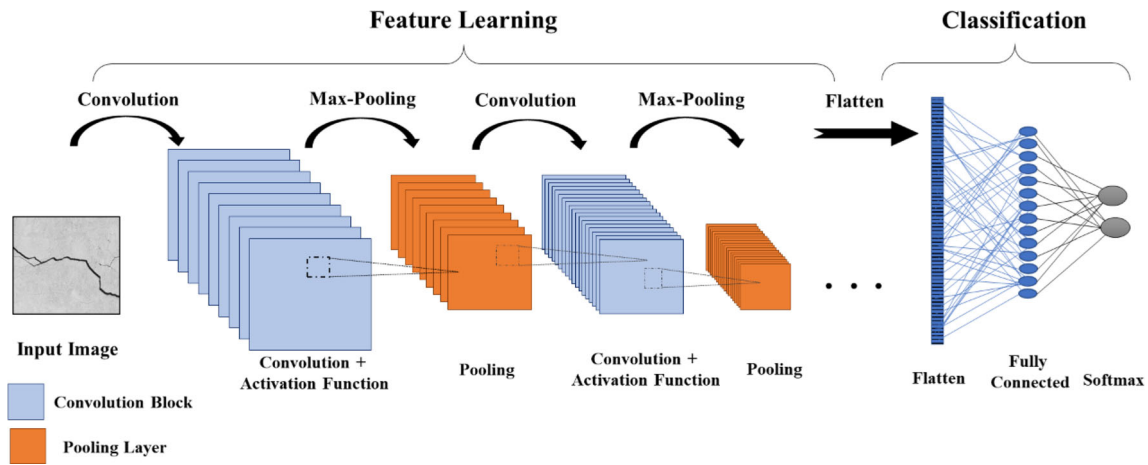


Fig. 2 Operating principle of the CNN architecture

3.2 LeNet-5 architecture

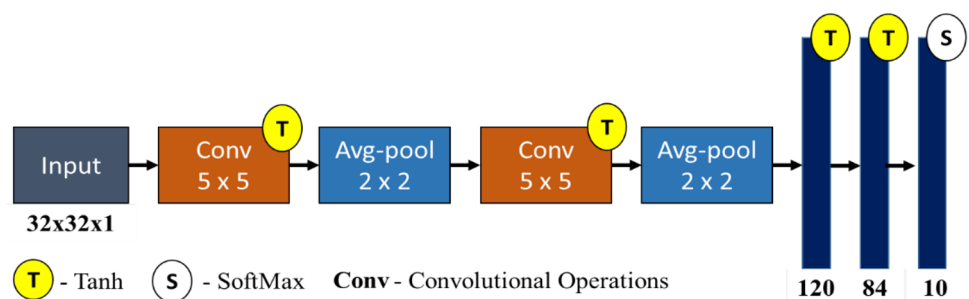
LeNet-5 is the basic and one of the simplest CNN architectures. It holds a total of five layers with two convolutional and three fully connected layers [26]. LeNet-5 uses Tanh as the activation function, whereas the softmax activation function is used at the final layers for convergence. The basic architecture of LeNet-5 is presented in Fig. 3.

The average-pooling layer acts as a subsampling layer and has trainable weights [27]. This architecture consists of 60,000 parameters and occupies a less storage space [44]. It has become the standard for stacking convolutional layers, pooling layers, and ending the network with one or more fully connected layers.

3.3 VGG16 architecture

VGG16 is the most commonly used version of CNN. It has a total of 16 layers with 13 convolutional and 3 fully connected layers [25, 44]. VGG16 has introduced the deeper way of designing the CNN. It uses ReLU as an activation function to improve the nonlinearity in the model, whereas the softmax function is used at the final layers for classification. The basic architecture of VGG16 is presented in Fig. 4.

Fig. 3 Basic LeNet-5 architecture



The main contribution of the VGG16 architecture is its homogeneous topology and the introduction of small kernel size. This network stacks more layers to make the architecture deeper and uses small-sized filters (2 × 2 and 3 × 3). VGG16 consists of 138 million parameters and takes up about 500 mb of storage space.

3.4 Inception architecture

Inception is a deeper version of CNN with 24 million parameters. It is the first architecture to employ batch normalization. Inception architecture is made up of inception modules that enable the deep representations of the input feature set by multiple convolutions. [55]. This architecture is capable of handling problems of representational bottleneck using batch normalization and by replacing large filters with small ones [44]. The basic architecture of Inception is presented in Fig. 5.

Inception occupies a storage space of 92 mb with a depth of 159. It uses ReLU as the activation function in the middle layers, whereas the softmax activation function is used for convergence. Each module in the Inception architecture has parallel towers of convolutions with different filters followed by concatenation [16]. This works by the principle of a layer-by-layer construction, in which one layer should analyze the correlation statistics of the last

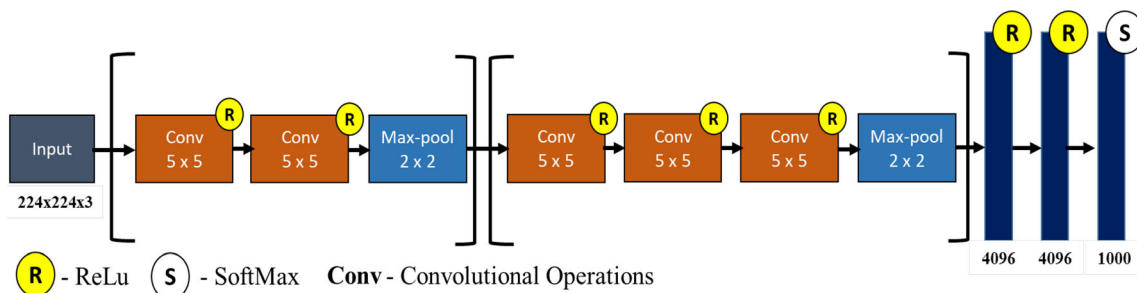


Fig. 4 Basic VGG16 architecture

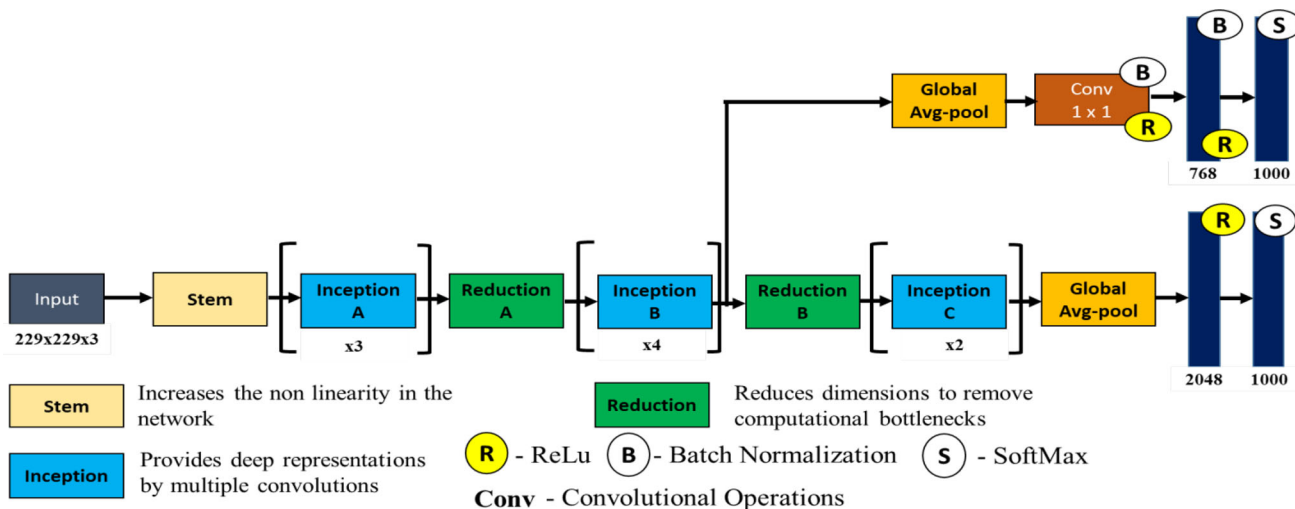


Fig. 5 Basic Inception architecture

layer and cluster them into groups of units with high correlation.

3.5 ResNet architecture

ResNet is one of the early adopters of batch normalization. It has 25 million parameters [56]. ResNet is the architecture that supports hundreds of convolutional layers. It is the first architecture to conduct identity mapping using skipped connections [44]. The process of the ResNet architecture is presented in Fig. 6.

The basic building blocks of ResNet are the convolutional and identity blocks. The convolutional block depends on the ReLU activation function and batch normalization. Finally, it uses the softmax activation function for convergence. This architecture makes it possible to design deeper CNNs up to 152 layers without compromising the generalization power of the model.

3.6 Comparison of familiar pretrained CNN models

The comparison of different configuration parameters of familiar pretrained CNN models, such as LeNet-5, VGG16, Inception, and ResNet, is presented in Table 1. Such a comparison is performed by considering the important aspects, such as the total storage space occupied, depth of the model, activation function, number of parameters, and error rate involved in the execution of the input data [2, 26, 44, 55]. The error rate data are presented based on the performance of the algorithm on the MNIST database and ImageNet database.

The error rate can be computed using log loss [59]. Log loss is a probabilistic measure, and it is used to measure the performance of the concrete crack image classification model. The log loss probability value lies between 0 and 1. The higher the log loss value, the higher the accuracy of the developed model [60]. The entire dataset is divided into training dataset and testing dataset. The images in both datasets are labeled as either crack or non-crack images. The test dataset images are provided as input to the developed model, and the predicted results are derived as

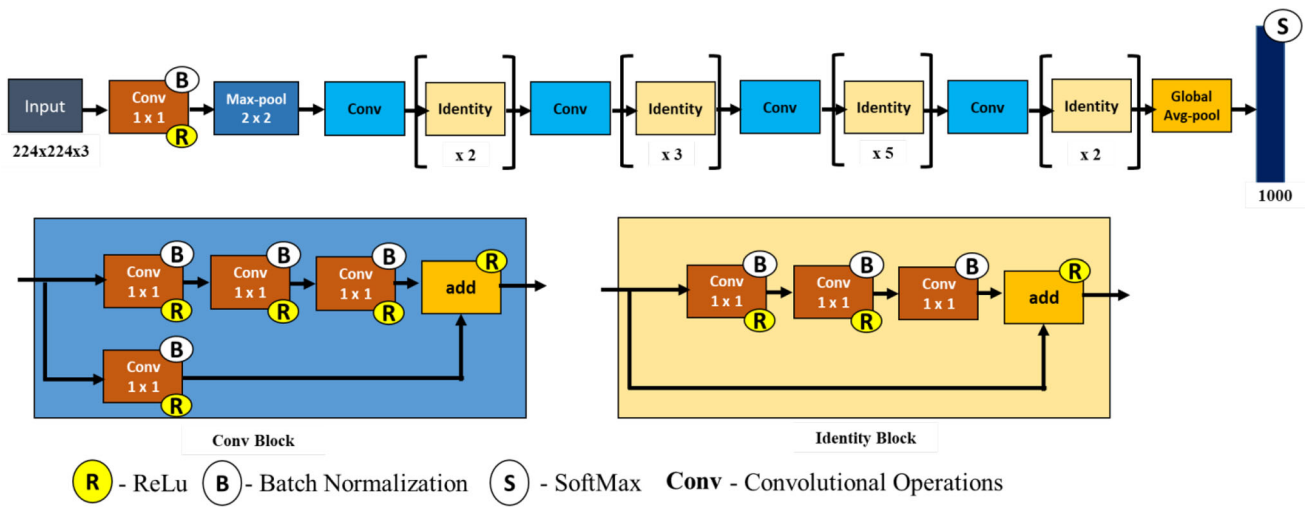


Fig. 6 Basic ResNet architecture

Table 1 Comparison of familiar pretrained CNN models

Model	Storage space (MB)	Depth	Activation function	Total parameters	Error rate
LeNet-5	26	5	Tanh	60,000	0.8 (MNIST)
VGG16	500	23	ReLU	138 Million	7.3 (ImageNet)
Inception	92	159	ReLU	24Million	3.5 (ImageNet)
ResNet	98	152	ReLU	25 Million	3.6 (ImageNet)

output from the model. The actual label and predicted label are used to calculate the log loss value.

For binary classification with a true label y and probability estimate p , the log loss per sample is defined as the negative log likelihood of the classifier given the true label, which is calculated as given in Eq. (1).

$$L_{\log}(y, p) = -\log \Pr(y|p) = -y \log(p) + (1 - y) \log(1 - p) \tag{2}$$

where $y \in \{0, 1\}$, and $p = \Pr(y = 1)$. The binary log loss given above is generalized for the binary case by considering $p_{i,0} = 1 - p_{i,1}$ and $y_{i,0} = 1 - y_{i,1}$. Expanding the inner sum over $y_{i,k} \in \{0, 1\}$ provides the values of the binary log loss. The comparison of the different pretrained CNN models considering the different parameters is presented in Table 1.

It can be seen from Table 1 that the LeNet-5-based CNN model holds a minimum depth compared with the other models [44]. Since the model only holds a minimum depth, the total storage space used during the execution of the input and the total number of parameters involved in the execution are also very minimal compared with those of the other models [4]. The diagrammatic representation of the model complexity of various pretrained models is presented in Fig. 7.

As the depth of the total number of layers in the CNN architecture increases, the complexity of the model also

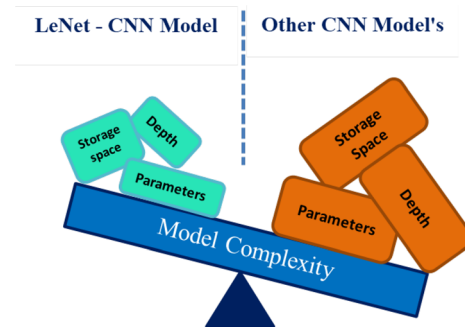


Fig. 7 Model complexity of CNN

increases [9]. The increase in depth exponentially increases the total number of parameters, which in turn results in the increase in the storage space. An additional insight is that increasing the depth and the total number of parameters does not improve the performance of the model [11]. The overall performance of the model is based on the optimization of the hyperparameters involved in the network. From the literature [26, 27], it is clear that the LeNet-5 architecture operates in a high speed than the other DCNN models owing to its spatial arrangement. Since concrete images are not diversified images like the images available in the ImageNet database, dependency on deep architecture results is not essential.

4 Methodology

4.1 Optimized LeNet (OLeNet) architecture

In the proposed work, the optimization of the hyperparameters of the LeNet-5 results in OLeNet has been attempted. Significant changes were carried out to optimize the process of the conventional LeNet-5 architecture. Moreover, two convolutional layers were added to the convolutional block of OLeNet. These convolutional layers enhance the property of spatial invariance to recognize the key features in the crack images. To improve the nonlinearity, the ReLU activation function was used as an alternative to the Tanh and sigmoid functions. Using the ReLU activation function makes it possible to easily backpropagate the errors and activate multiple layers of neurons. In OLeNet, a fully connected layer occupies most of the parameter results in co-dependency among the neurons during training, which degrades the individual power of each neuron, leading to overfitting of the training data. To overcome the problem of overfitting, the dropout function is also added in the proposed OLeNet architecture (Fig. 8).

OLeNet follows the same math behind the CNN operation. The mathematical relationship for the convolution in each location of P_y of the output y in the OLeNet architecture is expressed as Y_{OLeNet} in Eq. (3).

$$Y_{OLeNet}(P_y) = \sum w(P_G).x(P_y + P_G) \tag{3}$$

where w denotes the filter; G , the field in the convolutional layer; and P_G , the positions in the field G .

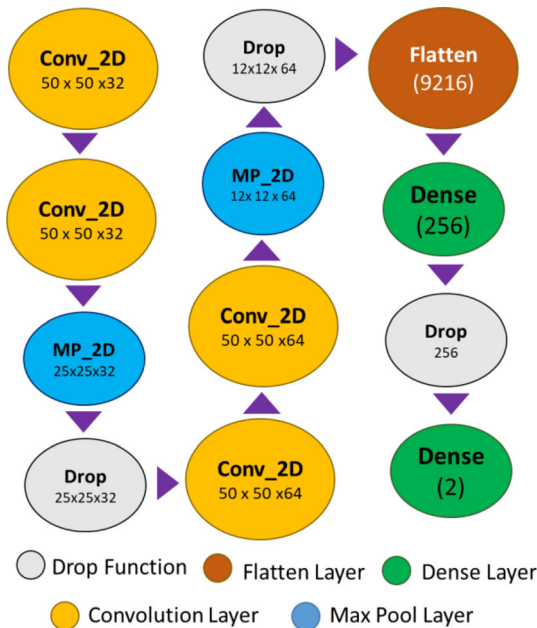


Fig. 8 OLeNet architecture configuration

The convolutional layer (conv2d) in the convolutional block ensures the extraction of more features from the image. The convolutional block now consists of two conv2d convolutional layers followed by a subsampler (max-pool2d) layer. This enables the model to perform a deeper feature extraction from the input image. For an input image (Fig. 9) of dimension $(h \times w \times d)$ and filter size of $(f_h \times f_w \times d)$, the dimensions of the convoluted image will be of the dimensions $(h - f_h + 1 \times w - f_w + 1 \times 1)$.

ReLU is used as an activation function to improve the nonlinearity in the model to achieve better performance. The ReLU activation layer is added to each convolutional block. This layer reduces the linearity between the extracted features. Moreover, it improves the ability of the model to predict a more diverse set of input data. ReLU is also capable of including more nonlinearity in the model. The ReLU activation function for an input variable x is expressed in Eq. (4).

$$\text{ReLU}(x) = \max(x, 0) \tag{4}$$

The process of ReLU as a transfer function in handling the image matrix is presented in Fig. 10.

The dropout function drops the less relevant data and only keeps the significant feature data. It improves the model to achieve better classification and feeds relevant data to fully connected dense layers. Since the training data have a low resolution (50×50) for training, it is reasonable to use a relatively low-power computational model for enhanced execution. Such models are enabled to run in low-power devices, such as smartphones and IoT devices.

The kernel convolutional and pooling operations in OLeNet play a significant role in optimization. Convolutional operation is the integral building block of CNN. In the OLeNet architecture, convolutional operation is the process in which a small matrix of numbers, called kernel, pass over the image pixel matrix and outputs a feature map based on the kernel values. The kernel values are utilized to detect a variety of features in the image, such as horizontal lines, vertical lines, intersections, and shapes. These values represent the features of various real-world objects with curves and lines. Convolutional operation is very powerful and important in the image classification as it can be adopted to learn the features of the object rather than taking them as absolute static values observed in standard neural

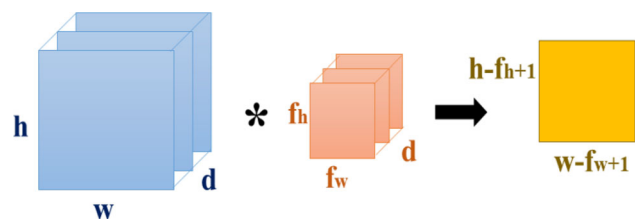
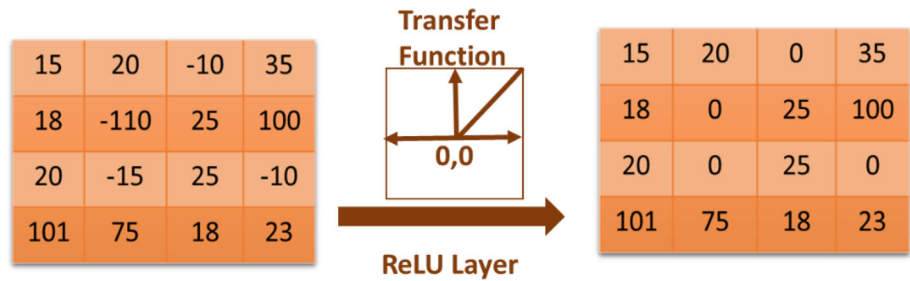


Fig. 9 Operating principle of convolution

Fig. 10 Process of ReLU as a transfer function



networks. Here, the values in the filters are the parameters that can learn through backpropagation using the labeled training data. The kernel convolutional operation in the OLeNet architecture is expressed in Eq. (5).

$$G[m, n] = (f \times m)[m, n] = \sum_j \sum_k h[j, k] \times f[m - j, n - k] \tag{5}$$

where f denotes the input image, and k denotes the kernel. The indices of rows and columns of the resultant matrix are denoted by m and n , respectively.

Pooling operation is the process of downsampling the features of the image generally given by the output of the CNN layer filters. Pooling downsamples the features as an effort to retain only the significant features for the sequential layers of the convolutional parts. In OLeNet, the operation carries through each channel, thereby affecting just the height and weight dimensions of the feature maps, keeping the number of channels intact. The pooling operation in OLeNet is calculated as in Eqs. (6) and (7).

$$\dim(\text{pooling}(a^{l-1})) = \left(\frac{n_h + 2p - f}{s} + 1, \frac{n_w + 2p - f}{s} + 1, n_c \right); s > 0 \tag{6}$$

$$\dim(\text{pooling}(a^{l-1})) = (n_h + 2p - f, n_w + 2p - f, n_c); s = 0 \tag{7}$$

where a^{l-1} denotes the feature map; a^0 , the input image; n_h , the height of the feature map a^{l-1} ; n_w , the width of the feature map a^{l-1} ; n_c , the number of channels of a^{l-1} ; p , the size of the padding used in the feature map before pooling; and f , the size of the pooling filter. Conventionally, the values of f and s are set to $f = 2$ and consider $s = 2$.

4.2 Implementation procedure

4.2.1 Concrete image dataset

The open-source dataset used in the proposed work for classification is collected from multiple buildings in the METU [2] (Table 2). The dataset consists of a total of 40,000 images of 227×227 pixels and was equally

grouped into “crack” and “non-crack” for the binary classification (Fig. 11). A total of 32,000, 4000, and 4000 images were used for the training, validation, and testing, respectively.

4.2.2 Proposed architecture

The overall execution process of the proposed work begins with the OLeNet architecture configuration. Some of the important hyperparameters of the deep learning architectures are the total number of layers, number of hidden units, activation function, epochs, and learning rate. In the OLeNet architecture, these hyperparameters are optimized to achieve enhanced performance with a shallow layer stack architecture arrangement. The shallow architecture depends on minimum computational resources. The overall computational architecture for the OLeNet-based crack detection model (Fig. 12) focuses on model training, model validation, and performance evaluation.

Image pre-processing is performed before building and training the CNN architecture model. High-resolution images have variance in terms of surface finish and illumination conditions. In terms of random rotation or flipping, no data augmentation performed. All the images from the input are read and stored in a numpy array together with their label. For negative images, the numpy array is encoded as [1, 0], and for positive images the numpy array is encoded as [0, 1]. Encoding improves the model training and prediction. Data transformation is performed by processing the input images and is saved as a numpy array. The saved numpy array can be directly loaded without the need to pre-process the data every time. The pseudocode presented in Table 3 explains the technical function of OLeNet for image classification.

Training data out of the entire data are used to build the OLeNet model. Epoch indicates the sum of one forward pass and backward pass for all the training samples. For better time optimization, the model is limited with the epoch value of 20. To reduce the memory space, the batch size of the input data is kept at a minimum. Since the proposed model is shallow with seven layers, the total number of the parameters used and the total memory space occupied are also kept at a minimum. The segregated

Table 2 Dataset details

Dataset details				Training – 32,000		Val – 4000		Test – 4000	
Total images	Image size	Crack	Non-crack	Crack	Non-crack	Crack	Non-crack	Crack	Non-crack
40,000	227 × 227	20,000	20,000	18,000	18,000	2000	2000	2000	2000



Fig. 11 Sample collection of crack and non-crack images

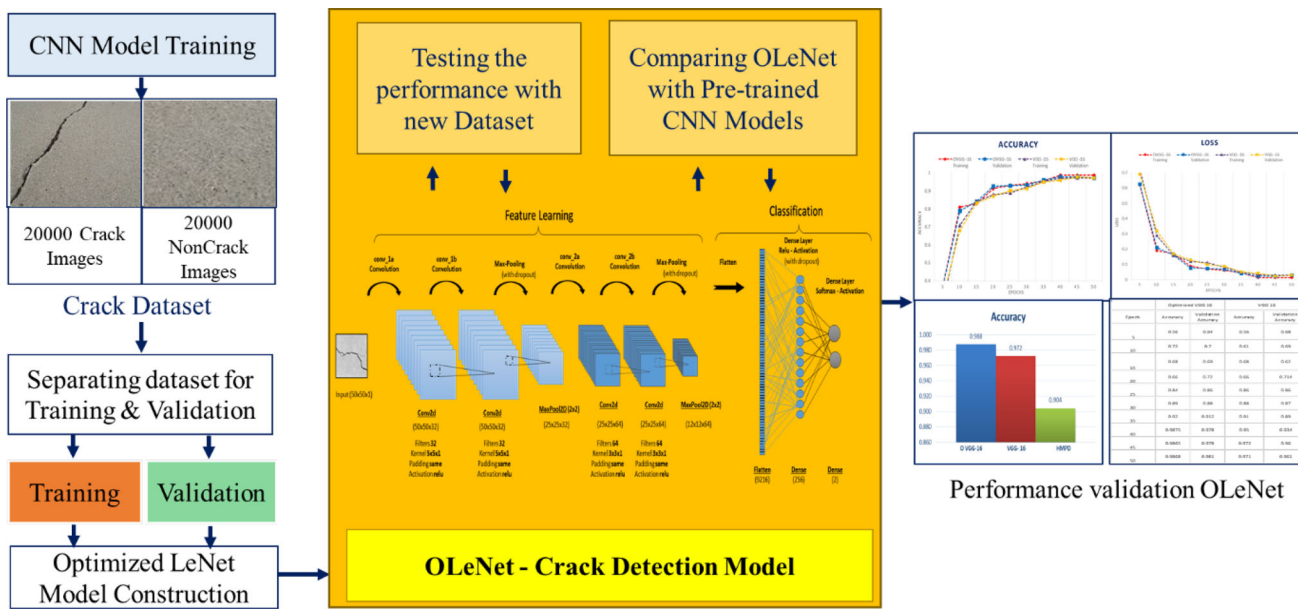


Fig. 12 Proposed OLeNet architecture for crack detection

validation data are utilized to ensure the finalized version of the OLeNet crack detection model. The performance of the OLeNet model is validated by comparing it with those of the other pretrained CNN models discussed in Sect. 5.

5 Results and discussions

In this section, the results of the proposed algorithm (OLENet) are compared with those of the other deep learning models, such as VGG16, Inception, and ResNet.

Section 5.1 validates the performance of OLeNet by considering the different evaluation measures. In Sect. 5.2, the performance of OLeNet is validated by comparing it with those of the other models in terms of hyperparameter values.

5.1 Overall performance of algorithms

Accuracy score, precision, recall, and F-Measure are used as metrics for analyzing the performance of the proposed model. The definition and formula for determining the

Table 3 Pseudocode of OLeNet

OLeNet process steps
<ol style="list-style-type: none"> 1. Input a batch i of input images X_i, along with its label vectors Y_i 2. Run the Feature Extraction algorithm on image batch X_i <ol style="list-style-type: none"> a. Pass through the Convolutional Block <ol style="list-style-type: none"> i. Initialize the n feature kernels f^l of size $(k \times k)$ of the l-th layer ii. Perform the convolutional function for each feature kernel <ol style="list-style-type: none"> ❖ for $f_j^l = f_1^l, f_2^l, f_3^l, \dots, f_n^l$ do <ol style="list-style-type: none"> a. Perform the convolutional function, $z_j^l = X_i \times f_j^l$ b. Perform the ReLU Activation to obtain output, $a_j^l = \max(z_j^l, 0)$ ❖ end for iii. Repeat for all feature kernels and different layers iv. Perform b. Perform the Max Pooling Function on the output tensor a_j^l c. Repeat this for the defined number of times mentioned in the architecture d. Finally, convert the tensor into linear array <i>via</i> Flattening. Thus, feature vector $a_{feature}$ is obtained 3. Run Feature Classification Network on the feature vector. <ol style="list-style-type: none"> a. Initialize weights w and bias b arrays of the Linear Network with 256 Neural Nodes b. Perform $z = w \cdot a_{feature} + b$ c. Perform the ReLU Activation Function $a^l = \max(z, 0)$ d. Initialize weights w and bias b arrays of the Linear Network with 2 Neural Nodes e. Perform $z = w \cdot a^l + b$ f. Perform the ReLU Activation Function $a^l = \max(z, 0)$ g. Apply the softmax function on a^l to obtain the probability distribution of the two classes Crack and Non-crack.

different performance metrics are explained in the following section.

Accuracy score denotes the measure of correctness of the model's prediction. If \hat{y}_i is the predicted label of the i th sample test image and y_i is its corresponding actual label, then the fraction of correct predictions over the $n_{samples}$ of the test images is computed as follows:

$$accuracy(y, \hat{y}) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} 1(\hat{y}_i = y_i) \quad (8)$$

The performance of the model is analyzed by observing the true positive (TP), true negative (TN), false positive (FP), and false negative (FN) metrics. A TP is an outcome in which the model correctly classifies the crack images.

Similarly, a TN is an outcome in which the model correctly classifies the non-crack images. An FP and FN are the outcomes in which the model incorrectly classifies the crack images and the non-crack images, respectively.

Precision denotes the estimation of the model's ability to perform with minimum FP results. It evaluates the robustness of the model and enables the model to avoid labeling an input image as crack that is actually non-crack (Eq. 9).

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (9)$$

Recall denotes the measure of the ability of the model to correctly identify the crack images (Eq. 10).

$$\text{Recall} = \frac{TP}{TP + FN} \tag{10}$$

F-Measure denotes the harmonic mean of precision and recall. It is utilized to evaluate the model with a more considerate score (Eq. 11).

$$\text{F-Measure} = 2 \times \left[\frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \right] \tag{11}$$

The overall performance analysis of the proposed OLeNet algorithm and the other pretrained CNN models is presented in Table 4. Confusion matrix (Fig. 13) clearly visualizes that OLeNet performs comparatively better than the other algorithms.

The performance measures (Table 5) derived from the confusion matrix present clear insights into the algorithms. The proposed OLeNet algorithms produce maximum accuracy, similar to VGG16 and Inception. Despite its shallow configuration, OLeNet withstands to produce maximum accuracy similar to other deep architectures. In addition to the accuracy, OLeNet remains competitive with other pretrained models in terms of the values of precision, recall, and F-Measure. The graphical representation of the performance is demonstrated in Figs. 14, 15, 16 and 17.

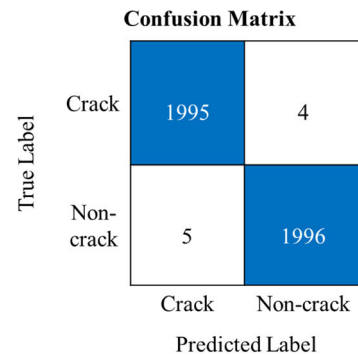
The OLeNet architecture was able to produce 99.8% accuracy and the same value for precision, recall, and F-Measure evaluations, respectively. The VGG16, Inception, and ResNet architectures produced an accuracy of 99.8%, 99.8%, and 97.1%, respectively. Being computationally lightweight in nature, the proposed shallow architecture is persistent in exhibiting better performance compared with the other pretrained deep CNN architectures.

5.2 Performance of OLeNet based on hyperparameters

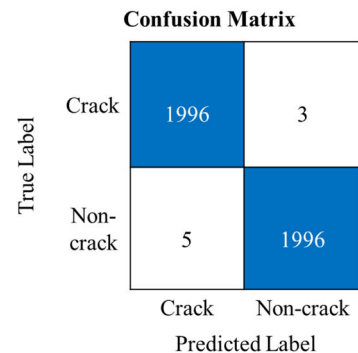
The hyperparameters of the deep learning model govern the entire training process of model building. The performance comparison between the OLeNet architecture and other pretrained CNN models provides a clear insight into the importance of hyperparameter optimization. The hyperparameter performance of OLeNet (Figs. 18, 19, 20,

Table 4 Confusion matrix

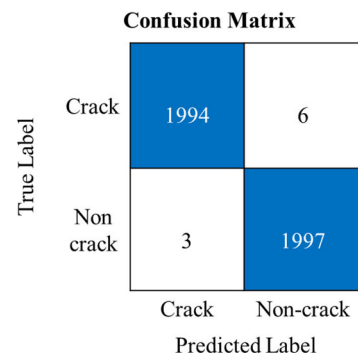
Algorithm	TP	TN	FP	FN
OLeNet	1995	1996	4	5
VGG16	1996	1996	3	5
Inception	1994	1997	6	3
ResNet	1890	1988	110	12



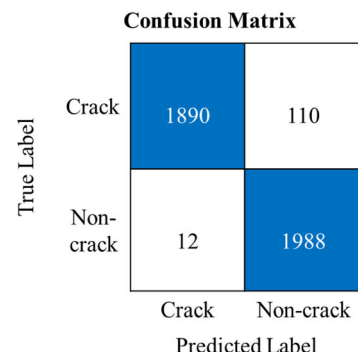
(a) OLeNet



(b) VGG16



(c) Inception



(d) ResNet

◀Fig. 13 Confusion matrices

Table 5 Performance measure

	Accuracy	Precision	Recall	F-Measure
OLeNet	0.998	0.998	0.998	0.998
VGG16	0.998	0.999	0.998	0.999
Inception	0.998	0.997	0.998	0.998
ResNet	0.970	0.945	0.994	0.969

21, 22, 23, 24, 25, 26) is compared with those of the other deep CNN architectures in terms of training accuracy, validation accuracy, training loss, and validation loss at each epoch.

To reduce the used computational power, minimization of the epoch value is required [14]. The performance of other pretrained models used for comparison and their visual representations (Figs. 20, 21) in this proposed work indicate the significance of hyperparameter optimization. The visual representation (Figs. 22, 23) clearly demonstrates that the proposed algorithm is capable of meeting the maximum accuracy and minimum loss within 20 epoch counts. In this work, it has been found that by reducing each epoch count, the computation time reduces by 11 s.

The VGG16 based-CNN model is capable of producing the maximum value in all the performance measures. Inception exhibits better performance compared with the ResNet architecture. The VGG16 and Inception architectures are capable of producing the same accuracy with OLeNet. The concept of identity mapping with the ResNet architecture complicates the network to obtain maximum accuracy in limited epoch counts, compared with other pretrained CNN models. The performance comparison between the CNN models and epoch values (Table 6)

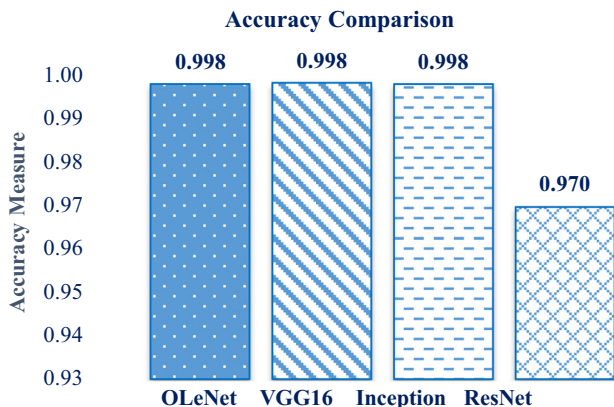


Fig. 14 Accuracy measure analysis

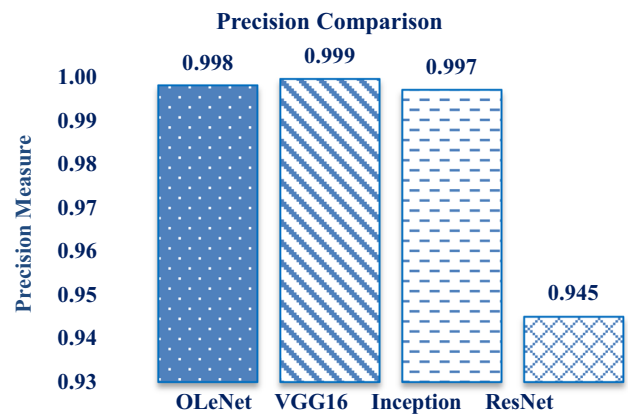


Fig. 15 Precision measure analysis

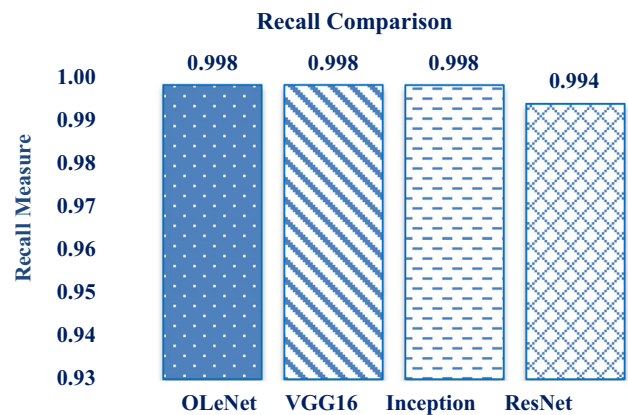


Fig. 16 Recall measure analysis

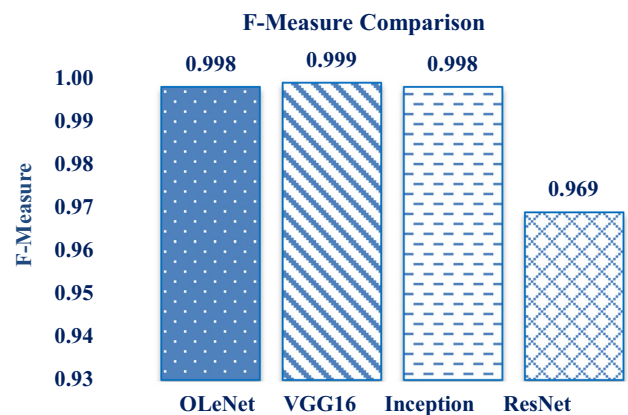


Fig. 17 F-Measure analysis

indicates the significance of hyperparameter optimization. The optimized version of LeNet exhibits better performance than the other CNN models (Figs. 22, 23, 24, 25, 26) at minimum epoch counts.

The computational performance of all the CNN models can be analyzed by comparing the accuracy and loss values with respect to epoch count. In this proposed work, the

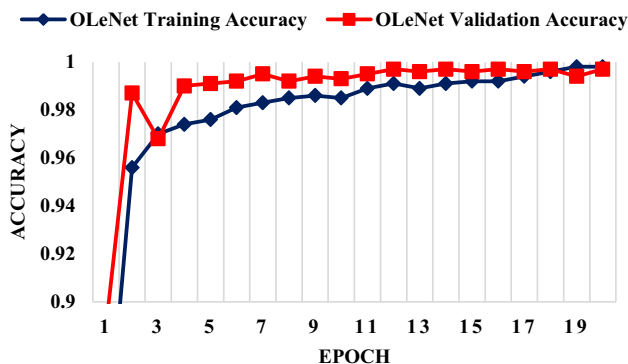


Fig. 18 OLeNet accuracy

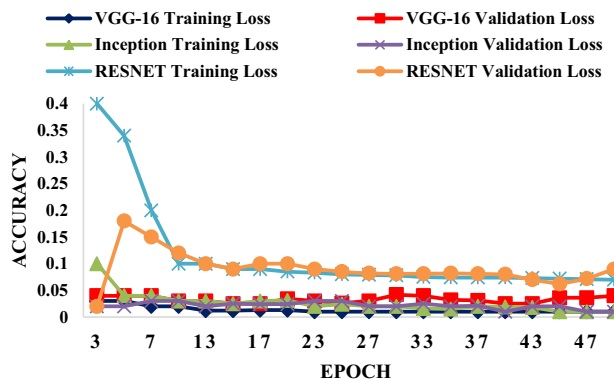


Fig. 21 Loss comparison of pretrained CNN models

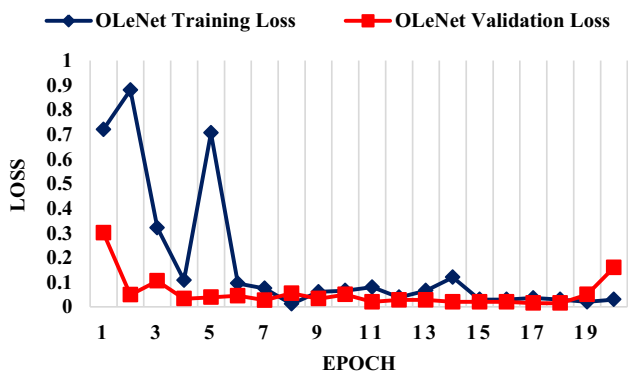


Fig. 19 OLeNet loss

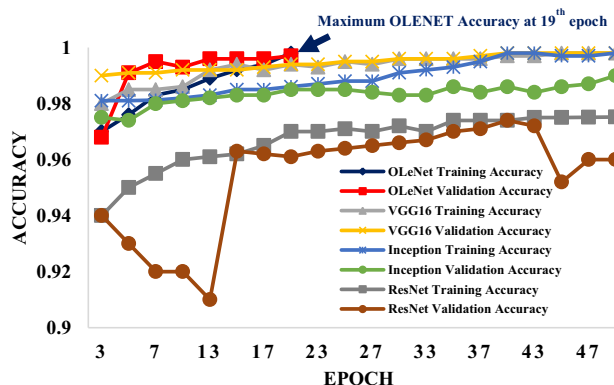


Fig. 22 OLeNet accuracy comparison with deep CNN models

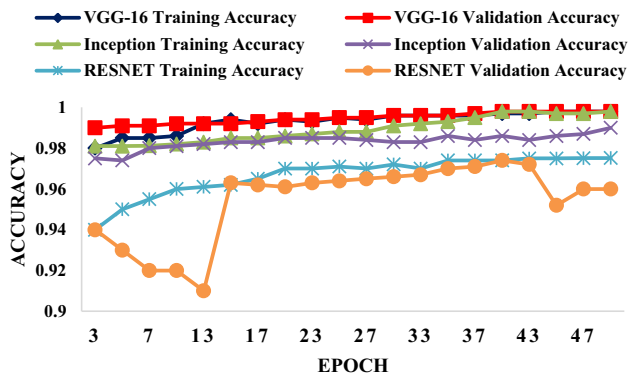


Fig. 20 Accuracy comparison of pretrained CNN models

maximum accuracy achieved is 99.8%. The VGG16- and Inception-based CNN models are capable of achieving this accuracy at the 45th and 47th epoch counts, respectively, whereas the proposed shallow OLeNet model is capable of achieving the same accuracy at the 19th epoch count. The minimum loss value is 0.01. The VGG-, Inception-, and ResNet-based CNN models can achieve this minimum loss at the 45th, 47th, and 42nd epoch counts, respectively, whereas the proposed OLeNet model can achieve this loss at the 18th epoch count.

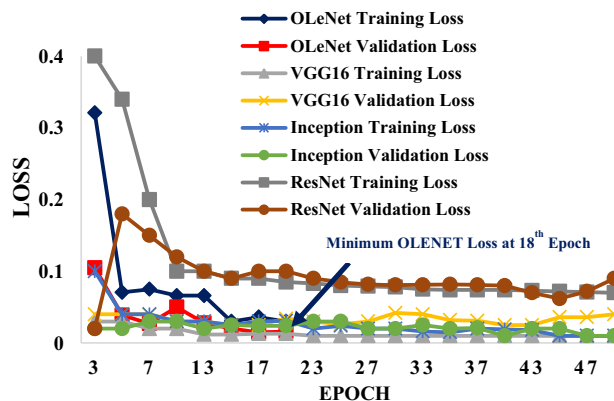


Fig. 23 OLeNet loss comparison with deep CNN models

In addition to the maximum accuracy and minimum loss values, OLeNet can stabilize the model at minimum epoch values compared with other pretrained CNN models (Fig. 26). The inclusion of the activation function along with the dropout function allows the model to skip the unwanted features at the minimum epoch count. The ReLU activation function improves the nonlinearity in the model,

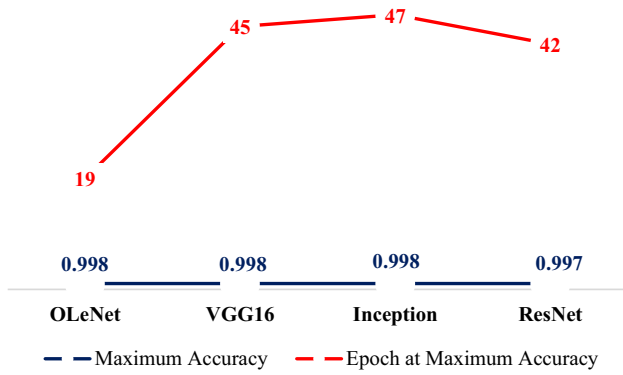


Fig. 24 Epoch at maximum accuracy

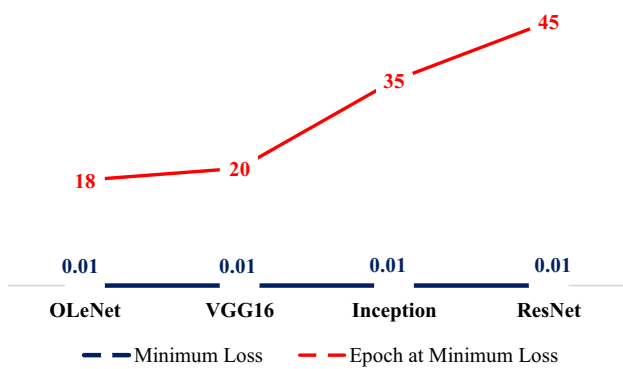


Fig. 25 Epoch at minimum loss

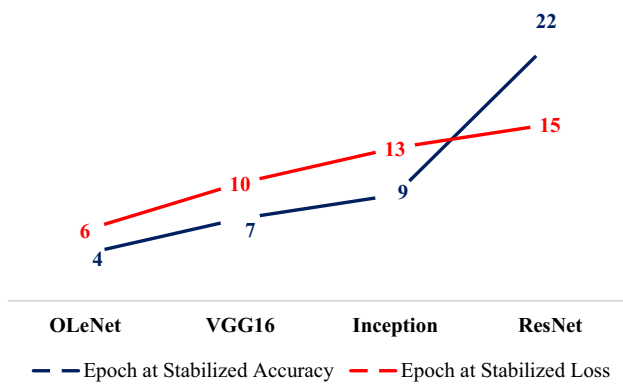


Fig. 26 Epoch at stabilized values

thus enabling the model to extract the best feature set from the available features. Due to hyperparameter optimization, the proposed model gets trained within 220 s for 40,000 images of 50 × 50 pixels within 20 epoch counts.

6 Conclusion

In this study, a shallow CNN-based architecture is employed for concrete surface crack detection. The entire procedure of the proposed method consists of fine-tuning of the LeNet-5 architecture with the METU dataset. Fine-tuning of the hyperparameters of LeNet-5 results in the establishment of OLeNet, which enables OLeNet to perform competitively with other pretrained deep learning models, such as VGG16, ResNet, and Inception. A shallow optimized CNN model (OLeNet) has been built and validated with 40,000 crack and non-crack images obtained from the METU dataset. Contrary to the previous research works, the proposed model does not depend on high-quality images and high-end computational devices for building real-time crack detection models used to identify the concrete cracks on the structures. The validation of the proposed OLeNet architecture has proven to achieve a maximum accuracy of 99.8% with a minimum computation of 19 epoch counts with its shallow layer stack (deep CNN architectures require a minimum of 45 epoch counts to achieve 99.8% accuracy). In addition, the proposed model gets trained within 220 s to achieve maximum accuracy (deep CNN architectures require a minimum computation time of 524 s to achieve maximum accuracy). Rather than depending on high-quality images by deep CNN models, the proposed shallow OLeNet architecture is capable of achieving maximum accuracy using simple images.

The minimum layer stack arrangement and dependency on simple images reduce the computational time and operational time of the OLeNet architecture. Despite the advantages, the system is not fully automatic, which remains as a limitation for the proposed method. The proposed architecture can be employed using low-power computational IoT devices for image classification to develop a fully functioning autonomous system. Therefore, future studies shall concentrate on integrating the proposed shallow CNN architecture with low-power computational

Table 6 Performance comparison with epoch

Algorithm	Max accuracy	Epoch at max accuracy	Min loss	Epoch at min loss	Epoch at stabilized accuracy	Epoch at stabilized accuracy
OLeNet	0.998	19	0.01	18	4	6
VGG16	0.998	45	0.01	20	7	10
Inception	0.998	47	0.01	35	9	13
ResNet	0.971	42	0.01	45	22	15

autonomous devices, such as unmanned aerial vehicles, to achieve enhanced performance.

Funding This work was supported by Korea Research Fellowship Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (No. 2019H1D3A1A01101442). This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIT) (No. 2019R1G1A1095215).

Code availability Not applicable.

Compliance with ethical standards

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

References

- Kim H, Ahn E, Shin M, Sim SH (2018) Crack and non-crack classification from concrete surface images using machine learning. *Struct Health Monit* 18:725–738
- Kang D, Cha Y-J (2019) Autonomous UAVs for structural health monitoring using deep learning and an ultrasonic beacon system with geo-tagging. *Comput-Aided Civ Infrastruct Eng* 33:885–902. <https://doi.org/10.1111/mice.12375>
- Gibb S, La HM, Louis S (2018) A genetic algorithm for convolutional network structure optimization for concrete crack detection. *IEEE Congress Evolut Comput*. <https://doi.org/10.1109/CEC.2018.8477790>
- LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. *Proc IEEE* 86:2278–2324
- Zhang C-W, Yang M-Y, Zeng H-J, Wen J-P (2019) Pedestrian detection based on improved LeNet-5 convolutional neural network. *J Algorithms Comput Technol*. <https://doi.org/10.1177/1748302619873601>
- Cook W, Barr PJ (2017) Observations and trends among collapsed bridges in New York state. *J Perform Constr Facil*. [https://doi.org/10.1061/\(ASCE\)CF.1943-5509.0000996](https://doi.org/10.1061/(ASCE)CF.1943-5509.0000996)
- Road Bureau, Ministry of Land, Infrastructure, Transportation, and Tourism, Roads in Japan. Accessed on http://www.mlit.go.jp/road/road_e/index_e.html.
- National Transportation Safety Board (2007) Collapse of I-35W Highway Bridge, Minneapolis, Minnesota. <http://www.dot.state.mn.us/i35wbridge/ntsb/finalreport.pdf>
- Spencer BF, Hoskere V, Narazaki Y (2019) Advances in computer vision-based civil infrastructure inspection and monitoring. *Engineering*. <https://doi.org/10.1016/j.eng.2018.11.030>
- Asakura T, Kojima Y (2003) Tunneling and underground space technology. *Tunnel Maint Jpn* 18:161–169. [https://doi.org/10.1016/S0886-7798\(03\)00024-5](https://doi.org/10.1016/S0886-7798(03)00024-5)
- Dung CV, Anh LD (2019) Autonomous concrete crack detection using deep fully convolutional neural network. *Autom Constr*. <https://doi.org/10.1016/j.autcon.2018.11.028>
- Koch C, Georgieva K, Kasireddy V, Akinci B, Fieguth P (2015) A review on computer vision based defect detection and condition assessment of concrete and asphalt civil infrastructure. *Adv Eng Inform* 29:196–210. <https://doi.org/10.1016/j.aei.2015.01.008>
- Kim I-H, Jeon H, Baek S-C, Hong W-H, Jung H-J (2018) Application of crack identification techniques for an aging concrete bridge inspection using an unmanned aerial vehicle. *Sensors*. <https://doi.org/10.3390/s18061881>
- Yan J, Downey A, Cancelli A, Laflamme S, Chen A, Li J, Ubertini F (2019) Concrete crack detection and monitoring using a capacitive dense sensor array. *Sensors*. <https://doi.org/10.3390/s19081843>
- Teidj S, Khamlichi A, Driouach A (2016) Identification of beam cracks by solution of an inverse problem. *Procedia Technol* 22:86–93. <https://doi.org/10.1016/j.protcy.2016.01.014>
- Chatzi EN, Hiriyyur B, Waisman H, Smyth AW (2011) Experimental application and enhancement of the XFEM–GA algorithm for the detection of flaws in structures. *Comput Struct* 89:556–570
- Rabinovich D, Givoli D, Vigdergauz S (2007) XFEM based crack detection scheme using a genetic algorithm. *Int J Numer Methods Eng* 71:1051–1080. <https://doi.org/10.1002/nme.1975>
- Cha Y-J, Choi W (2017) Deep learning-based crack damage detection using convolutional neural networks. *Comput-Aided Civ Infrastruct Eng* 32:361–378
- Adhikari RS, Moselhi O, Bagchi A (2014) Image-based retrieval of concrete crack properties for bridge inspection. *Autom Constr* 39:180–194. <https://doi.org/10.1016/j.autcon.2013.06.011>
- Noh Y, Koo D, Kang Y, Park D, Lee D (2017) Automatic crack detection on concrete images using segmentation via fuzzy C-means clustering. In: Proceedings of the 2017 international conference on applied system innovation (ICASI), Sapporo, Japan, pp 877–880
- Dawood T, Zhu Z, Zayed T (2017) Machine vision-based model for spalling detection and quantification in subway networks. *Autom Constr* 81:149–160
- Ali R, Gopal L G, Cha Y J (2018) Vision-based concrete crack detection technique using cascade features. In: Proceedings of the SPIE 10598, sensors and smart structures technologies for civil mechanical, and aerospace systems. <https://doi.org/10.1117/12.2295962>
- Dinh T H, Ha Q, La H M (2016) Computer vision-based method for concrete crack detection. In: Proceedings of the 2016 14th international conference on control, automation, robotics and vision (ICARCV), pp 1–6
- Fathi H, Dai F, Lourakis M (2015) Automated as-built 3D reconstruction of civil infrastructure using computer vision: achievements, opportunities, and challenges. *Adv Eng Inform* 29:149–161. <https://doi.org/10.1016/j.aei.2015.01.012>
- Seo J, Han S, Lee S, Kim H (2015) Computer vision techniques for construction safety and health monitoring. *Adv Eng Inform* 29:239–251
- Nishikawa T, Yoshida J, Sugiyama T, Fujino Y (2012) Concrete crack detection by multiple sequential image filtering. *Comput Aided Civ Infrastruct Eng* 27:29–47
- Islam M, Kim J-M (2019) Vision-based autonomous crack detection of concrete structures using a fully convolutional encoder–decoder network. *Sensors*. <https://doi.org/10.3390/s19194251>
- Teizer J (2015) Status quo and open challenges in vision-based sensing and tracking of temporary resources on infrastructure construction sites. *Adv Eng Inform* 29:225–238. <https://doi.org/10.1016/j.aei.2015.03.006>
- Yang J, Park M-W, Vela PA, Golparvar-Fard M (2015) Construction performance monitoring via still images, time-lapse photos, and video streams: now, tomorrow, and the future. *Adv Eng Inform* 29:211–224
- Radopoulou SC, Brilakis I (2015) Patch detection for pavement assessment. *Autom Constr* 53:95–104

31. Abdel-Qader I, Abudayyeh O, Kelly ME (2003) Analysis of edge-detection techniques for crack identification in bridges. *J Comput Civ Eng* 17:255–263
32. Prasanna P, Dana KJ, Gucunski N, Basily BB, La HM, LimParvardeh RSH (2016) Automated crack detection on concrete bridges. *IEEE Trans Autom Sci Eng* 13:591–599
33. Oh JK, Jang G, Oh S, Lee JH, Yi B, Moon YS, Lee JS, Choi Y (2009) Bridge inspection robot system with machine vision. *Autom Constr* 18:929–941
34. Li G, Zhao X, Du K, Ru F, Zhang Y (2017) Recognition and evaluation of bridge cracks with modified active contour model and greedy search-based support vector machine. *Autom Constr* 78:51–61
35. Na W, Tao W (2012) Proximal support vector machine based pavement image classification. In: *IEEE Fifth international conference on advanced computational intelligence (ICACI)*, pp 686–688
36. Abdel-Qader I, Pashaie-Rad S, Abudayyeh O, Yehia S (2006) PCA - based algorithm for unsupervised bridge crack detection. *Adv Eng Softw* 37:771–778
37. Choudhary GK, Dey S (2012) Crack detection in concrete surfaces using image processing, fuzzy logic, and neural networks. In: *Proceedings of the 2012 IEEE fifth international conference on advanced computational intelligence (ICACI)*, Nanjing, China, pp 404–411
38. Sri Preethaa KR, Sabari A (2020) Intelligent video analysis for enhanced pedestrian detection by hybrid metaheuristic approach. *Soft Comput* 24(16):12303–12311
39. Bishop CM (2006) *Pattern recognition and machine learning*. Springer-Verlag, New York City
40. Mhathesh TSR, Andrew J, Martin Sagayam K, Henesey L (2020) 3D convolutional neural network for bacterial image classification. In: Peter J, Fernandes S, Alavi A (eds) *Intelligence in big data technologies beyond the hype*. *Advances in intelligent systems and computing*, vol 1167. Springer, Singapore. https://doi.org/10.1007/978-981-15-5285-4_42
41. Zhang H, Tan J, Liu L, Wu Q M J, Wang Y, Jie L (2017) Automatic crack inspection for concrete bridge bottom surfaces based on machine vision. In *Proceedings of the 2017 Chinese automation congress (CAC)*, Jinan, China, 20–22 Oct 2017, pp 4938–4943
42. Cha YJ, Choi W, Suh G, Mahmoudkhani S, Buyukozturk O (2017) Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types. *Comput Aided Civ Infrastruct Eng*. <https://doi.org/10.1111/mice>
43. Zhang K, Cheng HD, Zhang B (2018) Unified approach to pavement crack and sealed crack detection using preclassification based on transfer learning. *J Comput Civ Eng* 32:04018001
44. Chen FC, Jahanshahi MR (2017) NB-CNN: Deep learning-based crack detection using convolutional neural network and Naïve Bayes data fusion. *IEEE Trans Ind Electron*. <https://doi.org/10.1109/TIE.2017.2764844>
45. Fan Z, Wu Y, Lu J, Li W (2018) Automatic pavement crack detection based on structured prediction with the convolutional neural network. [arXiv:1802.02208](https://arxiv.org/abs/1802.02208).
46. Wang K C, Zhang A, Li J Q, Fei Y, Chen C, Li B (2017) Deep learning for asphalt pavement cracking recognition using convolutional neural network. *Airfield and Highway Pavements*, pp 166–177
47. Zhang L, Yang F, Zhang Y D, Zhu Y J (2016) Road crack detection using deep convolutional neural network. In: *IEEE international conference on image processing (ICIP)*, pp 3708–3712. <https://doi.org/10.1109/ICIP.2016.7533052>
48. Pauly L, Hogg R D, Fuentes, Peel H (2017) Deeper networks for pavement crack detection. In: *Proceedings of the 34th ISARC*. 34th international symposium in automation and robotics in construction, IAARC, Taipei, Taiwan, pp 479–485
49. Ren S, He K, Girshick R, Sun J (2017) Faster R-CNN: towards real-time object detection with region proposal networks. *IEEE Tran Pattern Anal Mach Intell* 39:1137–1149. <https://doi.org/10.1109/TPAMI.2016.2577031>
50. Rawat W, Wang Z (2017) Deep convolutional neural networks for image classification: a comprehensive review. *Neural Comput* 29:2352–2449. https://doi.org/10.1162/neco_a_00990
51. Gopala Krishnan K (2018) Deep learning in data-driven pavement image analysis and automated distress detection: a review. *Data* 3:28. <https://doi.org/10.3390/data3030028>
52. Cao QD, Choe Y (2020) Building damage annotation on post-hurricane satellite imagery based on convolutional neural networks. *Nat Hazards* 103:3357–3376
53. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. *Nature* 521:436
54. Kim B, Yuvaraj N, Sri Preethaa KR, Santhosh R, Sabari A (2020) Enhanced pedestrian detection using optimized deep convolution neural network for smart building surveillance. *Soft Comput* 24(22):17081–17092
55. Zhang C, Patras P, Haddadi H (2019) Deep learning in mobile and wireless networking a survey. *IEEE Commun Surv Tutor* 21:2224–2287
56. Ortega-Zamorano F, Jerez JM, Gómez I, Franco L (2017) Layer multiplexing FPGA implementation for deep back-propagation learning. *Integr Comput-Aided Eng* 24(2):171–185
57. Yamaguchi T, Nakamura S, Saegusa R, Hashimoto S (2008) Image-based crack detection for real concrete surfaces. *IEEJ Trans Electr Electron Eng* 3:128–135
58. ASCE (2017) American Society of Civil Engineers (ASCE), *Infrastructure Report Card*. <https://www.infrastructurereportcard.org/>
59. Yuvaraj N, Sabari A (2016) Twitter sentiment classification using binary shuffled frog algorithm. *Intell Autom Soft Comput* 1:1–9
60. Yuvaraj N, Sri Preethaa KR (2017) Diabetes prediction in healthcare systems using machine learning algorithms on Hadoop cluster. *Cluster Comput* 22:1–9. <https://doi.org/10.1007/s10586-017-1532-x>

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