Front. Struct. Civ. Eng. 2021, 15(2): 305–317
https://doi.org/10.1007/s11709-021-0725-9

RESEARCH ARTICLE

Automated classification of civil structure defects based on convolutional neural network

Pierclaudio SAVINO*, Francesco TONDOLO

Department of Structural, Geotechnical and building Engineering, Politecnico di Torino, Torino 10129, Italy
**Corresponding author. E-mail: pierclaudio.savino@polito.it*

ABSTRACT Today, the most commonly used civil infrastructure inspection method is based on a visual assessment conducted by certified inspectors following prescribed protocols. However, the increase in aggressive environmental and load conditions, coupled with the achievement of many structures of the life-cycle end, has highlighted the need to automate damage identification and satisfy the number of structures that need to be inspected. To overcome this challenge, this paper presents a method for automating concrete damage classification using a deep convolutional neural network. The convolutional neural network was designed after an experimental investigation of a wide number of pretrained networks, applying the transfer-learning technique. Training and validation were conducted using a database built with 1352 images balanced between "undamaged", "cracked", and "delaminated" concrete surfaces. To increase the network robustness compared to images in real-world situations, different image configurations have been collected from the Internet and on-field bridge inspections. The GoogLeNet model, with the highest validation accuracy of approximately 94%, was selected as the most suitable network for concrete damage classification. The results confirm that the proposed model can correctly classify images from real concrete surfaces of bridges, tunnels, and pavement, resulting in an effective alternative to the current visual inspection techniques.

KEYWORDS concrete structure, infrastructures, visual inspection, convolutional neural network, artificial intelligence

1 Introduction

The management and efficiency of an infrastructural network has significant relevance for the growth of a country, presenting repercussions from both economic and safety perspectives. Preserving the structural integrity and reliability of bridges, tunnels, and roads is a difficult task. Although the construction of reinforced concrete structures makes it possible to exploit the static collaboration of two complementary materials such as concrete and steel, their union brings about problems of durability as time passes. As a result, the proper resources should be allocated to deal with aging structures still in service and interventions should be defined to plan and avoid catastrophic events. At the same time, in rapidly developing regions characterized by increasing traffic volume, the detection of defects is required from the early stages of the structures to avoid any further losses in structural capacity, durability, and maintenance costs.

The first step in identifying the damage state and potential risk conditions currently takes place through direct visual inspections of certified professionals, combined with the decision-making process. The early detection and classification of defects helps in planning an effective repair program, preventing a worsening of the conditions, and enabling low-cost maintenance. Cracks, spalling, and delamination are key indicators of the general conditions of a concrete structural member. In addition to visually showing the load conditions of a structure, cracks are the first mark of surface degradation caused by the corrosion of steel. Based on their shape and location, it is possible to discover hidden diseases and their potential causes. However, because inspection tasks are conducted manually, some issues need to be solved. Most of the structural placements are in dangerous locations, where a low accessibility could lead to inaccurate evaluations or defects to be occasionally ignored. In addition, the subjective nature of such assessments is strictly linked to

Article history: Received Sep 17, 2020; Accepted Nov 26, 2020

the expertise of the inspector. Consequently, the subjectivity and variability affect the damage detection, which is ineffective and not repeatable over time. Furthermore, a condition assessment can be expensive, with both a long logistic time and intensive labor.

Developments in modern sensors and ICT techniques can significantly help replace current defect inspection practices with a more repeatable, reliable, and automated civil infrastructure condition assessment. More powerful graphical processing units (GPUs) for parallel computing, the ability to transfer and collect big data through the Internet, and innovative learning architectures are changing the manner in which a wide range of industries operates by introducing artificial intelligence. Deep learning is a machine learning technique that has been spreading in recent years owing to the high performance achieved in deep learning. Based on neural network architectures, deep learning techniques applying classification tasks for directly analyzing images, text, or sounds, giving a relevant impact in fields such as autonomous driving systems, medical imaging, and face recognition. Furthermore, artificial neural networks have been developed to approximate the solution of partial differential equations of mechanical problems. In most of these approaches, a collocation method is employed to fit both the governing equations and the boundary conditions at randomly selected points. Anitescu et al. [1] proposed a collocation method for solving second order boundary value problems, such as Poisson and Helmholtz equations, based on an adaptive approach. In addition, Guo et al. [2] developed a collocation method for governing partial differential questions of Kirchhoff plate bending problems.

The implementation of computer vision techniques in self-navigating robots, such as unmanned aerial vehicles, ground robots, and consumer electronics, could represent a turning point for the automated structural inspection process. Images acquired mechanically contain visual information comparable to that obtained by a human inspector. Furthermore, images provide details from the entire field of view quickly, economically, and without contact.

Owing to the complex texture of the defects, various image backgrounds, changes in lighting, surface roughness, and finish, a convolutional neural network (CNN) is a suitable for the automatic classification of such images. Unlike traditional network architectures, which are characterized by simple structures and based on the manual extraction of features with image processing techniques, a CNN learns the features directly from the training data by applying a 2D convolutional layer. This is an important advantage because CNNs no longer require the image processing, design, or manual extraction of the features. In recent years, numerous studies have applied CNN-based algorithms to solve image-based object detection and classification problems. Fan et al. [3] presented a supervised algorithm using a CNN to learn the crack structure under different pavement conditions. Zhang et al. [4] developed a four-layer CNN to detect road cracks, achieving an accuracy of approximately 87%. To improve the network, pretrained deep networks on a large-scale image data set are often reused with transfer-learning techniques related to accuracy and time consumption. Kim and Cho [5] applied transfer learning to the AlexNet network by dividing the training set into cracks, intact surfaces, patterns of joints and edges, and plants. The training of a deep neural network led to an average precision of approximately 92%. Hung et al. [6] built a new data set from the Internet using normal, cracked, honeycomb, blistering, and moss labels. The transfer-learning approach was also applied to obtain a maximum accuracy of 93%. In addition, Zhu and Song [7] developed a model for detecting surface defects on concrete bridges based on transfer learning and a CNN. The VGG-16 was modified to classify the surface as normal or having cracks, plate fractures, corner breaks, edge exfoliation, skeleton exposure, or repairs with approximately 83% validation accuracy. Feng et al. [8] proposed a deep CNN with transfer learning for the damage detection of a hydro-junction infrastructure. For fast tunnel crack identification, Song et al. [9] built a tunnel crack data set and proposed a deep learning algorithm to define a complete analysis system for identifying tunnel cracks. Makantasis et al. [10] combined a CNN and a multi-layer perceptron for the tunnel defect inspection problem. Patterson et al. [11] trained a deep learning algorithm to automate post-earthquake reconnaissance image tagging tasks. Gulgec et al. [12] conducted a finite-element analysis to simulate the cracking of gusset plate connections in steel bridges and classify the strain field as “damaged” or “healthy”. To detect multiple types of damage, Cha et al. [13] defined a faster region-based CNN modifying the ZF-net architecture. A database of 297 images containing steel delamination and corrosion, bolt corrosion, and concrete cracks, reaching a mean precision of approximately 88%, was created. Soukup and Huber-Mörk [14] trained CNNs to detect rail surface defects using a database of photometric stereo images of rail surfaces in a dark-field configuration. In addition, Li et al. [15] improved the YOLO architecture to detect six types of surface defects on cold-rolled steel strips, improving the manufacturing quality of the entire steel strip line.

The above studies have highlighted the potential and versatility of neural networks in automatic classification tasks; however, some aspects must be overcome to make this procedure more reliable. Most of the research has been mainly limited to detecting only the surface cracks, excluding other types of damage typical of real structures such as delamination. Many times, the data sets are based only on ideal laboratory conditions, containing only images with high resolution and prefixed camera-object distance, excluding real structures in different environmental conditions. The main contribution of the present

paper is a deep learning-based algorithm that can classify various types of civil infrastructure surfaces in the "undamaged", "cracked", and "delaminated" classes. The robustness of the network is ensured by considering a database of raw images collected from the Internet, field bridge inspections, and Google Street View, containing real environmental conditions, background variety, and noise. The CNN architecture is developed using the transfer-learning approach through an in-depth comparison among the highest performing deep neural networks in terms of accuracy and prediction time. A further contribution was therefore identifying the best-performing network in identifying damages in civil infrastructures by analyzing a wide number of existing pretrained networks. The neural network developed is able to classify various types of structures and damage surfaces, and differing image quality, with an accuracy of approximately 94%. The experimental results on the validation data set highlight the promising performance of the deep learning model in providing a faster and cheaper overview of existing infrastructures than a current visual inspection. The selected network will also be the basis for future developments in automatic defect localization and quantification.

2 Convolutional neural network

A CNN is a common algorithm used in deep neural networks, a technique of machine learning in which computer models learn how to apply classification tasks directly from input data. CNNs learn from image data, eliminating the need for a manual feature extraction. Similar to the biological structure of a visual cortex in the human brain, a network can have hundreds of layers, each aimed at learning and detecting specific features from images. Extremely low-level features or kernels, such as brightness and edges, can gradually take on more complex shapes that uniquely define an object. The layers closer to the output interpret the extracted features related to the context of the classification task. CNNs generally contain convolution, activation, pooling, and fully connected layers [16]. Filters are applied to each input image, and the output of each layer is the input for the next layer. After learning the features in the surface layers, the network was moved to the classification. The last layer before the output layer is a fully connected layer that creates a vector of a size equal to the number of classes that the network can predict, containing the probabilities for each class (Fig. 1). Because a CNN can also be trained on millions of images and can have an extremely complex network architecture, the processing time required to train a model can be significantly reduced when using a GPU. Depending on the size of the data set, a CNN can be created from scratch, or existing networks can be re-trained with their own data sets for new recognition tasks. This procedure, called transfer learning, is a cost-effective solution for applying the deep learning technique without a large data set and long processing and training time.

2.1 Convolution layer

The convolution layers extract and learn features from their input images by means of neurons arranged in the feature maps. Each neuron is linked to a series of adjacent neurons in the following layer with a set of trainable weights called convolution kernels. Because within the same convolutional layer there are feature maps with different weights, several features can be extracted at each level [17]. In convolution operations, the kernel slides with a prefixed stride on each region of the input image, performing a matrix multiplication between the input pixel and the corresponding weights of the convolution kernel. Figure 2 shows an example of convolution operations involving a $2 \times 2 \times 1$ convolutional kernel of weights of 0.5 and a stride of 1. A larger stride size leads to a smaller output and computational cost but can lose features from the input data.

The bias term can be added to the weighted pixels according to the following equation:

$$Y_i = \sum_i x_i \cdot w_{ij} + b_i, \tag{1}$$

where x_i denotes the input pixel, w_{ij} and b_i represent the

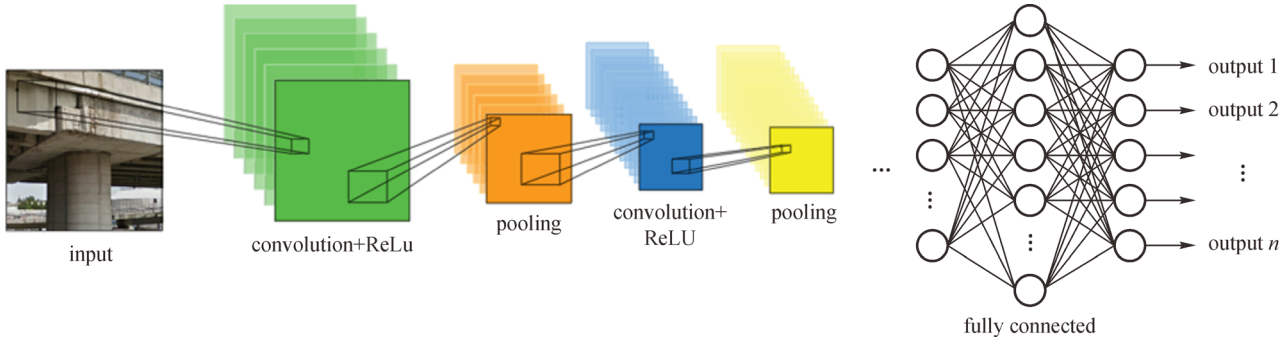

Fig. 1 CNN image classification architecture.

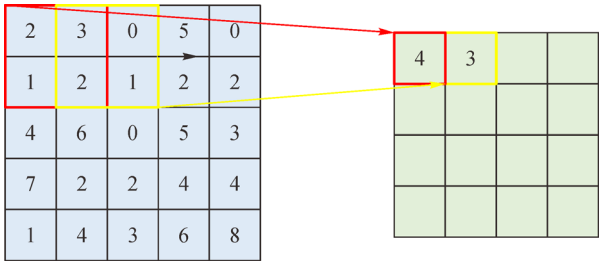

Fig. 2 Convolution operations.

weights and bias of the convolutional filter, and Y_i represents the output. Both parameters are fine-tuned during the training process such that the weight increases or decreases the effectiveness of a specific input, which makes the model more flexible by adjusting the output with the weighted sum of the inputs to the neuron. After the convolutional layer, the convolved results are sent to a nonlinear activation function that provides nonlinear behavior to the neural networks.

2.2 Activation layer

In the neural network architecture, nonlinear functions are introduced to extract nonlinear features when neurons are activated. The sigmoid and hyperbolic tangent functions are the first types of functions to be used. However, both functions present a "vanishing gradient problem" for extremely large and small arguments, making it impossible for neurons to acquire signals or transfer weights and data (Fig. 3). Subsequently, an effective activation function called a rectified linear unit (ReLU) [18] was introduced, and has become the most popular approach:

$$\mathrm{ReLU}(Y) = \max(0, Y)x. \tag{2}$$

Comparing the gradients of the function ReLU with both the sigmoid function and tanh function, the gradients of the ReLU are always 0 or 1. The learning is applied in a specific neuron when training samples generate a positive input for ReLU. These features facilitate a faster computation and convergence speed.

2.3 Pooling layer

After the convolution process, the output neurons include redundant information, which increases the number of calculations. The pooling layer improves the algorithm performance and decreases the computational cost required to process the data by reducing the size of the convolved feature. Furthermore, it can help to keep model training more robust to object orientation and scale changes by extracting the main features. The key features are extracted using max or average pooling. Max pooling provides the maximum value from the region where the filter covers the image, and average pooling gives the average values contained within the kernel. Figure 4 shows the difference between the average and max pooling.

Given a 4 × 4 input image size, a 2 × 2 filter size and stride of 2 are applied. The convolution and pooling layers form a single layer of the CNN architecture.

2.4 Fully connected layer

After the convolutional layers, it is necessary to take high-resolution data and solve the representations of the objects. For this purpose, the fully connected layer interprets all features extracted by the previous layers to produce class

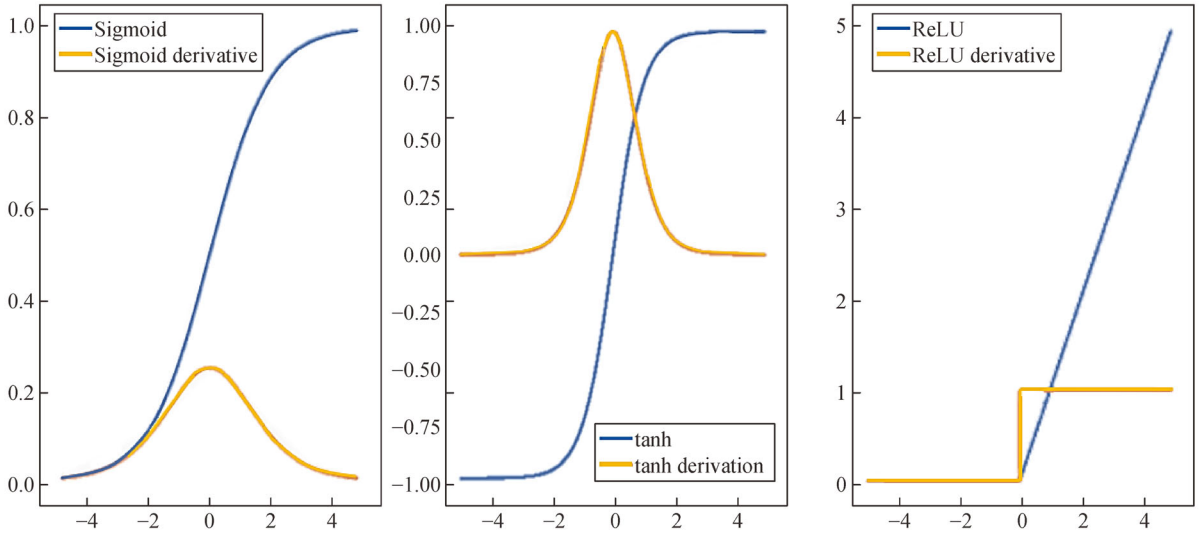

Fig. 3 Activation functions.

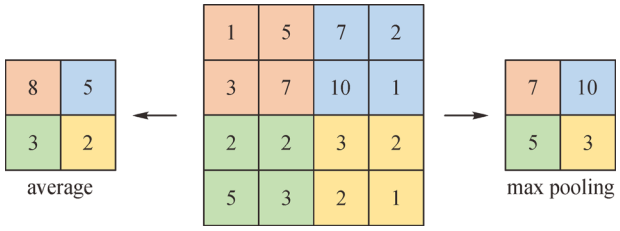

Fig. 4 Pooling operators.

labels. This can be considered a link between the classifier and the information output. As the name suggests, each neuron is connected to every neuron in the previous layer as a traditional neural network. Thus, the features extracted and learned in the shallow layers can be mapped onto the tag array. To attach the fully connected layer to the network, the size of the CNN output needs to be flattened into a vector of values. To obtain the classes of the input images, the softmax layer was finally located at the bottom of the CNN architecture. The output unit activation function for the multiclass classification problem is given by the softmax function:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}, \tag{3}$$

where $0 \leqslant \sigma(z)_j \leqslant 1$ and $\sum_{j=1}^{K} \sigma(z)_j = 1$. The output of the softmax layer represents the probability that the network associates the ith input with the jth class.

2.5 Softmax loss and stochastic gradient descent

Once the neural network is designed, the free parameters need to be optimized using learning algorithms to obtain the correct network output. The most common algorithm used to train a neural network on a training data set is backpropagation. Backpropagation allows the parameters to be updated by moving forward and backward across the network until the loss function between the predicted and actual outputs reaches its local minimum. The deviations between the predicted and actual classes are defined by the following:

$$loss = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{K} t_{ij} \ln \sigma(z)_j, \tag{4}$$

where N is the number of samples, K is the number of class, t_{ij} is a logical term that returns a value of 1 if the ith sample follows the jth class, and $\sigma(z)_j$ is the output for samples i and j. The most efficient way to minimize the deviations and update the weights is stochastic gradient descent using backpropagation. Starting from the initial values of the weight and loss, the algorithm provides the best weights by computing the global loss minimum according to the following equation:

$$w = w - \alpha \frac{\partial loss}{\partial w}, \tag{5}$$

where α indicates the learning rate, and w is the weight. When the gradient is positive, the weight is reduced; otherwise, when it is negative, the weight is increased. The amount of contribution that should be considered is defined by the learning rate. During the CNN training, the above process is repeated many times on a mini-batch of images until an epoch of iterations over the entire data set is completed. After each iteration, the gradient of the loss and an updating of the weights are applied.

3 Deep convolutional neural network with transfer learning

Humans have the inherent ability to acquire knowledge while learning an activity and use such knowledge to solve related tasks. Similarly, transfer learning is a widespread deep learning application employed for pretrained networks as the starting point of a new classification scenario. Fine-tuning a network using transfer learning is typically much faster than training a network from scratch and provides a high accuracy level even with fewer training images. Most pretrained networks have been developed for the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), the reference benchmark for large-scale object recognition [19]. These networks have been trained on over a million images contained in the ImageNet database and can classify and detect 1000 object categories. The great variety and size of the training data set guarantees a better ability to extract features from various types of images. Pretrained networks have different levels of accuracy, speed, and size, which should be considered when looking for a solution to a specific problem. In the present study, eight of the fastest pretrained CNN models were tested to conduct transfer learning on concrete damage classification tasks. From the experimental study aimed at evaluating the influence of training parameters, the most performing neural network was selected in terms of speed and accuracy.

3.1 Pretrained neural networks

Deep learning models have layered architectures in which different layers correspond to different feature learning. As mentioned above, the last layer is a fully connected layer to obtain the final output. This architecture allows the use of weights in feature extractor layers as the starting point for the training process and adapting them in response to the new problem. Furthermore, it allows the replacement of the

last fully connected layer, softmax layer, and classification output layer for the fine-tuning of the different classification problems. In the present study, eight different pretrained networks were modified, including AlexNet, SqueezeNet, ShuffleNet, ResNet-18, GoogLeNet, ResNet-50, MobileNet-v2, and NASNet-mobile.

AlexNet is a CNN trained on over a million images in the ImageNet LSVRC-2010 contest and can classify images belonging to 1000 different categories [20]. AlexNet won the ILSVRC-2012 contest with a top-5 error rate of 15.3%, compared to 26.2% for the second-best approach. The network contains five convolutional layers, some of which are connected to max pooling layers, and three fully connected layers with a softmax layer and a classification output layer at the end. To transfer the layers for the new classification task, the last three layers need to be fine-tuned to have a fully connected layer with a size equal to the number of classes in the new data.

SqueezeNet is a smaller CNN that achieves the same accuracy as AlexNet on ImageNet, with 50-times fewer parameters and is 510-times smaller than AlexNet [21]. The SqueezeNet architecture is composed of 1×1 squeeze convolutional layers and expands the layers with a mix of 1×1 and 3×3 convolution filters. Unlike most networks, in which the last layer with learnable weights is a fully connected layer, in SqueezeNet, the last learnable layer is the final convolutional layer. To retrain a pretrained network to classify new images, the convolutional layer should be replaced with a new convolutional layer having the same number of classes as the number of filters.

ShuffleNet is a computationally efficient CNN designed for devices with limited computing power [22]. The new architecture introduces a pointwise group convolution and channel shuffle to reduce the computational costs while maintaining the level of accuracy. ShuffleNet achieves about 13 times less of actual speedup compared to AlexNet while maintaining comparable accuracy on ImageNet data set.

ResNet-18 and ResNet-50 are residual CNNs with architectures of 18 and 50 layers deep, respectively [23]. Their introduction was the most revolutionary work in the deep network architecture. Because an increasing network depth through a simple stacking of the layers can lead to a vanishing gradient problem or to overfitting of the data, the core idea of the authors was the introduction of the so-called "identity shortcut connection" to bypass one or more layers. Residual connections allow the parameter gradients to propagate more easily from the output layer to the previous network layers, making it possible to train deeper networks with greater accuracy.

GoogLeNet was the winner of the ILSVRC-2014 competition, and as the name suggests, was developed by a Google team [24]. To reduce the use of the computing resources while increasing the depth and width of the network, a new architecture called "inception" was proposed. The idea of an inception layer is to apply filters with the most accurate detail (1×1) to a bigger version (5×5) that can work in parallel at the same level. To avoid a parameter explosion on the inception layers and perform faster computations, a 1×1 convolution is added as a dimension reduction module. Using the bottleneck approach, the depth and width can be increased without computational and overfitting problems. Instead of using a fully connected layer, the ends of the inception modules were connected to the global average pooling layer.

MobileNet-v2 [25] is a refinement of the v1 [26] mobile architecture, which introduced the idea of replacing the convolutional layers with depth-wise separable convolutions in order to decrease the computational cost. The convolution layer is split into a depth-wise convolution layer to filter the input and a 1×1 pointwise convolution layer to combine the filtered values and create new features. The new architecture presents an inverted residual structure, where the input and output residual blocks are opposite those of traditional residual models. This type of layer, based on the bottleneck principle, reduces the amount of data flowing through the network with a relevant computational gain. Furthermore, to help in gradient propagation, residual connections are introduced, as in the ResNet network.

NASNet-mobile was proposed as an algorithm to learn network architectures directly on the data set considered [27]. The authors searched for the best convolutional layer on the smallest CIFAR-10 data set and applied it to the ImageNet data set by stacking together multiple times. The NASNet model allows accurate results to be achieved with smaller model sizes and less complexity.

The main properties of the pretrained networks analyzed are reported in Table 1.

Table 1 Pretrained model properties

network	number of layers	size (Mb)	parameters (millions)
AlexNet	25	227	61
SqueezeNet	68	4.6	1.24
ShuffleNet	173	6.3	1.4
ResNet-18	72	44	11.7
GoogLeNet	144	27	7
ResNet-50	177	96	25.6
MobileNet-v2	155	13	3.5
NASNet-mobile	914	20	5.3

4 Experimental results and analysis

The classification accuracy of a deep learning model is strongly influenced by the size and quality of the image data set. However, most common data sets contain a

limited number of images, often showing only ideal laboratory conditions with no real structural surfaces or environmental conditions as a background. The proposed experimental program implementing a neural network for the automatic classification of concrete damage is based on the construction of a reference image data set, the analysis and detection of a reference artificial intelligent system using the transfer-learning approach, and finally the optimization and validation of the defined neural network. Before defining the reference neural network, the best set of hyperparameters was identified for each model. The training was conducted in a MATLAB 2019b environment on a workstation with the following configurations: Intel® Core™ i7-3537U CPU@ 2.00–2.50 GHz, 8.00 GB of RAM, and an NVIDIA GeForce GT720M GPU.

4.1 Image data set

Transfer learning is applied using data collected from the Internet, field bridge inspections, and Google Street View. This made it possible to select a greater and different variety of civil infrastructure images, increasing the practicality of research in the real field by simulating the acquisition from robotic platforms or drones. To match the input size of the networks, all image sizes were adjusted by a rescaling operation. The model trained on small images learns fewer features than one trained on large images, which are the most important to achieve proper efficiency in the classification task. The robustness of the networks, with the aim of being invariant to distortions in the image data, has been ensured by applying randomized augmentation techniques such as rotation, reflection, and translation. Furthermore, using an augmented image datastore to transform training data for each epoch increases the amount of training data and prevents the network from overfitting and learning the specific details of the training images. Figure 5 shows examples of datastore images classified as “undamaged”, “cracked”, and “delaminated”.

The original data set included 1352 images. To evaluate the network performance during training and detect if the network overfits the details of the training data, the data set is split into 80% for the training set and 20% for the validation set. The validation set contains images that the network has never seen prior to the validation process.

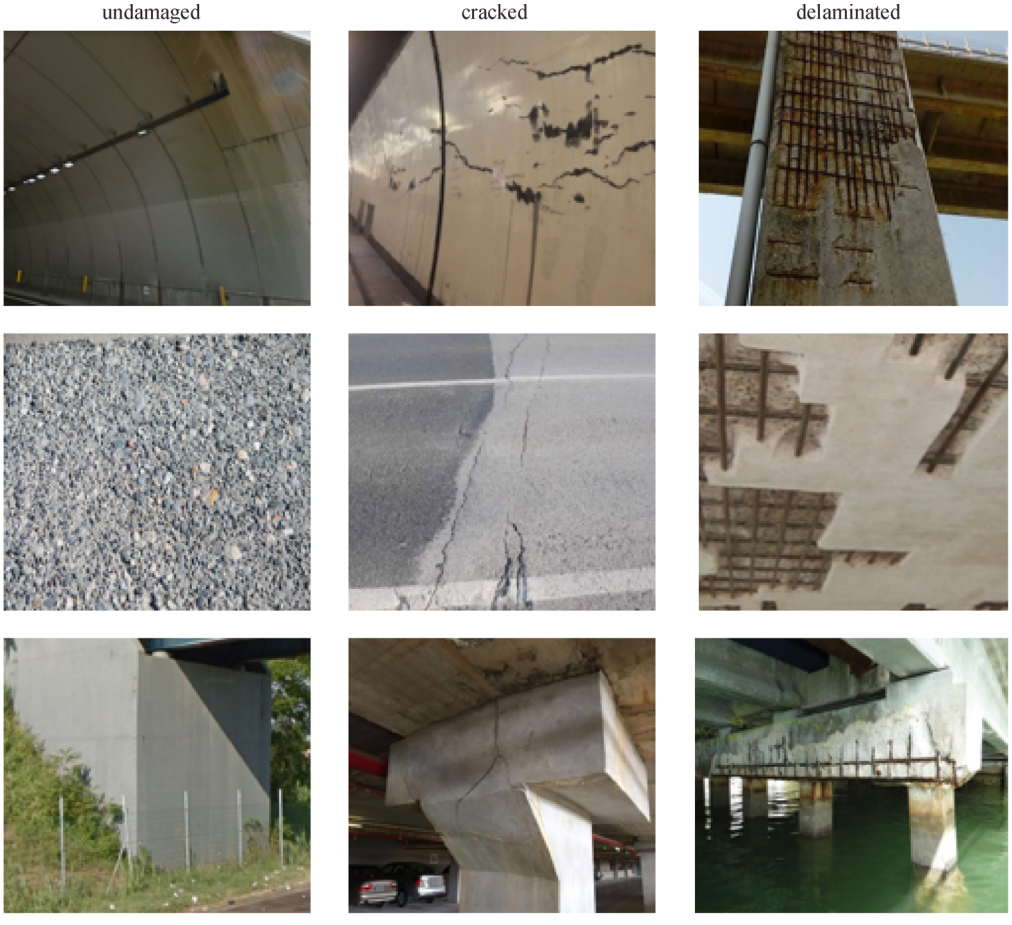

Fig. 5 Image data set examples.

When the training loss decreases and the validation loss increases, the training should be stopped because the network learns details regarding the training set that are irrelevant to the new images. The data set details for each class are presented in Table 2.

Table 2 Data set details

class	number of images	training set	validation set
undamaged	443	355	88
cracked	441	353	88
delaminated	468	374	94

To avoid unbalanced classes, the same order of magnitude is considered for the quantity of images.

4.2 Neural network training

The implementation of deep neural networks was achieved using the deep learning toolbox in MATLAB. Images are automatically labeled according to folder names and stored using the "imageDatastore" function. By installing the proper deep learning toolbox of each model, the architecture can be visualized, and the network layers are modified based on the layer replacements required by the transfer-learning technique. To make the images compatible with the input size of the network and perform randomized preprocessing operations, the "augmentedImageDatastore" is used. Once the training options are specified, the network is trained by employing the "trainNetwork" function, specifying the augmented image datastore, layers, and options.

After creating the data set and modifying the network architecture for transfer learning, the best training algorithm settings, that is, the learning rate, mini-batch size, and epoch number, should be determined to increase the network behavior. Other parameters, such as the dropout rate, have not been considered in the optimization process, as are not present in all networks and therefore are considered with the original value. Both the classification accuracy and training time are considered as the metric for an evaluation of such hyperparameters. Because their best value cannot be estimated from the data, the optimization process is applied through the iterations for various permutations. By monitoring the training progress, their influence on the training dynamics and stability can help in the detection of the best combination. The use of a mini-batch of the training data set to minimize errors and update the weights guarantees a faster convergence and better generalization, which is an approximation of the entire data set. Smaller learning rates require a long training time given the minor changes made to the weight update, whereas larger training rates require fewer training epochs but could lead to a suboptimal final set of weights or divergence. Given the above considerations, a good combination was found with a learning rate of 0.001 and a mini-batch size of 32. Finally, a maximum number of 12 epochs was selected as the appropriate number to reach an asymptotic behavior. The classifiers were trained using 396 iterations and validated every epoch. For each validation, the accuracy was evaluated as the ratio between the number of correctly classified images and the total number of images in the validation set.

The plot below (Fig. 6) shows a comparison among the different networks, between the validation accuracy and the time required to train the networks after 12 epochs, and the magnitude of the marker representing the networks is related to their byte dimension.

A wide variety of CNN architectures were obtained through a training experiment, including from those having the fastest and lowest precision to those having the greatest

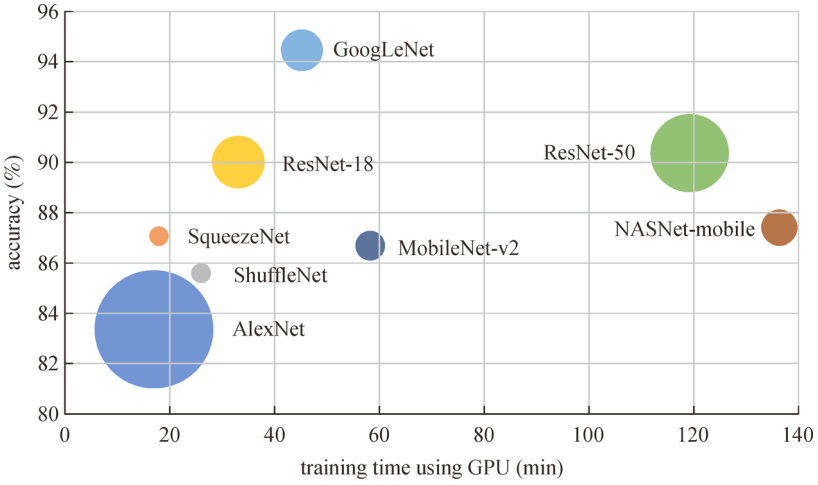

Fig. 6 Comparison of pretrained network.

complexity and accuracy. The pretrained network more related to automating the damage assessment was GoogLeNet, with a validation accuracy of 94.44% and a training time of approximately 45 min. Furthermore, owing to its relatively small disk and memory size, it is also compatible with predictions using mobile sensors with low computational resources or for its distribution over the Internet.

4.3 Neural network details and performance

The entire GoogLeNet architecture is 22 layers deep, with 3 convolution layers, 9 inception modules stacked linearly, and 2 deep layers, with a last fully connected layer. All convolutions inside the network use ReLU as the activation function. The input layer takes images of size 224 × 224 with RGB color channels. The detailed GoogLeNet architecture is presented in Table 3.

To retrain the pretrained network to classify "undamaged", "cracked", and "delaminated" images, the last fully connected layer is replaced with a new fully connected layer containing three outputs. A new classification layer was replaced by specifying the new class labels.

Using the set of hyperparameters specified in the previous paragraph, the following accuracy and loss progress are obtained for each iteration, after a training time on a single GPU of approximately 45 min (Fig. 7). The first aspect that can be observed is the quick drop in the loss function that motivates the optimal learning rate set. The slope of the accuracy trend highlights the effectiveness of transfer learning in achieving a high accuracy within a short amount of time.

Furthermore, the training progress showed the same error trends for both the validation and training sets. This means that the model behaves correctly on both the training and validation data, showing a general validity. For the training data set without data augmentation, the model achieved 100% accuracy after 150 iterations, leading to an over-specialization of the noise and details of the training data set. As a result, the model performance decreased and the validation accuracy dropped to 88.89%.

Once the network is trained, the confusion matrix is calculated between the true and predicted labels from the validation data set. Putting the true classes in the rows and the predicted classes in the column, the diagonal and off-diagonal regions correspond to the correct and incorrect observations, respectively (Fig. 8).

To better understand the performance measurement, a receiver operating characteristic (ROC) curve was plotted (Fig. 9). The ROC curve shows how the true positive and false positive rates relate, applying decision threshold values across an interval from 0 to 1. For each threshold, the true positive ratio (*TPR*), also called the sensitivity, and the false positive ratio (*FPR*) are calculated as follows:

$$TPR = \frac{TP}{TP + FN}, \tag{6}$$

Table 3 GoogLeNet pretrained model architecture

type	filter size / stride	output size	total learnables	depth
convolution	7 × 7 / 2	112 × 112 × 64	9472	1
max pool	3 × 3 / 2	56 × 56 × 64		0
convolution	3 × 3 / 1	56 × 56 × 192	114944	2
max pool	3 × 3 / 2	28 × 28 × 192		0
inception (3a)		28 × 28 × 256	163696	2
inception (3b)		28 × 28 × 480	388736	2
max pool	3 × 3 / 2	14 × 14 × 480		0
inception (4a)		14 × 14 × 512	376176	2
inception (4b)		14 × 14 × 512	449160	2
inception (4c)		14 × 14 × 512	510104	2
inception (4d)		14 × 14 × 528	605376	2
inception (4e)		14 × 14 × 832	868352	2
max pool	3 × 3 / 2	7 × 7 × 832		0
inception (5a)		7 × 7 × 832	1043456	2
inception (5b)		7 × 7 × 1024	1444080	2
avg pool	7 × 7 / 1	1 × 1 × 1024		0
dropout		1 × 1 × 1024		0
linear		1 × 1 × 1000	1025000	1
softmax		1 × 1 × 1000		0

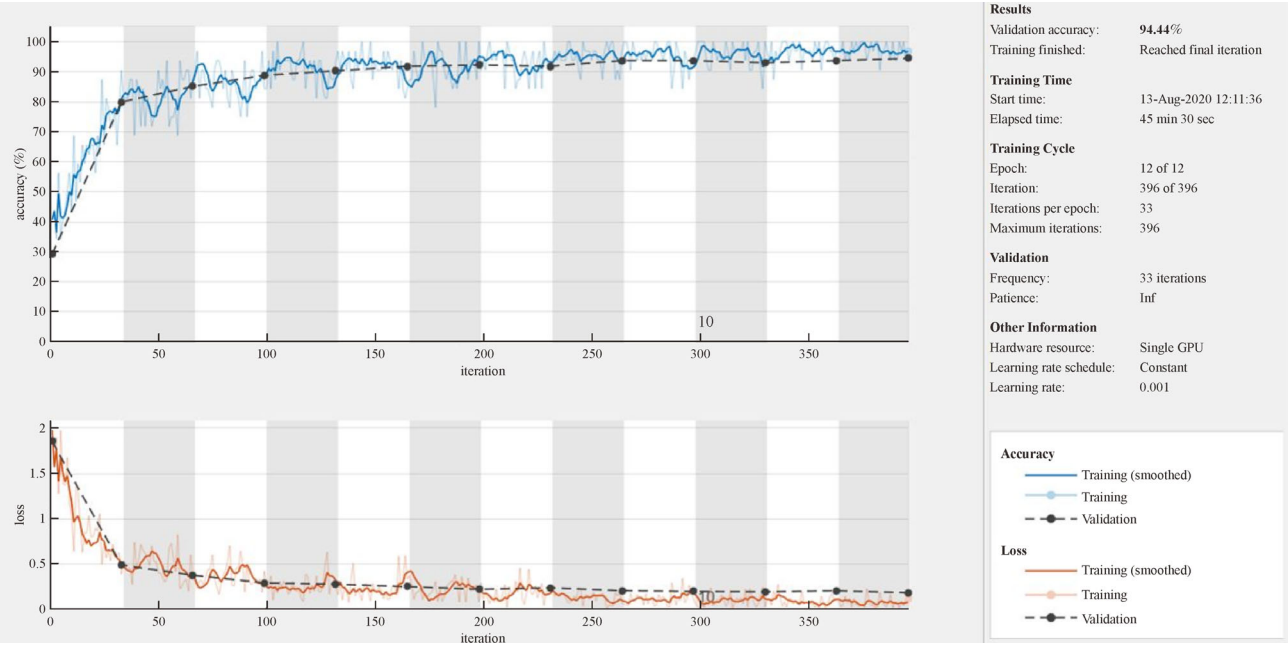

Fig. 7 Training progress.

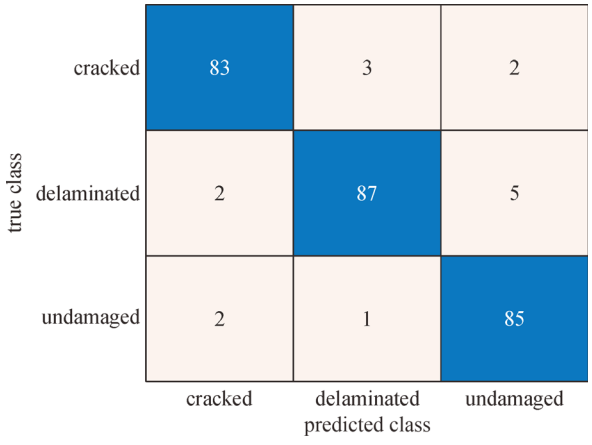

Fig. 8 Confusion matrix chart.

$$FPR = \frac{FP}{FP + TN}, \tag{7}$$

where TP is the number of positive instances correctly classified, FN is the number of positive instances misclassified, FP is the number of negative instances incorrectly classified, and TN is the number of negative instances correctly classified. Consequently, the proportion of negative instances correctly classified based on the total number of negative instances can be defined as the following a complementary metric:

$$Specificity = 1 - FPR. \tag{8}$$

When the threshold decreases, more positive values are obtained, thus increasing the sensitivity and decreasing the specificity. Similarly, when the threshold increases, more negative values are obtained, thus increasing the specificity and decreasing the sensitivity.

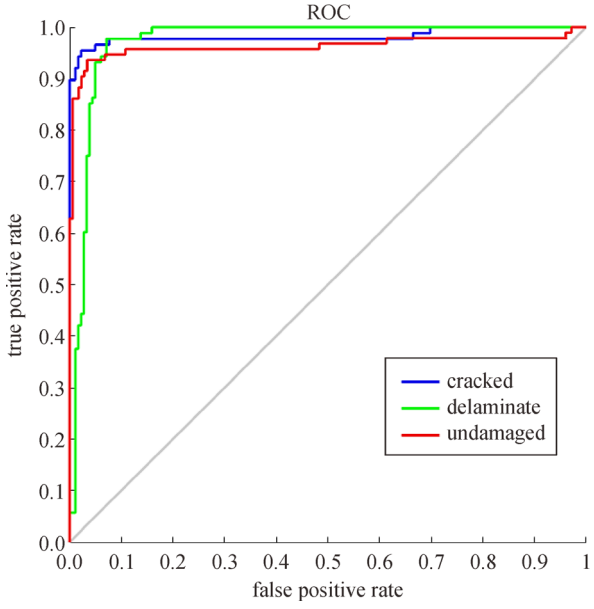

Fig. 9 Receiver operating characteristics.

In addition to detecting the most appropriate trade-off between sensitivity, specificity, and threshold, the ROC curve can be effectively employed to study the discriminative ability of the model. A classifier with curves close to the upper-left corner show a good measure of separability.

By contrast, an area under the curve (AUC) of zero means that the model reciprocates the classes and predicts the negative class as a positive and vice versa. When the AUC is 0.5, the model has no class discrimination capability and corresponds to a random model. The ability of the model to distinguish between classes is confirmed by the AUC measurements of approximately 0.98, 0.97, and 0.96 for the "cracked", "delaminated", and "undamaged" class, respectively.

The network correctly predicted 255 images in the validation set containing 270 images, achieving a global accuracy of 94.44%. The "cracked" class presents an accuracy of 94.31%, the "delaminated" class has an accuracy of 92.56%, and the "undamaged" class achieves a maximum accuracy of 96.59%. Figure 10 shows some validation results for correct and incorrect classifications.

From the above image, relevant considerations can be highlighted. First, the image correctly classified as "delaminated", which shows a street art picture on a pier, confirms the robustness of the model compared to the noisy images. Further observations can be made regarding the misclassification. In general, the probability of belonging to the predicted class is low or, in all cases, less than approximately 90%. Relating to the image misclassified as "undamaged" with a probability of 97.9%, one possible explanation is the problem of distance from the delaminated area on the bottom deck between the two piers, which makes the prediction difficult even for human inspection. The further image incorrectly classified with a high probability of 99.9% contains some relevant details of the delaminated surface, such as the oxide color, texture of the aggregates, and wide and deep cracks that mislead the network. By contrast, the images classified as "cracked" but containing delaminated elements demonstrate the need to improve the automatic classification of damage in the case of the presence of multiple classes.

In conclusion, GoogLeNet has shown the best performance in solving the problem of automating the inspection of civil infrastructures. The high accuracy of this model, compared to the quality of the images, is extremely close to real conditions, confirming its suitability as a good basis for developing future studies for a fully automated inspection.

5 Conclusions and perspectives

In this paper, a new deep neural network for automatic classification of the main defects in civil infrastructures is

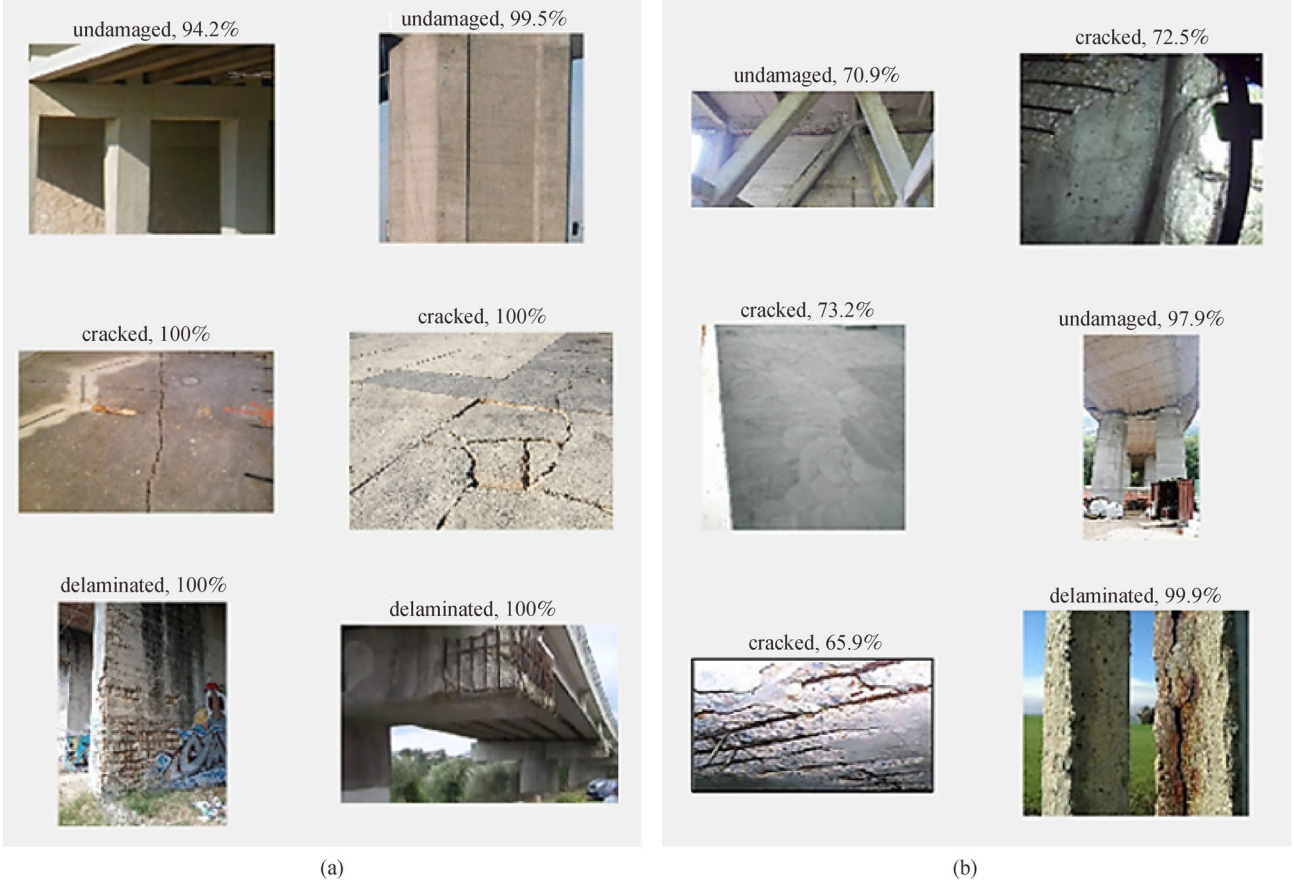

Fig. 10 Validation results: (a) correct classification and (b) misclassification.

proposed. The study employed an existing CNN modifying the last fully connected layer, building a robust CNN capable of classifying images with noisy backgrounds, containing different defect configurations for bridge, tunnel, and pavement structures. To define the most suitable network for concrete damage classification, an experimental investigation was conducted on eight pretrained networks. The major contribution was to train the network with images that simulate real collections in the field by means of a robotic platform, low-cost system, or drone. Nevertheless, an accuracy of 94.44% is achieved, highlighting the suitability of integration in smart management systems for the automatic inspection and assessment of civil infrastructures. Future developments will focus on the improvement of network learning not only to identify the defect classes, but also their detection and quantification.

References

1. Anitescu C, Atroshchenko E, Alajlan N, Rabczuk T. Artificial neural network methods for the solution of second order boundary value problems. Computers, Materials & Continua, 2019, 59(1): 345–359
2. Guo H, Zhuang X, Rabczuk T. A deep collocation method for the bending analysis of Kirchhoff plate. Computers, Materials & Continua, 2019, 59(2): 433–456
3. Fan Z, Wu Y, Lu J, Li W. Automatic pavement crack detection based on structured prediction with the convolutional neural network. arXiv preprint arXiv: 1802.02208, 2018
4. Zhang L, Yang F, Daniel Zhang Y, Zhu Y J. Road crack detection using deep convolutional neural network. In: IEEE International Conference on Image Processing 2016. Phoenix, AZ: ICIP, 2016, 3708–3712
5. Kim B, Cho S. Automated vision-based detection of cracks on concrete surfaces using a deep learning technique. Sensors (Basel), 2018, 18(10): 3452–3469
6. Hung P, Su N, Diep V. Surface classification of damaged concrete using deep convolutional neural network. Pattern Recognition and Image Analysis, 2019, 29(4): 676–687
7. Zhu J, Song J. An intelligent classification model for surface defects on cement concrete bridges. Applied Sciences (Basel, Switzerland), 2020, 10(3): 972–990
8. Feng C, Zhang H, Wang S, Li Y, Wang H, Yan F. Structural damage detection using deep convolutional neural network and transfer learning. KSCE Journal of Civil Engineering, 2019, 23(10): 4493–4502
9. Song Q, Wu Y, Xin X, Yang L, Yang M, Chen H, Liu C, Hu M, Chai X, Li J. Real-time tunnel crack analysis system via deep learning. IEEE Access: Practical Innovations, Open Solutions, 2019, 7: 64186–64197
10. Makantasis K, Protopapadakis E, Doulamis A, Doulamis N, Loupos C. Doulamis Nikolaos. Deep convolutional neural networks for efficient vision based tunnel inspection. In: IEEE International Conference on Intelligent Computer Communication and Processing 2015. Cluj-Napoca: Romania ICCP, 2015, 335–342
11. Patterson B, Leone G, Pantoja M, Behrouzi A. Deep learning for automated image classification of seismic damage to built infrastructure. In: Proceedings of the 11th National Conference in Earthquake Engineering 2018. Los Angeles, CA: Earthquake Engineering Research Institute, 2018
12. Gulgec N S, Takac M, Pakzad S N. Structural damage detection using convolutional neural networks. In: Barthorpe R, Platz R, Lopez I, et al., eds. Model Validation and Uncertainty Quantification. Conference Proceedings of the Society for Experimental Mechanics Series. Cham: Springer, 2017, 331–337
13. Cha Y J, Choi W, Suh G, Mahmoudkhani S, Büyüköztürk O. Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types. Computer-Aided Civil and Infrastructure Engineering, 2018, 33(9): 731–747
14. Soukup D, Huber-Mörk R. Convolutional neural networks for steel surface defect detection from photometric stereo images. In: Bebis G, et al., eds. Advances in Visual Computing. ISVC 2014. Lecture Notes in Computer Science. Cham: Springer, 2014, 668–677
15. Li J, Su Z, Geng J, Yin Y. Real-time detection of steel strip surface defects based on improved YOLO detection network. IFAC-PapersOnLine, 2018, 51(21): 76–81
16. Rawat W, Wang Z. Deep convolutional neural networks for image classification: A comprehensive review. Neural Computation, 2017, 29(9): 1–98
17. LeCun Y, Bengio Y, Hinton G. Deep learning. Nature, 2015, 521(7553): 436–444
18. Nair V, Hinton G E. Rectified linear units improve restricted Boltzmann machines. In: Proceedings of the 27th International Conference on Machine Learning 2010. Haifa: Israel ICML-10, 2010, 807–814
19. Russakovsky O, Deng J, Su H, Krause J, Satheesh S, Ma S, Huang Z, Karpathy A, Khosla A, Bernstein M, Berg A C, Fei F L. ImageNet large scale visual recognition challenge. International Journal of Computer Vision, 2015, 115(3): 211–252
20. Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks. Communications of the ACM, 2017, 60(6): 84–90
21. Iandola F N, Han S, Moskewicz M W, Ashraf K, Dally W J, Keutzer K. SqueezeNet: AlexNet-level accuracy with 50X fewer parameters and < 0.5 MB model size. 2017, arxiv.org/abs/1602.07360
22. Zhang X, Zhou X, Lin M, Sun J. ShuffleNet: An extremely efficient convolutional neural network for mobile devices. In: Conference on Computer Vision and Pattern Recognition 2018. Salt Lake City: USA IEEE/CVF, 2018, 6848–6856
23. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition 2016. Las Vegas, 2016, 770–778.
24. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Boston: USA IEEE, 2015, 1–9
25. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L C. MobileNetV2: Inverted residuals and linear bottlenecks. In:

Conference on Computer Vision and Pattern Recognition 2018. Salt Lake City: USA IEEE/CVF, 2018, 4510–4520

26. Howard A G, Zhu M, Chen B, Kalenichenko D, Wang W, Weyand T, Andreetto M, Adam H. MobileNets: Efficient convolutional neural networks for mobile vision applications. 2017, arXiv:1704.04861
27. Zoph B, Vasudevan V, Shlens J, Le Q V. Learning transferable architectures for scalable image recognition. In: Conference on Computer Vision and Pattern Recognition 2018. Salt Lake City: USA IEEE/CVF, 2018, 8697–8710