Received: 8 August 2017 | Revised: 6 June 2018 | Accepted: 11 June 2018

DOI: 10.1002/stc.2230

RESEARCH ARTICLE

WILEY

Convolutional neural networks for automated damage recognition and damage type identification

Ceena Modarres[1] | Nicolas Astorga[2] | Enrique Lopez Droguett[1,2] | Viviana Meruane[2]

[1]Mechanical Engineering Department, University of Maryland, College Park, Maryland, USA

[2]Mechanical Engineering Department, University of Chile, Santiago, Chile

Correspondence
Enrique Lopez Droguett, Mechanical Engineering Department, University of Chile, Santiago, Chile.
Email: elopezdroguett@ing.uchile.cl

Funding information
Chilean National Fund for Scientific and Technological Development (Fondecyt), Grant/Award Numbers: 1160494 and 1170535; Petroleum Institute, Abu Dhabi, UAE

Summary

Recurring expenses associated with preventative maintenance and inspection produce operational inefficiencies and unnecessary spending. Human inspectors may submit inaccurate damage assessments and physically inaccessible locations, like underground mining structures, and pose additional logistical challenges. Automated systems and computer vision can significantly reduce these challenges and streamline preventative maintenance and inspection. The authors propose a convolutional neural network (CNN)-based approach to identify the presence and type of structural damage. CNN is a deep feed-forward artificial neural network that utilizes learnable convolutional filters to identify distinguishing patterns present in images. CNN is invariant to image scale, location, and noise, which makes it robust to classify damage of different sizes or shapes. The proposed approach is validated with synthetic data of a composite sandwich panel with debonding damage, and crack damage recognition is demonstrated on real concrete bridge crack images. CNN outperforms several other machine learning algorithms in completing the same task. The authors conclude that CNN is an effective tool for the detection and type identification of damage.

KEYWORDS

convolutional neural networks, crack detection, damage diagnosis, deep learning, structural monitoring

1 | INTRODUCTION

Structural health monitoring and effective preventative maintenance programs are critical to ensuring cost-effective and safe operational efficiency. Larger systems like bridges or mining equipment can have unidentified damage that reduces the reliability of a system dramatically. In these cases, inspections are incredibly important to ensure that a dangerous and costly failure does not occur.

In many cases, individuals with domain knowledge and a detailed checklist conduct regularly scheduled inspections. In scheduling these inspections, a preventative maintenance program must balance system's safety and the cost and availability of a qualified individual. In some cases, these inspections can become dangerous or difficult. For example, mining equipment can be in hard to reach, dark, and wet places. The equipment can also be operationally complex or improperly designed. As a result, human inspections can leave an organization liable if an injury is sustained.

In response to these shortcomings, organizations have adopted automated visual inspection systems to reduce costs, increase accessibility, and improve safety. One such visual inspection system is unmanned aerial vehicles that

autonomously produce photographs of areas susceptible to damage.[1] The process can be further automated with the use of computer vision-based approaches to identify damage in real time. This paper proposes a particular deep learning-based computer vision approach for automated inspection.

Several computer vision-based approaches have been developed in the maintainability and reliability community to detect damage from images. Many of the presented methodologies act upon a particular type of image, like thermal imagery[2] or three-dimensional scene reconstruction[3] in order to simplify the detection task. Other approaches require significant filtering, feature extraction, and transformations to address varying field conditions like lighting or varying scale.[4,5]

In the past, data selectivity and significant preprocessing were required for traditional classification algorithms, like support vector machines[6] or artificial neural networks (ANNs), to identify the presence of damage from images with unique surfaces, scales, and lightings. However, advances in computational power over the past decade have made unique and complex neural network architectures practically feasible. This area of research, known as deep learning, has revolutionized many domains including image recognition, speech recognition, image segmentation,[7-9] and fault diagnosis of rolling element bearings.[10]

One of these architectures, the convolutional neural network (CNN), has wide application in the field of image processing and can handle distorted or shifted patterns.[11] Thus, CNN does not require preprocessing and can accurately detect damage and distinguish between different types of damage.

In this paper, the authors propose a CNN-based approach for detecting and classifying damage. With CNN, precise and accurate damage inspections can be conducted simply from images. Applications and variations of CNN to various other specific damage detection contexts (e.g., corrosion and crack detection) have been explored recently.[12,13] Moreover, organizations and analysts do not require complicated preprocessing procedures that could introduce additional methodological errors.

The paper proceeds as follows. In Section 2, the authors motivate and explain CNN. In Section 3, the authors validate and test CNN on a damage-based synthetic data set encompassing the identification of two damage types, whereas in Section 4, the CNN model is tested in a real-world concrete bridge cracks. Section 5 concludes by qualifying the intellectual contribution and recommending future works.

2 | CONVOLUTIONAL NEURAL NETWORKS

2.1 | Artificial neural networks

Artificial neural networks are a traditional learning algorithm and the basis of all deep learning methods. In order to grasp deep learning algorithms like CNNs, it is necessary to explore the simple feed-forward network.

Linear learning algorithms involve a combination of features with learned weights that predict a target. Artificial neural networks add an additional level of hierarchy to linear learning algorithms by developing a weighted combination of activation functions.[14] Essentially, a finite number of traditional learners are weighted to predict the target. This process is demonstrated in Equations 1–3 where a weighted (β) linear combination of hidden units or nodes ($\boldsymbol{z}$) is constructed from weighted ($\boldsymbol{w}$) features ($\boldsymbol{x}$) to predict a target ($\boldsymbol{y}$).

$$z_j = h\left(\sum_{i=1}^{D} w_{ji}x_i + b_k\right), \tag{1}$$

$$a_k = \sum_{j=1}^{M} \beta_{kj}z_j + b_j, \tag{2}$$

$$y_k = \sigma(a_k) = \frac{1}{1 + e^{-a_k}}. \tag{3}$$

The σ in Equation 3 is an example of a sigmoid activation function. An activation function, which is also present as the h function in Equation 1, can introduce additional nonlinearity. A combination of linear models (linear regression) is produced if the activation function is simply the identity. However, if the sigmoid function is invoked, the problem is transformed into a combination of probabilistic classifications.

There are several alternatives to the sigmoid activation function. They include the softmax function (Equation 4) for K-class classification and the rectified linear unit (ReLU; Equation 5) for nonzero values. The authors leverage both of these activation functions in their analysis.

$$S(\mathbf{z}) = \frac{e^{a_k}}{\sum_{k=1}^{K} e^{a_k}}, \tag{4}$$

$$ReLU(\mathbf{z}) = \max(0, a_k). \tag{5}$$

A reliability/maintainability analyst can control the number of hidden layers or the number of units within each hidden layer for his or her purposes. The generalizability of the algorithm may decline at very large degrees of freedom (learnable parameters). Thus, it is particularly important to address challenges associated with overfitting when training larger networks.

2.2 | Training

Machine learning classifiers typically maximize a cross-entropy loss function (Equation 6), which minimizes the difference between the probabilistic prediction ($\boldsymbol{f}$) and ground truth ($\boldsymbol{y}$) from data ($\boldsymbol{x}$). This is equivalent to maximizing the likelihood of seeing the target data given the predicted data and weights.

$$\max L(x) = -\sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log f_k(x_i). \tag{6}$$

This maximization for ANN is not analytically tractable. To train a scalable neural network, leveraging the batch or stochastic gradient descent (SGD) algorithm is necessary (Equation 7). Deriving this procedure for ANN's interconnected graph structure is often referred to as backpropagation, but the two terms are used interchangeably in this section. The canonical form of the SGD algorithm is displayed below:

$$\theta^{(\tau+1)} = \theta^{(\tau)} - \gamma \nabla L\left(\theta^{(\tau)}\right). \tag{7}$$

The full set of learnable weights (θ) is updated at iteration (τ) with the gradient of the loss function (∇L) scaled by a preset learning rate hyperparameter (γ).[14] The weights associated with hidden nodes and input data are updated according to the respective error (Equations 8–9).

$$\boldsymbol{w}^{(\tau+1)} = \boldsymbol{w}^{(\tau)} - \gamma\left(\frac{\partial L}{\boldsymbol{w}^{(\tau)}}\right), \tag{8}$$

$$\beta^{(\tau+1)} = \beta^{(\tau)} - \gamma\left(\frac{\partial L}{\beta^{(\tau)}}\right). \tag{9}$$

In order to create a linearly scalable learning procedure, the weights can be updated on a minibatch or subsample of the data. This also allows an analyst to update their model periodically as more data become available.

In practice, a minibatch of data is propagated through an initialized feed-forward network to produce a prediction (forward pass). The value of the gradient error, or loss, is calculated, and the weights associated with the network are updated (backward pass). Now, in the second iteration, the process starts over with a new minibatch of data. The process continues until it reaches an iteration limit or the analyst manually stops the learning process. Backpropagation and algorithms that are rooted in this paradigm are particularly powerful because they can learn as additional training observations become available. Thus, the model can adapt to possibly changing circumstances in predictive contexts.

Other research has introduced variations to the SGD algorithm for more efficient neural network training. One of these optimization procedures is adaptive moment estimation (ADAM). Rather than using the gradient of the loss function directly, ADAM updates moments of the gradient to update the parameter vector. ADAM has been shown to improve training performance on high-parameter vectors and large data sets.[15]

ADAM calculates bias corrected moments of the gradient (∇L) to update the parameter vectors. Equations (10)–(12) show the process of updating the first (m) and second (v) moments. A δ parameter is introduced as an exponential decay for these moving averages.[15]

$$g^{(\tau)} = \nabla L\left(\theta^{(\tau)}\right), \tag{10}$$

$$m^{(\tau)} = \delta_1 \cdot m^{(\tau\ -\ 1)} + (1-\delta_1)\cdot g^{(\tau)}, \tag{11}$$

$$v^{(\tau)} = \delta_2 \cdot v^{(\tau\ -\ 1)} + (1-\delta_2)\cdot g^{2(\tau)}. \tag{12}$$

Calculating $g^{(\tau)}$ in Equation 10 requires the evaluation of parameters associated with different layers of the neural network. However, the analytical derivative of computational graphs is difficult to calculate. In the deep learning context, automatic differentiation is often leveraged to approximate the $g^{(\tau)}$ values.[16] Using the chain rule, derivatives are broken down into smaller and easily solvable subexpressions that are calculated iteratively. In practice, an analyst sets either the independent or dependent variable constant and recursively solves for different nodes. This paper does not go into detail on how to implement an automatic gradient because this functionality is built into most deep learning libraries, but resources are available in the References.[16-18]

Because the first and second moments are often initialized at zero, Equations 13–14 correct for the possibility of a bias towards zero.

$$\widehat{m}^{(\tau)} = \frac{m^{(\tau)}}{(1-\delta_1)}, \tag{13}$$

$$\widehat{v}^{(\tau)} = \frac{v^{(\tau)}}{(1-\delta_2)}. \tag{14}$$

Equation 15 updates the parameter vector scaled by a step-size parameter (γ). A near-zero smoothing parameter ϵ is introduced to prevent division by 0.

$$\theta^{(\tau)} = \theta^{(\tau\ -\ 1)} - \gamma \frac{\widehat{m}^{(\tau)}}{\left(\sqrt{\widehat{v}^{(\tau)}} + \in\right)}. \tag{15}$$

In practice, the decay parameter for the first moment is often set as a default to δ_1 =0.9, the second moment set to $\delta_2 = 0.999$, and the step size set to $\gamma = 0.001$.

2.3 | Deep learning and convolutional neural networks

Deep learning is a subfield of machine learning research focused on modeling hierarchical representations or abstractions to define higher level concepts.[19] The field has gathered inspiration from human information processing systems. For example, human speech and visual perception systems are hierarchical in nature.[20-23]

For more complicated classification or regression problems, a single layer of hidden nodes is often ill suited to model hierarchical representations. Deep learning aims to address this problem by adding multiple hidden layers. Higher degrees of freedom associated with additional hidden layers may cause algorithms to overfit and capture noise rather than signal. Thus, deep learning architectures often have a unique structure that prevents overfitting and are tailored to the task.

Various deep learning architectures have been shown to outperform simpler techniques in speech recognition, natural language processing, and image processing.[24,25] In this paper, the authors invoke an architecture particularly well suited for image processing: CNNs (see Figure 1).

In order to counter high dimensionality and potential overfitting, local connectivity is assumed. Local connectivity assumes correlation between neighbor points. With this assumption, each hidden unit is connected to only a subset of the input provided from the previous layer. In practice, a filter transforms a square receptive local field of features (pixels for images). This approach reduces the quantity of learnable parameters and computational cost.

As visualized in Figure 1, CNNs use these locally connected receptive fields or learnable convolutional filters to create feature maps (convolutional layer). This collection of activation maps run through a pooling layer, which reduces

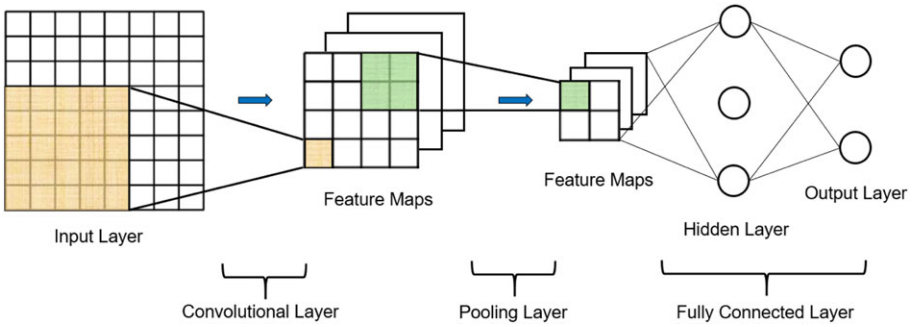


FIGURE 1 Convolutional neural network architecture

the size of each feature map for computational efficiency. The reduced feature maps are treated as an input to additional layers, and ultimately, the model predicts a target. This architecture allows a network to identify patterns regardless of scale or location.

The convolutional layer involves a set of learnable filters ($W_{i,j}$) convolving local regions ($x_{i,j}$) for features relevant to the output prediction. The convolutional layer produces a set number (k) of feature maps (h). The analyst can set the amount and size of the filters. Equation 16 displays the discrete convolution for an identity activation function:

$$h_{i,j}^{k} = \left(W^{k} * \ x\right)_{ij} + b_{k}. \tag{16}$$

Each set of weights W^k that transform the image into a feature map is replicated across the entire data set. Thus, each receptive field is the product of the same set of learnable filters. This also reduces the number of free parameters.

In the pooling and subsampling step, nonoverlapping hidden units within a feature map h^k are aggregated to reduce the overall size in the subsequent layer (y). Equation 17 displays max pooling, which identifies the max of a particular region.

$$y_{i,j}^{k} = \max_{p,q} h_{j \ + \ p,j \ + \ q}^{k}. \tag{17}$$

Pooling and subsampling operations reduce the operational efficiency especially if additional convolutional layers are added. However, max pooling increases translation invariance, that is, the pooled value would identify the same max feature regardless of its location within the neighborhood.

Typical CNN architectures often include fully connected hidden layers before generating a prediction. These layers mimic the ANN structure discussed in Section 2.1.

2.4 | Regularization

CNN's high degrees of freedom could produce models that perform well on the training data but generalize poorly to nontraining data. However, regularization procedures can be adopted to prevent this overfitting. One such procedure is known as dropout.[26]

In dropout, samples from a Bernoulli distribution (r) determine whether node's output ($\boldsymbol{z}_j$) drops from a hidden layer. The outcome is fewer learnable parameters, and a trained network that does not rely too heavily on any single output. Equation 18 presents the Bernoulli distribution from which r is drawn. Equation 19 displays the thinning of the hidden layer outputs by dropping nodes with an element-wise product of the Bernoulli distributed random variables and the hidden layer output. A graphical representation of dropout is presented in Figure 2.

$$r_{j}^{(\tau)} \sim \text{Bernoulli}(p), \tag{18}$$

$$\widetilde{\mathbf{z}}^{(\tau)} = \boldsymbol{r}^{\tau} \circ \mathbf{z}^{(\tau)}. \tag{19}$$

This sampling procedure prevents nodes from identifying noise and using this to segment and differentiate different types of damage. Alternative approaches include early stopping, which is a black art heuristic of ending convergence once training and validation errors begin to diverge, and data augmentation, which involves the process of flipping

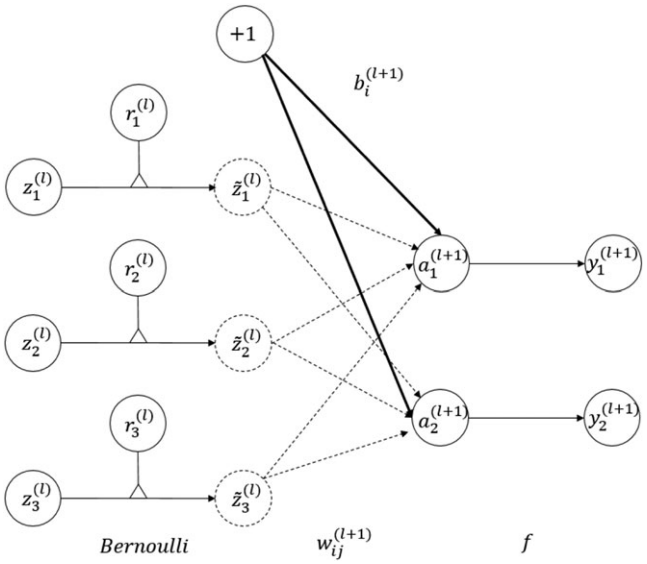


FIGURE 2 Graphical representation of dropout regularization

and transforming the images before training. An additional alternative is weight decay, which includes regularizing terms within the network.

The value of techniques like dropout and early stopping cannot be understated. A model that can be employed in a production environment must identify significant patterns of differentiation between damage and not confuse noise with a distinguishing feature.

2.5 | Developing a model architecture in practice

In practice, developing a model architecture can be an arduous task. Most nonindustry settings do not have the necessary infrastructure to conduct a large-scale search algorithm to explore the hyperparameter space of these networks. Thus, an often time-consuming manual search is required. This is currently an active area of research. Google's AutoML project seems to provide some promise in using reinforcement learning and evolutionary algorithms to design neural networks, but timely implementation of these approaches would require significant compute power unavailable to a typical researcher.[27,28]

Nonetheless, various rules of thumb have been developed in the larger deep learning community. First, neural networks should follow a pyramid structure. A network should have many nodes in the first few layers (wide) and reduce the amount of nodes progressively before reaching the output layer. In essence, this allows the network to compress the lower level features in the early layers in order to make easy classification decisions in the later layers. Pyramid structures have been shown to produce better results.[29]

Second, neural networks are sensitive to the initial learning rate. The default value is often 0.01, but performance varies by network architecture. This is likely the most important hyperparameter to optimize.[30] Third, if the size of your data set is not relatively vast, your algorithm will be susceptible to overfitting. Thus, the inclusion of regularization methods like dropout and early stopping becomes increasingly important. Fourth, it is authors' experience that a larger minibatch will speed up the training process but will generate noisier updates to the loss function.

Finally, weight initialization can also affect model convergence. The weights are often sampled from a Gaussian distribution with a mean (μ) of 0 and standard deviation (σ) of 1.[7] The sigmoid activation function in conjunction with Gaussian sampled initialization schemes can produce a vanishing/exploding gradient problem that prevents the network from updating during backpropagation. Moreover, recent research proposals to neural network architecture, like batch normalization and ReLU functions, have reduced this risk dramatically.[31] New initialization methods have been proposed in several papers.[32,33]

Transfer learning is an important aspect of practical deep learning. The idea behind transfer learning is to pretrain a CNN on a larger task, like classifying many different types of damage on terabyte-scale data. Then, for a particular damage identification task (e.g., cracks on one type of bridge), previous network's weights are frozen, and additional

layers are trained on top of the frozen layers. This not only saves an analyst time but also creates the opportunity to generalize a CNN to the identification of many image-based structural reliability objectives.

Until automatic model selection becomes a prevalent approach to machine learning, developing deep learning architectures will require significant effort. Therefore, the knowledge discussed in this section only scratches the surface of model development for a particular task. There are many unique attributes to model development for damage (e.g., crack) detection than, for example, differentiating between dogs and fish on internet images. Unique attributes of model building like regularization methods (e.g., dropout and weight decay), weight initialization, activation function selection, and layer sizes require exploration and experimentation in context. In the following section, a particular architecture for damage detection and classification is proposed.

3 | CONVOLUTIONAL NEURAL NETWORK-BASED APPROACH FOR DAMAGE DETECTION AND CLASSIFICATION

In this section, the authors propose a specific CNN architecture for the purpose of damage detection and classification. In order to complete this task, the authors constructed a network architecture that could be trained to perform the aforementioned maintenance tasks on different types of damage. The model is detailed in Section 3.1, validated on honeycomb structure images in Section 3.2, and tested on real concrete bridge cracks in Section 4.

3.1 | Convolutional neural network model architecture

On the basis of the discussion presented in Section 2.5, the proposed CNN model architecture is designed to process two-dimensional arrays representing pixel intensities in the damaged or undamaged images from honeycomb and concrete bridge structures. The architecture includes three stacks of convolutional–pooling layers. The first convolutional layer applies 4×5 filters resulting in 32 feature maps of 66×94, which are fed to a first pooling layer that reduces their dimensionality to 33×47. Next, a second convolutional layer applies 64 of 4×4 filters that are passed on to the pooling layer for reduction of the feature maps to 15×22. The last convolutional layer then processes these with sixty-four 4×3 filters and then pooling resulting in sixty-four 6×10 feature maps. All pooling layers perform max pooling as in Equation 17. Afterwards, there are three fully connected hidden layers of 2,024, 524, and 3 neurons, respectively. The neurons in each hidden layer (including convolutional/pooling layers) use a ReLU activation function to avoid exploding gradients, 0.75 dropout rate to prevent overfitting, ADAM optimization for efficient convergence, and a softmax cost function for entropically maximized probabilistic classification. All weights are initialized by sampling from a Gaussian distribution with a mean (μ) of 0 and standard deviation (σ) of 1. The proposed architecture is displayed in Figure 3.

The proposed architecture is designed to maximize the core advantage of a CNN in the context of structural damage images: learning hierarchical representation. In the following, we provide a discussion on the main building blocks of a CNN, namely, convolutional layers and pooling layers, and how they impact its ability to process structural damage images for damage recognition and type classification. We also compare these features with those of traditional feed-forward neural networks.

A convolutional layer delivers three properties: sparse weights, parameter sharing, and invariant representations. Instead of having every input unit interacting with every output unit as in a traditional neural network, a CNN uses sparse weights to allow for filters that are smaller than the actual image. In processing a small image (e.g., 30×30 pixels), a traditional neural network would still need half a million parameters and 900 inputs. However, a CNN can

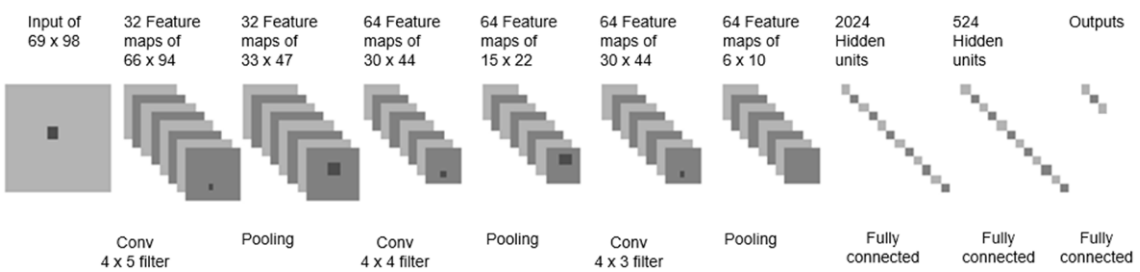


FIGURE 3 Convolutional neural network architecture for damage identification

detect small local meaningful features that only require a much smaller number of parameters, thus reducing the required computational resources and computing time.

In a traditional neural network, each element of the weight matrix is only used once when calculating the output of a given layer. However, parameter sharing allows a CNN to learn and use one set of parameters for multiple operations, thus making convolutional operations more efficient than dense matrix multiplications. Moreover, parameter sharing leads a convolutional layer to be invariant to translation. When dealing with images taken from a damaged structure, the convolutional layer outputs a two-dimensional feature map characterizing where certain damage objects (e.g., cracks) are in input images. Thus, when we feed a different image to the CNN with the object in a different location, its representation moves accordingly.

The multiple linear features generated by parallel convolutional operations are passed over to nonlinear activations. Typically, these outputs are then fed to pooling layers to progressively reduce the spatial size of the input representation. As a result, pooling also reduces the number of parameters and computations in the network, which also reduces overfitting. Furthermore, pooling makes the representation approximately invariant to small translations in the input image. For instance, a small distortion in the input would not change the output of pooling as one is taking the maximum (in max pooling) or the average value in a local neighborhood.

These properties of CNNs are fundamental for the problem at hand, that is, structural damage detection based on image processing. This is a task that usually involves careful construction and extraction of features from the data. For example, some common features include correlation, contrast, energy, homogeneity, and entropy.[4] These features are then fed to a traditional learning method with the task to identify, separate, and classify damage.[34,35] Feature extraction relies on prior engineering knowledge of the data such that choosing which features to include or exclude within the model is subjected to uncertainty and biases of the domain experts.

It is in this context that the proposed CNN architecture provides its most relevant contribution. By means of the aforementioned properties, a CNN delivers nonlinear representations of the input damage images to a higher level of abstraction and complexity isolated from the touch of human engineers directing the learning.[24] The proposed CNN model architecture is designed to give suitable outputs for the discovery of complex and high-dimensional representations without the need for manual feature extraction.

3.2 | Model validation on honeycomb structure images

In order to evaluate the effectiveness of the proposed CNN-based approach, the authors evaluated several algorithms on a sample case of images synthesized to resemble debonding damage in honeycomb structures. The algorithm was trained to identify the presence of damage and differentiate between different types of damage.

Honeycomb structures minimize material and weight but have high strength (see Figure 4). They are used in a wide variety of applications in the aerospace and automobile industries. Nevertheless, due to manufacturing defects or impact loads, these structures can experience imperfect bonding or debonding between the skin and the honeycomb core. Instances of debonding reduce the bending stiffness of the composite panel, which causes detectable changes in its vibration characteristics.[36-38] To investigate the identification of debonding damage in sandwich structures, a database with different damaged and undamaged scenarios was created.

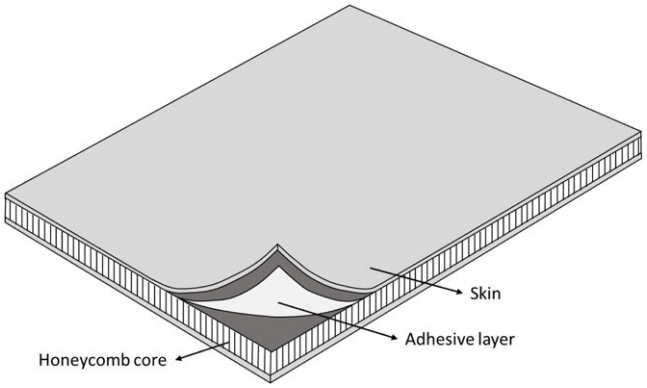


FIGURE 4 Scheme of a honeycomb sandwich panel

In the present study, the honeycomb panels are modeled with finite elements using a simplified three-layer shell model, and the adhesive layer between the skin and core is modeled using linear springs, with reduced rigidity in debonded sectors, as shown in Figure 5.

The numerical model is built in Matlab® using the Structural Dynamics Toolbox,[39] whereas the skins and honeycomb panel are modeled with standard isotropic four-node shell elements. The final model shown in Figure 6 has 10,742 shells and 7,242 spring elements. It has been demonstrated that this numerical model represents with precision a sandwich panel with debonding damage.[37]

The modal strain energy has shown to be a sensitive indicator of damage and has been frequently used to locate damage in one-dimensional structures, such as beams, frames, and truss structures.[40] The damage indices are computed using the modal strain energy of a plate in the undamaged and damaged states.[41] In the case of a plate-like structure, for a particular mode shape, ϕ_r, the strain energy associated with that mode shape is

$$U_r = \frac{D}{2}\int_0^b\int_0^a \left(\frac{\partial^2\phi_r}{\partial x^2}\right)^2 + \left(\frac{\partial^2\phi_r}{\partial y^2}\right)^2 + 2\nu\left(\frac{\partial^2\phi_r}{\partial x^2}\right)\left(\frac{\partial^2\phi_r}{\partial y^2}\right) + 2(1-\nu)\left(\frac{\partial^2\phi_r}{\partial x\partial y}\right)^2 dxdy, \tag{20}$$

where a and b are the plate dimensions and $D = Eh^3/12(1-\nu^2)$ is the bending stiffness of the plate. By dividing the plate in subregions, it is possible to locate damage by comparing the normalized strain energy of each subregion in the undamaged and damaged conditions. The energy associated with subregion *joke* for the rth mode shape is given by

$$U_{r,jk} = \frac{D_{jk}}{2}\int_{b_k}^{b_{k+1}}\int_{a_j}^{a_{j+1}} \left(\frac{\partial^2\phi_r}{\partial x^2}\right)^2 + \left(\frac{\partial^2\phi_r}{\partial y^2}\right)^2 + 2\nu\left(\frac{\partial^2\phi_r}{\partial x^2}\right)\left(\frac{\partial^2\phi_r}{\partial y^2}\right) + 2(1-\nu)\left(\frac{\partial^2\phi_r}{\partial x\partial y}\right)^2 dxdy; \tag{21}$$

therefore,

$$U_r = \sum_{k=1}^{N_x}\sum_{j=1}^{N_y} U_{r,jk}, \tag{22}$$

where N_x and N_y are the number of divisions in the x and y directions, respectively. The fractional energy at location jk is

$$F_{r,jk=}\frac{U_{r,jk}}{U_r}. \tag{23}$$

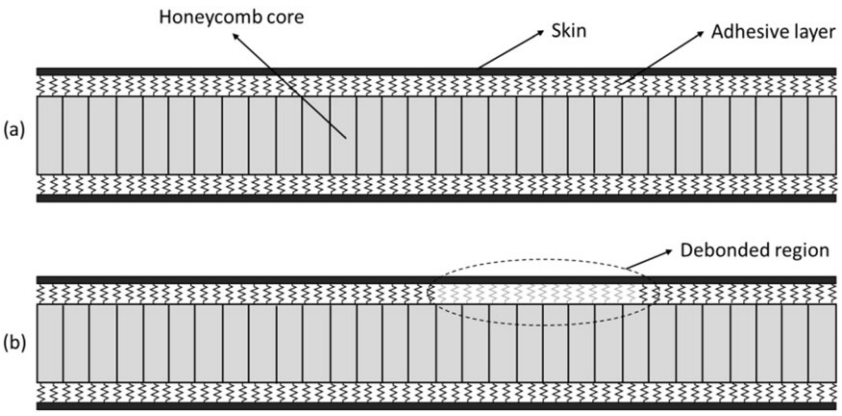


FIGURE 5 Lateral view of the numerical model: (a) undamaged panel and (b) panel with a debonded region

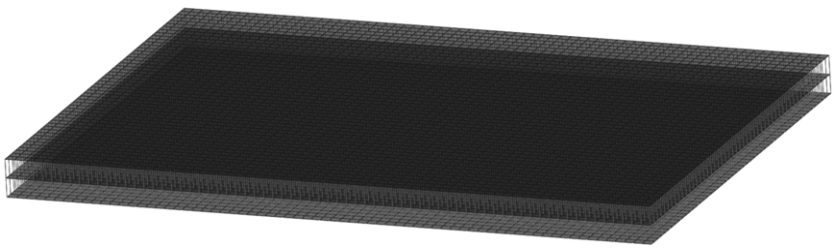


FIGURE 6 Finite element model of the sandwich panel

The same procedure can be used to compute the fractional energy at location *jk* for the *r*th mode of the damaged structure, $F^D_{r,jk}$. To account for all measured modes, the damage index for subregion *jk* is defined as

$$\beta_{jk} = \sum_{i=1}^{m} F^D_{r,jk} / \sum_{i=1}^{m} F_{r,jk}, \tag{24}$$

where *m* is the number of measured modes. Assuming that the collection of damage indices represents a population of a normally distributed random variable, a normalized damage index is obtained using

$$Z_{jk} = \left(\beta_{jk} - \overline{\beta}_{jk}\right) \Big/ \sigma_{jk}, \tag{25}$$

where $\overline{\beta}_{jk}$ and σ_{jk} represent the mean and standard deviation of the damage indices, respectively.

The panel used to construct the database consists of 0.25 × 0.35-m sandwich panels made from aluminum honeycomb core bonded to two aluminum skins.[37] The database is constructed with the following scenarios: 2,000 undamaged panels, 2,000 panels with circular-shaped debonded regions, and 2,000 panels with rectangular-shaped debonded regions. The location and size of the debonded regions are defined randomly with dimensions between 0.02 and 0.12 m.

For each panel in the database, the first six mode shapes are identified, and Equation 25 is used to obtain the distribution of damage indices through the panels. To simulate experimental measurements, the numerical modes are contaminated with 5% random noise. Figure 7 illustrates the damages indices obtained for three different scenarios. The pictures on the left show the damaged regions, whereas the pictures on the right show the distribution of damage indices through the panel.

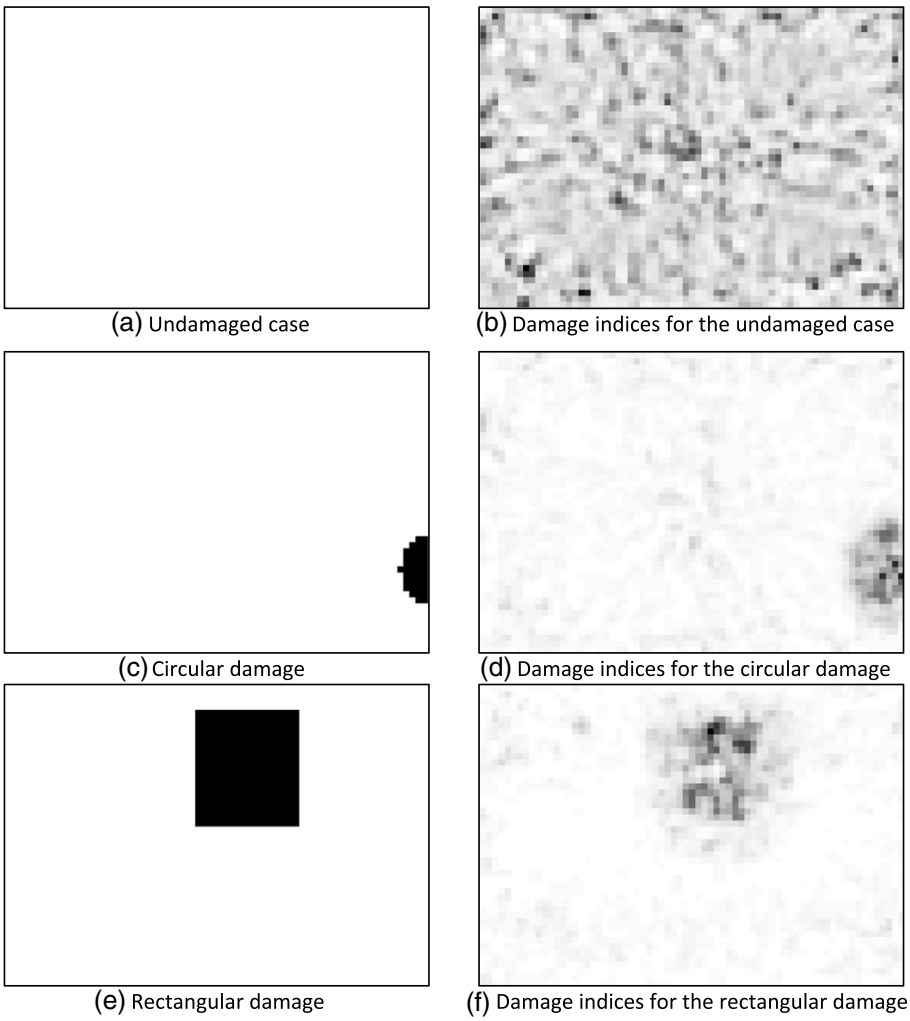


FIGURE 7 Examples of the damages indices obtained for different scenarios

A CNN model was trained and validated on the synthesized panel damage in Figure 7b,d,f. The model was trained on five thousand four hundred 69 × 98 images for a fixed 2,000 epochs. Exactly 10% of the data (600 images) were used as a test set, and 10% of the remaining training set was used for validation. To ensure the model was insusceptible to the vanishing or exploding gradient problem, verify that it works properly, can generalize well, and is insensitive to parameter initialization, the training procedure was rerun 20 times for 2,000 epochs (see Appendix A for details). Moreover, training and testing were carried out in a Linux-based computer with 4.2 GHz, 32 GB of RAM, a NVIDIA Tesla K40 GPU, and with Tensor flow 1.0, cuDNN 5.1, and Cuda 8.0.

CNN predicted the presence and type of damage at an accuracy of 98%. The confusion matrix denoting the performance of the model on each class is displayed in Table 1. The vertical labels on the left-hand side are associated with predicted values, whereas the horizontal labels at the top correspond to base truth.

Figure 8 presents the receiver operating characteristic for the CNN. The *y* axis represents the true-positive rate, or probability of detecting damage, and the *x* axis represents the false-positive rate. Often the threshold of confidence for a true positive can be varied in order to assess varying quality of performance. The area under the curve (AUC) represents the probability that a randomly chosen point was classified correctly as a true positive as opposed to an incorrect classification. Thus, a higher AUC reflects favorably on the performance of a model in precisely identifying potential damage. A lower AUC would mean more false positives and the potential for wasting resources on maintenance decisions.

Because the relative cost of an incorrect no damage prediction is much higher than an incorrect damage prediction, model accuracy is not a sufficient assessment of model quality. The precision and recall metrics are used to further explore model performance.

In classification problems, the precision metric represents the percentage of accurate predictions within a particular set of class predictions. In this case, it is important to explore whether the model can precisely predict the absence of damage, which it does at a rate of 98%. The recall is the percentage of a class that is identified correctly by the model. In this case, the recall metric for different types of damage provides information on model's ability to identify present damage and take proper corrective actions. For the circular and square damage types, the recall metrics were 98% and 95%, respectively.

Overall, the model performance provides support for CNN's ability to complete the tasks the authors have attributed to it. In the next section, the authors explore the performance of the proposed CNN-based approach and several other machine learning algorithms on a real-world concrete bridge crack data set.

TABLE 1 Synthetic damage confusion matrix

Image class	No damage	Circular damage	Square damage
No damage	0.980	0.009	0.005
Circular damage	0.015	0.947	0.005
Square damage	0.005	0.044	0.990

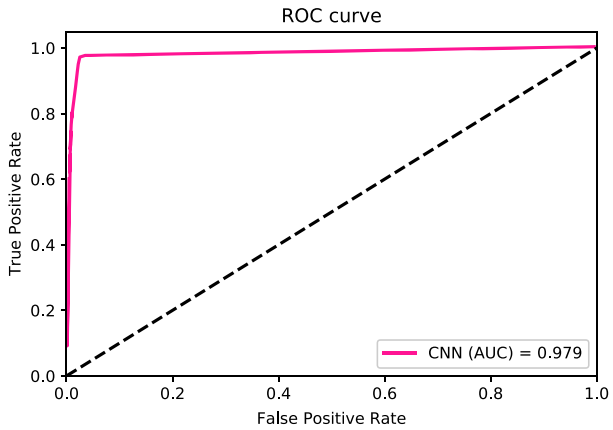


FIGURE 8 Receiver operating characteristic curve for convolutional neural network (CNN) on synthetic damage data. AUC: area under the curve

4 | REAL CONCRETE BRIDGE CRACK DATA SET

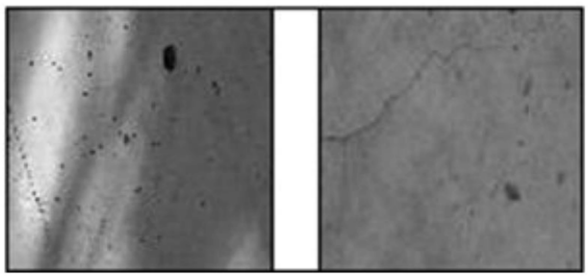
This section discusses an example of application of the proposed CNN-based approach in the context of concrete bridge structural health monitoring. Two scenarios are considered. The first one is composed of clean images of well-controlled concrete surfaces, where some light staining or lighting variations are present, but, overall, the quality of the concrete is high; this scenario is representative of concrete in a new structure (see Figure 9a). The second scenario encompasses noisy images of weathered and highly textured concrete surfaces, where staining is much more predominant and there is a higher variability on crack widths; the objective of this scenario is to represent situations that frequently occur during field inspections in which the deviations are prevalent from controlled imaging situations (see Figure 9b).

In both scenarios, the proposed CNN architecture was trained and tested with images from the corresponding dataset, that is, clean images for the first scenario and noisy images for the second one.

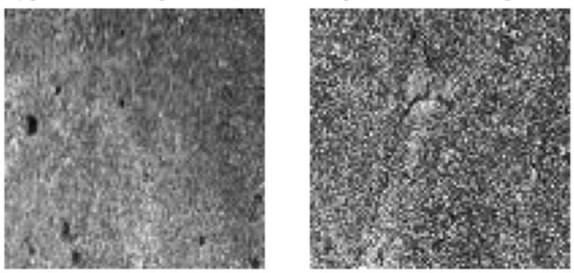
For both scenarios, the proposed CNN was trained on two thousand four hundred 96 × 96 real concrete crack images (see Figure 9) and compared with other machine learning algorithms. The data set was evenly distributed between images with and without damage (i.e., crack). The model was evaluated on a test set (10%), and performance was compared with a logistic regression with principal component analysis + logit, support vector machine, random forest, and multilayer perceptron (MLP). All hyperparameters were grid search optimized. Hardware and software configurations were the same as in the scenarios discussed in Section 3.

In this case, the only difference of this model to the one in the previous section is the use of square filters, which were more appropriate for the input. The details of the architecture are presented in Figure 10.

For the first scenario of clean set of images shown in Figure 9a, the proposed CNN achieves a predictive accuracy of 99.6%. From Table 2, note that the CNN also has better accuracy than all other algorithms.



(a) Clean set: typical undamaged (left) and damage (with crack, right) image examples.



(b) Noisy set: typical undamaged (left) and damage (with crack, right) image examples.

FIGURE 9 Concrete crack data

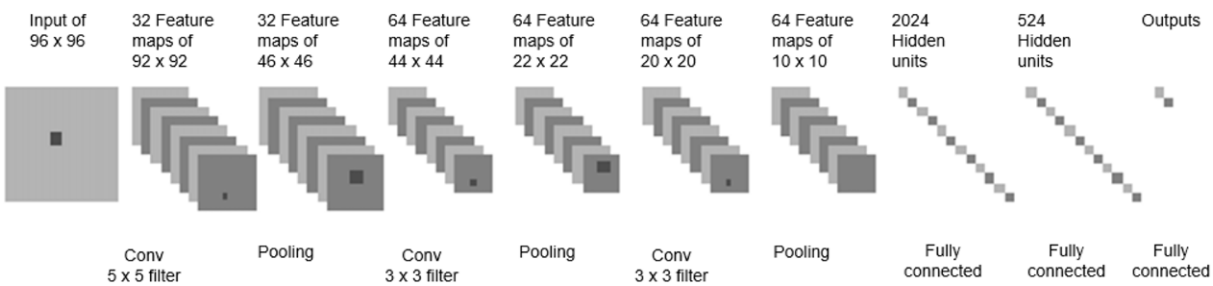


FIGURE 10 Convolutional neural network architecture for concrete crack classification

TABLE 2 Accuracy results for crack recognition experiment from clean images

Algorithm	Overall accuracy (%)
PCA + logit	69.2
SVM with RBF kernel	54.2
Random forest	96.3
MLP	93.7
Proposed CNN	99.6

Note. CNN: convolutional neural network; MLP: multilayer perceptron; PCA: principal component analysis; RBF: radial basis function; SVM: support vector machine.

The proposed CNN architecture also outperforms all other algorithms with a predictive accuracy of 98.8% on the noisy concrete crack images displayed in Figure 9b. The full results are presented in Table 3. It seems as though CNN's ability to predict hierarchical and nonlinear functions and its translation and scale invariance make it the most accurate model within this subset.

The model also attained desired precision and recall metrics: 98.3% precision on noncrack images and 98% recall on concrete cracks. The normalized confusion matrix is presented in Table 4.

In real-world settings, the crack images can vary in shape, size, texture, and many other attributes. Thus, evaluating the performance of a model in a controlled setting may be insufficient to show that CNN can be effective in identifying more heterogeneous cracks.

Thus, to verify the robustness of the CNN-based classifier for concrete bridge cracks, we now consider two different training datasets that are composed of varying proportions of the clean and noisy images shown in Figure 9a,b, respectively: (a) 75% of clean images and 25% of noisy images and (b) equal proportion of clean and noisy images. In both scenarios, both the CNN and MLP models were tested in a dataset composed of only noisy images.

Indeed, Figure 11a,b shows the receiver operating characteristic for the CNN and MLP models. Table 5 reports the accuracy, precision, and recall metrics for both scenarios. Note that for both scenarios, the CNN consistently outperforms the MLP for all metrics. The notably higher precision metrics signal that the implementation of CNN reduces the risk of false alarms, which can have cost- and time-saving effects on an organization. However, the high recall score signals that almost all cracks that theoretically exist would be identified by the algorithm. Of course, data quality and application settings will affect these outcomes in a production system.

On the basis of these results from a dataset with added heterogeneity (noisy plus clean images), the authors can argue that the CNN model is quite accurate and capable of handling real-world crack data. Thus, CNN is generalizable

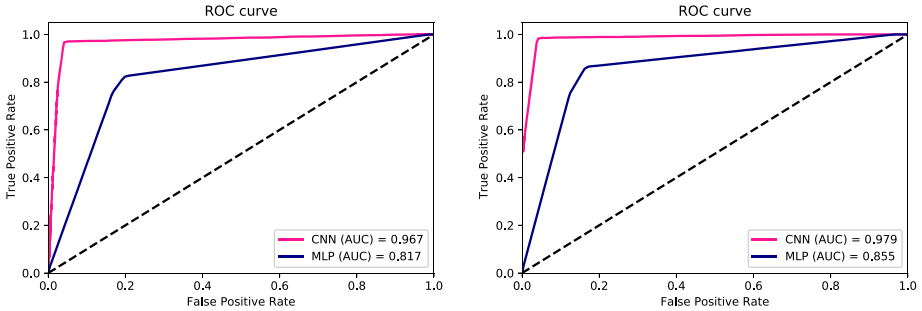
TABLE 3 Accuracy results for crack recognition experiment from noisy images

Algorithm	Overall accuracy (%)
PCA + logit	83.8
SVM with RBF kernel	50
Random forest	94.2
MLP	97
Proposed CNN	98.8

Note. CNN: convolutional neural network; MLP: multilayer perceptron; PCA: principal component analysis; RBF: radial basis function; SVM: support vector machine.

TABLE 4 Confusion matrix for concrete crack recognition

Image class	No crack	Crack
No crack	0.990	0.008
Crack	0.010	0.992



(a) 75% clean images training dataset. (b) 50% clean images training dataset.

FIGURE 11 Receiver operating characteristic curves for convolutional neural network (CNN) and multilayer perceptron (MLP). AUC: area under the curve

TABLE 5 Comparing metrics for CNN and MLP

	75% clean		50% clean	
Metrics	**CNN**	**MLP**	**CNN**	**MLP**
Accuracy	96.7	81.7	97.9	85.5
Precision	97.5	80.2	95.6	83.5
Recall	95.8	84.2	100	88.3

Note. Precision and recall metrics are reported for Class 1 only (i.e., crack is present). CNN: convolutional neural network; MLP: multilayer perceptron.

for the purpose of the damage identification tasks here considered and is a candidate for automated crack detection in real-world settings.

5 | CONCLUSIONS

Systems like oil pipelines, mining equipment, or large infrastructure projects can be inaccessible, hazardous, or even impossible for a human inspector to explore. In these cases, automatic damage detection is favorable. In this paper, the authors proposed the application of a deep learning-based approach to automate inspections reliably, accurately, and precisely. In particular, CNNs are employed to identify damage solely from images.

The authors introduced the deep architecture and training process of CNNs. The approach was then validated on an image data set of synthetic honeycomb structures. The model accurately classified different types of damage in these structures and attained other desirable performance metrics. Afterwards, the proposed CNN was trained, tested, and compared with other machine learning algorithms in several real-world concrete bridge crack scenarios. The network outperformed all other models explored.

The latter scenario involved training the CNN on a dataset composed of a mixture of clean and noisy concrete bridge images, whereas the testing was carried out solely on noisy images. The objective of this was clear: investigate the CNN model performance in detecting damage under harsher and somewhat different conditions as the ones seen during training. Although the CNN model achieved satisfactory performance metrics, its performance might suffer when faced with damage images with significant deviations from the context represented in the training dataset. Under these circumstances, the proposed CNN model architecture can be used as a starting point to explore new structural damage contexts. In many cases, the model would need to be retrained. Alternatively, transfer learning presents itself as a worthy approach to be explored in order to tackle the need for context changes.

The authors demonstrated how deep learning could be used to detect and identify damage, but the topics covered in this paper only scratch the tip of the iceberg. CNNs have shown tremendous promise not only in image classification and segmentation tasks but also in signal processing, topic modeling, cybersecurity, and many other domains. In the context of structural reliability, the authors have begun exploring the application of these methods for image segmentation. This approach not only can identify damage but also can locate and quantify damage within images of a structure.

Two major challenges need to be addressed concerning applications of deep learning techniques to engineering problems in general and to structural reliability in particular. The first involves the integration of deep learning techniques with physics of failure related to relevant structural damage mechanisms (e.g., crack initiation and propagation) to allow for dynamic damage tracking and therefore obtain prognostic metrics such as remaining useful life. The second one concerns the need to account for uncertainty in weights and prediction. For the latter, approximate inference in Bayesian neural networks has been used with various degrees of success on shallow architectures. The main problem with these methods (e.g., Laplace approximations, Hamiltonian Monte Carlo, and ensemble learning) is that they are not scalable for modern applications. However, Gal[42] has proposed a promising technique in 2016 called Monte Carlo dropout. Gal showed that in a neural network with arbitrary depth and nonlinearities, dropout samples applied before a weight layer are equivalent to an approximation of a probabilistic deep Gaussian process.

In summary, deep learning could revolutionize automated inspection systems, and this paper begins the process of exploring a wealth of potential applications.

ACKNOWLEDGMENTS

The authors would like to thank Prof. David Latinize from the George Mason University for kindly providing the concrete bridge images. The authors acknowledge the partial financial support of the Chilean National Fund for Scientific and Technological Development (Fondecyt) under Grants 1160494 and 1170535 as well as of the project entitled "Pipeline System Integrity Management" supported by the Petroleum Institute, Abu Dhabi, UAE.

ORCID


Enrique Lopez Droguett http://orcid.org/0000-0002-0790-8439

REFERENCES

1. Pagnano, Angelo, Michael Höpf, and Roberto Teti. "A roadmap for automated power line inspection. Maintenance and repair" Procedia CIRP 12 (2013): 234–239.
2. Ghidoni, S., Minella, M., Nanni, L., Ferrari, C., Moro, M., Pagello, E. and Menegatti, E. "Automatic crack detection in thermal images for metal parts." International Conference on Heating by Electromagnetic Sources, Padua. 2013.
3. Jahanshahi MR, Marsi SF, Padgett Curtis W, Sukhatme GS. An innovative methodology for detection and quantification of cracks through incorporation of depth perception. *Mach. Vis. Appl.* 2013;24(2):227-241.
4. Lattanzi D, Miller GR. Robust automated concrete damage detection algorithms for field applications. *J. Comput. Civ. Eng.* 2012;28(2):253-262.
5. Ellenberg, A.; Kontsos, A. M. F. and Bartoli, I. "Bridge related damage quantification using unmanned aerial vehicle imagery," *Struct Control Health Monit.*, vol. 23, n° 9, pp. 1168–1179, 2016.
6. Pontil M, Verri A. Support vector machines for 3D object recognition. *IEEE Trans Pattern Anal Mach Intell.* 1998;20(6):637-646.
7. Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*. 2012.
8. Kim, Yoon. "Convolutional neural networks for sentence classification" arXiv preprint arXiv:1408.5882 (2014).
9. Farabet C, Couprie C, Najman L, LeCun Y. Learning hierarchical features for scene labeling. *IEEE Trans Pattern Anal Mach Intell.* 2013;35(8):1915-1929.
10. Verstraete D, Ferrada A, Droguett EL, Meruane V, Modarres M. Deep learning enabled fault diagnosis using time-frequency image analysis of rolling element bearings. *Journal of Shock and Vibration*. 2017;17. https://doi.org/10.1155/2017/5067651
11. Zhang W, Itoh K, Tanida J, Ichioka Y. Parallel distributed processing model with local space-invariant interconnections and its optical architecture. *Appl Optics.* 1990;29(32):4790-4797.
12. Atha DJ, Jahanshahi MR. Evaluation of deep learning approaches based on convolutional neural networks for corrosion detection. *Journal of Structural Health Monitoring*. 2017. https://doi.org/10.1177/1475921717737051
13. Chen FC, Jahanshahi MR. NB-CNN: deep learning-based crack detection using convolutional neural network and naïve Bayes data fusion. *IEEE Transactions on Industrial Electronics*. 2017. https://doi.org/10.1109/TIE. 2017.2764844
14. Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. *The Elements of Statistical Learning*. Vol. 1. Springer, Berlin: Springer series in statistics, 2001.

15. Kingma, Diederik, and Jimmy Ba "Adam: a method for stochastic optimization" *arXiv preprint arXiv:*1412.6980 (2014).
16. Bengio J. Learning deep architectures for AI. *Machine Learning*. 2009;2(1).
17. Griewank A. On automatic differentiation. *Math. Program. Dev. Appl.* 1989;6(6):83-107.
18. Verma A. An introduction to automatic differentiation. *Curr. Sci. (Bangalore)*. 2000;78(7):804-807.
19. Deng L, Yu D. Deep learning. *Signal Processing*. 2014;7:3-4.
20. Hubel DH, Wiesel TN. Receptive fields, binocular interactions and functional architecture in the cat's visual cortex. *J. Physiology*. 1962;160(1):106-154.
21. Hubel DH, Wiesel TN. Receptive fields and functional architecture of monkey striate cortex. *J Physiology*. 1968;195(1):215-243.
22. Fukushima K. Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol Cybern*. 1980;34:193-202.
23. LeCun Y, Boser B, Denker JS, et al. Backpropagation applied to handwritten zip code recognition. *Neural Comput*. 1989;1(4):541-551.
24. LeCun Y, Bengio Y, Hinton G. Deep learning. *Nature*. 2015;521(7553):436-444.
25. Schmidhuber J. Deep learning in neural networks: an overview. *Neural Netw*. 2015;61:85-117.
26. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*. 2014;15(1):1929-1958.
27. Zoph, Barret, and Quoc, V. le. "Neural architecture search with reinforcement learning" *arXiv preprint arXiv:*1611.01578 (2016).
28. Real, Esteban, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc Le, Alex Kurakin"Large-scale evolution of image classifiers" *arXiv preprint arXiv:*1703.01041 (2017).
29. Ullah, Ihsan, and Alfredo Petrosino. "About pyramid structure in convolutional neural networks." Neural Networks (IJCNN), 2016 International Joint Conference on. IEEE, 2016.
30. Bengio Y. Practical recommendations for gradient-based training of deep architectures. In: *Neural Networks: Tricks of the Trade*. Springer Berlin Heidelberg; 2012:437-478.
31. Ioffe, Sergey, and Christian Szegedy. "Batch normalization: accelerating deep network training by reducing internal covariate shift." International Conference on Machine Learning 2015.
32. Glorot, Xavier, and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks." Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. 2010.
33. Kumar, Siddharth Krishna "On weight initialization in deep neural networks" arXiv preprint arXiv:1704.08863 (2017).
34. Sharma A, Amarnath M, Kankar PK. Feature extraction and fault severity classification in ball bearings. *J Vib Control*. 2016;22(1):176-192.
35. Wong PK, Zhong J, Yang Z, Vong CM. Sparse Bayesian extreme learning committee machine for engine simultaneous fault diagnosis. *Neurocomputing*. 2016;174:331-343.
36. Meruane V, del Fierro V, Ortiz-Bernardin A. A maximum entropy approach to assess debonding in honeycomb aluminum plates. *Entropy*. 2014;16(5):2869-2889.
37. Meruane V, del Fierro V. An inverse parallel genetic algorithm for the identification of skin/core debonding in honeycomb aluminium panels. *Struct Control Health Monit*. 2015;22(12):1426-1439.
38. Z. Liu, X. Zhong, T. Dong, C. He and B. Wu, "Delamination detection in composite plates by synthesizing time-reversed lamb waves and a modified damage imaging algorithm based on RAPID," *Struct. Control. Health Monit.*, vol. 24, N° 5, p. e1919, 2017.
39. Balmes, Etienne, Jean-Philippe Bianchi, and Jean-Michel Leclère. "Structural dynamics toolbox." Users Guide, Version 6 (2009).
40. Shi ZY, Law SS, Zhang LM. Structural damage localization from modal strain energy change. *J Sound Vib.* 1998;218(5):825-844.
41. Cornwell P, Doebling SW, Farrar CR. Application of the strain energy damage detection method to plate-like structures. *J Sound Vib*. 1999;224(2):359-374.
42. Gal, Yarin. "Uncertainty in deep learning." PhD Thesis, University of Cambridge, 2016.

How to cite this article: Modarres C, Astorga N, Droguett EL, Meruane V. Convolutional neural networks for automated damage recognition and damage type identification. *Struct Control Health Monit*. 2018;25:e2230. https://doi.org/10.1002/stc.2230

APPENDIX A

One potential concern with neural networks in general, and convolutional neural networks in particular, is the sensibility of the weight initialization and resulting stability of the results. To verify that the proposed architecture works properly and can generalize well and also verify its sensitivity to the parameters initialization, we take the example of application involving the concrete bridge crack detection discussed in Section 4 and provide the results from training 20 times the CNN-based model with random initialization of the weights and bias. Indeed, each one of the 20 runs was trained for 2,000 epochs, and the accuracy was calculated for the validation data set, as shown in Figure A1. The thick line represents the average accuracy, and the lighter line represents the standard deviation, both taken from the 20 runs. On the basis of Figure A1, one can see that the architecture is stable and has a small sensibility to the parameter initialization. Moreover, the best model in each one of the 20 runs of the validation data is chosen and used to estimate the accuracy in the test data set. This results in an average accuracy of 98.75% and standard deviation equals to 0.63%, thus corroborating the stability of the model with respect to the weight initialization.

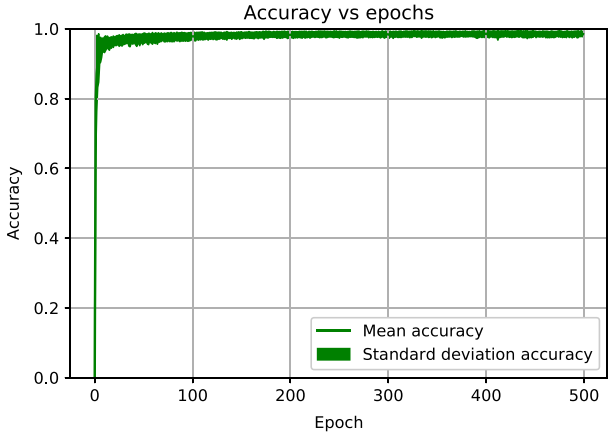


FIGURE A1 Average accuracy and standard deviation on the validation data set and based on 20 runs for 2,000 epochs each