Leadership in Sustainable Infrastructure

Leadership en Infrastructures Durables

Vancouver, Canada

May 31 – June 3, 2017/ *Mai 31 – Juin 3, 2017*

ADVANCED DEEP LEARNING BASED DAMAGE DETECTION USING COMPUTER VISION

Choi, Wooram[1], Cha, Young-Jin[1,3] and Buyukozturk, Oral[2]
[1] University of Manitoba, Canada
[2] Massachusetts Institute of Technology, USA
[3] young.cha@umanitoba.ca

Abstract: Prominent methods for monitoring structures currently rely on analysis of data measured from contact sensors that are physically attached to the structures. However, these approaches have a high possibility of false alarms due to noise, sensor malfunctions, and complex environmental effects. Thus, engineers still have to make on-site visits to confirm that damage has occurred. To address this challenge, this paper proposes a new method of concrete crack detection that uses a convolutional neural network (CNN) on images taken by cameras. In order to train the CNN, images of intact and cracked concrete surfaces structures were taken from the Faculty of Engineering buildings on the campus of the University of Manitoba. A CNN classifier—composed of eight components, including convolution, pooling, rectified linear unit, and softmax layers—was trained on the collected image dataset comprised of 40,000 images of 256×256-pixel resolution. A sliding window technique, which enabled us to scan images of any size larger than 256×256-pixel resolution, was implemented. The trained CNN classifier showed 97% accuracy in its analysis of 50 images larger than 4000×2500. In terms of computational cost, the trained network needs less than three seconds to scan those images.

1 INTRODUCTION

Vibration-based approaches using contact sensors have frequently been implemented for identifying damage and monitoring infrastructures. Lee et al. (2000) simulated a concrete beam member in a numerical model using the finite element method; they localized and quantified the crack through the observation of natural frequency shifts. The research was successful, but further research on multiple cracks is essential for a wider range of practical cases. A study that partly resolved the multiple-crack issue was carried out by Ruotolo and Surace (1997). They built numerical models of steel cantilever beams containing several cracks to localize and quantify existing damage through genetic-algorithm-based inverse problem solving. However, it was noted that the installation of multiple sensors is still required to deal with a structure with multiple members. Kurata et al. (2012) studied a large-scale remote monitoring system using wireless data transfer. They built a wireless system with various sensors attached to a suspension bridge to identify mode shapes. However, this type of sensor-implemented application is capable of monitoring only global behaviors. In addition, the complex requirements, such as installing multiple sensors and integrating their results effectively, are challenging. Resolving noisy sensor data also remains one of the toughest issues in monitoring with sensors. Several articles have pointed out the obstacles of environmental changes, such as changes in temperature and humidity (Xia et al. 2012; Cornwell et al. 1999). They imply that a monitoring system with sensors may need further calibrations to compensate for environmental effects, which leads to extra efforts to integrate even more sources. Even if one can solve all the aforementioned issues, a structure cannot be said to be damaged unless on-site inspections are conducted to confirm that the collected signals from sensors actually indicate the presence of damages.

These limitations have motivated researchers to propose many vision-based methods (Cha et al. 2016; Cha et al. 2017a; Chen et al. 2015), because visual data provide intuitively understandable information. A previous author (Abdel-Qader 2003) conducted a comparative study on finding concrete crack features using four different image-processing techniques, such as fast Haar transform (FHT), fast Fourier transform, Sobel edge detection, and Canny edge detection. Although the study found that FHT performs better than the others, the author pointed out that gradient-based algorithms are vulnerable to noisy inputs. In addition, finding edges is a fundamentally ill-posed problem (Ziou and Tabbone 1998) and cannot effectively generalize features existing in the real world. Even though machine learning algorithms has been integrated to several vision-based applications (Jahanshahi et al. 2013; Ramana et al. 2017), this type of methods always inherits the drawbacks of image-processing techniques and struggles to work well under real-world circumstances.

In this context, this paper aims to develop a robust method using a convolutional neural network (CNN) to overcome the current limitations of conventional vision-based methods, particularly for detecting concrete cracks. CNNs stand out in image classification among neural network applications (LeCun et al. 2015; Cha et al. 2017b). While standard neural networks (NNs) completely ignore the grid-like topology (e.g., images and time-dependent signals) of a dataset, CNNs are capable of effectively capturing grid-like topology. In addition, the sparsely connected neurons reduce computational burden, compared to the standard NNs.

2 ARCHITECTURE OF THE PROPOSED CNN

Figure 1 depicts the architecture of the proposed CNN, showing how input data are generalized by passing through the designed architecture. The proposed CNN consists of input, convolution, pooling, rectified linear unit (ReLU), softmax, and several other layers. The first layer is an input layer with 256×256×3-pixel resolution, where the dimensions indicate height, width, and channel, respectively. Input data pass through the architecture and are generalized with size reduction to 1×1×96 at L5. Then, the 96 data points are fed into a ReLU layer. Finally, a softmax layer predicts if the input is intact or cracked, after the last convolution layer (C4) in the diagram. In order to boost training speed, batch normalization is used after all the convolution layers except for the last one. A dropout layer is placed after L5, in which the components are irrelevant to size reduction but significantly helpful in terms of training speed.

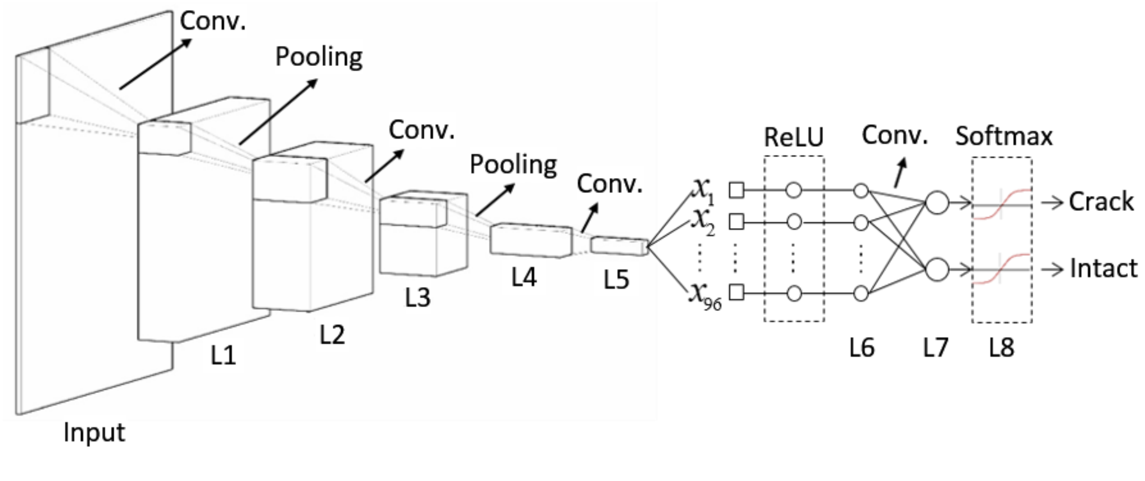


Figure 1: CNN architecture

3 CNN LAYERS

CNNs can be formed with single or multiple layers, but stacking multiple layers is common in most cases. If many layers exist within the architecture, the network can be defined as a deep CNN. Generally, the structure of CNNs includes input, convolution, pooling, activation, and output layers, where the layers between input and output layers are often defined by hidden layers. Other auxiliary layers, such as dropout and batch normalization layers, can be inserted between hidden layers as necessary. The operations of each layer are briefly explained in this section, but a more detailed explanation can be found in the manual for MatConvNet (Vedaldi and Lenc 2015), which is an open CNN toolbox with useful MATLAB wrappers, and MatConvNet is used to perform this study.

3.1 Convolution layer

Convolution is a key operation of CNNs; it first performs dot product calculation between an input array and a receptive field, and then the convolved values are summed to the output array. A receptive field conducts the convolution operation explained above across the input array. Figure 2 shows an example of convolution between a 5×5 input and a 3×3 receptive field. The size of the receptive field can be bigger or smaller than the given example, but it should be smaller than that of the input. The values of a receptive field are usually given by random numbers, which are later modified by a backpropagation algorithm using a gradient descent algorithm. It is notable that the size of the output in the example is reduced compared to the input array, which reduces computational costs. An optional hyperparameter of the operation is stride, describing how many shifts skip at each step, where higher stride values enable much faster calculation but also result in loss of input data.

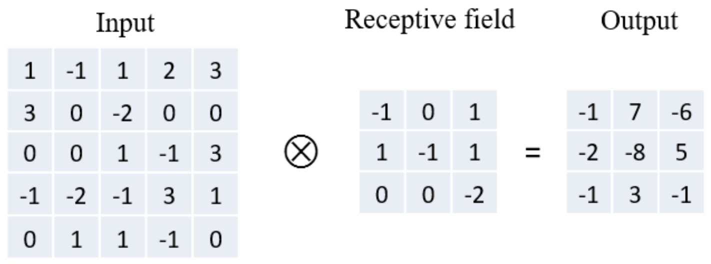


Figure 2: Example of convolution

3.2 Pooling layer

Pooling, also a key operation of CNNs, reduces the size of an input array. This process is frequently referred to as “down-sampling” or “sub-sampling”. There are different pooling operations, but mean and max pooling methods are widely used. Max pooling takes the maximum values, and mean (i.e., average) pooling takes mean values from a pooling region. A pooling layer performs the described operations across the input array. Figure 3 shows the examples of max and mean pooling methods, where the pooling size is 3×3. A comparative study revealed that max pooling does better than mean pooling in an image dataset (Scherer et al. 2010). We confirmed that max pooling outperforms mean pooling in our dataset.

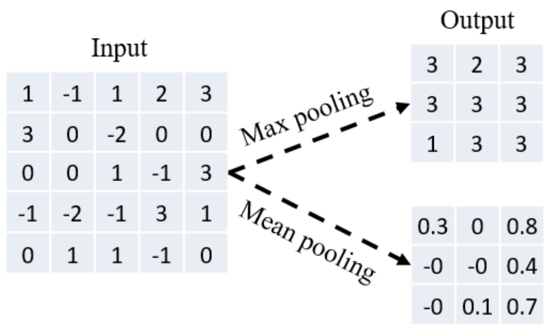


Figure 3: Examples of max and mean pooling methods

3.3 Rectified linear unit layer

Many activation functions, including y = tanh(x) and y = 1/(1+exp(-x)), have been proposed to adopt nonlinearity into neural network applications, but the ReLU has proven to be the best activation function for

CNN applications (Krizhevsky 2012; Nair and Hinton 2010); we also used ReLU in this study. Figure 4 shows the comparison of two activations functions and ReLU. Briefly, while the other activation functions have bounded output values of one, zero, or negative one, ReLU has no bounded outputs except in the case of inputs with negative values. Intuitively, the gradients of ReLU are always zero and one. These properties enable computations to be performed much faster and also result in better accuracy (Nair and Hinton 2010).

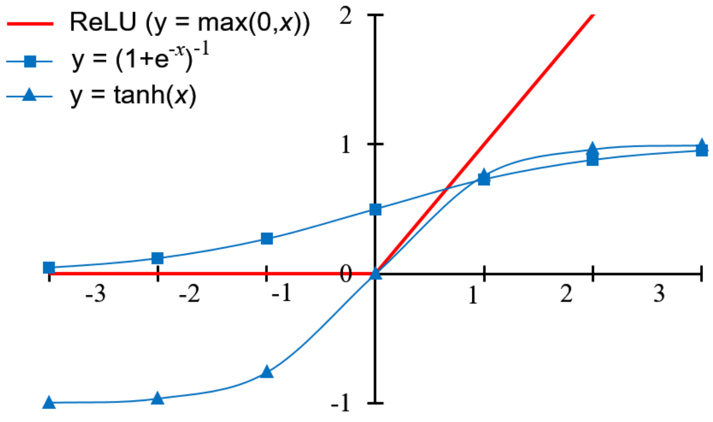


Figure 4: Activation functions

3.4 Batch normalization and dropout layer

One of the challenges of training CNNs is overfitting, a phenomenon in which a network can classify only the training dataset effectively. To overcome this challenge, dropout (Srivastava et al. 2014) and batch normalization (Ioffe and Szegedy 2015) layers are implemented. A dropout layer randomly disconnects neurons to reduce co-adaptations. Consequently, the network can generalize training examples more efficiently. A batch normalization layer reduces covariate shifts of the input. It enables use of significantly higher learning rates compared to a network without batch normalization, which shortens training time.

3.5 Softmax layer

In order to classify input data, a layer that predicts classes is necessary. In this study, the softmax function, given by Equation [1], was used with *i* examples, *n* classes, and *W* weights (i.e., receptive fields and filters). This function returns probabilities for each class by accepting input data, where the size of the input data is typically reduced by passing through a designed CNN architecture. When input data are fed into a softmax layer, the probabilities of each class are predicted by the function; the one with the highest probability is predicted to be in class *n*.

[1] $$P\left(y^{(i)}=n \mid x^{(i)};W\right)=\begin{bmatrix} p(y^{(i)}=1 \mid x^{(i)};W) \\ p(y^{(i)}=2 \mid x^{(i)};W) \\ \vdots \\ p(y^{(i)}=n \mid x^{(i)};W) \end{bmatrix}=\frac{1}{\sum_{j=1}^{n} e^{W_j^{\mathrm{T}} x^{(i)}}}\begin{bmatrix} e^{W_1^{\mathrm{T}} x^{(i)}} \\ e^{W_2^{\mathrm{T}} x^{(i)}} \\ \vdots \\ e^{W_n^{\mathrm{T}} x^{(i)}} \end{bmatrix} \text{ for } i=1\cdots m$$

4 COST FUNCTION AND BACKPROPAGATION USING SGD

In the beginning stage of training, the values of the receptive fields are given by random numbers; this results in a large number of errors, due to the deviations from the true answers of the training examples. The deviations were measured by Equation [2], where log likelihood was adopted for mathematical convenience.

[2] $$L = \frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{n} 1\left\{y^{(i)} = j\right\}\log\frac{e^{W_j^{\mathrm{T}}x^{(i)}}}{\sum_{l=1}^{n} e^{W_l^{\mathrm{T}}x^{(i)}}}\right] + \frac{\lambda}{2}\sum_{j=1}^{n} W_j^2$$

The term 1{$y^{(i)} = j$} is a logical expression that returns ones if the *i*th class is true for the *j*th class; otherwise, zeros. The last term of the hyperparameter λ in the equation is a regularization (i.e., weight decay) parameter that penalizes large weights; this is a well-known trick for preventing overfitting (Bengio 2012). The measured deviations can be narrowed by updating the values of the receptive fields, and stochastic gradient descent was used to do so in this study. The updating rule can be summarized by Equations [3] – [5], and the network repeats this process until the cost function converges to the global minimum.

[3] $$\nabla_W L\left(W; x^{(i)}, y^{(i)}\right) = \frac{1}{m}\sum_{i=1}^{m}\left[x^{(i)}\left\{1\left(y^{(i)} = j\right) - p\left(y^{(i)} = j \mid x^{(i)}; W\right)\right\}\right] + \lambda W_j$$

[4] $$\upsilon \leftarrow \varepsilon\upsilon - \alpha\nabla_{W_j} L(W; x^{(i)}, \mathrm{y}^{(i)})$$

[5] $$W_j \leftarrow W_j + \upsilon$$

5 TRAINING AND TESTING RESULTS

The designed network (Figure 1) was trained on 40,000 images of 256×256-pixel images showing intact and cracked concrete surfaces. The images were taken under extensively varied conditions (e.g., strong spotlight, shadow cast, etc.) to train a robust classifier. The ratio of the training sets to the validation sets was 4:1, and the validation existed in order to observe the training process over the epochs. Figure 5 shows the summarized training results. The validation error fluctuated until the 48th epoch, suggesting that the network needed more iterations. The trained network at the 51st epoch was considered to be the trained classifier for detecting cracks.

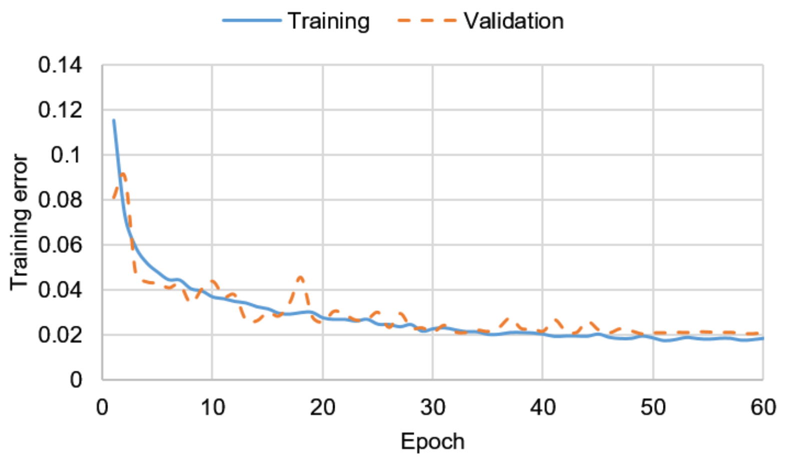
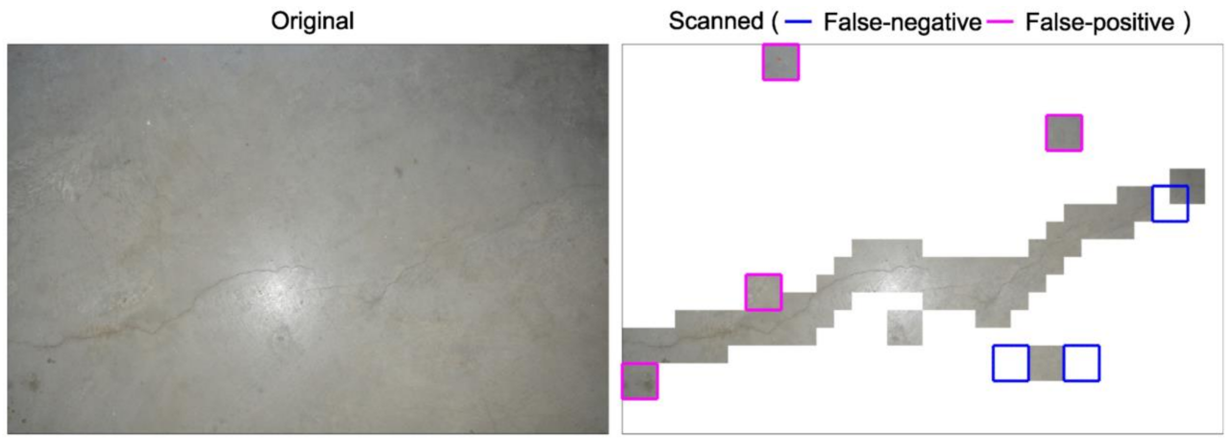
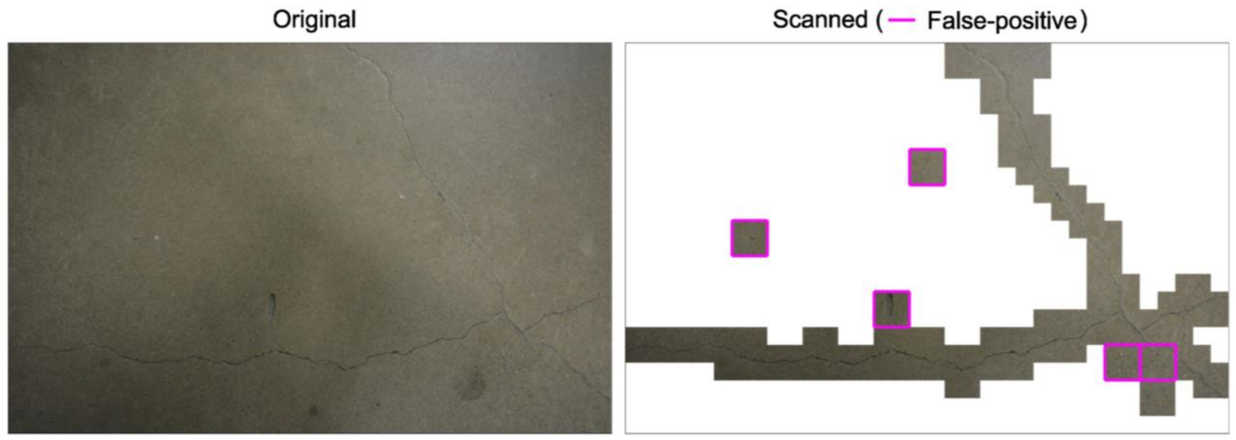


Figure 5: Summary of training



(a) Image (4352×2816 pixels) with thin cracks and a strong spotlight



(b) Shadowed image (4352×2816 pixels)

Figure 6: Scanned results of large images

The trained network was designed to accept 256×256-pixel images, which implies that the trained model cannot evaluate images that are larger or smaller than the designed pixel resolutions. Therefore, a sliding window technique was implemented as a post-processing tool. As a result, the proposed method can scan images of any size and localize cracks. To evaluate how the trained model generalizes crack features, 50 large images (greater than 4000×2500 pixels) were tested; the proposed methods achieved an average accuracy of 97% (Cha et al. 2017b) and recorded less than three seconds to scan those images. Two of the images are shown in Figure 6; (a) is an image with thin cracks and a strong spotlight, and (b) is a shadowed image.

6 CONCLUSION

A vision-based method for detecting concrete cracks from photographic images was developed. This method employed a CNN, which is the most robust image-classifying application known to date. To train the classifier, 40,000 images of 256×256-pixel resolution, captured under highly varied conditions, were fed into the designed CNN. The results of tests with large images demonstrated that this approach is much better for solving real-world problems than the conventional vision-based approaches that implement gradient-based image processing techniques. In the future, an advanced system capable of detecting at least five types of superficial damage—including voids, delamination, spalling, and corrosion of concrete and steel structures—will be proposed. This system will also be mounted on a drone and used for remote field inspections, as a partial substitute for traditional means of inspection.

REFERENCES

Abdel-Qader, Ikhlas, Osama Abudayyeh, and Michael E Kelly. 2003. Analysis of Edge-Detection Techniques for Crack Identification in Bridges. *Journal of Computing in Civil Engineering*, American Society of Civil Engineers, **17** (4): 255–63.

Bengio, Yoshua. 2012. Practical Recommendations for Gradient-Based Training of Deep Architectures in *Neural Networks: Tricks of the Trade.* 2nd ed., Springer, Berlin Heidelberg.

Cha, Young-Jin, Justin G Chen, and Oral Büyüköztürk. 2017. Output-Only Computer Vision Based Damage Detection Using Phase-Based Optical Flow and Unscented Kalman Filters. *Engineering Structures* **132:** 300–313.

Cha, Young-Jin, Wooram Choi, and Oral Buyukozturk. 2017. Deep Learning-Based Crack Damage Detection Using Convolutional Neural Network. *Computer-Aided Civil and Infrastructure Engineering* **32**(3): Published.

Cha, Young-Jin, Kisung You, and Wooram Choi. 2016. Vision-Based Detection of Loosened Bolts Using the Hough Transform and Support Vector Machines. *Automation in Construction*, **71**: 181–188.

Chen, Justin G, Neal Wadhwa, Young-Jin Cha, Frédo Durand, William T Freeman, and Oral Buyukozturk. 2015. Modal Identification of Simple Structures with High-Speed Video Using Motion Magnification. *Journal of Sound and Vibration* **345**: 58–71.

Cornwell, P, Charles R Farrar, Scott W Doebling, and Hoon Sohn. 1999. Environmental Variability of Modal Properties. *Experimental Techniques*, **23**(6): 45–48.

Ioffe, Sergey, and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv Preprint arXiv:1502.03167*.

Jahanshahi, Mohammad R., Sami F. Masri, Curtis W. Padgett, and Gaurav S. Sukhatme. 2013. An Innovative Methodology for Detection and Quantification of Cracks through Incorporation of Depth Perception. *Machine Vision and Applications* **24**(2): 227–241.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*: 1097–1105.

Kurata, M, J Kim, J P Lynch, G W Van der Linden, H Sedarat, E Thometz, P Hipley, and L-H Sheng. 2012. Internet-Enabled Wireless Structural Monitoring Systems: Development and Permanent Deployment at the New Carquinez Suspension Bridge. *Journal of Structural Engineering*, American Society of Civil Engineers, **139**(10): 1688–1702.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep Learning. *Nature*, Nature Research, **521**(7553): 436–44.

Lee, Young-Shin, and Myung-Jee Chung. 2000. A Study on Crack Detection Using Eigenfrequency Test Data. *Computers & Structures* **77**(3): 327–342.

Nair, Vinod, and Geoffrey E Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, Haifa, Israel: 807–814.

Ramana, Lovedeep, Wooram Choi, and Young-Jin Cha. 2017. Automated Vision-Based Loosened Bolt Detection Using the Cascade Detector. *IMAC-XXXV*, Orange County, CA, USA. Published.

Ruotolo, R, and Cecilia Surace. 1997. Damage Assessment of Multiple Cracked Beams: Numerical Results and Experimental Validation. *Journal of Sound and Vibration* **206**(4): 567–88.

Scherer, Dominik, Andreas Müller, and Sven Behnke. 2010. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. In *Proceedings* of *Artificial Neural Networks*, Springer, Verlag, Berlin, Heidelberg: 92–101.

Srivastava, Nitish, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning* Research 15(1): 1929–58.

Vedaldi, Andrea, and Karel Lenc. 2015. Matconvnet: Convolutional Neural Networks for Matlab. In Proceedings of the 23rd ACM International Conference on Multimedia, Brisbane, Australia: 689–692.

Xia, Yong, Bo Chen, Shun Weng, Yi-Qing Ni, and You-Lin Xu. 2012. Temperature Effect on Vibration Properties of Civil Structures: A Literature Review and Case Studies. Journal of Civil Structural Health Monitoring **2**(1): 29–46.

Ziou, Djemel, and Salvatore Tabbone. 1998. Edge detection techniques-an overview. Pattern Recognition and Image Analysis C/C of Raspoznavaniye Obrazov I Analiz Izobrazhenii **8**: 537-559.