Chapter 33
Structural Damage Detection Using Convolutional Neural Networks

Nur Sila Gulgec, Martin Takáč, and Shamim N. Pakzad

Abstract Detection of the deficiencies affecting the performance of the structures has been studied over the past few decades. However, with the long-term data collection from dense sensor arrays, accurate damage diagnosis has become computationally challenging task. To address such problem, this paper introduces convolutional neural network (CNN), which has led to breakthrough results in computer vision, to the damage detection challenge. CNN technique has the ability to discover abstract features which are able to discriminate various aspect of interest. In our case, these features are used to classify "damaged" and "healthy" samples modeled through the finite element simulations. CNN is performed by using a Python library called Theano with the graphics processing unit (GPU) to achieve higher performance of these data-intensive calculations. The accuracy and sensitivity of the proposed technique are assessed with a cracked steel gusset connection model with multiplicative noise. During the training procedure, strain distributions generated from different crack and loading scenarios are adopted. Completely unseen damage setups are introduced to the simulations while testing. Based on the findings of the proposed study, high accuracy, robustness and computational efficiency are succeeded for the damage diagnosis.

Keywords Structural health monitoring • Damage detection • Sensitivity analysis • Convolutional neural networks • Deep neural network

33.1 Introduction

The main difficulty of the structural damage detection is defining the unknown relation between the measurements and damage patterns. For this reason, damage diagnosis has been a challenging inverse problem in structural health monitoring for last few decades. In the 1990s, neural networks were proposed as a remedy for such poorly defined problems [1], and since then, they have been practiced to diagnose damages from the measurement data or its features without estimating the structure characteristics [2]. Literature has some studies which feed the neural network by using the inputs such as modal analysis of vibration response [3, 4], statistical parameters of vibration data [5], frequency response functions (FRFs) [6], wavelet transform coefficients of the acceleration data [2], and static displacements [7, 8]. In some studies, uncertainties in measurements are employed to improve the network architecture [9, 10]. Nevertheless, as mentioned in [11], in order to use the full potential of neural networks, researchers need to have more complex architectures. This complexity brings computational difficulty with it, when long-term data collection from dense sensor arrays are considered [12–14].

Proposed study addresses all these issues by developing a convolutional neural network (CNN) for damage identification. First of all, instead of creating hand-designed features like in traditional methods, our method allows the network to learn its damage features from the raw input. In other words, our study is capable of learning sophisticated damage features and create complex classifier boundaries. Secondly, by training a variety of loading cases, damage scenarios, and measurement noise levels, the proposed architecture accomplishes detecting damages even from the unseen damages with high accuracy. Lastly, the computational needs of the methodology are decreased by exploiting CNN's shared parameterization and GPU's massively parallel architecture.

N.S. Gulgec (✉) • S.N. Pakzad
Department of Civil and Environmental Engineering, Lehigh University, Imbt Labs, 117 ATLSS Drive, Bethlehem, PA 18015, USA
e-mail: nsg214@lehigh.edu

M. Takáč
Department of Industrial and Systems Engineering, Lehigh University, Harold S. Mohler Laboratory, 200 West Packer Avenue, Bethlehem, PA 18015, USA

R. Barthorpe et al. (eds.), *Model Validation and Uncertainty Quantification, Volume 3*,
Conference Proceedings of the Society for Experimental Mechanics Series, DOI 10.1007/978-3-319-54858-6_33

The rest of the paper is organized as follows. First, a brief explanation of CNNs is provided in Sect. 33.2; then, the data preparation is described in Sect. 33.3. In Sect. 33.4, description of proposed CNN architecture and the main findings are presented. Conclusions and future directions are given in Sect. 33.5.

33.2 Deep Neural Networks

Machine learning (ML) is a scientific field that is gradually developed from pattern recognition [15] (recognition of patterns and regularities in data) and learning theory in artificial intelligence. Deep learning (deep neural network or DNN) is a branch of machine learning which attempts to build a model using a deep graph organized in multiple linear layers and non-linear transformations [16]. In Fig. 33.1, a small DNN which consists of an input layer, two hidden layers, and an output layer is shown. The input layer feeds the input instance $x = (x_1, \ldots, x_p)^T$ to the network, and the output is the result of the network. In Fig. 33.1, each circle represents one neuron. An arrow represents a connection from the output of one neuron to the input of another. Each arrow is associated with a weight. The output of the neuron in a hidden layer is composed of a weighted sum of inputs composed with a non-linear mapping, e.g., sigmoid, tanh, or others.

For given input $x \in \mathbb{R}^p$, ML algorithms try to build a prediction function $\theta(x; w)$ parametrized by w (frequently called weights). In the simplest case this function is considered to be linear, i.e., $\theta(x; w) = x^T w$. After the family of the prediction function is fixed, a loss function, which measures the error between a prediction and the true value, is chosen. The most elementary loss function is $\ell(\theta(x; w), y) = \|\theta(x; w) - y\|^2$, where $y \in \mathbb{R}^c$ is the true observed value (sometimes called the label) of the input query x. Other commonly used loss functions are the so-called soft-max and cross entropy functions. It is assumed that there is a distribution of inputs and corresponding labels (X, Y). The learning problem then boils down to finding the best possible instance of the prediction function from the selected family, which amounts to finding the best possible values of the weights w. Mathematically speaking, one solves the following optimization problem [17–20]:

$$\min_w \mathbb{E}_{(X,Y)}[\ell(\theta(x; w), y)], \tag{33.1}$$

where the expectation is taken over the true distribution of inputs and labels (X, Y). Unfortunately, in practice, it is impossible to have exact knowledge about the true distribution. The common practice is to sample n data points $\{(x_i, y_i)\}_{i=1}^n$ (frequently called training data) from the unknown distribution, and minimize the empirical loss instead:

$$\min_w \tfrac{1}{n} \textstyle\sum_{i=1}^n \ell(\theta(x_i; w), y_i). \tag{33.2}$$

Convolutional Neural Networks Convolutional neural networks (CNN) are one of the most widely used types of deep neural networks and has been a breakthrough in visual and speech recognition for the last few years. The framework of CNN was first proposed by LeCun et al. [21] to classify handwritten digits. It did not work well with the large-scale image and video classification due to the lack of computing power and large training data at that time [22]. With the introduction of a highly parallel programmable unit called graphics processing unit (GPU), large scale visual recognition has become more pronounced. With the development in computing power and large-scale hierarchical image database [23], CNN architectures

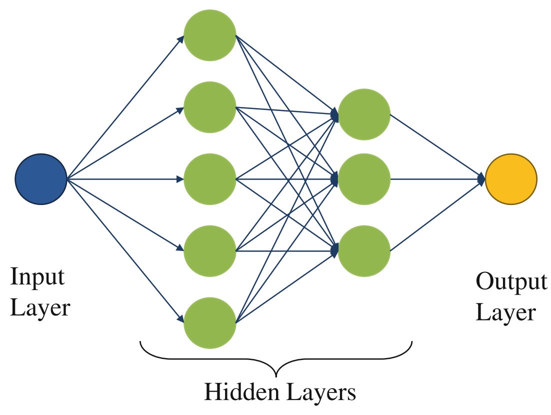


Fig. 33.1 A DNN with two hidden layers

kept evolving [24–26]. The performance improved drastically as the networks became more complex and deeper [27, 28]. The reason behind such success was the ability to keep temporal features of the input and using fewer parameters to reduce memory requirements [29].

Convolutional neural networks have three architectural frameworks such as local receptive fields, shared weights, and spatial sub-sampling [21]. Local receptive fields allow extracting multiple feature maps by sliding the same set of units all over the input. This property makes CNN robust to the translation and distortion of the input (i.e. the feature map will be shifted by the same amount of input shifting). Moreover, these feature maps use shared weights and biases, which reduce the learned parameters as well as the memory needs. Lastly, spatial sub-sampling reduces the resolution of the feature maps to avoid sensitivity of the outputs under shifts and rotations.

CNN architecture consists of three main types of layers:

1. **Convolutional layer** parameters are learnable weights and biases which are shared in the depth of the input. Feature maps are formed as these weights (or namely filters) are slid through the entire input, and the dot product between the filter and input are computed. This operation is called "convolution". Then, nonlinear activation functions activate the feature maps.
2. **Pooling layer** performs a downsampling operation (using e.g. maximum, average, sum operations) in the feature maps which reduce the dimensionality.
3. **Fully-connected layers** get the stacked convolutional layer outputs and compute the weighted sum of inputs composed with a non-linear mapping as described in Sect. 33.2.

33.3 Data Preparation

In order to have accurate damage diagnosis, the networks should be trained with well-known damage states [30]. Therefore, an analytic model of a steel gusset plate connection is simulated in ABAQUS by using shell elements. As shown in Fig. 33.2, the structural connection consists of two C8×11.5 channels and a steel plate (28 × 14 × 1/4 in.) welded together. Each connecting member is 20 in. long and has 8-in. overlap with the main plate. The model has the mesh size of 0.5 in. The behavior of the material is defined as elastic-perfectly plastic which has a yield strength of 36 ksi. Uniformly distributed load changing from 120 kips tensile to 100 kips compression is applied to the end of the channel members. Strain field in the direction of loading of the gusset plate is used to analyze the proposed methodology.

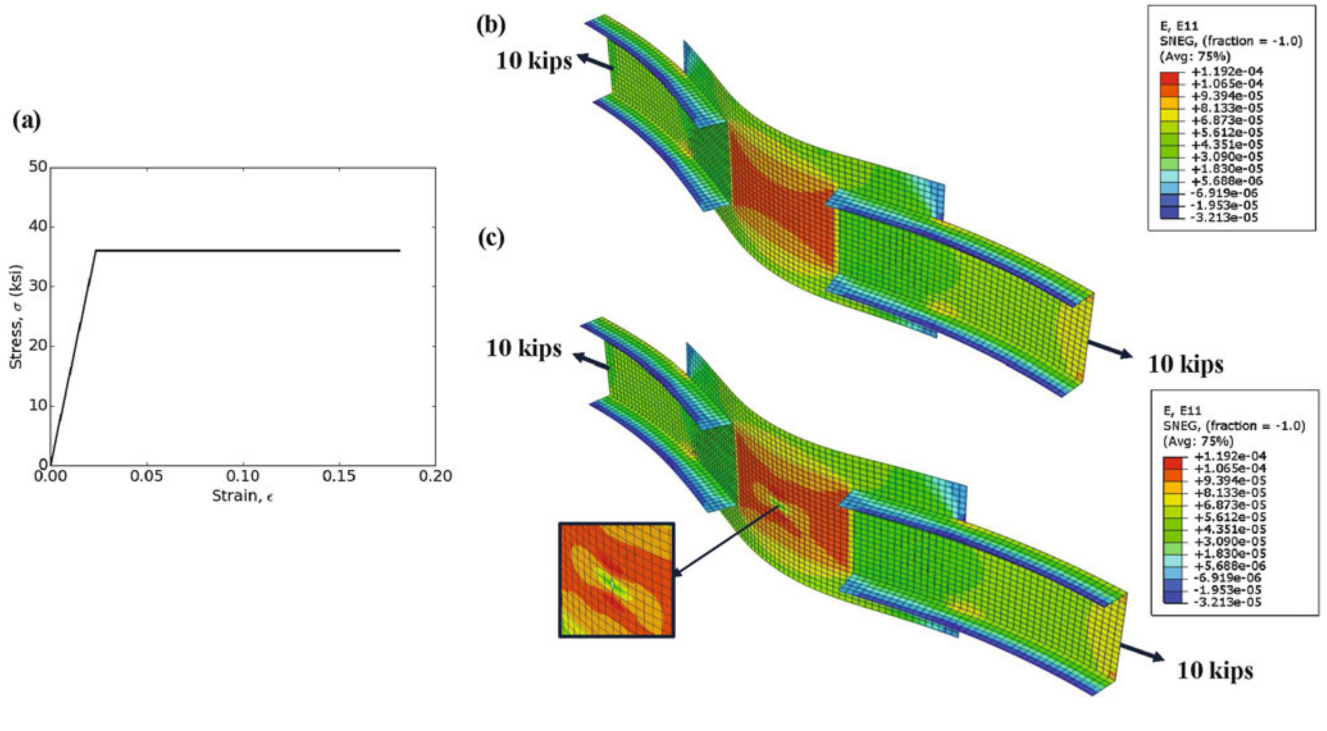


Fig. 33.2 (**a**) Material behavior; the setup of the (**b**) healthy and (**c**) single damaged gusset-plate

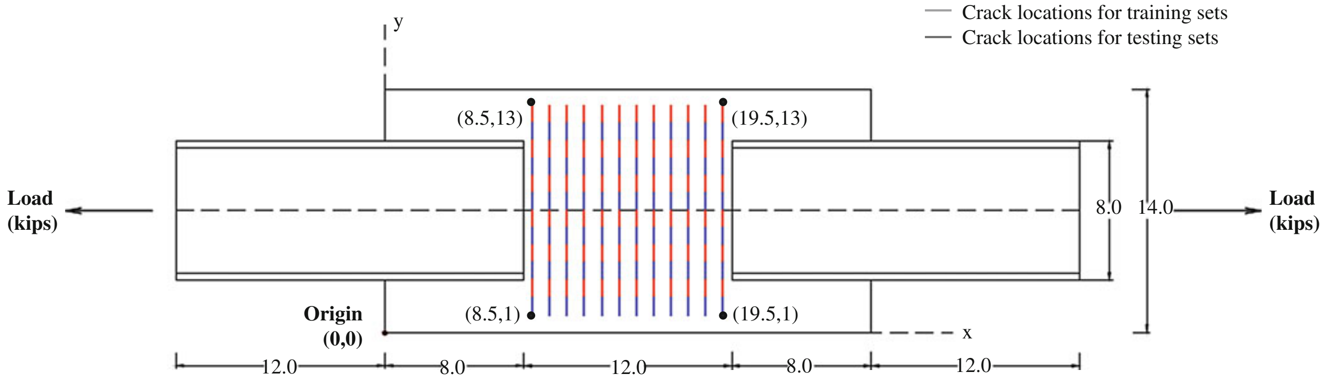


Fig. 33.3 The damage locations for training and test data sets

Variability in training, validation, and test data sets are formed by exploiting different loading cases, damage scenarios and noise levels. Damages are simulated as 1-in. long cracks and the crack locations are chosen randomly at the beginning of the analysis with a specified load in the range of $U[-100, 200]$ kips. The coordinates of the cracks changing between (8.5, 1) and (19.5, 13) are shown in Fig. 33.3. None of the coordinates of training samples are used in the testing samples to be able to evaluate the methodology with completely unseen damaged samples. Furthermore, inevitable uncertainty in the measurement and modeling processes is simulated as multiplicative noise in the samples. Total 24,000 "healthy" and 24,000 "single-damaged" unique samples are generated and distributed to the training, validation, and test data sets. The strain field of the gusset plate is represented as $28 \times 56 \times 1$ tensors as inputs to the network. These inputs are normalized by subtracting the data mean and dividing the data maximum.

33.4 Proposed Architecture for Damage Detection

33.4.1 CNN Architecture

The convolutional neural network can be built in various ways by using the sequence of convolutional layers (CONV), pooling layers (POOL) and fully connected layers (FC). Therefore, hyperparameter search mechanism should be performed to find the structure of optimal CNN architecture. In this study, a total of 50 networks are constructed with different hyperparameters such as learning rate, the number of convolutional layers, the number of FC layers, the size of hidden layers, filter and kernel sizes. The first run is performed for few epochs, then, 10% of the networks which have the worst test score are removed. The remaining networks are run for another set of epochs. The runs are computed until the best 10 networks remain. The variability in the hyperparameters are listed as: filter size: [(3x3), (5x5)], max-pooling size: (2x2), learning rate: $[2-2^{-8}]$, kernel size: $[2, 4, 8, 16, 32]$, the number of convolutional layer: $[1-3]$, the number of fully connected layer: $[1-3]$ and randomly selected hidden layer sizes.

The best network obtained as a result of the search mechanism is shown in Fig. 33.4. The network classifies 28×56 inputs of normalized strain distribution as "undamaged" and "damaged". The proposed architecture has two convolutional networks with kernels of $[16, 16]$ and filter sizes of [(5x5), (5x5)]. The feature maps of the last pooling layer, which are stacked together in an array, feed the fully connected layers with two hidden layers $[1388, 1212]$. Nonlinear function tanh() is used for the activation of the layers.

For a given input, prediction function can be parametrized by w (namely weights). In this study, the label of the final output is found by softmax classifier. The class i of the input x is predicted by choosing the maximum probability of softmax function defined as follows:

$$[\text{softmax}(\theta(x;w))]_i = \frac{e^{[\theta(x;w)]_i}}{\sum_j e^{[\theta(x;w)]_j}}, \qquad y_{prediction} = \underset{i}{\text{argmax}}([\text{softmax}(\theta(x;w))]_i). \tag{33.3}$$

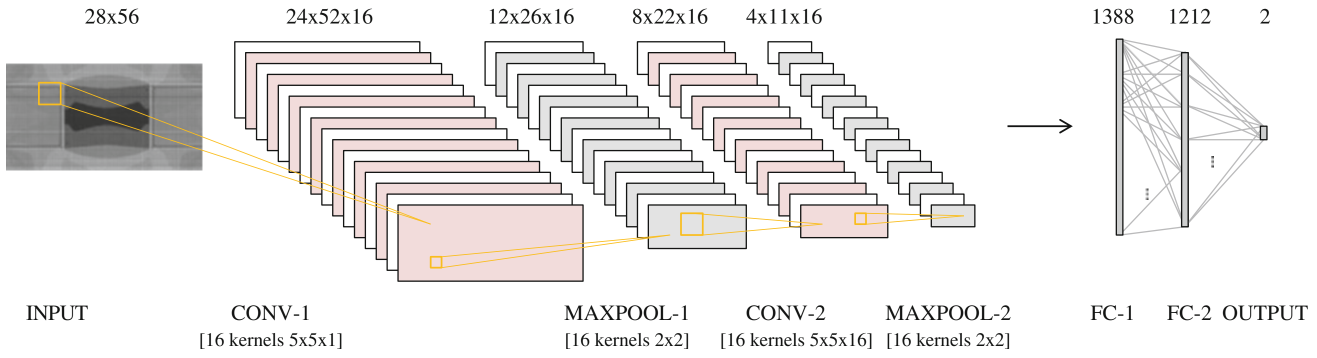


Fig. 33.4 Proposed CNN architecture

Table 33.1 Sensitivity analysis of the CNN network

	Noiseless	1% noise	2% noise
Testing error %	2.06	2.56	3.51
False negative ratio %	2.06	2.56	3.34
False positive ratio %	0.00	0.00	0.17

33.4.2 *Training*

Proposed study is trained by using a Python library called Theano to optimize the mathematical expressions including multi-dimensional arrays [31]. Higher performance of these data-intensive calculations is accomplished by NVIDIA Tesla K80 GPUs which enable parallelism for data processing. Furthermore, in order to reduce variations between the layers, weights are initialized by Xavier initialization for tanh function [32]. Weight initialization of ith layer is set to be in the uniform distribution in the interval $\left[-\sqrt{\frac{6}{n_{i-1}+n_i}}, \sqrt{\frac{6}{n_{i-1}+n_i}}\right]$ where n_{i-1} and n_i are the are the number of units in the $(i-1)$th and ith layer.

The practical advantage of choosing CNN as a model is that it can be trained efficiently using a stochastic gradient descent algorithm [33], where the gradients are computed by back-propagation [34]. The proposed model adopts the negative log-likelihood as the loss function of back-propagation process, where optimal model parameters θ^* are learned by maximizing the likelihood of the data set. In this study, stochastic gradient descent method is implemented to a batch size of 512 samples. Once the gradients are computed with stochastic gradient descent, the model is updated with the learning rate, $\eta = 0.033$. In order to prevent overfitting, validation set performance is monitored in every epoch. When model's performance is improved sufficiently on the validation set, no further optimization is implemented.

33.4.3 *Results*

Trained network is tested on previously unseen damage states which are not used for training procedure. The findings of the proposed CNN architecture is presented in this section. As shown in Table 33.1, this study identifies the unseen damages with 2.06% error. This error rate represents that the CNNs are capable of learning the damage features if enough training cases are provided. Furthermore, the introduction of different noise levels during the training helps network to learn damage features under uncertainty. As presented in Table 33.1, test error doesn't change drastically under the 1% and 2% multiplicative noise introduced to the samples. Therefore, our methodology is robust under the slight noise. The validation and test error sensitivity of the samples that are without noise and with 1% and 2% noise are also shown in Fig. 33.5.

As presented in Table 33.1, the majority of the misclassified observations belong to the damage status that is detected as "healthy". In SHM, incorrect classification of damaged samples (namely false negative) is not desirable; thus, the damage locations of misclassifications are further investigated. The gusset plate area shown in Fig. 33.3 is converted into a matrix having the number of false negatives in each damage location for test data set (Fig. 33.6). For example, the element in the first column of the first row represents the misclassified crack from (8.5, 11) to (8.5, 12); the element in the first column of the second row represents the misclassified crack from (8.5, 10) to (8.5, 11), and so on. The number of false negatives is also color coded as (green: < 5, yellow: 5–10, red: >10) to provide better visualization. As can be seen from Fig. 33.6, most

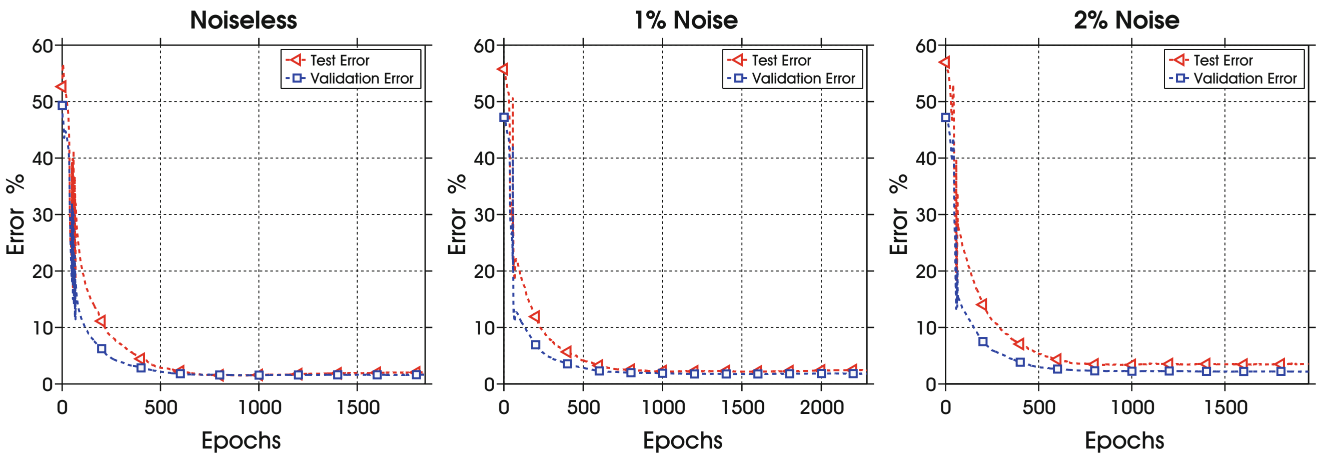
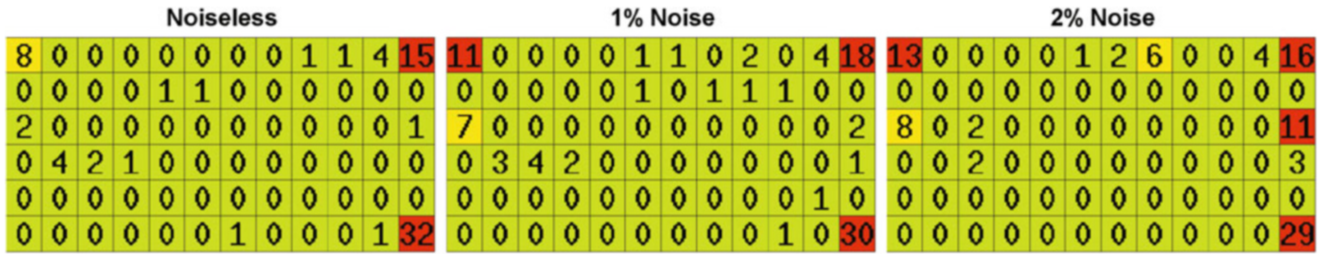


Fig. 33.5 Error sensitivity of the test samples with different noise levels


Noiseless
1% Noise
2% Noise

Noiseless

8	0	0	0	0	0	0	0	1	1	4	15
0	0	0	0	1	1	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	1
0	4	2	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	0	0	0	1	32

1% Noise

11	0	0	0	0	1	1	0	2	0	4	18
0	0	0	0	0	1	0	1	1	1	0	0
7	0	0	0	0	0	0	0	0	0	0	2
0	3	4	2	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	1	0	30

2% Noise

13	0	0	0	0	1	2	6	0	0	4	16
0	0	0	0	0	0	0	0	0	0	0	0
8	0	2	0	0	0	0	0	0	0	0	11
0	0	2	0	0	0	0	0	0	0	0	3
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	29

Fig. 33.6 The false negatives per each simulated crack location for test data set with 0% noise (*left*), 1% noise (*middle*) and 2% noise (*right*)

of the misclassifications occur in the regions where the stress concentration is lower compared to the center of the plate. Therefore, false negative samples do not fall in high-risk region where damages can propagate failure of the structure.

33.5 Conclusion

The major challenge of the structural damage detection is finding the unknown relation between the measurements and damage patterns. To address such problem, this paper introduces convolutional neural network (CNN) which has the ability to discover abstract features that are able to discriminate various aspect of interest. In our study, these feature maps are used to classify "damaged" and "healthy" cases modeled through the analytic simulations. The computational needs of the methodology are decreased by exploiting CNN's shared parameterization and GPU's massively parallel architecture.

Based on the results of the proposed study, high accuracy, robustness, and computational efficiency are succeeded for the damage diagnosis challenge. However, further research is necessary to be able to benefit more from convolutional neural networks. Future work aims to (1) determine the location and severity of the damages, in addition to the detection of damage existence, (2) reduce the number of false negatives by penalizing it, (3) test the network on the real experimental setup.

Acknowledgements Research funding is partially provided by the National Science Foundation through Grant No. CMMI-1351537 by Hazard Mitigation and Structural Engineering program, and by a grant from the Commonwealth of Pennsylvania, Department of Community and Economic Development, through the Pennsylvania Infrastructure Technology Alliance (PITA). Martin Takáč was supported by National Science Foundation grant CCF-1618717.

References

1. Flood, I., Kartam, N.: Neural networks in civil engineering. II: systems and application. J. Comput. Civ. Eng. **8**(2), 149–162 (1994)
2. Shi, A., Yu, X.-H.: Structural damage detection using artificial neural networks and wavelet transform. In: 2012 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications (CIMSA) Proceedings, pp. 7–11. IEEE, New York (2012)
3. Hearn, G., Testa, R.B.: Modal analysis for damage detection in structures. J. Struct. Eng. **117**(10), 3042–3063 (1991)
4. Hadzima-Nyarko, M., Nyarko, E.K., Morić, D.: A neural network based modelling and sensitivity analysis of damage ratio coefficient. Expert Syst. Appl. **38**(10), 13405–13413 (2011)
5. Shu, J., Zhang, Z., Gonzalez, I., Karoumi, R.: The application of a damage detection method using artificial neural network and train-induced vibrations on a simplified railway bridge model. Eng. Struct. **52**, 408–421 (2013)
6. Fang, X., Luo, H., Tang, J.: Structural damage detection using neural network with learning rate improvement. Comput. Struct. **83**(25), 2150–2161 (2005)
7. Szewczyk, Z.P., Hajela, P.: Damage detection in structures based on feature-sensitive neural networks. J. Comput. Civ. Eng. **8**(2), 163–178 (1994)
8. Zhao, J., Ivan, J.N., DeWolf, J.T.: Structural damage detection using artificial neural networks. J. Infrastruct. Syst. **4**(3), 93–101 (1998)
9. Bakhary, N., Hao, H., Deeks, A.J.: Damage detection using artificial neural network with consideration of uncertainties. Eng. Struct. **29**(11), 2806–2815 (2007)
10. Nazarko, P., Ziemiański, L.: Application of artificial neural networks in the damage identification of structural elements. Comput. Assist. Mech. Eng. Sci. **18**(3), 175–189 (2011)
11. Flood, I.: Towards the next generation of artificial neural networks for civil engineering. Adv. Eng. Inform. **22**(1), 4–14 (2008)
12. Yao, R., Pakzad, S.N., Venkitasubramaniam, P.: Compressive sensing based structural damage detection and localization using theoretical and metaheuristic statistics. Struct. Control Health Monit. **24**, e1881 (2017). doi:10.1002/stc.1881
13. Yao, R., Pakzad, S.N., Venkitasubramaniam, P., Hudson, J.M.: Iterative spatial compressive sensing strategy for structural damage diagnosis as a big data problem. In: Dynamics of Civil Structures, vol. 2, pp. 185–190. Springer, New York (2015)
14. Shahidi, S.G., Gulgec, N.S., Pakzad, S.N.: Compressive sensing strategies for multiple damage detection and localization. In: Dynamics of Civil Structures, vol. 2, pp. 17–22. Springer, New York (2016)
15. Bishop, C.M.: Pattern recognition. Mach. Learn. **128**, 1–58 (2006)
16. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436–444 (2015)
17. Shalev-Shwartz, S., Ben-David, S. Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press, New York (2014)
18. Nilsson, N.J.: Introduction to machine learning. An Early Draft of a Proposed Textbook (1996)
19. Smola, A., Vishwanathan, S.V.N.: Introduction to Machine Learning, pp. 32–34. Cambridge University, New York (2008)
20. Alpaydin, E.: Introduction to Machine Learning. MIT, Cambridge (2014)
21. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
22. Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G.: Recent advances in convolutional neural networks. arXiv preprint, arXiv:1512.07108 (2015)
23. Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L.: Imagenet: a large-scale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition, 2009 (CVPR 2009), pp. 248–255. IEEE, New York (2009)
24. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Proceedings of the 25th International Conference on Neural Information Processing Systems. Curran Associates Inc. (2012)
25. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint, arXiv:1409.1556 (2014)
26. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: European Conference on Computer Vision, pp. 818–833. Springer, Berlin (2014)
27. 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, pp. 1–9 (2015)
28. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. arXiv preprint, arXiv:1512.03385 (2015)
29. LeCun, Y., Bengio, Y.: Convolutional Networks for Images, Speech, and Time-Series. MIT Press, Cambridge (1995)
30. Elkordy, M.F., Chang, K.C., Lee, G.C.: Neural networks trained by analytically simulated damage states. J. Comput. Civ. Eng. **7**(2), 130–145 (1993)
31. Theano Development Team. Theano: a Python framework for fast computation of mathematical expressions. arXiv e-prints, abs/1605.02688, May (2016)
32. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Aistats, vol. 9, pp. 249–256 (2010)
33. Robbins, H., Monro, S.: A stochastic approximation method. Ann. Math. Stat. **22**(3), 400–407 (1951)
34. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Cogn. Model. **5**(3), 1 (1988)