Front. Struct. Civ. Eng. 2022, 16(11): 1365–1377
https://doi.org/10.1007/s11709-022-0882-5

RESEARCH ARTICLE

Bridging finite element and deep learning: High-resolution stress distribution prediction in structural components

Hamed BOLANDI[a, b*], Xuyang LI[a, b], Talal SALEM[a], Vishnu Naresh BODDETI[b], Nizar LAJNEF[a]

[a] *Department of Civil and Environmental Engineering, Michigan State University, East Lansing, MI 48824, USA*
[b] *Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824, USA*
[*] *Corresponding author. E-mail: bolandih@msu.edu*

ABSTRACT Finite-element analysis (FEA) for structures has been broadly used to conduct stress analysis of various civil and mechanical engineering structures. Conventional methods, such as FEA, provide high fidelity results but require the solution of large linear systems that can be computationally intensive. Instead, Deep Learning (DL) techniques can generate results significantly faster than conventional run-time analysis. This can prove extremely valuable in real-time structural assessment applications. Our proposed method uses deep neural networks in the form of convolutional neural networks (CNN) to bypass the FEA and predict high-resolution stress distributions on loaded steel plates with variable loading and boundary conditions. The CNN was designed and trained to use the geometry, boundary conditions, and load as input to predict the stress contours. The proposed technique's performance was compared to finite-element simulations using a partial differential equation (PDE) solver. The trained DL model can predict the stress distributions with a mean absolute error of 0.9% and an absolute peak error of 0.46% for the von Mises stress distribution. This study shows the feasibility and potential of using DL techniques to bypass FEA for stress analysis applications.

KEYWORDS Deep Learning, finite element analysis, stress contours, structural components

1 Introduction

Stress analysis is an essential part of engineering and design. The development of various design systems continuously imposes higher demands on computational costs while preserving accuracy. Numerical analysis methods, such as structural finite element analysis (FEA), are typically used to conduct stress analysis of various structures. Researchers commonly use FEA methods to evaluate the design, safety, and maintenance of different structures in various fields, including aerospace, automotive, architecture, and civil, structural systems.

The current workflow for FEA applications includes: a) modeling geometry and its components, which can be time-consuming based on the system complexity; b) specifying material properties, boundary conditions, and loading; c) applying a meshing strategy for geometry. The time-consuming and complexity of current FEA workflows make it impractical in real-time or near real-time applications, such as in the aftermath of a disaster or during extreme disruptive events that require immediate corrections to avoid catastrophic failures.

Based on the steps of FEA described above, performing a complete stress analysis with conventional FEM has a high computational cost. To resolve this issue, we propose a Deep Learning (DL) method [1,2] to construct deep neural networks (DNN), which, once trained, allow bypass of FEA. This method may enable real-time stress analysis by leveraging machine learning (ML) algorithms. DNNs can model complicated, nonlinear relationships between input and output data. Thus, these models help us acquire adequate knowledge for predictions of unseen problems.

Data-driven approaches that model physical phenomena have been lauded for their significant and growing successes. Most recent works have included design and

Article history: Received Mar 15, 2022; Accepted Jul 1, 2022

topology optimization [3–6], data-driven approaches in fluid dynamics [7–10], molecular dynamics simulation [11–14], and material properties prediction [15–18]. Also, Atalla et al. and Levin et al. [19,20] have used neural regression for FEA model updating. Recently, DL has shown promise in solving conventional mechanics problems. Some researchers used DL for structural damage detection, a promising alternative to conventional structural health monitoring methods [21–24]. Javadi et al. [25] used a typical neural network in FEA as a surrogate for the traditional constitutive material model. They simplified the geometry into a feature vector which approaches cases that are hard to generalize or more complicated. The numerical quadrature of the element stiffness matrix in the FEA on a per-element basis was optimized by Oishi and Yagawa [26] using DL. Their approach helps to accelerate the calculation of the element stiffness matrix.

A convolutional neural network (CNN) is a type of neural network which has shown great performance in several applications related to Computer Vision and Image Processing. The significant learning ability of CNN is mainly due to several feature extraction stages that can intrinsically learn representations from the feeding data. Madani et al. [27] developed a CNN architecture for stress prediction of arterial walls in atherosclerosis. Also, Liang et al. [28] proposed a CNN model for aortic wall stress prediction. Their method is expected to allow real-time stress analysis of human organs for a wide range of clinical applications.

In this work, we tackle the limitations of stress analysis using FEA. We propose an end-to-end DL method to predict the stress distribution in 2D linear elastic steel plates. The algorithm takes geometry, boundary conditions, and load as input and renders the von Mises stress distribution as an output. We model steel gusset plates with loading applied at different edges, different boundary conditions, and varying complex geometries. A dataset initialized with 104448 samples with varying geometry, boundary conditions, and loads is used to train and evaluate the network.

2 Background on deep learning and convolutional neural network

Artificial intelligence (AI) developed into ML over time from pattern recognition and learning theory [1]. Samuel [29] defined ML as a "field of study that allows computers to learn without being explicitly programmed". ML algorithms can learn from data, and during the learning process, they build models which will be used to make decisions or data-driven predictions. DL is a subfield of ML which focuses on modeling hierarchical representations or abstractions to define higher-level concepts. The DL community is making significant advances in solving problems that AI has struggled to solve for many years [1]. Data of high dimensions have proven to be highly useful in discovering complex structures. Thus, DL is practical for many domains such as government and business, specifically computer vision and image recognition. These methods have shown significant performance in image classification [30], natural sentence classification [31], and image segmentation [32].

DL techniques can extract features; however, we should be careful in choosing the appropriate technique to use when dealing with a specific task. Within these approaches, CNNs have been demonstrated to be particularly efficient at acquiring a representation of the input data, including grid type data such as matrices or images. LeCun et al. [33] proposed the initial skeleton of CNN to classify handwritten digits. Over the last few years, massive hierarchical image databases, GPU programmable units, and highly parallel computing have significantly improved CNN. CNN architectures have developed greatly since the earliest work [34–36], and the performance has improved remarkably as the networks have become more complicated and deeper layers [37].

CNNs use four concepts to enhance their performance: local connections, weight sharing, pooling, and multiple layers. CNNs are composed of a series of steps. The first step involves a convolutional layer, with units in this layer organized in feature maps. Local patches in the feature maps of the previous layer are connected to each unit by a set of weights known as a filter bank. In the second step, the output of this locally weighted sum is then passed through a nonlinearity, such as a ReLU or other activation functions. The weights are then passed on to the pooling layer. Pooling layers are used to assemble semantically similar features into one. Finally, a series of convolutional, nonlinear, and pooling stages are stacked, followed by even more convolutional and fully-connected layers. A CNN uses backpropagating gradients similar to a typical deep network, allowing all the filter banks to be trained simultaneously [37].

3 Deep learning in civil and mechanical engineering

Artificial neural networks with multilayer perceptron (MLPs) have been used in civil and mechanical engineering for many years. Researchers use ANN for structural analysis [38–40], regression of material constitutive properties [41,42], and materials' failure and damage [43,44]. Gulgec et al. [23] proposed a CNN architecture to classify simulated damaged and intact samples and localize the damage in steel gusset plates. Modarres et al. [45] studied composite materials to identify the presence and type of structural damage using CNNs. Also, for detecting concrete cracks without calculating the defect features, Zang et al. [46] proposed a vision-based method based on CNNs. An approach for predicting stress distribution on all layers of non-uniform 3D parts was presented by Khadilkar et al. [47]. More recently, Nie et al. [48] developed a CNN-based approach to predict the stress field in a 2D linear cantilever beam. However, the above works used DL techniques for structural analysis. Guo et al. [49] studied the bending analysis of Kirchhoff plates of various shapes, loads, and boundary conditions. Anitescu et al. [50] presented an artificial neural network-based collocation method for solving boundary value problems. Samaniego et al. [51] studied DNN as the basis of a technique for approximating data, and such Networks have shown promising results in areas such as visual recognition. Zhuang et al. [52] proposed a deep autoencoder-based energy method (DAEM) for the bending, vibration, and buckling analysis of Kirchhoff plates. Guo et al. [53] presented a modified neural architecture search method based on physics-informed DL for stochastic analysis of heterogeneous porous materials. Guo et al. [54] proposed a deep collocation method (DCM) based on transfer learning for solving potential problems in non-homogeneous media. To our knowledge, the current work is the first 'DL-FE substitution' approach to perform a fast and accurate prediction of high-resolution stress distributions in 2D steel plates.

4 Method

4.1 Data generation

Two-dimensional steel plate structures with five edges, E1 to E5 denoting edges 1 to 5, as shown in Fig. 1, are considered to be made of homogeneous and isotropic linear elastic material. The 2D steel plates have similar geometry to that of gusset plates, as used for connecting beams and columns to braces in steel structures. The boundary conditions and loading angles simulate conditions that are similar to those affecting common gusset plate structures under external loading. Analysis of the behavior of these components is essential since various reports have observed failures of gusset plates subject to lateral loads [55–58]. The distributed static loads applied to the plates in this study range from 1 to 5 kN with intervals of 1 kN. Moreover, loads are applied with three angles, including π/6, π/4, and π/3, on either one or two edges of the plate. The load is decomposed to its horizontal and vertical direction components. Also, four types of boundary conditions are considered, as shown in Fig. 2, based on real gusset plates' boundary conditions. All the translational and rotational displacements are fixed at the boundary conditions. All input variables used to initialize the population are shown in Table 1. The minimum and maximum range for the width and height of the plate are from 30 to 60 cm. Various

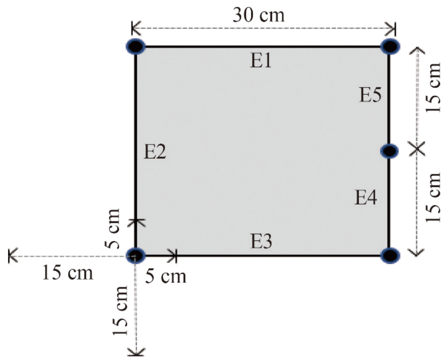

Fig. 1 Basic schematic topology for initializing the steel plate geometries.

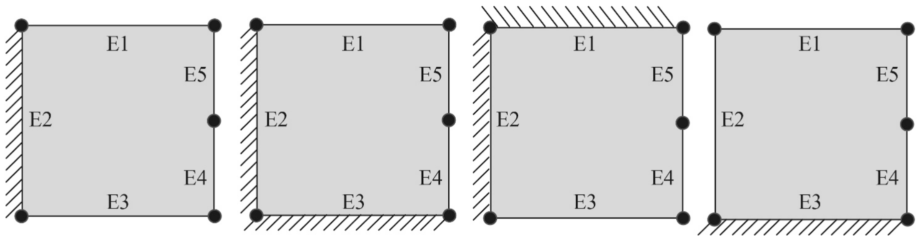

Fig. 2 Different types of boundary conditions for initializing population.

Table 1 Input variables

geometry	boundary conditions	load position	load angle (°)	load magnitude (kN)
pentagon	E2	E5, E4, E4 E5	30, 45, 60	1, 2, 3, 4, 5
pentagon	E2 E3	E5	30, 45, 60	1, 2, 3, 4, 5
pentagon	E1 E2	E4	30, 45, 60	1, 2, 3, 4, 5
pentagon	E3	E2 E5, E1 E2 E5	30, 45, 60, 90	1, 2, 3, 4

geometries are generated by changing the position of each node in horizontal and vertical directions, as shown in Fig. 1, which lead to 1024 unique pentagons. The material properties remain unchanged and isotropic for all samples.

4.1.1 Input data

The geometry is encoded into a 600 × 600 matrix as a single channel binary image. 0 (black) and 1 (white) denotes the outside and inside of the geometry, as shown in Fig. 3 (a). The boundary conditions are also represented by another 600 × 600-pixel binary images, where the constrained edges are defined by 1 (white) (Fig. 3(b)). Moreover, each horizontal and vertical component of the load is encoded as one 600 × 600-pixel single-channel colored image, as shown in Figs. 3(c) and 3(d). The magnitudes of the decomposed horizontal and vertical components of the loads vary between 0.5 and 4.33 kN. These loads are normalized between (100,0,0) and (255,0,0) as RGB colors to create a color image where the colored part represents the location and magnitude of the load (Figs. 3(c) and 3(d)).

4.1.2 Output data

FEA is performed using the Partial Differential Equation (PDE) solver in the MATLAB toolbox to obtain the stress distributions of each sample. The MATLAB PDE toolbox mesh generator only generates unstructured triangulated meshes incompatible with CNN. Since each element should be represented by one pixel in an image, we develop a 600 × 600 grid surface equal to the dimensions of the most significant possible geometry. The stress values are then interpolated between the triangular elements and grids to determine a stress distribution compatible with our CNN network.

The stress values of all the elements outside the material geometry are assigned a zero, as shown in Fig. 3(e). The dimensions of the largest sample are 600 mm × 600 mm, and the smallest are 300 mm × 300 mm. Therefore, the size of each element is 1 mm × 1 mm, which means that each image has 360000 pixels. This high-resolution dataset offers significant accuracy. The maximum and minimum von Mises stress values for elements among the entire dataset are 96366 and –0.73 MPa, respectively. We normalized all the output data between 0 and 1 to ensure faster convergence and encoded it to 600 × 600 matrices.

5 Convolutional neural network architecture

The CNN can be built using a sequence of convolutional layers. The convolutional layers learn to encode the input in simple signals and reconstruct the input [59]. Our CNN architecture consists of three stages of layers: The first stage is downsampling which consist of seven convolutional layers (E1, E2, E3, E4, E5, E6, E7), and the second stage has three layers (RS1, RS2, and RS3) of Squeeze-Excitation and Residual blocks (SE-ResNet). In addition, the Inception and MobileNetV2 blocks are swapped with the SE-ResNet block to check if these modules can further enhance the network's performance. The third stage is upsampling, consisting of six deconvolutional layers (D1, D2, D3, D4, D5, D6), as illustrated in Fig. 4.

5.1 Residual block

We use residual blocks [37] to address the vanishing gradient problem. In addition, residual blocks are computationally lightweight and result in only very small increases in model complexity. The shortcut connection

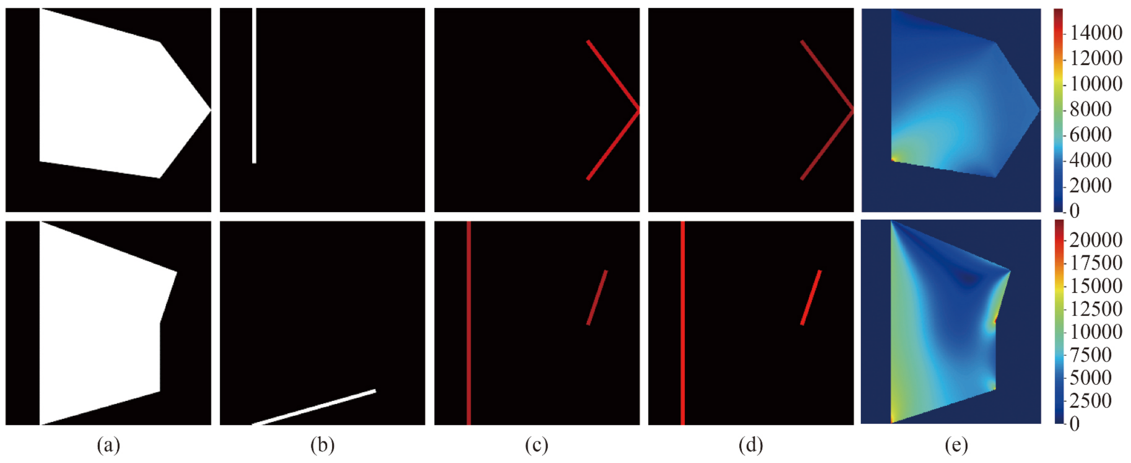

Fig. 3 Input and output representation for stress distribution prediction: (a) geometry, (b) boundary condition, (c) horizontal load, (d) vertical load, (e) output.

in residual bocks simply performs identity mapping, and its output is added to the output of the stacked layers.

5.2 Squeeze-and-Excitation block

Squeeze-and-Excitation (SE) blocks improve the representative capacity of the network, enabling dynamic channel-wise feature recalibration. An SE block can be implemented with five phases. 1) The number of channels and the input convolutional block are given to the algorithm. 2) Using average pooling, the function reduces each channel to a single numeric value. 3) A fully connected layer is used to add nonlinearity followed by a ReLU function and the output channel complexity is also will be reduced. 4) Sigmoid activation followed by a second fully connected layer provides smooth gating for each channel. 5) Finally, the function, weight feature maps of each convolutional block based on the network results. Figure 5 depicts an SE-ResNet module which the SE block transformation. F_{tr} is regarded as the non-identity branch of a residual module. Before summation of the identity branch, both SE act. Using both SE and ResNet in the network outperforms using ResNet [60].

5.3 Inception block

Inception Modules are used to reduce the computational cost of CNNs. Since neural networks have to deal with a vast array of images, each with different content, they must be carefully designed. Using the vanilla version of the inception module, we can perform a convolution on the input meaning three different sizes of filters (1 × 1, 3 × 3, 5 × 5) instead of one. Also, max pooling is performed. The outputs are then concatenated and sent to the next layer. Therefore, convolutions occur at the same level in CNNs, where the network gets wider, not deeper. Compared with shallower and less wide CNNs, this method offers significant quality gains at a modest computational cost increase [36]. Figure 6 depicts the inception module.

5.4 MobileNetV2 block

MobileNetV2 is based on an inverted residual block with shortcut connections between thin bottleneck layers [61]. A lightweight depth-wise convolution technique is used in the intermediate layer to filter features as a source of nonlinearity. The nonlinearities must be removed in the narrow layers to maintain representational power. In

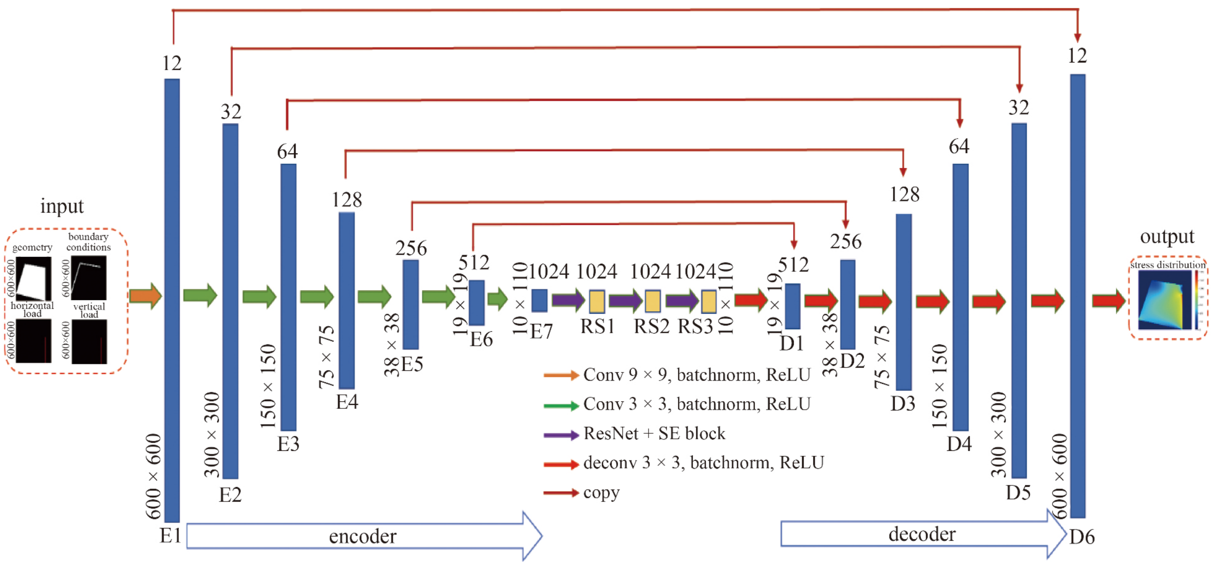

Fig. 4 Proposed CNN architecture.

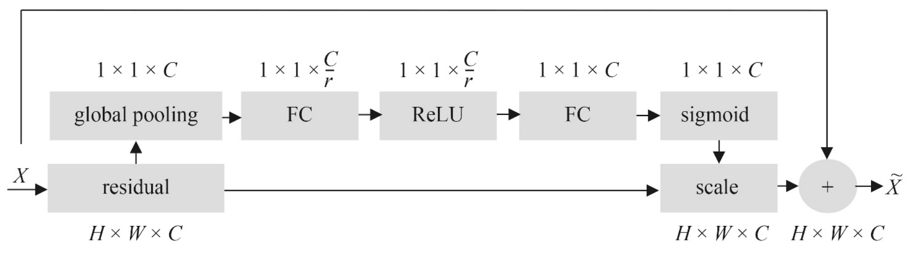

Fig. 5 SE-ResNet module.

general, in this model, the bottlenecks encode the intermediate inputs and outputs of the model, while the inner layer encodes how the model can transform from lower-level concepts such as pixels to higher-level features such as categories of images. Lastly, shortcuts can improve training speed and accuracy, just like traditional residual connections. Figure 7 depicts the MobileNetV2 module.

5.5 Network layers and hyperparameters

All the details of the network layers and hyperparameters can be found in Tables 2 and 3. As can be seen, the models consist of 7 Conv layers, 3 different bottleneck blocks, and 6 ConvT layers. Of various combinations of Conv and ConvT layers with maximum channels of 512, 1024, 2048, and 4096, the model with 1024 channels shows the best performance. Therefore, we keep the network with 1024 channels as the primary model and swap the bottleneck each time with the SE-ResNet, Inception, and MobileNetV2. We keep the bottleneck dimension the same for all models to match the ConvT first layer. The batch size is set to 16, leading to the best accuracy compared to other batch sizes. Different learning rates from 1e−3 to 1e−6, and 1e−5 lead to the best convergence.

6 Loss function and performance metrics

We used *MSE* (mean squared error) for the training loss defined in Eq. (1). *MSE* gives a more significant penalty to large errors than *MRE* (mean relative error). Also, the errors are normally distributed. Using *MAE* (mean absolute error), *MRE*, *PMAE* (percentage mean absolute error), *PAE* (peak absolute error), and *PPAE* (percentage peak absolute error) helps evaluate the overall quality of predicted stress distribution. These metrics are defined in Eqs. (2)–(5), respectively.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(s(i) - \hat{s}(i))^2, \tag{1}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|s(i) - \hat{s}(i)|, \tag{2}$$

where $s(i)$ is the stress value at a node i computed by FEA as the ground truth and, $\hat{s}(i)$ is the corresponding predicted stress by the DL model. Also, n is the total number of elements in each sample which is 360000 in our work. Symbol | | denotes the absolute value. Our model's prediction and ground truth are displayed as 600 × 600 resolution images.

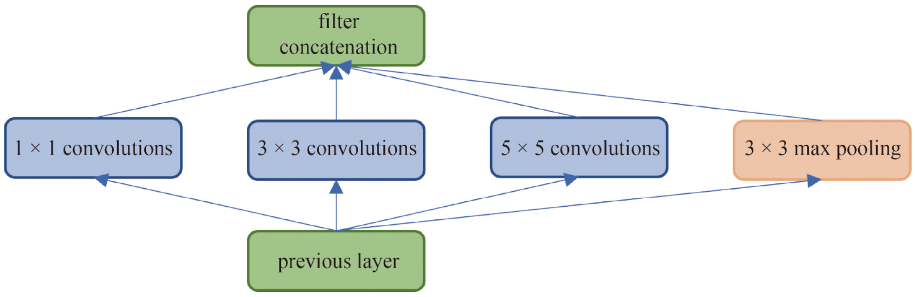

Fig. 6 Inception module.

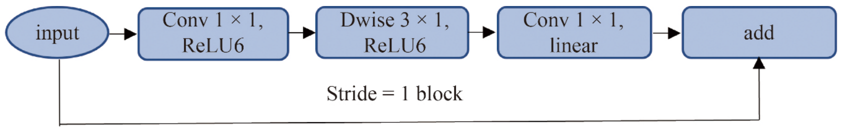

Fig. 7 MobileNetV2 module.

Table 2 Network layers

item	number of layers	first layer ($H \times W \times C$)	last layer ($H \times W \times C$)	activarion
Conv	7	600 × 600 × 12	10 × 10 × 1024	ReLU
SE-ResNet	3	10 × 10 × 1024	10 × 10 × 1024	Sigmoid-ReLU
Inception	3	10 × 10 × 1024	10 × 10 × 1024	ReLU
EffiientNet	1	10 × 10 × 1024	10 × 10 × 1024	ReLU6
ConvT	6	19 × 19 × 512	600 × 600 × 12	ReLU

Table 3 Network hyperparameters

batch_size	learning rate	weight deacy	expand ratio	loss functions
16	1.00E−05	1.00E−07	6	MSE-MAE

For measurement of the accuracy of predictions by comparing them to the ground truth, we use *MRE*:

$$MRE = \frac{1}{n}\sum_{i=1}^{n}\frac{|s(i)-\hat{s}(i)|}{\in +\max(s(i),\hat{s}(i))}\times 100, \quad (3)$$

where the ∈ term is a small value to avoid a division by zero.

The percentage mean absolute error is defined as:

$$PMAE = \frac{MAE}{\max\{s(i)\}-\min\{s(i)\}}\times 100, \quad (4)$$

where $\max\{s(i)\}$ is the maximum value in a set of ground truth stress values, and $\min\{s(i)\}$ is the minimum value.

PAE and *PPAE* measure the accuracy of the most significant stress value in the predicted stress distribution. *PAE* and *PPAE* are defined as:

$$PAE = \max\{s(i)\}-\max\{\hat{s}(i)\}, \quad (5)$$

$$PPAE = \frac{PAE}{\max\{s(i)\}}\times 100. \quad (6)$$

7 Results and discussion

All codes are written in PyTorch Lightning and run on two NVIDIA TITAN RTX 24G GPUs. We use AdamW (Adam algorithm with weight decay) optimizer to speed up the convergence of models. We train and evaluate different models based on Table 3 to find the model with the best performance. The training data size of models 1 to 3 is 83558, and the testing data size is 20890, randomly divided with a train/test ratio of 80%–20%. Figure 8 shows *MSE* and *MAE* losses as a function of epochs in model 1. Figures 8(a) and 8(b) are linear and logarithmic scales. Figure 8(a) shows that the *MSE* and *MAE* curves rapidly decline after a few epochs. However, Fig. 8(b) gives a more precise representation of the model's behavior. Figure 8(b) shows that *MSE* is smaller than *MAE*, but both have similar general trends.

We save the best checkpoint during validation, and all error metrics are based on the best checkpoint. Models 4 to 6 are validated with K-fold cross-validation to ensure that the model is generalizable. To reduce the

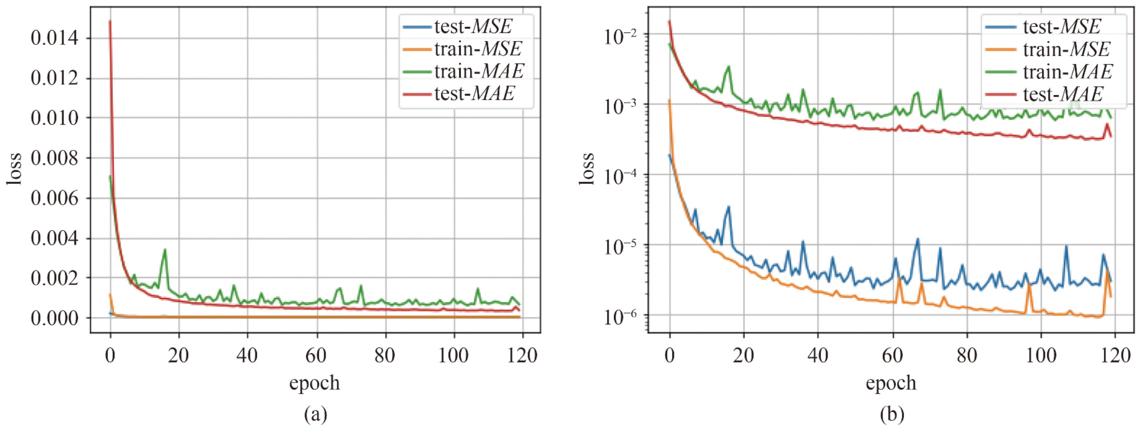

Fig. 8 *MSE* and *MAE* curves on training and testing data with two scales: (a) linear scale, (b) logarithmic scale.

Table 4 Error metrics for models at best checkpoints (units: MPa)

model	dataset	bottleneck	*MSE*	*MAE*	*MRE* (%)	*PMAE* (%)	*PPAE* (%)	*PAE*
model 1	classic	SE-ResNet	0.21	58.80	3.80	0.90	0.46	30.00
model 2	classic	Inception	0.62	31.55	1.54	0.57	2.66	150.55
model 3	classic	MobileNet	0.38	51.99	2.83	0.93	4.36	246.50
model 4	Fold 1	SE-ResNet	0.59	34.90	1.80	0.63	2.19	124.00
model 5	Fold 2	SE-ResNet	0.64	25.04	1.10	0.45	0.12	6.93
model 6	Fold 3	SE-ResNet	0.61	32.03	1.42	0.57	0.93	52.89
mean of K-fold cross-validation			0.61	30.65	1.44	0.55	1.08	61.27
STD of K-fold cross-validation			0.02	4.14	0.28	0.07	0.85	48.15

computational cost, we divide the dataset into three folds. K-fold cross-validation shows the best performance in all models based on most metrics, as can be seen in Table 4, which means the model is generalizable.

We replace the SE-ResNet block in the bottleneck with the Inception and MobileNetV2 block in models 2 and 3, respectively. Model 1, has the best performance in terms of *PPAE*, with an error of 0.46% and model 2 is the best model, based on *PMAE* with a 0.57% error. Figure 9 depicts the performance of different models in terms of *MAE*. As can be seen in Fig. 9, models 3 and 1, which have MobileNetworkV2 and SE-ResNets in a bottleneck, have almost the same performance, and model 2 with inception block is the best in terms of *MAE*s. We deem these results satisfactory for stress distribution predictions, specifically the *PPAE*, the most critical load value for stress distribution in engineering domain applications.

Figure 10 illustrates the cumulative distribution of *PMAE* and *PPAE* in the test dataset of model 1. Figure 10(a) shows the probability of mean in *PMAE* is 80%, which means that about 80% of predicted samples have a *PMAE* of less than 0.9, and 50% of samples have a *PMAE* of less than 0.46, which is the median. Figure 10(b) shows that about 99% of predicted samples have a *PPAE* of less than 0.46, and 50% of the predicted samples have a *PPAE* of 0.06.

The predictions produced from some randomly selected samples from the test dataset of model 1 are visualized in Fig. 11. Each row represents a sample. Columns (a) to (d) represent geometry, boundary conditions, and load in horizontal and vertical directions, respectively. Columns (e) and (f) represent the ground truth and predicted stress distributions. As can be seen, there is a high fidelity fit between ground truth and predicted stress distributions in both maximum stress and stress distribution in the different samples. Also, some inaccurate predictions are shown in Fig. 12. These predictions still provide useful information.

7.1 Effect of dataset size on the performance of the network

We break the data into different sizes to evaluate the effect of data size on the network's performance of model 1. Therefore, besides training with the entire dataset, 104448 samples, we train the network with 10000, 20000, 30000, 40000, 50000, and 70000 samples. Figure 13 demonstrates that training with just 10% of the dataset can achieve a mean error of 1.85%, which is acceptable in most engineering applications. Also, it can be seen that if we want to accomplish a mean error of less than 1%, we should train the network with at least 90% of the dataset.

We also evaluate the effect of data size on the Gaussian

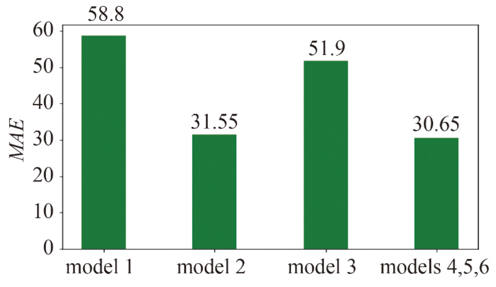

Fig. 9 Comparison of different models in terms of *MAE*.

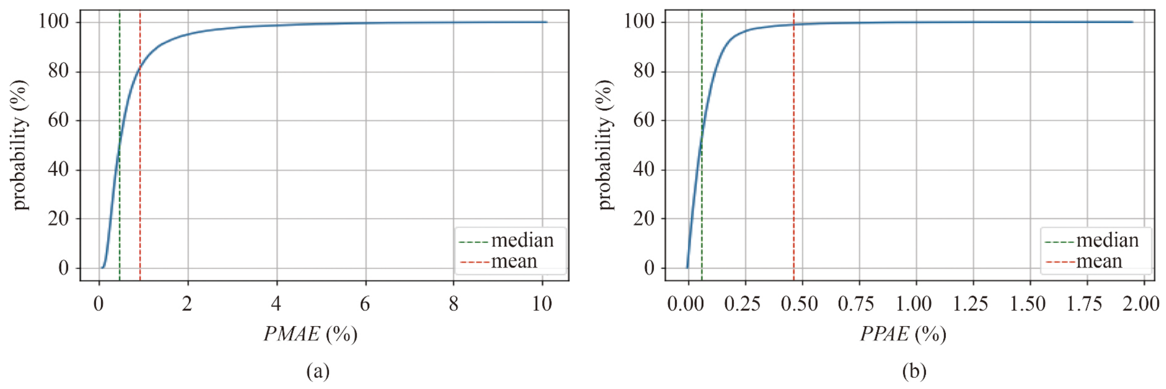

Fig. 10 Cumulative distribution of *PMAE* and *PPAE*: (a) *PMAE* of samples less than mean and median on the test dataset; (b) *PPAE* of samples less than mean and median on the test dataset.

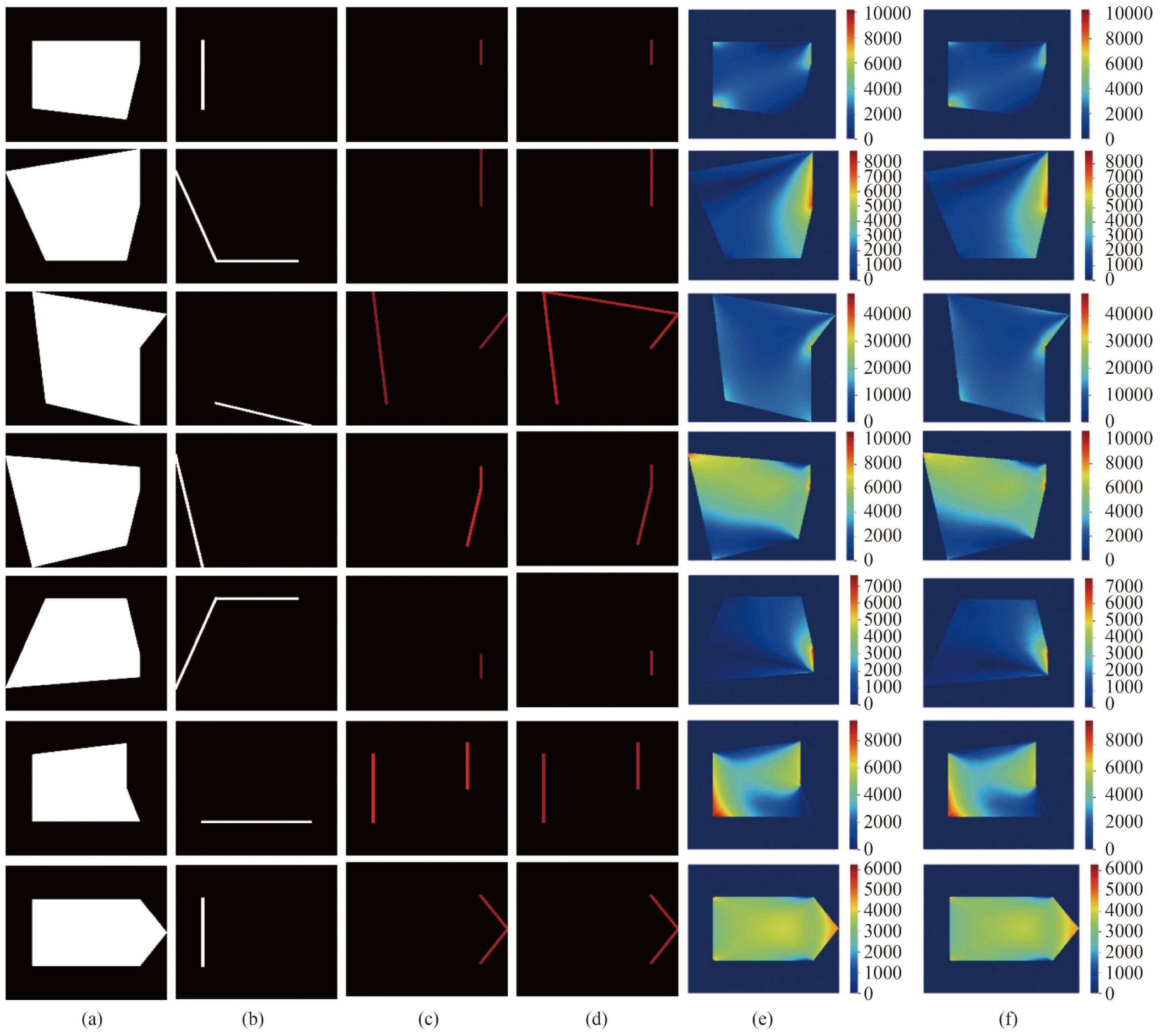

Fig. 11 Predicted stress distribution and corresponding inputs with different loads and boundary conditions scenarios: (a) geometry, (b) boundary conditions, and load in a (c) horizontal and (d) vertical direction; (e) ground truth and (f) predicted stress distribution, respectively (unit: MPa).

distributions of *PMAE* and *PPAE*, illustrated in Figs. 14(a) and 14(b). As shown in Fig. 14(a), increasing the data size decreases the standard deviation of *PMAE*. However, a 70000 data size and the total data size have almost the same standard deviation. Figure 14(b) shows that the standard deviation of *PPAE* decreases when the data size increases from 50000 to 70000. As a result, we should train the network with at least 70000 examples, 67% of our dataset, to ensure *PPAE*'s acceptable standard deviation accuracy.

8 Conclusions

In this work, we used end-to-end DL techniques. We developed a CNN to alleviate the need for finite element methods for prediction of high-resolution stress distributions in loaded steel plates. The CNN was designed and trained to use the geometry, boundary conditions, and load as input, and it provided high-resolution stress contours as the output. We used the PDE toolbox of MATLAB to generate the output data for training, containing 104448 FEM samples. We trained and evaluated different models to find the model with the best performance. The best model can predict the stress distributions with a mean absolute error of 0.9% and a maximum stress error of 0.46% in the von Mises stress distribution. The effects of dataset size on the model performance were also studied. Training the network with just 10% of the dataset achieved a mean error of 1.85%,

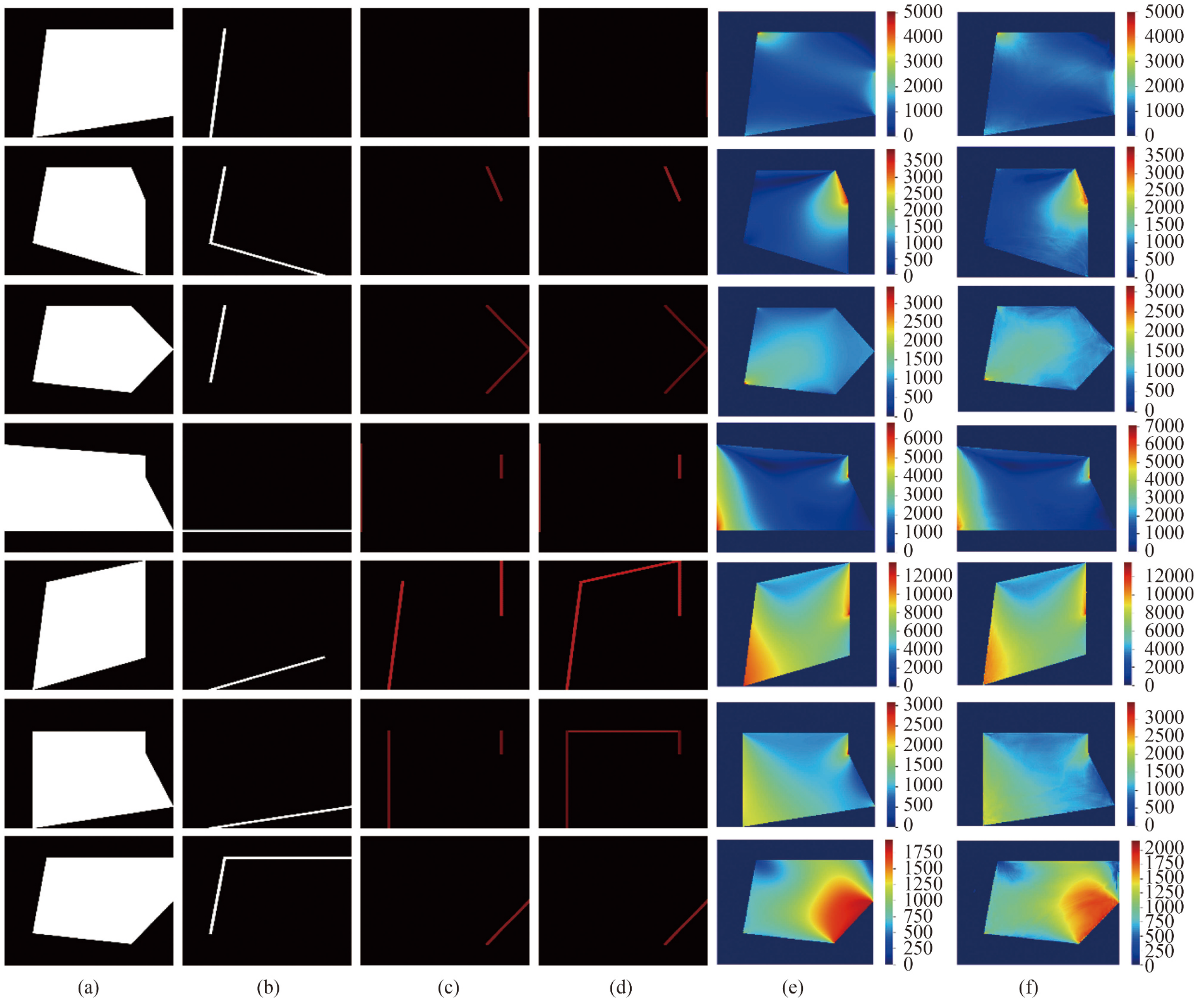

Fig. 12 Inaccurate predicted stress distribution and corresponding inputs with different loads and boundary conditions scenarios: (a) geometry, (b) boundary conditions, and load in a (c) horizontal and (d) vertical direction; (e) ground truth and (f) predicted stress distribution, respectively (unit: MPa) .

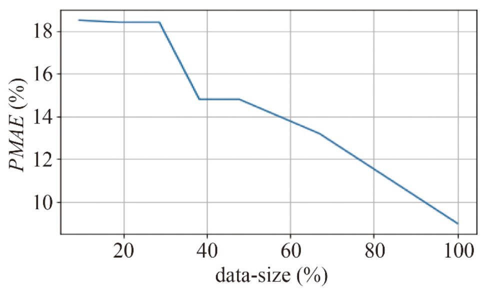

Fig. 13 PMAE at different data sizes.

which can be considered acceptable in specific engineering applications. Moreover, we evaluated the effect of dataset size on the Gaussian distribution of mean and maximum stress errors. Increasing the data size decreased the standard deviation of mean error. The standard deviation of maximum stress error also decreased with increase of the number of samples. Furthermore, the Gaussian distributions of mean and maximum stress errors demonstrated that a greater quantity of data induced less standard deviation in *PMAE* and *PPAE*.

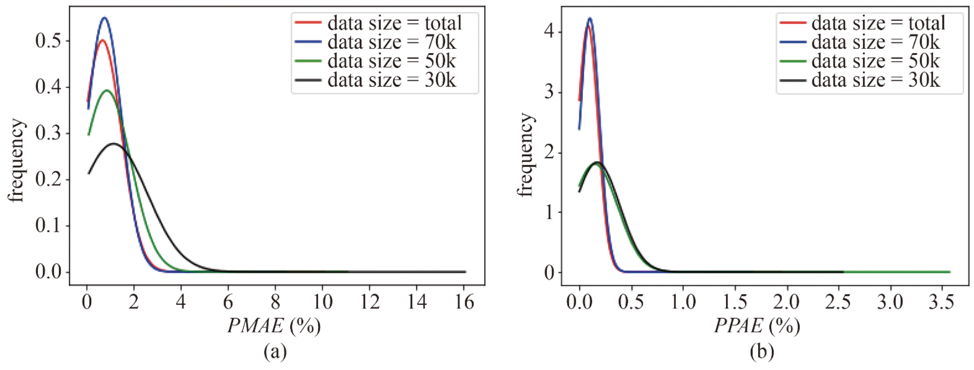

Fig. 14 Gaussian distribution of *PMAE* and *PPAE*. (a) *PMAE*, (b) *PPAE*.

Acknowledgements This research was funded in part by National Science Foundation (Grant No. CNS 1645783).

References

1. LeCun Y, Bengio Y, Hinton G E. Deep learning. Nature, 2015, 521(7553): 436−444
2. Schmidhuber J. Deep learning in neural networks: An overview. Neural Networks, 2015, 61: 85−117
3. Umetani N. Exploring generative 3D shapes using autoencoder networks. Pages, 2017, 24: 1−4
4. Yu Y, Hur T, Jung J. Deep learning for topology optimization design. 2018, arXiv:1801.05463
5. Zhang W, Jiang H, Yang Z, Yamakawa S, Shimada K, Kara L B. Data-driven upsampling of point clouds. Computer-Aided Design, 2019, 112: 1−13
6. Ulu E, Zhang R, Yumer M E, Kara L B. A data-driven investigation and estimation of optimal topologies under variable loading configurations. In: Computational Modeling of Objects Presented in Images. Fundamentals, Methods, and Applications. Pittsburgh: Springer International Publishing, 2014
7. Roy A G, Conjeti S, Karri S P K, Sheet D, Katouzian A, Wachinger C, Navab N. Relaynet: retinal layer and fluid segmentation of macular optical coherence tomography using fully convolutional networks. Biomedical Optics Express, 2017, 8(8): 3627−3642
8. Arvind T. Mohan and Datta V. Gaitonde. A Deep Learning-based approach to reduced order modeling for turbulent flow control using LSTM neural networks. 2018, arXiv:1804.09269
9. Farimani A B, Gomes J, Pande V S. Deep learning the physics of transport phenomena. 2017, arXiv:1709.02432
10. Kim B, Azevedo V C, Thuerey N, Kim T, Gross M, Solenthaler B. Deep fluids: A generative network for parameterized fluid simulations. Computer Graphics Forum. 2019, 38(2): 59–70
11. Goh G B, Hodas N O, Vishnu A. Deep learning for computational chemistry. Journal of Computational Chemistry, 2017, 38(16): 1291−1307
12. Mardt A, Pasquali L, Wu H, Noé F. VAMPnets for deep learning of molecular kinetics. Nature Communications, 2018, 9(1): 1−11
13. Montavon G, Rupp M, Gobre V, Vazquez-Mayagoitia A, Hansen K, Tkatchenko A, Müller K R, Anatole von Lilienfeld O. Machine learning of molecular electronic properties in chemical compound space. New Journal of Physics, 2013, 15(9): 095003
14. Tribello G A, Ceriotti M, Parrinello M. A self-learning algorithm for biased molecular dynamics. Proceedings of the National Academy of Sciences of the United States of America, 2010, 107(41): 17509−17514
15. Mohammadi Bayazidi A, Wang G G, Bolandi H, Alavi A H, Gandomi A H. Multigene genetic programming for estimation of elastic modulus of concrete. Mathematical Problems in Engineering, 2014, 474289
16. Sarveghadi M, Gandomi A H, Bolandi H, Alavi A H. Development of prediction models for shear strength of SFRCB using a machine learning approach. Neural Computing & Applications, 2019, 31(7): 2085−2094
17. Mousavi S M, Aminian P, Gandomi A H, Alavi A H, Bolandi H. A new predictive model for compressive strength of HPC using gene expression programming. Advances in Engineering Software, 2012, 45(1): 105−114
18. Bolandi H, Banzhaf W, Lajnef N, Barri K, Alavi A H. An Intelligent model for the prediction of bond strength of FRP bars in concrete: A soft computing approach. Technologies, 2019, 7(2): 42
19. Atalla M J, Inman D J. On model updating using neural networks. Mechanical Systems and Signal Processing, 1998, 12(1): 135−161
20. Levin R I, Lieven N A J. Dynamic finite element model updating using neural networks. Journal of Sound and Vibration, 1998, 210(5): 593−607
21. Fan Z, Wu Y, Lu J, Li W. Automatic pavement crack detection

based on structured prediction with the convolutional neural network. 2018, arXiv:1802.02208

22. Dung C V, Anh L D. Autonomous concrete crack detection using deep fully convolutional neural network. Automation in Construction, 2019, 99: 52−58
23. Gulgec N S, Takáč M, Pakzad S N. Convolutional neural network approach for robust structural damage detection and localization. Journal of Computing in Civil Engineering, 2019, 33(3): 04019005
24. Cha Y J, Choi W, Büyüköztürk O. Deep learning-based crack damage detection using convolutional neural networks. Computer-Aided Civil and Infrastructure Engineering, 2017, 32(5): 361−378
25. Javadi A A, Tan T P, Zhang M. Neural network for constitutive modelling in finite element analysis. Computer Assisted Mechanics and Engineering Sciences, 2003, 10(4): 523−530
26. Oishi A, Yagawa G. Computational mechanics enhanced by deep learning. Computer Methods in Applied Mechanics and Engineering, 2017, 327: 327−351
27. Madani A, Bakhaty A, Kim J, Mubarak Y, Mofrad M R. Bridging finite element and machine learning modeling: stress prediction of arterial walls in atherosclerosis. Journal of Biomechanical Engineering, 2019, 141(8): 084502
28. Liang L, Liu M, Martin C, Sun, W. A deep learning approach to estimate stress distribution: A fast and accurate surrogate of finite-element analysis. Journal of The Royal Society Interface, 2018, 15(138): 20170844
29. Samuel A L. Some studies in machine learning using the game of checkers. IBM Journal of Research and Development, 1959, 3(3): 210−229
30. Alpaydin E. Introduction to machine learning. Cambridge, MA: MIT Press, 2014
31. Kim Y. Convolutional neural networks for sentence classification. 2014, arXiv:1408.5882
32. Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems 25. Lake Tahoe: Curran Associates, 2012
33. LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998, 86(11): 2278−2324
34. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. 2014, arXiv:1409.1556
35. Zeiler M D, Fergus R. Visualizing and understanding convolutional networks. In: European Conference on Computer Vision. New York: Springer, 2014
36. 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. New York: IEEE, 2015
37. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Boston: IEEE, 2015
38. Jenkins W M. Neural network-based approximations for structural analysis. In: Developments in Neural Networks and Evolutionary Computing for Civil and Structural Engineering. Cambridge: Civil-Comp Press Edinburgh, 1995
39. Waszczyszyn Z, Ziemiański L. Neural networks in mechanics of structures and materials—New results and prospects of applications. Computers & Structures, 2001, 79(22−25): 2261−2276
40. Goh A T C, Wong K S, Broms B B. Multivariate modelling of FEM data using neural networks. Computers & Structures, 2001, 79(22−25): 2261−2276
41. Huber N, Tsakmakis Ch. Determination of constitutive properties from spherical indentation data using neural networks. Part II: Plasticity with nonlinear isotropic and kinematic hardening. Journal of the Mechanics and Physics of Solids, 1999, 47(7): 1589−1607
42. Settgast C, Abendroth M, Kuna M. Constitutive modeling of plastic deformation behavior of open-cell foam structures using neural networks. Mechanics of Materials, 2019, 131: 1−10
43. Abendroth M, Kuna M. Determination of deformation and failure properties of ductile materials by means of the small punch test and neural networks. Computational Materials Science, 2003, 28(3–4): 633–644
44. Wu X, Ghaboussi J, Garrett J H Jr. Use of neural networks in detection of structural damage. Computers & Structures, 1992, 42(4): 649−659
45. Modarres C, Astorga N, Droguett E L, Meruane V. Convolutional neural networks for automated damage recognition and damage type identification. Structural Control and Health Monitoring, 2018, 25(10): e2230
46. Zang C, Imregun M. Structural damage detection using artificial neural networks and measured for data reduced via principal component projection. Journal of Sound and Vibration, 2001, 242(5): 813−827
47. Khadilkar A, Wang J, Rai R. Deep learning-based stress prediction for bottom-up sla 3d printing process. International Journal of Advanced Manufacturing Technology, 2019, 102(5−8): 2555−2569
48. Nie Z, Jiang H, Kara L B. Stress field prediction in cantilevered structures using convolutional neural networks. Journal of Computing and Information Science in Engineering, 2020, 20(1): 011002
49. Guo H, Zhuang X, Rabczuk T. A deep collocation method for the bending analysis of Kirchhoff plate. 2021, arXiv:2102.02617
50. Anitescu C, Atroshchenko E, Alajlan N, Rabczuk T. Artificial neural network methods for the solution of second-order boundary value problems. Computers, Materials & Continua, 2019, 59(1): 345−359
51. Samaniego E, Anitescu C, Goswami S, Nguyen-Thanh V M, Guo H, Hamdia K, Zhuang X, Rabczuk T. An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications. Computer Methods in Applied Mechanics and Engineering, 2020, 362: 112790
52. Zhuang X, Guo H, Alajlan N, Zhu H, Rabczuk T. Deep autoencoder based energy method for the bending, vibration, and buckling analysis of Kirchhoff plates with transfer learning. European Journal of Mechanics. A, Solids, 2021, 87: 104225
53. Guo H, Zhuang X, Rabczuk T. Stochastic analysis of heterogeneous porous material with modified neural architecture

search (NAS) based physics-informed neural networks using transfer learning. 2020, arXiv:2010.12344

54. Guo H, Zhuang X, Chen P, Alajlan N, Rabczuk T. Analysis of three-dimensional potential problems in non-homogeneous media with physics-informed deep collocation method using material transfer learning and sensitivity analysis. Engineering with Computers, 2022, 1−22
55. Zahraei S M, Heidarzadeh M. Destructive effects of the 2003 bam earthquake on structures. Asian Journal of Civil Engineering, 2007, 8(3): 329−342
56. Zahrai S M, Bolandi H. Towards lateral performance of CBF with unwanted eccentric connection: A finite element modeling approach. KSCE Journal of Civil Engineering, 2014, 18(5): 1421−1428
57. Zahrai S M, Bolandi H. Numerical study on the impact of out-of-plane eccentricity on lateral behavior of concentrically braced frames. International Journal of Steel Structures, 2019, 19(2): 341−350
58. Bolandi H, Zahrai S M. Influence of in-plane eccentricity in connection of bracing members to columns and beams on the performance of steel frames. Journal of Civil Engineering, 2013, 24(1): 91−102
59. Masci J, Meier U, Cireşan D, Schmidhuber J. Stacked convolutional auto-encoders for hierarchical feature extraction. In: International Conference on Artificial Neural Networks. Heidelberg: Springer, 2011
60. Hu J, Shen L, Sun G. Squeeze-and-excitation networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Salt Lake City: IEEE, 2018
61. Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L C. Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Salt Lake City: IEEE, 2018