Bridging Finite Element and Deep Learning: High-Resolution Stress Distribution Prediction in Structural Components

Hamed Bolandi [1,*], Xuyang Li [1], Talal Salem[1], Vishnu Naresh Boddeti [2], Nizar Lajnef [1]

[1] **Department of Civil and Environmental Engineering, Michigan State University, East Lansing, MI 48824,**

[2] **Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824,**

* Corresponding author, Email: 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 solutions but require solving large linear systems that can be computationally intensive. Instead, Deep learning (DL) techniques can generate solutions 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: (i) Modeling the geometry and its components, which can be time-consuming based on the system complexity; (ii) Specifying material properties, boundary conditions, and loading; (iii) Applying a meshing strategy for the geometry. The time-consuming and complexity of the current FEA workflow 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 to bypass FEA. This method may enable real-time stress analysis by leveraging 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 topology optimization [3, 4, 5, 6], data-driven approaches in fluid dynamics [7, 8, 9, 10], molecular dynamics simulation [11, 12, 13, 14], and material properties prediction [15, 16, 17, 18]. Also, Atalla et al. and Levin et al. [19, 20] have used neural regression for FEA model updating. Recently, deep learning has shown promise in solving conventional mechanics problems. Some researchers used deep learning for structural damage detection, a promising alternative to conventional structural health monitoring methods [21, 22, 23, 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 hard to generalize to more complicated cases. The numerical quadrature of the element stiffness matrix in the FEA on a per-element basis was optimized by Oishi et al. using deep learning [26]. 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 deep learning 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 modeled steel gusset plates with loading applied at different edges, different boundary conditions, and varying complex geometries. A dataset initialized with 104,448 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 machine learning over time from pattern recognition and learning theory [29]. Samuel [30] defined machine learning 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 a model 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 artificial intelligence has struggled to solve for many years [29]. 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, precisely computer vision and image recognition. These methods have shown significant performance in image classification [31], natural sentence classification [32], and image segmentation [33].

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 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. [34] 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 developed within the years [35-37], and the performance improved remarkably as the networks became more complicated and deeper [38,39].

CNN's use four concepts to enhance their performance: local connections, weight sharing, pooling, and multiple layers. CNN's are composed of a series of steps. The first layer is 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. 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 [29].

3. Deep learning in Civil and Mechanical Engineering

Artificial neural networks with multilayer perceptron (MLPs) have been used in civil and mechanical engineering for years. Researchers use ANN for structural analysis [40 41, 42], regression of material constitutive properties [43, 44], and materials' failure and damage [45, 46, 47]. Gulgec et al. [48] proposed a CNN architecture to classify simulated damaged and intact samples and localize the damage in steel gusset plates. Modares et al. [49] studied composite materials to identify the presence and type of structural damage using convolutional neural

networks. Also, for detecting concrete cracks without calculating the defect features, Cha et al. [50] proposed a vision-based method based on convolutional neural networks (CNNs). An approach for predicting stress distribution on all layers of non-uniform 3D parts was presented by Khadilkar et al. [51]. More recently, Nie et al. [52] developed a CNN-based approach to predict the stress field in a 2D linear cantilever beam. However, these works use DL techniques for structural analysis. Guo et al. [53] studied the bending analysis of Kirchhoff plates of various shapes, loads, and boundary conditions. Itescu et al. [54] present an artificial neural network-based collocation method for solving boundary value problems. Samaniego et al. [55] studied Deep Neural Networks as a technique for approximating data, and they have shown promising results in areas such as visual recognition. Zhuang et al. [56] propose a deep autoencoder-based energy method (DAEM) for the bending, vibration, and buckling analysis of Kirchhoff plates. Guo et al. [57] presented a modified neural architecture search method based on physics-informed deep learning for stochastic analysis of heterogeneous porous materials. Guo et al. [58] proposed a deep collocation method (DCM) based on transfer learning for solving potential problems in non-homogeneous media. To our knowledge, this 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 homogeneous and isotropic linear elastic material. The 2D steel plates approach the geometry of gusset plates. The boundary conditions and loading angles simulate

similar conditions in common gusset plate structures under external loading. Gusset plates connect beams and columns to braces in steel structures. The behavior and analysis of these components are essential since various reports have observed failures of gusset plates subject to lateral loads [59-62]. The distributed static loads applied to the plates in this study ranged from 1 to 5 kN with intervals of 1 kN. Moreover, loads were applied with three angles, including $\pi/6$, $\pi/4$, and $\pi/3$, 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, like real gusset plates' boundary conditions. All the translational and rotational displacements were 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 cm to 60 cm. Various geometries are generated by changing the position of each node in horizontal and vertical directions, as shown in Fig. 1, which led to 1024 unique pentagons. The material properties remain unchanged and isotropic for all samples.

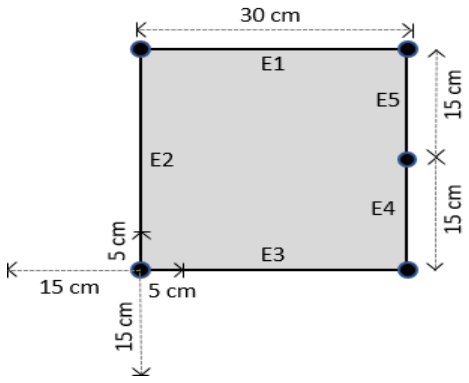


Fig 1. Basic schematic topology for initializing the steel plate geometries.

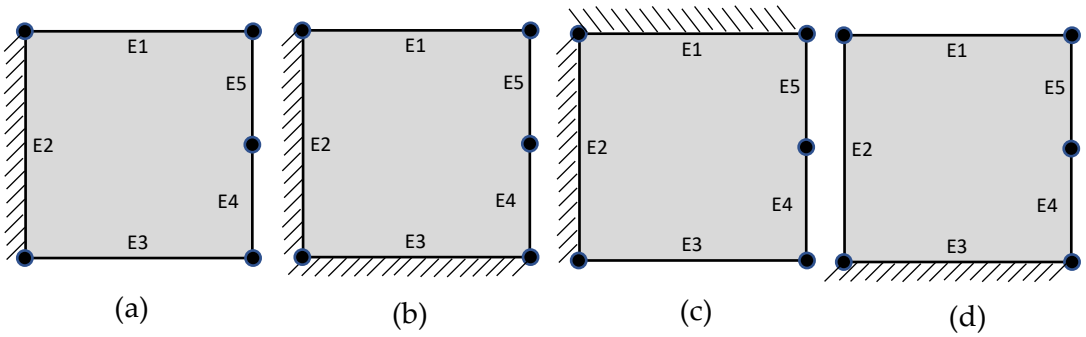


Fig 2. Different types of boundary conditions for initializing population.

Table 1. Input variables

Geometry	Boundary conditions	Load position	Load angle (degree)	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

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) denote 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. (3b)). Moreover, each horizontal and vertical component of the load is encoded as one 600 × 600-pixel single-channel colored image, as shown in Fig. 3(c) and 3(d). The magnitude of the horizontal and vertical components of the loads, after decomposition, varies between 0.5 kN 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 (Fig. 3(c) and 3(d)).

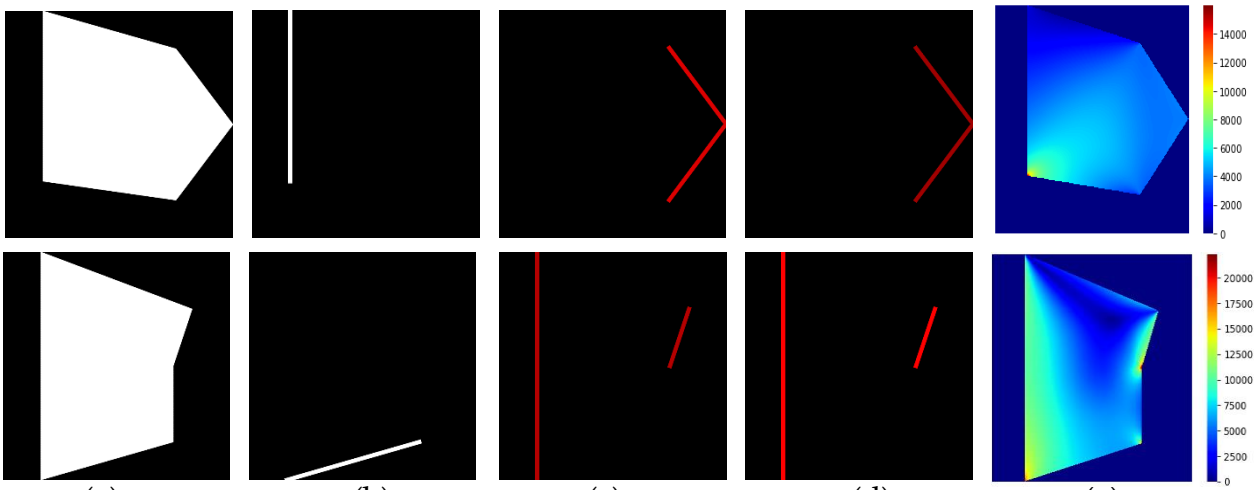


Fig 3. Input and output representation for stress distribution prediction: (a) geometry, (b) boundary condition, (c) horizontal load, (d) vertical load, (e) output

4.1.2 Output Data

FEA was 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 × 600 mm, and the smallest are 300 × 300 mm. Therefore, the size of each element is 1×1 mm, which means that each image has 360000 pixels. This high-resolution dataset led to achieving significant accuracy. The maximum and minimum von Mises stress values for elements among the entire dataset are 96,366 MPa 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. CNN 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 [63]. Our CNN architecture consists of three types of layers: The first stage is downsampling layers which consist of seven convolutional layers (E1, E2, E3, E4, E5, E6, E7), and the second stage is three layers (RS1, RS2, and Rs3) of Squeeze-Excitation and Residual blocks (SE-ResNet). In addition, we have swapped the SE-ResNet block with the Inception and MobileNetV2 blocks to check if these modules will further enhance the network's performance. The last is upsampling layers, consisting of six deconvolutional layers (D1, D2, D3, D4, D5, D6), as illustrated in Fig. 4.

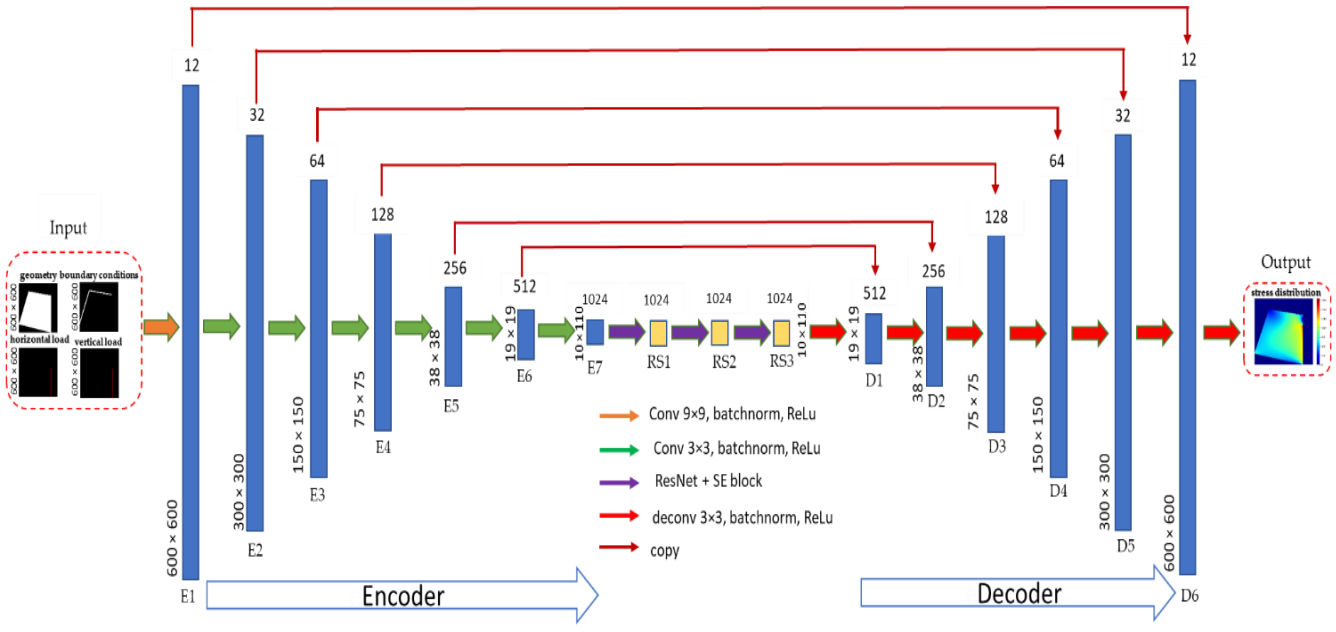


Fig 4. Proposed CNN architecture

5.1 Residual block

We used residual blocks [64] to address the vanishing gradient problem. In addition, SE blocks are computationally lightweight and result in only very small increases in model complexity [65]. As illustrated in Fig. 5, the formulation of F(x)+x can be realized by feedforward neural networks with shortcut connections. The shortcut connection simply performs identity mapping, and its output is added to the output of the stacked layers [65].

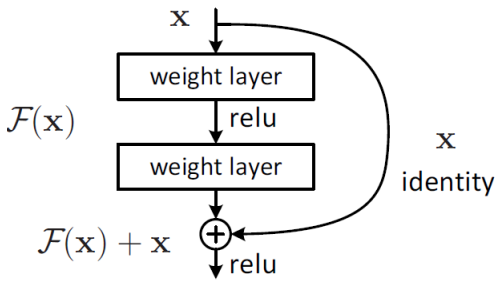


Fig 5. The building block of residual learning [64]

5.2 Squeeze and Excitation block

As depicted in Fig. 6, Squeeze-and-Excitation blocks improve the representative capacity of the network, enabling dynamic channel-wise feature recalibration. A SE-block can be implemented with five steps. First, we feed the input x as a convolutional block. The current number of channels to the SE function, F_{tr} in Fig. 6 is the convolutional operator for the transformation of X to U. Then, in the second phase, each channel is squeezed into a single numeric value using the average pooling. Additionally, the third phase of a fully connected layer is followed by a ReLu function, which applies necessary nonlinearity to reduce the output channel complexity. Then in the fourth phase, SE blocks can be used directly with residual networks. Fig. 7 depicts a 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 squeeze and excitation act. Using both SE and ResNet in the network outperforms using ResNet [65].

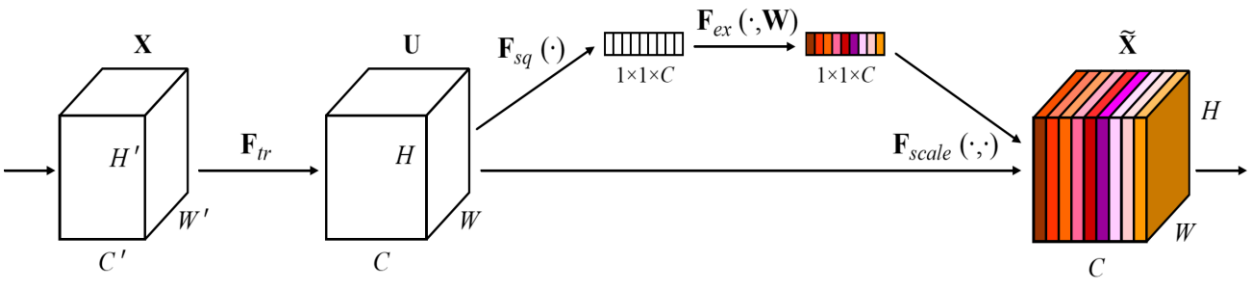


Fig 6. The building block of Squeeze-and-Excitation [65]

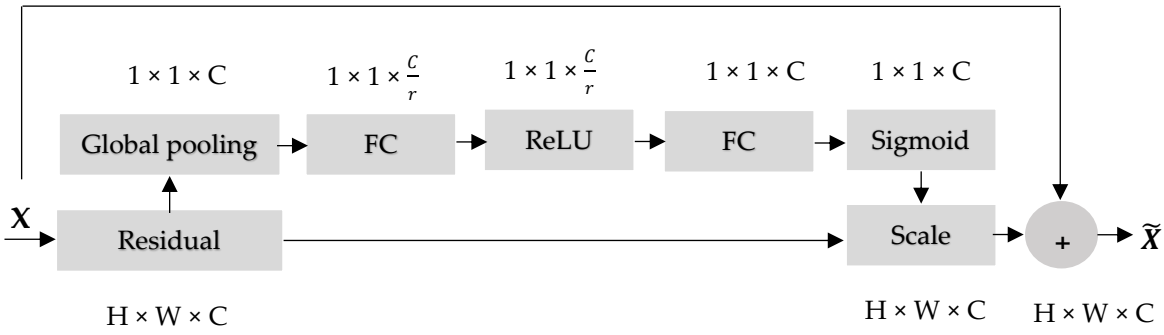


Fig 7. SE-ResNet module

5.3 Inception block

The Inception Modules are used to reduce the computational cost of convolutional neural networks (CNNs). Since neural networks have to deal with a vast array of images, each with varying 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 (1x1, 3x3, 5x5) 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 networks, this method

offers significant quality gains at a modest computational cost increase [66]. Fig.8 depicts the inception module.

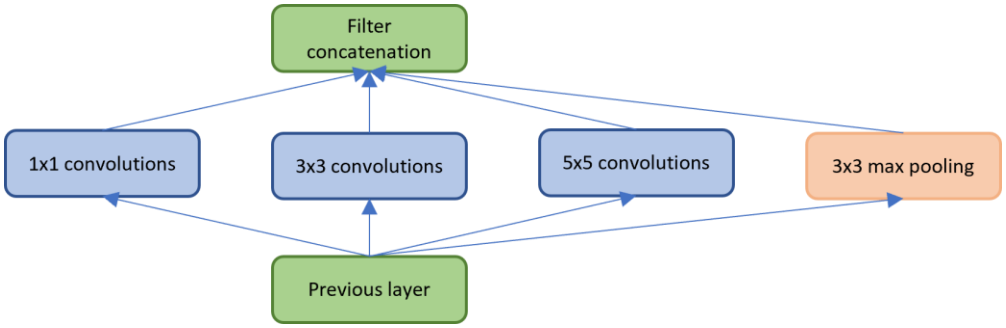


Fig 8. Inception module

5.4 MobileNetV2 block

MobileNetV2 is based on an inverted residual block with shortcut connections between thin bottleneck layers [67]. 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 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. Fig.9 depicts the MobileNetV2.

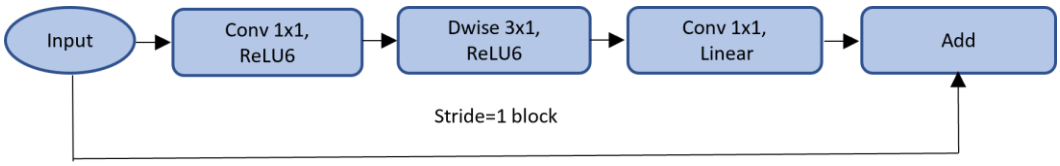


Fig 9. MobileNetV2 module

5.5 Network layers and hyperparameters

All the details of the network layers and hyperparameters can be found in Table 2. As can be seen, the models consist of seven Conv layers, three different bottleneck blocks, and six ConvT layers. We tried 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 kept the network with 1024 channels as the primary model and swapped the bottleneck each time with the SE-ResNet, Inception, and MobileNetV2. We kept 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. We tried different learning rates from 1e-3 to 1e-6, and 1e-5 led to the best convergence.

Table 2. Network layers and hyperparameter

Network Layers

	Number of layers	The first layer (HxWxC)	Last layer (HxWxC)	Activation
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
EfficentNet	2	10×10×1024	10×10×1024	ReLU6
ConvT	6	19×19×512	600×600×12	ReLU

Network Hyperparameters

batch size	learning rate	weight decay	epochs of training	expand ratio	Loss functions
16	1.00E-05	1.00E-07	120	6	MSE-MAE

6. Loss function and performance metrics

We used MSE (mean squared error) for the training loss defined in Equation 1. MSE gives a more significant penalty to large errors than MRE (mean relative error). Also, the errors will be 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 Equations 2,3,4,5, respectively.

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

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

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.

Measure the accuracy of predictions by comparing them to the ground truth; we used MRE:

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

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

The percentage mean absolute error is defined as:

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

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

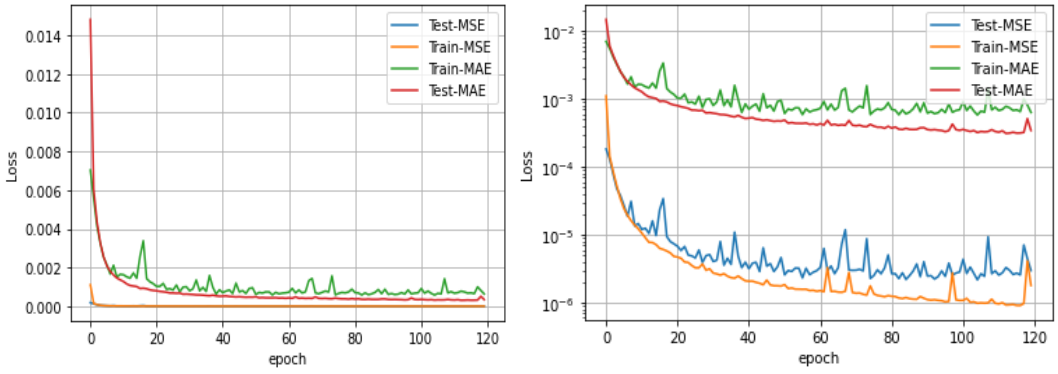
PAE and PPAE measure the accuracy of the most significant stress value in the predicted stress distribution. PAE and PPAE are defined as:

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

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

9. Results and discussion

All codes are written in PyTorch Lightning and run on two NVIDIA TITAN RTX 24G GPUs. We used AdamW (Adam algorithm with weight decay) optimizer to speed up the convergence of our models. We trained and evaluated different models based on Table 3 to find the model with the best performance. The training data size of models 1 to 3 is 83,558, and the testing data size is 20,890, randomly divided with a train/test ratio of 80% - 20%. Fig. 10 shows MSE and MAE losses as a function of epochs in model 1. Fig. 10(a) and 10(b) are linear and logarithmic scales. Fig. 10(a) shows that the MSE and MAE curves rapidly declined after a few epochs. However, Fig. 10(b) gives a more precise representation of the model's behavior. Fig. 10(b) shows that MSE is less than MAE, but both have almost similar trends.



(a) (b)

Fig 10. MSE and MAE curves on training and testing data with two scales: (a) linear scale, (b) logarithmic scale.

We saved 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 our model is generalizable. To reduce the computational cost, we divided the dataset into three folds. K-fold

cross-validation shows the best performance in all models based on most metrics in Table 3, which means our model is generalizable.

Table 3. Error metrics for models at best checkpoints (Units: mm-MPa-N)

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

We replaced 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. Fig.11 depicts the performance of different models in terms of MAE. As it can be seen in Fig. 11 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 MAEs. We deem these results satisfactory for stress distribution predictions, specifically the PPAE, the most critical load value for stress distribution in engineering domain applications.

Fig. 12 illustrates the cumulative distribution of PMAE and PPAE in the test dataset of model 1. Fig. 12(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. Fig. 12(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.

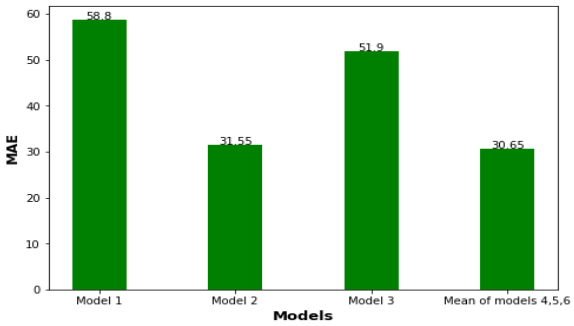
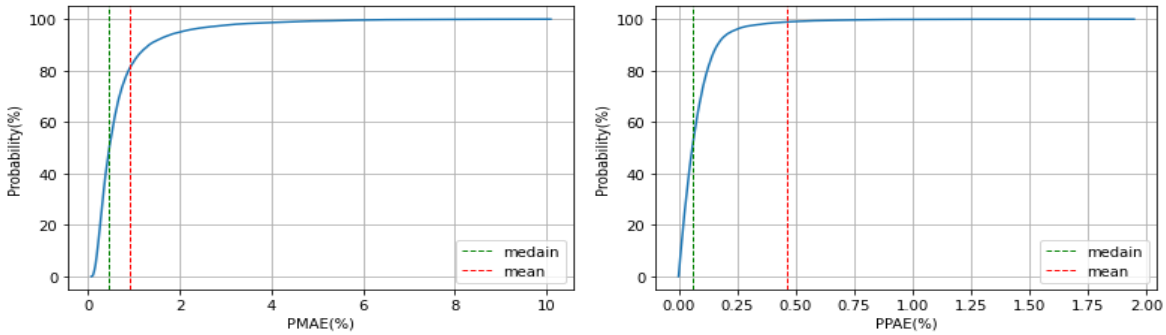


Fig 11. Comparison of different models in terms of MAE

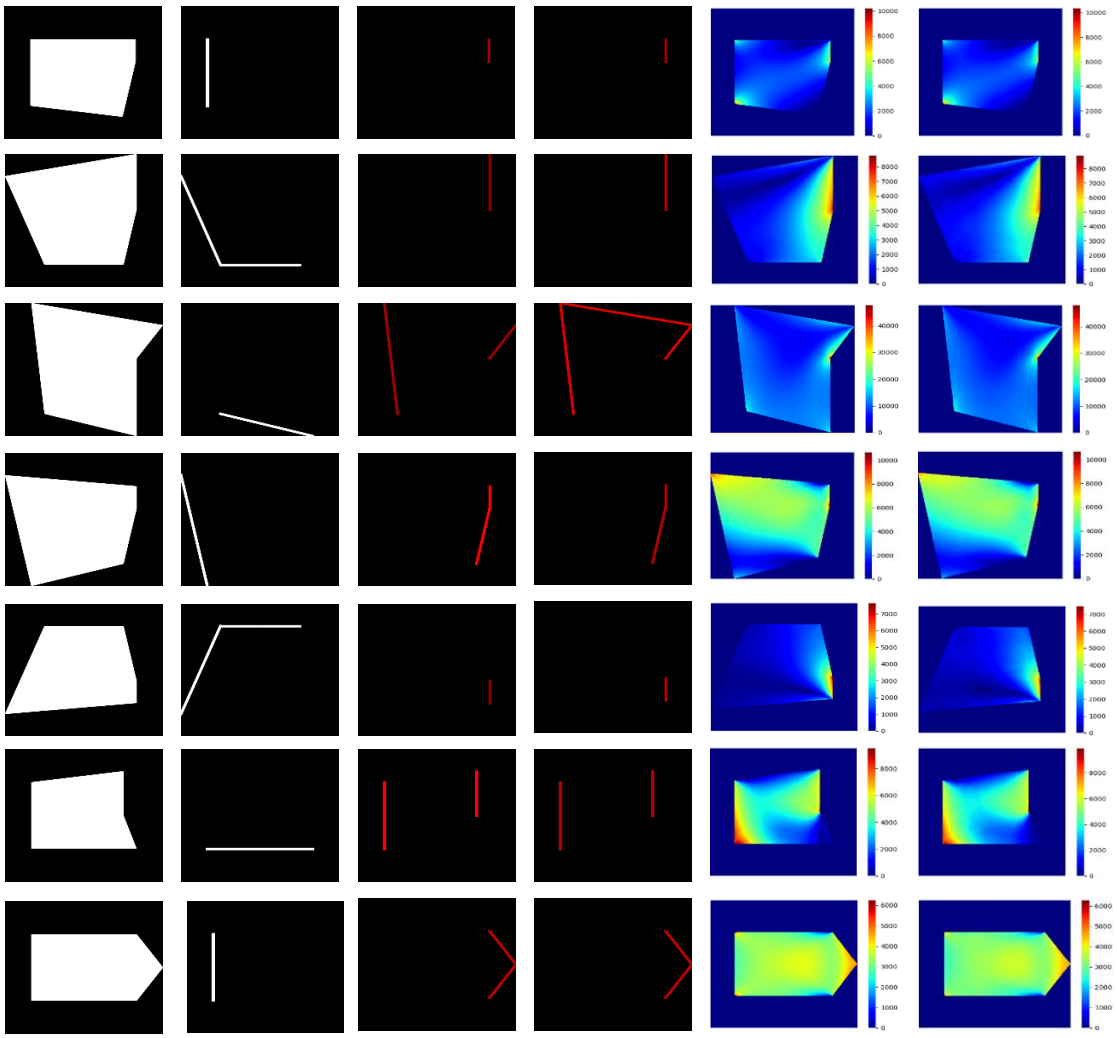


(a) (b)

Fig 12. 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

The prediction results of some randomly selected samples from the test dataset of model 1 are visualized in Fig. 13. 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 it 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. 14. These predictions still provide useful information.



(a) (b) (c) (d) (e) (f)

Fig 13. Predicted stress distribution and corresponding inputs with different loads and boundary conditions scenarios. Columns (a) to (d) represent geometry, boundary conditions, and load in a horizontal and vertical direction. Columns (e) and (f) represent ground truth and predicted stress distribution, respectively. (Units = mm-MPa-N)

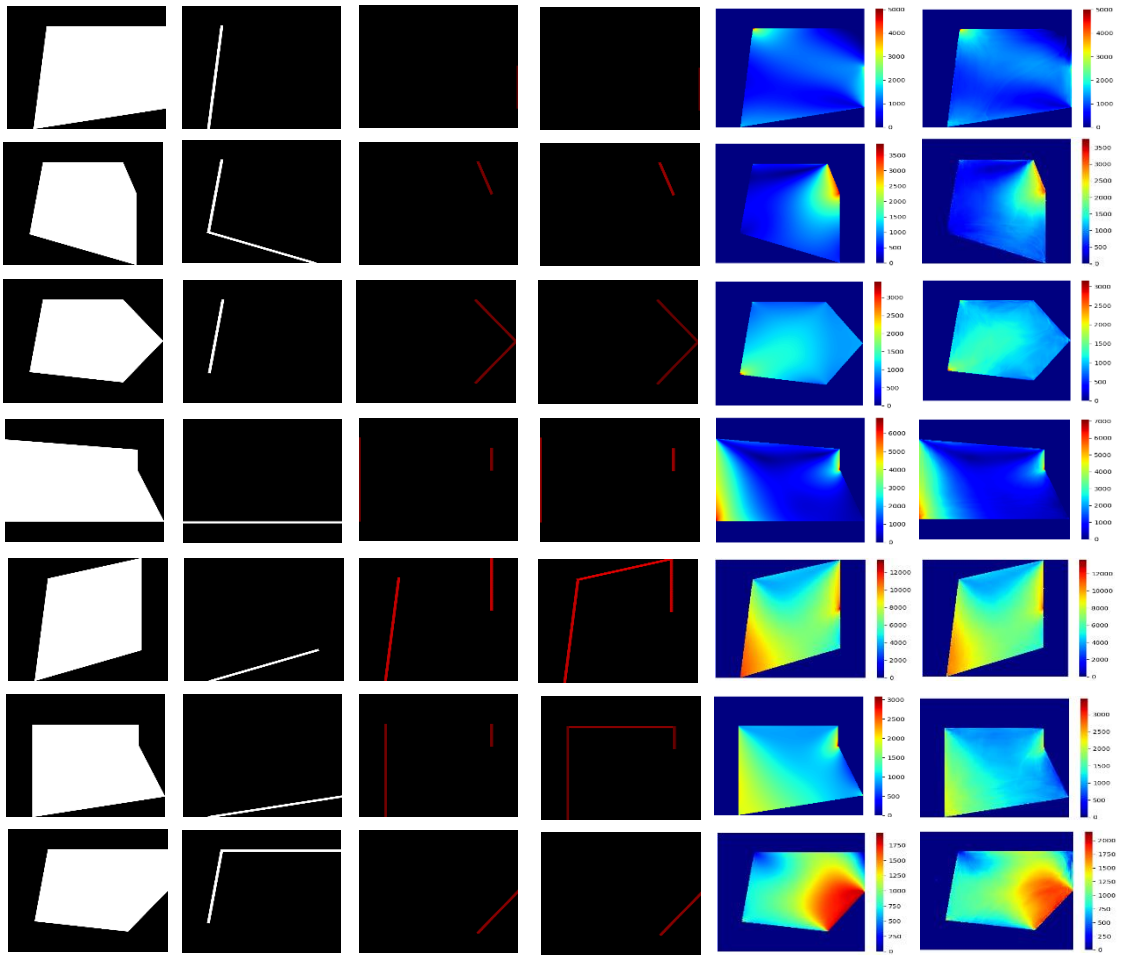


Fig 14. Inaccurate predicted stress distribution and corresponding inputs with different loads and boundary conditions scenarios. Columns (a) to (d) represent geometry, boundary conditions, and load in a horizontal and vertical direction. Columns (e) and (f) represent ground truth and predicted stress distribution, respectively. (Units = mm-MPa-N)

9.1 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, 104,448 samples, we train the network with 10k, 20k, 30k, 40k, 50k, and 70k samples. Fig. 15 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.

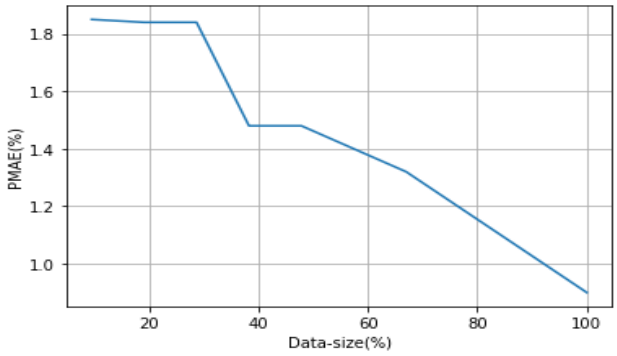


Fig 15. PMAE at different data sizes.

We also evaluate the effect of data size on the gaussian distributions of PMAE, and PPAE illustrated in Figs 16(a) and 16(b). As shown in Fig 16(a), increasing the data size decreases the standard deviation of PMAE. However, 70k and the total data size have almost the same standard deviation. Fig. 16(b) shows that the standard deviation of PPAE decreased when the data size increased from 50K to 70K. As a result, we should train the network with at least 70k examples, 67% of our dataset, to improve PPAE's standard deviation accuracy.

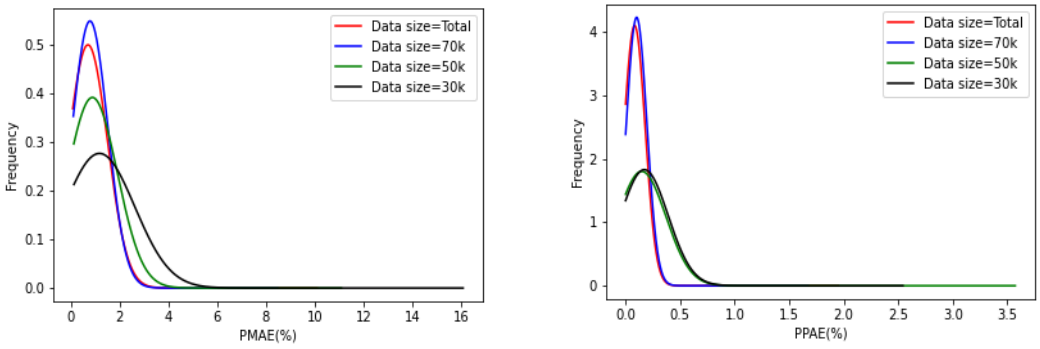


Fig 16. Gaussian distribution of PMAE and PPAE. (a) PMAE, (b) PPAE

10. Conclusion

In this paper, we used end-to-end deep learning techniques. We developed a convolutional neural network (CNN) to alleviate the need for finite element methods to predict 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 returns high-resolution stress contours as the output. We used the PDE toolbox of MATLAB to generate the output data for training, containing 104,448 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 are also studied. Training the network with just 10% of the dataset achieved a mean error of 1.85%, 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 decreases the standard deviation of mean error. The standard deviation of maximum stress error decreased with the increasing number of samples. Furthermore, the Gaussian distributions of mean and maximum stress errors demonstrate that more data induce less standard deviation in PMAE and PPAE.

Funding: This research was funded in part by National Science Foundation, grant number CNS 1645783.

References

[1] Le Cun Y, Bengio Y, Hinton GE. 2015 Deep learning. Nature 521, 436–444. (doi:10.1038/nature14539)

[2] Schmidhuber J. 2015 Deep learning in neural networks: an overview. Neural Netw. 61, 85–117. (doi:10.1016/j.neunet.2014.09.003)

[3] Nobuyuki Umetani. Exploring generative 3d shapes using autoencoder networks. Pages 1–4, 11 2017.

[4] Yonggyun Yu, Taeil Hur, and Jaeho Jung. Deep learning for topology optimization design. CoRR, abs/1801.05463, 2018.

[5] Wentai Zhang, Haoliang Jiang, Zhangsihao Yang, Soji Yamakawa, Kenji Shimada, and Levent Burak Kara. Data-driven upsampling of point clouds. CoRR, abs/1807.02740, 2018.

[6] Erva Ulu, Rusheng Zhang, Mehmet Ersin Yumer, and Levent Burak Kara. 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, pages 387–399. Springer International Publishing, 2014.

[7] Abhijit Guha Roy, Sailesh Conjeti, Sri Phani Krishna Karri, Debdoot Sheet, Amin Katouzian, ChristianWachinger, and Nassir Navab. Relaynet: retinal layer and fluid segmentation of macular optical coherence tomography using fully convolutional networks. Biomed. Opt. Express, 8(8):3627–3642, Aug 2017.

[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. arXiv e-prints, page arXiv:1804.09269, Apr 2018.

[9] Amir Barati Farimani, Joseph Gomes, and Vijay S. Pande. Deep learning the physics of transport phenomena. CoRR, abs/1709.02432, 2017.

[10] Byungsoo Kim, Vinicius C. Azevedo, Nils Thuerey, Theodore Kim, Markus H. Gross, and Barbara Solenthaler. Deep fluids: A generative network for parameterized fluid simulations. CoRR, abs/1806.02071, 2018.

[11] Garrett B. Goh, Nathan O. Hodas, and Abhinav Vishnu. Deep learning for computational chemistry. Journal of Computational Chemistry, 38(16):1291–1307, 2017.

[12] Andreas Mardt, Luca Pasquali, Hao Wu, and Frank Noé. VAMPnets for deep learning of molecular kinetics. Nature Communications, 9:5, Jan 2018.

[13] Grégoire Montavon, Matthias Rupp, Vivekanand Gobre, Alvaro Vazquez-Mayagoitia, Katja Hansen, Alexand re Tkatchenko, Klaus-Robert Müller, and O. Anatole von Lilienfeld. Machine learning of molecular electronic properties in chemical compound space. New Journal of Physics, 15:095003, Sep 2013.

[14] Gareth A. Tribello, Michele Ceriotti, and Michele Parrinello. A self-learning algorithm for biased molecular dynamics. Proceedings of the National Academy of Sciences, 107(41):17509–17514, 2010.

[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. Math. Probl. Eng. 2014, doi.:10.1155/2014/474289.

[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 Comput. Appl. 2015, doi:10.1007/s00521-015-1997.

[17] Mousavi, S. M., Aminian, P., Gandomi, A. H., Alavi, A. H., & Bolandi, H. (2012). A new predictive model for compressive strength of HPC using gene expression programming. Advances in Engineering Software, 45(1), 105-114.

[18] Bolandi, H., Banzhaf, W., Lajnef, N., Barri, K., & Alavi, A. H. (2019). An Intelligent Model for the Prediction of Bond Strength of FRP Bars in Concrete: A Soft Computing Approach. Technologies, 7(2), 42.

[19] M.J. Atalla and D.J. Inman. On model updating using neural networks. Mechanical Systems and Signal Processing, 12(1):135 – 161, 1998.

[20] R.I. Levin and N.A.J. Lieven. Dynamic finite element model updating using neural networks. Journal of Sound and Vibration, 210(5):593 – 607, 1998.

[21] Fan, Z., Wu, Y., Lu, J., & Li, W. (2018). Automatic pavement crack detection based on structured prediction with the convolutional neural network. arXiv preprint arXiv:1802.02208.

[22] Dung, C. V. (2019). Autonomous concrete crack detection using deep fully convolutional neural network. Automation in Construction, 99, 52-58.

[23] Gulgec, N. S., Takáč, M., & Pakzad, S. N. (2019). Convolutional neural network approach for robust structural damage detection and localization. Journal of Computing in Civil Engineering, 33(3), 04019005.

[24] Cha, Y. J., Choi, W., & Büyüköztürk, O. (2017). Deep learning-based crack damage detection using convolutional neural networks. Computer-Aided Civil and Infrastructure Engineering, 32(5), 361-378.

[25] A.A. Javadi and T P. Tan. Neural network for constitutive modeling in finite element analysis. Computer Assisted Mechanics and Engineering Sciences, 10, 01 2003.

[26] Atsuya Oishi and Genki Yagawa. Computational mechanics enhanced by deep learning. Computer Methods in Applied Mechanics and Engineering, 327:327 – 351, 2017. Advances in Computational Mechanics and Scientific Computation—the Cutting Edge.

[27] Madani, A., Bakhaty, A., Kim, J., Mubarak, Y., & Mofrad, M. R. (2019). Bridging finite element and machine learning modeling: stress prediction of arterial walls in atherosclerosis. Journal of biomechanical engineering, 141(8).

[28] Liang Liang, Minliang Liu, Caitlin Martin, and Wei Sun. A deep learning approach to estimate stress distribution: a fast and accurate surrogate of finite-element analysis. Journal of The Royal Society Interface, 15, 01 2018.

[29] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. nature, 521(7553), 436-444

[30] Samuel, A. L. 1959. "Some studies in machine learning using the game of checkers." IBM J. Res. Dev. 3 (3): 210–229. http://doi.org/10.1147/rd .33.0210

[31] Alpaydin, E. 2014. Introduction to machine learning. Cambridge, MA: MIT Press.

[32] Kim, Y. (2014). Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882

[33] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

[34] LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner. 1998. "Gradient-based learning applied to document recognition." Proc. IEEE 86 (11): 2278–2324. https://doi.org/10.1109/5.726791.

[35] Krizhevsky, A., I. Sutskever, and G. E. Hinton. 2012. "ImageNet classification with deep convolutional neural networks." In Advances in neural information processing systems, 1097–1105. Neural Information Processing Systems

[36] Simonyan, K., and A. Zisserman. 2014. "Very deep convolutional networks for large-scale image recognition." http://arXiv.org/abs/1409.1556.

[37] Zeiler, M. D., and R. Fergus. 2014. "Visualizing and understanding convolutional networks." In Proc., European Conf. on Computer Vision, 818–833. New York: Springer.

[38] Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. 2015. "Going deeper with convolutions." In Proc., IEEE Conf. on Computer Vision and Pattern Recognition, 1–9. New York: IEEE.

[39] He, K., X. Zhang, S. Ren, and J. Sun. 2015. "Deep residual learning for image recognition." Preprint, submitted December 10, 2015. http://arXiv.org/abs/1512.03385.

[40] WM Jenkins. Neural network-based approximations for structural analysis. In Developments in Neural Networks and Evolutionary Computing for Civil and Structural Engineering, pages 25–35. Civil-Comp Press Edinburgh, 1995.

[41] Zenon Waszczyszyn and Leonard Ziemia´nski. Neural networks in mechanics of structures and materials–new results and prospects of applications. Computers & Structures, 79(22-25):2261–2276, 2001.

[42] ATC Goh, KS Wong, and BB Broms. Multivariate modelling of fem data using neural networks. CIVIL-COMP95 Developments in Neural Networks and Evolutionary Computing for Civil and Structural Engineering, pages 59–64, 1995. results and prospects of applications. Computers & Structures, 79(22-25):2261–2276, 2001.

[43] N Huber and Ch Tsakmakis. 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, 47(7):1589–1607, 1999.

[44] Christoph Settgast, Martin Abendroth, and Meinhard Kuna. Constitutive modeling of plastic deformation behavior of open-cell foam structures using neural networks. Mechanics of Materials, 2019.

[45] Martin Abendroth and Meinhard Kuna. Determination of deformation and failure properties of ductile materials by means of the small punch test and neural networks. computational materials Science, 28(3-4):633–644, 2003.

[46] X Wu, J Ghaboussi, and JH Garrett Jr. Use of neural networks in detection of structural damage. Computers & structures, 42(4):649–659, 1992.

[47] C Zang and M Imregun. Structural damage detection using artificial neural networks and measured for data reduced via principal component projection. Journal of sound and vibration, 242(5):813–827, 2001.

[48] Gulgec, N. S., Takáč, M., & Pakzad, S. N. (2019). Convolutional neural network approach for robust structural damage detection and localization. Journal of Computing in Civil Engineering, 33(3), 04019005.

[49] Modarres, C., Astorga, N., Droguett, E. L., & Meruane, V. (2018). Convolutional neural networks for automated damage recognition and damage type identification. Structural Control and Health Monitoring, 25(10), e2230.

[50] Cha, Y. J., Choi, W., & Büyüköztürk, O. (2017). Deep learning-based crack damage detection using convolutional neural networks. Computer-Aided Civil and Infrastructure Engineering, 32(5), 361-378.

[51] Khadilkar, A., Wang, J., & Rai, R. (2019). Deep learning-based stress prediction for bottom-up sla 3d printing process. The International Journal of Advanced Manufacturing Technology, 102(5-8), 2555-2569.

[52] Nie, Z., Jiang, H., & Kara, L. B. (2020). Stress Field Prediction in Cantilevered Structures Using Convolutional Neural Networks. Journal of Computing and Information Science in Engineering, 20(1).

[53] Guo, H., Zhuang, X., & Rabczuk, T. (2021). A deep collocation method for the bending analysis of Kirchhoff plate. arXiv preprint arXiv:2102.02617.

[54] Anitescu, C., Atroshchenko, E., Alajlan, N., & Rabczuk, T. (2019). Artificial neural network methods for the solution of second-order boundary value problems. Computers, Materials, and Continua, 59(1), 345-359.

[55] Samaniego, E., Anitescu, C., Goswami, S., Nguyen-Thanh, V. M., Guo, H., Hamdia, K., ... & Rabczuk, T. (2020). 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, 362, 112790.

[56] Zhuang, X., Guo, H., Alajlan, N., Zhu, H., & Rabczuk, T. (2021). Deep autoencoder based energy method for the bending, vibration, and buckling analysis of Kirchhoff plates with transfer learning. European Journal of Mechanics-A/Solids, 87, 104225.

[57] Guo, H., Zhuang, X., & Rabczuk, T. (2020). Stochastic analysis of heterogeneous porous material with modified neural architecture search (NAS) based physics-informed neural networks using transfer learning. arXiv preprint arXiv:2010.12344.

[58] Guo, H., Zhuang, X., Chen, P., Alajlan, N., & Rabczuk, T. (2022). 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, 1-22.

[59] Zahraei, S. M., & Heidarzadeh, M. (2007). Destructive effects of the 2003 bam earthquake on structures.

[60] Zahrai, S. M., & Bolandi, H. (2014). Towards lateral performance of CBF with unwanted eccentric connection: A finite element modeling approach. KSCE Journal of Civil Engineering, 18(5), 1421-1428.

[61] Zahrai, S. M., & Bolandi, H. (2019). Numerical Study on the Impact of Out-of-Plane Eccentricity on Lateral Behavior of Concentrically Braced Frames. International Journal of Steel Structures, 19(2), 341-350.

[62] Bolandi, h., & Zahra, s. (2013). Influence of in-plane eccentricity in connection of bracing members to columns and beams on the performance of steel frames.

[63] Masci, J., Meier, U., Cireşan, D., & Schmidhuber, J. (2011, June). Stacked convolutional auto-encoders for hierarchical feature extraction. In International conference on artificial neural networks (pp. 52-59). Springer, Berlin, Heidelberg.

[64] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[65] Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-excitation networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7132-7141).

[66]. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1-9).

[67] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4510-4520).