ISA Transactions 103 (2020) 177–191

Contents lists available at ScienceDirect

ISA Transactions

journal homepage: www.elsevier.com/locate/isatrans

Research article

Deep learning for computational structural optimization

Long C. Nguyen [a], H. Nguyen-Xuan [a,b,*]

[a] CIRTECH Institute, Ho Chi Minh City University of Technology (HUTECH), Ho Chi Minh City, Viet Nam
[b] Department of Architectural Engineering, Sejong University, 209 Neungdong-ro, Gwangjin-gu, Seoul 05006, Republic of Korea

ARTICLE INFO

Article history:
Received 3 September 2019
Received in revised form 8 February 2020
Accepted 30 March 2020
Available online 10 April 2020

Keywords:
Deep neural network
Deep learning
Optimization
Truss structures
Chebyshev polynomial
Activation function

ABSTRACT

We investigate a novel computational approach to computational structural optimization based on deep learning. After employing algorithms to solve the stiffness formulation of structures, we used their improvement to optimize the structural computation. A standard illustration of 10 bar-truss was revisited to illustrate the mechanism of neural networks and deep learning. Several benchmark problems of 2D and 3D truss structures were used to verify the reliability of the present approach, and its extension to other engineering structures is straightforward. To enhance computational efficiency, a constant sum technique was proposed to generate data for the input of multi-similar variables. Both displacement and stress enforcements were the constraints of the optimized problem. The optimization data for cross sections with the objective function of total weight were then employed in the context of deep learning. The stochastic gradient descent (SGD) with Nesterov's accelerated gradient (NAG), root mean square propagation (RMSProp) and adaptive moment estimation (Adam) optimizers were compared in terms of convergence. In addition, this paper devised Chebyshev polynomials for a new approach to activation functions in single-layer neural networks. As expected, its convergence was quicker than the popular learning functions, especially in a short training with a small number of epochs for tested problems. Finally, a split data technique for linear regression was proposed to deal with some sensitive data.

1. Introduction

An artificial neural network (ANN) is a construction of calculation inspired by the composition of intellectual neural network. The architecture intelligence creates a great quantity of rudimental computing neurons using complex dispatch system, which the brain is capable of accomplishing exceedingly intricate calculations. In the mid-20th century, learning with neural network was first set up. Paradigm is efficient and recently has gained cutting-edge performance on numerous learning tasks. During the first 10 years of the 21st century, scientists have fostered some primary conceptual innovations namely: the deep learning technique, or deep belief network (DBN) in [1], rectified linear unit (ReLU) in [2], and drop out algorithm in [3]. Deep learning (DL) is a branch of machine learning based on a set of algorithms that attempt to simulate complex problems by using networks of neurons. This technique shows a higher level of machine learning algorithms which

- uses a group of multiple layers with non-linear processing units to get feature extraction and transformation.
- considers each consecutive layer as its input taken from preceding layer's output.
- learns in supervised (e.g., regression) and/or unsupervised (e.g., clustering) methods.
- gives the representation corresponding to abstraction, and forms a hierarchy of concepts being learnt in many levels.
- employs various neurons of gradient descent in training process via the back-propagation algorithm.

The crucial idea behind ANN is that several neurons can be united together as an association of communication to fulfill the complicated computation. The structure is described as a graph or a map whose nodes are stand for the neurons and every single (directed) border presented in the chart is responsible for conjoining the outputs of some neurons to the inputs of the others. DL represents learning with multiple processing layers in the hidden layers and includes linear and non-linear transformations [4]. However, some newer algorithms in the structured analysis have been applied since the article on structural engineering application of neural networks (NNs) was first published in [5]. Some simplifications of the problems lack relevant CPU that the required power supply is not adequate to the high demand of analysis process. Therefore, finding a faster algorithm has never stop. Some advance algorithms [6] based on the evolutionary algorithm via schematic diagram of a networked

* Corresponding author at: CIRTECH Institute, Ho Chi Minh City University of Technology (HUTECH), Ho Chi Minh City, Viet Nam.
E-mail address: ngx.hung@hutech.edu.vn (H. Nguyen-Xuan).

https://doi.org/10.1016/j.isatra.2020.03.033
0019-0578/

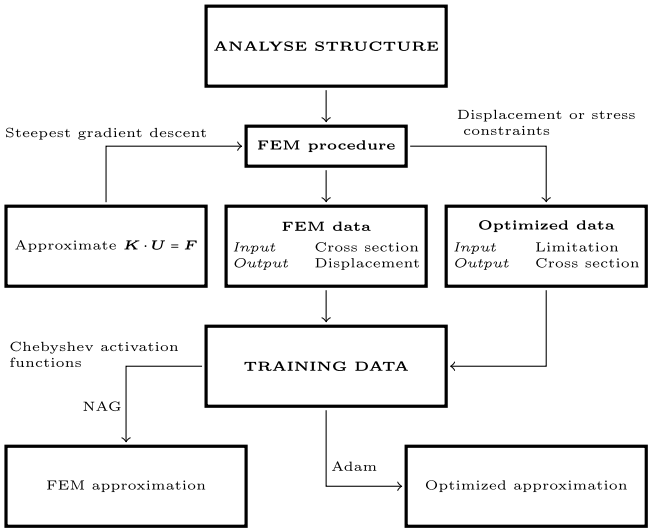

Fig. 1. Flowchart of DL structural optimization.

Table 1
The classes of neural networks based on layer architectures.

Single – Layer Neural Networks		Input Layer – Output Layer
Multilayer – Layer Neural Networks	Shallow Neural Networks	Input Layer – Hidden Layer – Output Layer
	Deep Neural Networks	Input Layer – Hidden Layers – Output Layers

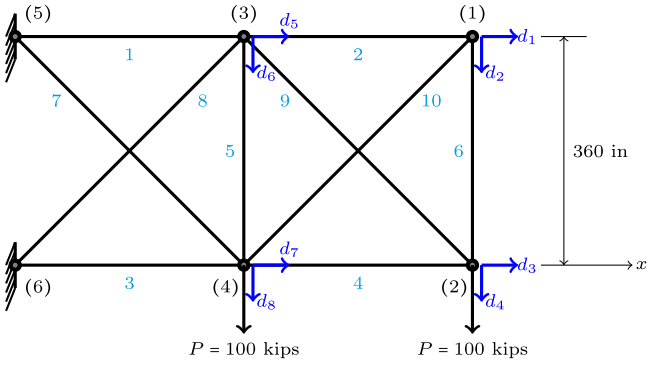

Fig. 2. A 10 – bar plan truss system.

interaction system with information process units and information transmission links have been found recently. Some further applications of DL to cellular imaging computational biology were then reported in [7]. It is evident that DL is necessary for biological data multi-dimension with rapid increasing and acquisition rate, which would have been challenging with the use of conventional analysis strategies. Their research provided analytical approaches in regulatory genomics [8] and cellular imaging to gain biological insights. It also pinpointed some possible pitfalls and limitations, which could help showing computational biologists when and how to present specific applications and provide tips for practical use of DL. Another important application of DL was computational chemistry [9], in which deep neural networks were widely applicable. DL possesses high ubiquity and broad applicability to a wide range of practical challenges relating to big data. Some topics of researches included quantitative structure activity relationship, virtual screening, protein structure prediction, quantum chemistry, materials design, property prediction, saving human lives as complexity science and information systems can contribute in [10] and social dilemmas in [11] where individual interests were at odds with the interests of others, and artificial intelligence might have had a particularly hard time making the right decision. DL techniques were also employed in translational bio-informatics, medical imaging, pervasive sensing, medical informatics, and public health [12]. The research provided a critical analysis of relative merits, potential pitfalls as well as prospect of the technique. However, there have been only a few applications of using DL in mechanics problems. In addition, neural networks also showed their limitations and numerical instability. A number of neural network investigations have referred to material modeling. For instance, the constitutive equations of materials of diverse types [13,14], usage associated with the constitutive exemplars and innovative training algorithms and network structure [15,16], the optimization of material composition of composites [17], and the modeling of hysteresis curves in a variety of topics [18,19]. Recently, Lee et al. [20] employed DL algorithms in structural analysis.

In this study, we propose a new methodology for solving structural optimization problems using DL. The flow chart displayed in Fig. 1 gives an overview of the present study. Using the well-known 10 – bar truss structure as an illustrative example, we propose some architectures of deep neural networks for the optimized problems based on the optimizers: SGD with NAG, RMSProp and Adam. In addition, we have found some better activation functions from Chebyshev polynomial in single-layer neural networks and provided solution to 2D (52 – bar and 200 – bar truss) and 3D (25 – bar space truss) structures. The extension of the new activation function to other structural problems is straightforward. Such findings can match with the current trend of the Fourth Industrial Revolution in which DL algorithms play a major role in big data analysis of structural engineering. A new technique to create training data with a constant sum is introduced. By using NN and DL, we can approximate the process and simplify the solution after activating learning data. Finally, a new way to split data for linear regression is proposed to deal with some sensitive data.

2. An illustration of how deep learning work

2.1. A brief on deep learning

From a simple start, the neural network has progressed to much more complicated one. The original networks known as the single-layer neural networks, shared an uncomplicated architecture only with input and output layers. Then, the hidden layers are attached into single-layer neural network. By this way, it gives out the multi-layer neural networks. As a consequence, the structure of the multi-layer neural networks include an input layer, hidden layer(s) as well as an output layer. A neural network with only one hidden layer is called shallow neural network or a vanilla neural network. A multi-layer neural network containing two or more latent layers is known as a deep neural network. The majority of the contemporary neural networks used presently are deep neural networks. Table 1 recapitulates the subdivisions of network using the layer architecture.

The neural networks momentarily confronted with some troubles. The performance of this networks in practice failed the expectations. Those networks did not give any positive results, and they even yielded poorer results. The back-propagation training along with the additional latent layers often led to dissatisfactory outcomes. Lacking of proper training is a cause for their poor performance. Three main slow-stoppers are: the vanishing gradient, over-fitting and computational load. Because of a simple architecture and concept of neural networks, there was not much

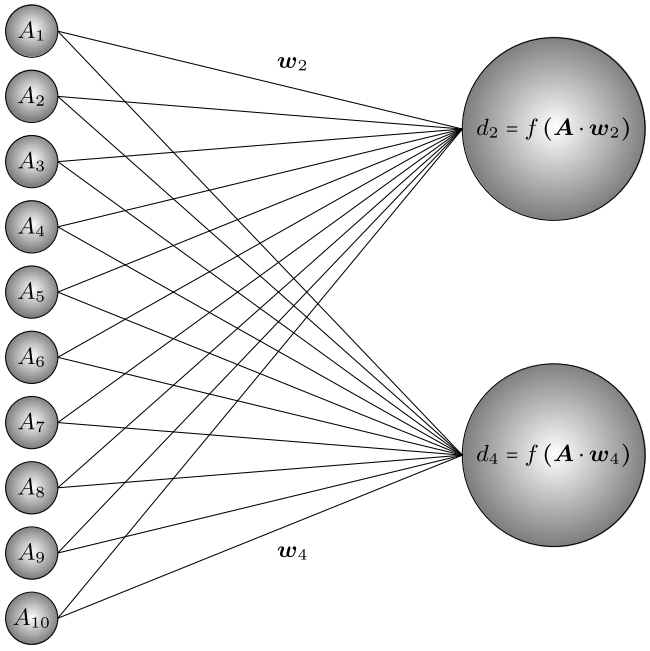

Fig. 3. Single neural network architecture for the 10 – bar plan truss system.

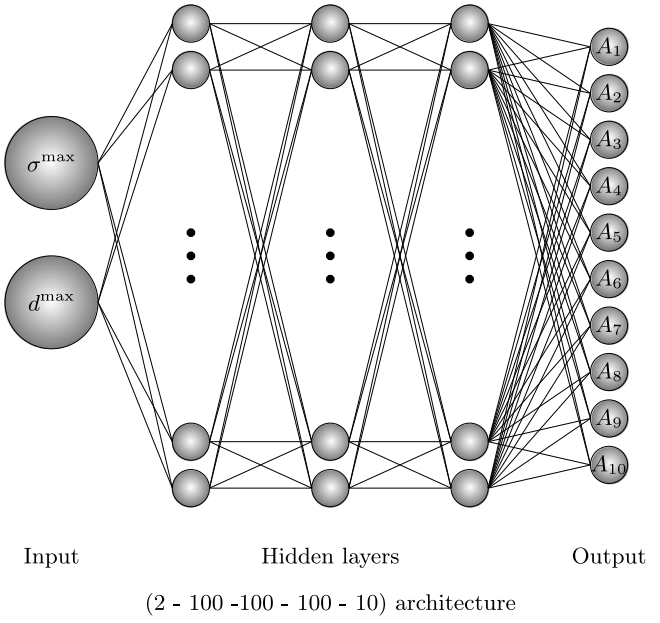

Fig. 4. Neural network architecture for cross-section optimization problem in the 10 – bar plan truss system.

to improve them. Finally, the neural networks were claimed to have no more potential for improvements and, therefore, forgotten. After that, they still remained overlooked for approximately 20 years. DL came to play in the middle of the first decade of 21st century. DL is a machine learning technique that employs deep neural networks containing two or more hidden layers. The vanishing gradient problem is minimized significantly as a result of the RELU activation function and cross entropy-driven learning regulation. The advanced gradient descent technique's usage brings many merits. Since the deep neural network are easily at risk to be over-fitting, deep learning unravel this difficulty by using dropout or regularization.

2.2. A 10 – bar plan truss system

We come up an illustrative example of a 10 – bar plan truss system as displayed in Fig. 2. The researchers investigated this structure for the construction of neural network architectures in [20,21]. The 10 – bar plan truss system has ten cross sectional areas which are input layer units. A single neural network is employed with (10 – 2) architecture in Fig. 3 for FEM problem whose output are displacements. In addition, the optimization problem whose output are cross sectional areas will be investigated by DL in different situation. The former employs stochastic gradient descent (SGD) with Nesterov accelerated gradient (SGDN) (2 – 100 – 10) architecture whereas the latter uses Adam (2 – 100 – 100 – 100 – 10) described in Fig. 4.

This (10 – 2) architecture has two units which are the maximum vertical displacements of node numbers d_2 and d_4 at its output layer. Meanwhile, the optimization problem of (2 – 100 – 10) and (2 – 100 – 100 – 100 – 10) architectures used optimized cross sectional areas as output units. Minimizing the entire mass of the structure and facing up to the restraints at the same time is considered a problem of the unprejudiced of the 10–bar plan truss system in the optimization. Based on purposes of applying the neural networks, we use a simple model to describe. A common question from users is how to obtain a result of optimized cross sections with respect to the total weight under given limitations of displacements and stresses over all nodes and all bars. Model of the problem is defined as:

$$\begin{aligned} &\text{Minimize} \quad \text{Weight}(\boldsymbol{A}) = \textstyle\sum_{i=1}^{10} \rho_i l_i A_i \\ &\text{subject to} \quad \begin{cases} |d_j| & \le d^{\max} \quad \forall j \quad = 1, \ldots, 8 \\ |\sigma_e| & \le \sigma^{\max} \quad \forall e \quad = 1, \ldots, 10 \end{cases} \end{aligned} \tag{1}$$

Hence, for each input of the limitation of displacements $d^{\max}$ and stresses $\sigma^{\max}$, the optimization process is executed to achieve one output of cross sections. Then, DL simplifies the solution using a wide range of inputs and stores them as training data. After training, an optimized solution is derived from an arbitrary new input without repeatedly analyzing the structure.

2.3. Steepest descent algorithm

Optimization is one of the most important problems in engineering applications, especially in civil structures. Let us start by defining a finite element (FE) function whose inputs are cross sections and outputs are displacements or stresses. The FE function is executed by finite element method (FEM) and any other functions which can numerically approximate the FE function are called the *FE activation function*. In the process of FEM, when solving the stiffness system of equations $\boldsymbol{K} \cdot \boldsymbol{U} = \boldsymbol{F}$, the inverse matrix $\boldsymbol{K}^{-1}$ has to be calculated. That makes the process of taking derivatives or optimization difficult. A need for simpler solution leads to FE activation function. There are many numerical methods to solve the stiffness system of equations. In this study, we adopt the SD algorithm as an introduction. In the next sections, the algorithm will be improved to solve deep learning problems effectively. The FEM for linear problems starts from the weak form of the Poisson's equation and, in truss structures, often meets Dirichlet boundary conditions. The symmetric bilinear functional which defines the stiffness matrix leads to a symmetric matrix. The important property here is the Poincaré's inequality which gives the positive definite for the stiffness matrix $\boldsymbol{K}$. Therefore, $\boldsymbol{K}$ is a symmetric and positive definite (*SPD*) matrix. Let $\boldsymbol{b}$ be a vector and $Q(x)$ be the quadratic functional defined by

$$Q(x) = \frac{1}{2}\boldsymbol{x}^T \cdot \boldsymbol{K} \cdot \boldsymbol{x} - \boldsymbol{x}^T \cdot \boldsymbol{b} \tag{2}$$

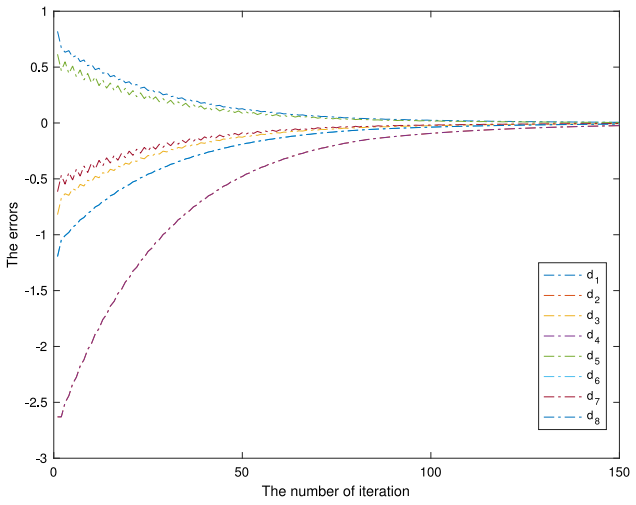

Fig. 5. The displacement errors between SG algorithm and inverse stiffness matrix solution in the FEM problem for the 10 – bar plan truss system.

And then $\boldsymbol{K}\cdot\bar{\boldsymbol{x}}=\boldsymbol{b}\Longleftrightarrow Q(\bar{\boldsymbol{x}})<Q(\boldsymbol{x})$ for all $\boldsymbol{x}\neq\bar{\boldsymbol{x}}$. That means the solution of a linear system $\boldsymbol{K}\cdot\boldsymbol{x}=\boldsymbol{b}$ with *SPD* – matrix can be found by minimizing the quadratic functional $Q(\boldsymbol{x})$. Achieving this utilizes *gradient method* or *method of steepest descent*, also called *method of gradient descent*. The key idea behind this method is to start at some point $\boldsymbol{x}_0$, and find the direction of the steepest descent of the values of $Q(\boldsymbol{x})$ and move in that direction as long as the value of $Q(\boldsymbol{x})$ descends. The direction of the steepest descent is opposite to its gradient at that point. The process repeats until the error between the last two steps is acceptable. From $grad(Q)=\boldsymbol{K}\cdot\boldsymbol{x}-\boldsymbol{b}$, the direction opposite to the gradient of $Q(\boldsymbol{x})$ is equal to the residual $\boldsymbol{r}=\boldsymbol{b}-\boldsymbol{K}\cdot\boldsymbol{x}$ of the system $\boldsymbol{K}\cdot\boldsymbol{x}=\boldsymbol{b}$.

We can use the exact *line search* for a quadratic function. Assume that a point $\boldsymbol{x}_0\in\mathbb{R}^n$ and a vector $\boldsymbol{v}\in\mathbb{R}^n$ are given. Then the equation $\boldsymbol{x}=\boldsymbol{x}_0+\eta\boldsymbol{v}$, $\alpha\in\mathbb{R}$, represents a line going through the point $\boldsymbol{x}_0$ in the direction of $\boldsymbol{v}$. The η is called *learning rate*. Therefore, to solve the problem $\boldsymbol{K}\cdot\boldsymbol{x}=\boldsymbol{b}$, we just need to find the minimum of the functional $Q(\boldsymbol{x})$ on that line, which also means to find the minimum of the function $f(\eta)=Q(\boldsymbol{x}_0+\eta\boldsymbol{v})$ of one real variable η:

$$\frac{\partial f(\eta)}{\partial\eta}=\boldsymbol{v}^T\cdot\boldsymbol{K}\cdot\boldsymbol{x}_0+\eta\boldsymbol{v}\cdot\boldsymbol{K}\cdot\boldsymbol{v}-\boldsymbol{v}^T\cdot\boldsymbol{b}=0\Leftrightarrow\eta=\frac{\boldsymbol{v}^T\cdot(\boldsymbol{b}-\boldsymbol{K}\cdot\boldsymbol{x}_0)}{\boldsymbol{v}^T\cdot\boldsymbol{K}\cdot\boldsymbol{v}}\quad(3)$$

Algorithm 1 summarizes the steepest descent while Fig. 5 illustrates the convergence for the 10– bar plan truss system.

Algorithm 1 The steepest descent algorithm:

Initial Point Choose $\boldsymbol{x}_0$. Let $k=0$.
Residual $\boldsymbol{r}_k=\boldsymbol{b}-\boldsymbol{K}\cdot\boldsymbol{x}_k$.
Stop Condition Check if $\|\boldsymbol{r}_k\|<\varepsilon$ for a chosen error ε, then STOP.
Learning Rate $\eta_k=\dfrac{\boldsymbol{r}_k^T\cdot\boldsymbol{r}_k}{\boldsymbol{r}_k^T\cdot\boldsymbol{K}\cdot\boldsymbol{r}_k}$.
Updated Point $\boldsymbol{x}_{k+1}=\boldsymbol{x}_k+\eta_k\boldsymbol{r}_k$.

2.4. Training data

The training data will be collected by FEM based on the (10 – 2) architecture and then we optimize the total weight with different limitations based on the (2 – 100 – 100 – 100 – 10) architecture. The material density is $\rho=0.1\ lb/in^3$ and the Young's modulus is $E=10^4$ kilo–pounds per square inch (ksi). For each input, the vector of output is formed by $[d_2,d_4]$. Therefore, the matrix of output is a 500×2 matrix. For the first one, single-layer neural networks, the inputs are cross sections $\boldsymbol{A}=[A_1,A_2,\ldots,A_{10}]$ where $1.62\leq A_i\leq33.50\ (in^2)$ which are randomly generated 500 times. Practically, creating input is more difficult. We need to randomly create cross sections so that they can cover almost any case in [1.62, 33.50]. One of the best techniques is to randomly generate a vector of cross sections $\boldsymbol{A}=[A_1,A_2,\ldots,A_n]$ where $1.62\leq A_i\leq33.50\ (in^2)$ so that the sum of all elements in $\boldsymbol{A}$ is a constant, $\sum_{i=1}^{n}A_i=C$. We just need to control the constant C from the lower bound $1.62n$ to the upper bound $33.50n$. To achieve this, firstly, we create a random vector $\boldsymbol{q}$ in (0, 1) which is a common random function in most programming. Secondly, a convex combination transforms from (0, 1) to (a,b) where $b>a\in\mathbb{R}$ are arbitrary numbers, $\boldsymbol{r}=a\cdot(\boldsymbol{1}-\boldsymbol{s})+b\cdot\boldsymbol{s}$. As for the next step, $sumR=\sum_{i=1}^{n}r_i$ and we use it for a condition. If $sumR=C$, $\boldsymbol{A}=\boldsymbol{r}$. If $sumR>C$ or $sumR<C$ then we need to reduce or increase the scale of the vector $\boldsymbol{r}$ by a factor λ so that $sumR=C$, respectively. The algorithm is illustrated in Algorithm 2.

Algorithm 2 Constant sum technique

$(0,1)$ ***random*** Random a vector with n elements $\boldsymbol{s}=rand(n,1)$.
(a,b) ***random*** Convex Combination $\boldsymbol{r}=a\cdot(\boldsymbol{1}-\boldsymbol{s})+b\cdot\boldsymbol{s}$.
Sum r Set $sumR=\sum_{i=1}^{n}r_i$.
Condition if $sumR=C$, return $\boldsymbol{A}=\boldsymbol{r}$
else if $sumR>C$
go to Reduce Scale
else
go to Increase Scale
end if
Reduce Scale Set $\lambda=\dfrac{C-n\cdot a}{sumR-n\cdot a}$
Return $\boldsymbol{A}=(1-\lambda)\cdot\boldsymbol{a}+\lambda\cdot\boldsymbol{r}$
Increase Scale Set $\lambda=\dfrac{C-sumR}{n\cdot b-sumR}$
Return $\boldsymbol{A}=(1-\lambda)\cdot\boldsymbol{r}+\lambda\cdot\boldsymbol{b}$

Hence, we can create a 500-linear data of the constant C in [16.2, 335.0], equivalent to 500 random input vectors of cross sections and store input, and output as training data. The displacements range values between the minimum lines and maximum lines which are illustrated in Fig. 6.

Secondly, for the optimization problem in this example, we use 1000 input data for training and 10 input data for testing where the range of input limitations for displacements is [1.5, 10] (in) and the input limitations for stresses is [5, 100] (ksi). The constant sum technique is used for the two parameters. Fig. 7 illustrates the output cross sections.

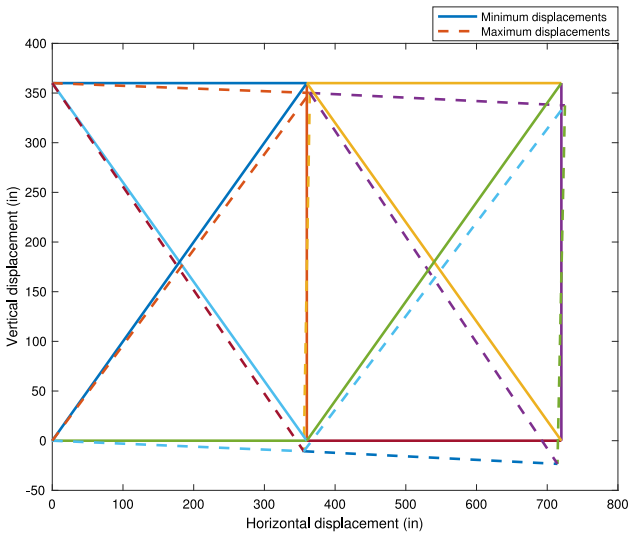

Fig. 6. Minimum and maximum displacement illustration for the 10 – bar plan truss system.

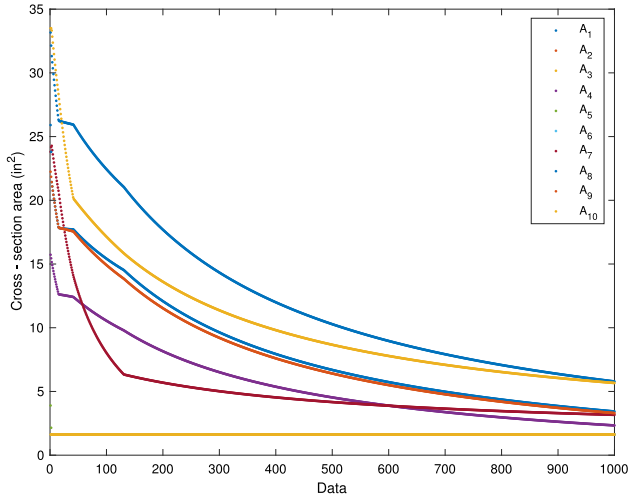

Fig. 7. The optimized cross sections according to a wide range data of maximum displacements and maximum stresses in the optimization problem of the 10 – bar plan truss system.

2.5. The FE activation function

The SD suffers difficulty in plotting the route through ravines, i.e., regions where exterior curves are much sharper in one dimension than in another. We need an extension to reduce the risk of getting stuck in a local minimum. The *momentum method* is a solution which uses a momentum term in analogy to the mass of Newtonian particles that move through a viscous medium in a conservative force field. From the SD, the update in the form $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \eta \nabla_{\boldsymbol{x}} f(\boldsymbol{x}_k)$. We now get another update

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k - \boldsymbol{v}_k \\ \boldsymbol{v}_k &= \gamma \boldsymbol{v}_{k-1} + \eta \nabla_{\boldsymbol{x}} f(\boldsymbol{x}_k) \end{aligned} \tag{4}$$

where $\gamma \approx 0.9$. This method is an extension of the back propagation algorithm which trains artificial neural networks. However, the momentum method is based on a phenomenon where a ball rolling downhill blindly follows the slope, which is highly unsatisfactory. Our intention is to possess a smarter sphere that alerts itself to coming destinations in order to have the speed diminished before the hill inclines upward. *Nesterov accelerated gradient* (NAG) is a special way which allows our momentum term to take this type of precognition. The changing of momentum is the sum of two vectors, momentum vector and gradient vector at the current step. The changing of Nesterov momentum is the sum of two vectors, momentum vector and gradient vector at the approximation of the next step. Therefore, we have the updated formulas:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k - \boldsymbol{v}_k \\ \boldsymbol{v}_k &= \gamma \boldsymbol{v}_{k-1} + \eta \nabla_{\boldsymbol{x}} f(\boldsymbol{x}_k - \gamma \boldsymbol{v}_{k-1}) \end{aligned} \tag{5}$$

Stochastic gradient descent (often shortened to SGD), also known as incremental gradient descent, is a stochastic approximation of the gradient descent optimization and iterative method for minimizing an objective function that is written as a sum of differentiable functions. In other words, SGD looks for minima or maxima by iteration. At step k, we compute derivative of loss function based on solely one data point, and then update $\boldsymbol{x}$ by this derivative with NAG. Each computation for all data points is called one *epoch*. The objective function is a loss function which comes from an approximate function called *activation function* and the training data. Table 2 shows the list of popular activation functions. We now consider that the function $f_{FEM} : \mathbb{R}^{10} \longrightarrow \mathbb{R}^2$ is defined by finite element method $f_{FEM}(\boldsymbol{A}) = \boldsymbol{d} = [d_2, d_4]$ where the cross-sections are $\boldsymbol{A} = [A_1, \ldots, A_{10}]$. The first activation functions are

- The logistic function $f(z) = \dfrac{1}{1+e^{-z}}$.
- The softplus function $f(z) = \ln(1+e^z)$.

The learning rate $\eta = 10^{-4}$ and the values of loss function (the errors between real data and NAG result data) are illustrated in Fig. 8. In other case, the learning rate is $\eta = 10^{-5}$ and activation functions are

- The hyperbolic tangent function $f(z) = tanh(z)$.
- The rectified linear unit $f(z) = \begin{cases} 0, & z < 0 \\ z, & z \geq 0 \end{cases}$
- The identity function $f(z) = z$.

The loss functions are shown in Fig. 9. The errors after 500 epochs are described in Table 3. We can see that the softplus function is the best one to approximate f_{FEM} among the five popular activation functions.

Table 2
Popular activation functions.

Name	Equation	Derivative
Identity $(-\infty, +\infty)$	$f(z) = z$	$f'(z) = 1$
Logistic $(0, 1)$	$f(z) = \dfrac{1}{1+e^{-z}}$	$f'(z) = f(z)[1-f(z)]$
SoftPlus $(0, +\infty)$	$f(z) = \ln(1+e^z)$	$f'(z) = \dfrac{1}{1+e^{-z}}$
TanH $(-1, 1)$	$f(z) = Tanh(z)$	$f'(z) = 1 - f(z)^2$
Rectified Linear Unit (ReLU) $(0, +\infty)$	$f(z) = \begin{cases} 0, & z < 0 \\ z, & z \geq 0 \end{cases}$	$f'(z) = \begin{cases} 0, & z < 0 \\ 1, & z \geq 0 \end{cases}$

Table 3
Errors at 500th epoch for solutions in the 10 – bar plan truss system with various popular activation functions.

Name	ReLU	Identity	TanH	Logistic	SoftPlus
Error	0.0036	0.0023	0.0023	0.0020	0.0019

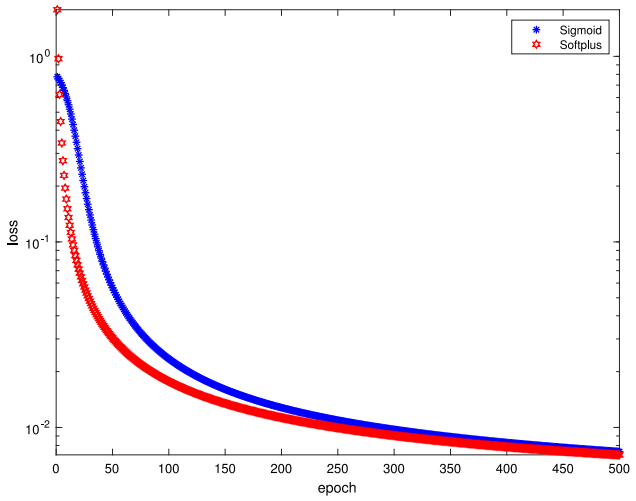

Fig. 8. Loss function for 500 epochs with Logistic and Softplus for solutions of the 10 – bar plan truss system.

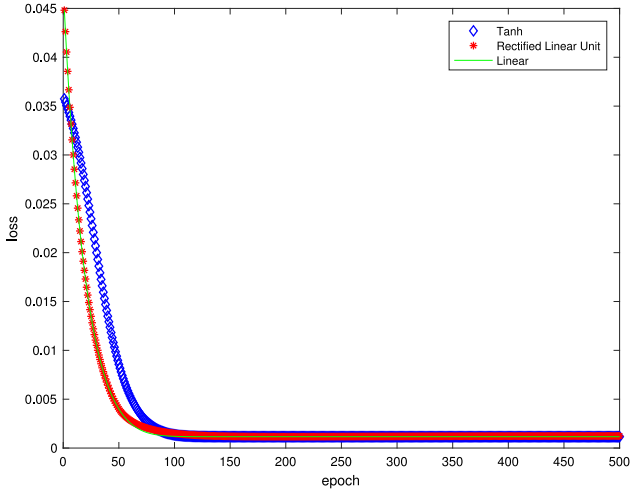

Fig. 9. Loss function for 500 epochs with Tanh, Rectified Linear Unit and Identity for the solutions in the 10 – bar plan truss system.

2.6. A new Chebyshev activation function

In the training data, the change of displacements depends on the sum of all cross sections at the input data – the constant sum. The tendency of the input data is a constant decline from maximum to minimum with respect to the constant sum. The output data are plotted in Fig. 10.

Therefore, the choice of activation functions is really important in this case. We suggest the *Chebyshev polynomial* for the activation function. The original problem comes from the *Chebyshev differential equation* $(1-x^2)y_{xx} - xy_x + n^2y = 0$ where $n = 0, 1, 2, 3, \ldots$. If $x = \cos t$, $y_{tt} + n^2y = 0$ in which general solution $y = A\cos nt + B\sin nt$ or $y = A\cos(n\cos^{-1}x) + B\sin(n\cos^{-1}x)$ or equivalently $y = AT_n(x) + BU_n(x)$ where $|x| < 1$, $T_n(x)$ and $U_n(x)$ are defined as Chebyshev polynomials of the first and the second kind of degree n, respectively. In addition, if $x = \cosh t$, the general solution $y = A\cosh nt + B\sinh nt$ or $y = A\cosh(n\cosh^{-1}x) + B\sinh(n\cosh^{-1}x)$ or equivalently $y = AT_n(x) + BU_n(x)$ where $|x| > 1$. The first six Chebyshev polynomials of the first kind are listed in Table 4.

In a large structure with many bars or in a big training data, the number of epochs and the number of layers are equivalent to waiting time. Since an epoch means one time the algorithm scans for the whole training data, we have to repeat the same process for each bar. Therefore, needed some activation functions must quickly be converged in several first epochs. Chebyshev polynomials are the solution to the algorithm in a very small number of epochs. The order of the polynomials depends on the form of the training data. In the 10 – bar planar truss structure, at (10 – 2) architecture, d_2 and d_4 with second, third and fourth Chebyshev polynomials are implemented and illustrated in Fig. 11. The method SGD and NAG, the learning rate $\eta = 10^{-5}$ and $\gamma = 0.9$ are employed in this example.

As observed, the activation functions of Chebyshev polynomials are converged more quickly than others to get a negligible error after only 15 epochs. In comparison with other popular

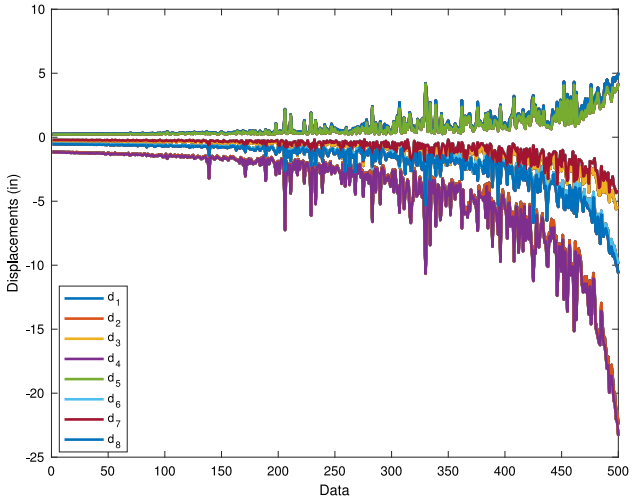

Fig. 10. Profile of output displacements for the 10 – bar plan truss system when input sum of all cross sections are varied.

Table 4
The first six order Chebyshev polynomials.

n	0	1	2
$T_n(x)$	1	x	$2x^2 - 1$
n	3	4	5
$T_n(x)$	$4x^3 - 3x$	$8x^4 - 8x^2 + 1$	$16x^5 - 20x^3 + 5x$

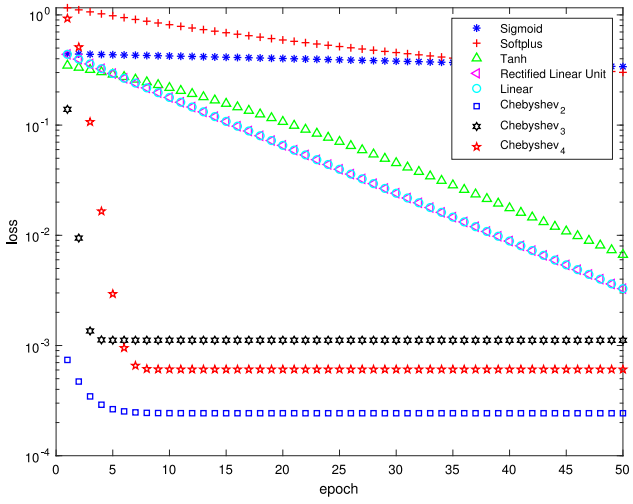

Fig. 11. Loss convergence of solutions in the 10 – bar plan truss system when using single neural network with popular activation functions and Chebyshev polynomial activation functions.

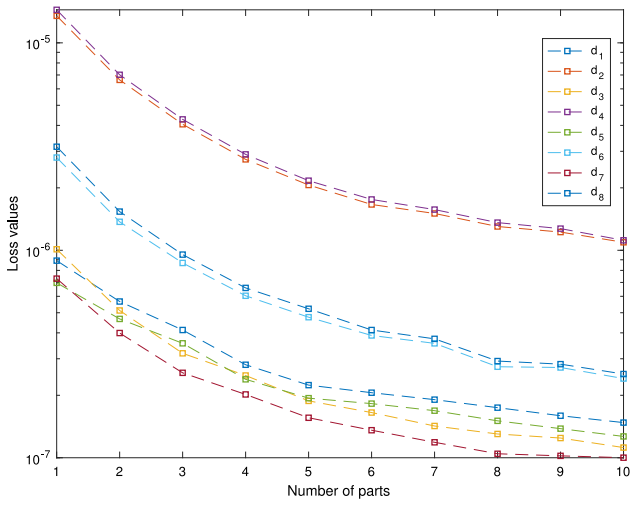

Fig. 12. Loss values of displacements in the 10 – bar plan truss system when using split linear regression based on identity activation function.

activation functions, Chebyshev polynomials can perform better and faster. We can use higher order Chebyshev polynomials. However, it does not mean that the loss function converges more quickly than the lower order. A suitable order is indeed needed for the best convergence.

2.7. Split linear regression

In some specific situations, the training data are very challenging, so that we need to use non-linear regression to deal with and sometimes, the activation functions are in higher order or very "non-linear". It takes a lot of time to train and lead the problem to complexity. In addition, when the data receive another points, updating and training again for the whole data is costly. Therefore, we introduce a new method for these complex data. It is *split linear regression* in which the data are split into many parts. Linear regression approximates each part of the data. This method not only facilitates a more accurate data approximation but also simplifies the update since the data just need to training in one part. The relation between loss values and the number parts after splitting are illustrated in Fig. 12 for the 10 – bar truss structure. As seen, the loss values decrease six times when the data are split into 10 parts at d_2 and d_4.

2.8. Optimization

Gradients of highly complex functions like the neural networks either vanish or explode as the energy is propagated through the function. The effect is cumulative — the more complex the function is, the worse the problem becomes. RMSProp is a clever way to deal with the problem — the unpublished, adaptive learning rate method proposed by Geoff Hinton in Lecture 6e of his Coursera Class. It uses a moving average of squared gradients to normalize the gradient itself. This balances the step size: decreases the step for large gradient to avoid exploding, and increases the step for small gradient to overcome vanishing. Accordingly,

$$\begin{aligned} r_k &= (1-\beta)\cdot[\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_k)]^2 + \beta r_{k-1} \\ \boldsymbol{v}_{k+1} &= \frac{\eta}{\sqrt{r_k}}\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_k) \\ \boldsymbol{x}_{k+1} &= \boldsymbol{x}_k - \boldsymbol{v}_{k+1} \end{aligned} \tag{6}$$

Table 5
Error comparison between optimizers for the cross-sectional optimization problem in the 10 – bar plan truss system.

Optimizer	SGD + NAG	RMSProp	Adam
RMSE	2.63	0.12	$4.19 \cdot 10^{-2}$
Loss	12.43	8.71	5.23

We also divide the learning rate by an exponentially decaying average of squared gradients. Hinton et al. [1] suggested a possible β being around 0.9, whereas a good value for learning rate η is 0.001. Adaptive Moment Estimation, (Adam) [22], is another method that computes adaptive learning rates for each parameter. In this method, the learning rate was divided by an average decayed exponentiation of squared gradients. Adam also kept an exponentially decaying average of past gradients $\boldsymbol{v}_k$, similar to momentum. While momentum could be noticed as a ball going its way down a steep gradient, with friction favoring plane minima in the error superficiality, Adam acted as a weighty ball. We calculate the average deteriorated of past and past squared gradients $\boldsymbol{v}_k$ and r_k respectively as follows

$$\begin{aligned} \boldsymbol{v}_k &= \beta_1 \boldsymbol{v}_{k-1} + (1-\beta_1)\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_k) \\ r_k &= \beta_2 r_{k-1} + (1-\beta_2)\cdot[\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_k)]^2 \end{aligned} \tag{7}$$

$\boldsymbol{v}_k$ and r_k are estimates of the first moment (the mean) and second moment (the uncentered variance) of the gradients, respectively. As $\boldsymbol{v}_k$ and r_k are initialized as vectors of 0's, the authors of Adam observed that they are predisposed towards zero, especially during the initial time steps and when the decay rates are small (i.e. β_1 and β_2 are close to 1). These biases have been estimated by the authors based on a correction:

$$\begin{aligned} \boldsymbol{v}_k &= \frac{\boldsymbol{v}_k}{1-\beta_1^t} \\ r_k &= \frac{r_k}{1-\beta_2^t} \end{aligned} \tag{8}$$

They then use these to renew the parameters which yields the Adam update rule:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \frac{\eta}{\sqrt{r_k}+\epsilon}\boldsymbol{v}_k \tag{9}$$

In the next step, the choice of activation functions is the most important thing regarding accuracy. We used SGD with NAG, RMSProp, Adam optimizers and Relu activation function f to approximate the optimized function ($f_{opt} : \mathbb{R}^2 \rightarrow \mathbb{R}^{10}$) with respect to z as variables $z = \boldsymbol{\omega}_i^m \cdot \boldsymbol{l}$ where m is the order of the layer and i is the order of nodes in that layer illustrated in Fig. 13. In this 10 bars problem, the learning rate $\eta_{SGD+NAG} = 10^{-8}$, $\eta_{RMSProp} = 2 \cdot 10^{-3}$, $\eta_{Adam} = 5 \cdot 10^{-3}$ and $\gamma = 0.9$. The learning rate drop period was 100 epochs. The learning rate drop factor was 95% for SGD + NAG and RMSProp, 85% for Adam. The squared gradient decay factor was 0.99. The gradient decay factor was 0.95. The size of the mini-batch is 10 and the total of epochs was 5,000. The contribution of momentum parameter from the previous step was 0.95. The loss values and root mean square errors (RMSE) are plotted in Fig. 14. The testing values of cross sections by DL after training and by data for all of three optimizers are illustrated in Fig. 15. The neural network architecture for this problem was (2 – 100 – 100 – 100 – 10). The evaluation of error using optimizers is displayed in Table 5.

3. Numerical validation

3.1. A 52 – bar planar truss structure

One of the most popular planar truss structures is the 52 bars as illustrated in Fig. 16. The truss structure was investigated for

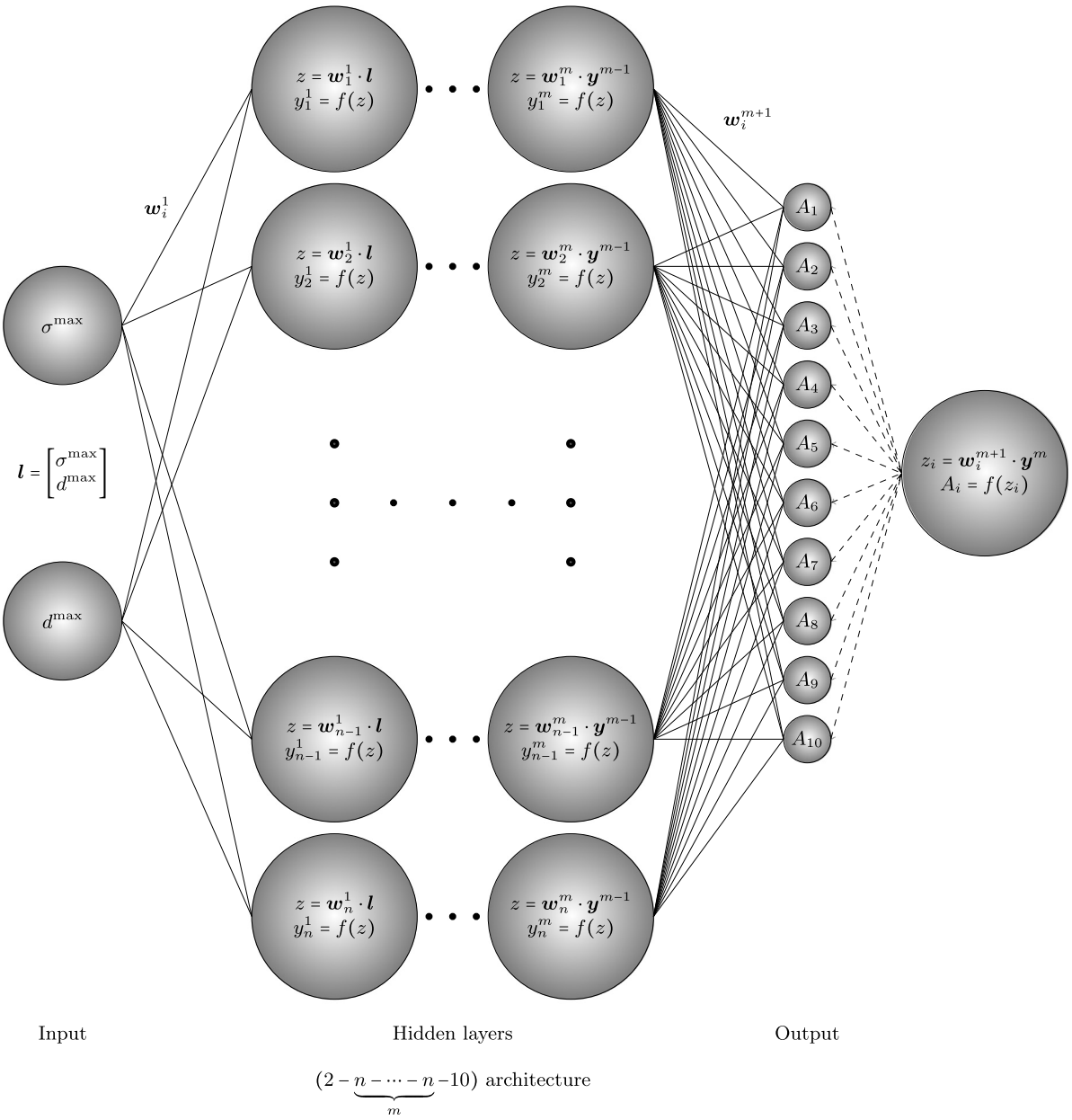

Fig. 13. Activation function role in deep neural network for the cross-sectional optimization problem in the 10 – bar plan truss system.

the application of neural network approximation. This problem was previously solved by Lee et al. [23], Li et al. [24], Sadollah et al. [25,26], Kaveh and Mahdavi [27], etc. The Young's modulus is $E = 207$ (GPa) and the mass density is $\rho = 7860$ (kg/m^3). The vertical loads were set as $P_x = 100$ (kN) and $P_y = 200$ (kN). The members of this structure were divided into 12 groups: (1) A_1 – A_4, (2) A_5 – A_{10}, (3) A_{11} – A_{13}, (4) A_{14} – A_{17}, (5) A_{18} – A_{23}, (6) A_{24} – A_{26}, (7) A_{27} – A_{30}, (8) A_{31} – A_{36}, (9) A_{37} – A_{39}, (10) A_{40} – A_{43}, (11) A_{44} – A_{49}, and (12) A_{50} – A_{52} corresponding to 12 design variables. The design variables were selected from the interval $[\min_A, \max_A]$ where $\min_A = 0.111$ and $\max_A = 33.5$ (in^2).

Firstly, for single-layer neural networks, FEM was used to collect the training data based on the (12 – 32) architecture whose input of cross sections and output of displacements were generated randomly. The input was a running vector of cross sections $\boldsymbol{A} = [A_{(1)}, A_{(2)}, \ldots, A_{(12)}]$ where $0.111 \leq A_{(i)} \leq 33.50$ (in^2) was generated 500 times, randomly. For each input, the vector of output was formed $[d_9, \ldots, d_{40}]$, and therefore, the matrix of output was a 500×32 matrix. The technique of constant sum was used for the input generating of cross sections where the tendency of the data was a constant decline from maximum to minimum with respect to the constant sum. The inputs are depicted in Fig. 17. The input data contained 500 vectors of cross sections ranging from maximum to minimum with respect to the constant sum. After the data was trained, the FE activation function was created and executed to compare the results with popular activation functions concerning loss values. In the SGD with NAG, the learning rate $\eta = 10^{-5}$ and $\gamma = 0.9$ are used. In this example, the displacements were very large and, therefore, the data were scaled by 2-norm to avoid some overloaded numbers in gradient steps or vanishing gradient. The loss values are shown in Fig. 18. The Chebyshev polynomials exhibited the fastest convergence in comparison with other existing activation functions. At the first 30 epochs, the second Chebyshev polynomial retained the smallest error. The appropriate order depends on the form of the output.

Secondly, we consider the problem under the split linear regression method. The computation was illustrated at the top

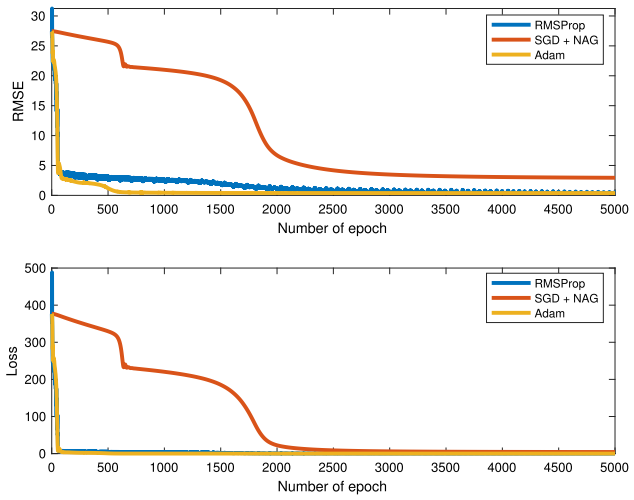

Fig. 14. Loss values and RMSE in the training process for the cross-sectional optimization problem in the 10 – bar plan truss system.

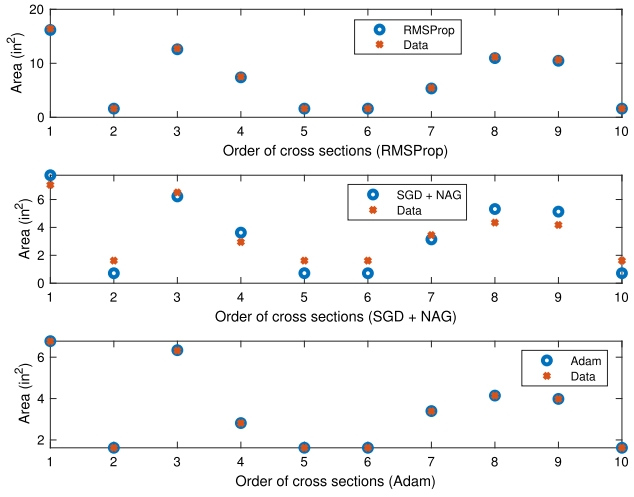

Fig. 15. Optimized cross sections after training and testing for the cross-sectional optimization problem in the 10 – bar plan truss system.

eight node in x-direction. The loss values were investigated from linear regression (1 – split part) to 10 – split linear regression and are shown in Fig. 19. As expected, the loss values declined considerably when the data was more complex such as u^x_{13}, u^x_{16}, u^x_{17} and u^x_{18}. In contrast, they gradually went down when the data was close to linear.

Finally, the authors studied the optimization problem with the same method. Assumed that all truss members were subject to a displacement limitation which varied from 0.1 to 0.8 (m) in both horizontal and vertical directions. Minimizing the entire mass of the structure and facing up to the restraints at the same time is a problem of the unprejudiced of the 52–bar truss. The results of optimized cross sections for this structure were stored in the training data in the form of a 1000 vectors or a 1000×12 matrix and are illustrated in Fig. 20. The input was created as an increasing vector of displacement limitations. The model of the problem is described as

$$\begin{aligned} &\text{Minimize} \quad \text{Weight}(\boldsymbol{A}) = \textstyle\sum_{i=1}^{52} \rho_i l_i A_i \\ &\text{subject to} \quad \left|d_j\right| \le d^{\max} \end{aligned} \tag{10}$$

where $j = 9, \ldots, 40$. We used SGD with NAG, RMSProp, Adam optimizers and Relu activation function f to approximate the

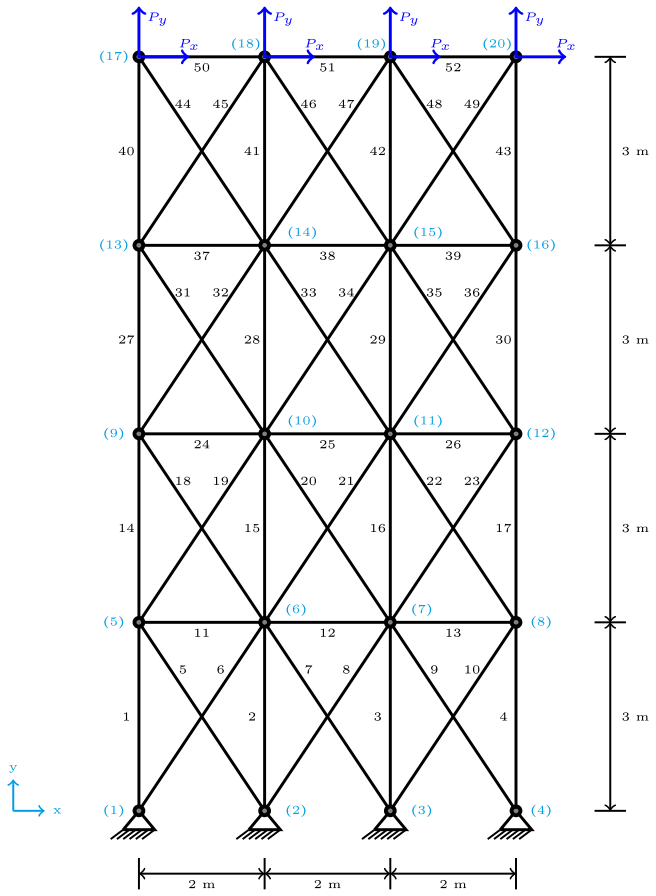

Fig. 16. A 52 – bar planar truss structure.

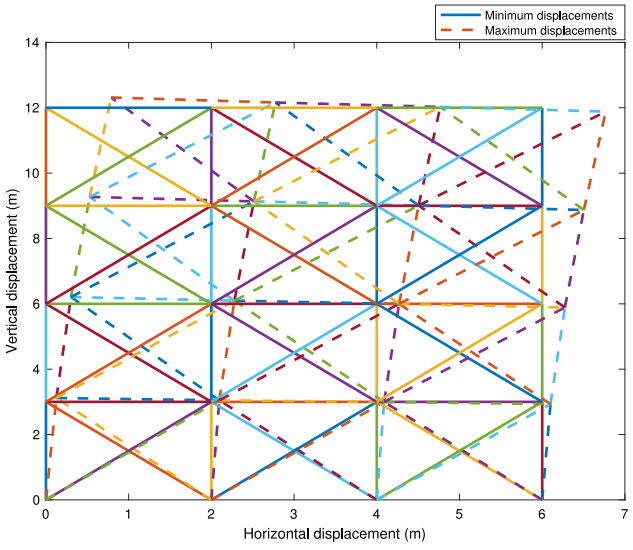

Fig. 17. Minimum and maximum displacement illustration for the 52 – bar planar truss structure.

optimized function $f_{opt} : \mathbb{R} \to \mathbb{R}^{12}$. In this 52 bars problem, the learning rate $\eta_{SGD+NAG} = 10^{-7}$, $\eta_{RMSProp} = 3 \cdot 10^{-5}$, $\eta_{Adam} = 9 \cdot 10^{-5}$ and $\gamma = 0.9$ are used. The learning rate drop period was 100 epochs. The learning rate drop factor was 85% for Adam and RMSProp, 95% for SGD + NAG. The squared gradient decay factor was 0.99. The gradient decay factor was 0.95. The size of the mini-batch was 10 and the total of epochs was 5,000. The contribution

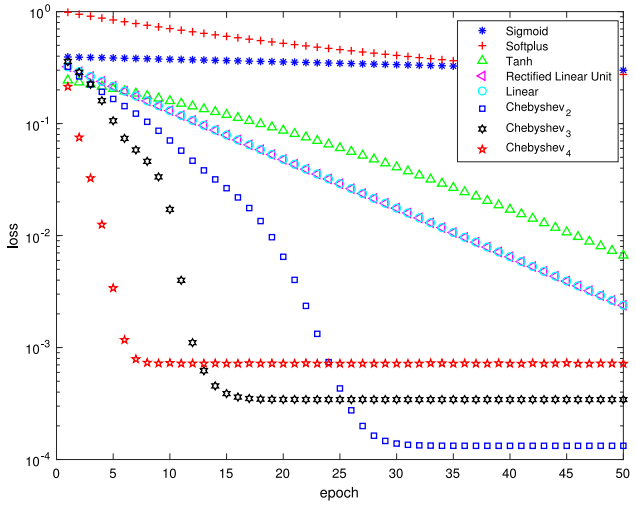

Fig. 18. Loss convergence of solutions in the 52 – bar planar truss structure when using Single Neural Network with popular activation and Chebyshev polynomial activation functions.

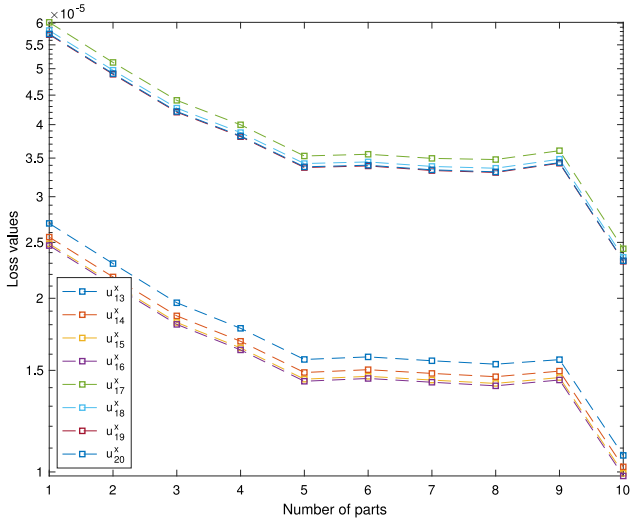

Fig. 19. Loss values of displacements for the 52 – bar planar truss structure when using split linear regression based on identity activation function.

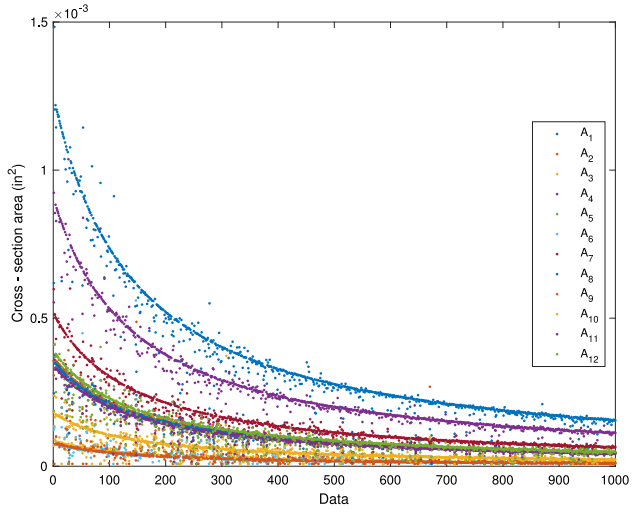

Fig. 20. The optimized cross sections according to a wide range data of displacement limitations in the optimization problem of the 52 – bar planar truss structure.

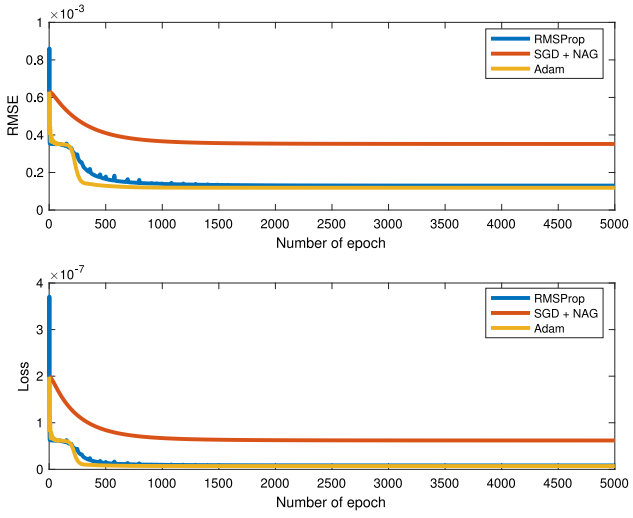

Fig. 21. Loss values and RMSE in the training process for the cross-sectional optimization problem in the 52 – bar planar truss structure using RMSProp, SGD with NAG and Adam methods.

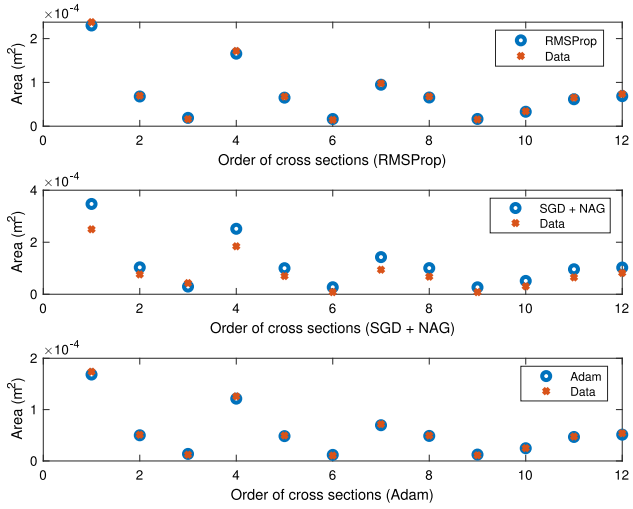

Fig. 22. Optimized cross sections after training and testing for the cross-sectional optimization problem in the 52 – bar planar truss structure.

Table 6
Error comparison between optimizers for the cross-sectional optimization problem in the 52 – bar planar truss structure.

Optimizer	SGD + NAG	RMSProp	Adam
RMSE	$6.52 \cdot 10^{-4}$	$1.3 \cdot 10^{-4}$	$1.19 \cdot 10^{-4}$
Loss	$6.2 \cdot 10^{-8}$	$8.5 \cdot 10^{-9}$	$7.0 \cdot 10^{-9}$

of momentum parameter from the previous step was 0.95. The loss values and root mean square errors (RMSE) are plotted in Fig. 21. The testing values of cross sections by DL after train and by data for all of three optimizers are illustrated in Fig. 22. The neural network architecture for this problem was (2 – 100 – 100 – 100 – 10). Results of optimizers are summarized in Table 6.

3.2. A 200 – bar planar truss structure

The second problem is a 200–bar truss structure as shown in Fig. 23. This example was previously investigated by Togan and Daloglu [28], Talebpour et al. [29], and Azad and Hasancebi [30]. The material density and Young's modulus read $\rho = 0.283$

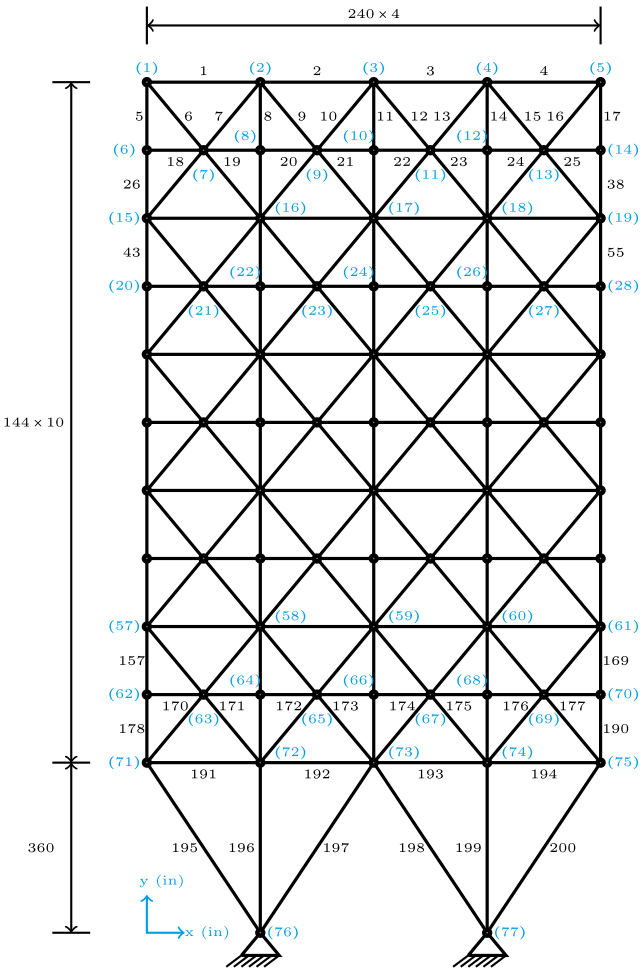

Fig. 23. A 200 – bar planar truss structure.

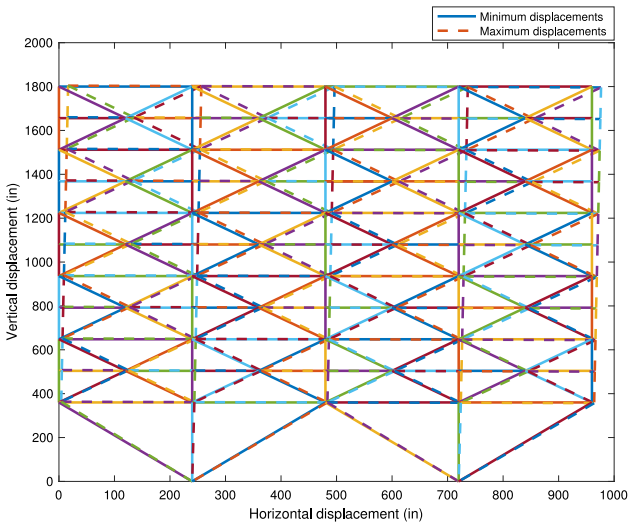

Fig. 24. Minimum and maximum displacement illustration for the 200 – bar planar truss structure.

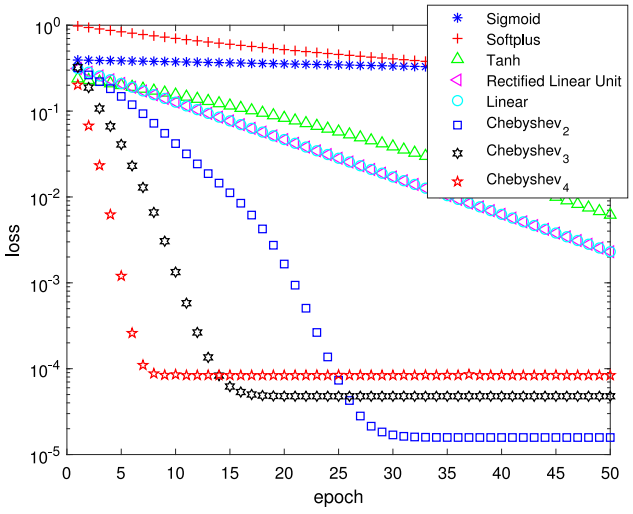

Fig. 25. Loss convergence of the displacements for the 200 – bar planar truss structure when using single neural network with popular activation and Chebyshev polynomial activation functions.

(lb/in^3) and $E = 3 \cdot 10^4$ (ksi), respectively. These 200 bars were categorized into 29 groups according to 29 design variables. This system was supported by three loading conditions: case #1: 1 (kip) acting in the positive x direction at nodes 1, 6, 15, 20, 29, 34, 43, 48, 57, 62, and 71; case #2: 10 (kips) acting in the negative y direction at nodes 1, 2, 3, 4, 5, 6, 8, 10, 12, 14, 15, 16, 17, 18, 19, 20, 22, 24, 26, 28, 29, 30, 31, 32, 33, 34, 36, 38, 40, 42, 43, 44, 45, 46, 47, 48, 50, 52, 54, 56, 57, 58, 59, 60, 61, 62, 64, 66, 68, 70, 71, 72, 73, 74, and 75; and case #3: combination of #1 and #2. The cross- section areas were in the interval [0.10, 33.70] (in^2). In this example, we investigated the structure in case #3. Hence, we can get the minimum and maximum displacements as shown in Fig. 24.

Firstly, for single-layer neural networks, FEM collected training data based on the (29 – 150) architecture whose inputs were cross sections and outputs were displacements. The inputs of cross sections

$$\boldsymbol{A} = [A_{(1)}, A_{(2)}, \ldots, A_{(29)}]$$

where $0.10 \leq A_{(i)} \leq 33.70$ (in^2) and were generated 500 times, randomly. For each input, the vector of output was formed $[u_1, \ldots, u_{150}]$, and therefore, the output was a 500×150 matrix. The constant sum technique was used for the input generation of cross sections where the direction of data declined continuously from maximum to minimum with respect to the constant sum. In the GD with NAG, the learning rate $\eta = 10^{-5}$ and $\gamma = 0.9$ were used. In this example, the displacements were very large and so, the data were scaled by 2-norm to avoid some overloaded numbers in gradient steps or vanishing gradient. The second, third and fourth Chebyshev polynomials were used for comparison. As indicated in Fig. 25, the third and fourth order Chebyshev polynomials reached the fastest convergence.

Secondly, we considered the problem using split linear regression method. The investigation focused to the top right four node in x-direction. The loss values were computed from linear regression (1 – split part) to 10 – split linear regression and are depicted in Fig. 26. It was seen that the loss values reduced by a half when the data was split into 10 parts.

Finally, the stress limitations of all members were at the interval [0.1, 15] (ksi). The objective of the 200 – bar truss problem was to minimize the total weight of the structure within those constraints. The result of optimized cross sections for this structure is stored in the training data with a form of 1000 vectors or a 1000×29 matrix shown in Fig. 27. The input was an increasing vector of the stress limitation. The model of the

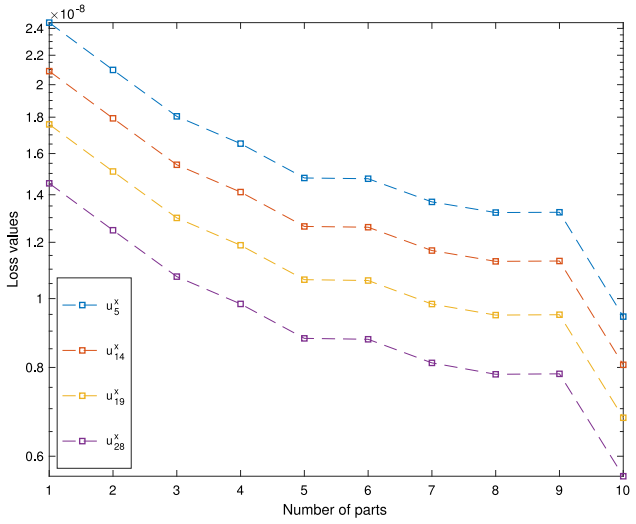

Fig. 26. Loss values of displacements for the 200 – bar planar truss structure when using split linear regression based on identity activation function.

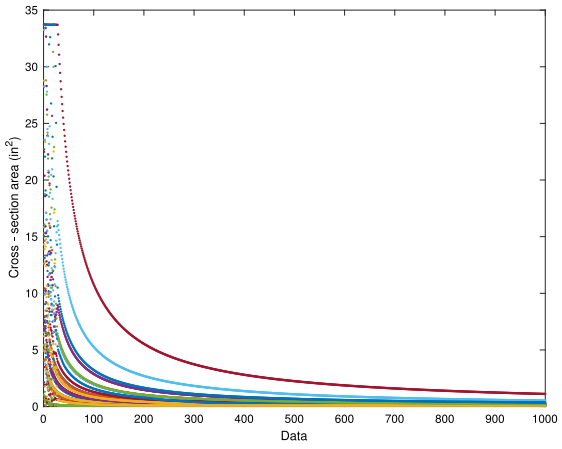

Fig. 27. The optimized cross sections according to a wide range data of stress limitations in the optimization problem of the 200 – bar planar truss structure.

problem is defined as

$$\begin{array}{ll}\text{Minimize} & \text{Weight}(\boldsymbol{A}) = \sum_{i=1}^{200} \rho_i l_i A_i \\ \text{subject to} & |\sigma_k| \le \sigma^{\max}\end{array} \tag{11}$$

where $k = 1, \ldots, 200$. We employed SGD with NAG, RMSProp, Adam optimizers and Relu activation function f to approximate the optimized function $f_{opt} : \mathbb{R} \to \mathbb{R}^{29}$. In this 200–bar problem, the learning rate $\eta_{SGD+NAG} = 10^{-7}$, $\eta_{RMSProp} = 5 \cdot 10^{-4}$, $\eta_{Adam} = 5 \cdot 10^{-4}$ and $\gamma = 0.9$ were used. The learning rate drop period was 100 epochs. The learning rate drop factor was 99% for Adam and RMSProp, 95% for SGD + NAG. The squared gradient decay factor was 0.99. The gradient decay factor was 0.95. The size of the mini-batch was 10 and the number of epochs was 10,000. The contribution of momentum parameter from the previous step was 0.95. The loss values and root mean square errors (RMSE) are plotted in Fig. 28. The testing values of cross sections by DL after training and by data for all of three optimizers are illustrated in Fig. 29. The neural network architecture for this problem was (2 – 100 – 100 – 100 – 10). Comparison in errors between optimizers is given in Table 7. It is seen that the Adam optimizer delivers the best performance.

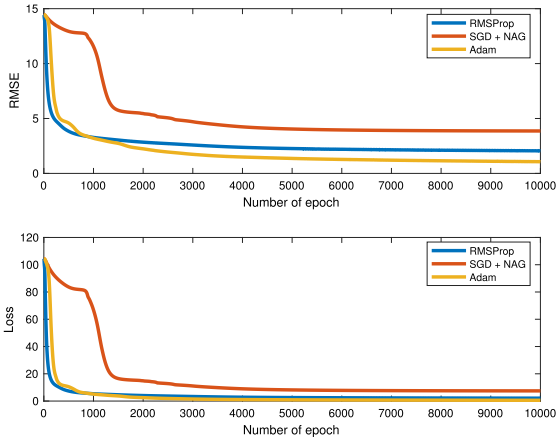

Fig. 28. Loss values and RMSE in the training process for the cross-sectional optimization problem in the 200 – bar planar truss structure using RMSProp, SGD with NAG and Adam methods.

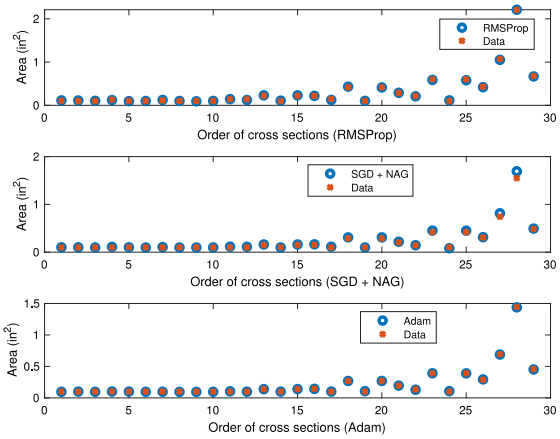

Fig. 29. Optimized cross sections after training and testing for the cross-sectional optimization problem in the 200 – bar planar truss structure.

Table 7
Error comparison between optimizers for the cross-sectional optimization problem in the 200 – bar planar truss structure.

Optimizer	SGD + NAG	RMSProp	Adam
RMSE	3.86	2.05	1.07
Loss	7.5	2.1	0.6

3.3. A 25 – bar space truss structure

The third example considered the problem with a 25 – bar space truss structure as shown in Fig. 30. This issue was formerly studied by Wu and Chow [31], Lee et al. [23], Li et al. [24], Sadollah et al. [25], Kaveh and Ghazaan [32], etc. The Young's modulus is $E = 10^4$ (ksi) and the material density is $\rho = 0.1$ (lb/in^3). The structure included 25 members which were divided into 8 groups: (1) A_1, (2) $A_2 - -A_5$, (3) $A_6 - -A_9$, (4) $A_{10} - -A_{11}$, (5) $A_{12} - -A_{13}$, (6) $A_{14} - -A_{17}$, (7) $A_{18} - -A_{21}$ and (8) $A_{22} - -A_{25}$. The loads are shown in Table 8. The design variables were selected from the interval [0.1, 3.4] (in^2). Hence, we can determine the minimum and maximum displacements of nodes with respect to the set of cross sections as given in Fig. 31.

Firstly, for single-layer neural networks, FEM collected training data based on the (8 – 18) architecture whose inputs were cross sections and outputs were displacements. The inputs were cross sections $\boldsymbol{A} = [A_{(1)}, A_{(2)}, \ldots, A_{(8)}]$ where $0.1 \le A_{(i)} \le 3.4$ (in^2) and

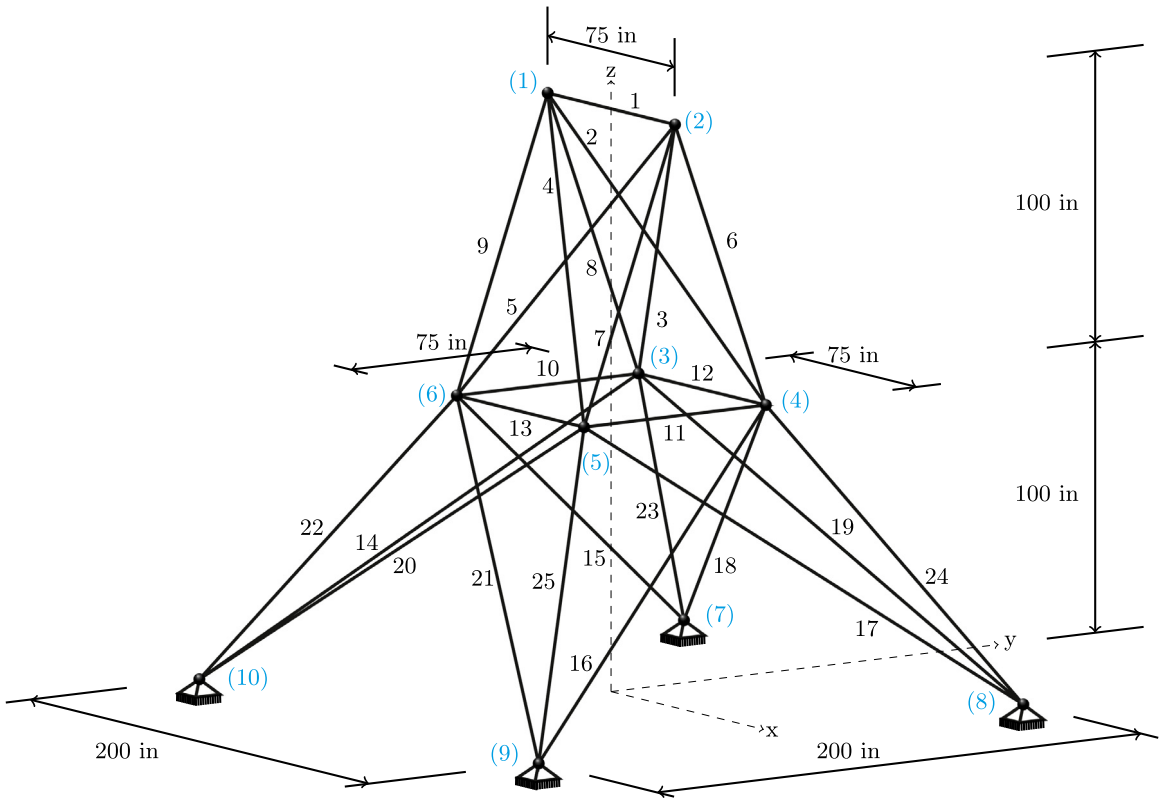

Fig. 30. A 25 – bar space truss structure.

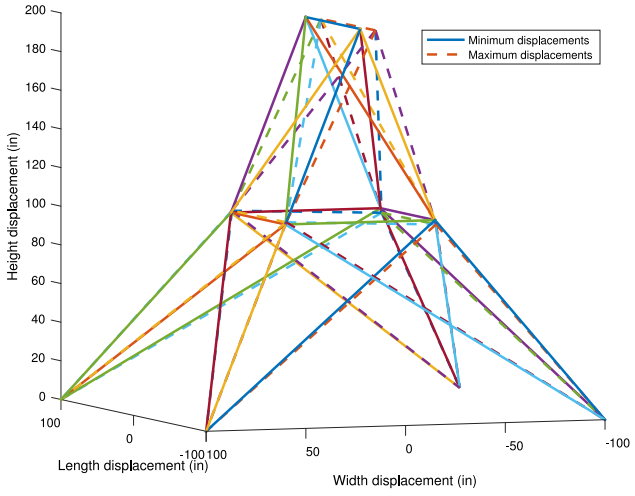

Fig. 31. Minimum and maximum displacement illustration for the 25 – bar space truss structure.

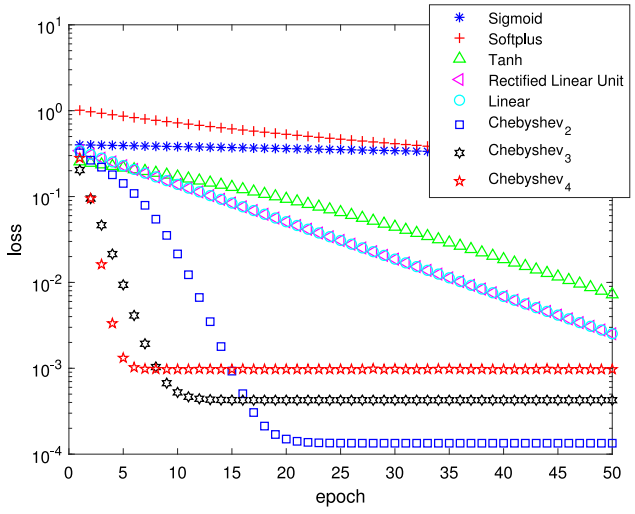

Fig. 32. Loss convergence of the displacements in the 25 – bar space truss structure when using single neural network with popular activation and Chebyshev polynomial activation functions.

were generated 500 times, randomly. For each input, the vector of output was formed $[u_1, \ldots, u_{18}]$, and therefore, the matrix of output was a 500×18 matrix. The constant sum technique was used and the direction of data declined continuously from the maximum to minimum value. In the SGD with NAG, the learning rate $\eta = 10^{-5}$ and $\gamma = 0.9$ were chosen. In this example, the displacements were very large and so, the data were scaled by 2-norm to avoid some overloaded numbers in gradient steps or vanishing gradient. The second, third and fourth order Chebyshev polynomial were studied. In Fig. 32, three Chebyshev polynomials showed the fastest convergence.

Table 8
Loads for the 25 – bar space truss structure.

Node	P_x	P_y	P_z
1	1	−10	−10
2	0	−10	−10
3	0.5	0	0
6	0.6	0	0

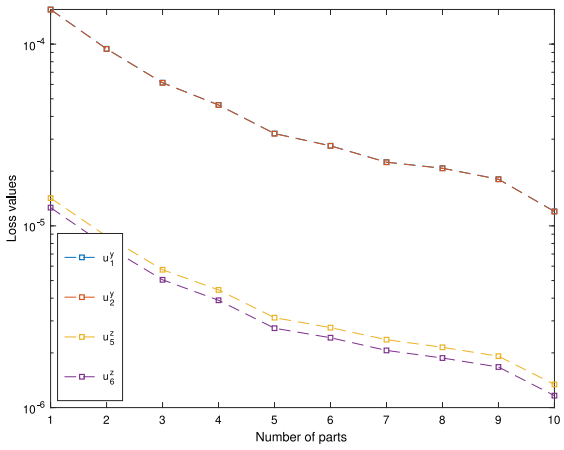

Fig. 33. Loss values of displacements in the 25 – bar space truss structure when using split linear regression based on identity activation function.

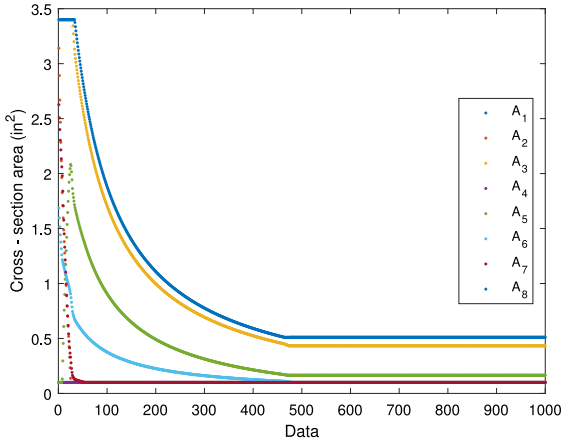

Fig. 34. The optimized cross sections according to a wide range data of displacement limitations in the optimization problem of the 25 – bar space truss structure.

Secondly, we considered the problem using split linear regression method. The computation aimed to node 1, node 2 in y-direction and node 5, node 6 in z-direction. The loss values were investigated from linear regression (1 – split part) to 10 – split linear regression and are displayed in Fig. 33. The loss values considerably declined when the data was more complex such as u_1^y and u_2^y. In contrast, they gradually went down when the data was close to linear as u_5^z and u_6^z.

Finally, we assumed that all nodes were subjected to displacement limitations in the interval $[0.25, 5]$ (in) and in three directions x, y, z. The stress limitations were ± 40 (ksi). The objective of the 25 – bar space truss problem was to minimize the total weight of the structure while meeting those constraints. The result of optimized cross sections given in Fig. 34 was stored in the training data with a form of 1000 vectors or a 1000×8 matrix. The input formed an increasing vector of displacement limitations. The model of the problem is defined as:

$$\begin{aligned} &\text{Minimize} && \text{Weight}(\boldsymbol{A}) = \textstyle\sum_{i=1}^{25} \rho_i l_i A_i \\ &\text{subject to} && \begin{matrix} |u_j| \\ |\sigma_k| \end{matrix} \begin{matrix} \leq u^{\max} \\ \leq \sigma^{\max} \end{matrix} \end{aligned} \tag{12}$$

where $j = 1, \ldots, 18$; $k = 1, \ldots, 25$; $u^{\max} \in [0.25, 5]$ (in) and $\sigma^{\max} = 40$ (ksi). We used SGD with NAG, RMSProp, Adam optimizers and Relu activation function f to approximate the optimized function $f_{opt} : \mathbb{R} \to \mathbb{R}^8$. In this 25 bars space problem,

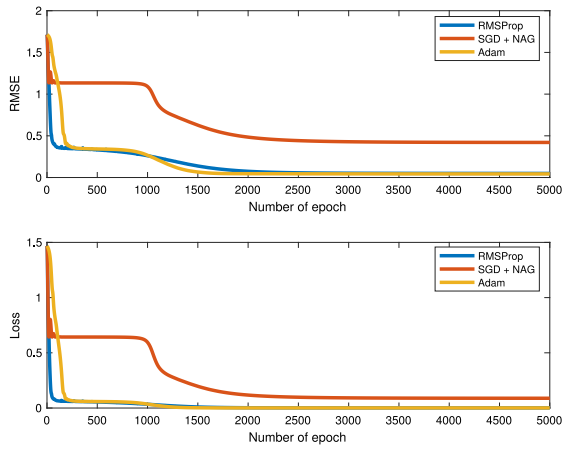

Fig. 35. Loss values and RMSE in the training process for the cross-sectional optimization problem in the 25 – bar space truss structure using RMSProp, SGD with NAG and Adam methods.

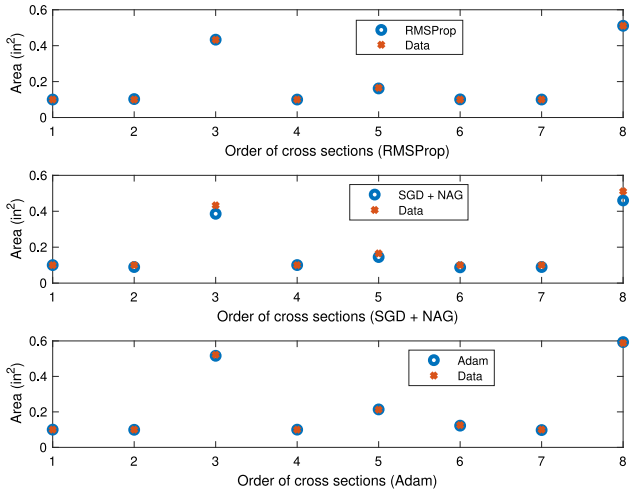

Fig. 36. Optimized cross sections after training and testing for the cross-sectional optimization problem in the 25 – bar space truss structure.

Table 9
Error comparison between optimizers for the cross-sectional optimization problem in the 25 – bar space truss structure.

Optimizer	SGD+NAG	RMSProp	Adam
RMSE	0.42	0.05	0.04
Loss	$8.8 \cdot 10^{-2}$	$1.2 \cdot 10^{-3}$	$9.4 \cdot 10^{-4}$

the learning rate $\eta_{SGD+NAG} = 10^{-5}$, $\eta_{RMSProp} = 5 \cdot 10^{-4}$, $\eta_{Adam} = 5 \cdot 10^{-4}$ and $\gamma = 0.9$ were used. The learning rate drop period was 100 epochs. The learning rate drop factor was 85% for all of optimizers. The squared gradient decay factor was 0.99. The gradient decay factor was 0.95. The size of the mini-batch was 10 and the total of epochs was 5,000. The contribution of the momentum parameter from the previous step was 0.95. The loss values and root mean square errors (RMSE) are plotted in Fig. 35. The testing values of cross sections by DL after training and by data for all three optimizers are illustrated in Fig. 36. The neural network architecture for this problem was (2 – 100 – 100 – 100 – 100 – 10). Finally, the comparison of errors between optimizers is shown in Table 9.

4. Conclusions

This paper addressed an alternative approach in the context of deep learning for computational structural optimization. It can be applied with a wide range of activation functions and optimizers. In the numerical examples, the Chebyshev polynomials played a key role in establishing new activation functions that achieve better solution with short training in single-layer neural networks. In addition, split linear regressions as a new idea which performed well, even better than the single-layer neural networks, for some sensitive data. Finally, the Adam optimizer reproduced the best prediction in comparison with other ones. The important contribution of this study is to integrate DL into computational structural engineering. The key feature is how to choose activation functions, training parameters and deep neural network architectures. Regarding some disadvantages, the present approach required additional time due to generating data using the constant sum technique and arisen some tolerances based on numerical optimized method using finite element approach. Moreover, our research is particularly promising when it comes to the solution for various engineering problems in practice. It follows that an integration of the current work in additive manufacturing or 3D printing is inevitably appealing to a further investigation. Ultimately, some evolutionary algorithms (e.g., population extremal optimization) can be applied with NN or DL not only for the problems but also for a wide range of mechanical systems. A combination of such techniques will be investigated in forthcoming work. Finally, an alternative technique based on balancing composite motion optimization (BCMO) [33] may be good choice to improve the performance of the proposed approach.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

Research is supported by Vingroup Innovation Foundation (VINIF), Viet Nam in project code VINIF.2019.DA04.

References

[1] Hinton GE, Osindero S, Teh YW. A fast learning algorithm for deep belief nets. Neural Comput 2006;18(7):1527–54.
[2] Nair V, Hinton GE. Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10). 2010. pp. 807–814.
[3] Hinton GE, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov RR. Improving neural networks by preventing co-adaptation of feature detectors. 2012, arXiv preprint arXiv:1207.0580.
[4] LeCun Y, Bengio Y, Hinton GE. Deep learning. Nature 2015;521(7553):436.
[5] Adeli H. Neural networks in civil engineering: 1989–2000. Comput-Aided Civ Infrastruct Eng 2001;16(2):126–42.
[6] Du W, Zhang M, Ying W, Perc M, Tang K, Cao X, et al. The networked evolutionary algorithm: A network science perspective. Appl Math Comput 2018;338:33–43.
[7] Angermueller C, Pärnamaa T, Parts L, Stegle O. Deep learning for computational biology. Mol Syst Biol 2016;12(7):878.
[8] Park Y, Kellis M. Deep learning for regulatory genomics. Nature Biotechnol 2015;33(8):825.
[9] Goh GB, Siegel C, Vishnu A, Hodas NO, Baker N. Chemception: A deep neural network with minimal chemistry knowledge matches the performance of expert-developed qsar/qspr models. 2017, arXiv preprint arXiv:1706.06689.
[10] Helbing D, Brockmann D, Chadefaux T, Donnay K, Blanke U, Woolley-Meza O, et al. Saving human lives: What complexity science and information systems can contribute. J Stat Phys 2015;158(3):735–81.
[11] Perc M, Ozer M, Hojnik J. Social and juristic challenges of artificial intelligence. Palgrave Commun 2019;5(1):1–7.
[12] Ravì D, Wong C, Deligianni F, Berthelot M, Andreu-Perez J, Lo B, et al. Deep learning for health informatics. IEEE J Biomed Health Inform 2017;21(1):4–21.
[13] Furukawa T, Yagawa G. Implicit constitutive modelling for viscoplasticity using neural networks. Internat J Numer Methods Engrg 1998;43(2):195–219.
[14] Lefik M, Boso DP, Schrefler BA. Artificial neural networks in numerical modelling of composites. Comput Methods Appl Mech Engrg 2009;198(21–26):1785–804.
[15] Ghaboussi J, Pecknold DA, Zhang M, Haj-Ali RM. Autoprogressive training of neural network constitutive models. Internat J Numer Methods Engrg 1998;42(1):105–26.
[16] Oeser M, Freitag S. Modeling of materials with fading memory using neural networks. Internat J Numer Methods Engrg 2009;78(7):843–62.
[17] Ootao Y, Kawamura R, Tanigawa Y, Imamura R. Optimization of material composition of nonhomogeneous hollow sphere for thermal stress relaxation making use of neural network. Comput Methods Appl Mech Engrg 1999;180(1–2):185–201.
[18] Gawin D, Lefik M, Schrefler BA. ANN approach to sorption hysteresis within a coupled hygro-thermo-mechanical FE analysis. Internat J Numer Methods Engrg 2001;50(2):299–323.
[19] Yun GJ, Ghaboussi J, Elnashai AS. Self-learning simulation method for inverse nonlinear modeling of cyclic behavior of connections. Comput Methods Appl Mech Engrg 2008;197(33–40):2836–57.
[20] Lee S, Ha J, Zokhirova M, Moon H, Lee J. Background information of deep learning for structural engineering. Arch Comput Methods Eng 2018;25(1):121–9.
[21] Haftka RT, Gürdal Z. Elements of structural optimization. Springer Science & Business Media; 2012.
[22] Kingma DP, Ba J. Adam: A method for stochastic optimization. 2014, arXiv preprint arXiv:1412.6980.
[23] Lee KS, Geem ZW, Lee SH, Bae KW. The harmony search heuristic algorithm for discrete structural optimization. Eng Optim 2005;37(7):663–84.
[24] Li LJ, Huang ZB, Liu F. A heuristic particle swarm optimization method for truss structures with discrete variables. Comput Struct 2009;87(7–8):435–43.
[25] Sadollah A, Bahreininejad A, Eskandar H, Hamdi M. Mine blast algorithm for optimization of truss structures with discrete variables. Comput Struct 2012;102:49–63.
[26] Sadollah A, Eskandar H, Bahreininejad A, Kim JH. Water cycle, mine blast and improved mine blast algorithms for discrete sizing optimization of truss structures. Comput Struct 2015;149:1–16.
[27] Kaveh A, Mahdavi VR. Colliding bodies optimization method for optimum discrete design of truss structures. Comput Struct 2014;139:43–53.
[28] Toğan V, Daloğlu AT. An improved genetic algorithm with initial population strategy and self-adaptive member grouping. Comput Struct 2008;86(11–12):1204–18.
[29] Talebpour MH, Kaveh A, Kalatjari VR. Optimization of skeletal structures using a hybridized ant colony-harmony search-genetic algorithm. Iran J Sci Technol Trans Civil Eng 2014;38(C1):1.
[30] Azad SK, Hasançebi O. An elitist self-adaptive step-size search for structural design optimization. Appl Soft Comput 2014;19:226–35.
[31] Wu SJ, Chow PT. Steady-state genetic algorithms for discrete optimization of trusses. Comput Struct 1995;56(6):979–91.
[32] Kaveh A, Ghazaan MI. A comparative study of CBO and ECBO for optimal design of skeletal structures. Comput Struct 2015;153:137–47.
[33] Duc-Le Thang, Nguyen Quoc-Hung, Nguyen-Xuan H. Balancing composite motion optimization. Inform Sci 2020;520:250–70.