



Deep autoencoder based energy method for the bending, vibration, and buckling analysis of Kirchhoff plates with transfer learning

Xiaoying Zhuang^{a,c}, Hongwei Guo^c, Naif Alajlan^d, Hehua Zhu^a, Timon Rabczuk^{b,d,*}

^a Department of Geotechnical Engineering, Tongji University, Shanghai, China

^b Institute of Structural Mechanics, Bauhaus University Weimar, Weimar, Germany

^c Chair of Computational Science and Simulation Technology, Faculty of Mathematics and Physics, Leibniz Universität Hannover, Hannover, Germany

^d ALISR Laboratory, College of Computer and Information Sciences, King Saud University, P.O. Box 51178, Riyadh 11543, Saudi Arabia

ARTICLE INFO

Keywords:

Deep learning
Autoencoder
Activation function
Energy method
Kirchhoff plate
Vibration
Buckling
Transfer learning

ABSTRACT

In this paper, we present a deep autoencoder based energy method (DAEM) for the bending, vibration and buckling analysis of Kirchhoff plates. The DAEM exploits the higher order continuity of the DAEM and integrates a deep autoencoder and the minimum total potential principle in one framework yielding an unsupervised feature learning method. The DAEM is a specific type of feedforward deep neural network (DNN) and can also serve as function approximator. With robust feature extraction capacity, the DAEM can more efficiently identify patterns behind the whole energy system, such as the field variables, natural frequency and critical buckling load factor studied in this paper. The objective function is to minimize the total potential energy. The DAEM performs unsupervised learning based on generated collocation points inside the physical domain so that the total potential energy is minimized at all points. For the vibration and buckling analysis, the loss function is constructed based on Rayleigh's principle and the fundamental frequency and the critical buckling load is extracted. A scaled hyperbolic tangent activation function for the underlying mechanical model is presented which meets the continuity requirement and alleviates the gradient vanishing/explosive problems under bending. The DAEM is implemented using Pytorch and the LBFGS optimizer. To further improve the computational efficiency and enhance the generality of this machine learning method, we employ transfer learning. A comprehensive study of the DAEM configuration is performed for several numerical examples with various geometries, load conditions, and boundary conditions.

1. Introduction

Thin plates are commonly used in engineering and mechanics (Ventsel and Krauthammer, 2001) due to their light weight. Their mechanical analysis are of major importance in engineering practice. Due to the limitations of analytical methods, especially in solving complicated-shaped plate, a variety of numerical methods have been developed including the finite element method (Bathe, 2006; Hughes, 2012), boundary element method (Katsikadelis, 2016; Brebbia and Walker, 2016), meshfree method (Nguyen et al., 2008; Bui and Nguyen, 2011), isogeometric analysis (IGA) formulations (Nguyen et al., 2015), numerical manifold method (Zheng et al., 2013; Guo and Zheng, 2018; Guo et al., 2019a) and recently deep learning based methods (Anitescu et al., 2019; Guo et al., 2019b; Nguyen-Thanh et al., 2019).

The current wave of deep learning method started around 2006 (Hinton et al., 2006; Bengio et al., 2007). It is a feature learning method with neural network architectures including multiple hidden

layers (LeCun et al., 2015). Equipped with this hierarchical structure, it can extract information from complicated raw input data with multiple levels of abstraction through a layer-by-layer process (Goodfellow et al., 2016). Various variants such as multilayer perceptron (MLP), convolutional neural networks (CNN) and recurrent/recursive neural networks (RNN) (Patterson and Gibson, 2017) have been developed and applied to e.g. image processing (Yang et al., 2018a; Kermany et al., 2018), object detection (Ouyang et al., 2015; Zhao et al., 2019), speech recognition (Amodei et al., 2016; Nassif et al., 2019), biology (Yue and Wang, 2018; Ching et al., 2018) and even finance (Heaton et al., 2017; Fischer and Krauss, 2018).

Artificial neural networks (ANN) can be traced back to the 1940's (McCulloch and Pitts, 1943) but they became especially popular in the past few decades due to the vast development in computer science and computational science such as backpropagation technique and

* Corresponding author at: Institute of Structural Mechanics, Bauhaus University Weimar, Weimar, Germany.

E-mail addresses: zhuang@hot.uni-hannover.de (X. Zhuang), guo@hot.uni-hannover.de (H. Guo), najlan@ksu.edu.sa (N. Alajlan), zhuhehua@tongji.edu.cn (H. Zhu), timon.rabczuk@uni-weimar.de (T. Rabczuk).

<https://doi.org/10.1016/j.euromechsol.2021.104225>

Received 21 April 2020; Received in revised form 20 November 2020; Accepted 18 January 2021

Available online 26 January 2021

0997-7538/© 2021 Elsevier Masson SAS. All rights reserved.

Nomenclature

k	Generalized strains
M	Generalized stresses
n, s	Unit normal and tangent directions on the boundary
Q	Shear forces
$w = \{w_1, w_2\}$	Weights of encoding and decoding layers
Γ_1	Clamped boundary
Γ_2	Simply-supported boundary
Γ_3	Free boundary
λ	Buckling load factor
ν	Poisson ratio
Ω	Physical domain of the middle surface
ω	Natural frequency of the plate
Π, U, W_{ext}, K	Total potential energy, strain energy, the potential energy of external forces, kinetic energy
σ	Activation function
$f, g, g \circ f$	Mapping defined by encoder, decoder, and autoencoder
θ_x, θ_y	Rotation about the x, y -axis
\bar{M}_n	Prescribed moments to the normal of the boundaries
D_0	Bending rigidity
E	Young's modulus
h	Plate thickness
k_w, k_{θ_n}	Penalty parameters
$N_{i,p}$	B-Spline basis function
$R_{i,p}$	NURBS basis function
ρ	Density
$w(x, y)$	Deflection of middle plane

advances in deep neural networks. Due to the simplicity and feasibility of ANNs to deal with nonlinear and multi-dimensional problems, they were applied in inference and identification by data scientists (Dias et al., 2004). Artificial neural networks have already been used to solve partial differential equations (PDEs) two decades ago (Lagaris et al., 1998, 2000; McFall and Mahan, 2009) but **shallow ANNs are unable to learn complex nonlinear patterns effectively**. With improved theories incorporating robust training strategies, regularization techniques, advanced network architectures such as unsupervised pre-training, stacks of auto-encoder variants, and deep belief nets, it seems now feasible to design and train deeper and more complicated neural networks that can exploit larger datasets and systematic information with satisfying performance, which could be an interesting alternative to classical methods such as FEM.

According to the **universal approximation theorem** (Funahashi, 1989; Hornik et al., 1989), **any continuous function can be approximated by a feedforward neural network with one single hidden layer**. However, the number of neurons of the hidden layer tends to increase exponentially with increasing complexity and non-linearity of a model. Recent studies show that **deep neural networks (DNNs) render better approximations for nonlinear functions** (Mhaskar and Poggio, 2016). Some researchers studied the application of deep learning in discovering the solution of PDEs. Weinan et al. developed a deep learning-based numerical method for high-dimensional parabolic PDEs and back-forward stochastic differential equations (Weinan et al., 2017; Han et al., 2018). In Weinan and Yu (2018), they proposed a deep Ritz method for variational problems. Raissi et al. (2019) introduced Physics Informed Neural Networks (PINNs) for supervised learning of

nonlinear partial differential equations. Beck et al. (2019) employed deep learning to solve nonlinear stochastic differential equations and Kolmogorov equations. Sirignano and Spiliopoulos (2018) provided a theoretical proof for deep neural networks as PDE approximators, and concluded that it converged as the number of hidden layers tend to infinity for quasilinear parabolic PDEs. He et al. (2018) investigated the relationship between deep neural networks (DNNs) with rectified linear unit (ReLU) function as the activation function and continuous piecewise linear (CPWL) functions. Mao et al. (2020), Zhang et al. (2019), Pang et al. (2019) and Yang et al. (2018b) applied physics-informed neural networks to the solution of partial differential equations in fluid mechanics, and investigated the convergence and generalization of physics-informed neural networks, which incorporated the partial differential equations in the loss. Anitescu et al. (2019), Guo et al. (2019b), and Nguyen-Thanh et al. (2019) applied deep neural networks for finding the solutions for second and fourth order boundary value problems. Another interesting recent application of Physics Informed Neural Networks (PINNs) was presented by Haghighat et al. (2020). Different from most of the physics-informed neural networks mentioned above, our neural network is driven by the total potential energy of the system rather than the strong form, which is helpful in dealing with higher order partial differential equations.

The learning ability of deep neural networks has been enhanced with different architectures, such as **deep belief network (DBN) or deep autoencoder (DAE)** (Shao et al., 2017; Yu et al., 2019). DAE is widely used in **dimensionality reduction and feature learning**. It has also been proven to be an effective way to learn and describe latent codes that reflect meaningful variations in data with an **encoding and decoding layer** (Snoek et al., 2012). DAE seems therefore ideally suited for **learning and describing the underlying physical patterns from the governing partial differential equation or associated potential energy**. In this paper, we therefore propose a deep autoencoder based energy learning method for Kirchhoff plate analysis. In this context, we exploit the higher order continuity of the DAE approximation.

The paper is organized as follows: First, we describe the classical Kirchhoff plate model, including the elastic bending, vibration and buckling theory. Then, the key ingredients of the deep autoencoder theory is introduced, before an activation function is presented for this mechanical model and the energy driven deep learning model is described. Subsequently, the deep autoencoder based energy method is presented in detail. Finally, we demonstrate the efficiency and accuracy of the DAEM for various benchmark problems including comparisons between models with and without transfer learning technique.

2. Kirchhoff plate model

Consider a Kirchhoff plate as shown in Fig. 1. The displacement field can be expressed as:

$$\begin{aligned} u(x, y, z) &= -z \frac{\partial w}{\partial x}, \\ v(x, y, z) &= -z \frac{\partial w}{\partial y}, \\ w(x, y, z) &= w(x, y). \end{aligned} \quad (2.1)$$

where the relation between lateral deflection $w(x, y)$ of the middle surface ($z = 0$) and rotations about the x, y -axis is given by

$$\theta_x = \frac{\partial w}{\partial x}, \quad \theta_y = \frac{\partial w}{\partial y}. \quad (2.2)$$

The transversal deflection of the mid-plane is regarded as field variable and the corresponding bending and twisting curvatures are generalized strains:

$$k_x = -\frac{\partial^2 w}{\partial x^2}, \quad k_y = -\frac{\partial^2 w}{\partial y^2}, \quad k_{xy} = -2\frac{\partial^2 w}{\partial x \partial y}. \quad (2.3)$$

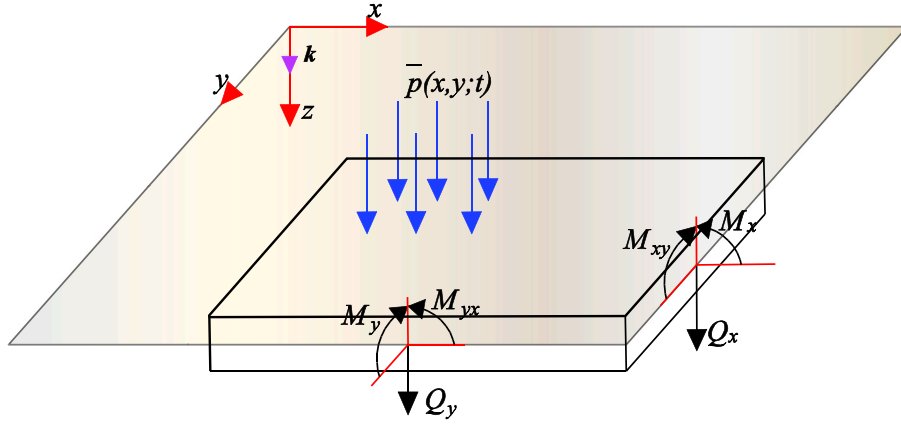


Fig. 1. Kirchhoff thin plate in the Cartesian coordinate system.

The geometric equations can be then obtained as:

$$\mathbf{k} = \begin{Bmatrix} k_x \\ k_y \\ k_{xy} \end{Bmatrix} = - \begin{Bmatrix} \frac{\partial^2 w}{\partial x^2} \\ \frac{\partial^2 w}{\partial y^2} \\ 2 \frac{\partial^2 w}{\partial x \partial y} \end{Bmatrix} = \mathbf{L} w, \quad (2.4)$$

with \mathbf{L} being the differential operator defined as $\mathbf{L} = - \begin{pmatrix} \frac{\partial^2}{\partial x^2} & \frac{\partial^2}{\partial y^2} & 2 \frac{\partial^2}{\partial x \partial y} \end{pmatrix}^T$.

Accordingly, the bending and twisting moments shown in Fig. 1 can be expressed as:

$$\begin{aligned} M_x &= -D_0 \left(\frac{\partial^2 w}{\partial x^2} + \nu \frac{\partial^2 w}{\partial y^2} \right), \\ M_y &= -D_0 \left(\frac{\partial^2 w}{\partial y^2} + \nu \frac{\partial^2 w}{\partial x^2} \right), \\ M_{xy} &= M_{yx} = -D_0 (1 - \nu) \frac{\partial^2 w}{\partial x \partial y}. \end{aligned} \quad (2.5)$$

Here, $D_0 = \frac{Eh^3}{12(1-\nu^2)}$ is the bending rigidity, E and ν denote the Young's modulus and Poisson ratio, and h is the thickness of the thin plate. It can be rewritten in a matrix form

$$\mathbf{M} = \mathbf{D} \mathbf{k} \quad (2.6)$$

with $\mathbf{D} = D_0 \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & (1-\nu)/2 \end{bmatrix}$. The shear forces of the mid-surface

are obtained in terms of the bending and twisting moments as

$$Q_x = \frac{\partial M_x}{\partial x} + \frac{\partial M_{xy}}{\partial y}, \quad Q_y = \frac{\partial M_{xy}}{\partial x} + \frac{\partial M_y}{\partial y}. \quad (2.7)$$

The boundary conditions can be categorized into three parts, namely,

$$\partial\Omega = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3. \quad (2.8)$$

At the **clamped boundary**, Γ_1 : $w = \bar{w}$, $\frac{\partial w}{\partial n} = \bar{\theta}_n$, $w = \bar{w}$, $\bar{\theta}_n$ are functions of the arc length along this boundary. At the **simply supported boundary**, Γ_2 : $w = \bar{w}$, $M_n = \bar{M}_n$, where \bar{M}_n is also a function of arc length along this boundary. At the **free boundary conditions**, Γ_3 : $M_n = \bar{M}_n$, $\frac{\partial M_{ns}}{\partial s} + Q_n = \bar{q}$, where \bar{q} is the load exerted along this boundary where n, s corresponds to the normal and tangent directions along the boundaries.

The total potential energy consists of the strain energy U and the potential energy of the external forces W :

$$\Pi = U + W_{ext}. \quad (2.9)$$

where

$$\begin{aligned} U &= \frac{1}{2} D_0 \int_{\Omega} \left[(k_x + k_y)^2 + 2(1-\nu) (k_{xy}^2 - k_x k_y) \right] d\Omega \\ &= \frac{1}{2} \int_{\Omega} \mathbf{k}^T \mathbf{D} \mathbf{k} d\Omega. \end{aligned} \quad (2.10)$$

and

$$W_{ext} = - \left[\int_{\Omega} p(x, y) w d\Omega + \int_{\Gamma_3} \bar{q} w d\Gamma - \int_{\Gamma_M} \bar{M}_n \frac{\partial w}{\partial n} d\Gamma \right], \quad (2.11)$$

with $\Gamma_M = \Gamma_2 \cup \Gamma_3$.

For vibration analysis assuming that the plate is undergoing harmonic vibrations, we can approximate the vibrating mid-surface of the plate by

$$w(x, y, t) = W(x, y) \sin \omega t, \quad (2.12)$$

The maximum kinetic energy can be obtained by choosing $\cos \omega t = 1$ as

$$K_{max} = \frac{\omega^2}{2} \int_{\Omega} \rho h W^2(x, y) d\Omega. \quad (2.13)$$

The maximum strain energy U_{max} occurs when $\sin \omega t = 1$. Accounting for the **Rayleigh's principle**, the lowest natural frequency of a vibrating plate can be obtained by setting:

$$K_{max} = U_{max} \quad (2.14)$$

where $U = \frac{1}{2} D_0 \int_{\Omega} \left[(\nabla^2 W)^2 + 2(1-\nu) \left(\left(\frac{\partial^2 W}{\partial x \partial y} \right)^2 - \frac{\partial^2 W}{\partial x^2} \frac{\partial^2 W}{\partial y^2} \right) \right] d\Omega$.

The Rayleigh's quotient can be defined as

$$\omega^2 = \frac{2U_{max}}{\int_{\Omega} \rho h W^2(x, y) d\Omega}, \quad (2.15)$$

which will be used in the subsequent analysis.

Moreover, we will introduce the **energy criterion to the classical eigenvalue buckling of Kirchhoff plates**. Bifurcation of an initial configuration of equilibrium occurs (Ventsel and Krauthammer, 2001) when the increment in the total potential energy of plate upon buckling equals zero:

$$\Delta \Pi = 0 \quad (2.16)$$

The increment in the total potential energy of the plate upon buckling $\Delta \Pi$ can be expressed as strain energy of the bending and twisting of a plate U plus the work done by in-plane forces (Ventsel and Krauthammer, 2001),

$$\begin{aligned} \Delta \Pi &= \frac{D_0}{2} \int_{\Omega} \left[(\nabla^2 w)^2 + 2(1-\nu) \left(\left(\frac{\partial^2 w}{\partial x \partial y} \right)^2 - \frac{\partial^2 w}{\partial x^2} \frac{\partial^2 w}{\partial y^2} \right) \right] d\Omega \\ &\quad + \frac{1}{2} \int_{\Omega} \left[\bar{N}_x \left(\frac{\partial w}{\partial x} \right)^2 + \bar{N}_y \left(\frac{\partial w}{\partial y} \right)^2 + 2\bar{N}_{xy} \frac{\partial w}{\partial x} \frac{\partial w}{\partial y} \right] d\Omega. \end{aligned} \quad (2.17)$$

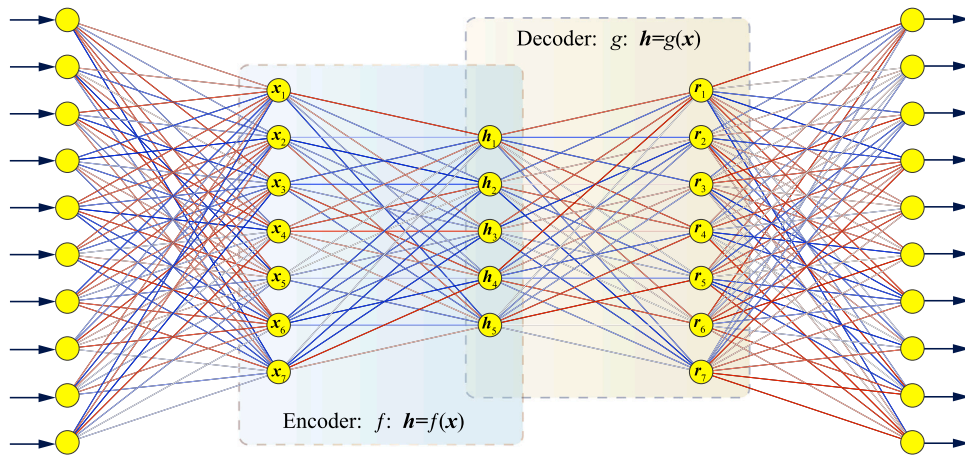


Fig. 2. Basic structure of an autoencoder. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Here, w denotes the perturbed transverse deflection and in-plane forces can be chosen as $\bar{N}_x = \lambda N_x, \bar{N}_y = \lambda N_y, \bar{N}_{xy} = \lambda N_{xy}$ where the load parameter λ is multiplied to the referenced values of the stress resultants. In practical application, N_x, N_y, N_{xy} are often set to be unity and thus λ becomes the desired buckling load factor and can be obtained from Eq. (2.16):

$$\lambda = \frac{-\frac{D_0}{2} \int_{\Omega} \left[(\nabla^2 w)^2 + 2(1-\nu) \left(\left(\frac{\partial^2 w}{\partial xy} \right)^2 - \frac{\partial^2 w}{\partial x^2} \frac{\partial^2 w}{\partial y^2} \right) \right] d\Omega}{\frac{1}{2} \int_{\Omega} \left[N_x \left(\frac{\partial w}{\partial x} \right)^2 + N_y \left(\frac{\partial w}{\partial y} \right)^2 + 2N_{xy} \frac{\partial w}{\partial x} \frac{\partial w}{\partial y} \right] d\Omega}, \quad (2.18)$$

which will also be used in the subsequent analysis. The minimum of the load parameter is the critical buckling load.

3. Basic theory of a deep autoencoder

In deep learning, engineers have further enhanced the learning ability of deep neural networks with different architectures, such as deep belief network (DBN) or deep autoencoder (DAE), to mention a few. Autoencoders play a fundamental role in unsupervised learning and are widely chosen deep architectures for dimensionality reduction and feature learning, which have been proven to be an effective way to learn and describe latent codes that reflect meaningful variations from raw input data.

3.1. Network architecture

Autoencoders are a specific type of feedforward neural networks including an encoder and a decoder. The basic structure is shown in Fig. 2. The network reconstructs the input by mapping them from a high-dimensional space to a low-dimensional space enabling the hidden layer to learn a better representation of the input and the decoder layer then reconstructs the results to another space.

As shown in Fig. 2, the fully connected feedforward neural network is composed of multiple layers: an input layer, one encoding layer, one “bottleneck” layer, one decoding layer and one output layer. Each layer consists of one or more nodes called neurons, indicated by small coloured circles in Fig. 2. For the interconnected structure, every two neurons in neighbouring layers have a connection, which is represented by a connection weight. The weight between neuron k in the hidden layer $l-1$ and neuron j in hidden layer l is denoted by w_{jk}^l . No connection exists among neurons in the same layer as well as in the non-neighbouring layers. Data flows through this neural network via connections between neurons, starting from the input layer, through the encoding layer over the hidden layer to the decoding layer and finally through the output layer.

Shown in Fig. 2, two parts in different coloured shadows are shown. The encoder can be represented as the mapping through an activation function:

$$f : h = f(x) \quad (3.1)$$

and likewise, a decoding denotes a mapping:

$$g : r = g(h) \quad (3.2)$$

that produces a reconstruction. The autoencoder defines a compound mapping

$$AE : g \circ f : r = g(f(x)). \quad (3.3)$$

Let δ be the nonlinear activation function on each layer. There are many choices for the activation function and we will propose an improved version of the \tanh -activation function to analyse the mechanical response of Kirchhoff plates. Combined with weight ω_1 and bias vector b_1 of the encoder, the nonlinear mapping can be written as

$$f : h = \delta(\omega_1 x + b_1) \quad (3.4)$$

where x is the input data. Similarly, the decode maps can be written as

$$g : r = \delta(\omega_2 h + b_2) \quad (3.5)$$

where h is the output from encoder and r output of the decoder.

Default activation functions available in Pytorch such as the \tanh do not necessarily yield the best results for every model. We will show later in numerical experiments that it results sometimes in unstable results for the deep autoencoder based energy method (DAEM). Therefore, we suggest a modified activation function:

$$\delta(x) = \tanh\left(\frac{\pi}{2}x\right). \quad (3.6)$$

As shown in Fig. 3, this activation function yields better results due to the following reasons: (1) A larger range of upper values are forced towards +1 and a larger range of lower values are close to -1, with steeper gradients for the mid-range, which can be seen in Fig. 3. Thus, the training is spread uniformly through the feedforward neural network. (2) For Kirchhoff plate problems, Navier successfully solved those problems with trigonometric Fourier series. Considering that $\sin\left(\frac{\pi}{2}x\right), \cos\left(\frac{\pi}{2}x\right)$ is periodically changing from -1 to +1, it can be also adopted as activation function suitable for dynamic analysis. However, periodic activation functions might lead to a “rippling” cost function with bad local minima since a low and high input may produce the same output making the neural network very difficult to train. The hyperbolic functions, however, satisfy many identities analogous to the trigonometric identities and also map input between -1 and +1.

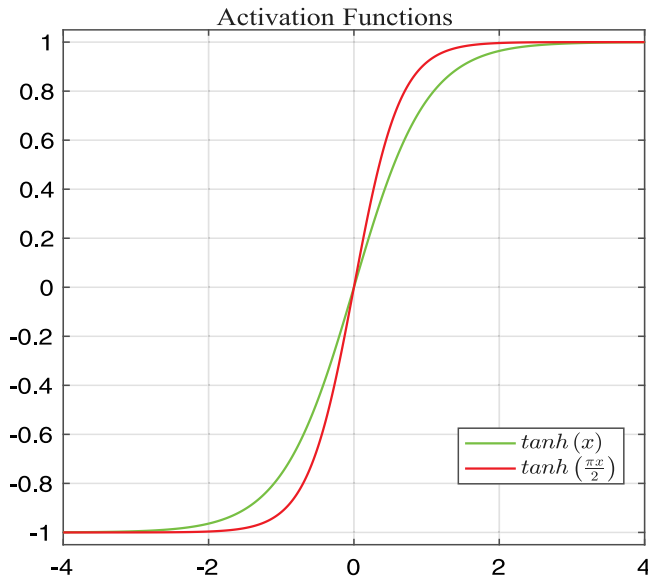


Fig. 3. New activation and original activation in each forms.

In this case, the scaled hyperbolic function $\tanh\left(\frac{\pi}{2}x\right)$ is preferable, as it yields stable and accurate results and will not slow down the training process. This small change in the activation function has largely improved the Vanishing/Exploding gradient problems in the deep autoencoder neural networks for this model, as will be validated later in numerical example section. The modified new activation has been studied under different layers and different neurons for cases in which Vanishing/Exploding gradient problems are observed for the \tanh activation function. However, for more general mechanical cases, the adaptive activation function taken $\tanh(\beta x)$ (Jagtap et al., 2020) is recommended.

The number of neurons on each hidden layer and the number of hidden layers can be arbitrarily chosen and are invariably determined through a trial and error procedure or a pruning technique (Anwar et al., 2017). In the numerical example section, the detailed configuration of this deep autoencoder has been studied and offers an optimum selection of number of hidden layers while keeping the number of neurons on each layer as small as possible. In summary, an autoencoder defines a continuous function $(g \circ f)(x; \theta)$ depending on the input data x and parametrized by $\theta = \{\omega_1, b_1, \omega_2, b_2\}$ consisting of weights and biases in each layer. It provides an efficient way to approximate unknown field variables and identifies those physical codes behind the model.

3.2. Energy driven autoencoder architecture

The scheme of the proposed energy driven autoencoder architecture for different plate problems is shown in Fig. 4. Its basis is the universal approximation theorem (Hornik, 1991). Based on universal approximation theorem, the deflection can be approximated by the deep autoencoder. Consequently, all physical variables related to deflection can be obtained by the neural network inside the blue dashed box. The total potential energy, which is the function of deflection, can be expressed by the autoencoder, resulting in an energy driven neural network sharing the same hyper-parameters with the deflection. The loss can be constructed from the total potential energy and the boundary conditions. The goal is to minimize the loss function w.r.t. the weights and biases. The detailed formulation will be presented in Section 4.

3.3. Basic algorithm for backpropagation

Like other feedforward neural networks, deep autoencoders can be trained with all techniques in deep learning such as minibatch gradient descent method with gradients computed by backpropagation. The minibatch gradient descent method (Ruder, 2016) refers to a variation of the gradient descent algorithm splitting the training dataset into small batches. It calculates the model error and updates the model hyperparameters, which will reduce the variance of parameter updates with more stable convergence and resulting in a higher computationally efficiency. Backpropagation is an important and computationally efficient mathematical tool to compute gradients in deep learning (Nielsen, 2015). It consists of two main phases, propagation and weight update. The chain rule is recursively applied during the whole process.

For the deep autoencoder based energy method, first, the field variable is approximated by a deep autoencoder $(g \circ f)(x; \theta)$. The components of the linear strain tensor are derivatives of the field variable and can be approximate by a set of deep autoencoders sharing the same hyperparameters. In order to find the hyperparameters of the deep autoencoder including weights and biases, a loss function $L(f, w)$ is constructed (Janocha and Czarnecki, 2017). The backpropagation algorithm for the deep autoencoder can be summarized as:

- **Input:** Input dataset x^1, \dots, x^n , prepare activation y^1 for input layer;
- **Feedforward:** For each layer $x^l, l = 2, 3, \dots, L$, compute $a^l = \sum_k W^l y^{l-1} + b^l$, and $\sigma(a^l)$;
- **Output error:** Compute the error $\delta^L = \nabla_{y^L} L \odot \sigma'_L(a^L)$
- **Backpropagation error:** For each $l = L - 1, L - 2, \dots, 2$, compute $\delta^l = ((W^{l+1})^T \delta^{l+1}) \odot \sigma'_l(a^l)$;
- **Output:** The gradient of the loss function is given by $\frac{\partial L}{\partial w_{jk}^l} = y_k^{l-1} \delta_j^l$ and $\frac{\partial L}{\partial b_j^l} = \delta_j^l$.

Here, \odot denotes the Hadamard product.

4. Deep autoencoder based energy method

Different from our previous work on the deep collocation method (Guo et al., 2019b), the presented approach minimizes the total potential energy of this system, which leads to two key advantages. Firstly, it reduces the requirements on the differentiability and continuity of the approximating function and hence requiring less gradient computations compared to the deep collocation method. And secondly, natural boundary conditions are automatically satisfied in the energy method, which is especially helpful for fourth order problems. In order to increase the efficiency of the energy driven machine learning method, transfer learning (TL) is used in DAEM.

4.1. Energy method for bending analysis

Let us consider Kirchhoff plate bending problems in the context of the DAEM. Recalling Eq. (2.9) is the total potential energy, the entire problem can be boiled down to minimizing the total potential energy subjected to essential boundary conditions. The transversal deflection w is approximated with the aforementioned deep autoencoder $(g \circ f)(x; \theta)$ denoted by $w^h(x; \theta)$. Substituting $w^h(x_\Omega; \theta)$ into Eq. (2.9), Eq. (2.10), and Eq. (2.12), results in

$$\Pi(x_\Omega; \theta) = U(x_\Omega; \theta) + W_{ext}(x_\Omega; \theta), \quad (4.1)$$

where

$$U(x_\Omega; \theta) = \frac{1}{2} D_0 \int_\Omega [(k_x(x_\Omega; \theta) + k_y(x_\Omega; \theta))^2 + 2(1 - \nu)(k_{xy}(x_\Omega; \theta)^2 - k_x(x_\Omega; \theta)k_y(x_\Omega; \theta))] d\Omega. \quad (4.2)$$

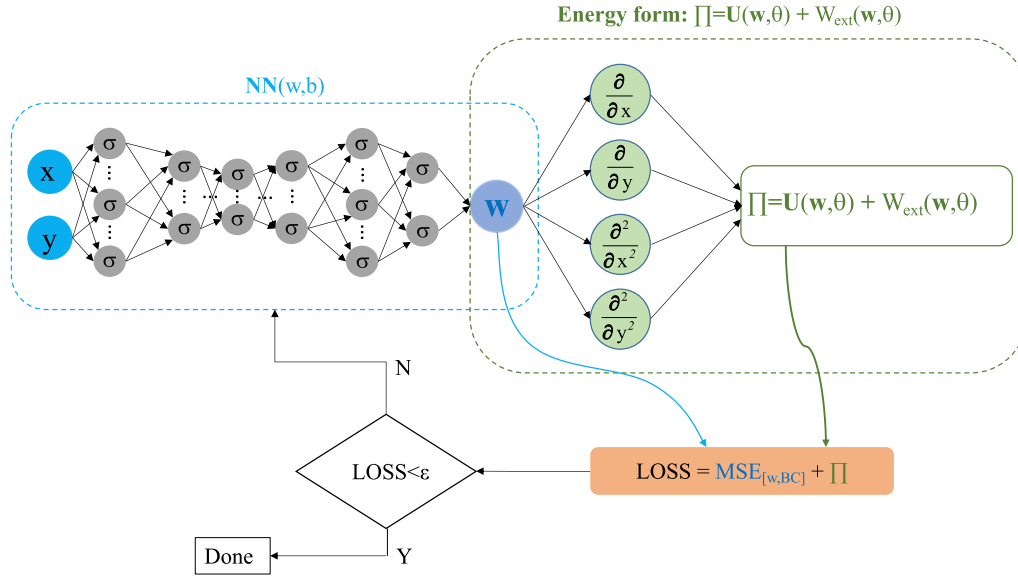


Fig. 4. Schematic of an energy driven autoencoder model.

and

$$W_{\text{ext}}(\mathbf{x}_\Omega; \theta) = - \left[\int_{\Omega} p(\mathbf{x}_\Omega) w^h(\mathbf{x}_\Omega; \theta) d\Omega + \int_{\Gamma_3} \tilde{q} w^h(\mathbf{x}_{\Gamma_3}; \theta) d\Gamma - \int_{\Gamma_M} \tilde{M}_n \frac{\partial w^h(\mathbf{x}_{\Gamma_M}; \theta)}{\partial n} d\Gamma \right], \quad (4.3)$$

with $\Gamma_M = \Gamma_2 \cup \Gamma_3$, which yields an energy driven deep neural network $\Pi(\mathbf{x}_\Omega; \theta)$. Moreover, $k_x(\mathbf{x}_\Omega; \theta)$, $k_y(\mathbf{x}_\Omega; \theta)$, and $k_{xy}(\mathbf{x}_\Omega; \theta)$ can be obtained by substituting $w^h(\mathbf{x}_\Omega; \theta)$ into Eq. (2.3). The boundary conditions from Section 2 can also be learnt by the neural network approximation $w^h(\mathbf{x}_\Gamma; \theta)$: On Γ_1 , we have

$$w^h(\mathbf{x}_{\Gamma_1}; \theta) = \tilde{w}, \quad \frac{\partial w^h(\mathbf{x}_{\Gamma_1}; \theta)}{\partial n} = \tilde{\theta}_n. \quad (4.4)$$

On Γ_2 ,

$$w^h(\mathbf{x}_{\Gamma_2}; \theta) = \tilde{w}, \quad \tilde{M}_n(\mathbf{x}_{\Gamma_2}; \theta) = \tilde{M}_n, \quad (4.5)$$

where $\tilde{M}_n(\mathbf{x}_{\Gamma_2}; \theta)$ can be obtained from Eq. (2.5) by combining $w^h(\mathbf{x}_{\Gamma_2}; \theta)$.

On Γ_3 ,

$$M_n(\mathbf{x}_{\Gamma_3}; \theta) = \tilde{M}_n, \quad \frac{\partial M_{ns}(\mathbf{x}_{\Gamma_3}; \theta)}{\partial s} + Q_n(\mathbf{x}_{\Gamma_3}; \theta) = \tilde{q}, \quad (4.6)$$

where $M_{ns}(\mathbf{x}_{\Gamma_3}; \theta)$ can be obtained from Eq. (2.5) and $Q_n(\mathbf{x}_{\Gamma_3}; \theta)$ can be obtained from Eq. (2.7) by combining $w^h(\mathbf{x}_{\Gamma_3}; \theta)$. Note that n, s refer to the normal and tangent directions along the boundaries. The induced energy driven neural network $\Pi(\mathbf{x}; \theta)$ shares the same parameters as $w^h(\mathbf{x}; \theta)$. To impose essential boundary conditions, the penalty method is employed, which results in the penalized variational formulation as follows:

$$\Pi_p = \Pi + \frac{1}{2} \int_{\Gamma_w} k_w (w - \tilde{w})^2 ds + \frac{1}{2} \int_{\Gamma_{\theta_n}} k_{\theta_n} \left(\frac{\partial w}{\partial n} - \tilde{\theta}_n \right)^2 ds, \quad (4.7)$$

where k_w and k_{θ_n} are user-specified penalty parameters. The **loss function for the proposed DAEM** can be constructed according to Eq. (4.7):

$$\text{Loss}(\theta) = \Pi + k_w \text{MSE}_{\Gamma_w} + k_{\theta_n} \text{MSE}_{\Gamma_{\theta_n}}, \quad (4.8)$$

with

$$\begin{aligned} \Pi &= \Pi(\mathbf{x}_\Omega; \theta), \\ \text{MSE}_{\Gamma_w} &= \frac{1}{N_{\Gamma_w}} \sum_{i=1}^{N_{\Gamma_w}} \left\| w^h(\mathbf{x}_{\Gamma_w}; \theta) - \tilde{w} \right\|^2, \\ \text{MSE}_{\Gamma_{\theta_n}} &= \frac{1}{N_{\Gamma_{\theta_n}}} \sum_{i=1}^{N_{\Gamma_{\theta_n}}} \left\| \frac{\partial w^h(\mathbf{x}_{\Gamma_{\theta_n}}; \theta)}{\partial n} - \tilde{\theta}_n \right\|^2, \end{aligned} \quad (4.9)$$

where $\Gamma_w = \Gamma_1 \cup \Gamma_2$, $\Gamma_{\theta_n} = \Gamma_1$; $\mathbf{x}_\Omega \in \mathbb{R}^N$, $\theta \in \mathbb{R}^K$ are the neural network parameters and $\text{Loss}(\theta) = 0$, $w^h(\mathbf{x}; \theta)$ is a solution to transversal deflection. The mean square error formulation refers to the discrete form of the integral at the essential boundary conditions ensuring our trained model has no outlier predictions with huge errors.

Note that the proposed **DAEM requires** a method to **evaluate the integrals** and corresponding quadrature points are deployed as input datasets. **We adopt traditional multivariate numerical quadrature methods such as Gaussian quadrature and the Monte Carlo integration method.** The details concerning suitable integration schemes fitted to the proposed model can be found in **Appendix**. The Monte Carlo integration method is easy to implement with randomly distributed sampling points generated inside the domain. Thus, it is most suitable for regular shaped domains. However, for more general cases, when NURBS geometries (Piegl and Tiller, 2012) are employed — as in example 5.5, the Gaussian quadrature method is employed.

4.2. Energy method for vibration and buckling analysis

The loss function for the vibration and buckling analysis has to be modified. The key objective is to obtain the fundamental natural frequency and critical buckling, respectively. Recalling Eq. (2.15), the Rayleigh quotient is defined and derived from Rayleigh's principle and the lowest natural frequency can be retrieved from the minimization of Eq. (2.15) accounting for essential boundary conditions, namely:

$$\begin{aligned} \omega_p^2 &= \frac{2U_{\text{max}}(\mathbf{x}_\Omega; \theta)}{\int_{\Omega} \rho h W^2(\mathbf{x}_\Omega; \theta) d\Omega} + \frac{1}{2} \int_{\Gamma_w} k_w (w - \tilde{w})^2 ds \\ &\quad + \frac{1}{2} \int_{\Gamma_{\theta_n}} k_{\theta_n} \left(\frac{\partial w}{\partial n} - \tilde{\theta}_n \right)^2 ds. \end{aligned} \quad (4.10)$$

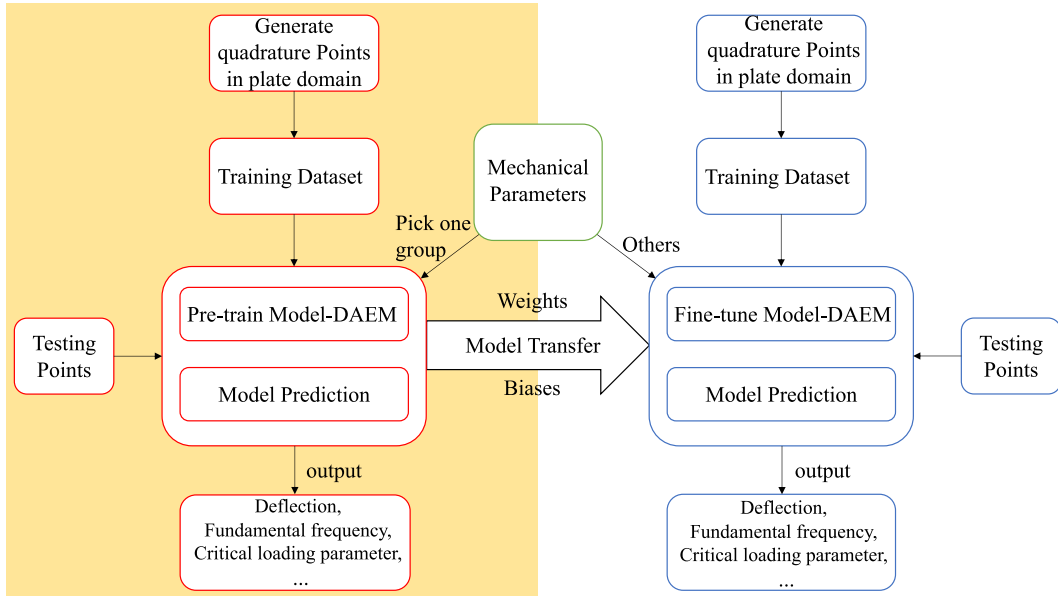


Fig. 5. Transfer learning.

Also the mode shape function is approximated by the deep autoencoder $W^h(\mathbf{x}; \theta)$. Accordingly, the loss function can be defined as:

$$Loss(\theta) = \frac{2U_{max}(\mathbf{x}_\Omega; \theta)}{\int_\Omega \rho h W^2(\mathbf{x}_\Omega; \theta) d\Omega} + k_w MSE_{\Gamma_w} + k_{\theta_n} MSE_{\Gamma_{\theta_n}}, \quad (4.11)$$

with

$$MSE_{\Gamma_w} = \frac{1}{N_{\Gamma_w}} \sum_{i=1}^{N_{\Gamma_w}} \|W^h(\mathbf{x}_{\Gamma_w}; \theta) - \bar{W}\|^2, \quad (4.12)$$

$$MSE_{\Gamma_{\theta_n}} = \frac{1}{N_{\Gamma_{\theta_n}}} \sum_{i=1}^{N_{\Gamma_{\theta_n}}} \left\| \frac{\partial W^h(\mathbf{x}_{\Gamma_{\theta_n}}; \theta)}{\partial \theta_n} - \bar{\theta}_n \right\|^2,$$

where $\Gamma_w = \Gamma_1 \cup \Gamma_2$, $\Gamma_{\theta_n} = \Gamma_1$; $\mathbf{x}_\Omega \in R^N$, $\theta \in R^K$ are the neural network parameters. Some modification to the loss function is needed to ensure $w^h(\mathbf{x}; \theta)$ is a nontrivial solution. To meet this end, we normalize the mode shape function and ensure the inner product of the mode shape function is unity. This leads to the modified loss function

$$Loss(\theta) = \frac{2U_{max}(\mathbf{x}_\Omega; \theta)}{\int_\Omega \rho h W^2(\mathbf{x}_\Omega; \theta) d\Omega} + k_p \left(\int_\Omega W^2(\mathbf{x}_\Omega; \theta) d\Omega - 1 \right)^2 + k_w MSE_{\Gamma_w} + k_{\theta_n} MSE_{\Gamma_{\theta_n}}, \quad (4.13)$$

where k_p is also a penalty factor. A factor between 1 to 100 already yields good numerical results. The loss function for the buckling analysis can be written as

$$Loss(\theta) = \lambda(\mathbf{x}_\Omega; \theta) + k_p \left(\int_\Omega (w^h(\mathbf{x}_\Omega; \theta))^2 d\Omega - 1 \right)^2 + k_w MSE_{\Gamma_w} + k_{\theta_n} MSE_{\Gamma_{\theta_n}}, \quad (4.14)$$

where λ is the load factor. From Eq. (2.18), we obtain

$$\lambda(\mathbf{x}_\Omega; \theta) = \frac{-\frac{D_0}{2} \int_\Omega \left[(\nabla^2 w^h)^2 + 2(1-\nu) \left(\left(\frac{\partial^2 w^h}{\partial x y} \right)^2 - \frac{\partial^2 w^h}{\partial x^2} \frac{\partial^2 w^h}{\partial y^2} \right) \right] d\Omega}{\frac{1}{2} \int_\Omega \left[N_x \left(\frac{\partial w^h}{\partial x} \right)^2 + N_y \left(\frac{\partial w^h}{\partial y} \right)^2 + 2N_{xy} \frac{\partial w^h}{\partial x} \frac{\partial w^h}{\partial y} \right] d\Omega} \quad (4.15)$$

where w^h is the approximation of the transversal deflection by the deep autoencoder, i.e. $w^h(\mathbf{x}_\Omega; \theta)$. Now, we can find the set of parameters

θ such that the approximated deflection $w^h(\mathbf{x}; \theta)$ minimizes the loss $Loss(\theta)$, i.e.

$$w^h = \arg \min_{\theta \in R^K} Loss(\theta). \quad (4.16)$$

These hyperparameters are obtained by backpropagation as mentioned before. The L-BFGS (Liu and Nocedal, 1989) optimizer with backpropagation is adopted to tune those hyperparameters of the deep autoencoder with few restrictions. Thus, the solution to thin plate bending, vibration and buckling problems using the energy driven deep autoencoder model is reduced to an optimization problem.

4.3. DAEM with transfer learning technique

To enhance the generality of the model for different material parameters, transfer learning is employed. For different materials parameters, the deep autoencoder model has to be trained again on the same dataset, the knowledge learnt for one model is not retained or accumulated, which can be time consuming, especially for a large database and data with complex patterns. Transfer learning is an alternative to conventional training. It focuses on storing knowledge while solving one problem and applying it to a different but related problem. The basic architecture of the transfer learning technique fitted to our method is shown in Fig. 5. The first step – of pre-training our model – is to generate points on the mid-surface. Then, one group of mechanical parameters is chosen and trained on the input dataset. The knowledge learnt including weights and biases form the pre-trained model is then inherited to formulate the fine-tune model, which will be used for predicting the remaining mechanical parameter groups. With transfer learning, the computational time for training can be greatly reduced and the neural network performance improved.

With generated input dataset employing points on middle surface and on the boundaries, the general procedure of the proposed DAEM can be summarized as follows:

5. Numerical experiments

In this section, we demonstrate the performance of DAEM for several numerical examples involving plate bending, vibration and buckling analysis. The simulations are done on a 64-bit macOS Mojave

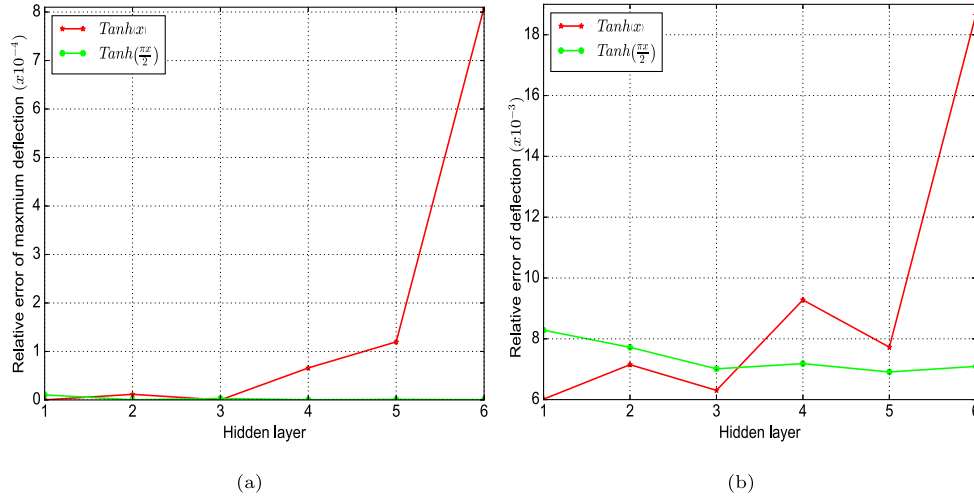


Fig. 6. Relative error of (a) maximum central deflection and (b) all deflection predicted by Tanh and proposed activation function with DNN.

Algorithm 1: Procedure for deep encoder based energy method

- Input:** Create Random Sampling Points \mathbf{x}_Q inside the physical domain and \mathbf{x}_F on the boundaries.
- Output:** The predicted field variables
- Data:** Testing data set: \mathbf{x}^*
1. Fix the number of neurons on input layer D_{in} , hidden layers H , encoding layers iH , $i \in \{1, \dots, N_{+f}\}$, and output layer D_{out} ; Choosing activation function δ and proper optimizer; Fixing the number of hidden layers N_{hl} and number of encoding layers N_{edl} ; Training iteration N_{iter} .
 2. **Model Training:**
 3. **for** i from 1 to N_{iter} **do**
 - (I) calculate activation function \mathbf{h}_i on hidden layers in Eq. (3.4).
 - (II) calculate the reconstructed output \mathbf{r}_i from \mathbf{h}_i in Eq. (3.5).
 - (III) choose and compute the loss function from bending loss function Eq. (4.8), the vibration loss function Eq. (4.13) or the buckling loss function Eq. (4.14).
 - (IV) back-propagate error gradient and update weights and bias.
 4. **Inference:** Make predictions and inference based on the trained deep autoencoder.

server with Intel(R) Core(TM) i7-8850H CPU, 32 GB memory. We found that a deep neural structure with less width is preferred over a shallow structures. Hence, we mainly show results for increasing the number of hidden layers rather than increasing the number of neurons. The accuracy of the numerical results are determined through the relative error of the maximum deflection and the deflection over the whole plate. The relative error for all numerical examples are given by

$$e = \frac{\|w_{predict} - w_{analytical}\|}{\|w_{analytical}\|} \quad (5.1)$$

$\|\cdot\|$ referring to the l^2 -norm.

5.1. Bending analysis

Three benchmark problems are presented including a square plate with sinusoidal distributed loading, an annular disc and a plate on an elastic foundation.

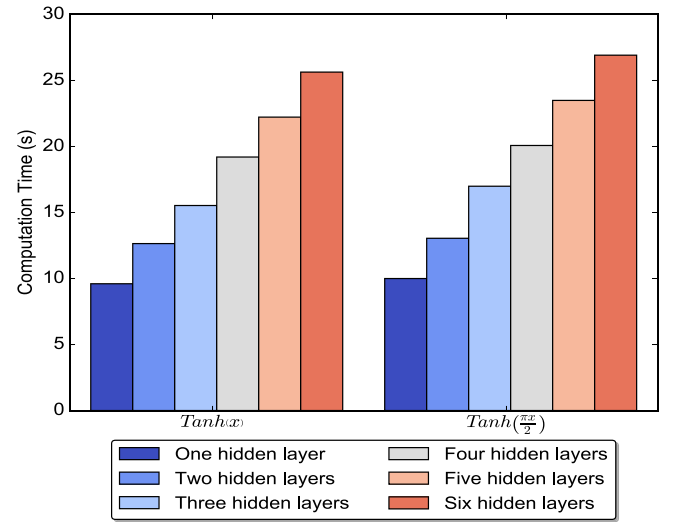


Fig. 7. The computational cost of two activation functions with increasing hidden layers.

5.2. Square plate under a sinusoidally distributed load

Let us consider a simply-supported square plate under a sinusoidal distributed transverse loading. The sinusoidal distributed load is expressed by

$$p = \frac{p_0}{D} \sin\left(\frac{\pi x}{a}\right) \sin\left(\frac{\pi y}{b}\right), \quad (5.2)$$

where a and b indicate the width and length of the plate, respectively. The analytical solution for this problem is given by (Timoshenko and Woinowsky-Krieger, 1959):

$$w = \frac{p_0}{\pi^4 D \left(\frac{1}{a^2} + \frac{1}{b^2}\right)^2} \sin\left(\frac{\pi x}{a}\right) \sin\left(\frac{\pi y}{b}\right). \quad (5.3)$$

First, we study the accuracy and efficiency of the proposed activation function for a deep feedforward neural network with 10 neurons per hidden layer. The relative error of the maximum deflection at the central plate and the deflection over the whole plate vs. the increasing of hidden layers is shown in Fig. 6. The modified hyperbolic tangent activation function $\tanh\left(\frac{\pi x}{2}\right)$ is less dependent on the number of hidden layers than the original hyperbolic tangent activation function.

The computational cost of those two schemes is shown in Fig. 7. The relative error of the deflection is shown in Table 1. While the gradient

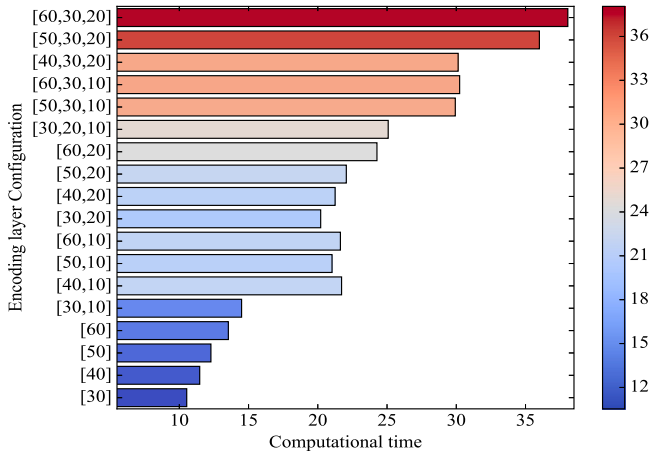


Fig. 8. The computational time for DAEM with different encoding configurations (s).

Table 1

The relative error of deflection for DAEM with different activation functions.

Encoder configuration	Relative error of deflection	
	$Tanh(x)$	$Tanh(\frac{\pi}{2}x)$
Encoding layer, [30]	0,0063740	0,0060618
Encoding layer, [40]	NaN	0,0058869
Encoding layer, [50]	0,0130702	0,0090355
Encoding layer, [60]	0,0077438	0,0104713
Encoding layers, [30,10]	0,0083107	0,0070922
Encoding layers, [40,10]	0,0082238	0,0066074
Encoding layers, [50,10]	0,0061002	0,0082771
Encoding layers, [60,10]	0,0075465	0,0058044
Encoding layers, [30,20]	0,0075115	0,0078926
Encoding layers, [40,20]	0,0087727	0,0086784
Encoding layers, [50,20]	NaN	0,0055310
Encoding layers, [60,20]	NaN	0,0084229
Encoding layers, [30,20,10]	0,0083553	0,0069302
Encoding layers, [50,30,10]	0,0121318	0,0077830
Encoding layers, [60,30,10]	0,0102968	0,0053692
Encoding layers, [40,30,20]	0,0075632	0,0075470
Encoding layers, [50,30,20]	0,0067328	0,0065867
Encoding layers, [60,30,20]	NaN	0,0060054

Encoding layer [num] refers to neurons on layers of the encoder.

The flip of [num] is the configuration for decoder.

explodes for some problems – indicated by NaN (not a real number) – for the original $Tanh$ activation function, the **modified $tanh$ activation function always yields stable results**. The encoding configuration refers here to the hidden layers and neuron numbers specified for the encoder. For the decoder a symmetric configuration is adopted.

Next, we investigate the recommended deep autoencoder configurations by comparing various encoders with varying layers and neurons per layer. As shown in Fig. 8, an increasing number of encoding layers results – as expected – in increased computational cost.

The relative errors obtained by different encoder schemes are shown in Fig. 9. The colorbar corresponds to the value of the relative error for the deflection. **The results improve with increasing number of layers**. However, adding more neurons to each layer does not necessarily improve the accuracy, especially for encoder [60]. The encoder with three layers yields the **optimum solution** at [60, 30, 10]. The results are already quite accurate for only one encoding layer.

Next, we test the influence of quadrature points on the accuracy of the solution by calculating the relative error of the maximum deflection and deflection. A series of randomly distributed quadrature points ranging from [100 16900] are used to calculate the integrals. The numerical results are shown in Fig. 10. It can be concluded that with the **increasing in quadrature points, the accuracy of the predicted deflection improves**.

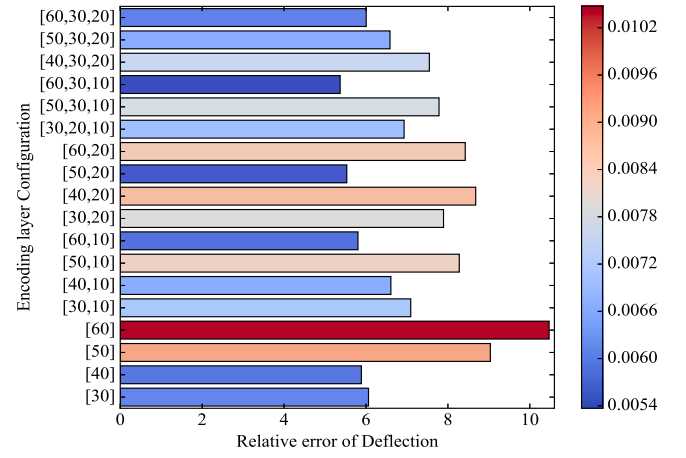


Fig. 9. The relative error of deflection for DAEM with different encoding configurations. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

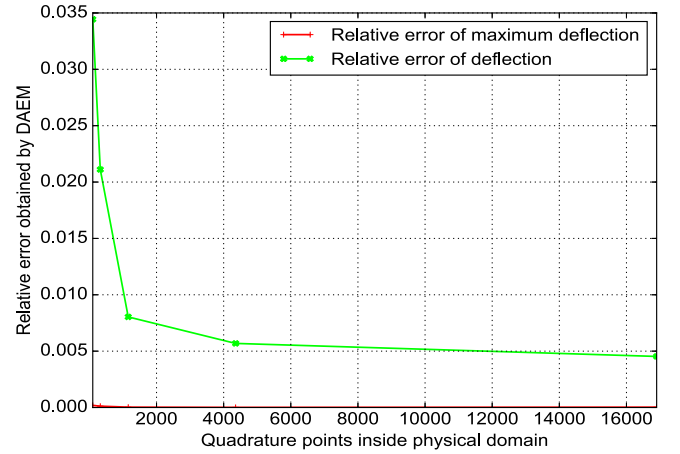


Fig. 10. The relative error of deflection for DAEM with increasing quadrature points.

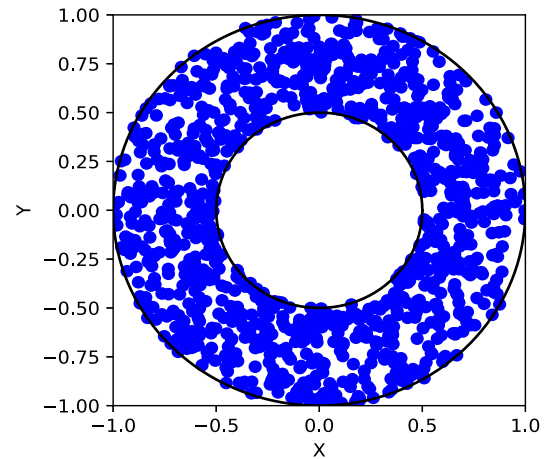


Fig. 11. Random points generated inside the annular plate.

5.2.1. Annular plate under uniformly distributed pressure

Next, we study an annular plate, which is simply-supported on the outer circle and free on the inner circle. The analytical solution for this problem is (Timoshenko and Woinowsky-Krieger, 1959):

$$w = \frac{qa^4}{64D} \left\{ - \left[1 - \left(\frac{r}{a} \right)^4 \right] + \frac{2\alpha_1}{1+\nu} \left[1 - \left(\frac{r}{a} \right)^2 \right] - \frac{4\alpha_2\beta^2}{1-\nu} \log \left(\frac{r}{a} \right) \right\}, \quad (5.4)$$

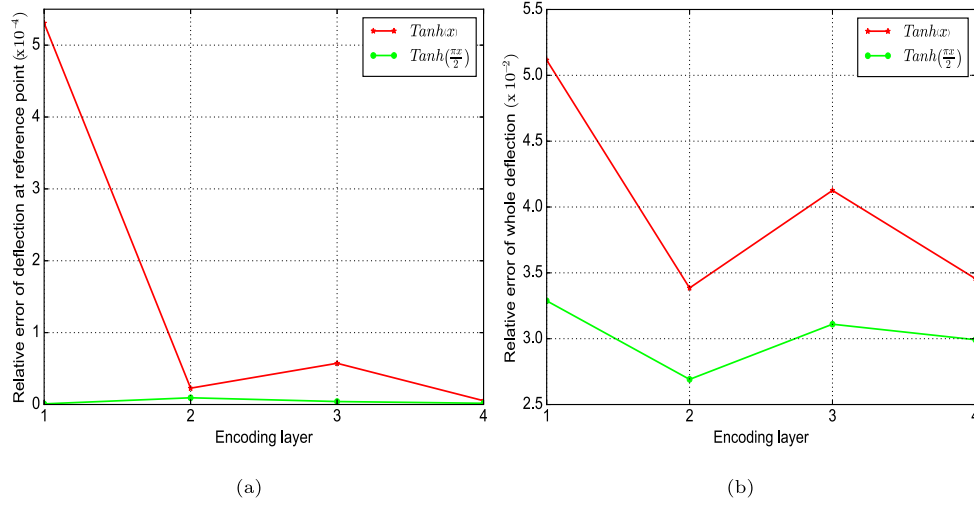


Fig. 12. Relative error of (a) deflection at reference point and (b) all deflection predicted by Tanh and proposed activation function with DAEM.

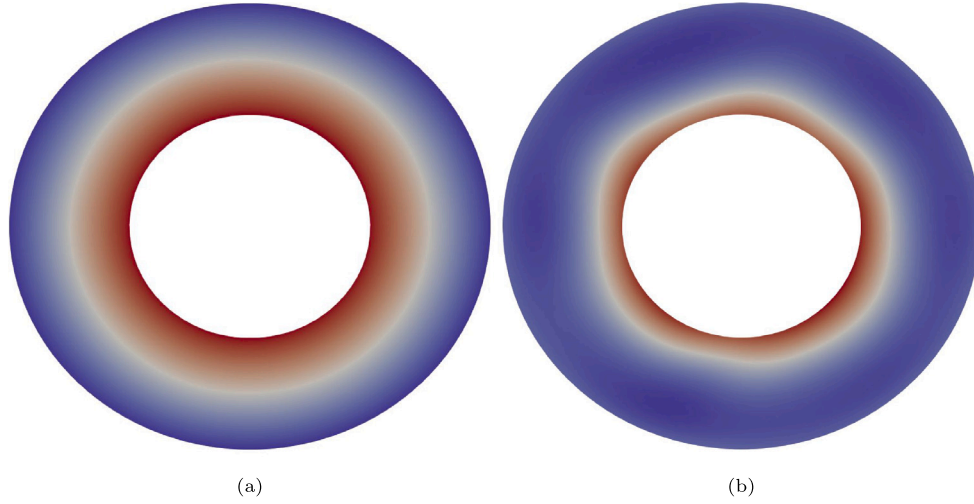


Fig. 13. (a) Deflection and (b) Absolute deflection contour predicted by DAEM.

where $\alpha_1 = (3 + \nu)(1 - \beta^2) - 4(1 + \nu)\beta^2\kappa$, $\alpha_2 = (3 + \nu) + 4(1 + \nu)\kappa$, $\beta = \frac{b}{a}$, $\kappa = \frac{\beta^2}{1 - \beta^2} \log \beta$, a , b being the outer and inner radius of the annular plate, respectively. The random generated points inside the annular plate for evaluating the loss are shown in Fig. 11.

The results at a reference point $(\frac{a+b}{2}, 0)$ are also predicted and compared with the analytical solution. Moreover, different activation function types are compared. We again observe the exploding gradient problem for DAEM with the $Tanh$ activation function, which can be alleviated by the $Tanh(\frac{x}{2})$, see Fig. 12. Contour plots of the deflection and absolute deflection are depicted in Fig. 13 showing that the predicted deflection agrees well with the analytical solution.

5.2.2. Square plate on Winkler foundation

Finally, we study a simply-supported plate on Winkler foundation assuming the foundation's reaction $p(x, y)$ is expressed by $p(x, y) = kw$, k being the foundation modulus. For a plate on a continuous Winkler foundation, the potential energy needs to be added to the total potential energy, Eq. (4.1):

$$W_s(\mathbf{x}_\Omega; \theta) = \frac{1}{2} \int_{\Omega} kw^h(\mathbf{x}_\Omega; \theta)^2 d\Omega. \quad (5.5)$$

The analytical deflection is given by (Timoshenko and Woinowsky-Krieger, 1959):

$$w = \frac{16p}{ab} \sum_{m=1,3,5,\dots}^{\infty} \sum_{n=1,3,5,\dots}^{\infty} \frac{\sin \frac{m\pi x}{a} \sin \frac{n\pi y}{b}}{mn \left[\pi^4 D \left(\frac{m^2}{a^2} + \frac{n^2}{b^2} \right)^2 + k \right]}. \quad (5.6)$$

Different configurations of the deep autoencoder are tested. The relative errors for deflection and maximum deflection are shown in Figs. 14 and 15, respectively. The colorbars in both figures indicate the relative error for the deflection and maximum deflection, respectively. Increasing the layers leads to more accurate results can be observed. For some cases, increasing the width of the neural network does not necessarily improve the accuracy of predicted results, such that a deep neural network is preferable. The computational cost is depicted in Fig. 16. As expected, more encoding layers increase the computational cost. However, note that this also increases the training cost. Once the network has been trained, the solution will be obtained much faster and with transfer learning technique, the training time can be reduced.

5.3. Vibration analysis

Next, we apply DAEM to extract the fundamental frequency in a transversal vibration analysis. The results are compared with reference

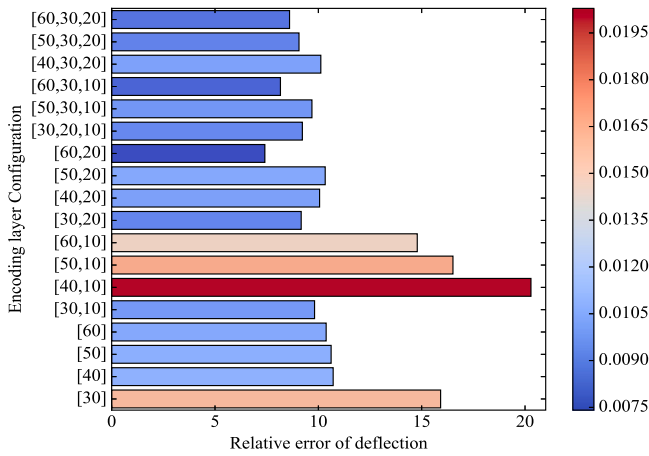


Fig. 14. The relative error of deflection for DAEM with different encoding configurations. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

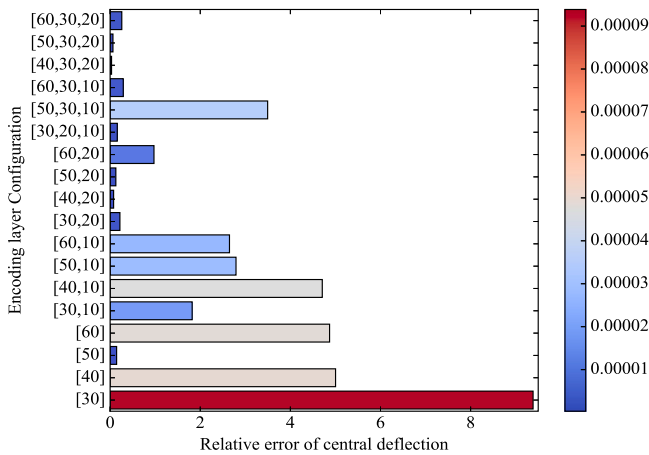


Fig. 15. The relative error of maximum deflection for DAEM with different encoding configurations. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

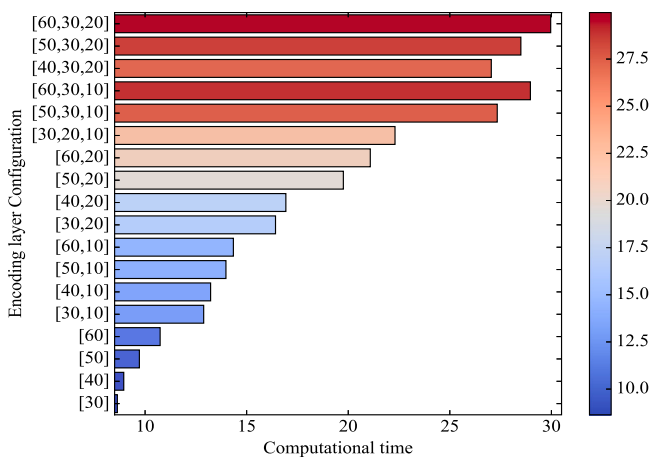


Fig. 16. The computational time of DAEM with different encoding configurations (s).

solutions from Zhang et al. (2018), Lam et al. (1989), Liew et al. (2003) and Shufrin and Eisenberger (2016). Let us consider a square plate with a square cutout as shown in Fig. 17, ξ is the ratio of the inner to outer square size. Various boundary conditions are studied to demonstrate

Table 2

Comparison of frequency parameter predicted by DAEM with other reference methods.

Cutout ratio ξ	Non-dimensional fundamental frequency parameter $\bar{\Omega} = \omega L^2 \sqrt{\rho h / D}$				
	DAEM	HBM method	Modified Ritz method	FEM	Discrete Ritz method
0	19,7382	19,7390	19,7400	19,7520	19,7390
0,1	19,3508	19,4440	19,1830	19,3570	19,4130
0,2	19,0284	19,1280	18,7620	19,1200	19,0380
0,3	19,3834	19,4450	19,1830	19,3570	19,3910
0,4	20,8201	20,7530	20,7850	20,7320	20,7240
0,5	23,4641	23,4530	23,6640	23,2350	23,4410
0,6	28,2706	28,3750	28,8440	28,2410	28,5260
0,7	38,1596	37,5720	38,1580	35,5790	37,8920
0,8	58,0804	57,4120	58,0620	57,4520	57,8380
0,9	120,9580	120,0200	121,2300	120,3900	120,9900

the robustness. The non-dimensional fundamental frequency parameter $\bar{\Omega} = \omega L^2 \sqrt{\rho h / D}$ with different cutout ratio ξ is investigated for model evaluation. A deep autoencoder with encoding layers [40, 20] is adopted as this architecture provided accurate results for the bending analysis while being computationally efficient.

The predicted nondimensional fundamental frequency for different cutout ratios is depicted in Table 2. The predicted results agree well with reference results using the HBM method (Zhang et al., 2018), Modified Ritz method (Lam et al., 1989), FEM (Lam et al., 1989), and Discrete Ritz method (Shufrin and Eisenberger, 2016).

The fundamental mode shapes for different cutout ratios are shown in Table 3 and agree well with results in Zhang et al. (2018).

Next, we perform a vibration analysis of the imperfect square plate for two other boundary conditions, i.e. 1: all outer edges are clamped and 2: two opposite outer edges clamped, while the other two outer edges simply-supported. The cutout ratio of this square plate is selected to be $\xi = 0.4$. The results are summarized in Table 4 and agree well with results from Zhang et al. (2018).

5.4. Buckling analysis

In this section, we study a skew plate with different skew angles θ and aspect ratio $\xi = a/b$ subjected to uniaxial inplane compressive loading, see Fig. 18. We consider simply-supported and clamped boundary conditions. The impacts of different skew angles, aspect ratios on the critical buckling load factor are computed and compared with the reference solution presented in Srinivasa et al. (2012).

The nonlinear encoding layer adopted in this study is [60, 20], which is suitable for varying geometries. The simply-supported skew plate is studied first and the associated results are shown in Table 5. The numerical results are compared with results of the Rayleigh–Ritz method, FEM, CQUAD4 and CQUAD8 (Srinivasa et al., 2012). As the skew angle increases, the critical buckling load parameter $K_{cr} = \frac{\lambda_{cr} b^2 h}{\pi^2 D}$ increases. For the clamped skew plate, we exemplary show results for the skew plate with aspect ratio $\xi = 1$. The numerical results can be found in Table 6. The predicted results also agree well with the reference solution.

Let us focus exemplary on the mode shape of the simply-supported plate with varying aspect ratios and a fixed skew angle of $\theta = 30$. The predicted mode shapes for each case are illustrated in Table 7. The buckling mode shapes for the clamped skew plate and a fixed aspect ratio of $\xi = 1$ and varying skew angle are listed in Table 8. The predicted results agree well with the ones in Srinivasa et al. (2012). Note that once the hyperparameters of this neural network are obtained, the network can be used to predict similar problems quickly and since the training is an unsupervised learning, no target value on the solution is needed.

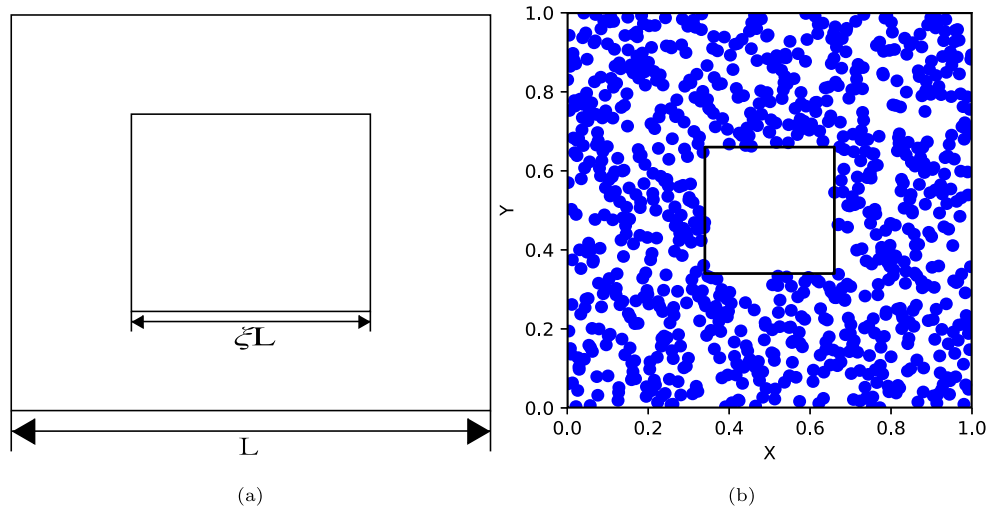


Fig. 17. (a) Kirchhoff thin plate in the Cartesian coordinate system and (b) random generated points inside the plate.

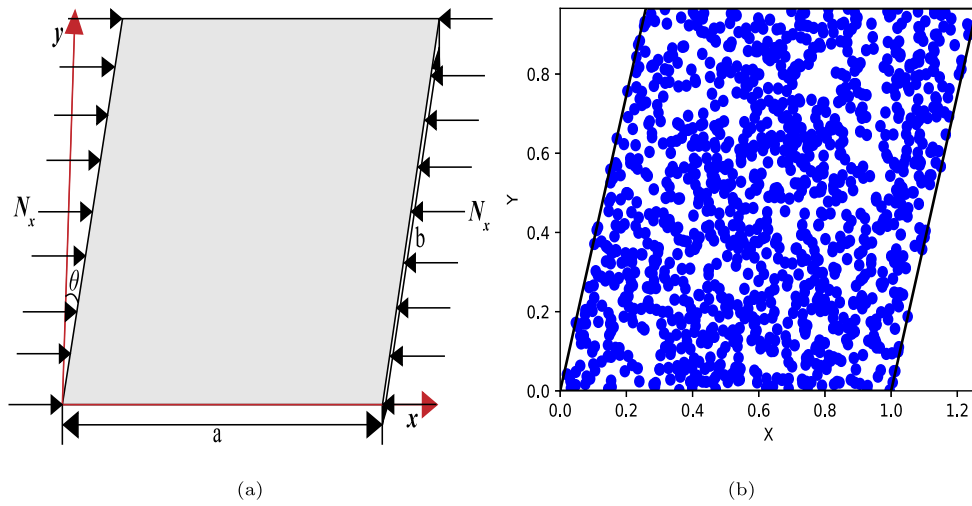


Fig. 18. (a) Kirchhoff thin plate in the Cartesian coordinate system and (b) random generated points inside the plate.

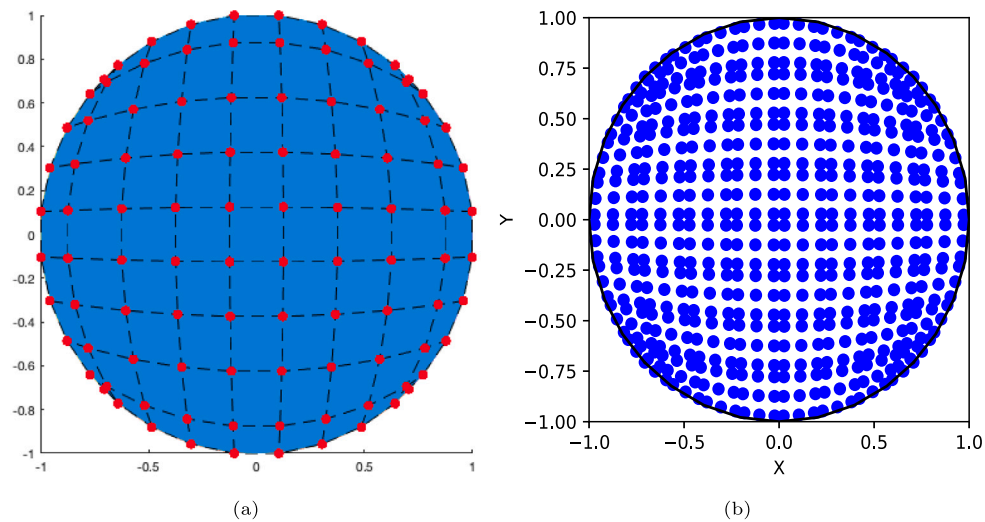
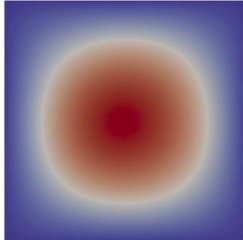
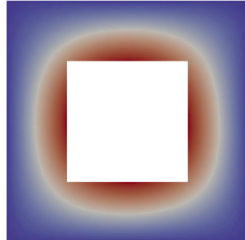
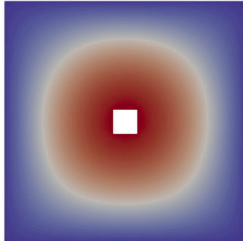
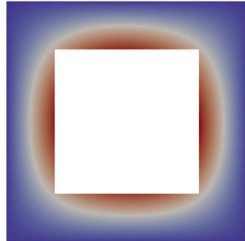
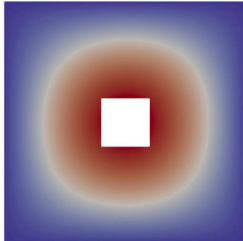
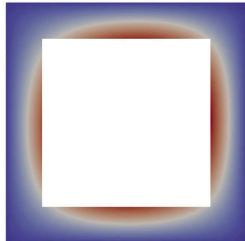
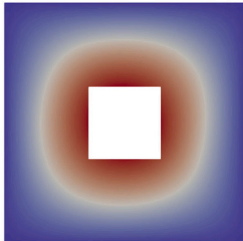
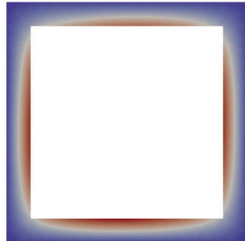
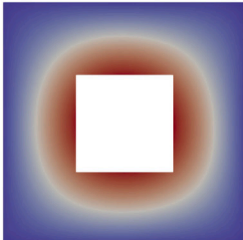



Fig. 19. (a) Circular plate represented by NURBS surface and (b) Gaussian points on the NURBS surface.

Table 3
Fundamental mode shapes predicted by DAEM with different cutout ratio.

Cutout ratio ξ	Fundamental mode shape	Cutout ratio ξ	Fundamental mode shape
0		0,5	
0,1		0,6	
0,2		0,7	
0,3		0,8	
0,4		0,9	

5.5. Analysis of circular plate with computer-aided design and transfer learning

In this section, we study the effect of transfer learning. Therefore, let us consider a clamped circular plate under a uniform load p_0 as illustrated in Fig. 19; 19(b) shows the quadrature points. For this numerical example, an analytical solution is available (Ventsel and Krauthammer, 2001):

$$w = \frac{p_0}{64} (R^2 - (x^2 + y^2))^2 \quad (5.7)$$

where R is the radius of the circular plate.

The relative error of the maximum deflection and the deflection over the circular domain for different material properties employing transfer learning is studied. The deep autoencoder is generated for models with Young's modulus $E = 10920$ and Poisson ratio $\nu = 0.3$. The same configurations of neural networks, with encoding layers

[48, 32, 16], 576 Gauss points and 50 epochs are applied. Then, several mechanical parameter combinations with $E = [10000, 20000, 30000]$ and Poisson ratios $\nu = [0.2, 0.3, 0.4]$ are investigated with a fine-tuning model, whose weights and biases are inherited from the defined pre-trained deep autoencoder. The numerical results are listed in Tables 9 and 10. We can predict the deflection for different mechanical parameters accurately. Moreover, with transfer learning, the predicted results are still quite accurate.

To demonstrate the efficiency of the proposed DAEM with transfer learning, the corresponding loss vs. iteration graph is depicted in Fig. 20. Including the energy in the loss 'complicates' training for conventional method. The deep autoencoder without transfer learning takes hundreds of iterations to decrease to the minimum. With transfer learning (TL), which is illustrated in dashed lines, the loss decreases faster and converges to a minimum value much faster. Both Tables show that the DAEM can predict the deflection for different mechanical

Table 4

Fundamental frequency parameter and mode shapes predicted by DAEM considering different boundary conditions ($\xi = 0.4$).

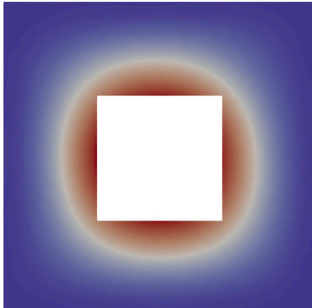
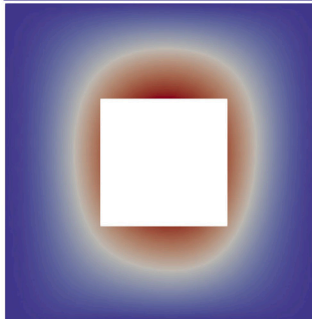
Boundary conditions	$\bar{\Omega} = \omega L^2 \sqrt{\rho h / D}$	Fundamental mode shape
CCCC	49,3091	
CSCS	35,4996	

Table 5

Comparison of critical buckling load parameter predicted by DAEM with other reference methods.

Aspect ratio ξ	Skew angle θ	Non-dimensional critical buckling load parameter K_{cr}				
		DAEM	Rayleigh–Ritz method	FEM	CQUAD4	CQUAD8
0,5	0°	6,2575	6,2500	6,2510	6,2010	6,2180
0,5	15°	7,0172	7,0000	6,9800	6,8550	6,9080
0,5	30°	9,9614	10,0200	9,9400	9,8950	10,0000
0,5	45°	19,4074	19,3000	9,4200	18,9510	19,2520
1	0°	4,0007	4,0000	4,0000	3,9190	4,0000
1	15°	4,5073	4,4800	4,4000	4,3060	4,3550
1	30°	5,8504	6,4100	5,9300	5,7610	5,8750
1	45°	10,5208	12,3000	10,3600	9,5260	9,9540
1,5	0°	4,3710	0,0000	0,0000	4,2560	4,2700
1,5	15°	4,6558	4,7700	4,6800	4,6400	4,6480
1,5	30°	5,9504	6,3700	5,8900	5,9550	5,8650
1,5	45°	9,1843	10,9000	8,9500	9,0760	9,1390
2	0°	3,9354	0,0000	0,0000	3,8850	3,9030
2	15°	4,3499	4,3300	4,3400	4,2710	4,3130
2	30°	5,5677	6,0300	5,5900	5,5960	5,6050
2	45°	8,9418	10,3000	8,8000	8,8550	8,8710

parameters accurately. Moreover, with transfer learning, the predicted results agree well with the results without using transfer learning. In some case such as $E = 10000$ and $\nu = 0.4$, the accuracy with transfer learning even improves, especially for the predicted maximum deflection for the circular plate. **For the fine tuning model, it requires less training data and reduces the training times.** The training time for both schemes are shown in Table 11.

Let us consider a simply-supported circular plate, which has been solved by IGA with a feature-preserving automatic meshing algorithm in Liu and Jeffers (2018). The mechanical and geometric parameters are: elastic modulus $E = 1e7$, Poisson ratio $\nu = 0.3$, Radius $R = 1$, and thickness $h = 0.02$. The exact solution for this problem is given by

$$w = -\frac{qR^2(3+\nu)(x^2+y^2)}{32D(1+\nu)} + \frac{q(x^2+y^2)^2}{64D} + \frac{qR^4(5+\nu)}{64D(1+\nu)} \quad (5.8)$$

The relative error using different methods are summarized in Table 12. DAEM yields the most accurate results for the central deflection of the circular plate for the considered set-ups. The relative error

Table 6

Critical buckling load parameter of clamped skew plate ($\xi = 1$).

Skew angle θ	Non-dimensional critical buckling load parameter K_{cr}				
	DAEM	Rayleigh–Ritz method	FEM	CQUAD4	CQUAD8
0°	10,0909	10,0000	10,0800	9,8540	10,0000
15°	10,7821	10,9000	10,8400	10,6900	10,7750
30°	13,7013	13,5800	13,6000	13,5030	13,5370
45°	20,9890	20,4000	20,7600	20,0920	20,1050

of the deflection is similar to the Abaqus results with 3020 nodes. The training time for DAEM with and without transfer learning are 1.4146 s and 22.284 s respectively, while the simulation takes 0.93381 s for our IGA (Nguyen et al., 2015) solution with 1156 control points.

The loss vs. iteration graph – with and without transfer learning – is shown in Fig. 21.

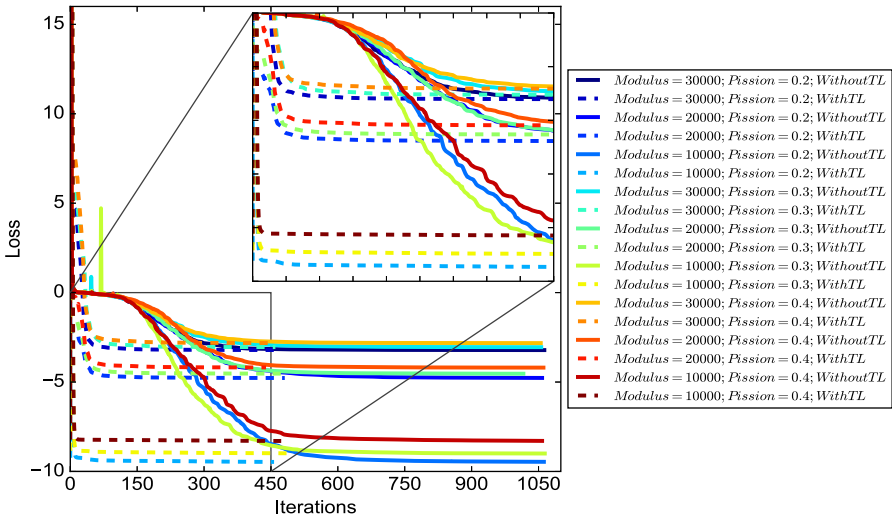


Fig. 20. The loss vs. iteration graph for different mechanical parameters with and without transfer learning.

Table 7
Buckling mode shapes of simply-supported skew plate predicted by DAEM with different aspect ratio and $\theta = 30^\circ$.

Aspect ratio ξ	Buckling mode shape	Aspect ratio ξ	Buckling mode shape
0.5		1	
1.5		2	

Table 8
Buckling mode shapes of clamped skew plate predicted by DAEM with different skew angle and $\xi = 1$.

Skew angle θ	Buckling mode shape	Skew angle θ	Buckling mode shape
0°		15°	
30°		45°	

Table 9

Relative error of maximum deflection with varying material properties.

E	ν					
	0.4		0.3		0.2	
	Without TL	With TL	Without TL	With TL	Without TL	With TL
30 000	1.33418129e-02	5.50687881e-03	1.90206674e-03	3.48579491e-03	7.23576546e-03	4.02013461e-03
20 000	5.78035627e-03	1.63184272e-04	3.59302563e-03	1.30789621e-04	1.72201792e-03	5.05553352e-04
10 000	3.49674528e-03	5.17103407e-05	3.15482450e-03	7.03469301e-04	2.41353777e-03	1.84408824e-03

Table 10

Relative error of deflection with varying material properties.

E	ν					
	0.4		0.3		0.2	
	Without TL	With TL	Without TL	With TL	Without TL	With TL
30 000	1.43941847e-02	1.06900852e-02	9.02399727e-03	9.31125250e-03	1.06958096e-02	9.71241595e-03
20 000	8.99004218e-03	6.61340973e-03	6.51702429e-03	6.55191283e-03	5.93318304e-03	6.26520573e-03
10 000	5.28733396e-03	5.39455341e-03	3.46101788e-03	4.88208336e-03	4.66406572e-03	4.39780683e-03

Table 11

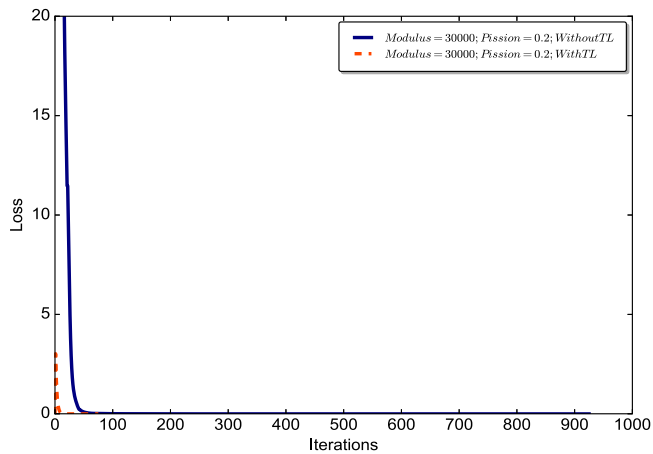
Computational time for DAEM with varying material properties (s or sec).

E	ν					
	0.4		0.3		0.2	
	Without TL	With TL	Without TL	With TL	Without TL	With TL
30 000	3.99291151e+01	1.14354441e+01	2.95081432e+01	1.17924690e+01	3.05779700e+01	1.24916120e+01
20 000	3.37831869e+01	1.00544729e+01	2.75741990e+01	1.29943578e+01	2.78740423e+01	1.06970160e+01
10 000	2.93843648e+01	1.10882282e+01	2.82655830e+01	1.12400260e+01	2.75469341e+01	1.08533742e+01

Table 12

Numerical results in deflection using different numerical methods.

Method	DAEM		Liu and Jeffers (2018)		Abaqus		IGA (Nguyen et al., 2015)		Exact solution
	Without TL	With TL	$\phi \leq 1\%$	$\phi \leq 0.1\%$	Shell element S3	Shell element S3	Order 3	Order 3	
Nodes	576	576	481	3004	1185	3020	324	1156	–
Centre deflection	8.6994e-03	8.6952e-03	8.6900e-03	8.6990e-03	8.5640e-03	8.6550e-03	8.6743e-03	8.6901e-03	8.6953e-03
e_{MD}	0.0470%	0.0008%	0.0580%	0.050%	1.5070%	0.460%	0.2416%	0.0604%	–
e_D	0.4076%	0.3449%	0.1810%	0.012%	0.910%	0.311%	0.2545%	0.0638%	–

 e_{MD} and e_D refers to the relative error for central deflection and deflection respectively.**Fig. 21.** The loss vs. iteration graph for the DAEM with and without transfer learning.

6. Conclusions

In this paper, we proposed a deep autoencoder based energy method for bending, vibration and buckling analysis of Kirchhoff plates. The deep autoencoder is suitable for unsupervised feature extraction. It is combined with the minimum total potential energy principle to the mechanical analysis of Kirchhoff plate. It has successfully captured the underlying physical patterns for several benchmark problems. Moreover, a tailored activation is proposed for the deep autoencoder based

energy method, which has been proven to be more stable and alleviated the gradient explosion problem without compromising computational efficiency. To calculate the total potential energy, the fitted Monte Carlo integration is adopted. For some problems, we exploited an IGA (NURBS-based) representation of the geometry and for convenience used Gauss quadrature to evaluate the integral in the loss function. Once the deep autoencoder based energy method is trained, it can predict the physical features quickly. Moreover, it is compatible to any geometry representation including CAD based designs, which allows for modelling complex shapes and a faster design-to-analysis procedure.

The deep autoencoder based energy method has been applied to extract the fundamental frequency, critical buckling load and corresponding mode shapes based on Rayleigh's principle. Different benchmark examples for bending, vibration and buckling analysis and different geometries and boundary conditions have been investigated to validate the proposed method. We also determined the most favourable deep autoencoder configuration. The proposed autoencoder based energy method is also integrated with transfer learning technique improving accuracy and/or efficiency. In the future, we intend to extend the approach to more complex problems including coupled problems and/or geometric and material nonlinearities.

CRediT authorship contribution statement

Xiaoying Zhuang: Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing - original draft, Writing - review & editing. **Hongwei Guo:** Conception and design of study, Acquisition of data, Analysis and/or interpretation of data,

Writing - original draft, Writing - review & editing. **Naif Alajlan:** Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing - original draft, Writing - review & editing. **Timon Rabczuk:** Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing - original draft, Writing - review & editing.

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.

Acknowledgements

The authors extend their appreciation to the Distinguished Scientist Fellowship Program (DSFP) at King Saud University for funding this work.

Appendix. Integration schemes for DAEM

For the two dimensional Monte-Carlo integration method, let us consider the integral $\int_{\Omega} p(x, y) w^h(x_{\Omega}; \theta) d\Omega$ in Eq. (4.3). It can be evaluated by

$$\int_{\Omega} p(x_{\Omega}) w^h(x_{\Omega}; \theta) d\Omega = \frac{A}{N_{\Omega}} \sum_{i=1}^{N_{\Omega}} p(x_{i,\Omega}) w^h(x_{i,\Omega}; \theta) \quad (\text{A.1})$$

$x_{i,\Omega}$ denoting the dataset generated by the random sampling in the physical domain A being the area of the mid-surface and N_{Ω} is the number of random distributed points inside the physical domain.

Alternatively, we could use Gaussian quadrature to evaluate integrals such as $\int_{\Omega} p(x, y) w^h(x_{\Omega}; \theta) d\Omega$ in Eq. (4.3). Therefore, we parametrize the domain Ω with NURBS function yielding

$$\int_{\Omega} p(x_{\Omega}) w^h(x_{\Omega}; \theta) d\Omega = \int_{\hat{\Omega}} p(\mathfrak{R}(\xi, \eta)) w^h(\mathfrak{R}(\xi, \eta); \theta) |J_{\mathfrak{R}}(\xi, \eta)| d\xi d\eta \quad (\text{A.2})$$

where \mathfrak{R} denotes the mapping from the parametric space to the domain Ω and $|J_{\mathfrak{R}}|$ refers to the determinant of its Jacobian matrix. By transforming from the parametric space to parent space defined by mapping Φ , yield

$$\begin{aligned} & \int_{\hat{\Omega}} p(\mathfrak{R}(\xi, \eta)) w^h(\mathfrak{R}(\xi, \eta); \theta) |J_{\mathfrak{R}}(\xi, \eta)| d\xi d\eta \\ &= \int_{\Omega_p} p(\mathfrak{R}(\Phi(s, t))) w^h(\mathfrak{R}(\Phi(s, t)); \theta) |J_{\mathfrak{R}}(\Phi(s, t))| |J_{\Phi}(s, t)| ds dt \quad (\text{A.3}) \end{aligned}$$

where Φ denotes the mapping from parent space Ω_p to the physical space $\hat{\Omega}$ and $|J_{\Phi}|$ refers to the determinant of its Jacobian matrix from parent space to parametric space. The discrete form for Eq. (A.4) using the Gaussian quadrature rules can be expressed as:

$$\begin{aligned} & \int_{\Omega_p} p(\mathfrak{R}(\Phi(s, t))) w^h(\mathfrak{R}(\Phi(s, t)); \theta) |J_{\mathfrak{R}}(\Phi(s, t))| |J_{\Phi}(s, t)| ds dt = \\ & \sum_I^{n_{gp}} \omega_I p(\mathfrak{R}(\Phi(s_I, t_I))) w^h(\mathfrak{R}(\Phi(s_I, t_I)); \theta) |J_{\mathfrak{R}}(\Phi(s_I, t_I))| |J_{\Phi}(s_I, t_I)|. \quad (\text{A.4}) \end{aligned}$$

References

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., et al., 2016. Deep speech 2: End-to-end speech recognition in english and mandarin. In: International Conference on Machine Learning. pp. 173–182.

Anitescu, C., Atroshchenko, E., Alajlan, N., Rabczuk, T., 2019. Artificial neural network methods for the solution of second order boundary value problems. *Comput. Mater. Contin.* 59 (1), 345–359.

Anwar, S., Hwang, K., Sung, W., 2017. Structured pruning of deep convolutional neural networks. *ACM J. Emerg. Technol. Comput. Syst. (JETC)* 13 (3), 32.

Bathe, K.-J., 2006. Finite element procedures. Klaus-Jürgen Bathe.

Beck, C., E, W., Jentzen, A., 2019. Machine learning approximation algorithms for high-dimensional fully nonlinear partial differential equations and second-order backward stochastic differential equations. *J. Nonlinear Sci.* <http://dx.doi.org/10.1007/s00332-018-9525-3>.

Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H., 2007. Greedy layer-wise training of deep networks. In: *Advances in Neural Information Processing Systems*. pp. 153–160.

Brebbia, C.A., Walker, S., 2016. Boundary element techniques in engineering. Elsevier.

Bui, T.Q., Nguyen, M.N., 2011. A moving kriging interpolation-based meshfree method for free vibration analysis of Kirchhoff plates. *Comput. Struct.* 89 (3–4), 380–394.

Ching, T., Himmelstein, D.S., Beaulieu-Jones, B.K., Kalinin, A.A., Do, B.T., Way, G.P., Ferrero, E., Agapow, P.-M., Zietz, M., Hoffman, M.M., et al., 2018. Opportunities and obstacles for deep learning in biology and medicine. *J. R. Soc. Interface* 15 (141), 20170387.

Dias, F.M., Antunes, A., Mota, A.M., 2004. Artificial neural networks: a review of commercial hardware. *Eng. Appl. Artif. Intell.* 17 (8), 945–952.

Fischer, T., Krauss, C., 2018. Deep learning with long short-term memory networks for financial market predictions. *European J. Oper. Res.* 270 (2), 654–669.

Funahashi, K.-I., 1989. On the approximate realization of continuous mappings by neural networks. *Neural Netw.* 2 (3), 183–192. [http://dx.doi.org/10.1016/0893-6080\(89\)90003-8](http://dx.doi.org/10.1016/0893-6080(89)90003-8), URL <http://www.sciencedirect.com/science/article/pii/0893608089900038>.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT press.

Guo, H., Zheng, H., 2018. The linear analysis of thin shell problems using the numerical manifold method. *Thin-Walled Struct.* 124, 366–383.

Guo, H., Zheng, H., Zhuang, X., 2019a. Numerical manifold method for vibration analysis of Kirchhoff's plates of arbitrary geometry. *Appl. Math. Model.* 66, 695–727.

Guo, H., Zhuang, X., Rabczuk, T., 2019b. A deep collocation method for the bending analysis of kirchhoff plate. *CMC-Comput. Mater. Contin.* 59 (2), 433–456.

Haghighat, E., Raissi, M., Moure, A., Gomez, H., Juanes, R., 2020. A deep learning framework for solution and discovery in solid mechanics. *arXiv preprint arXiv:2003.02751*.

Han, J., Jentzen, A., Weinan, E., 2018. Solving high-dimensional partial differential equations using deep learning. *Proc. Natl. Acad. Sci.* 115 (34), 8505–8510.

He, J., Li, L., Xu, J., Zheng, C., 2018. Relu deep neural networks and linear finite elements. *arXiv preprint arXiv:1807.03973*.

Heaton, J., Polson, N., Witte, J.H., 2017. Deep learning for finance: deep portfolios. *Appl. Stoch. Models Bus. Ind.* 33 (1), 3–12.

Hinton, G.E., Osindero, S., Teh, Y.-W., 2006. A fast learning algorithm for deep belief nets. *Neural Comput.* 18 (7), 1527–1554.

Hornik, K., 1991. Approximation capabilities of multilayer feedforward networks. *Neural Netw.* 4 (2), 251–257.

Hornik, K., Stinchcombe, M., White, H., 1989. Multilayer feedforward networks are universal approximators. *Neural Netw.* 2 (5), 359–366. [http://dx.doi.org/10.1016/0893-6080\(89\)90020-8](http://dx.doi.org/10.1016/0893-6080(89)90020-8), URL <http://www.sciencedirect.com/science/article/pii/0893608089900208>.

Hughes, T.J., 2012. The finite element method: linear static and dynamic finite element analysis. Courier Corporation.

Jagtap, A.D., Kawaguchi, K., Karniadakis, G.E., 2020. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. *J. Comput. Phys.* 404, 109136.

Janocha, K., Czarnecki, W.M., 2017. On loss functions for deep neural networks in classification. *Schedae Inform.* 1/2016, <http://dx.doi.org/10.4467/20838476si.16.004.6185>.

Katsikadelis, J.T., 2016. The boundary element method for engineers and scientists: theory and applications. Academic Press.

Kermany, D.S., Goldbaum, M., Cai, W., Valentim, C.C., Liang, H., Baxter, S.L., McKeown, A., Yang, G., Wu, X., Yan, F., et al., 2018. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell* 172 (5), 1122–1131.

Lagaris, I.E., Likas, A., Fotiadis, D.I., 1998. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Trans. Neural Netw.* 9 (5), 987–1000.

Lagaris, I.E., Likas, A.C., Papageorgiou, D.G., 2000. Neural-network methods for boundary value problems with irregular boundaries. *IEEE Trans. Neural Netw.* 11 (5), 1041–1049.

Lam, K., Hung, K., Chow, S., 1989. Vibration analysis of plates with cutouts by the modified Rayleigh-ritz method. *Appl. Acoust.* 28 (1), 49–60. [http://dx.doi.org/10.1016/0003-682X\(89\)90030-3](http://dx.doi.org/10.1016/0003-682X(89)90030-3), URL <http://www.sciencedirect.com/science/article/pii/0003682X89900303>.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. *nature* 521 (7553), 436.

Liew, K., Kitipornchai, S., Leung, A., Lim, C., 2003. Analysis of the free vibration of rectangular plates with central cut-outs using the discrete ritz method. *Int. J. Mech. Sci.* 45 (5), 941–959. [http://dx.doi.org/10.1016/S0020-7403\(03\)00109-7](http://dx.doi.org/10.1016/S0020-7403(03)00109-7), URL <http://www.sciencedirect.com/science/article/pii/S0020740303001097>.

Liu, N., Jeffers, A.E., 2018. A geometrically exact isogeometric kirchhoff plate: Feature-preserving automatic meshing and c 1 rational triangular Bézier spline discretizations. *Internat. J. Numer. Methods Engrg.* 115 (3), 395–409.

- Liu, D.C., Nocedal, J., 1989. On the limited memory BFGS method for large scale optimization. *Math. Programm.* 45 (1–3), 503–528.
- Mao, Z., Jagtap, A.D., Karniadakis, G.E., 2020. Physics-informed neural networks for high-speed flows. *Comput. Methods Appl. Mech. Engrg.* 360, 112789.
- McCulloch, W.S., Pitts, W., 1943. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.* 5 (4), 115–133.
- McFall, K.S., Mahan, J.R., 2009. Artificial neural network method for solution of boundary value problems with exact satisfaction of arbitrary boundary conditions. *IEEE Trans. Neural Netw.* 20 (8), 1221–1233.
- Mhaskar, H.N., Poggio, T., 2016. Deep vs. shallow networks: An approximation theory perspective. *Anal. Appl.* 14 (06), 829–848.
- Nassif, A.B., Shahin, I., Attili, I., Azzeh, M., Shaalan, K., 2019. Speech recognition using deep neural networks: a systematic review. *IEEE Access*.
- Nguyen, V.P., Anitescu, C., Bordas, S.P., Rabczuk, T., 2015. Isogeometric analysis: an overview and computer implementation aspects. *Math. Comput. Simulation* 117, 89–116.
- Nguyen, V.P., Rabczuk, T., Bordas, S., Duflot, M., 2008. Meshless methods: A review and computer implementation aspects. *Math. Comput. Simulation* 79 (3), 763–813. <http://dx.doi.org/10.1016/j.matcom.2008.01.003>, URL <http://www.sciencedirect.com/science/article/pii/S0378475408000062>.
- Nguyen-Thanh, V.M., Zhuang, X., Rabczuk, T., 2019. A deep energy method for finite deformation hyperelasticity. *Eur. J. Mech. A Solids* 103874.
- Nielsen, M.A., 2015. *Neural networks and deep learning*, Vol. 25. Determination press San Francisco, CA, USA.
- Ouyang, W., Wang, X., Zeng, X., Qiu, S., Luo, P., Tian, Y., Li, H., Yang, S., Wang, Z., Loy, C.-C., et al., 2015. Deepid-net: Deformable deep convolutional neural networks for object detection, In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2403–2412.
- Pang, G., Lu, L., Karniadakis, G.E., 2019. Fpinns: Fractional physics-informed neural networks. *SIAM J. Sci. Comput.* 41 (4), A2603–A2626.
- Patterson, J., Gibson, A., 2017. *Deep learning: A practitioner's approach*. O'Reilly Media, Inc..
- Piegl, L., Tiller, W., 2012. *The NURBS book*. Springer Science & Business Media.
- Raissi, M., Perdikaris, P., Karniadakis, G., 2019. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* 378, 686–707. <http://dx.doi.org/10.1016/j.jcp.2018.10.045>, URL <http://www.sciencedirect.com/science/article/pii/S0021999118307125>.
- Ruder, S., 2016. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.
- Shao, H., Jiang, H., Zhao, H., Wang, F., 2017. A novel deep autoencoder feature learning method for rotating machinery fault diagnosis. *Mech. Syst. Signal Process.* 95, 187–204.
- Shufrin, I., Eisenberger, M., 2016. Semi-analytical modeling of cutouts in rectangular plates with variable thickness – free vibration analysis. *Appl. Math. Model.* 40 (15), 6983–7000. <http://dx.doi.org/10.1016/j.apm.2016.02.020>, URL <http://www.sciencedirect.com/science/article/pii/S0307904X16300944>.
- Sirignano, J., Spiliopoulos, K., 2018. DGM: A deep learning algorithm for solving partial differential equations. *J. Comput. Phys.* 375, 1339–1364.
- Snoek, J., Adams, R.P., Larochelle, H., 2012. Nonparametric guidance of autoencoder representations using label information. *J. Mach. Learn. Res.* 13 (Sep), 2567–2588.
- Srinivasa, C., Suresh, Y., Kumar, W.P., 2012. Buckling studies on laminated composite skew plates. *Int. J. Comput. Appl.* 37 (1), 35–47.
- Timoshenko, S.P., Woinowsky-Krieger, S., 1959. *Theory of plates and shells*. McGraw-hill.
- Ventsel, E., Krauthammer, T., 2001. *Thin plates and shells: theory: analysis, and applications*. CRC press.
- Weinan, E., Han, J., Jentzen, A., 2017. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Commun. Math. Stat.* 5 (4), 349–380.
- Weinan, E., Yu, B., 2018. The deep ritz method: a deep learning-based numerical algorithm for solving variational problems. *Commun. Math. Stat.* 6 (1), 1–12.
- Yang, L., MacEachren, A., Mitra, P., Onorati, T., 2018a. Visually-enabled active deep learning for (geo) text and image classification: a review. *ISPRS Int. J. Geo-Inf.* 7 (2), 65.
- Yang, L., Zhang, D., Karniadakis, G.E., 2018b. Physics-informed generative adversarial networks for stochastic differential equations. *arXiv preprint arXiv:1811.02033*.
- Yu, J., Zheng, X., Wang, S., 2019. A deep autoencoder feature learning method for process pattern recognition. *J. Process Control* 79, 1–15.
- Yue, T., Wang, H., 2018. Deep learning for genomics: A concise overview. *arXiv preprint arXiv:1802.00810*.
- Zhang, D., Lu, L., Guo, L., Karniadakis, G.E., 2019. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. *J. Comput. Phys.* 397, 108850.
- Zhang, Y., Wang, C., Pedroso, D., Zhang, H., 2018. Extension of hencky bar-net model for vibration analysis of rectangular plates with rectangular cutouts. *J. Sound Vib.* 432, 65–87. <http://dx.doi.org/10.1016/j.jsv.2018.06.029>, URL <http://www.sciencedirect.com/science/article/pii/S0022460X18303900>.
- Zhao, Z.-Q., Zheng, P., Xu, S., Wu, X., 2019. Object detection with deep learning: A review. *IEEE Trans. Neural Netw. Learn. Syst.*.
- Zheng, H., Liu, Z., Ge, X., 2013. Numerical manifold space of hermitian form and application to kirchhoff's thin plate problems. *Internat. J. Numer. Methods Engrg.* 95 (9), 721–739.