Applied Intelligence (2022) 52:916–926
https://doi.org/10.1007/s10489-021-02474-4

A deep artificial neural network architecture for mesh free solutions of nonlinear boundary value problems

Riya Aggarwal[1] · **Hassan Ugail**[2] · **Ravi Kumar Jha**[3]

Accepted: 23 April 2021 / Published online: 13 May 2021
© The Author(s) 2021

Abstract

Seeking efficient solutions to nonlinear boundary value problems is a crucial challenge in the mathematical modelling of many physical phenomena. A well-known example of this is solving the Biharmonic equation relating to numerous problems in fluid and solid mechanics. One must note that, in general, it is challenging to solve such boundary value problems due to the higher-order partial derivatives in the differential operators. An artificial neural network is thought to be an intelligent system that learns by example. Therefore, a well-posed mathematical problem can be solved using such a system. This paper describes a mesh free method based on a suitably crafted deep neural network architecture to solve a class of well-posed nonlinear boundary value problems. We show how a suitable deep neural network architecture can be constructed and trained to satisfy the associated differential operators and the boundary conditions of the nonlinear problem. To show the accuracy of our method, we have tested the solutions arising from our method against known solutions of selected boundary value problems, e.g., comparison of the solution of Biharmonic equation arising from our convolutional neural network subject to the chosen boundary conditions with the corresponding analytical/numerical solutions. Furthermore, we demonstrate the accuracy, efficiency, and applicability of our method by solving the well known thin plate problem and the Navier-Stokes equation.

Keywords Artificial neural networks · Biharmonic equation · Boundary value problems · Von-Kármán equation · Navier-Stokes equation · Mesh free solutions

1 Introduction

Artificial neural networks (ANNs) can be utilised to create suitable tools to solve large-scale computational problems [2, 37]. ANNs involve the computational inference of simulated neuronal units, where every neuron has a real-valued input. Multiplying these units results in the corresponding neuronal coefficients. Then, using the results, a bias term is calculated. Finally, an investigating function, known as the activation function, is needed to determine a real-valued output.

Applications of ANNs are found in almost every field of computer science and engineering [31]. Networks can learn and store complex mapping relations of the input-output models, which are highly useful. Applications of ANNs, for example, appear to be numerous in the domain of healthcare. These include automated diagnosis of congestive heart failure using ECG signals [2], generating quantitative computed tomography reports for the diagnosis of pulmonary tuberculosis [25] and finding minute mammographic differences for the diagnosis of certain types of cancers [16].

Similarly, ANN architectures enhanced with backpropagation have recently been prevalent in solving problems that have been considered very challenging in the past. For example, Jha et al. [20] have proposed a deep learning model for designing a visual question answering system. This qualitative study shows how one might train a single but fast convolutional neural network (CNN) model by modifying weights in the parameter prediction layers. In

✉ Riya Aggarwal
Riya.aggarwal@uon.edu.au

Hassan Ugail
h.ugail@bradford.ac.uk

[1] School of Mathematics and Physical Sciences, The University of Newcastle, Callaghan, NSW 2308, Australia

[2] Centre for Visual Computing, University of Bradford, Bradford, UK

[3] Indian National Centre for Ocean Information Services, Hyderabad, India

[29], the authors discuss a study related to the sclera recognition system using CNNs. This CNN framework is formed using four major convolutional units linked with the corresponding convolutional layers. This is a self-learning model where the output of the first layer is fed to the next so that the model is better trained to provide efficient results. Similarly, a multi-layer feedforward neural network for internet traffic classification was proposed in [30]. They have used a model with four hidden layers for handling large amounts of imbalanced data.

Thus, recently, ANNs have shown great promise in solving many challenging problems in applied mathematics and computing. In this sense, several ways to solve partial differential equations (PDEs) using feedforward neural networks by substituting approximate solutions into the corresponding differential operator [22, 38] have been proposed. For example, Abdeljaber et al. [1] studied an interesting active vibration problem of cantilevered beam induced by a pulse concentrated load using an ANN architecture. Consequently, they proposed a novel methodology for the active control of flexible cantilever plates using piezoelectric sensor/actuator pairs. Similarly, Jafari et al. [18] solved a fuzzy differential equation using a feedforward neural network with three hidden layers. Various other differential equations have been solved using the ANN architectures, e.g., [10, 11, 34, 35]. However, much of these recent problems solved in the area of PDEs and boundary value problems are of first order belonging to the linear domain. Thus, the resulting optimisation problem solved within the ANN architecture is constrained, making the application domain rather limited.

The challenge we address in this paper is that we propose a general ANN architecture for solving a broad range of nonlinear boundary value problems. For this purpose, we utilise a deep neural network that can learn patterns from the data [27] using trial solutions as input during the learning phase. Here, we solve an unconstrained minimisation problem instead of solving the traditional forms of constrained minimisation problems [14]. Moreover, the method proposed here is free from meshes, which is a considerable advantage since meshes can be infeasible and costly when solving PDEs in higher dimensions [26]. Here, the convolutional neural network replaces the mesh formulation where trial solutions help to train the network. The first part of the sample data is the necessary boundary conditions of the PDEs, while the second part consists of a feedforward neural system containing flexible parameters or weights, giving rise to the solution [9, 23]. Thus, the approximate trial solution provides precision-controlled coefficients without the need for domain discretisation.

The work we describe here is novel in that we demonstrate that an ANN-based nonlinear machine learning model is used to solve general nonlinear boundary value problems. To our knowledge, this is the first time a general ANN-based solution is proposed for such boundary value problems with diverse applications in mathematics, physics and engineering. Specifically, in this work, we show that neural networks can learn accurately about the solution of PDEs, such as the Poisson's equation and the Biharmonic equation, even when such equations are solved in complicated domains. Notably, one can utilise this approach to solve PDEs arising from boundary value problems without worrying about creating and manipulating complex numerical meshes. We demonstrate the efficiency and superiority of the proposed ANN-based method over commonly utilised numerical methods such as the finite difference and the finite element methods.

The remainder of the paper is structured as follows. In Section 2, we introduce the proposed ANN architecture for solving nonlinear boundary value problems. Subsequently, in that section, we discuss the methodology and provide experimental validation of the proposed method. In Section 3, we provide examples showing the application of the proposed method. More specifically, we solve the Von-Kármán and the Navier-Stokes equation subject to given boundary conditions. Finally, in Section 4, we summarise and conclude the paper.

2 The proposed network architecture

The neural network architecture we have propose for obtaining mesh free solutions of nonlinear boundary value problems is inspired by deep and fully connected feedforward neural networks [6]. The ANN is a map between the input and output of the system, which can be linear or nonlinear [17]. The resulting output is a linear activation. Thus, the ANN defines a map such that $\mathbf{R}^n \rightarrow \mathbf{R}^n$. The system forming the initial/boundary value problem describes this map. In some cases, the learning process of the multi-layer network can be time-consuming and ill-conditioned. An increasing number of input parameters and proper error minimisation techniques can be useful to cope with such difficulties. The ANN we consider here consists of $n + 1$ layers with $0 \leq l_i \leq l$ where layer 0 is the input layer and layer l is the output. The activation functions in the hidden layers can be of general form [36], e.g., sigmoids, rectified linear units, or hyperbolic tangents. For this work, unless otherwise stated, we use sigmoids in the hidden layers.

2.1 A general formulation for solving boundary value problems

A general form of the nonlinear boundary value problem considered here can be formulated such that:

$$F(\mathbf{x}, u(\mathbf{x}), \nabla u(\mathbf{x}), \nabla^2 u(\mathbf{x}), .., \nabla^m u(\mathbf{x})) = \tilde{g}(\mathbf{x}), \quad \mathbf{x} \in D, \tag{1}$$

subject to certain boundary conditions, where $\mathbf{x} = (x_1, x_2, .., x_n) \in \mathbf{R}^n$ is an independent variable vector over the domain, $D \subset \mathbf{R}^n$, $m = \{1, 2, ..., n\}$ is the order of boundary value problem, $\tilde{g}(\mathbf{x})$ is the source term and $u(\mathbf{x})$ is an unknown solution. The function F is considered as non-singular, and all the partial derivatives and higher order derivatives in terms of $\{\mathbf{x}, u, \nabla u\}$ are assumed to be well defined. The finite domain in the n-dimensional space for the independent variables is discretised with a unit h such that:

$$\mathbf{x}_{l\mathbf{a}} = \mathbf{x}_{l0} + \mathbf{a}(l)h, \quad \mathbf{x}_{\mathbf{la}} \in D, \tag{2}$$

where $\mathbf{a}(l) = \mathbf{a}, l = \{1, 2, .., n\}$ is the discrete index vector with each of the components representing the discrete coordinate for the individual variables $\mathbf{x}_l$ and $\mathbf{x}_{l0}$ representing the initial values. Hence, the discretised boundary value problem becomes:

$$F_{la}(\mathbf{x}_{la}, u(\mathbf{x}_{la}), \nabla u(\mathbf{x}_{la}), \nabla^2 u(\mathbf{x}_{la}), .., \nabla^m u(\mathbf{x}_{la})) = \tilde{g}(\mathbf{x}_{la}). \tag{3}$$

To obtain a numerical solution of the boundary value problem, the following method is adopted, which assumes the grid discretisation of the domain D and the boundary value problem is defined as described in (2) and (3), respectively. The unknown function u is a two variable entity, $u = u(\mathbf{x})$, such that it is defined over a two-dimensional domain D with the following trial solution:

$$u_T(\mathbf{x}) = A(\mathbf{x}) + B(\mathbf{x})N(\mathbf{x}, \mathbf{p}), \tag{4}$$

where the weights need to be learned by the neural network $N(\mathbf{x}, \mathbf{p})$ to approximate the solution with $A(\mathbf{x})$ along with the boundary conditions and $B(\mathbf{x})$ satisfying $B(\mathbf{x})|_{\delta D} \equiv 0$. Note that the choice of A and B are not restricted. There are several ways to construct a solution and in [4] one could find further details on how to approach it. The main problem here is how to transform the differential equation into a minimisation problem. Let us explain this further by way of an example as follows.

2.2 Solving poisson's equation

Let us consider the general form of Poisson's equation to be solved as:

$$F(\mathbf{x}, u(\mathbf{x}), \nabla u(\mathbf{x}), \nabla^2 u(\mathbf{x})) = \tilde{g}(\mathbf{x}), \quad \mathbf{x} \in D. \tag{5}$$

Then, the minimisation problem will look like:

$$\min_{\mathbf{p}} \sum_{\mathbf{x} \in D} F(\mathbf{x}, u(\mathbf{x}, \mathbf{p}), \nabla u(\mathbf{x}, \mathbf{p}), \nabla^2 u(\mathbf{x}, \mathbf{p})). \tag{6}$$

At this stage, the problem is changed from a constrained to an unconstrained optimisation problem since the boundary conditions are satisfied by a trial solution already, which makes it easier to handle. Now, the minimised error $\mathbf{x}_{la}$ are points in D such that:

$$E(\mathbf{p}) = \sum_i \left\{ \Delta u(\mathbf{x}_{ia}) - \tilde{g}(\mathbf{x}_{ia}) \right\}^2.$$

Let us consider a multilayer perceptron with m_p input units, the hidden layer with h_a activation function units (which is a sigmoid in our case) and an output unit. For a given input vector $\mathbf{x}$, the output of the network is:

$$N = \sum_{i=1}^{h_a} v_i \sigma(z_i), \quad z_i = \sum_{j=1}^{m_p} w_{ij} x_j + h_i,$$

where w_{ij} is the weight of the input unit, (j), v_i is the weight of the hidden unit, $\sigma(z)$ is the sigmoid transfer function and h_i is the hidden unit i. Then it is straightforward to see that:

$$\frac{\delta^k N}{\delta x_j^k} = \sum_{i=1}^{h_a} v_i w_{ij}^k \sigma(z_i)^{(k)},$$

where $\sigma(z_i)^{(k)}$ denotes the k^{th} order derivative of the sigmoid.

It is worth mentioning that when we compute our loss function, various high order derivatives can be picked into a collection, forming a single unknown function. This collection of unknown functions can include the PDE analytic solution and its derivatives up to the third order, or it can also include only the PDE solution and its second-order derivatives. However, in this case, the loss function is written in the least square sense, bearing a resemblance to the well known least square finite element method [21]. In constructing the architecture, it is important to bear in mind that the PDE solution and its derivatives share nearly the same deep neural network. The structure of the model can be seen in Fig. 1, where we need to update our boundary/trial solution network before updating the main part of the neural network.

2.3 Enabling learning in the neural network

The major challenge with multi-layer perceptron neural networks is the difficulty in measuring the efficiency of the weights within the hidden layers, whereby the aim is to obtain the lowest performance error. Unfortunately, there is no straightforward mechanism to measure the degree of error within the hidden layers during the training. Hence, we

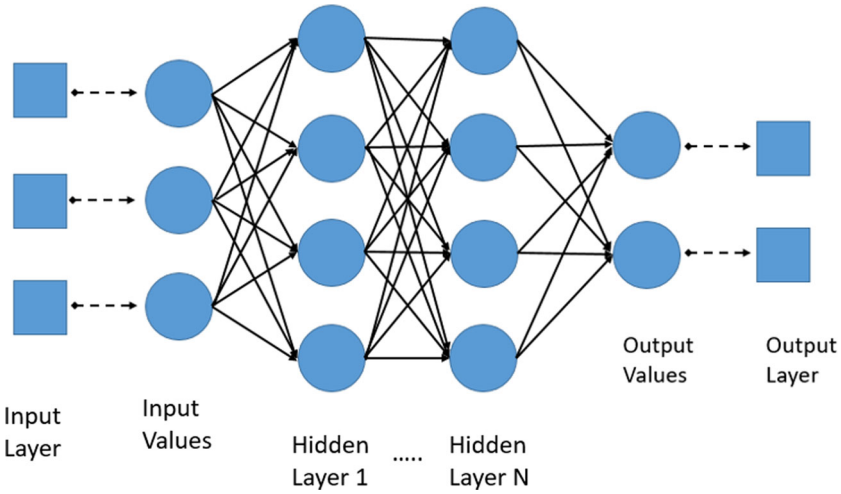

Fig. 1 A structural diagram showing the proposed feedforward neural network. It consists of an input layer, two or more hidden layers, and an output layer containing one or more artificial neurons

propose the following backpropagation learning procedure to be applied during the training phase. It includes:

- the set of sample/domain data points $\{\mathbf{x_{la}}\}$,
- the value for the learning rate α and the criteria to terminate the algorithm,
- and the methodology for updating weights and initial values of the weights.

Note, here, to start with, we take random initial weights lying between -0.5 to 0.5. There are n input nodes that will become $n+1$ total input nodes after including an extra bias node. A single hidden layer consists of h_n nodes with sigmoid as the activation function. The single scalar output from the previous information is then given by:

$$N(x, v, \tilde{W}) = v^t \sigma(\tilde{W}\tilde{x}), \tag{7}$$

where $v \in \mathbf{R}^{h_n}$ and $\tilde{W} \in \mathbf{R}^{h_n \times \{n+1\}}$ are the specific neural network parameters that replaces the general parameter $\mathbf{p}$. The input variable is $\tilde{x} = [x^T, 1]^T$. The function $\sigma : \mathbf{R}^{h_n} \to \mathbf{R}^{h_n}$ represents a component-wise activation function that operates on the hidden layer. The overall task here is to choose the discretised setting given in (2) and various hidden nodes h_n in a layer, and then optimise the (5) to get an approximation of $u_t(x, \tilde{p})$.

As shown in Fig. 1, a feed-forward neural network comprises of an input layer, one or more hidden layers and an output layer. Parameters are propagated from the input layer to the output layer through the hidden layers. The number of nodes in the hidden layer is determined by the degree of polynomial considered. If an nth degree polynomial is used, the number of nodes in the hidden layer would be $n+1$. The coefficients of the polynomial can be used as initial weights from the input to the hidden layers and from the hidden layers to the output layer. The number of hidden layers, the number of neurons per layer and the choice of activation functions are essential considerations we have put in place while designing our neural network. As the number of hidden layers increases, the network can be thought of as an adaptive nonlinear dynamical system composed of numerous neurons connected by various attributes and capable of approximating a wide range of complex functions. This is a key feature of the network we have proposed here. Thus, since an ANN can be used to accurately approximate very complicated functions, it can be regarded as a learnable entity which in our case is regarded as the solution of the PDE in question.

It is important to bear in mind that the number of network parameters, including the weights of internode links and biases associated with hidden and output nodes, should be less than the size of the training set. Otherwise, the network will be susceptible to over-fitting, which means that the predictions on the training set will improve while the power of generalisation will deteriorate. Regularisation methods are used to control over-fitting. This is also a feature which is inherent in the ANN proposed here.

Another feature we have taken onboard is the consideration given in selecting the bias terms. Each neuron in the ANN is supplied with a bias, including the output neurons but excluding the input neurons. The connections between neurons in subsequent layers are represented by matrices of weights. We let b_i^l denote the bias of neuron i in layer l. The weight between neuron k in the layer l_1 and neuron i in the layer l is denoted by w_{ik}^l. The activation function in the layer l is denoted by σ_l, regardless of the type of activation. We assume for simplicity that a single activation function is used for each layer. The output from neuron i in the layer l is denoted by y_i^l.

Drawing parallels with a given numerical method for the accuracy of the solution $\tilde{p}$ obtained, we assume the grid and the hidden layer as the dominant choices in the solution parameter space. Thus, conceptually, the accuracy of the solution is expected to improve with a finer grid and a larger hidden layer. However, this comes with a high computational cost and also with the undesirable effect

of potential over-fitting. Note, the aim here to achieve an estimate of the solution with sufficient accuracy. Whilst we do that, we must bear in mind to minimise the computational cost and complexity of the problem itself. We explain our ANN formulation further by way of another example, as discussed below.

2.4 Solving the biharmonic equation

The Biharmonic operator or bi-Laplacian operator is used in various formulations such as solving the airy stress problem, incompressible fluid mechanics, and elasticity problems [5]. This operator also has applications in geometric design, e.g., [4, 19]. Here, we solve the two dimensional Biharmonic equation in a rectangular domain together with mixed boundary conditions using the method proposed in the previous section. Consider the equation:

$$\begin{aligned} \nabla^4 \Phi(\mathbf{x}) &= \left(\frac{\delta^4}{\delta x^4} + \frac{\delta^4}{\delta x^2 \delta y^2} + \frac{\delta^4}{\delta y^4}\right)\Phi(\mathbf{x}) = \tilde{g}(\mathbf{x}), \\ \Phi(\mathbf{x}) &= \psi_1(\mathbf{x}),\ \mathbf{x} \in \delta\Omega, \\ \frac{\delta \Phi}{\delta n}(\mathbf{x}) &= \psi_2(\mathbf{x}),\ \mathbf{x} \in \delta\Omega, \end{aligned} \tag{8}$$

where $\Omega = [a, b] \times [c, d]$, Φ is a four times continuously differentiable function and $f, \psi_i,\ \{i = 1, 2\}$ are known functions. In our case, $\frac{\delta}{\delta n}$ refers to the normal derivative where n is a unit vector orthogonal to the surface. In other words, if n is the unit vector providing the direction, then the derivative is given as,

$$\frac{\delta \Phi}{\delta n} \equiv \nabla \Phi \cdot n.$$

If $n = (1, 0)$, i.e., in the direction of x-axis, then we just get Φ_x, which is the standard partial derivative in the direction of the x-axis. Similarly, if $n = (0, 1)^T$, then we get Φ_y.

To find the appropriate trial solution, first, we write all the boundary conditions as follows:

$$\begin{aligned} \Phi(a, y) &= \psi_1(a, y)\ , \ \frac{\delta \Phi}{\delta n}(a, y) = -\psi_2(a, y), \\ \Phi(b, y) &= \psi_1(b, y)\ , \ \frac{\delta \Phi}{\delta n}(b, y) = \psi_2(b, y), \\ \Phi(x, c) &= \psi_1(x, c)\ , \ \frac{\delta \Phi}{\delta n}(x, c) = -\psi_2(x, c), \\ \Phi(x, d) &= \psi_1(x, d)\ , \ \frac{\delta \Phi}{\delta n}(x, d) = \psi_2(x, d). \end{aligned} \tag{9}$$

The trial solution is written in a similar way as proposed in the previous section such that:

$$\Phi_t(\mathbf{x}, \mathbf{p}) = \phi_1(\mathbf{x}) + \phi_2(\mathbf{x}, N(\mathbf{x}, \mathbf{p})), \tag{10}$$

where,

$$\phi_1(\mathbf{x}) = \Sigma_{i=1}^{4} A_i x^i + \Sigma_{i=1}^{4} B_i y^i, \tag{11}$$

and

$$\phi_2(\mathbf{x}, N) = (x-a)^2(x-b)^2(y-c)^2(y-d)^2 N(\mathbf{x}, \mathbf{p}), \tag{12}$$

where ϕ_2 vanishes on the boundary. Now, after substituting (11) and (12) in (10) and then in (9), we can find the unknown constants A_i and B_i, for $i = \{1, 2, 3, 4\}$. In practice, we can generate a feasible trial solution satisfying the boundary conditions since the determinant of the coefficient matrix for the defined system is not identically zero. Hence, the minimum error is written as:

$$E(\mathbf{p}) = \sum_i \sum_j \left\{ |\nabla^4 \Phi(\mathbf{x}_{la}) - \tilde{g}(\mathbf{x}_{la})|^P \right\}^{1/P}, \quad i \in 1, 2, .., n,\ j \in 1, 2, .., m,$$

where $\mathbf{x}_{la}$ are points in Ω.

Algorithm 1 The ANN Algorithm for forward and backward propagation for the chosen boundary value problem.

Input: m×n dimensional discretised domain data in both directions,
$\mathbf{x} = [x_1, x_2, ..., x_m]$, $\mathbf{y} = [y_1, y_2, ..., y_n]$.
Output: Deep neural network solution.
Data: PDE, boundary conditions, analytic solution and trial solution.

```
Generate dataset on the boundary and in the inner domain of Ω
for hl := 1 → #HiddenLayers do
    for i := 1 → #RowunitsinLayerhl do
        for j := 1 → #ColoumnunitsinLayerhl do
            Find the layer activation using the sigmoid;
            Compute inputs for the next layer;
W ← Assemble the final output in a matrix, namely weightmatrix;
for h := 1 → #HiddenLayers do
    Find the error in the partial derivatives;
    Find the error in the previous layer;
Calculate loss function;
end
```

The formulation is further described in Algorithm 1. To explain this further, we take a specific case of the Biharmonic equation. We solve it and compare the solution of our proposed neural network based solution with other existing numerical methods such as finite element method (FEM) and finite difference method (FDM) for (14) as shown in Table 1. We consider the following nonlinear Biharmonic equation with non-zero boundary conditions such that:

$$\Psi_{xxxx} + 2\Psi_{xxyy} + \Psi_{yyyy} + \Psi_{xx} + 2\Psi_{xy} + \Psi_{yy} + \Psi_x + \Psi_y + \Psi^2 = \tilde{g}(x, y), \tag{13}$$

Table 1 Comparison between standard numerical methods and our proposed method for the Biharmonic example

Method	Finite Difference Method (FDM)	Finite Element Method (FEM)	Proposed Method (ANN)
Implementation	Centred on the Taylor series approximation of the DE with a uniformly distanced node grid	Centred on the equivalent governing integral relationship using finite segments/elements	Centred on the trial solution approximation of DE satisfying BCs
Advantages	• Mesh less • Easy to apply • No numerical integration involved • Low memory requirements	• Easier for complex and irregular geometries • Symmetrical and sparse matrices • Accurate	• Mesh less • Less computational complexity • Requires very low memory space • No numerical integration • Detection of complex nonlinear relationships between dependent and independent variables implicitly
Disadvantages	• Not suitable for complex geometries • Requires a very fine grid for accuracy • Time consuming • Not suitable for infinite problems	• Large memory requirements for storage of the stiffness matrix • Greater number of calculations • Mesh requirement • Require complicated integral relations	• Gradient calculation • Training can be extensive • Requires trial solutions
Computational time for the Example	2.0421 sec (25×25 grid)	1.9204 sec (289 nodes and 512 triangular meshgrid)	0.9615 sec (25×25 grid)
Computer specifications	i7-7600U CPU, 2.80 GHz 64 bit OS, 16 GB (RAM)	i7-7600U CPU, 2.80 GHz 64 bit OS, 16 GB (RAM)	i7-7600U CPU, 2.80 GHz 64 bit OS, 16 GB (RAM)
Program used	MATLAB	MATLAB	Python (Jupyter)

where,

$$\tilde{g}(\mathbf{x}) = e^x(2\pi^4 - 2\pi^2 + 3)\sin(\pi y) + e^{2x} - e^{2x}\cos(\pi y)^2 + 3e^x\pi\cos(\pi y), \quad (14)$$

with boundary conditions as follows,

$$\Psi(\mathbf{x}) = e^x \sin(\pi y),\ \forall\,(\mathbf{x}) \in [0,1]\times[0,1],$$
$$\frac{\delta\Psi}{\delta x}(\mathbf{x}) = e^x\sin(\pi y),\ x = 0, 1,$$
$$\frac{\delta\Psi}{\delta y}(\mathbf{x}) = \pi e^x\cos(\pi y),\ y = 0, 1.$$

Using the boundary conditions, the trial function is constructed as:

$$\Psi_t(\mathbf{x}) = e^x\sin(\pi y) + x^2y^2(x-1)^2(y-1)^2N(\mathbf{x},\mathbf{p})$$

The ANN solution and the analytical solution for the chosen Biharmonic problem are shown in Figs. 2 and 3, respectively. Figure 4 shows a detailed comparison of our proposed approach with the FEM and FDM through the use of L^2 error. This error is calculated for different grid sizes starting from 5×5 to 30×30.

For further comparison of the errors between various methods compared here, we have also utilised the root-mean-square (RMS) error:

$$\text{Error} = \sqrt{\frac{\sum_{i=1}^{N}(T_i - R_i)^2}{N}},$$

where N is the total number of nodes, T_i is the exact solution, and R_i is the numerical solution at the particular

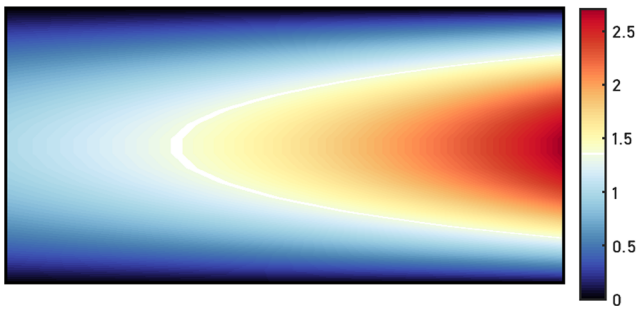

Fig. 2 The ANN solution for the example Biharmonic problem

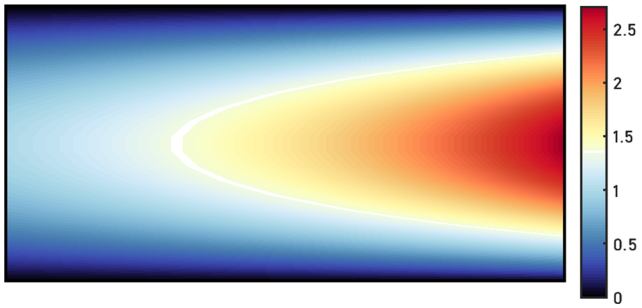

Fig. 3 The exact solution for the example Biharmonic problem

node i. The error values for the Biharmonic example is shown in Table 2.

2.5 Effects of the hidden layers

An optimum number of hidden layers is an important and critical requirement when deciding to create a chosen ANN framework. The key idea is to efficiently calculate the number of hidden units needed in a multi-layer network to obtain a particular approximation. For this purpose, we work out the number of independent variables that should be modified to create an arbitrary smooth function for a given order of approximation. At the same time, we also work out the number of values that will be changed based on the overall number of parameters in the network. Figure 5 shows the effect of the hidden layers as we change the discretisation in the x and y directions.

Our experiments, for this example and for a number of other examples, show that the use of 10 hidden layers results in good agreement of the ANN-based solution when compared with the exact and the numerical solutions. The important point to be noted here is that as we increase the number of hidden layers and discretisation across both axes, the training time increases. Therefore, we suggest using the number of hidden layers between 5 and 10, depending on the training data and the complexity of the problem.

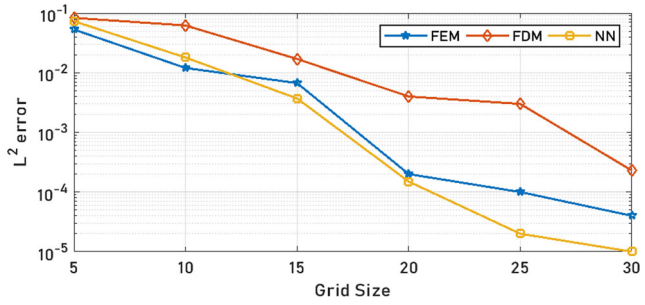

Fig. 4 L^2 errors in the solutions of the example Biharmonic problem

Table 2 Comparison of the RMS error values in the solution of the Biharmonic example

Hidden Layers	FEM (RMS error)	FDM (RMS error)	ANN error
5	5e-08	2e-07	3e-08
10	5e-08	2e-07	7e-09

3 Examples

So far, we have discussed how the ANN for solving the chosen boundary value can be setup. We have also shown how the method can be utilised to solve simpler yet general problems such as the Poisson's and the Biharmonic equation. In this section, we show how the ANN framework discussed above can be used to solve two further boundary value problems namely, the the Von-Kármán equation and the Naiver-Stokes Equation. Both these problems are of complex nature and of practical importance.

3.1 Solving the Von-Kármán equation

The analysis of large square and rectangular plate deflections is one of the most studied architectural problems. The applications of this problem include the design of aviation structures, ship hulls, bridges and spacecraft. For example, the lateral loads on an aircraft are subjected to the thin plate effects in the pressure cabin or lift and bending on the wings.

The generic Kirchhoff linear plate bending theory is appropriate only for minor deflections that neglect mid-surface strains and subsequent in-plane stresses. With a relatively strong lateral deflection, the external force increases. Both the plate's central surface and the membrane forces play a major role in moving the side loads. Initially, the Von-Kármán's extension to large deformations was assisted through the pioneering work of Leitao [24]. In film relationships, we must account for the nonlinear terms for the sake of large flow. Consequently, we obtain two nonlinear fourth-order cross-displacement equations for which there is no analytic solution.

Consequently, to solve this problem, one has to resort to numerical procedures. For example, the FDM technique was used in [7] to examine large deflections of rectangular plates subject to lateral pressure and edge loading combination. In [12], the authors discuss some of the earlier final component implementations for large deformations. In [33], the stress-based finite element approach was introduced. In [32], a boundary element method helps to analyse the static, dynamic and buckling behaviour of thin plates. These numerical methods can solve the Von-Kármán plate equations and are more versatile than the rival semi-analytical methods. However, they all come with high computational costs.

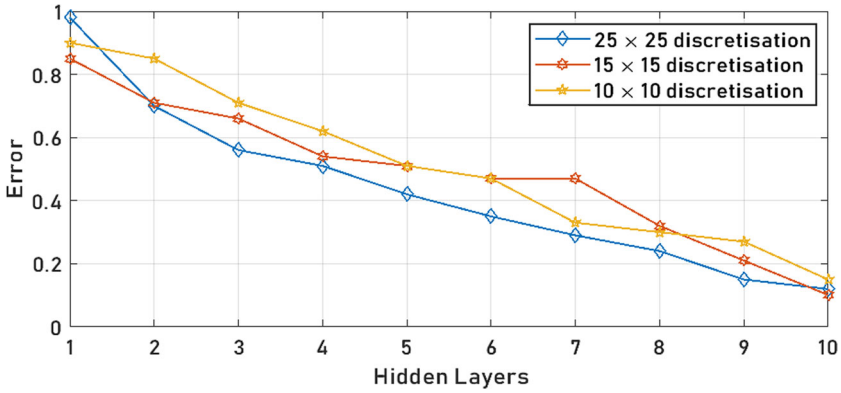

Fig. 5 The effect of the hidden layers compared to levels of domain discretisation

3.1.1 Problem formulation

Let us define a side-by-side general plate system composed of non-homogeneous elastoplastic plates with varying physical properties in the domain $\Omega \in \mathbf{R^2}$ such that:

$$\Omega = \cup_{i=1}^{k} \Omega_i = \{\mathbf{x} \mid a^{k-1} \leq x \leq a^k,\ 0 \leq y \leq b,\ a^0 = 0\},$$

where $k > 0$ is the number of plates that form the system. The problem of bending of the system with the discontinuous coefficient is written as:

$$D_i \left[\frac{\delta^4 \Phi}{\delta x^4} + \nu_i \frac{\delta^4 \Phi}{\delta y^4} + \frac{\delta^4 \Phi}{\delta y^4} + \nu_i \frac{\delta^4 \Phi}{\delta x^4} + 2(1-\nu_i) \frac{\delta^4 \Phi}{\delta x^2 \delta y^2} \right] = F(\mathbf{x}), \tag{15}$$

where Φ is the bending of the beam corresponding to the vertically applied force F, $D_i = \frac{E_i h^3}{12(1-\nu_i^2)}$ are the cylindrical stiffness coefficients of the plates of the system, E, μ and h represents Young's modulus, Poisson's constant, and the thickness of the beam (assumed to be uniform throughout). Due to the plate boundaries, the boundary conditions for the equilibrium condition are defined to be the physical conditions such that it is assumed that the clamped boundary and the boundary condition are simply assisted. Though the plates in the system have different cylindrical stiffness coefficients of D_i, the coefficients of (15) lead to discontinuity at the common bounds of the system. Let us consider two plates with different properties, i.e., $i = 2$ in a rectangular domain, $\Omega = \{\mathbf{x} = (x, y) \mid -l_1 \leq x \leq l_1, 0 \leq y \leq w_1\}$, as shown in Fig. 6. The case considered is with the boundary at $x = 0$. With that, the following functions can be easily seen to satisfy the simply supported boundary condition, the clamped boundary condition of Ω and the common boundary, respectively, such that:

$$\begin{aligned} \Phi_{t1}(\mathbf{x}) &= -\sin \pi x \sin \pi y + x^2 y^2 (x+l_1)^2 (y-w_1)^2 N(\mathbf{x}, \mathbf{p}), \\ \Phi_{t2}(\mathbf{x}) &= -(1-\cos 2\pi x)(1-\cos 2\pi y) + x^2 y^2 (x-l_2)^2 (y-w_1)^2 N(\mathbf{x}, \mathbf{p}). \end{aligned} \tag{16}$$

The function $\Phi(\mathbf{x}) = \{\Phi_{t1} \cup \Phi_{t2}\}$ is geometrically equivalent such that the plates are connected with an ideal hinge on the common boundary points, and there is absolute rigid support under the hinge. The function $\Phi(\mathbf{x})$ also corresponds to the bending along the common boundary. This means that the plates are composed of welding, and there is absolute rigid support under the hinge.

To examine the effect of the Young's modulus on the bending of the system, we solve the problem using conditions whereby $E_1 = 5000\, kn/cm^2$, $E_2 = 3000\, kn/cm^2$ and $E_1 = 3000\, kn/cm^2$ and $E_2 = 5000\, kn/cm^2$ and shown in Fig. 7. The ANN-based solution is derived using 6 hidden layers. A comparison is carried out with the FEM, which shows an L^2 error of $1.3413e-08$. The error between the ANN solution and the analytical solution is $1.066e-11$, with 7 hidden layers and a grid size of 20×20.

3.2 Solving the naiver-stokes equation

Here we present an efficient deep learning technique relying on the ANN for the model reduction of the Navier-Stokes Equation [3] for incompressible flow. Consider a two dimensional Navier-Stokes Equation with the continuity and

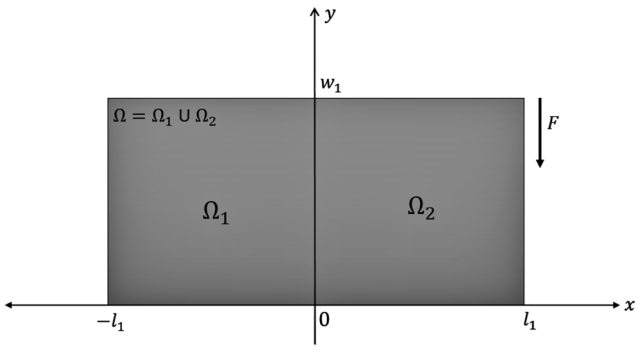

Fig. 6 Plate system of two beams with load F

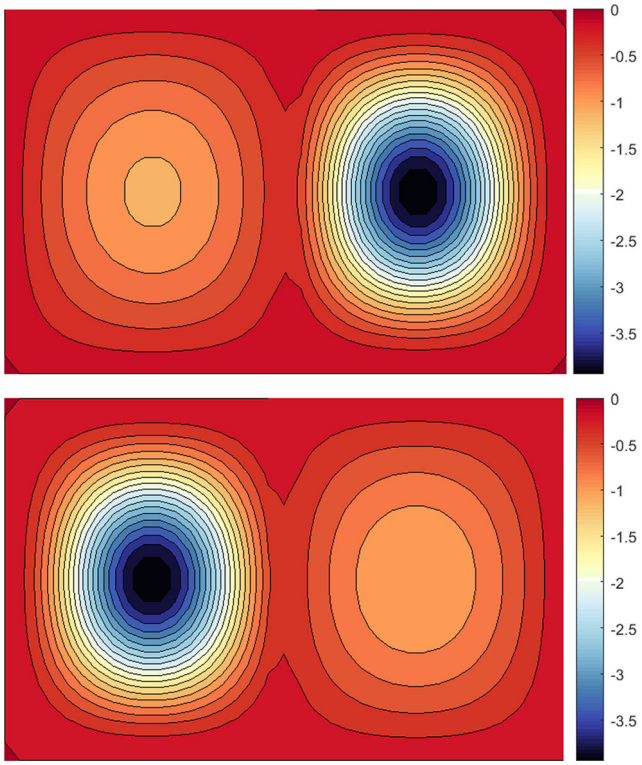

Fig. 7 Numerical solution with $E_1 = 5000$ and $E_2 = 3000$ in the top figure, $E_1 = 3000$ and $E_2 = 5000$ in the bottom figure, and $\nu_i = 0.3, i = 1, 2$

momentum elements describing an incompressible laminar flow in the non-dimensional form such that:

$$\begin{aligned} -\nu\triangle\mathbf{v} + (v\cdot\triangle)\mathbf{v} + \nabla p = f \quad &\text{(momentum equation)},\\ \nabla\cdot(\mathbf{v}) = 0 \quad &\text{(continuity equation)}, \end{aligned} \tag{17}$$

where $\Omega = \{(x, y) = [a, b] \times [c, d]\} \in \mathbf{R^2}$, $\mathbf{v}$ is the velocity vector, p is pressure, ν is the viscosity, and f represents body forces. We also introduce a stream function $\Psi(x, y)$ such that it satisfies the continuity equation:

$$u = \frac{\delta\Psi(x,y)}{\delta y} \text{ and } v = -\frac{\delta\Psi(x,y)}{\delta x}. \tag{18}$$

Using the stream function from the (18), we can rewrite the equation after eliminating the pressure term from the momentum equations, leading to the vorticity transport equation with two boundary conditions such that:

$$\frac{1}{Re}\triangle^2\Psi - \frac{\delta\Psi}{\delta y}\triangle\frac{\delta\Psi}{\delta x} + \frac{\delta\Psi}{\delta y}\triangle\frac{\delta\Psi}{\delta y} = \frac{\delta f_1}{\delta y} - \frac{\delta f_2}{\delta x}. \tag{19}$$

For this problem, we define the trial solution as:

$$\Psi_t(\mathbf{x}) = (x(x-1)y(y-1))^2 + x^2y^2(x-1)^2(y-1)^2N(\mathbf{x}, \mathbf{p}).$$

In this example, we have divided the domain space $[0, 1]^2$ to a 16×16 grid. Figure 8 shows the ANN solution of the defined Naiver-stokes problem. The L^2 error between the FEM and the analytical solution is $1.213e-07$, and that for the ANN solution (using 7 hidden layers) and the corresponding analytic solution is $1.560e-11$.

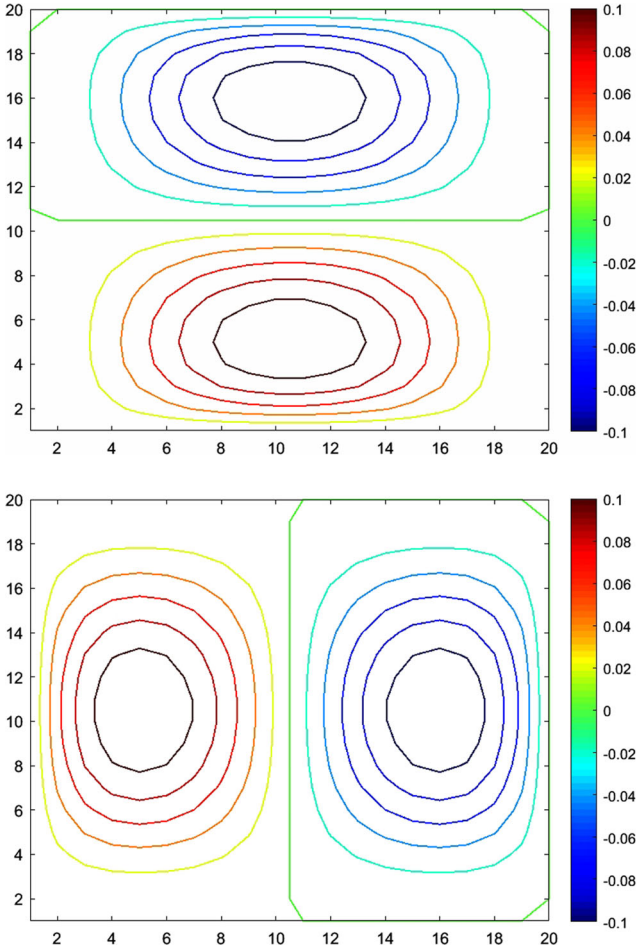

Fig. 8 Numerical velocity $\mathbf{u}$ and $\mathbf{v}$, respectively

4 Conclusions

In this paper, we discuss a method of solving nonlinear boundary value problems. Our proposed approach relies on the function approximation capabilities of the feedforward neural networks. Thus, we have designed and trained an ANN framework capable of solving general nonlinear boundary value problems whereby the resulting solutions are mesh free.

We can use the proposed method to solve general nonlinear boundary value problems whose analytic solutions are not readily available. That is a significant advantage of this method. Other benefits of this method include its generalisability for solving higher-order and nonlinear problems. For example, in this paper, we have described how the ANN

framework proposed can be utilised for solving the Navier–Stokes equation [13, 15, 28] and the Von-Kármán equation [8].

Generating numerical solutions of nonlinear boundary value problems, especially for higher-order PDEs, is a challenging task. However, the method proposed here appears to be feasible to address such critical problems with good agreement with the corresponding analytic solutions and state of the art techniques for obtaining numerical solutions. We show how it is feasible to set up the ANN framework to incorporate the necessary initial/boundary conditions for the chosen problem and optimise the chosen ANN parameters as well as weights using trial functions. The use of trial functions in the training process helps to provide precision-controlled coefficients without the need for complex domain discretisation. We have also examined the performance of the network subject to different numbers of hidden layers on the chosen ANN. According to our studies, the number of hidden layers between 5 and 10 provides the optimum results.

From the studies we have described here, one can conclude that there is great potential for the ANN framework presented in this paper to be further studied and extended. It will be useful to experiment with various optimisation techniques to reduce the number of parameters in the network. That will help in further optimising the training process and computation of the solutions. It will also be helpful to combine the proposed ANN framework with other existing and pre-trained deep learning models so that solving more complex boundary value problems can be carried out accurately and efficiently. Finally, it will be useful to extend the proposed ANN framework to solve PDEs in very high dimensions, e.g., dimensions > 100. Solving very high dimensional boundary value problems is a highly challenging problem using the present state of the art FEM and FDM based techniques.

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Abdeljaber O, Avci O, Inman DJ (2016) Active vibration control of flexible cantilever plates using piezoelectric materials and artificial neural networks. Journal of Sound and Vibration 363:33–53
2. Acharya UR, Fujita H, Oh SL, Hagiwara Y, Tan JH, Adam M, Tan RS (2019) Deep convolutional neural network for the automated diagnosis of congestive heart failure using ECG signals. Appl Intell 49:16–27
3. af Klinteberg L, Askham T, Mary KsCK (2020) A fast integral equation method for the two-dimensional Navier-Stokes equations. J Comput Phys 409:109353
4. Aggarwal R, Ugail H (2019) On the solution of poisson's equation using deep learning. In: 2019 13th international conference on software, knowledge, information management and applications (SKIMA). IEEE, pp 1–8
5. Artioli E, de Miranda S, Lovadina C, Patruno L (2017) A stress/displacement virtual element method for plane elasticity problems. Comput Methods Appl Mech Eng 325:155–174
6. Baggenstoss PM (2019) On the duality between belief networks and feed-forward neural networks. IEEE Transactions on Neural Networks and Learning Systems 30(1):190–200
7. Brown JC, Harvey JM (1969) Large deflections of rectangular plates subjected to uniform lateral pressure and compressive edge loading. J Mech Eng Sci 11(3):305–317
8. Brunetti M, Favata A, Paolone A, Vidoli S (2020) A mixed variational principle for the FÖPpl–Von KÁRmán equations. Appl Math Model 79:381–391
9. Charniak E (2019) Introduction to deep learning. The MIT Press
10. Chaudhari P, Oberman A, Osher S, Soatto S, Carlier G (2018) Deep relaxation: partial differential equations for optimizing deep neural networks. Research in the Mathematical Sciences, 5(30)
11. Chen TQ, Rubanova Y, Bettencourt J, Duvenaud DK (2018) Neural ordinary differential equations. In: Advances in neural information processing systems, pp 6571–6583
12. Ciarlet PG (2002) The finite element method for elliptic problems. Society for Industrial and Applied Mathematics
13. Farrell PE, Mitchell L, Wechsung F (2019) An augmented lagrangian preconditioner for the 3d stationary incompressible navier–stokes equations at high reynolds number. SIAM J Sci Comput 41(5):A3073–A3096
14. Gill PE, Murray W, Wright MH (2019) Practical optimization. Society for Industrial and Applied Mathematics
15. Girault V, Raviart P-A (1979) Finite element approximation of the navier-stokes equations. Lecture Notes in Mathematics 749. Berlin, Heidelberg, New York, Springer Verlag
16. Hinton B, Ma L, Mahmoudzadeh AP, Malkov S, Fan B, Greenwood H, Joe B, Lee V, Kerlikowske K, Shepherd J (2019) Deep learning networks find unique mammographic differences in previous negative mammograms between interval and screen-detected cancers: a case-case study. Cancer Imaging 19:41. https://doi.org/10.1186/s40644-019-0227-3
17. Hornik K, Stinchcombe M, White H (1990) Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. Neural Networks 3(5):551–560
18. Jafari R, Razvarz S, Gegov A (2018) Fuzzy differential equations for modeling and control of fuzzy systems. In: International conference on theory and applications of fuzzy systems and soft computing. Springer, pp 732–740
19. Jha RK, Ugail H, Haron H, Iglesias A (2018) Multiresolution discrete finite difference masks for rapid solution approximation of the poisson's equation. In: 2018 12th international conference on software, knowledge, information management & applications (SKIMA). IEEE, pp 1–7

20. Jha S, Dey A, Kumar R, Kumar V (2019) A novel approach on visual question answering by parameter prediction using faster region based convolutional neural network. IJIMAI 5(5):30–37
21. Jiang B, Povinelli LA (1990) Least-squares finite element method for fluid dynamics. Comput Methods Appl Mech Eng 81(1):13–37
22. Jin Y, Li Y (2019) Neural network approximation for nonlinear partial differential equations with quasi-newton optimization and piecewise strategy. In: 2019 IEEE 31st international conference on tools with artificial intelligence (ICTAI). IEEE, pp 1643–1647
23. Khan A, Sohail A, Zahoora U, Qureshi AS (2020) A survey of the recent architectures of deep convolutional neural networks, vol 53, pp 5455–5516
24. Leitao VMA (2001) A meshless method for kirchhoff plate bending problems. International Journal for Numerical Methods in Engineering 52(10):1107–1130
25. Li X, Zhou Y, Du P, Lang G, Xu M, Wu W (2020) A deep learning system that generates quantitative CT reports for diagnosing pulmonary tuberculosis. Applied Intelligence
26. Liu GR (2016) An overview on meshfree methods: for computational solid mechanics. International Journal of Computational Methods 13(05):1630001
27. Liu W, Wang Z, Liu X, Zeng N, Liu Y, Alsaadi FE (2017) A survey of deep neural network architectures and their applications. Neurocomputing 234:11–26
28. Madhusudanan A, Illingworth SJ, Marusic I (2019) Coherent large-scale structures from the linearized navier–stokes equations. J Fluid Mech 873:89–109
29. Maheshan MS, Harish BS, Nagadarshan N (2020) A convolution neural network engine for sclera recognition. International Journal of Interactive Multimedia & Artificial Intelligence, 6(1)
30. Manju N, Harish BS, Nagadarshan N (2020) Multilayer feed-forward neural network for internet traffic classification. IJIMAI 6(1):117–122
31. Moayedi H, Mosallanezhad M, Rashid AS, Jusoh MAM, Wan AWJ (2020) A systematic review and meta-analysis of artificial neural network application in geotechnical engineering: theory and applications. Neural Comput Applic 32:495–518
32. O'Donoghue PE, Atluri SN (1987) Field/boundary element approach to the large deflection of thin flat plates. Computers & Structures 27(3):427–435
33. Punch EF, Atluri SN (1986) Large displacement analysis of plates by a stress-based finite element approach. Computers & Structures 24(1):107–117
34. Raissi M, Perdikaris P, Karniadakis GE (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
35. Ruthotto L, Haber E (2019) Deep neural networks motivated by partial differential equations. Journal of Mathematical Imaging and Vision 62:352–364. (2020). https://doi.org/10.1007/s10851-019-00903-1
36. Sonoda S, Murata N (2017) Neural network with unbounded activation functions is universal approximator. Appl Comput Harmon Anal 43(2):233–268
37. Tkáč M, Verner R (2016) Artificial neural networks in business: two decades of research. Appl Soft Comput 38:788–804
38. Winovich N, Ramani K, Lin G (2019) Convpde-uq: convolutional neural networks with quantified uncertainty for heterogeneous elliptic partial differential equations on varied domains. J Comput Phys 394:263–279

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Riya Aggarwal is a research scholar at the University of Newcastle, Australia. Her research interests include the development of algorithms for the strain field reconstruction measures using neutrons, machine learning and data analysis. She received her BSc. (Mathematics) and Masters (Applied Mathematics) from India in the year 2014 and 2016, respectively.

Hassan Ugail is the director of the Centre for Visual Computing in the Faculty of Engineering and Informatics at the University of Bradford, UK. He has a first class BSc Honours degree in Mathematics from King's College London and a PhD in the field of geometric design from the School of Mathematics at the University of Leeds. Professor Ugail's research interests include computer based geometric and functional design, imaging and machine learning.

Ravi Kumar Jha is a project scientist at the Indian National Centre for Ocean Information Services (INCOIS), Hyderabad, India. His research includes data analysis and management, quality control and visualisation. He works in various fields of Applied science and Applied Mathematics. He has bachelors in Mathematical Science and a masters degree in Applied Mathematics.