Available online at www.sciencedirect.com

ScienceDirect

Comput. Methods Appl. Mech. Engrg. 369 (2020) 113217

Computer methods in applied mechanics and engineering

www.elsevier.com/locate/cma

Data-driven inverse modelling through neural network (deep learning) and computational heat transfer

Hamid Reza Tamaddon-Jahromi, Neeraj Kavan Chakshu, Igor Sazonov, Llion M. Evans, Hywel Thomas, Perumal Nithiarasu*

Zienkiewicz Centre for Computational Engineering, Swansea University, Swansea SA1 8EN, United Kingdom

Received 5 December 2019; received in revised form 10 June 2020; accepted 14 June 2020
Available online xxxx

Abstract

In this work, the potential of carrying out inverse problems with linear and non-linear behaviour is investigated using deep learning methods. In inverse problems, the boundary conditions are determined using sparse measurement of a variable such as velocity or temperature. Although this is mathematically tractable for simple problems, it can be extremely challenging for complex problems. To overcome the non-linear and complex effects, a brute force approach was used on a trial and error basis to find an approximate solution. With the advent of machine learning algorithms it may now be possible to model inverse problems faster and more accurately. In order to demonstrate that machine learning can be used in solving inverse problems, we propose a fusion between computational mechanics and machine learning. The forward problems are solved first to create a database. This database is then used to train the machine learning algorithms. The trained algorithm is then used to determine the boundary conditions of a problem from assumed measurements. The proposed method is tested for the linear/non-linear heat conduction, convection–conduction, and natural convection problems in which the boundary conditions are determined by providing three, four, and five temperature measurements. This study demonstrates that the proposed fusion of computational mechanics and machine learning is an effective way of tackling complex inverse problems.

Keywords: Inverse modelling; Computational mechanics; Machine learning; Heat conduction; Heat convection–conduction; Natural convection

1. Introduction

Inverse modelling is an important area within the engineering and scientific community. The inverse problem, which occurs in the modelling of engineering systems, differ in important ways from the forward (direct) problem. The forward problem is practically well posed and a numerical evaluation of the solution can be accomplished with any specified accuracy of computation with prescribed boundary conditions. In contrast, the inverse problem principally refers to the estimation of an unknown boundary by using measurements (observations) taken from inside the domain geometry. A practical issue faced in the application of inverse modelling for parameter estimation is ill-posedness that is characterised by instability and non-uniqueness of the solution [1]. Given a linear thermal conduction model, there is certainly a single, unique solution of Laplacian's equation which describes the

* Corresponding author.
E-mail address: P.Nithiarasu@swansea.ac.uk (P. Nithiarasu).

https://doi.org/10.1016/j.cma.2020.113217

temperature field behaviour. A numerical evaluation of that solution can be accomplished with any specified accuracy of computation with prescribed temperature boundary conditions. Unlike direct problems, solutions obtained from inverse problems are not unique and optimisation is generally needed to obtain results within a small region of uncertainty. In this case, some heuristic methods of solution for inverse problems and choosing regularisation parameters [2,3] are formalised in terms of their capabilities to treat ill-posed (unstable) problems. The basis of such formal methods resides on the idea of reformulating the inverse problem in terms of an approximate well-posed problem, by utilising some kind of regularisation (stabilisation) technique. The objective of a heuristic method is to produce a solution in a reasonable time frame that is good enough for solving the problem at hand. This solution may not be the best of all the solutions to this problem, but it is still valuable because finding it does not require an excessively long time [4,5]. Note that, it is possible to obtain a solution to the inverse problem by employing a brute-force approach [6]. However, when considering thousands of parameters, the computational expense of brute-force approaches makes them unfeasible to use.

Inverse problems arise in a wide range of applied sciences. Jaluria [7] discussed several inverse problems that arise in a variety of practical processes and presented some of the approaches used to solve them and obtained acceptable and realistic results. These problems included the heat treatment process, wall temperature distribution in a furnace and transport in a plume or jet involving the determination of the strength and location of the heat source by employing a few selected data points downstream. This author employed an optimisation procedure based on Lagrange multipliers [8] to narrow the uncertainty in the non-unique solutions and obtain an essentially unique wall temperature distribution. In addition, Bangian-Tabrizi and Jaluria [9] developed a method based on a search and optimisation approach to solve the inverse natural convection problem of a two-dimensional heat source on a vertical flat plate. There, the data was used to find a function that captures the trends of the temperature on the plate (forward problem). These temperature functions were then used along with Particle Swarm Optimisation (PSO) [10], a search based optimisation algorithm, to find the appropriate locations for sensors on the plate in order to monitor the temperature and pinpoint the location and the strength of the unknown source on the same plate. Moreover, Voronin and Zaroli [11] presented several techniques in the field of inverse modelling, with aim to give an idea of when and how these techniques should be used for linear and non-linear applications. In addition, Yaman et al. [12] introduced inversion-based engineering applications and investigated some of the important ones from a mathematical point of view. These authors introduced the main ideas of many algorithms whose aims were to find locations, shapes, and boundary parameters of obstacles. Furthermore, Szénási and Felde [13] presented a parallel algorithm implemented on graphics accelerators to solve the two-dimensional inverse heat conduction problem based on PSO method.

Trial and error learning in decision problems has traditionally been the main method in data analysis, an approach that becomes impossible when datasets are large. Machine Learning (ML) is a potentially more efficient and accurate approach to the solution by proposing clever alternatives to analysing huge volumes of historical data to forward-looking predictive models. ML has attracted strong interest over many years and is a standard tool today in many applied parts of science. The main advantage of ML lies in that the computer can achieve the purpose of self-learning and predict the trend through operating algorithms. Because of this feature, the computer can be continuously trained, the training dataset can be increased, and over time more accurate results can be obtained through data accumulation by developing fast and efficient algorithms [14,15]. The main difference between ML and conventionally programmed artificial intelligence (AI) algorithms is the ability to process data without being explicitly programmed. This means that an engineer is not required to provide instructions to a machine on how to treat each type of data record. Instead, a machine defines these rules itself relying on input data. Therefore, it can be seen that the research on and selection of the algorithms are the most important part of ML on the whole [16].

Deep Learning (DL), a branch of ML, has emerged recently that uses multiple layers to progressively extract higher level features from raw input [17,18]. DL can mainly be attributed to the massive amounts of available training data and the significantly increased computational power allowing the training of deep neural networks. Compared to traditional ML methods, DL is about "deeper" neural networks that provide a hierarchical representation of the data. Deep learning finds the correct mathematical manipulation to turn the input into the output, whether it be a linear relationship or a non-linear relationship [18].

1.1. Fusion of computational mechanics and machine learning

Artificial Neural Networks (ANN) have been successfully applied to some research fields in computational mechanics over the last three decades [19]. The practical application of complex engineering systems can be limited by the lack of required data. Data-driven models based on ML approach are valid candidates as an alternative to physically based models. These data-driven approaches are able to find highly complex and non-linear patterns in data of different types and sources, then transform raw data to feature spaces, so-called models. These models are then applied for predictions, rather than using massive computer resources to find approximate solutions for a variety of partial differential equations. There has been a steady growth in the use of ANN concepts in the computational mechanics area in recent years [20–22]. Yang [20] pointed out some advances in ANN and its successes in dealing with a variety of important thermal problems. In this work, some current ANN shortcomings, the development of advances in ANN-based hybrid analysis, and its future prospects have been indicated. Guo et al. [21] proposed a general approximation model for real-time prediction of non-uniform steady laminar flow in a 2D or 3D domain based on Convolutional Neural Networks (CNNs) [22]. These authors showed that CNNs would estimate the velocity field two orders of magnitude faster than a GPU-accelerated CFD solver and four orders of magnitude faster than a CPU-based CFD solver at a cost of a low error rate. Moreover, Oishi and Yagawa [23] focused on the rule extraction capability of deep learning, showing the possibility of its new application in computational mechanics in which conventional neural networks had never been successful, including the optimisation of numerical quadrature of the FEM stiffness matrix. Furthermore, Chanda et al. [24] used ANN with genetic algorithms to solve inverse modelling for heat conduction problems. These authors focused on validating and comparing a technique for estimation of principal thermal conductivities of anisotropic composite materials. In addition, Kirchdoerfer and Ortiz [16] investigated the performance of the data-driven solver with the aid of two particular examples of application in the context of computational mechanics: the static equilibrium of non-linear three-dimensional trusses and of finite-element discretised linear-elastic solids.

There is a significantly large number of application scenarios that could benefit from integration of data science and ML with computational mechanics.

Supervised Machine Learning (SML) provides a powerful tool to classify and process data using a learning algorithm to train a model and generate a prediction for the response to new data or the test dataset [17]. In the present paper, the effectiveness of SML is evaluated with various heat transfer mechanisms including the linear/non-linear heat conduction, convection–conduction, and non-linear natural convection problems. For the datasets in questions, it was decided that the SML approach was more appropriate that unsupervised ML. Besides capturing the input–output relationships in the training set, the approximated function can be used to map new input examples supplied in a test dataset. Then, the boundary conditions are determined by providing data at selected points within the model domain. ML algorithms do this by introducing a model with flexible parameters which are learned from the training dataset, often by adjusting the parameters to minimise a 'cost function' which measures the distance of the inferred mapping from the actual one [14–19]. In this work, the mean squared error is used for loss function when compiling the model. The appropriate accuracy function is inferred automatically from the loss function.

1.2. Aim of the work

The aim of the current work is to demonstrate that ML techniques can be used in modelling and solving ill-posed inverse problems that arise in many challenging applications in engineering and industrial applications.

It is also required to investigate possible ways to accelerate the existing conventional ML algorithms and to identify new ML techniques, which can compete with the traditional ones in model accuracy.

In particular, DL approaches using neural networks with multiple internal layers have become popular over the last decade. The proposed approach aims to design and implement optimal Neural Networks for inverse problems in linear and non-linear systems.

2. Methodology

In this section, methods and machine learning algorithms used for inverse analysis of benchmark heat transfer problems are defined and explained.

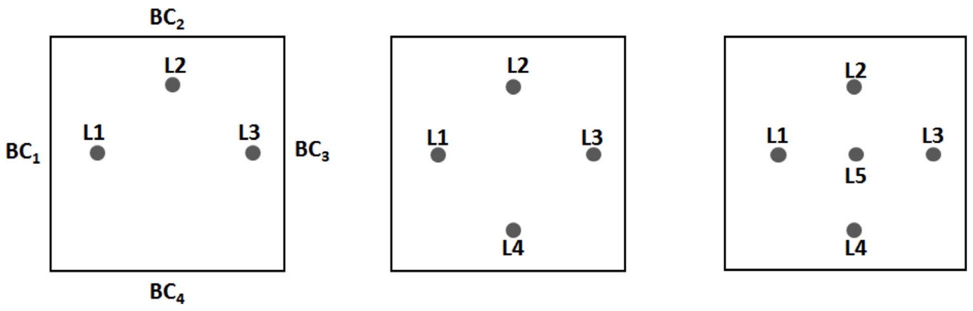


Fig. 1. Locations and coordinates of three, four, and five measured temperatures inside a unit square domain, (0, 1)×(0, 1); L1(0.2, 0.5), L2(0.5, 0.8), L3(0.8, 0.5), L4(0.5, 0.2), L5(0.5, 0.5).

2.1. Neural network architecture

Benchmark inverse problems, relating to heat transfer (linear and non-linear conduction, convection–conduction, natural convection), [25,26], are solved using Multi-Layer Perceptions. The architecture of these neural networks have been designed, using a trial and error method, to allow for mostly 'shallow' learning, networks having less than four hidden layers. However, shallow networks are unable to reach the same levels of accuracy compared with deep networks with the same number of network parameters [27]. In the present work, there are two to four fully-connected hidden layers with 4 to 64 neurons in the network, in addition to the input and output layers. For the inverse problem, measured temperatures are given inside the domain geometry (square) probing mainly with three, four, and five arbitrary locations, see Fig. 1. Note that, for the natural convection problems, more than five temperature locations are also considered, see Section 3.3. These temperature locations are used to estimate the unknown four boundary temperatures ($BC_1 - BC_4$).

The general structure or configuration of the proposed neural network consists of L number of hidden layers along with one input and one output layer. Each hidden layer, input layer and output layer have K, N and J number of neurons respectively. Eq. (1) shows the general structure of the neural network:

$$\mathrm{BC}_j = \sigma\left(\sum_{m=1}^{K^L} W_{jm}^L G_m^L\right), \qquad j = 1, \dots, 4 \tag{1}$$

where G_k^1 is

$$G_k^1 = f\left(\sum_{n=1}^{N} W_{kn}^1 T_n + B_k^1\right), \qquad k = 1, \dots, K^1 \tag{2}$$

and G_k^l is

$$G_k^l = f\left(\sum_{m=1}^{K^{l-1}} W_{km}^l G_m^{l-1} + B_k^l\right), \qquad \begin{array}{l} k = 1, \dots, K^l \\ l = 2, \dots, L \end{array} \tag{3}$$

where $\sigma(x) = \frac{1}{1+\exp(-x)}$ and $f(x) = \tanh(x)$ are non-linear activation functions for the output layer and hidden layers respectively, W are weights and B are biases, or trainable parameters. The input layer takes temperature values (T_n) measured at N field locations, thus has N number of input neurons. Here, the output layer produces values calculated for 4 boundary conditions, hence number of neurons in this layer, J, is four and each of them is activated using a sigmoid activation function whilst a Tanh (hyperbolic tangent) activation function is used to activate values from hidden layers in which the output of each neuron is permitted to be both positive and negative in the interval $[-1, 1]$. Network weights and biases of Neural Networks (NNs) are tuned based on data using the adaptive moment estimation (Adam) [28] algorithm. The main part of the ANN methodology is the learning or training process in which the errors determined at the output layer are successively reduced by adjusting the weights and biases throughout the network. The back-propagation algorithm changes the weights towards a lower

Table 1
ANN parameters — heat transfer problem.

Number of neurons in the input layer	3–10
Number of neurons in the output layer	4
Number of hidden layers	2–4
Number of neurons in the hidden layers	4–64
Input and hidden activation layers	Tanh (hyperbolic tangent)
Output activation layer	Sigmoid function
Number of samples in dataset	15,000
Batch-size	3000
Number of epochs	100–10,000
Validation-split	0.3
Optimiser	Adam

error at the end. Thus, the aim during the training process is to minimise the loss function, $f_{loss(training)}$, which is the deviation between the target value versus the predicted value, by taking the regularisation into account:

$$f_{\text{loss(training)}} = \frac{1}{J}\sum_{j=1}^{J}\left[BC(j)_{\text{Target}} - BC(j)_{\text{Pred}}\right]^2, \qquad J = 4 \tag{4}$$

The detailed network configuration and the parameters used for the heat transfer problems are shown in Table 1. In this study, dataset containing 15,000 samples each is generated for different heat transfer problems by the Finite Element Method (FEM) [29] using a 10 by 10 structured mesh of tri elements. FEM has been established as a well known numerical approach, for more information many sources exist such as Hughes [30]. More details are also given in Section 3.1. The dataset is split randomly into train/test sets following a 70:30 ratio (70% of the data for training and 30% of the data for testing).

In order to justify our approach of using neural networks to perform inverse analysis, interpretability of networks must be established by analysing the most prominent weights on how they are affected by their previous connections as well as effect on further connections. The Shapley explanation is the most direct way to create an interpretable machine learning model using Shapley values. Shapley values have been introduced in game theory since 1953 [31] but only recently they have been used in the feature importance context [32]. Shapley values show the contributions of each feature (each selected point inside the domain) to the prediction (average impact on model output, i.e. boundary values). The Shapley value for a certain feature, f (out of F total features), given a characteristic function (prediction), p, is:

$$\phi_f(p) = \sum_{S\subseteq F\setminus\{f\}} \frac{|S|!(|F|-|S|-1)!}{|F|!}\left(p(S\cup f) - p(S)\right) \tag{5}$$

where S is all feature subsets used in the model, $|S|$ and $|F|$ indicate the sizes of S and F. In words, $\phi_f(p)$ is the incremental impact of including feature f in set S averaged over all sets $S \subseteq F \setminus \{f\}$. In this context, the Shapley value explains the difference between the prediction and the global average prediction which is entirely distributed among the features. However, with F features, the number of possible subsets required in Eq. (5) is 2^F, which increases exponentially when F increases, meaning that the exact solution to this problem becomes computationally intractable problem with large numbers of features. This has led to approximations, of which the SHAP (SHapley Additive exPlanations) method is the most recognised. In this work, we use Deep SHAP, a high-speed approximation algorithm for SHAP values in deep learning models [33].

3. Results and discussion

3.1. Heat conduction

The first problem considered is that of a steady-state heat conduction in a square two-dimensional domain with no heat generation. The general steady-state equation governing heat conduction in Cartesian coordinates is:

$$\nabla\big(K(T)\nabla T\big) = 0, \tag{6}$$

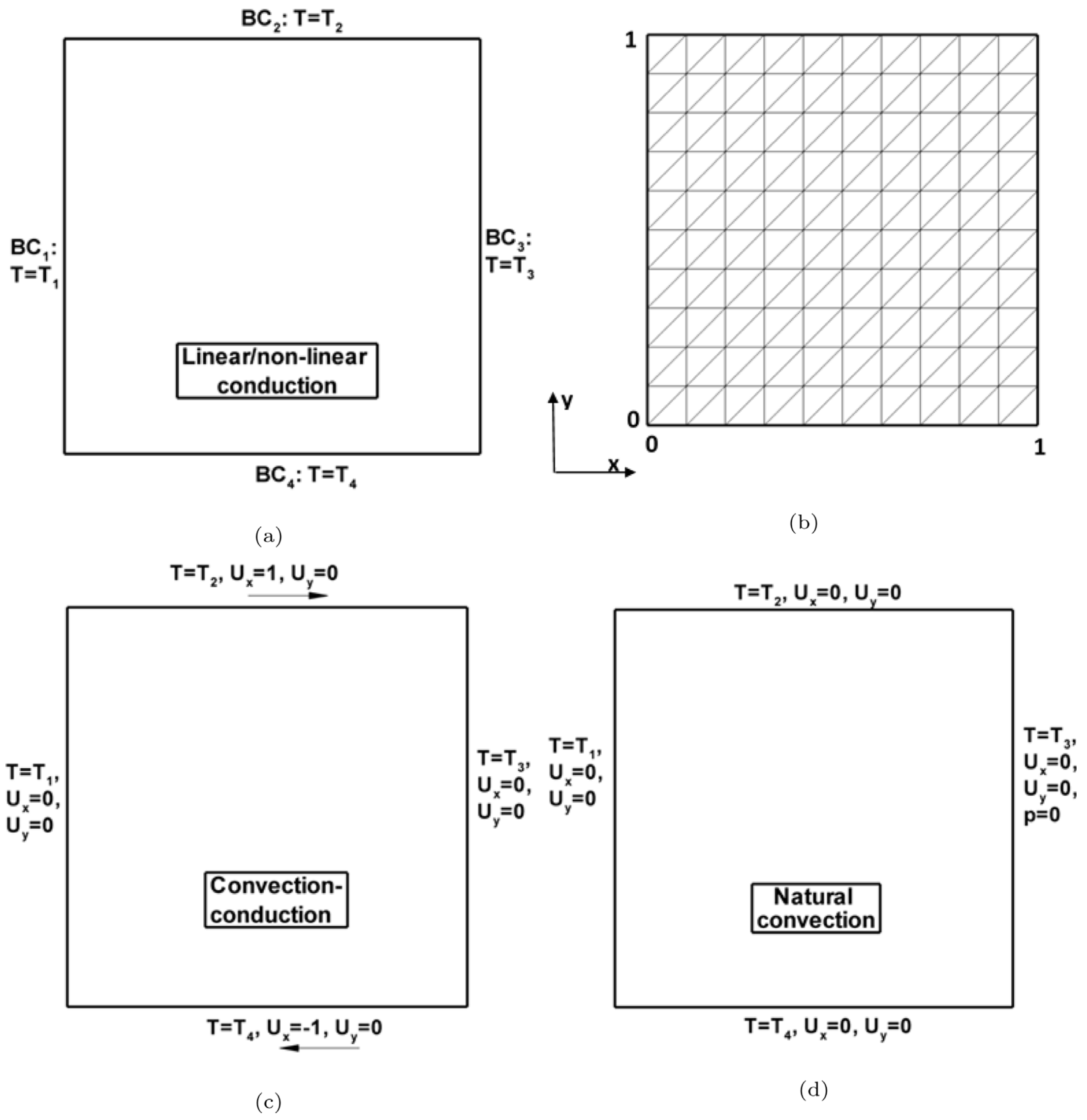


Fig. 2. Heat conduction and convection problems: (a) Linear/nonlinear conduction examples. (b) Finite element mesh used for all simulations. (c) convection–conduction problem. (d) Natural convection in a square cavity.

where T is the temperature and $K(T)$ is temperature dependent conductivity. In this work, both $K(T)=k_0$ (linear conduction) and $K(T)=k_0T$ (non-linear conduction) are considered. Note that, for the non-linear case, a Jacobi iterative scheme is used for the solution of the temperature.

The geometry and the mesh used for the heat conduction problem are illustrated in Fig. 2. Here, the prescribed temperatures are given in the Dirichlet form and the initial conditions are generally taken as quiescent by default. The boundary values for 15,000 samples are generated randomly, ranging between 0 and 1. Note that, any range for boundary values could have been chosen to solve the governing equations using Finite Element Method (FEM). In this work, boundary values are non-dimensional temperatures. They are normalised by the maximum temperature on the boundary, so ranging between 0 to 1. The finite element codes used in this study have been previously validated [34].

The range of NN architectures that were tested is defined in Table 1. The accuracy results for various NN configurations have been collated and are shown in Table 2.

Table 2
Heat conduction (linear and non-linear) problem model performance with three, four, and five measured temperatures,
Inputs: Measured temperatures at
3-points: [(0.1,0.5),(0.5,0.9),(0.9,.8)],
4-points: [(0.1,0.5),(0.5,0.9),(0.9,.8),(0.5,0.1)],
5-points: [(0.1,0.5),(0.5,0.9),(0.9,.8),(0.5,0.1),(0.9,0.8)];
Outputs: Estimation of boundaries BC_1 to BC_4.

Neurons at each layer	Temperature points	Trainable parameters	Number of Epochs	Accuracy linear case	Accuracy non-linear case
3-16-4	3	2 632	100	0.729	0.735
3-16-4	3	2 632	200	0.737	0.758
3-16-4	3	2 632	500	0.748	0.759
3-64-4	3	9 064	100	0.741	0.756
3-64-4	3	9 064	200	0.742	0.758
3-64-4	3	9 064	500	0.748	0.760
3-64-32-16-4	3	11 096	100	0.728	0.745
3-64-32-16-4	3	11 096	200	0.742	0.759
3-64-32-16-4	3	11 096	500	0.749	0.762
4-16-4	4	2 696	100	0.910	0.918
4-16-4	4	2 696	200	0.951	0.956
4-16-4	4	2 696	500	0.966	0.967
4-64-4	4	9 320	100	0.971	0.967
4-64-4	4	9 320	200	0.963	0.965
4-64-4	4	9 320	500	0.970	0.967
4-64-32-16-4	4	11 160	100	0.935	0.942
4-64-32-16-4	4	11 160	200	0.959	0.969
4-64-32-16-4	4	11 160	500	0.972	0.967
5-16-4	5	2 760	100	0.932	0.927
5-16-4	5	2 760	200	0.957	0.957
5-16-4	5	2 760	500	0.967	0.963
5-64-4	5	9 576	100	0.967	0.973
5-64-4	5	9 576	200	0.961	0.967
5-64-4	5	9 576	500	0.968	0.967
5-64-32-16-4	5	11 224	100	0.937	0.936
5-64-32-16-4	5	11 224	200	0.961	0.959
5-64-32-16-4	5	11 224	500	0.976	0.968

First, considering the cases with three temperature points, the NNs exhibit an accuracy of between 72.8% and 76.2% for the prediction of boundary values. For example, an accuracy of 74.9% is reached for the NN configuration of 3-64-32-16-4 for the linear heat conduction problem (with 11,096 trainable parameters) after 500 epochs.

A few general observations can be made for this case. Firstly, the accuracy increases with respect to the number of trainable parameters. However, having said that, after 500 epochs, the difference in accuracy of the architectures presented in Table 2 is negligible. That is 74.8% to 74.9% and 75.9% to 76.2% in the linear and non-linear counterparts, respectively. The other noteworthy point is that the non-linear cases have a higher accuracy than the linear ones.

The loss values predicted by the neural network for trained and tested data are around 2% for almost all the architectures, after an initial stabilisation period. As indicated in Fig. 3 for 3-64-32-16-4 NN with three temperature points, the loss is consistent with increasing number of epochs. Comparing the behaviour of the validation dataset against the training one, good performance is observed by the predictions of the NNs.

Next, the model with four field points is considered, results also shown in Table 2. Here, the boundary conditions are determined by providing four temperature measurements. In this case, the accuracies to predict the boundary values are in the range of 91.0% to 97.2% for linear heat conduction case and between 91.8% and 96.9% for the non-linear counterpart. This is a significant increase in accuracy compared with the cases with three field points.

As with the previous cases, the accuracy increases with the number of trainable parameters but there is a less notable difference in the accuracy of the linear and non-linear results. The three-layer NNs take more epochs to

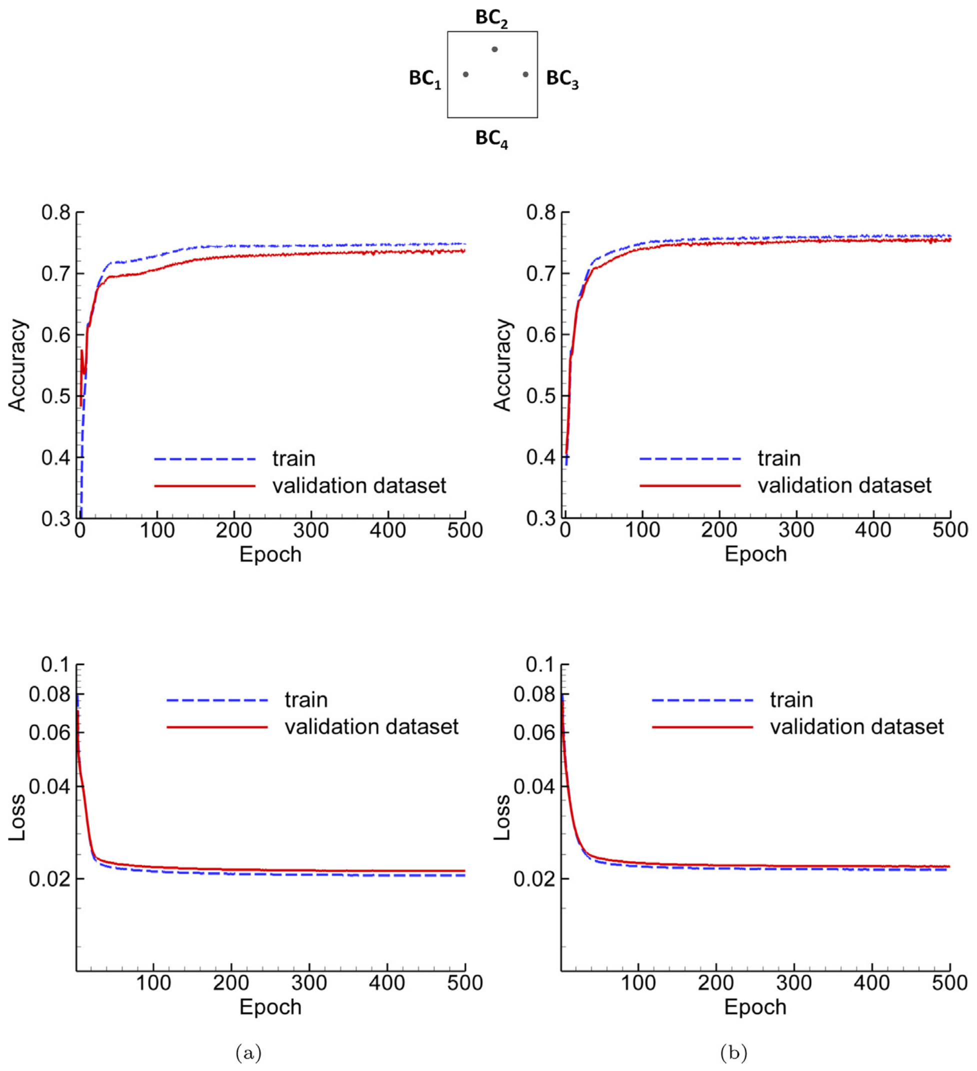


Fig. 3. Model accuracy and loss for the training and validation dataset with three measured temperatures, (a) Linear heat conduction ($K(T)=k_0$, $k_0=1$), (b) Non-linear heat conduction ($K(T)=k_0T$, $k_0=1$), 3-64-32-16-4 network configuration.

near convergence compared with the cases with three field points. Another point to note is that there is increased agreement between the behaviour of the training and validation datasets, as shown in Fig. 4. Here, the generated solutions show a loss error of around 0.07% for the linear case and 0.2% for the non-linear case.

Next we consider the case with five temperature field points, also in Table 2. Here there is no significant difference observed in accuracy when compared to the four temperature measurements cases, despite an increase in the number of trainable parameters. The NN accuracies also converge at a similar rate.

Returning to consider the large increase in accuracy from the cases with three measured temperatures to those with four (approximately 75% and 96%, respectively), it is possible to investigate the importance of the contribution of the field points to the predicted BC values by use of SHAP values. Fig. 5 shows the feature importance computed by the mean of the absolute SHAP values on model output for boundaries BC$_1$ to BC$_4$ with 3(4)-64-32-16-4 network configuration. The length of each grey block represents the feature importance of the corresponding model output.

Comparing the sub-figures of Fig. 5 it can be seen that the field point with the highest influence on the BC is the one which is nearest, as could be expected. Fig. 5g is the only one where there is not a field point which contributes significantly more than the others, this is due to the points being of approximately equal distance from

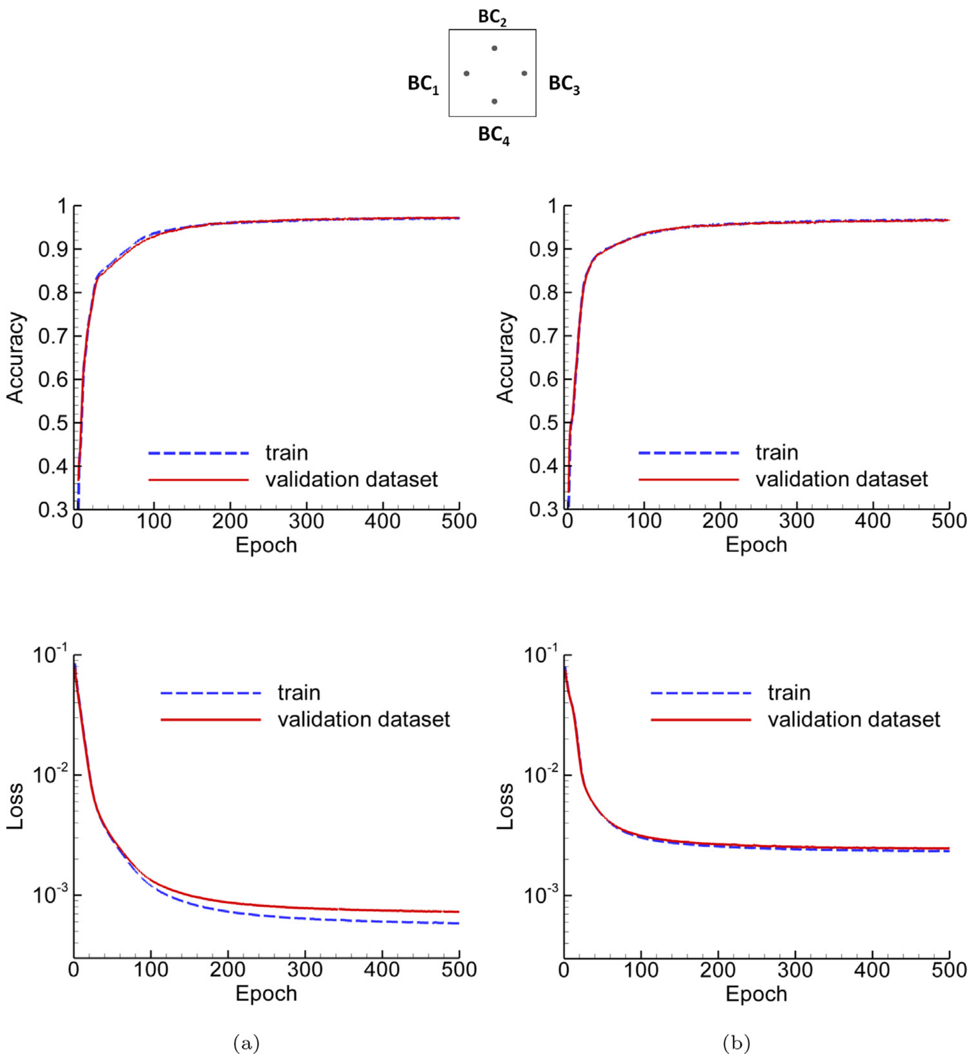


Fig. 4. Model accuracy and loss for the training and validation dataset with four measured temperatures, (a) Linear heat conduction, (b) Non-linear heat conduction, 4-64-32-16-4 network configuration.

the boundary. To highlight this observation, for the four point equivalent in Fig. 5h, the SHAP value for T_{L4} (not present in the three point case) is significantly higher than T_{L1}–T_{L3} due to its proximity to BC$_4$.

3.2. Heat convection–conduction

Next, the steady-state heat convection–conduction model is considered. The equation in two dimensions for the planar configuration can be expressed as:

$$(\boldsymbol{v}\nabla)T = \nabla^2 T, \tag{7}$$

where v=(v_x, v_y) is the velocity field. A schematic representation of the domain is depicted in Fig. 2c. In the case of convection–conduction problem, the velocity components are assumed to be constant. A steady, constant velocity field is generated throughout the domain by solving the steady-state incompressible Navier–Stokes equations. The boundary conditions used to generate the velocity field are given in Fig. 2c. Then, with the constant values of velocity for the entire domain, the temperature is solved to a tolerance of 10^{-6}. Heat convection–conduction arises in many problems in engineering, such as the heat sinks used in the electronic industry to dissipate heat from electronic components to the ambient, the dissipation of heat in electrical windings to the coolant, the heat exchange process

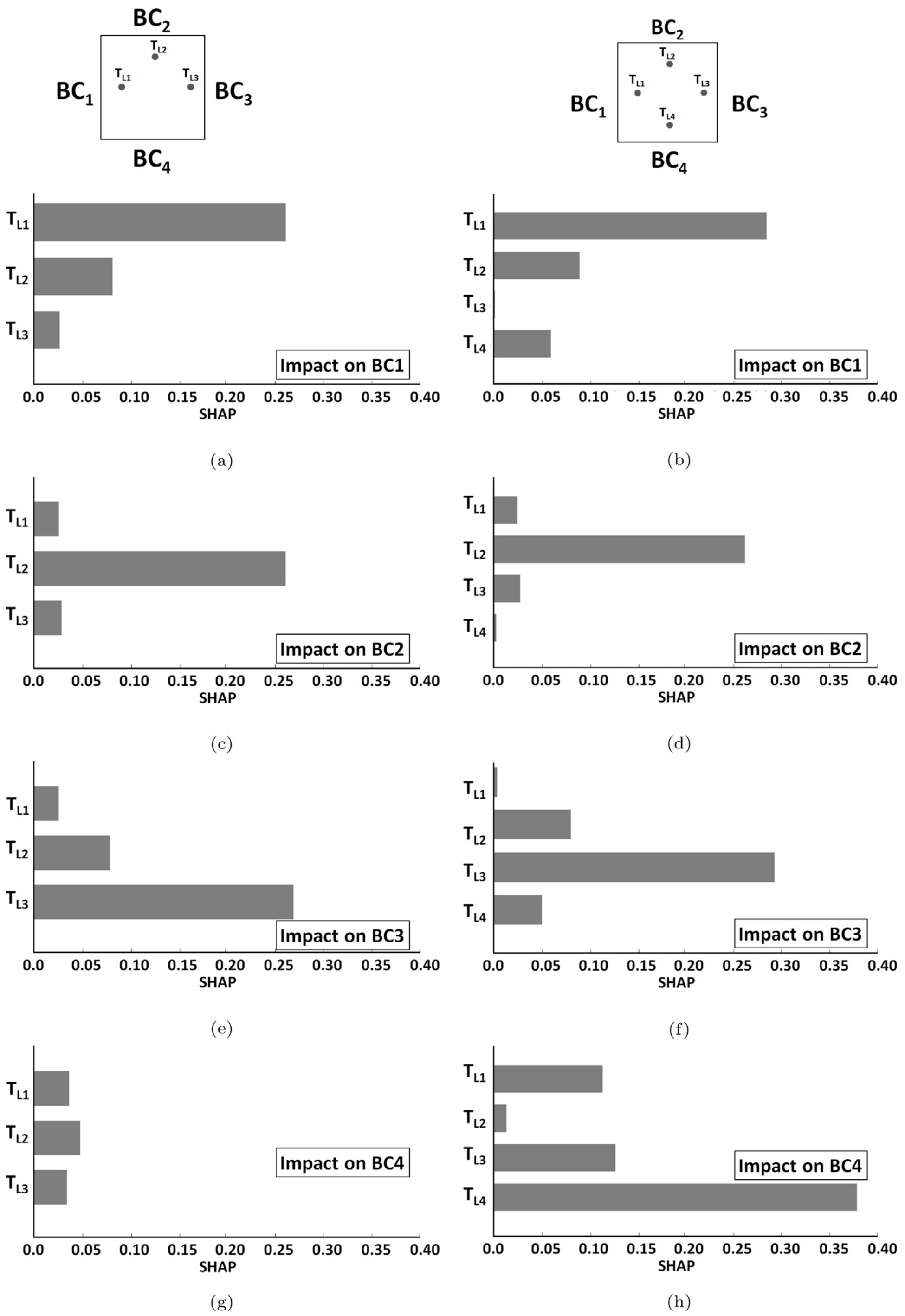


Fig. 5. Mean (|SHAP value|), average impact on model output (BC_1–BC_4), 3(4)-64-32-16-4 network configuration. Linear conduction problem.

in heat exchangers and the cooling of gas turbine blades in which the temperature of the hot gases is greater than the melting point of the blade material [25].

Following the heat conduction problem and the observations on accuracies for different NNs in Section 3.1 (Table 2), the artificial neural network architectures 3(4,5)-64-4 and 3(4,5)-64-32-16-4 NNs with 200 and 500 number of epochs are used for the heat convection–conduction case.

Table 3
Heat convection–conduction problem model performance with three, four, and five measured temperatures,
Inputs: Measured temperatures at
3-points: [(0.1,0.5),(0.5,0.9),(0.9,.8)],
4-points: [(0.1,0.5),(0.5,0.9),(0.9,.8),(0.5,0.1)],
5-points: [(0.1,0.5),(0.5,0.9),(0.9,.8),(0.5,0.1),(0.9,0.8)];
Outputs: Estimation of boundaries BC_1 to BC_4.

Neurons at each layer	Temperature points	Trainable parameters	Number of Epochs	Accuracy
3-64-4	3	9 064	200	0.748
3-64-4	3	9 064	500	0.749
3-64-32-16-4	3	11 096	200	0.747
3-64-32-16-4	3	11 096	500	0.753
4-64-4	4	9 320	200	0.966
4-64-4	4	9 320	500	0.970
4-64-32-16-4	4	11 160	200	0.951
4-64-32-16-4	4	11 160	500	0.972
5-64-4	5	9 576	200	0.963
5-64-4	5	9 576	500	0.964
5-64-32-16-4	5	11 224	200	0.958
5-64-32-16-4	5	11 224	500	0.970

As shown in Table 3, all NN models show a similar level of accuracy to the cases in the heat conduction problem. That is, approximately 75%, 97%, and 97% for the cases of having three, four, and five measured temperatures, respectively. For the five-layer 3(4)-64-32-16-4 NN, the model accuracy and loss for the training and validation datasets with three, four, and five measured temperatures present similar trends when comparing with the linear/non-linear conduction counterparts. Furthermore, very little differences are observed between the training and validation dataset in this case, confirming no model overfitting. This is also true when comparing model loss for the training and validation dataset with a rapid decline in loss values. This leads to model stability and non-overfitting, indicating desirable model performance.

3.3. *Natural convection heat transfer*

Finally, the more complex case of steady-state natural convection was considered. In non-dimensional form, the governing equations may be represented as:

$$\nabla \boldsymbol{v} = 0, \tag{8a}$$

$$(\boldsymbol{v}\nabla)\boldsymbol{v} = -\nabla p + \mathrm{Pr}\,\nabla^2 \boldsymbol{v} + \mathrm{Gr}\,\mathrm{Pr}^2\, T\, \hat{\boldsymbol{y}}, \tag{8b}$$

$$(\boldsymbol{v}\nabla)T = \nabla^2 T, \tag{8c}$$

where p is the pressure, $\hat{\boldsymbol{y}}$ is the unit vector in the y-direction, and Pr and Gr are the Prandtl and Grashof numbers, respectively (see [25] for more details). Gr is taken here to be between $\mathrm{Gr} = 1$ to $\mathrm{Gr} = 10^5$ (from simple to more complex physical systems) and Pr=0.71 which corresponds to air. Natural convection is generated by the density difference induced by the temperature differences within a fluid system. In most natural convection problems, flow is generated by either a temperature variation or a concentration variation in the fluid system, which leads to local density differences [25,26]. Therefore, in such flows, a body force term $\mathrm{Gr}\,\mathrm{Pr}^2\mathrm{T}$ (8b) needs to be added to the y-momentum equation to include the effect of local density differences. In addition, and as shown in Fig. 2d, a fixed pressure ($p = 0$) is imposed on the right hand side boundary.

Table 4 provides the accuracies for the natural convection problem under various NNs for $Gr = 1$ with 200 and 500 number of epochs. In this simple case and with 500 epochs, the accuracy of 74.9% is achieved with the five-layer 3-64-32-16-4 NN when selecting three measured temperatures for predicting boundary values. An accuracy of 97.2% is observed with four and five selected data points inside the domain, see Table 4. These findings for the

Table 4
Natural convection problem model performance with three, four, and five measured temperatures, $Gr = 1$,
Inputs: Measured temperatures at
3-points: [(0.1,0.5),(0.5,0.9),(0.9,.8)],
4-points: [(0.1,0.5),(0.5,0.9),(0.9,.8),(0.5,0.1)],
5-points: [(0.1,0.5),(0.5,0.9),(0.9,.8),(0.5,0.1),(0.9,0.8)];
Outputs: Estimation of boundaries BC_1 to BC_4.

Neurons at each layer	Temperature points	Trainable parameters	Number of Epochs	Accuracy
3-64-4	3	9 064	200	0.742
3-64-4	3	9 064	500	0.744
3-64-32-16-4	3	11 096	200	0.747
3-64-32-16-4	3	11 096	500	0.749
4-64-4	4	9 320	200	0.962
4-64-4	4	9 320	500	0.972
4-64-32-16-4	4	11 160	200	0.958
4-64-32-16-4	4	11 160	500	0.972
5-64-4	5	9 576	200	0.959
5-64-4	5	9 576	500	0.965
5-64-32-16-4	5	11 224	200	0.957
5-64-32-16-4	5	11 224	500	0.972

Table 5
Natural convection problem model performance with 3–10 measured temperatures inside a unit square domain, various Gr,
Inputs: Measured temperatures at
3-points: [(0.1,0.5),(0.5,0.9),(0.9,.8)],
4-points: [(0.1,0.5),(0.5,0.9),(0.9,.8),(0.5,0.1)],
5-points: [(0.1,0.5),(0.5,0.9),(0.9,.8),(0.5,0.1),(0.9,0.8)],
6-points: [(0.1,0.5),(0.5,0.9),(0.9,.8),(0.5,0.1),(0.9,0.8),(0.9,0.3)],
7-points: [(0.1,0.5),(0.5,0.9),(0.9,.8),(0.5,0.1),(0.9,0.8),(0.9,0.3),(0.1,0.2)],
8-points: [(0.1,0.5),(0.5,0.9),(0.9,.8),(0.5,0.1),(0.9,0.8),(0.9,0.3),(0.1,0.2),(0.9,0.5)],
10-points: [(0.1,0.5),(0.5,0.9),(0.9,.8),(0.5,0.1),(0.9,0.8),(0.9,0.3),(0.1,0.2),(0.9,0.5),(0.1,0.3),(0.5,0.2)];
Outputs: Estimation of boundaries BC_1 to BC_4.

Neurons at each layer	Temperature points	Grashof number(Gr)	Trainable parameters	Number of Epochs	Accuracy
4-64-32-16-4	4	1	11,160	500	0.972
4-64-32-16-4	4	10^2	11,160	500	0.972
4-64-32-16-4	4	10^3	11,160	500	0.945
4-64-32-16-4	4	10^4	11,160	500	0.849
4-64-32-16-4	4	10^5	11,160	500	0.738
4-64-32-16-4	4	10^5	11,160	500	0.738
4-64-32-16-4	4	10^5	11,160	5,000	0.742
4-64-32-16-4	4	10^5	11,160	10,000	0.755
3-64-32-16-4	3	10^5	11,096	10,000	0.672
4-64-32-16-4	4	10^5	11,160	10,000	0.755
5-64-32-16-4	5	10^5	11,224	10,000	0.805
6-64-32-16-4	6	10^5	11,288	10,000	0.843
7-64-32-16-4	7	10^5	11,352	10,000	0.848
8-64-32-16-4	8	10^5	11,416	10,000	0.855
10-64-32-16-4	10	10^5	11,544	10,000	0.906

natural convection at the level of Gr = 1 are consistent with the results reported above for the heat conduction and convection–conduction cases.

As the Grashof number (Gr) increases from Gr = 1 to Gr = 10^5, the non-linearity of the physical model affects the accuracy for the prediction of the boundary values. Table 5 presents the detailed model performance for a range

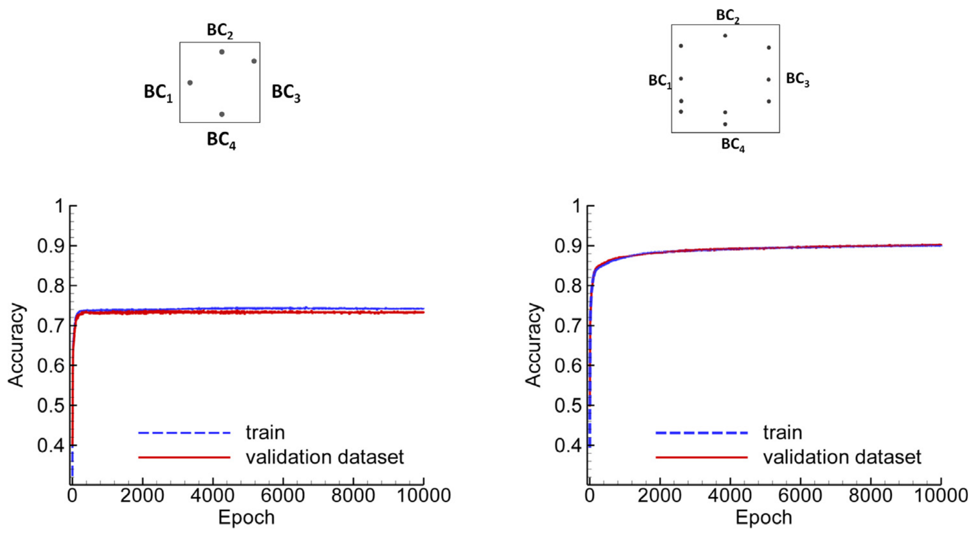


Fig. 6. Model accuracy for the training and validation dataset with four and ten measured temperatures, Natural convection, $Gr = 10^5$, 4(10)-64-32-16-4 network configuration.

of Grashof numbers with three to ten temperature points. Considering the example for four temperature points where only the Grashof number is increased from Gr = 1 to Gr = 10^5, NN architecture and number of epochs are consistent, the accuracy is seen to decrease from 97.2% to 73.8%.

To mitigate the decreasing accuracy, as previously demonstrated, it is possible to increase the number of epochs or temperature points. By increasing the number of epochs to 5,000 then 10,000, a relative increase in accuracy of 0.4% then 1.3% is achieved with $Gr = 10^5$. Progressively increasing the number of temperature points from 4 to 10 sees a further incremental accuracy increase of 15.1%, up to an accuracy of 90.6%. See Fig. 6 for comparison of accuracy with four and ten points with increasing number of epochs.

To investigate how the features influence the model's prediction we again apply the SHAP value-based interpretability, see Fig. 7. Differently to the linear conduction problem, it can be observed in this more complex non-linear case that the point nearest the BC is not always the greatest influencer on predicted values. For example, in Fig. 7a, point T_{L4} has less of an influence on BC_4 than T_{L3} despite being significantly nearer in proximity. Conversely it is also possible to investigate which points have the least influence as a whole and may potentially be discarded if necessary. In Fig. 7b, it can be seen that T_{L4} has the least impact on all four BCs. In comparing the SHAP values for the four and ten point cases, it can be seen that on average the values are higher in the latter. This means that the points contribute more significantly to the predictions, consequently leading to approximately a 15% increase in accuracy.

Finally, it is shown that choosing different positions for the temperature points inside the domain geometry affect the accuracy and loss error for the training and validation datasets. Fig. 8 shows two cases where the only difference are the locations of the field points. Here, the model accuracy is 55% with set A compared to 76% achieved with set B due to more favourable positioning of the points for the prediction of the BCs. This aspect may be used in practice to optimise the location of field points for greater accuracy and minimise the number required.

3.4. *Annulus geometry — heat conduction problem*

We extend the method to a more complex (realistic) annulus geometry with a radius ratio (ratio between outer and inner) of 5 (Fig. 9a). In this setting, both Dirichlet and Neumann (heat flux) boundary conditions [25] are considered. The nondimensional temperatures on BC_1 are between 0 to 1 and the nondimensional heat flux (q on BC_2) is assumed to be between 0 and 10 (Fig. 9b). For this linear heat conduction problem, 15000 samples are generated by solving steady-state equation in cartesian coordinates using the Finite Element Method (FEM). Following the heat conduction problem and the observations on accuracies for different NNs in Sections 3.1–3.3,

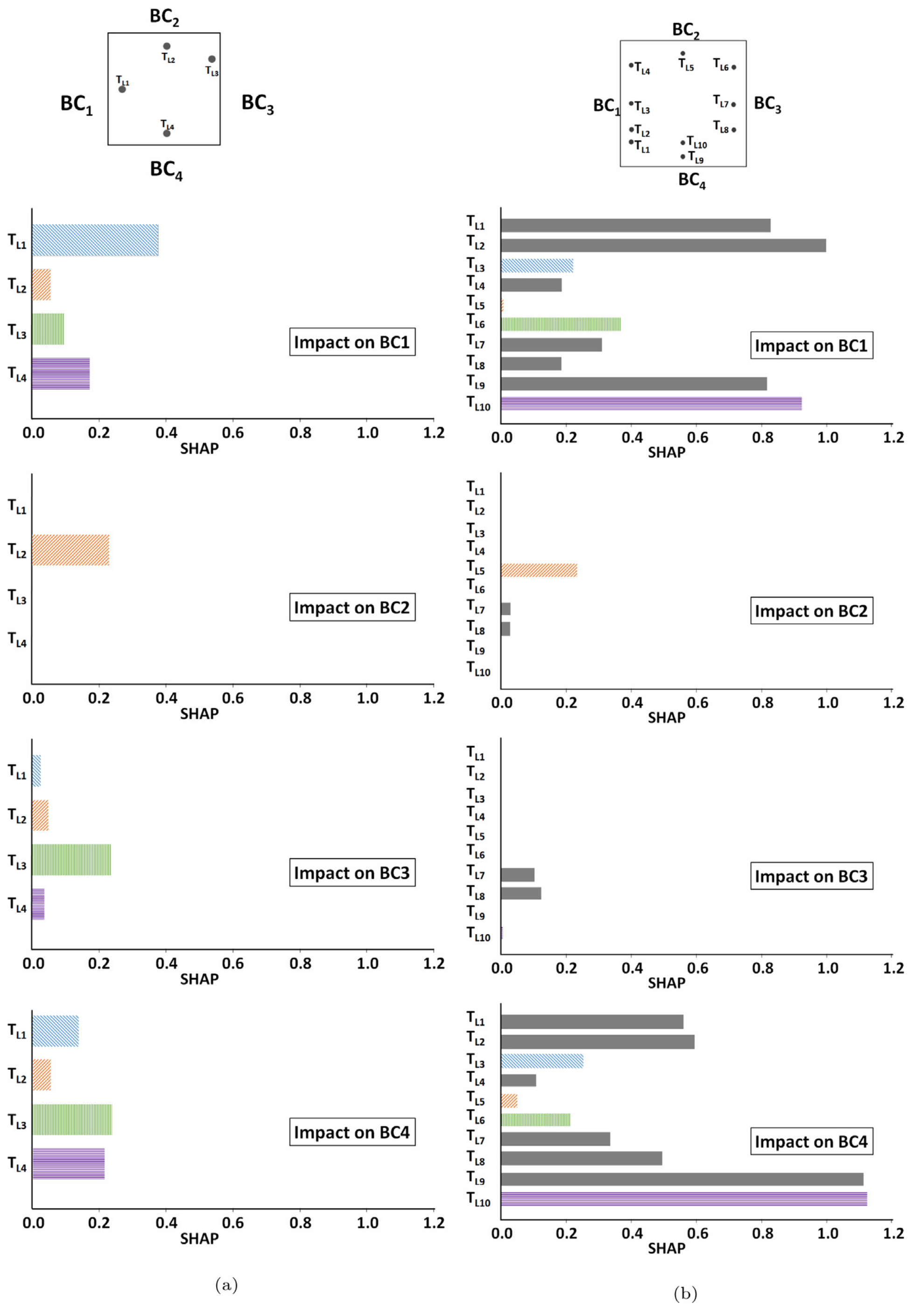
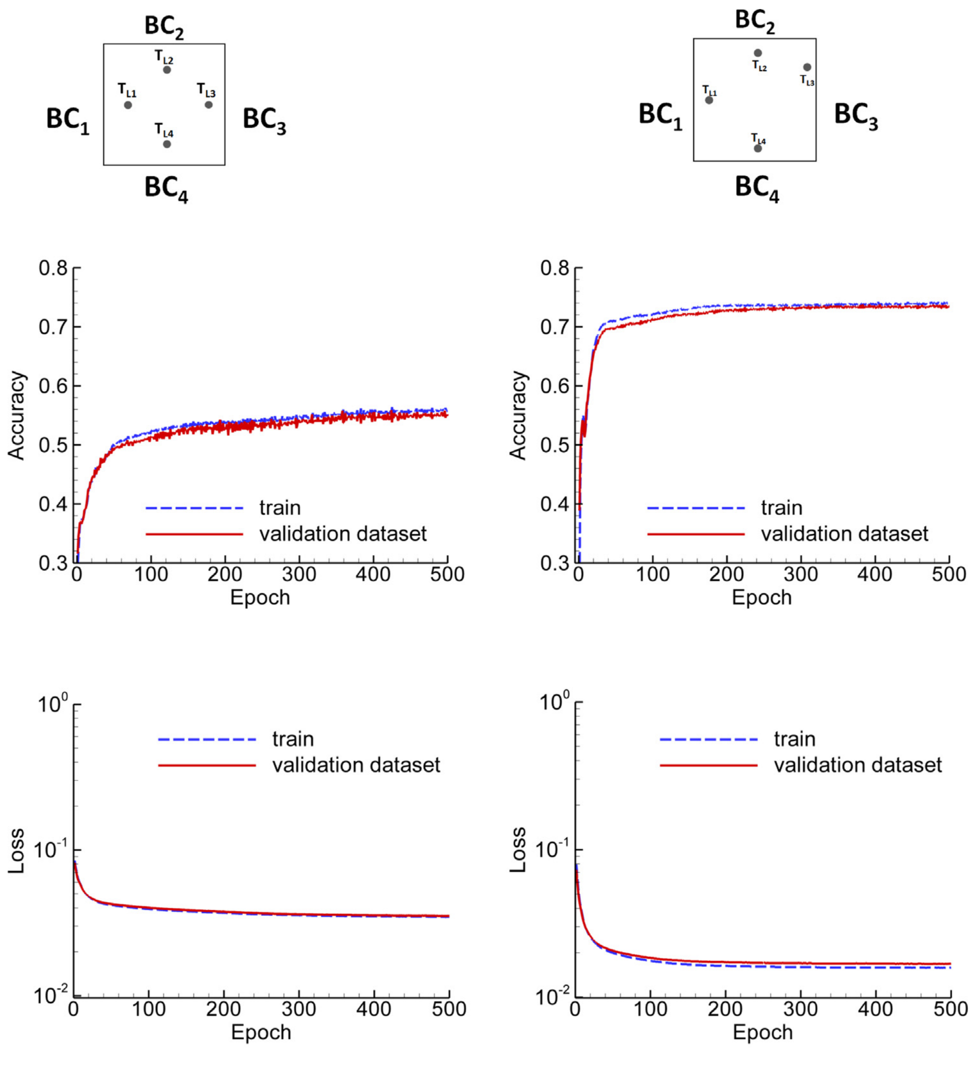


Fig. 7. Mean (|SHAP value |), average impact on model output (BC$_1$–BC$_4$), Natural convection, $Gr = 10^5$, 4(10)-64-32-16-4 network configuration, (a) 4-points , (b) 10-points.

the artificial neural network architectures 3(4, 5)-64-2 and 3(4, 5)-64-32-16-2 NNs with 1000 number of epochs are used for this case. Measured temperatures inside the domain geometry with three, four, and five arbitrary locations are shown in Fig. 9a. Table 6 provides the accuracies for the linear conduction problem with 3(4,5)-64-2 and 3(4,5)-64-32-16-2 NNs. Considering the cases with three temperature points, the 3-64-2 NN exhibits an accuracy



(a) (b)

Fig. 8. Model accuracy and loss for the training and validation dataset with four measured temperatures for two different set of locations: (a) A={L1(0.2,0.5), L2(0.5,0.8), L3(0.8,0.5), L4(0.5,0.2)}; (b) B={L1(0.1,0.5), L2(0.5,0.9), L3(0.9,0.8), L4(0.5,0.1)}, Natural convection, $Gr = 10^5$, 4-64-32-16-4 network configuration.

of 70.1% for the prediction of boundary values BC_1 and BC_2. Furthermore, an accuracy of 72.4% is reached for the NN configuration of 3-64-32-16-2 with 10,756 trainable parameters after 1000 epochs, see Table 6 and Fig. 10. The loss values predicted by the neural network for trained and tested data are around 0.0007% for 3-64-32-16-2 NN architecture, after an initial stabilisation period. Here, no significant differences were found in performance when comparing training and testing sets. This is consistent with results shown in Sections 3.1–3.3 for the heat conduction, convection–conduction, and natural convection problems with the square geometry problem.

The results for the model with four field points (L_1-L_4 in Fig. 9a) are shown in Table 6 and Fig. 10. In this case, the accuracies are 84.0% and 90% for 4-64-2 NN and 4-64-32-16-2 NN, respectively. This is almost 19% increase in accuracy compared with the cases with three field points.

Next we consider the case with five temperature field points (L_1-L_5 in Fig. 9a). The results for this case are shown in Table 6. Here there is no significant difference observed in accuracy when compared to the four temperature measurements. This finding agrees with the results reported above for the heat conduction with square geometry problem when comparing four and five temperature field points.

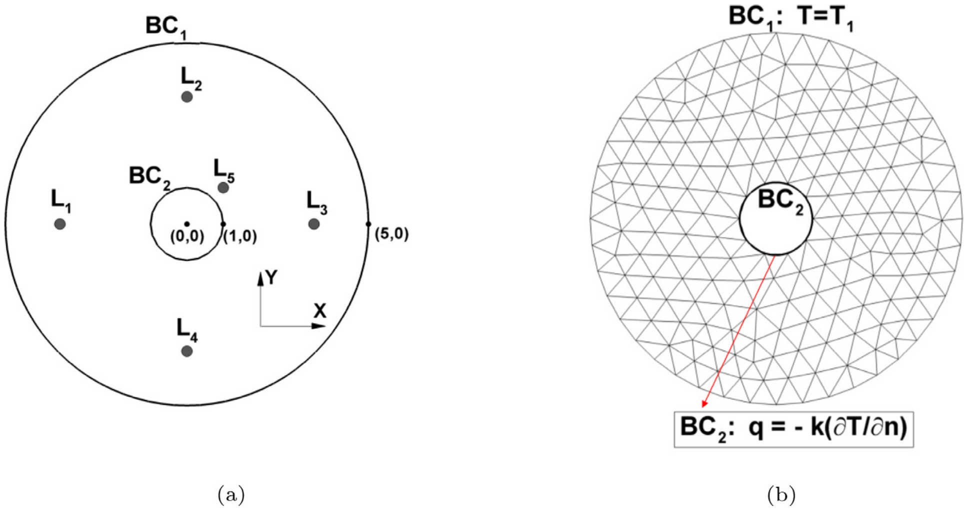


Fig. 9. Heat conduction problem, (a) Locations and coordinates of three, four, and five measured temperatures: L1(−3.5,0.0), L2(0.0,3.5), L3(3.5,0.0), L4(0.0,−3.5), L5(1.0,1.0), (b) Finite element mesh with Dirichlet (BC_1) and Neumann (BC_2, k = 1) boundary conditions, Annulus geometry.

Table 6
Heat conduction problem model performance with three, four, and five measured temperatures, Annulus geometry,
Inputs: Measured temperatures at
3-points: [(−3.5,0.0),(0.0,3.5),(3.5,0.0)],
4-points: [(−3.5,0.0),(0.0,3.5),(3.5,0.0),(0.0,−3.5)],
5-points: [(−3.5,0.0),(0.0,3.5),(3.5,0.0),(0.0,−3.5),(1.0,1.0)];
Outputs: Estimation of boundaries BC_1 and BC_2.

Neurons at each layer	Temperature points	Trainable parameters	Number of Epochs	Accuracy
3-64-2	3	8 436	1000	0.701
3-64-32-16-2	3	10 756	1000	0.724
4-64–2	4	8 564	1000	0.869
4-64-32-16-2	4	10 788	1000	0.900
5-64–2	5	8 692	1000	0.911
5-64-32-16-2	5	10 820	1000	0.922

4. Conclusions

The effectiveness of a Machine Learning (ML) approach for solving inverse problems has been presented for linear/non-linear heat conduction, convection–conduction, and natural convection problems. The correctness of the algorithms was checked by comparing the statistical evaluation metrics, such as accuracy and loss on in/out of sample data. For almost all the case studies in this work, no significant differences were found in performance when comparing training and testing sets.

For the linear heat conduction problem, the model accuracies of 75%, 98%, and 98% were achieved to predict the boundary temperatures given three, four and five locations inside the domain geometry, respectively. In this work, it is also demonstrated that the proposed methodology is capable of representing the model performance well and is a useful modelling tool for predicting the non-linear heat conduction problems as well,

The problem of finding the best solution for a non-linear ill-posed problem and the theory for this kind of problems is much less developed than the corresponding theory for linear problems. In this work, we currently avoid non-uniqueness solutions by assuming that the boundary conditions are known. However, the strategy to have a unique solution is to use a combination of unsupervised learning and probability distribution. This requires an extremely large database, that accounts for different boundary condition values, combinations and types. By

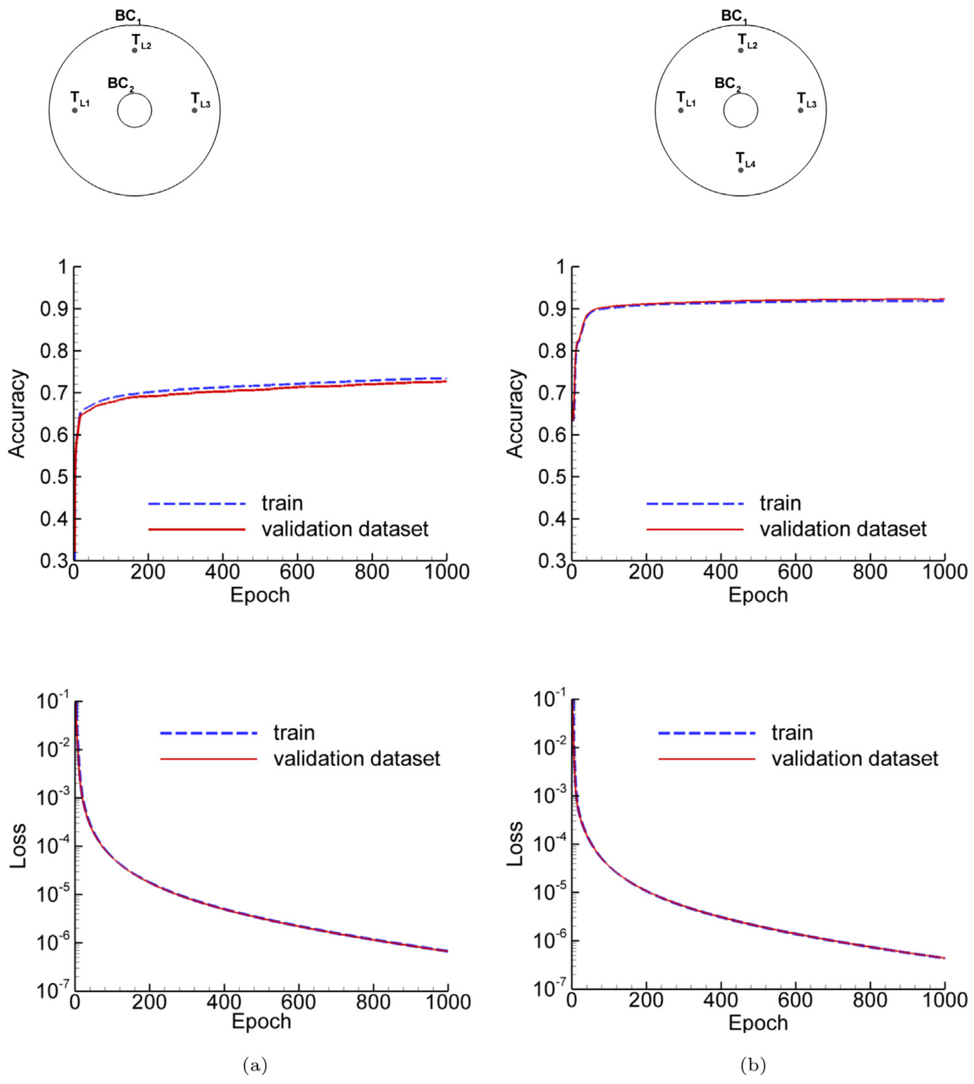


Fig. 10. Model accuracy and loss for the training and validation dataset with three and four measured temperatures, Linear heat conduction, (a) three points 3-64-32-16-2 (b) four points 4-64-32-16-2 network configurations, Annulus geometry.

performing a K-means clustering on such a database, we first identify various clusters having similar boundary condition types, then identifying the values for each of the boundary conditions within the cluster. This can be achieved by estimating the occurrence of each of its value within the selected cluster and calculating probability distribution for the values. Further, value with the highest probability is chosen as the unique solution for that particular boundary condition. This approach allows us to calculate a unique solution when ill-posed problems with non-uniqueness solution is presented. Unfortunately, it is not feasible to implement this idea because of the need for a much larger database, hence we leave it for future work.

Additionally, an inverse modelling ML approach is considered for the steady-state heat convection–conduction and natural convection problems (from simple to more complex physical systems) for further validation and deeper understanding of Neural Networks (NN) concepts in computational mechanics. The study of these problems assist in the overall drive to design and implement the most appropriate NN and testing general applications for more complex systems. In this study, it is shown that the results for heat convection–conduction and simple natural convection cases are comparable with the linear/non-linear heat conduction counterparts. High model accuracy is attainable for complex natural convection problems by increasing certain variables, as with the other cases. The variable that had the most significant impact on accuracy was the number and positioning of temperatures points inside the domain geometry.

In addition, we have demonstrated the extended application of our NN model to a more complex setting with an annulus geometry model problem using both Dirichlet and Neumann (heat flux) boundary conditions.

Furthermore, we have employed the SHapley Additive exPlanations (SHAP) value analysis concept to get detailed feature importance for the model interpretation.

The inverse analysis ML algorithms used have provided promising results. The presented method works with any non-linear problems and the method could easily be applied to a wide range of complex problems. The modelling performance for both training and testing indicates that there exists good agreement between the target values and predicted values for all the instances. On a single-core computer a single FEM forward problem used in this work completed in the order of seconds, training a ML NN on 15,000 datasets completed in the order of few minutes. Once training is complete, new inverse predictions may be obtained using the trained model in milliseconds. In comparison to the trial and error approach of conventional inverse modelling the ML approach presented here is more robust, is less reliant on the individual user and has a scope to be used on significantly more complex systems. This work focused on simulated data, future effort could implement a similar approach to non-uniform experimental datasets.

Acknowledgment

This work was partially funded by EPSRC grant EP/R012091/1.

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.

References

[1] M.N. Ozisik, H.R.B. Orlande, Inverse Heat Transfer, Taylor and Francis, 2000.
[2] A.N. Tikhonov, A.V. Goncharsky, V.V. Stepanov, A.G. Yagola, Numerical Methods for the Solution of Ill-Posed Problems, Kluwer, Springer, Dordrecht, 1995.
[3] H.W. Engl, M. Hanke, A. Neubauer, Regularization of Inverse Problems, in: Mathematics and Its Applications, vol. 375, Kluwer Academic Publishers Group, Dordrecht, 2000.
[4] E. Ippoliti, Heuristic reasoning, in: Studies in Applied Philosophy, Epistemology and Rational Ethics, Springer International Publishing, Switzerland, 2015, pp. 1–2.
[5] J. Pearl, Heuristics: Intelligent Search Strategies for Computer Problem Solving, United States:Addison-Wesley Pub. Co., Inc, Reading, MA, 1984, p. 3, Retrieved June 13, 2017.
[6] M.R. Fellows, F.V. Fomin, D. Lokshtanov, F. Rosamond, S. Saurabh, Y. Villanger, Local search: Is brute-force avoidable? J. Comput. System Sci. 78 (2012) 707–719.
[7] Y. Jaluria, Solution of inverse problems and thermal system, J. Thermal Sci. Eng. Appl. (2018) http://dx.doi.org/10.1115/1.4042353, TSEA-18-1485.
[8] Y. Jaluria, Design and Optimization of Thermal Systems, second ed., CRC Press, Boca Raton, FL, 2008.
[9] A. Bangian-Tabrizi, Y. Jaluria, An optimization strategy for the inverse solution of a convection heat transfer problem, Int. J. Heat Mass Transfer 124 (2018) 1147–1155.
[10] Y. Zhang, W. Shuihua, G. Ji, A comprehensive survey on particle swarm optimization algorithm and its applications, Math. Probl. Eng. 2015 (2015) 1–38, http://dx.doi.org/10.1155/2015/931256.
[11] S. Voronin, C. Zaroli, Survey of computational methods for inverse problems, 2018, http://dx.doi.org/10.5772/intechopen.73332.
[12] F. Yaman, V.G. Yakhno, R. Potthast, A survey on inverse problems for applied sciences, Math. Probl. Eng. (2013) http://dx.doi.org/10.1155/2013/976837, Article ID 976837.
[13] S. Szénási, I. Felde, Using multiple graphics accelerators to solve the two-dimensional inverse heat conduction problem, Comput. Methods Appl. Mech. Engrg. 336 (2018) 286–303.
[14] Y. Bengio, Learning deep architectures for AI, Found. Trends Mach. Learn. 2 (2009) 1–127.
[15] G. Capuano, J.J. Rimoli, Smart finite elements: A novel machine learning application, Comput. Methods Appl. Mech. Engrg. 345 (2019) 363–381.
[16] T. Kirchdoerfer, M. Ortiz, Data-driven computational mechanics, Comput. Methods Appl. Mech. Engrg. 304 (2016) 81–101.
[17] F. Chollet, Deep Learning with Python, Manning Publications Co., 2018.
[18] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444.
[19] G. Yagawa, H. Okuda, Neural networks in computational mechanics, Arch. Comput. Methods Eng. 3 (4) (1996) 435–512.

[20] K.T. Yang, Artificial neural networks (ANNs), A new paradigm for thermal science and engineering, J. Heat Transfer 130 (2008) / 093001-1.

[21] X. Guo, W. Li, F. Iorio, Convolutional neural networks for steady flow approximation, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge, Discovery and Data Mining, 2016, pp. 481–490.

[22] J. Schmidhuber, History of computer vision contests won by deep CNNs on GPU, 2017, Retrieved 14 January 2019.

[23] A. Oishi, G. Yagawa, Computational mechanics enhanced by deep learning, Comput. Methods Appl. Mech. Engrg. 327 (2017) 327–351.

[24] S. Chanda, C. Balaji, S.P. Venkateshan, G.R. Yenni, Estimation of principal thermal conductivities of layered honeycomb composites using ANN–GA based inverse technique, Int. J. Therm. Sci. 111 (2017) 423–436.

[25] P. Nithiarasu, R.W. Lewis, K.N. Seetharamu, Fundamentals of the Finite Element Method for Heat and Fluid Flow, John Wiley and Sons Ltd, 2016.

[26] P. Nithiarasu, K.N. Seetharamu, T. Sundararajan, Finite element modelling of flow, heat and mass transfer in fluid saturated porous media, Arch. Comput. Meth. Engng. 9 (1) (2002) 3–42, http://dx.doi.org/10.1007/BF02736231.

[27] G. Urban, K.J. Geras, S. Ebrahimi Kahou, Ö. Aslan, S. Wang, R. Caruana, A. Mohamed, M. Philipose, M. Richardson, Do deep convolutional nets really need to be deep (or even convolutional)? 2016, CoRR, abs/1603.05691.

[28] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv:1412.6980.

[29] O.C. Zienkiewicz, R.L. Taylor, P. Nithiarasu, The Finite Element Method for Fluid Dynamics, seventh ed., Elsevier, 2013.

[30] T.J.R. Hughes, The Finite Element Method: Linear Static and Dynamic Finite Element Analysis, Courier Corporation, 2012.

[31] L. Shapley, A value for n-person games, in: Contrib. Theory of Games, II, in: Ann. Math. Stud., 28, 1953, pp. 307–317.

[32] A.B. Owen, C. Prieur, On shapley value for measuring importance of dependent inputs, SIAM/ASA J. Uncertain. Quantif. 5 (1) (2017) 986–1002.

[33] S.M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, in: Advances in Neural Information Processing Systems, Vol. 30, 2017, pp. 4765–4774.

[34] D. Ding, P. Townsend, M.F. Webster, Computer modelling of transient thermal flows of non-Newtonian fluids, J. Non-Newton. Fluid Mech. 47 (1993) 239–265.