Development of an equation free surrogate model using deep learning algorithm for heat transfer simulation

Somayeh afzali
University of Qom
Mohammad Kazem Moayyedi (moayyedi@alum.sharif.edu)
University of Qom https://orcid.org/0000-0003-4016-1557
Faranak Fotouhi
University of Qom

Research Article

Keywords: Neural network, Deep Learning, Weakly Supervised Learning, Steady State Heat Equation, Boundary Condition

Posted Date: May 4th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1017350/v1

Abstract

The Partial differential equations are one of the main tools in modeling many phenomena in real life. Since the formation and solving of the governing equations requires high processing time and computational costs. This study seeks to provide a method based on deep learning algorithms that can solve the equations independently of direct and numerical solution methods only by applying the boundary conditions of the problem on the neural network. This work explores the application of this paradigm on two-dimensional steady-state heat transfer equation. Due to the challenges of preparation large data with large dimensions, a novel method has been proposed to reduce the necessity of gathering data in large dimensions and size. In this method, first the steady-state heat transfer pattern encrypted in a kernel using thermal data in small size and dimensions. It is then used to train the steady-state predictor network without observing any steady-state heat transfer data simply by imposing restrictions on its outputs. This model was compared with a supervised model that trained by the large size of labeled data in the original dimensions. The result signifies that the proposed model has more accuracy, learning capability, and higher speed during the training stage.

1. Introduction

There are different physical phenomena in nature that respond differently in various situations and times. Analyzing and evaluating the different behaviors of phenomena in different situations and conditions provides a better understanding of their future behavior. It is practically impossible to use empirical methods to identify all natural phenomena. Therefore, modeling is used to describe events and predict the future behavior of the system. Dynamic systems modeling not only can be used as a way to analyze data and test hypotheses, but also to help design and develop theoretical theories [1]. The Partial differential equations are one of the main tools in modeling many phenomena in real life. Some phenomena in nature do not have a definite physical equation and changes in its behavior cannot be interpreted by a comprehensive rule. In addition, many of the differential equations governing systems are nonlinear and very complex, so their solutions are very difficult to obtain and usually cannot be solved by mathematical methods. Moreover, solving them numerically costs a lot of time and money. In return, deep learning algorithms are able to quickly simulate dynamic phenomena without knowledge of the governing differential equations. Unlike conventional methods, deep-learning models learn to use data-driven methods to generate realistic solutions and greatly reduce the amount of computation required, while having high accuracy.

Deep learning algorithms can be used to infer and simulate any dynamic phenomenon by receiving data from observations or simulations. They can also be used directly to learn and predict phenomena whose root cause is complex or still unknown. In recent years, many advances in machine vision and natural language processing have been made through in deep learning [2–4]. Deep learning approach does not have access to the physical laws governing the dynamic system. However, it encodes appropriate information about the dynamics of the system. It makes it possible to teach the neural network how the dynamic system works. Moreover, it performs modeling of complex systems faster, more accurately with

lower computational cost. Furthermore, in neural network models, unlike numerical modeling, no limiting assumptions are required. In this regard, amount of data is one of the key components in solving problems related to deep learning. According to Andrew Ng, founder and leader of Google Brain, " Deep learning is like a rocket whose its engine, is deep learning models and its fuel are huge amounts of data that are fed to these algorithms"[5]. In supervised tasks, such as image recognition and processing[6], speech recognition [7, 8] and machine translation [9], large datasets had to be assembled before the neural network architectures particularly suited to these applications could emerge and achieve human-level performance on these tasks. Currently, there is a huge amount of data in various fields, most of which are un-labeled and a small amount of which is labeled data. One of the great challenges that deep learning faces is data optimization. The data set should be large and diverse enough with accurate labeling.It is always important to provide solutions that minimize the need to provide labeled data with large dimensions and size. The present study has presented a method based on deep learning to model one of the types of transfer phenomena in which the need for labeled data in large dimensions and size has been eliminated. The transport phenomena, studies the exchange of energy, mass, momentum [10], and charge between systems. It is widely used in all engineering fields. One of the types of transport phenomena is heat transfer. The study of heat transfer has particular importance in almost all fields of engineering especially in the field of mechanical engineering. For decades, engineers and scientists have been trying to design and optimize the elements and systems needed to transfer and store heat energy by solving the heat transfer problems. Heat conduction, also called diffusion, is the direct microscopic exchange of kinetic energy of particles through the boundary between two systems. When an object is at a different temperature from another body or its surroundings, heat flows between the two systems so that they both reach the same temperature, at which point they are in thermal equilibrium. Such spontaneous heat transfer always occurs from a region of high temperature to another region of lower temperature, as described in the second law of thermodynamics. This study investigates the problem of heat transfer in a two-dimensional field. The field is a square plate made of some conductive material that is insulated along its edges. Heat is applied to the conductive plate. Our goal is to model the propagation of thermal energy through the plate.

The present study seeks to present an equation free based surrogate model using a weakly supervised method. It is able to transfer the dynamics of the system to a neural network in such a way that it produces correct solutions from the steady state of the conductive plate after applying the heat. It uses the observation of the minimum number of training data related to steady-state heat transfer.

The proposed weakly supervised method, in order to reduce the need for data, seeks to import the knowledge of the relevant field to a neural network by applying restrictions on the output space. In the steady-state heat transfer phenomenon, the pattern in the thermal data can be encoded in a kernel so that the main network (steady-state temperature distribution predictor) can produce correct solutions of the future state of the system without the need to observe (calculate) thermal field data.

2. Modeling Of Steady-state Heat Transfer Using Classical And Traditional Methods

Heat transfer using the traditional methods can be studied through three different approaches: mathematical methods, experimental methods and numerical calculations [11]. The mathematical approach was first popularized in the 17th to 19th centuries by pioneers such as Daniel Bernoulli, Leonard Euler, and Hermann von Helmholtz. The governing partial differential equation is one of the most important manifestations of mathematical modeling, which is used to model a number of phenomena. The complexity of mathematical methods in solving the governing partial differential equations, together with its inability to predict and calculate important factors and components, increased the tendency to use experimental methods. For example, The Wright brothers designed their aircraft based on experimental wind tunnel experiments. This trend continued into the twentieth century, and engineers used many experiments to analyze heat transfer problems. It was not practically possible to use experimental methods to understand all the facts and phenomena. Moreover, solving complex governing equations by mathematical methods with high accuracy requires very high time and computational power and is sometimes even impossible. Therefore, computational approaches called numerical modeling are used. Numerical modeling deals with the approximate solution of the differential equations governing the problem. Computational fluid dynamics is a computational method in which the governing equations of the physical processes are solved approximately using the numerical solution methods such as finite difference, finite volume, or finite element. The finite difference solution pattern of this form of equations (elliptic equations) is an iterative method. It is used to calculate relatively exact solutions for partial differential equations. It incorporates a precise rule that is derived from the analysis and interpretation of the partial differential equation governing the steady-state heat transfer.

Equation (1) shows the equation of steady-state heat transfer in a two-dimensional domain [12]. In this equation, T (x, y) represents the temperature of the point (x, y) after the application of heat at equilibrium moment.

$$\frac{\P^2 T}{\P x^2} + \frac{\P^2 T}{\P y^2} = 0 \qquad (1)$$

Solutions to the Laplace equation are the harmonic functions which satisfy the initial condition of the system. Since, the heat is applied only to the boundaries of the plate, Equation (1) is known as the Dirichlet boundary problem. In some cases, exact solutions to the equation are available. Otherwise, as mentioned, the equation must be solved approximately through a numerical approach such as finite difference [11]. Solving the two-dimensional steady state heat equation using the finite difference method involves discretizing the equation (1). The discretized form of the equation for node (i, j) would be:

$$T_{i,j} = \frac{(T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1})}{4} \qquad (2)$$

The nodal relation expressed in the equation (1) is solved iteratively by applying the rule above at each node (point in the grid) until convergence.

3. A Surrogate Model Based On Deep Learning For Steady-state Heat Transfer Simulation

Deep learning algorithms are able to detect a variety of linear and nonlinear relationships in data. These algorithms, like methods based on mathematical and numerical modeling, do not need to propose a pattern in the form of governing differential equations. There is no assumption about the nature of the distribution of the data observed in them. Deep learning algorithms based on deep neural networks, with very high flexibility allow models in principle to learn successively higher orders of features from the raw data input, making the application of deep learning modeling in physics highly attractive. There are already several examples of research exploring the application of deep learning techniques within the physics and engineering communities [13–16], including applications for accelerating fluid simulations in graphics generation [17] and shape optimization for drag reduction [18].

This approaches have shown deep learning models perform exceptionally well in classification and regression tasks. In this research, it will be shown that deep learning models are also effective and efficient at generating realistic samples from the arbitrary data distributions. The present study seeks to develop an equation free based surrogate model using deep learning algorithm algorithms to transfer physics and its governing laws to the deep neural network. This research eliminates the need to directly solve the differential equations governing the steady-state heat transfer. This model is used to predict the heat transfer in a two-dimensional space, with less computation, higher speed, and acceptable accuracy. In this section, first, the proposed method in the present study is introduced. Then the architecture and implementation of the proposed model will be described.

3.1. Approach

The aim of this research is to use the deep learning algorithms to train a convolutional neural network to generate a model that can directly infer the solution of the Laplace equation (1) when receiving the desired initial boundary conditions as input. In the proposed method, a convolutional kernel is used to encode the pattern of the thermal data so that using it, the main network is forced to produce correct solutions of the future state of the conductor without direct observation of the thermal data during training.

It is assumed that temperature is known only on the boundary points of the conductive plate. Therefore, the only non-zero areas in the input data are the borders. The zero values on the interior points indicate that the temperature of these points is affected by the neighboring points. The desired output is a matrix with the dimensions of the input data that represents the temperature values at any point on the plate after the heat transfer process has converged to a steady-state temperature distribution. In this research, a weakly supervised method has been used to model steady-state heat transfer.

According to figure 1, model training in supervised learning uses data and labels (i.e. the correct answer) related to the problem as input. The input is given to the deep learning algorithms to extract the pattern, the relationship between the data and the tag in one step. The output is in the form of a trained model. In this research, the data set is square matrices of zero values with different boundary conditions that show the temperature distribution on the plate. The label is also matrices with the size and dimensions similar to the input data that show the temperature distribution of the plate at the steady state form. An example of data (input) and label (output) is shown in Figure 5.

In this research, the introduced general approach for model training is a method consisting of two training steps. According to figure 2, in the first step(a), a limited number of task-related labels, which refer to steady-state heat transfer data in two-dimensional space and on a small scale (8 × 8), are given to deep learning algorithm. Then, from these labels, the patterns that describe the steady-state heat transfer in the two-dimensional space are extracted and encrypted. This pattern is shown in figure 4. Then, in the second step(b), the pattern extracted from the first stage, along with the quantified data with different random values, is given to the deep learning algorithms.
In fact, this approach uses a pattern derived from the labels instead of labels with the training data to teach the main model. In supervised methods, that data is assumed to be the correct output for each input data that is closer to the correct response (i.e. label) corresponding to the input data. However, in this approach, the desired output would be data that follows the pattern extracted from the first step.

3.2. Implementation of the proposed method

In this research, convolutional neural networks have been used in order to implement the proposed method. The convolutional neural network architecture, an example of a modern deep learning architecture, is a multilayer stack of optimizable convolution operations which compute non-linear input transformations. Each operation in the stack transforms its input in a manner that increases the selectivity and uniqueness of the output representation. In the following, the architectures of neural networks used in the proposed method, its implementation and modeling to simulate the phenomenon of steady-state heat transfer will be described in detail.

3.2.1. The first step of implementation: Extraction of steady-state heat transfer pattern

The neural network used in the first step is a simple neural network with only a two-dimensional convolutional layer that has a learnable convolutional kernel of size 3 × 3. The bias value in this layer is set to zero. During the training process, the network encrypts the latent knowledge and laws related to the two-dimensional steady state heat transfer in its kernel. It then uses this knowledge to train the main network. Regardless of the original size of the data, it is possible to use steady-state heat transfer data in any size to train the network. It is because the steady state heat transfer pattern will be the same at any scale of a two-dimensional space. Therefore, the need to provide high-dimensional data will be eliminated by selecting the steady-state heat transfer data in the smallest dimensions and using them for large-scale

heat transfer simulations. Therefore, the two-dimensional data of size 8×8 with different random boundary conditions are generated and considered for training the first network. Each data is a square with zero values. The four edges of which are set with four different random temperatures. This data was given to a two-dimensional CFD simulator to calculate the steady-state temperature with high accuracy using the finite difference method. Then, the obtained data was used as data and the two-dimensional data with zero values were used as labels for training the mentioned network. The architecture of this network is shown in Figure3:

By training the network with steady state heat transfer data and passing the two-dimensional kernels across each data, the average square of the output values is calculated as the error associated with each data. The kernel resulting from the trained network on steady-state heat transfer data extracted from the CFD simulator of size 8×8, is shown in Figure 4.

In the resulting kernel, all kernel nodes have a value close to zero except the central node and four adjacent nodes. The nodes have the almost same value, which is equal to a quarter of the value of the central node. The value obtained from the kernel passing through any point of the two-dimensional steady-state heat transfer data is almost zero. Therefore, the temperature at each point on the two-dimensional space with equilibrium conditions should be the average of its four adjacent points. This indicates the finite difference method described in section 2.1. In fact, it is by iteration of this rule that the finite-difference method for solving partial differential equations typically solves this problem with high accuracy.

3.2.2. The Second step of implementation: generating the steady-state temperature distribution predictor model

After training the mentioned neural network and extracting the steady-state heat transfer pattern in two-dimensional space, the second step begins in order to train the steady-state temperature distribution predictor network. It draws the two-dimensional steady-state temperature distribution by receiving data with different boundary conditions. The training of this network is carried out through the generated network in the previous step. It incorporates a manner without observing any steady-state heat transfer data, only through the limitations that the first network applies to its output data.

The architecture of this network consists of two parts. The first part contains a two-dimensional convolutional encoder-decoder network adapted from the u-net architecture in article [19]. The convolutional network is comprised of several two-dimensional encoding convolutional layers that gradually decrease the input image size in each layer to reach an image of size 1×1 in the bottleneck. Then the decoder network layers (which consist of transposed convolutions on the output of the previous layer concatenated with the corresponding encoding layer), inversely expand the reduced representation to an input-sized image. Moreover, a large amount of low-level information is shared directly between the equivalent size layers in the two encoder and decoder networks by the use of skip connections [19]. The second part of the network consists of a series of lambda layers defined to pass the boundary values of the input to the output of the network. The architecture of this network is shown in Figure 5. The input

shows the two-dimensional squared data with the boundary conditions and the output is equilibrium conditions corresponding to input data.

The goal is for the output generated by the network to show the equilibrium distribution corresponding to the input data. The training of this network is carried out, using the model extracted by the neural network of the previous step. Because the pattern defined in kernel of the network states the conditions necessary to create a steady-state temperature distribution condition, it is able to persuade the temperature distribution predictor network to generate correct solutions from the steady state without using labeled thermal data. For this purpose, the training of the steady-state temperature distribution predictor network is performed through the network trained in the previous step (network containing the steady-state temperature distribution pattern), without observing any thermal data and only using the restrictions that apply to the output data. In this step, the pattern generated in the previous step and data with different random boundary conditions have been used by the deep learning algorithms to produce the final model that simulates steady-state heat transfer. For this purpose, the steady-state temperature distribution predictor network is designed and created and it is trained with the neural network produced in the previous step (which contains the steady-state temperature distribution pattern).
As shown in Figure 6, the network containing the steady-state temperature distribution pattern, with its frozen kernel, is connected to the end of the steady-state temperature distribution predictor network to train it. Temperature distributions with different boundary conditions after passing the first network enter the network containing the steady-state temperature distribution to calculate their level of conformity with the equilibrium conditions pattern defined in the kernel. The boundary values of the input data are transferred to the output boundaries of the main network before entering the network. This is in order to generate the equilibrium temperatures distribution at the output corresponding to the input data. The network with a convolutional kernel containing the steady-state temperature distribution pattern is set up that way the correct steady-state data generate after passing through the main network for values that were considered zero in this study. The deviation rate of the data generated from the network from the equilibrium conditions pattern is calculated by the network containing the steady-state temperature distribution pattern by passing the kernel across the data, according to Equation 3 .and it is used as a loss function to calculate the error of the data obtained from the steady-state temperature distribution predictor network:

$$loss = mean(Conv2D(\mathrm{ker}nel, output))^2 \quad (3)$$

By minimizing the above error during the training process, the network will gradually adjust its parameters so that the generated data is valid in the pattern defined in the kernel. Finally, after obtaining the desired accuracy, the network containing the steady-state temperature distribution pattern is separated from it. Now, the steady-state temperature distribution predictor network will be able to simulate equilibrium temperature distribution for any desired data outside the training data range.

According to the presented method in this research, the used training data are steady-state heat data of size 8×8 that can be used to simulate the heat distribution in high dimensions. Simulating the problem at high dimensions through the supervised learning and labeled data will require equilibrium heat distribution data at the same high dimensions. Providing the labeled data is one of the major challenges in deep learning tasks that require high processing time and computational costs.

4. Analyzing And Evaluating The Results Of The Proposed Model And The Supervised Model

In this section, in addition to the proposed model, a supervised model has been designed and trained using the labeled steady-state heat transfer data in order to evaluate the efficiency of the proposed model in equilibrium condition simulation. The network architecture used in the proposed model comprised of 12 hidden layers including 6 encoder layers and 6 decoder layers in the first part(u-net network), and 4 lambda layers in the second part in order to transfer 4 boundary values of the input to the output. Further details of this network architecture, including the number and type of layers, output dimensions, number of parameters of each layer, and how the layers are connected to each other, are shown in Table 1.

Table 1
Architectural details of the steady-state temperature distribution predictor network

layer number	layer type	output dimensions	parameters	previous layer
1	Input layer	(None, 64, 64, 1)	0	
2	conv2d	(None, 64, 64, 16)	160	1
3	batchnormalization	(None, 64, 64, 16)	54	2
4	activation	(None, 64, 64, 16)	0	3
5	conv2d	(None, 64, 64, 16)	2320	4
6	batchnormalization	(None, 64, 64, 16)	64	5
7	activation	(None, 64, 64, 16)	0	6
8	maxpooling2d	(None, 32, 32, 16)	0	7
9	conv2d	(None, 32, 32, 32)	4640	8
10	batchnormalization	(None, 32, 32, 32)	128	9
11	activation	(None, 32, 32, 32)	0	10
12	conv2d	(None, 32, 32, 32)	9248	11
13	batchnormalization	(None, 32, 32, 32)	128	12
14	activation	(None, 32, 32, 32)	0	13
15	maxpooling2d	(None, 16, 16, 32)	0	14
16	conv2d	(None, 16, 16, 64)	18496	15
17	batchnormalization	(None, 16, 16, 64)	256	16
18	activation	(None, 16, 16, 64)	0	17
19	conv2d	(None, 16, 16, 64)	36928	18
20	batchnormalization	(None, 16, 16, 64)	256	19
21	activation	(None, 16, 16, 64)	0	20
22	maxpooling2d	(None, 8, 8, 64)	0	21
23	conv2d	(None, 8, 8, 128)	73856	22
24	batchnormalization	(None, 8, 8, 128)	512	23
25	activation	(None, 8, 8, 128)	512	24
26	conv2d	(None, 8, 8, 128)	147584	25
27	batchnormalization	(None, 8, 8, 128)	512	26

layer number	layer type	output dimensions	parameters	previous layer
28	activation	(None, 8, 8, 128)	0	27
29	maxpooling2d	(None, 4, 4, 128)	0	28
30	conv2d	(None, 4, 4, 256)	295168	29
31	batchnormalization	(None, 4, 4, 256)	1024	30
32	activation	(None, 4, 4, 256)	0	31
33	conv2d	(None, 4, 4, 256)	590080	32
34	batchnormalization	(None, 4, 4, 256)	1024	33
35	activation	(None, 4, 4, 256)	0	34
36	maxpooling2d	(None, 2, 2, 256)	0	35
37	conv2d	(None, 2, 2, 512)	1180160	36
38	batchnormalization	(None, 2, 2, 512)	2048	37
39	activation	(None, 2, 2, 512)	0	38
40	conv2d	(None, 2, 2, 512)	2359808	39
41	batchnormalization	(None, 2, 2, 512)	2048	40
42	activation	(None, 2, 2, 512)	0	41
43	maxpooling2d	(None, 1, 1, 512)	0	42
44	conv2d	(None, 1, 1, 1024)	4719616	43
45	batchnormalization	(None, 1, 1, 1024)	4096	44
46	activation	(None, 1, 1, 1024)	0	45
47	conv2d	(None, 1, 1, 1024)	9438208	46
48	batchnormalization	(None, 1, 1, 1024)	4096	47
49	activation	(None, 1, 1, 1024)	0	48
50	upsampling2d	(None, 2, 2, 1024)	0	49
51	concatenate	(None, 2, 2, 1536)	0	50, 42
52	conv2d	(None, 2, 2, 512)	7078400	51
53	batchnormalization	(None, 2, 2, 512)	2048	52
54	activation	(None, 2, 2, 512)	0	53
55	conv2d	(None, 2, 2, 512)	2359808	54

layer number	layer type	output dimensions	parameters	previous layer
56	batchnormalization	(None, 2, 2, 512)	2048	55
57	activation	(None, 2, 2, 512)	0	56
58	upsampling2d	(None, 4, 4, 512)	0	57
59	concatenate	(None, 4, 4, 768)	0	58, 35
60	conv2d	(None, 4, 4, 256)	1769728	59
61	batchnormalization	(None, 4, 4, 256)	1024	60
62	activation	(None, 4, 4, 256)	0	61
63	conv2d	(None, 4, 4, 256)	590080	62
64	batchnormalization	(None, 4, 4, 256)	1024	63
65	activation	(None, 4, 4, 256)	0	64
66	upsampling2d	(None, 8, 8, 256)	0	65
67	concatenate	(None, 8, 8, 384)	0	66, 28
68	conv2d	(None, 8, 8, 128)	442496	67
69	batchnormalization	(None, 8, 8, 128)	512	68
70	activation	(None, 8, 8, 128)	0	69
71	conv2d	(None, 8, 8, 128)	147584	70
72	batchnormalization	(None, 8, 8, 128)	512	71
73	activation	(None, 8, 8, 128)	0	72
74	upsampling2d	(None,16,16, 128)	0	73
75	concatenate	(None,16,16, 192)	0	74, 21
76	conv2d	(None, 16, 16, 64)	110656	75
77	batchnormalization	(None, 16, 16, 64)	256	76
78	activation	(None, 16, 16, 64)	0	77
79	conv2d	(None, 16, 16, 64)	39928	78
80	batchnormalization	(None, 16, 16, 64)	256	79
81	activation	None, 16, 16, 64)	0	80
82	upsampling2d	(None, 32, 32, 64)	0	81
83	concatenate	(None, 32, 32, 96)	0	82, 14

layer number	layer type	output dimensions	parameters	previous layer
84	conv2d	(None, 32, 32, 32)	27680	83
85	batchnormalization	(None, 32, 32, 32)	128	84
86	activation	(None, 32, 32, 32)	0	85
87	conv2d	(None, 32, 32, 32)	9248	86
88	batchnormalization	(None, 32, 32, 32)	128	87
89	activation	(None, 32, 32, 32)	0	88
90	upsampling2d	(None, 64, 64, 32)	0	89
91	concatenate	(None, 64, 64, 48)	0	90, 7
92	conv2d	(None, 64, 64, 16)	6928	91
93	batchnormalization	(None, 64, 64, 16)	64	92
94	activation	(None, 64, 64, 16)	0	93
95	conv2d	(None, 64, 64, 16)	2320	94
96	batchnormalization	(None, 64, 64, 16)	64	95
97	activation	(None, 64, 64, 16)	0	96
98	conv2d	(None, 64, 64, 1)	17	97
99	lambda	(None, 1, 62, 1)	0	1
100	lambda	(None, 62, 62, 1)	0	98
101	concatenate	(None, 63, 62, 1)	0	100, 99
102	lambda	(None, 1, 62, 1)	0	1
103	lambda	(None, 64, 1, 1)	0	1
104	concatenate	(None, 64, 62, 1)	0	101, 102
105	concatenate	(None, 64, 63, 1)	0	103, 104
106	lambda	(None, 64, 1, 1)	0	1
107	concatenate	(None, 64, 64, 1)	0	105, 106

The structure and architecture of the supervised model, like the u-net network used in the first part of the steady-state temperature distribution predictor network in the proposed model. The architecture of both models is considered in exactly the same conditions in terms of the number of main layers, the number of

learnable parameters as well as the number of training data used for training. The input data to the monitored model network are matrices with zero values of size 64 × 64. The four edges are set with four different numbers between 0 and 100. The labels are the equilibrium condition corresponding to the square input data. The input data to the monitored model network are matrices with zero values in the size of 64× 64×64, the four edges of which are set with four different numbers between 0 and 100. The labels are the equilibrium condition corresponding to the square input data. In this experiment, the data generated by the finite difference method obtained using the CFD simulation is used as an accuracy criterion to measure the accuracy of the outputs generated by the networks of both models. In the model created by the proposed method, only 400 data of size 8 × 8 were used to extract the steady-state heat transfer pattern during this experiment. In this model, the steady-state temperature distribution predictor network is trained to minimize the error defined in Equation 2 without directly observing any heat distribution data. In contrast, the supervised model used 5,000 steady-state heat transfer labeled data to train its network. The two models have been compared and evaluated with each other from different aspects such as generated output data, changes in the percentage of average absolute error during the training process, as well as the method and process of learning to produce the steady state heat transfer data.

4.1. Comparison and evaluation of the proposed and supervised models

Figure 7 shows the outputs simulated by the proposed model and the supervised model. The first column of the figure shows the input data in this experiment. The input data to the models are matrices of size 64 × 64 with zero values, the 4 edges of which are set with 4 different temperatures between 0 and 100 degrees Celsius. Each matrix shows the thermal conditions of the conductive plate at the first moment of heat application. Therefore, the only non-zero entries in the input are at the boundaries. These areas are marked with dark blue in the figure. The 4 edges around the plate, which are marked with different colors, represent the different temperatures applied to the conductive plate in the first moment. The boundary conditions for the ten input data displayed in this column are listed in 4 different temperatures in Table 2. The data displayed in the next three columns show the temperature distribution of the input data at any point on the screen using the three different methods after the heat transfer process converges to an equilibrium temperature distribution. The data solved by the finite difference method are assumed as real equilibrium conditions and placed in the second column. The next two columns show the outputs obtained from the two proposed and supervised models in the relevant boundary conditions, respectively.

Table 2 Boundary conditions(in degrees Celsius) of the input data shown in Figure 7

number	left	right	Bottom	top
1	9	23	71	0
2	45	94	55	55
3	5	45	2	40
4	84	99	92	3
5	74	71	12	29
6	34	55	9	60
7	38	0	12	59
8	11	78	5	12
9	100	21	68.44	6.9
10	34	11	12	29

4.2. Comparison of changes in the percentage of the average per-pixel output error of the two proposed and supervised models during the training process

Figure 8 shows the changes in the percentage of the average per-pixel output error of the two proposed and supervised models during different periods of the training process in this experiment. The pink curve corresponds to the supervised model, and the blue curve corresponds to the proposed model. In this figure, the horizontal and vertical axes represent the training epoch and the average per-pixel output error corresponding to the relevant training epoch, respectively. This error is calculated using the equation (4). y_{true} is the correct equilibrium temperature distribution matrix obtained using the finite difference method in this experiment. $y_{prediction}$ also represents the temperature distribution matrix predicted by the relevant method. The mean operator calculates the average values of the resulting matrix.

$$mape = mean\left\{\frac{|\ y_{true} - y_{prediction}\ |}{y_{true}}\right\} \quad (4)$$

As for the training procedure, experience so far indicates that while training deep neural networks, it is often useful to reduce the learning rate as the training progresses [20]. In this experiment, using the Adam optimizer[21], the learning rates of the two models were set with different values in descending order from 10^{-3} to 10^{-6} during consecutive courses during the training process and is shown in Table 3.

The training of the supervised and proposed models has been done during 4000 and 2200 epoches, respectively, with batch size in size of 128 samples. Every iterations of the optimizer on a single NVIDIA Tesla K80 GPU card takes around 18 seconds, for both models. Before starting the learning process, the weights of the networks of both models were set to random with identical values. Therefore, both have started the training process with the same weights. At the beginning of the learning process, the mean absolute percentage error (MAPE) for both models is 87%. According to Figure 8, the error reduction process is performed at a very high speed in early training when the simulation error of both networks is high. However, this process gradually becomes slower by gaining more accuracy. In the initial moments of training, the simulation error curve of the supervised model has a much steeper slope than the proposed model, which shows that in early training the learning ability of the supervised model is much higher than the proposed model.

In this model, the error reduction process does not last long, so that it slows down sharply from the 200th epoch onwards. Finally, the training of this model stops after about 4000 epoches, gaining an average error of 2.19 percent per pixel. In contrast, it has lasted longer in the supervised model, although the error reduction process is relatively slow from the beginning. In this model, the error reduction process has become faster gradually with more training epochs. Until the curves of the both models intersected each other at the epoch 1800 with error rate of 4.9%. From this epoch onwards, the proposed model has fewer errors than the supervised model.

The slope of the curve related to the mentioned model slows down after about 2000 epochs, and finally this model succeeds to obtain a lower error rate with a smaller number of training epochs. The training of this model ends after 2400 training epochs with the average output error rate of 0.68% per pixel. From the analysis and comparison of the two curves, it can be infered that the learning speed of the supervised model is much higher in the early training, but then the proposed model succeeds in gaining higher accuracy with a smaller number of training epochs. This indicates the high learning capability of the proposed model in simulating the two-dimensional heat transformation. Moreover, the training epochs of the supervised model is faster than the supervised model with 1600 less training epochs.

Table 3 The learning rates of the two proposed and supervised models during different training epochs in the training process

proposed model		supervised model	
epoch	learning rate	epoch	learning rate
600	10^{-3}	300	10^{-3}
1000	8×10^{-4}	2500	8×10^{-4}
200	5×10^{-4}	1000	2×10^{-4}
200	10^{-4}	200	10^{-4}
200	5×10^{-5}	200	10^{-5}
200	10^{-5}	200	10^{-6}

4.3. Comparing the learning process of the two proposed and supervised models during the training process

Table 4 shows the learning process of the two proposed and supervised models in steady-state heat transfer simulation during different training epochs, and compares them with each other. The initial temperature distribution of the square control volume shown in the table at the left, right, bottom and top borders of the conductive plate is 34, 55, 9 and 60 ° C, respectively. The information displayed in each column of the table shows the output of the desired model and the average per-pixel output error of the model in the relevant training epoch. Although the two models eventually produced the same output, they each adopted a different learning method to produce the desired output. According to the information in the left-hand column of the table, which describes the learning process of the supervised model, early in the training, the network tends to generate outputs that draw an overall structure of the equilibrium condition on the output by observing the steady-state heat transformation data. It then slowly adds more detail to the desired output. In this model, the fundamental changes are applied in the first training epochs and intangible changes are made to the output image from the 250 epoch onwards. The right - hand column of the table shows the output of the proposed model in each training epoch. The proposed model tends to reduce the value obtained by passing the kernel containing the equilibrium conditions pattern across the output. Therefore, it is taught by this criterion that the temperature of each point should be the average of the temperatures of the four adjacent points.

Consequently, a plate with constant temperatures also satisfies the desired conditions. Since the borders of a plate with a size of 64 × 64 form a relatively small fraction of the whole plate, early in the training, the model tries to satisfy the pattern in the kernel by producing outputs in which the entire plate is zero except for the early training. As the learning process progresses and with the aim of further reducing the amount obtained from the kernel passing across the output, the network gradually moves towards producing outputs that follow this pattern to a greater extent. Therefore, in the early training, compared to

the proposed model, there is a large distance between the correct output (correct equilibrium heat distribution) and the learning process is very slow.

However, the output produced by this model is more accurate. It is due to the fact that if follows the principle of the equilibrium temperature pattern encrypted in the kernel. The supervised model draws an overall structure of the equilibrium heat distribution from the beginning and then completes it by adding more details of the equilibrium temperature conditions. However, in the proposed model, a completely opposite procedure occurs. In this model, the temperature lines are smooth from the beginning and possess details, but the desired overall structure is achieved by more training epochs. Therefore, the model has a high error rate early in the learning process, but over the time and with the formation of overall structure of the equilibrium heat distribution, the error reduction process proceeds faster than the supervised model.

5. Conclusions

In this study, the two-dimensional steady-state heat conduction problem is investigated and an approach based on deep learning is presented that is able to simulate the equilibrium heat distribution without using the differential equations under the arbitrary boundary conditions and outside the range of training data observed by the steady-state temperature distribution predictor network with good and acceptable accuracy. Although the data generated by the numerical modeling using the finite difference method are very accurate, it requires high computational power, especially for the production of large-size data. The proposed method is able to generate output with acceptable accuracy by using deep learning in a shorter time and also by spending much less computational amount than the finite difference method. Using a weakly supervised learning algorithm, this method encods the local laws defined in the two-dimensional steady-state heat transfer in a convolutional kernel and uses this pattern to simulate the future state of a conductive heat transfer in a steady state form. The proposed deep learning model, through the use of two neural networks, is able to extract the steady-state temperature distribution pattern using a small amount of steady-state heat transfer data in small size and use it to modeling in large sizes. Due to the high accuracy of the proposed model compared to the supervised model which uses a large amount of large-size labeled data for network training, it can be said that the proposed method, in addition to being a cost-effective method in terms of the need to provide data in large volume and size. It has acceptable simulation accuracy.

Declarations

Ethical approval

This article has been simulated data of heat transfer on a two-dimensional heat transfer on a solid plate and no human data collection was involved in this research, therefore no ethical letter approval is needed.

Funding details

This research was not supported by any outside funding

Conflict of interest

This research does not have any conflict of interest

Informed Consent

This research does not need any informed consent

Authorship contributions

Authors contributed equally on all aspect of research and writing the article

References

1. Irwin M, Wang Z (2017) Dynamic Systems Modelling. Ohio State University, USA
2. Krizhevsky A, Sutskever.I, Hinton GE (2012) ImageNet Classification with Deep Convolutional Neural Networks, in Advances in neural information processing systems pp. 1097–1105
3. Johnson M, Schuster M, Le Q, Krikun M, Wu Y, Chen Z, Thorat N, Viégas F, Wattenberg M, Corrado G, Hughes M, Dean J (2017) Google's Multilingual Neural Machine Translation System: Enabling Zero-Shot Translation. Transactions of the Association for Computational Linguistics 5:339–351
4. Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press, Cambridge
5. El-Amir HHamdyM (2019) Deep Learning Pipeline. Building a Deep Learning Model with TensorFlow, Apress
6. Khoshgoftaar ShortenC (2019) T.M. A survey on Image Data Augmentation for Deep Learning. J Big Data **6**, pp.60
7. Zhang Y (2013) Speech Recognition Using Deep Learning Algorithms, Computer Science
8. Nassif AB, Shahin I, Attili I, Azzeh M, Shaalan k (2019) Speech Recognition Using Deep Neural Networks: A Systematic Review, in *IEEE Access*, vol. 7, pp. 19143-19165
9. Zong Z, Hong C (2018) On Application of Natural Language Processing in Machine Translation, *3rd International Conference on Mechanical, Control and Computer* Engineering *(ICMCCE)*, Huhhot, pp. 506-510
10. E BirdRBStewartW, Lightfoot EN (2007) Transport phenomena. John Wiley & Sons
11. Ganji D, Sabzehmeidani Y, Sedighiamiri A (2018) Nonlinear System in Heat Transfer Mathematical Modeling and Analytical Methods. 35-36
12. R,C, Daileda (2012) The two dimensional heat equation. Trinity University,San Antonio,Texas, March
13. Lu C, Liu Q, Sun Q, Hsieh C, Zhang S, Shi L, Lee C (2020) Deep Learning for Optoelectronic Properties of Organic Semiconductors. The Journal of Physical Chemistry C 2020 124(13):7048–7060
14. Strubbe RyczkoK (2019) I.Deep Learning and Density Functional Theory

15. Bikmukhametov.T, Jäschke.J. Combining machine learning and process engineering physics towards enhanced accuracy and explainability of data-driven models,Elsevier BV.138
16. Frank M, Drikakis D, Charissis V (2020)Machine-Learning Methods for Computational Science and Engineering, Computation
17. Tompson J, Schlachter K, Sprechmann P, Perlin K (2019) *Accelerating eulerian fluid simulation with convolutional networks*. Paper presented at 5th International Conference on Learning Representations, ICLR 2017, Toulon, France
18. Guo X, Li W, Iorio F (2016) Convolutional neural networks for steady flow approximation, in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'16) (ACM, New York, NY, USA, 2016)
19. Ronneberger O, Fischer P, Brox T (2015) U-Net: Convolutional Networks for Biomedical Image Segmentation. in International Conference on Medical Image Computing and Computer-Assisted Intervention, Springer International Publishing, 234–241
20. Raissi M, Yazdani A, Karniadakis G (2018) Hidden Fluid Mechanics: A Navier-Stokes Informed Deep Learning Framework. for Assimilating Flow Visualization Data, Science
21. Kingma DP, Ba j (2014) Adam: A Method for Stochastic Optimization, computer science, conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015

Table 4

Table 4 is available in the Supplementary Files section

Figures

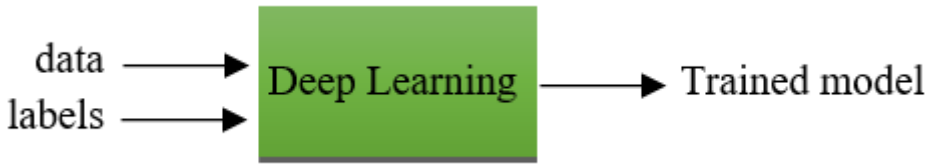

Figure 1

Model training by supervised method

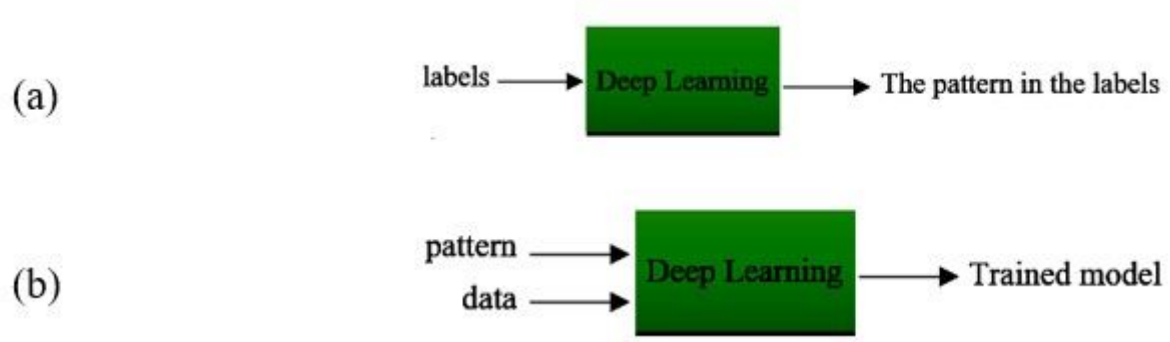

Figure 2

The first(a) and second(b) steps of model training in the proposed method

Figure 3

Network containing the steady-state temperature distribution pattern architecture

2.21×10^{-6}	-2.52×10^{-2}	3.45×10^{-7}
-2.52×10^{-2}	1.01×10^{-1}	-2.52×10^{-2}
3.20×10^{-7}	-2.51×10^{-2}	3.24×10^{-5}

Figure 4

The kernel resulting from the training of the network

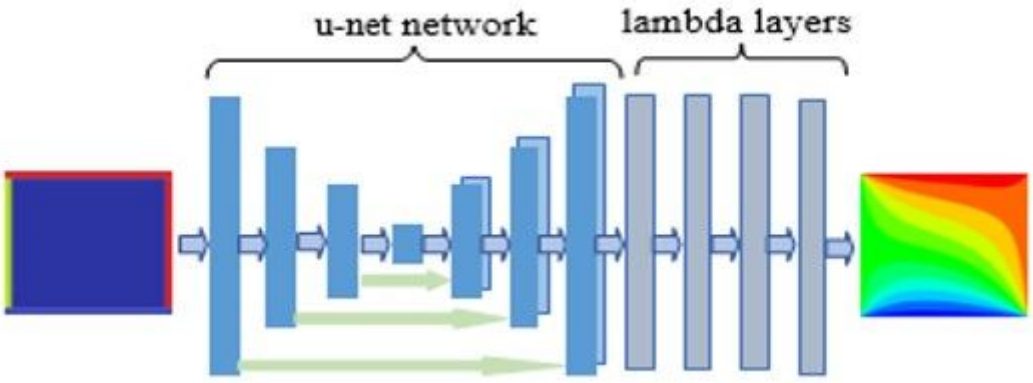

Figure 5

The steady-state temperature distribution predictor network architecture

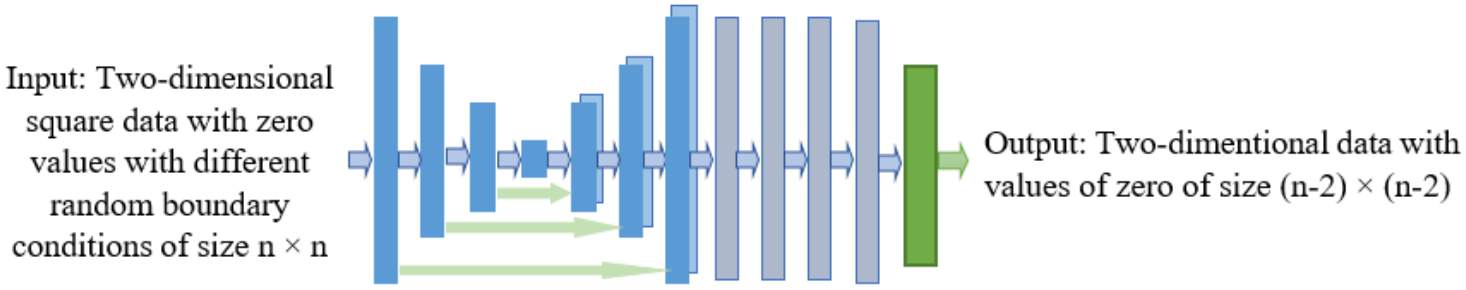

Figure 6

The steady-state temperature distribution predictor network training

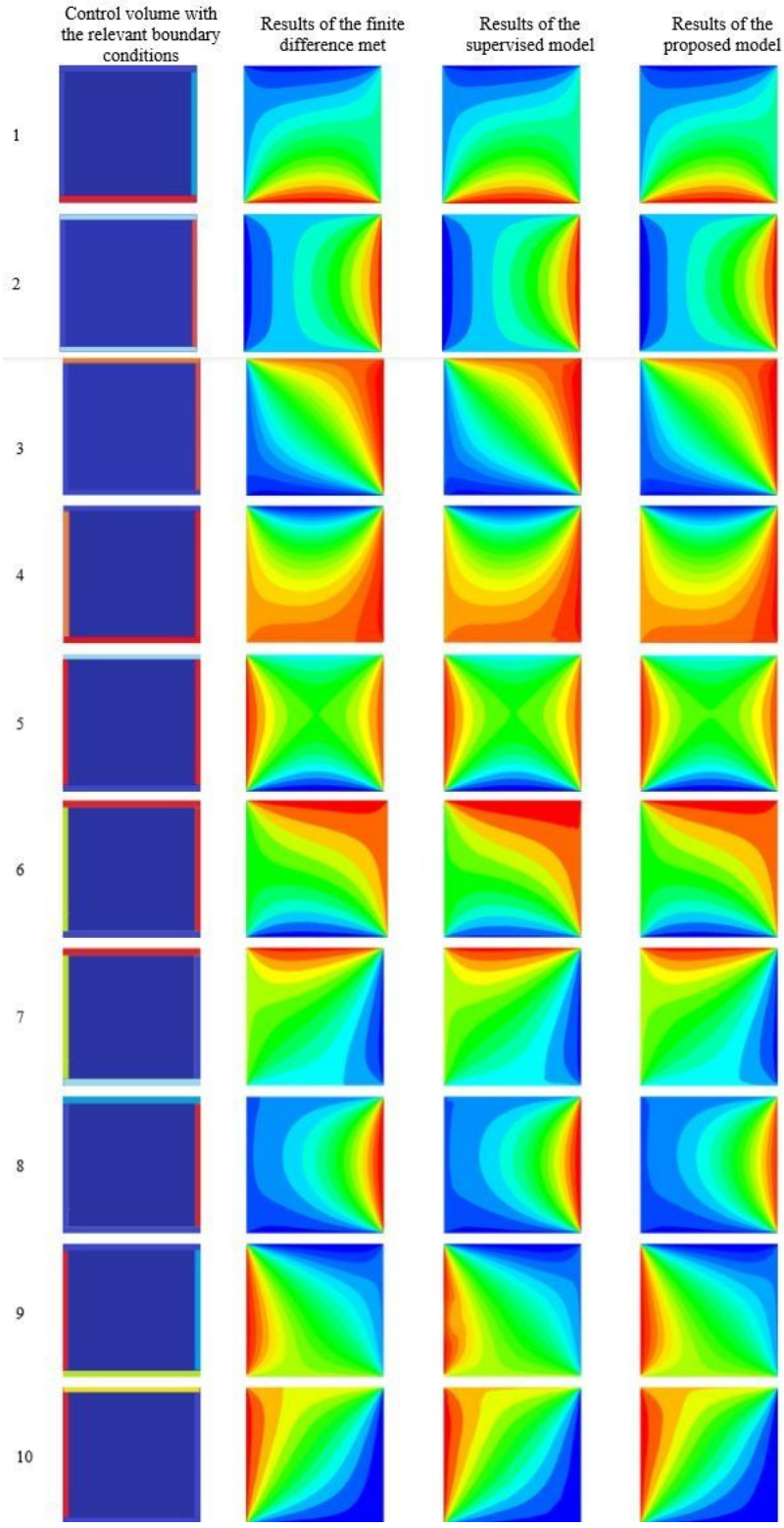

Figure 7

Results of the equilibrium heat distribution simulation with different boundary conditions in size of 64 × 64 × 64, using finite difference method, supervised and proposed models

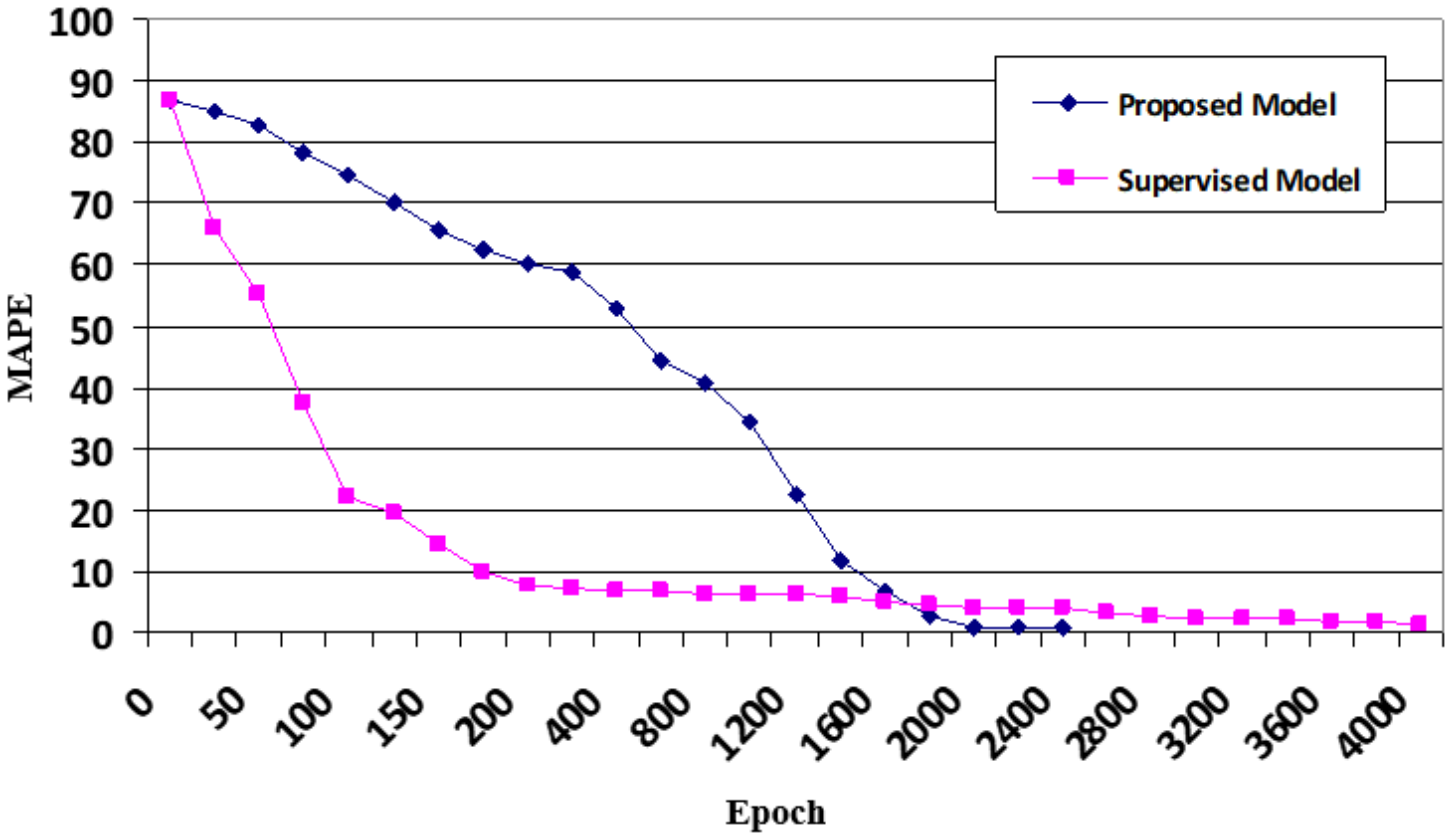

Figure 8

Comparing the mean absolute percentage error curves of the two proposed and supervised models during the training process

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- Table4.docx