Deep Learning Prediction of Heat Propagation on 2-D Domain via Numerical Solution

Behzad Zakeri[1(✉)], Amin Karimi Monsefi[2], and Babak Darafarin[3]

[1] University of Tehran, Tehran, Iran
Behzad.Zakeri@ut.ac.ir
[2] Shahid Beheshti University, Tehran, Iran
A_karimimonsefi@sbu.ac.ir
[3] Amirkabir University of Technology, Tehran, Iran
babak.darafarin@aut.ac.ir

Abstract. Deep learning's role in tackling complicated engineering problems becomes more and more effective by advances in computer science. One of the classical problems in physics is representing the solution of heat propagation in the arbitrary 2-D domain. Study of two-dimensional heat transfer provides a precious bed for related physical issues. In this work, by using finite volume method, we solved the two-dimensional heat equation on the arbitrary domain with specified limitations (considering three heated rectangular obstacles inside the main domain) for 100000 different cases. These cases were divided into big batches in order to reduce the computational cost. The solution for each case was used as sample data to train our deep neural network. After the training process, deep learning results have been compared to results which were produced by the commercial program (ANSYS). After analyzing deep learning efficiency, obviously, our network successfully was able to predict the solution of heat transfer physics with satisfactory precision.

Keywords: Deep neural network · Regularization · Laplace equation · Heat conduction

1 Introduction

Deep learning as a subset of artificial intelligence plays a significant role in various studies, and nowadays, in most of the complicated cases, deep learning is being used to simplify complex computations. Sound recognition [24], pattern recognition [12], and suggestion of relevant topics on the Internet websites [27] are the only small numbers of enormous applications of this powerful tool. The key feature of deep learning is that the layers which are used in learning procedure are not designed by the human, and they perform by using data which is fed to the

M. Bohlouli et al. (Eds.): CiDaS 2019, LNDECT 45, pp. 161–174, 2020.
https://doi.org/10.1007/978-3-030-37309-2_13

network as an inputs [16]. This feature of deep learning makes it an appropriate choice for representation systematic solution for a wide range of problems with high complexity [2].

On the other hand, Partial Differential Equations as cornerstones of studying dynamical systems, portray a precise image of many natural processes [23]. Although using PDEs to describe natural phenomena is an elegant way of modelling, several factors such as high dimensionality, and complex domains, made it tough to perform analytical solutions in most cases [10]. Representing new methods for solving partial differential equations have always been an interesting subject in computational science [8]. By advances in machine learning and specifically in deep learning the idea of using artificial neural networks for solving differential equations became more favorable [5,33].

The utility of deep learning method compared to other existing numerical methods for solving differential equations has several advantages. Deep learning provides a valuable approach to deal with uncertainties and nonlinearities in differential equations like stochastic PDEs [21]. Furthermore, it is noticeable that deep learning method is a perfect tool to prevent instability in solving procedure. Instability in other numerical methods is a big concern and in some cases caused divergence in solving procedure [1]. In deep learning algorithms, the stability of the solution highly dependent on weights W which are chosen by solver itself [25]. The heat equation is one of the most important equations in mathematical physics and engineering [22]. Heat conduction has been modelled by Joseph Fourier using the second-order partial differential equation [9]. Two-dimensional heat conduction (known as ***Laplace*** equation) in the rectangular domain is one of the famous classical problems in engineering mathematics and heat transfer, and there are extended numerical and analytical solutions for it [3,14]. However, if the domain of the solving equation changes from rectangular to an arbitrary domain, it will be really tough to represent the analytical solution for that case, and the only way would be using numerical methods [19].

We are interested in establishing a new method for studying natural phenomena which are modelled by PDEs, specifically problems related to fluid dynamics and heat transfer. In this paper, we focus on the case of 2-dimensional steady state heat propagation inside the rectangular domain considering three smaller rectangular obstacles with different temperatures inside it. We solved the governing equation for a large number of cases with different positions, sizes and temperatures in obstacles by taking advantage of the finite difference method. By changing the conditions of the problem and providing various situations, we provide sufficient labelled inputs for our learning algorithm.

The fully connected deep neural network has been designed with several hidden layers for this case. Since the number of input data was extremely large, over-fitting methods have used to prevent this phenomenon in the learning

procedure. After the learning process, the outputs of the network compared to results for the same conditions which were produced by commercial software for solving heat transfer and fluid flow problems, ANSYS FLUENT 19.0. Moreover, in the specified case with no heated obstacle, the deep learning result has been compared to the analytical solution extracted by orthogonal functions using Fourier series.

2 Related Work

This work is a combination of two separate fields of study pursued by a wide range of research communities. In this perspective, it is tried to find the best and optimum deep learning algorithm to find the solution of the specified heat conduction problem by taking advantage of the finite volume method in order to generate learning data.

Laplace equation plays an important role in many scientific fields, such as complex analysis [29], electromagnetic fields [34], fluid flow and heat transfer. One of the first successful tries for solving this equation on the arbitrary domain using numerical methods has been performed by Bruch and Zyvoloski for heat conduction purposes in 1974 [4]. Although mesh based numerical methods are strong tools in engineering problems, stability, the dependency of the solution to the mesh and Necessity of resolving the problem by changing the conditions of the problem are disadvantages of these methods, and motivate scientists to search for analytical solutions or at least mesh-less methods [17]. Many efforts went on finding an analytical solution for the Laplace equation on the arbitrary domain. Crowdy represent an analytical solution for potential flow (Laplace equation) past through obstacles on the infinite domain [6]. However, his attempts cannot solve the same problem for heat propagation in a finite domain, because of the difference in the boundary conditions.

Deep learning as an intelligence tool for prediction of the behaviour of the dynamical systems widely has been used in thermal-fluid sciences. Miyanawala and Jaiman has conducted an efficient deep learning technique for the model reduction of the unsteady Navier-stocks equation flow problems [20]. Several other types of research have been done related to the simulation and prediction of the fluid flow dynamics [11,13,18,31].

Since predicting the solution of differential equations using deep neural networks requires having a large number of labelled correct input data, weakly supervised learning algorithm using appropriate chosen convolutional kernel would be could be a good choice for simple cases. This method can learn directly from the initial condition [26]. Although this technique (which known as the physical informed network) predict the solution with good accuracy for simple physics, we focus on conventional learning methods to study the accuracy of these methods in dealing with such problems.

3 Methodology

This section is divided into two main parts. Firstly, the physics of heat propagation in a two-dimensional domain (Ω) explain briefly, and also it is described how input data for the learning procedure have been generated. In the second part, the deep learning approach and our algorithms are discussed.

3.1 Heat-Transport

Looking at details of two-dimensional heat propagation mathematics, the first step in this part is defining a proper space for solving the governing equation. We have decided to choose the square domain for solving the Laplace equation. Each side of the domain has its own boundary condition. Inside the main square, three heated rectangular obstacles with arbitrary sizes and positions have been considered Fig. 1. The aim of this definition is training our deep neural network to learn the pattern of heat propagation, and make it able to predict the correct temperature contours when the user gives the boundary conditions, positions and sizes of the obstacles.

It is important to note that because of the high computational cost, we consider the same temperature for all three obstacles, although this assumption can be changed easily.

The general heat conduction formation known as 2-D transient heat conduction can be written in the form of Eq. 1 :

$$\frac{\partial T}{\partial t} = \alpha \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) \tag{1}$$

where $\alpha > 0$ is a coefficient of the thermal diffusivity of the plate, and $T = T(x, y, t)$ demonstrates the temperature value in the given position and time.

However, in this study, we are interested in studying the steady state of heat transfer. So, the Eq. 1 would reform to Eq. 2 :

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \tag{2}$$

To solve Eq. 2 we need to specify the boundary conditions of the problem.

To solve the equation. in the simple rectangular domain with simplified boundary conditions, there are several analytical methods, such as separation of variables and using the error function. However, these methods in dealing with more complicated B.C or domains become useless, and it is necessary to use numerical methods.

In this work, we considered constant temperature on our boundaries which is known as the Dirichlet boundary condition [7].

Equations 3 and 4 depict the boundary conditions definition as follows :

$$\partial\Omega = \sum_{i=1}^{4} \partial T_i + \sum_{s=1}^{3} \partial A_s \tag{3}$$

$$T|_{\partial\Omega} = cte \tag{4}$$

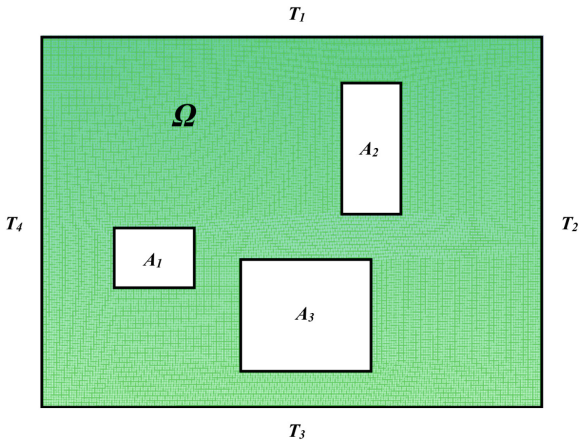


Fig. 1. Sample domain with obstacles

3.2 Data Generation

In order to provide proper input data as nourishment of the deep learning algorithm, the Laplace equation has been solved numerically for various conditions. These data after some treatment have been used in the input layer to train the network correctly.

Finite Volume Method. There are several numerical methods which iteratively solve equations which are not possibly solved by analytical methods. Finite Volume is one of the comprehensive methods that can deal with complex problems in solving differential equations. Although the concept of finite volume is based on 3-D problems, it can easily be extended to less topological dimensions [32].

To solve the Laplace equation using FVM, we need to discretize $\nabla^2 T = 0$. The temperature of the node (i, j) Fig. 2 calculates as follows :

$$\int_{\Delta V} \frac{\partial}{\partial x}\left(\frac{\partial T}{\partial x}\right) dx.dy + \int_{\Delta V} \frac{\partial}{\partial y}\left(\frac{\partial T}{\partial y}\right) dx.dy = 0 \tag{5}$$

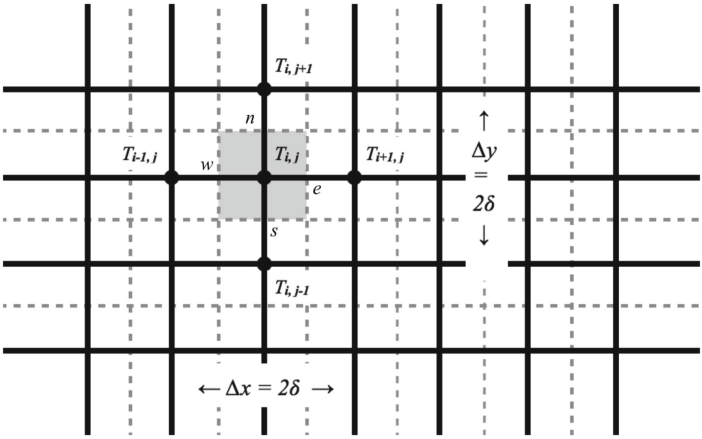


Fig. 2. Discretization of the domain

With assuming uniform square mesh and also considering linear temperature flux change along directions calculation continues as follows :

$$\Delta y = \Delta x \rightarrow A_e = A_w = A_n = A_s \tag{6}$$

$$\Gamma = \frac{A}{\delta} \tag{7}$$

$$4\Gamma T_p = \Gamma(T_w + T_s + T_e + T_n) \tag{8}$$

Based on Eq. 8, temperature of the node (i, j) can be calculated by Eq. 9 :

$$T_{i,j} = \frac{T_{i+1,j} + T_{i-1,j} + T_{i,j+1} + T_{i,j-1}}{4} \tag{9}$$

Equation 9 was solved iteratively with Dirichlet boundary condition until convergence.

Input Data Preparation. For easier analysis of produced data, we divide the solution of the Eq. 9 into 40 big batches. Each batch contains input and output files. The input file has been performed by 2500 combination of 19 separate elements, such as the width and height of the main domain, size and position of each rectangular obstacle, and also temperatures of each side of the domain. For each set of input elements, a specified solution has been assigned using Eq. 9.

Algorithm 1 demonstrates the procedure of solving the Eq. 9 for each input matrix by assuming discussed conditions.

Input:
width, height, top_temperature, right_temperature, left_temperature
bottom_temperature
first_rectangle, second_rectangle, third_rectangle, fixed_temperature
Result:
Temperatire Distribution
Initialization:
$\{T_{i,j}\}_{i=1,j=1}^{width,height} \leftarrow 0$
$\{T_{1,j}\}_{j=1}^{height} \leftarrow top\_temprature$
$\{T_{i,height}\}_{i=1}^{width} \leftarrow right\_temprature$
$\{T_{i,1}\}_{i=1}^{width} \leftarrow left\_temprature$
$\{T_{width,j}\}_{j=1}^{height} \leftarrow bottom\_temprature$
$SetFixedTempratureInRectangle(T, first\_rectangle, fixed\_temprature)$
$SetFixedTempratureInRectangle(T, second\_rectangle, fixed\_temprature)$
$SetFixedTempratureInRectangle(T, third\_rectangle, fixed\_temprature)$
$dt \leftarrow 0.25$
$TOL \leftarrow 1e-6$
while $error > TOL$ **do**
  tmp ← T
  for $i \leftarrow 1$ *to width* **do**
    for $j \leftarrow 1$ *to height* **do**
      if $\neg PointIsInRectangles(T_{i,j})$ **then**
        $tmp\_x \leftarrow tmp_{i+1,j} - 2 * tmp_{i,j} + tmp_{i-1,j}$
        $tmp\_y \leftarrow tmp_{i,j+1} - 2 * tmp_{i,j} + tmp_{i,j-1}$
        $T_{i,j} \leftarrow dt * (tmp\_x + tmp\_y) + tmp_{i,j}$
      else
        continue
      end
    end
  end
  $error \leftarrow Max(Abstract(Subtract(tmp, T)))$
end

Algorithm 1. Numerical data generation algorithm

Data Treatment. To prepare inputs of our deep neural network, firstly, by taking advantage of average and variance, our data has been normalized. Then, each element of the input matrices which indicates 19 initial data and address of that element (i and j) will be considered as input features for the neural network.

The output of the deep learning network will be compared to the solution for the corresponding element which is extracted from the output file.

In order to ensure that our network will not be biased by a small proportion of matrices, we have considered an ***acceptance rate*** to guarantee that no more than a specified percentage of elements will be picked from a certain matrix.

3.3 Deep Learning

Deep learning is formed by three main part which are the input layer, hidden layer and output layer. The input layer is a port for importing data into the network. These data have been sent to the network in matrix form. In this study, by using 21 neurons data was transferred from the input layer to the hidden layer. Hidden Layer contains several sublayers, and each of them is made by the specified number of neurons. This stage as the main part of the learning procedure should learn the way that our certain physics work and predict the correct temperature distribution. Finally, the output layer reports the results to the user.

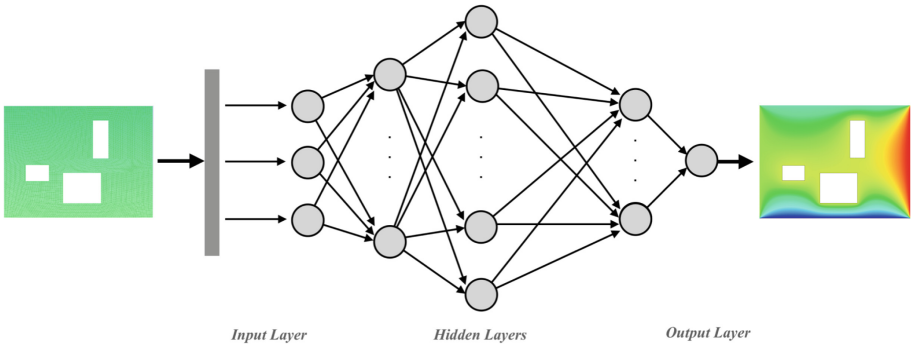


Fig. 3. Deep neural network diagram

Figure 3 illustrates the architecture of the deep learning process. In this architecture, the hidden layer consists of L layers. The schematic function of each neuron in the hidden layer can be shown as Fig. 4. Input data for each neuron receives from all neurons in the previous layer. These inputs by using vector weight (W) and the bias value (B), linearly combined ($\overrightarrow{W}X + B$) and the output result for the neuron is calculated. The process at each neuron will get finished by implementing the activation function. In this stage we used the ***Leaky Relu*** activation function as shown in Eq. 10 :

$$Leaky\text{Re}lu(x) = \begin{cases} x, & x > 0 \\ x * 0.01, & x \leq 0 \end{cases} \tag{10}$$

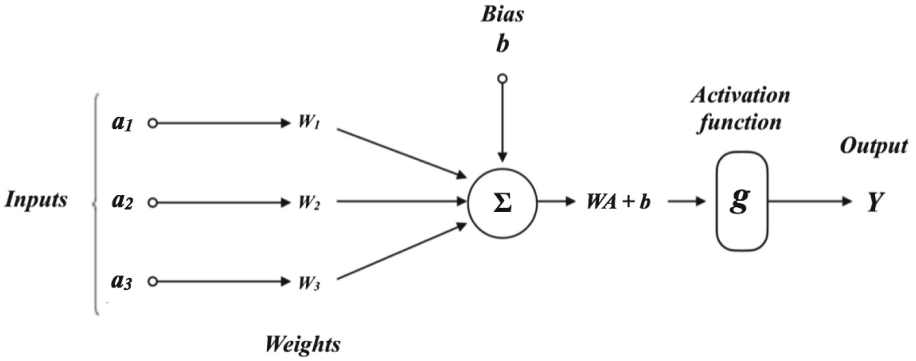


Fig. 4. Single neuron diagram

In general for layer l, according to the Fig. 4 output of the layer l is equal to $Z^{[l]}$ which is shown in Eq. 11 :

$$Z^{[l]} = W^{[l]} * A^{[l-1]} + B^{[l]} \tag{11}$$

Where $w^{[l]}$ is the weight matrix of input for layer l, $A^{[l-1]}$ is input of the layer, and $B^{[l]}$ is a vector of bias values of this layer.

Also, the input of the layer $A^{[l]}$ is defined as follows :

$$A^{[l]} = g^{[l]}(Z^{[l]}) \tag{12}$$

The function $g^{[l]}$ in Eq. 12 represent the activation function in layer l.

Before starting the learning procedure, the values of the $B^{[l]}$ are 0, and the elements of the matrix $W^{[l]}$ are initialized randomly between 0 and 1.

The purpose of the learning network is finding proper w and B for each layer which minimizes the error function.

$$\min_{W,B} J(W, B) \tag{13}$$

Where J is error function which is defined as follows :

$$J(W, B) = \frac{1}{m} \|Y' - Y\|_2^2 \tag{14}$$

In Eq. 14, Y' demonstrates amount of data which is generated by deep learning, also Y and m are real data and numbers of input data respectively.

To prevent over fitting three regularization techniques which are ***Dropout*** [28], ***Momentum*** [30] and ***Weight decay*** [15] have been utilized simultaneously.

After implementation of these three methods Eq. 14 reformed to Eq. 15 as follows :

$$J(W,B) = \frac{1}{m} \left\| Y' - Y \right\|_2^2 + \frac{\lambda}{2m} \left\| W \right\|_2^2 \tag{15}$$

Where λ is a coefficient which should be set in a way that minimizes the error function.

There are several optimization methods to minimize error function 15. In this work, we checked different optimizers to get the best accuracy, and finally, we chose ***SGD*** (Stochastic Gradient Descent) as our optimizer function. This algorithm, by updating the parameters θ_n of the object $J(\theta_n)$ (as shown in Eq. 16) tries to find the best parameters for minimizing the error function.

$$\theta_{n+1} = \theta_n - \alpha \frac{\partial}{\partial \theta_n} J(\theta_n) \tag{16}$$

In Eq. 16 θ is a vector parameter, also J and α are cost function and slope parameter respectively.

The *SGD* algorithm can estimate the gradient of the parameters only by using a limited number of training examples.

Finally, to find out the learning parameters we categorize the generated data into 3 main categories. From all generated data, 98% has been allocated for training, and the percentage of validating and testing was 1% for each. Also, for more precision and less run time, training data were divided into 1000 mini-batches.

4 Results

In this section, results which have been generated by deep learning is compared to true data by taking advantage of different experiments. In the first stage, deep learning's was analyzed based on the error rate in training and test time. The next step was comparing deep learning results by ANSYS answers. And finally, the accuracy of our network was analyzed by the utility of the analytical solution for the simplified case.

4.1 Analyzing Deep Learning Results

In this section, deep learning precision was analyzed by changes the number of epochs and varying threshold coefficient. For this purpose, we used a different number of epochs (from 100 to 2000) in the input layer. Also by changing the threshold coefficient, it is possible to monitor the effect of epochs in the final results. In this experiment, 98% of true data was considered for training the network, and 2% for validation and test.

The *Mean Square Error* index has been used to calculate the training and test error. This index is defined as follows 17 :

$$MSE = \frac{\sum_{i=0}^{n} \left(y^i - y'^i\right)^2}{n} \tag{17}$$

The *Threshold* concept has utilized in order to compare the true data with the results from the deep learning method. Whereas y' is the deep learning calculated quantity and y represents the amount of numerical solution generated by FVM.

If the threshold quantity was more than left-hand side of Eq. 18, then both values will be assumed as equal.

$$|y - y'| < \theta \tag{18}$$

Table 1. Epochs' number effect on deep learning results

Epoch number	Training error	Test error	*Th(1)*	*Th(0.1)*	*Th(0.01)*
100	0.984	4.125	75.12%	71.78%	67.47%
200	0.876	3.745	77.83%	74.34%	69.87%
300	0.821	3.424	79.47%	77.54%	72.39%
500	0.700	2.145	87.08%	83.75%	80.19%
1000	0.576	1.406	94.19%	91.07%	89.49%
2000	0.319	0.958	97.19%	93.67%	91.87%

Looking at the Table 1 in more detail, clearly, by increasing the epoch number, precision of the final results was increased for all thresholds. Also, by considering an epoch number, the precision decrease in smaller thresholds.

4.2 Comparison with ANSYS

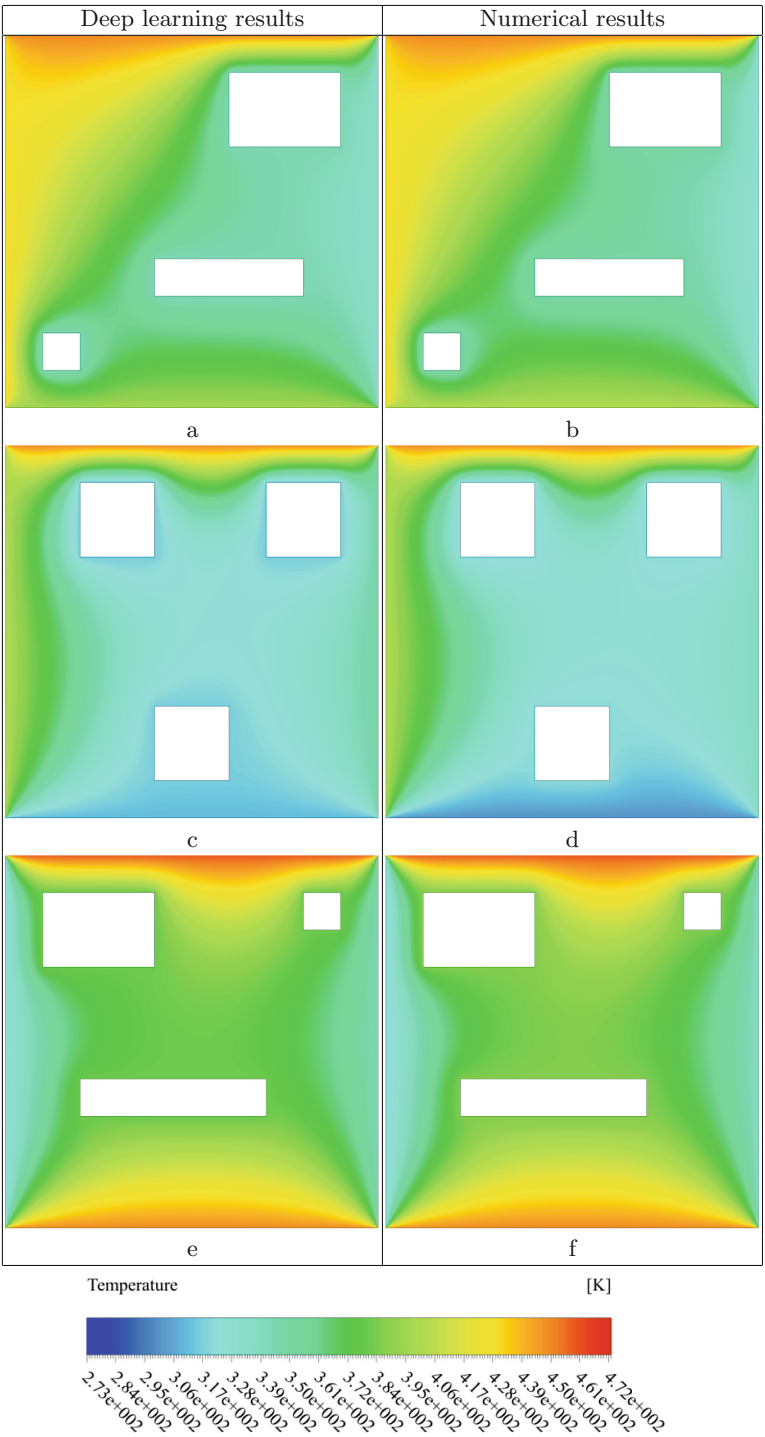
For engineering purposes, we need to visualize the results of computations to make it easier for engineers to have better judgment about them.

In this part, we compare the results which are extracted by deep learning algorithm with the output of the commercial program (ANSYS Fluent 19.0).

Same geometries with high-quality mesh have been generated and imported to the Fluent solver. All the computations have conducted by the second-order scheme, and the calculations have proceeded until the full convergence.

In Table 2 three sample cases of deep learning and numerical results are compared. Although ANSYS results were quite similar to deep learning output, in regions that thermal gradient was more than other areas, deep learning could not perfectly estimate the temperature distribution.

Table 2. Comparing deep learning with ANSYS results



5 Conclusion

We have shown that deep learning successfully can learn the physics of heat transfer in two-dimensional space. We found that there are various factors which directly influence the quality of the deep learning prediction, such as optimizer method, activation function and momentum variable. It is found that the stochastic gradient descent obviously has better performance in comparison to other optimizers. Our deep learning results sufficiently were similar to ANSYS results considering the number of data which were utilized for training the network. Overall, deep learning as a strong tool can provide an amazing method for representing the numerical solution for different kinds of PDEs.

References

1. Ascher, U.M.: Numerical Methods for Evolutionary Differential Equations. vol. 5. Siam (2008)
2. Baker, B., Gupta, O., Naik, N., Raskar, R.: Designing neural network architectures using reinforcement learning. arXiv preprint arXiv:1611.02167 (2016)
3. Bergman, T.L., Incropera, F.P., Lavine, A.S., Dewitt, D.P.: Introduction to Heat Transfer. Wiley (2011)
4. Bruch Jr., J.C., Zyvoloski, G.: Transient two-dimensional heat conduction problems solved by the finite element method. Int. J. Numer. Methods Eng. **8**(3), 481–494 (1974)
5. Chakraverty, S., Mall, S.: Artificial Neural Networks for Engineers and Scientists: Solving Ordinary Differential Equations. CRC Press (2017)
6. Crowdy, D.G.: Analytical solutions for uniform potential flow past multiple cylinders. Eur. J. Mech. B/Fluids **25**(4), 459–470 (2006)
7. Dirichlet, P.G.L.: Über einen neuen Ausdruck zur Bestimmung der Dichtigkeit einer unendlich dünnen Kugelschale, wenn der Werth des Potentials derselben in jedem Punkte ihrer Oberfläche gegeben ist. Dümmler in Komm (1852)
8. Fan, E.: Extended tanh-function method and its applications to nonlinear equations. Phys. Lett. A **277**(4–5), 212–218 (2000)
9. Grattan-Guinness, I., Fourier, J.B.J., et al.: Joseph Fourier, 1768-1830; a survey of his life and work, based on a critical edition of his monograph on the propagation of heat, presented to the Institut de France in 1807. MIT Press (1972)
10. Han, J., Jentzen, A., Weinan, E.: Solving high-dimensional partial differential equations using deep learning. Proc. Nat. Acad. Sci. **115**(34), 8505–8510 (2018)
11. Jeong, S., Solenthaler, B., Pollefeys, M., Gross, M., et al.: Data-driven fluid simulations using regression forests. ACM Trans. Graph. (TOG) **34**(6), 199 (2015)
12. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: convolutional architecture for fast feature embedding. In: Proceedings of the 22nd ACM International Conference on Multimedia, pp. 675–678. ACM (2014)
13. Kim, B., Azevedo, V.C., Thuerey, N., Kim, T., Gross, M., Solenthaler, B.: Deep fluids: a generative network for parameterized fluid simulations. arXiv preprint arXiv:1806.02071 (2018)
14. Kreyszig, E.: Advanced Engineering Mathematics. Wiley (2010)
15. Krogh, A., Hertz, J.A.: A simple weight decay can improve generalization. In: Advances in Neural Information Processing Systems, pp. 950–957 (1992)

16. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436 (2015)
17. Li, H., Mulay, S.S.: Meshless Methods and Their Numerical Properties. CRC Press (2013)
18. Ling, J., Kurzawski, A., Templeton, J.: Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. J. Fluid Mech. **807**, 155–166 (2016)
19. Minkowycz, W.: Advances in Numerical Heat Transfer. vol. 1. CRC Press (1996)
20. Miyanawala, T.P., Jaiman, R.K.: An efficient deep learning technique for the Navier-Stokes equations: application to unsteady wake flow dynamics. arXiv preprint arXiv:1710.09099 (2017)
21. Nabian, M.A., Meidani, H.: A deep neural network surrogate for high-dimensional random partial differential equations. arXiv preprint arXiv:1806.02957 (2018)
22. Narasimhan, T.: Fourier's heat conduction equation: history, influence, and connections. Rev. Geophys. **37**(1), 151–172 (1999)
23. Robinson, J.C.: Infinite-Dimensional Dynamical Systems: An Introduction to Dissipative Parabolic PDEs and the Theory of Global Attractors. vol. 28. Cambridge University Press (2001)
24. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. Int. J. Comput. Vis. **115**(3), 211–252 (2015)
25. Ruthotto, L., Haber, E.: Deep neural networks motivated by partial differential equations. arXiv preprint arXiv:1804.04272 (2018)
26. Sharma, R., Farimani, A.B., Gomes, J., Eastman, P., Pande, V.: Weakly-supervised deep learning of heat transport via physics informed loss. arXiv preprint arXiv:1807.11374 (2018)
27. Singhal, A., Sinha, P., Pant, R.: Use of deep learning in modern recommendation system: a summary of recent works. arXiv preprint arXiv:1712.07525 (2017)
28. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
29. Stewart, I., Tall, D.: Complex Analysis. Cambridge University Press (2018)
30. Sutskever, I., Martens, J., Dahl, G., Hinton, G.: On the importance of initialization and momentum in deep learning. In: International Conference on Machine Learning, pp. 1139–1147 (2013)
31. Tompson, J., Schlachter, K., Sprechmann, P., Perlin, K.: Accelerating eulerian fluid simulation with convolutional networks. arXiv preprint arXiv:1607.03597 (2016)
32. Versteeg, H.K., Malalasekera, W.: An Introduction to Computational Fluid Dynamics: The Finite Volume Method. Pearson Education (2007)
33. Yadav, N., Yadav, A., Kumar, M.: An Introduction to Neural Network Methods for Differential Equations. Springer (2015)
34. Zhang, K., Li, D., Chang, K., Zhang, K., Li, D.: Electromagnetic Theory for Microwaves and Optoelectronics. Springer (1998)