Artificial Neural Network in 1D Heat Transfer Problem

Theodoret P. Agatho[1, a)], Pranowo[1, b)], and Andi W. R. Emanuel[1,c)]

[1]*Department of Informatics, Faculty of Industrial Technology, University of Atma Jaya Yogyakarta, Babarsari St. No. 43, Yogyakarta, 55281, Indonesia*

[a)] Corresponding author: 190710233@students.uajy.ac.id
[b)]pranowo@uajy.ac.id
[b)]andi.emanuel@uajy.ac.id

Abstract. In this paper, an Artificial Neural Network (ANN) was used to help solve a one-dimensional unsteady conduction heat transfer problem. In this case, the derivatives of space were discretized by ANN, and the time was integrated by Euler's Method. All calculation programs were written in Matlab. Subsequently, the parameter and runtime of ANN were observed, and the results were compared to Finite Difference Method (FDM) results and the exact solution. A number of numerical experiments have been performed, and ANN showed a high degree of accuracy (with mean square error 10^{-4} and below) and the capability to outperform the accuracy of FDM in certain cases.

INTRODUCTION

ANN is well known for its capability to solve complex problems by taking the patterns in data where rigorous rules can't be defined, which means it can't be ruled rigidly. It is made possible through "learning" or "training", where the weight and bias adjustment of ANN is done iteratively based on error between prediction and observation data. Hence, this predominance behavior is seen as promising since it means many problems from a variety of fields can be solved by ANN alone as long as sufficient observed data is available. However, further implementation may lead to bigger data requirement and be susceptible to overfitting over observational data. The limitation on conventional ANN capability to process raw data [1] encourages the last decade of research to further enhance ANN performability.

Recent scientific research in mathematical modeling has found that this problem can be mitigated by taking physics into consideration. In this case, by satisfying the governing equation described by Partial Differential Equations (PDEs) rather than solely relying on data, ANN becomes comparable to numerical discretization, such as FDM, which is used to solve PDEs. Nevertheless, solving PDEs with ANN has advantages over numerical approaches, such as (1) its capability to acquire results when the numerical approaches are failing because the law is partially known [2] or because of the introduction of noise [3] and (2) its potential to yield a more accurate result [4].

The applications of physic law to ANN, known as part of the Physic-Informed Neural Network (PINN) family, have been extensively studied in the following refs [5-13]. ANN has been used to solve FDM in wave equations [5, 6], heat transfer equations [7], and Navier-Stokes equations [8, 9]. In addition, it's able to solve more complicated mathematical cases, such as fractional PDE [10] and high-dimensional PDE [11].

From the preceding discussion, it's clear that this matter is well-established and that further research is needed. This paper tries to take a closer look at the capability and limitations of ANN in the 1D Heat Transfer Problem.

The paper was divided into 4 sections: background and ideas were introduced in Section 1. Section 2 covered the basic theory and formulas of ANN and time-dependent PDEs. Results were discussed later in Section 3, and conclusions were drawn in Section 4.

THEORY BACKGROUND

Heat Transfer

Consider the following simple one-dimensional heat transfer equation, where the thermal conductivity, density, and heat capacity are constant over time:

$$\frac{\partial \mathrm{y}}{\partial \mathrm{t}} = \alpha \frac{\partial^2 y}{\partial x^2}, \tag{1}$$
$$0 < x < 1, y(0) = 0 = y(1),$$

Where $\alpha = 1$ and the initial value are $y_t = \sin(\pi x)$. Thus, the exact solution is $y_t = \sin(\pi x)e^{-(\pi^2 t)}$. When time is integrated by applying implicit Euler's method, equation (1) turns as follows:

$$\frac{y_{n+1} - y_n}{\Delta t} = \frac{\partial y}{\partial t}, \tag{2}$$

$$\frac{y_n}{\Delta t} = \frac{y_{n+1}}{\Delta t} - \alpha \frac{\partial^2 y_{n+1}}{\partial x^2}, \tag{3}$$

$$y_n = y_{n+1} - \Delta t\left(\alpha \frac{\partial^2 y_{n+1}}{\partial x^2}\right) \tag{4}$$

Artificial Neural Network

To solve the problem, a feed-forward neural network was used. The architecture of this ANN consisted of one input node x, one hidden layer consisting of N-many nodes, and one output node y. The Sigmoid function was used as the activation function (φ) in the hidden layer and gradient descent as the optimizer. It's worth taking note that the differentiation of network output y must be able to satisfy the underlining physics. Following is the used ANN architecture.

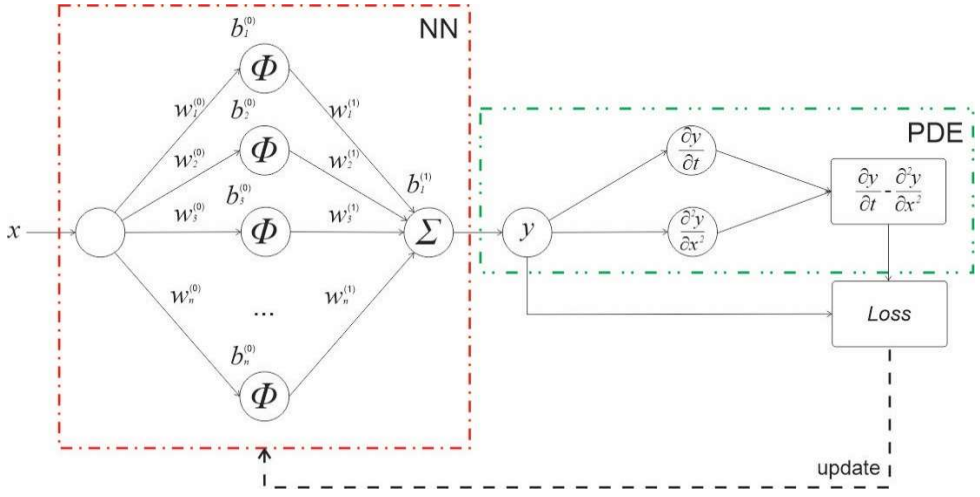

FIGURE 1. The purposed ANN Architecture

From the preceding equation (4), ANN was used to predict y_{n+1}, where y_n is a known constant, substituting FD. Therefore, y_{n+1} was calculated with the following formulation:

$$y_{n+1} = w^{(1)}\varphi\left(w^{(0)}x + b^{(0)}\right) + b^{(1)} \tag{5}$$

Where weight and bias values were initialized randomly in interval [-2, 2]. Weight and value adjustment are carried out similarly to ordinary ANN, which is based on loss, but preceding physic law, which contains boundary conditions, was also taken into consideration in loss; thus, the loss function is as follows:

$$l = \left\langle \left(y_{n+1} - \Delta t\left(\alpha \frac{\partial^2 y_{n+1}}{\partial x^2} \right) - y_n \right)^2 \right\rangle + \left(y_{n+1}(0)\right)^2 + \left(y_{n+1}(1)\right)^2, \tag{6}$$

Consequently, the gradient descent, which in this case is for the weight, is given as:

$$\frac{\partial l}{\partial w} = \left\langle 2\left(y_{n+1} - \Delta t\left(\alpha\nabla^2 y_{n+1}\right) - y_n\right)\left(\frac{\partial y_{n+1}}{\partial w} - \Delta t\left(\alpha \frac{\partial \nabla^2 y_{n+1}}{\partial w} \right)\right)\right\rangle + 2\left(y_{n+1}(0)\right)\left(\frac{\partial y_{n+1}(0)}{\partial w} \right) + 2\left(y_{n+1}(1)\right)\left(\frac{\partial y_{n+1}(1)}{\partial w} \right), \tag{7}$$

Where,

$$\nabla^2 y_{n+1} = \frac{\partial^2 y_{n+1}}{\partial x^2} = w^{(1)}\left(w^{(0)}\right)^2 \varphi\left(w^{(0)}x + b^{(0)}\right), \tag{8}$$

$$\nabla^2 \varphi\left(w^{(0)}x + b^{(0)}\right) = \left(1 - \varphi\left(w^{(0)}x + b^{(0)}\right)\right)\left(1 - 2\varphi\left(w^{(0)}x + b^{(0)}\right)\right), \tag{9}$$

And x is coordinate input. Equation (7) is also true toward bias (b) where weight (w) is substituted. The next prediction over the change of time, y_{n+1} before become current y_n and the learning process is repeated. This process is repeated until the accumulated time reaches the targeted time. The subsequent flowchart shows the program flow.

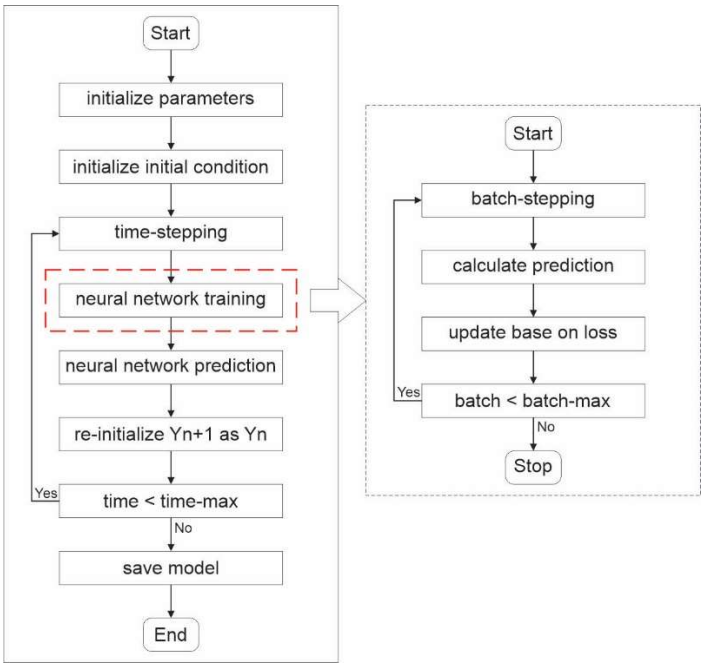

FIGURE 2. The purposed ANN Program flowchart

RESULT AND DISCUSSION

The calculations were done using MATLAB R2021b program on a laptop with an AMD Ryzen 5 5600H and 16 GB of RAM. ANN managed to achieve an accurate result with a low error compared to the numerical result error toward the exact solution. Various parameter tunings have been tested with 10 repetitions for each tuning, and the average runtime and Mean Square Error (MSE) were noted.

TABLE 1. ANN and FD results at total time is 0.1 and the total node of x is 41, while the learning rate of ANN is 0.001.

dt	Total neural	Total epoch	Total batch	Neural Network		Finite Difference	
				Average runtime (sec)	Average MSE	Average runtime (sec)	Average MSE
0.01	16	1024	128	3.060	1.65E-04	1.85E-04	1.51E-04
	32			3.877	1.24E-04		
	64			5.320	1.12E-04		
	16	2048	256	11.977	1.42E-04		
	32			15.329	1.38E-04		
	64			21.259	1.49E-04		
	16	4096	512	47.718	1.44E-04		
	32			62.042	1.46E-04		
	64			84.480	1.52E-04		
0.001	16	1024	128	30.326	3.45E-05	1.01E-03	1.94E-06
	32			38.575	6.29E-05		
	64			52.535	5.48E-05		
	16	2048	256	118.645	3.50E-06		
	32			150.967	1.01E-05		
	64			207.974	8.40E-06		
	16	4096	512	479.716	1.85E-06		
	32			610.386	3.17E-06		
	64			821.918	3.58E-06		

The best average Mean Square Error (MSE) found in tests compared to FD is 1.12E-04, which is 1.35 times better. A sample was taken to give a picture of the ANN result with the following parameters:

TABLE 2. ANN parameters with average MSE of 1.85E-06

Parameter	Value
dt	0.001
time	0.1
total node	41
learning rate	0.01
total neural	16
total epoch	4096
total batch	512

Precede parameters were taken because the following ANN managed to achieve lower average MSE to FD and was required to calculate the highest of iterations PDEs due to its low value of change over time (dt), hence increase the challenge for ANN. The following graph was created from parameters in Table 2, which took 494.613 seconds to finish.

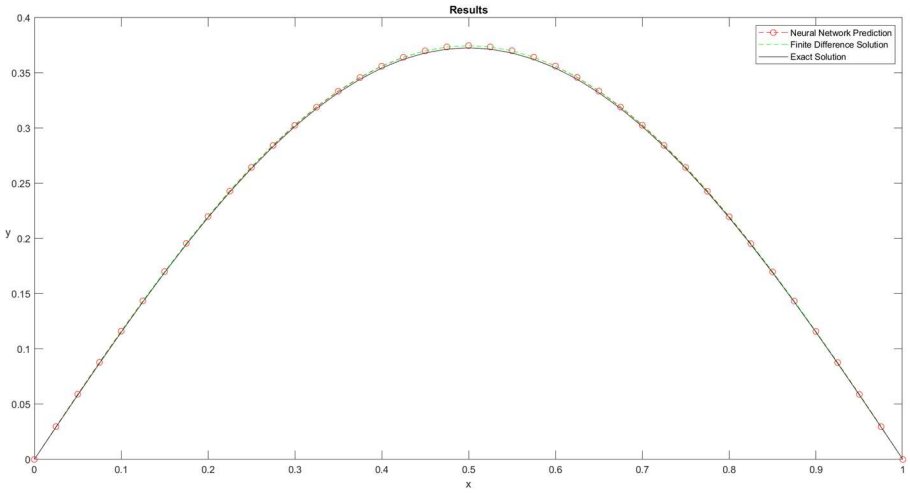

FIGURE 3. ANN, FDM, and exact solution curve result

The MSE of correspondent figure 3 is 1.49E-06, While the boundary error at $y(0)$ is 1.98E-05 and $y(1)$ is 1.64E-05. This shows ANN is able to outperform FD while maintaining physics, such as boundary conditions.

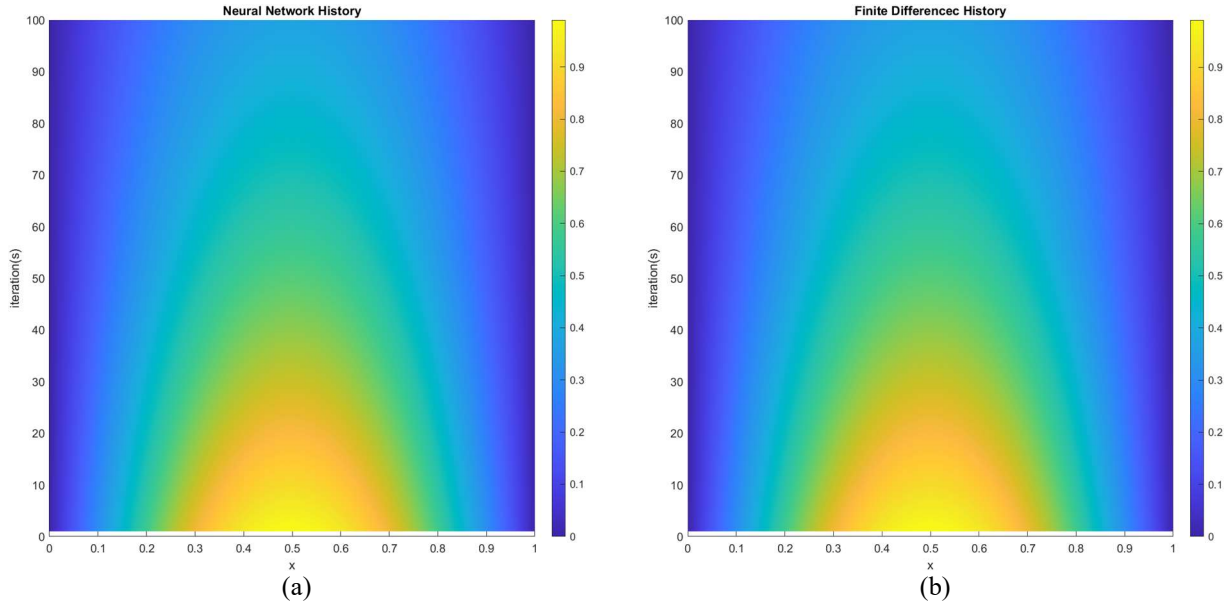

FIGURE 4. The history of temperature change on each point over time where (a) is with ANN and (b) is with FDM

CONCLUSION

ANN has been used to discretize derivatives of space, where time was integrated by Euler's Method. The implementation of ANN as an alternative to FD showed promising results, as it managed to outperform FD in terms of error in several parameter tunings, as shown in the previous discussion. Furthermore, ANN managed to predict the whole unknown partial differential result rather than only partially. Nevertheless, the lengthy running time of ANN is a major setback.

ACKNOWLEDGMENTS

The work has been financially supported in part by University of Atma Jaya Yogyakarta. Corresponded Matlab programs are available in here [14]. The ANN program writing is influenced heavily by following code [15] from refs [16].

REFERENCES

1. Y. LeCun, Y. Bengio, and G. Hinton, Nature 521, 436-444 (2015).
2. G. E. Karniadakis, Ioannis G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, Nature 3, 422-440 (2021).
3. M. Raissi, P. Perdikaris, and G. E. Karniadakis, J. Comput. Phys. **378**, 686-707 (2019).
4. C. Bajaj, L. McLennaan, T. Andeen, and A. Roy, Mach. Learn.: Sci. Technol. **4**, 015013 (2023).
5. M. R.-Behesht, C. Huber, K. Shukla, and G. E. Karniadakis, J. Geophys. Res. **127**, e2021JB023120 (2022).
6. S. Alkhadhr and M. Almekkawy, J. Sens. **23**, 2792 (2023).
7. N. Zobeiry and K. D. Humfeld, Eng. Appl. Artif. Intell. **101**, 104232 (2021).
8. X. Jin, S. Cai, and G. E. Karniadakis, J. Comput. Phys. **426**, 109951 (2021).
9. H. Elvazi, M. Tahani, P. Schatter, and R. Vinuesa, Phys. Fluids **34**, 075117 (2022).
10. P. Guofei, L. Lu, and G. E. Karniadakis, SIAM J. Sci. Comput. **41**, A2603-A2626 (2019).
11. J. Han, A. Jentzen, and Weinan E, "Solving high-dimensional partial differential equations using deep learning" in *Proceedings of the National Academy of Sciences of the United States of America* (2018), Vol. 115, pp. 8505-8510.
12. C. Song, T. Alkhalifah, and U. B. Waheed, Geophys. J. Int. **225**, 846-659 (2021).
13. O. Noakoasteen, S. Wang, Z. Peng, and C. Christodoulou, IEEE Trans. Antennas Propag. **1**, 9158400 (2020).
14. T. P. Agatho, (2023), available at https://github.com/Theodoret/PINN_1D_HeatTransfer_Euler, accessed on 11 May 2023.
15. A. Almqvist, "Physics-informed neural network solution of 2nd order ODE:s" (2023), available at https://www.mathworks.com/matlabcentral/fileexchange/96852-physics-informed-neural-network-solution-of-2nd-order-ode-s, MATLAB Central File Exchange, retrieved May 3, 2023.
16. A. Almqvist, Lubricants **9**, 82 (2021).