AN IMPROVED TRAINING METHOD FOR WAVELET NEURAL NETWORKS FOR SOLVING ORDINARY AND PARTIAL DIFFERENTIAL EQUATIONS

TAN LEE SEN

UNIVERSITI SAINS MALAYSIA

2022

AN IMPROVED TRAINING METHOD FOR WAVELET NEURAL NETWORKS FOR SOLVING ORDINARY AND PARTIAL DIFFERENTIAL EQUATIONS

by

TAN LEE SEN

Thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

October 2022

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Prof. Dr. Zarita binti Zainuddin, my main supervisor, for her excellent supervision, continuous support, and constant encouragement and motivation. Every piece of her advice in all aspects have helped me from the beginning to the completion of this thesis. I would also like to extend my gratitude to Associate Professor Dr. Ong Pauline, my field supervisor, for her valuable and constructive suggestions that have improved the quality of my research work. Also, I would like to acknowledge Dr. Mohd Shareduwan bin Mohd Kasihmuddin, my current supervisor, for providing continuous help.

Also, I would like to take this opportunity thank the staff and administration members of the School of Mathematical Sciences and related postgraduate departments of Universiti Sains Malaysia for providing help during my study. Furthermore, I would like to extend my heartfelt thanks to Universiti Sains Malaysia for supporting this research under the Research University Grant 1001/PMATHS/8011041 and Bridging Research Grant 304/PMATHS/6316079. Not to forget, I am very grateful to my parents and siblings, for their support and care throughout my study. Finally, I would like to thank my postgraduate friend, Ummi Nurmasyitah binti Hassan and all of those who have encouraged me during the execution of the research.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

A	given initial values of ODEs
a	power exponent
B	given boundary values of ODEs
b,c	training input domain
$b1$	bias term
C	sensory modality
D	number of dimensions of butterflies
$d,d1,d2$	dilation parameters
f	fragrance
$f1$	input function of hidden layers
G	given derivative initial value of ODEs
g	current best solution
$H(x,y)$	given function of ODEs and PDEs
$h_0(y),h_1(y),q_0(x),q_1(x)$	given boundary values of PDEs
I	stimulus intensity
i,r	random number between 0 and 1
J	number of butterflies in population
j,k,m,n	integer index
K	given number of PDEs
M,N	number of hidden nodes
$MaxI$	maximum number of iterations
NI	number of training points
NP	number at which MAE of run achieves acceptability criteria of MAE
NT	total number of independent runs
O	objective function
o	constant in dynamic switch probability
p	switch probability
pp	function of calculating exponential increment of switch probability

$s, s1, s2$	translation parameters
$\boldsymbol{s}, \boldsymbol{s1}$	translation position
t	current iteration number
$u(x), u(x,y)$	output of ANNs or WNNs
v	weights connecting input layer and hidden layer of ANNs
w	synaptic weights connecting hidden layer and output layer
X	butterflies' position vectors
$\bar{X}$	butterflies' positions values
x, y, z	spatial variables
$\boldsymbol{x}$	spatial position
$\hat{y}$	exact solution
$Z(x,y)$	function of first part of trial solutions of PDEs
Δw	step-size of weights
η	learning rate
μ	momentum
σ	outputs of second hidden layer of deep WNNs
ϕ	wavelet activation function connecting second hidden layer and first hidden layer of deep WNNs
φ	activation function of first hidden layer of ANNs or WNNs
φ', φ''	first and second order derivatives of wavelet activation functions
$\partial E / \partial w$	derivative of an error function with respect to weight parameters

LIST OF ABBREVIATIONS

AE	Absolute Error
ANNs	Artificial Neural Networks
BENNs	Bernstein Neural Networks
BOA	Butterfly Optimization Algorithm
BP	Backpropagation
BVPs	Boundary Value Problems
CHNNs	Chebyshev Neural Networks
CNNs	Cosine Neural Networks
DEs	Differential Equations
ELM	Extreme Learning Machine
FDEs	Fractional Differential Equations
GA	Genetic Algorithm
GW	Gaussian Wavelet
IBOA	Improved Butterfly Optimization Algorithm
IPA	Interior Point Algorithm
IVPs	Initial Value Problems
LENNs	Legendre Neural Networks
MAE	Mean Absolute Error
MH	Mexican Hat
MIN	Minimum
MAX	Maximum
MSE	Mean Square Error
MR	Morlet
MBP	Momentum Backpropagation
MLPs	Multilayer Perceptrons
ODEs	Ordinary Differential Equations
PDEs	Partial Differential Equations
PSO	Particle Swarm Optimization

PSNNs	Power Series Neural Networks
RBFNs	Radial Basis Function Networks
SOS	Symbiosis Organisms Search
SQP	Sequential Quadratic Programming
STD	Standard Deviation
WNNIBOA	Wavelet Neural Networks with Improved Butterfly Optimization Algorithm
WNNBOA	Wavelet Neural Networks with Butterfly Optimization Algorithm
WNNPSO	Wavelet Neural Networks with Particle Swarm Optimization
WNNMBP	Wavelet Neural Networks with Momentum Backpropagation
WNNs	Wavelet Neural Networks

LIST OF APPENDICES

SUATU KAEDAH LATIHAN YANG DITAMBAHBAIK UNTUK RANGKAIAN NEURAL WAVELET BAGI MENYELESAIKAN PERSAMAAN PEMBEZAAN BIASA DAN SEPARA

ABSTRAK

Persamaan pembezaan telah digunakan secara meluas untuk pemodelan kajian dan aplikasi yang tidak terkira banyaknya dalam pelbagai disiplin. Beberapa tahun kebelakangan ini, rangkaian neural buatan (RNB) telah mendapat perhatian untuk menghampirkan penyelesaian persamaan pembezaan dengan suatu keadah yang lebih cekap. Di dalam tesis ini, rangkaian neural wavelet (RNW) digunakan dan diprogramkan untuk menyelesaikan persamaan pembezaan biasa (PPB) dan persamaan pembezaan separa (PPS). Kegunaan RNW muncul oleh kerana ia mempunyai struktur yang lebih padat dan kelajuan pembelajaran yang lebih pantas sementara mengekalkan sifat penghampiran universal konvensional RNB. Meskipun RNW mempunyai sifat yang superior, algoritma pembelajarannya mestilah ditangani secara bijaksana memandangkan algoritma pembelajaran yang digunakan untuk menyesuaikan parameter pemberat bagi RNW boleh mempengaruhi kualiti ketepatan dan penumpuan RNW. Kebelakangan ini, algoritma pengoptimuman rama-rama (APR), suatu algoritma metaheuristik, telah menunjukkan keputusan yang efisien dalam masalah pengoptimuman global dan pelbagai aplikasi. Namun begitu, APR mempunyai masalah penumpuan pramatang di optima tempatan. Dengan demikian, suatu penambahbaikan algoritma pembelajaran yang berdasarkan metaheuristik, iaitu algoritma pengoptimuman rama-rama ditambah baik (APRP), telah dicadangkan dan kemudian digunakan untuk mempertingkatkan proses pembelajaran RNW. Selain itu, eksperimen berangka dalam penyelesaian persamaan pembezaan yang sama dijalankan dengan menggunakan tiga algoritma pembelajaran yang lain, iaitu algorithm APR, pengoptimuman kerumunan zarah (PKZ) dan rambatan balik momentum (RBM) dalam RNW. Analisis statistik berdasarkan keputusan daripada bilangan pengulangan yang tidak bersandar dan mencukupi dijalankan untuk mengesahkan keberkesanan kaedah yang dicadangkan dari segi ketepatan, keteguhan dan penumpuan. Perbandingan prestasi dengan kaedah-kaedah lain termasuk RNW, RNB dan kaedah berangka mengesahkan dengan jelas ketangkasan kaedah RNW dengan algoritma pengoptimuman rama-rama ditambah baik (RNWAPRP) yang dicadangkan. Kemudian, RNW dengan ilmu pengetahuan yang dikodkan dalam suatu set pemberat berangka telah diuji dengan beberapa titik ujian untuk suatu penilaian mendalam bagi prestasi model RNW yang telah dilatih. Selain itu, prestasi RNW dalam penyelesaian PPB telah diperiksa dengan pelbagai fungsi pengaktifan wavelet, iaitu, Mexican Hat (MH), Wavelet Gaussian (WG) dan Morlet (MR). Antara fungsi pengaktifan wavelet, WG merupakan satu fungsi pengaktifan wavelet yang berpotensi bagi RNWAPRP dalam penyelesaian PPB. Kemudiannya, suatu penghampiran RNW dalam bagi penyelesaian PPS yang sama telah dicadangkan di dalam tesis ini. Keputusan statistik menunjukkan bahawa prestasi model RNW yang mempunyai lapisan tersembunyi tunggal adalah lebih superior daripada RNW dalam.

AN IMPROVED TRAINING METHOD FOR WAVELET NEURAL NETWORKS FOR SOLVING ORDINARY AND PARTIAL DIFFERENTIAL EQUATIONS

ABSTRACT

Differential equations (DEs) have been widely used for modelling countless studies and applications in various disciplines. In recent years, artificial neural networks (ANNs) have gained attention in solving DEs in a more efficient way to approximate solutions of DEs. In this thesis, wavelet neural networks (WNNs) were employed and programmed to solve ordinary differential equations (ODEs) and partial differential equations (PDEs). The utilization of the WNNs emerges from having more compact structure and faster learning speed while retaining the universal approximation property of the conventional ANNs. Despite the superior properties of the WNNs, its training algorithm must be judiciously addressed since the training algorithm which is employed for adjusting weight parameters of WNNs can consequently affect the quality of the accuracy and convergence of the WNNs. Recently, butterfly optimization algorithm (BOA) which is a metaheuristic algorithm, has shown efficient results to global optimization problems and various applications. However, the BOA has a problem of premature convergence at local optima. Thus, an improved metaheuristic-based training algorithm, namely, improved butterfly optimization algorithm (IBOA), was proposed and subsequently used to enhance the training process of the WNNs. Also, numerical experiments on solving the same DEs were conducted using three other training algorithms, namely, BOA, particle swarm optimization (PSO) and momentum backpropagation (MBP) in the WNNs. Statistical analyses of the results based on a sufficient number of independent runs were conducted to validate the effectiveness of the proposed methods in terms of the accuracy, robustness and convergence. Performance comparison with other methods including WNNs, ANNs and numerical methods clearly verified the ascendency of the proposed wavelet neural networks with improved butterfly optimization algorithm (WNNIBOA) methods. Then, the WNNs with the encoded knowledge in a set of numeric weights were tested with several testing points for an in-depth evaluation of the performance of the trained WNNs models. Also, the performance of the WNNs in solving ODEs was examined by varying the wavelet activation functions, namely, Mexican Hat (MH), Gaussian wavelet (GW) and Morlet (MR). Among the wavelet activation functions, the Gaussian wavelet (GW) is a potential wavelet activation function for the WNNIBOA in solving ODEs. Subsequently, a deep WNNs approximation for the solutions of the same PDEs was proposed in this thesis. Statistical results showed that the performance of the WNNs with single hidden layer model was superior to that of the deep WNNs.

CHAPTER 1

INTRODUCTION

1.1 Introduction

Differential equations (DEs) are a powerful mathematical language which give us a mathematical way of understanding of how functions relate to its derivatives in a form of equations. They have been extensively used to formulate and model many ranges of problems and applications such as fundamental theories of physics, engineering problems, chemical reactions, and biological processes. For instance, Bernoulli equation, Schrödinger equation and Newton's second law in the fundamental theories of physics are expressed by means of the DEs (Panda and Pani, 2018). On the other hand, the susceptible-infectious-removed model, which is a biological mathematical model is formulated by DEs to predict the spreading rate of specific epidemics.

Solutions to DEs are vital to a wide range of investigations in various aspects of study. By solving DEs, useful outcomes are obtained, i.e., solving the seismic model can yield cost-effective wave propagation information, and as such, geophysicists can save the cost for drilling (Liao, 2011). Some simple DEs can be solved by analytic methods. However, some DEs which do not have exact solutions are approximately solved using mesh-based traditional numerical methods such as finite difference and finite element methods. These classical methods which approximate solutions at mesh grids are found to be computationally expensive when high dimensional DEs are considered. Additionally, an additional interpolation procedure is needed to yield testing solutions which have not been considered during computation because the numerical solutions are only valid in the considered grid of points (Mall and Chakraverty, 2014).

In response to the importance of the DEs in many disciplines, the construction of efficient and reliable numerical methods has become an ongoing research topic. Recently, machine learning algorithm has opened up a new attractive category of numerical methods. This algorithm can learn to adjust its parameters in response to problems which can reap great rewards by making solving DEs closely automated. Examples of machine learning algorithms that have been developed for solving DEs are support vector machines and artificial neural networks (ANNs). Particularly, ANNs for solving DEs have attracted much attention (Mall and Chakraverty, 2017). Mathematically speaking, ANNs are an intelligent mathematical model mimicking the ability of information processing of the human brain. Its structure comprises of interconnected nodes in which each connection carries a numeric weight. An ANN will adjust its numeric weights through a training method so that it can learn to solve problems. When the nodes are linked with optimal weights, the trained networks are able to solve the problems.

In the late years of 20th century, the ANNs for solving DEs were reported. In recent years, many researchers have switched to more computationally efficient ANNs alternatives for solving DEs. To solve DEs using the ANNs, the problem of solving DEs is usually formulated as an optimization problem, aiming at minimizing an unsupervised error function. Given a set of training points to the network, the ANNs model with only a few hidden nodes in the single hidden layer is able to approximate solutions of DEs. Unlike the numerical methods, the computational complexity of the ANNs is relatively simple and does not increase quickly when the number of training points increases (Mall and Chakraverty, 2014). Due to its advantageous properties such as ease of implementation, adaptability, universal approximator and generalization capability, the application of the ANNs in solving DEs deserves to be further explored as powerful alternative methods in solving different kinds of DEs.

1.2 Motivations of Study

In recent years, various ANNs models in solving DEs have been proposed and seen an exponential growth. Among different kinds of ANNs models, multilayer perceptrons (MLPs) model has been frequently preferred in the literature to solve DEs such as ordinary differential equations (ODEs), partial differential equations (PDEs) and fractional differential equations (FDEs). Nevertheless, slow training problems and local optima stagnation are the common issues of the MLPs (Zainuddin and Ong, 2012; Zainuddin and Pauline, 2011). These weaknesses have led to the emergence of new ANNs models. As a result, a great deal of works using other ANNs models for solving DEs have been introduced (Mall and Chakraverty, 2014; Mall and Chakraverty, 2016; Rizaner and Rizaner, 2018). Despite having promising results in previous studies, the approximation accuracy of the ANNs algorithms can be further improved.

One of the outstanding ANNs models originally conceived by Zhang and Benveniste (1992) is wavelet neural networks (WNNs), combining wavelets and ANNs. The specialty of the WNNs model is on the processing elements in its hidden layers which include wavelet activation functions, translation, and dilation parameters. These elements make the WNNs a more compact topology compared with the other ANNs approaches and a fast training speed (Ong and Zainuddin, 2019). Besides, universal approximation potential of the ANNs is preserved in the WNNs. In fact, before the introduction of WNNs, wavelets have been used for solving DEs. The compact support property of a wavelet is a useful feature for the approximation of functions and differential operators which has been proven by Esteban-Bravo and

Vidal-Sanz (2007). Additionally, the WNNs model has been successfully used for solving various problems such as classification, prediction, and solving PDEs (Ong *et al.*, 2018; Zhang *et al.*, 2016; Zainuddin and Pauline, 2011; Ong and Zainuddin, 2019; Li *et al.*, 2013). The great success of the WNNs in solving various applications and its beneficial advantages have motivated the investigation of its capability in solving ODEs and PDEs. Moreover, not much in-depth research on solving DEs using WNNs has been conducted. Yet, the benefits of the WNNs with the commonly used wavelet activation functions, namely, Mexican Hat (MH), Gaussian wavelet (GW) and Morlet (MR) in solving DEs have not been explored.

Recently, various deep ANNs models with more than one hidden layer have been proposed in tackling various types of challenging applications, with promising results reported. These promising results have motivated the employment of more hidden layers in the WNNs which can be an advantageous model for solving PDEs.

When it comes to the training method aspect, some ANNs models in the previous studies were impaired by the adopted training methods, jeopardizing the approximation capability of the ANNs. Similarly, the approximation capability of the WNNs depends heavily on the adopted training algorithm because the underlying logic for using WNNs is to find optimal weights for its outputs. A good training method gives the WNNs a good quality set of optimal weights which can consequently yield a WNNs model with an excellent approximation capability while a poor training method makes the WNNs less efficient. Therefore, a WNNs model with a good training method is essential to educate the WNNs for providing more accurate DEs solutions.

1.3 Problem Statements

Since the approximation accuracy of the ANNs algorithms in the previous studies can be further improved, new intelligent computational techniques based on the WNNs are developed to solve ODEs and PDEs. However, given the superiority of the WNNs, it is necessary to set correct operation of a WNN, in terms of its training method, initialization of the translation and dilation parameters, network architecture, and activation functions in the hidden layers (Zainuddin and Pauline, 2011).

Traditionally, the numeric weights of WNNs are computed with backpropagation (BP) algorithm, which adjusts the network parameters based on the gradient information of an error function (Cao *et al.*, 2010). The BP algorithm is generally useful; however, it might get trapped into local minima, and its speed of learning process is slow. In recent years, to address these computational shortcomings of the BP algorithm, the metaheuristic algorithm has captured attention to improve efficiency in the WNNs (Zhang *et al.*, 2016; Yang *et al.*, 2018; Chitsaz *et al.*, 2015). Unlike the BP algorithm, the metaheuristic algorithm is a gradient-free algorithm which can increase the flexibility of the WNNs model. Therefore, in this thesis, improving the performance of the WNNs by emphasizing the training method is the main concern. This enhancement is accomplished by integrating a metaheuristic-based method in adjusting the weight parameters of the WNNs.

Recently, a novel metaheuristic algorithm, namely, the butterfly optimization algorithm (BOA) which is inspired by the food foraging behavior of butterflies has been proposed (Arora and Singh, 2019). Due to the simplicity of the BOA, it has been applied to various kinds of problems. Also, the BOA has been adopted for training ANNs (Jalali *et al.*, 2019). Based on their simulation results in terms of accuracy, the BOA performed better than the well-regarded metaheuristic training algorithms, including grasshopper optimization algorithm, flower pollination algorithm, genetic algorithm (GA), particle swarm optimization (PSO) and differential evolution. However, the main drawback of the BOA is the premature convergence due to its weak exploitation capability (Arora *et al.*, 2018).

To improve the exploitation capability of the BOA, a new improvement of the BOA, namely, the improved butterfly optimization algorithm (IBOA) is made in this thesis. The proposed IBOA training method is able to avoid the local optima and premature convergence problems which enable it to find a good quality set of weight parameters for the WNNs. As a result, the intelligent computational tool combining the strengths of the WNNs and the IBOA training method can achieve higher approximation ability, and thus, it is expected to improve the accuracy of the solutions of DEs over the existing ANNs and numerical methods.

Apart from investigating the training method in the WNNs, the types of wavelet activation functions and the number of hidden layers are also investigated in this thesis.

1.4 Research Objectives

The objectives of this thesis are:

1. To develop a new training method, namely, IBOA for enhancing the training process of novel WNNs methods in solving ODEs and PDEs.
2. To verify the effectiveness of the wavelet neural networks with improved butterfly optimization algorithm (WNNIBOA) in solving ODEs and PDEs by comparing with the WNNs trained with the other training methods, namely, the momentum backpropagation (MBP), PSO and BOA, other ANNs and numerical methods.

3. To investigate the performance of the WNNs in solving ODEs with three different wavelet activation functions, namely, the MH, GW and MR.
4. To propose a deep WNNs model for solving PDEs.

1.5 Methodology

To achieve objective 1, new WNNs models with a single hidden layer are developed and formulated to solve the first and second order linear and nonlinear ODEs, as well as elliptic PDEs. Moreover, three modifications are incorporated into the proposed IBOA by incorporating dynamic switch probability and modifying the sensory modality parameter and local search update equation to improve the problem of premature convergence and slow convergence that are suffered by the BOA.

For objective 2, the performance comparison between the WNNIBOA and the WNNs trained with the other training methods is made under the same WNN architectures and stopping criteria. The performance of the WNNs models in solving DEs is evaluated using two performance metrics, namely, Absolute Error (AE) and Mean Absolute Error (MAE). To achieve objectives 3 and 4, the same ODE examples in Chapter 4 and the same PDE examples in Chapter 5 are considered, respectively.

1.6 Scope of Thesis

This study will be confined in this section to fulfil the aforementioned objectives.

- Three WNNs architectures, two single WNNs models with one hidden layer, one for solving only first and the second order ODEs and the other for solving only two dimensional elliptic PDEs with the Dirichlet boundary conditions, and one deep WNNs model with two hidden layers for solving the same PDEs, are proposed.
- Only the standard BOA, PSO and MBP training algorithms are incorporated in

the WNNs for solving the ODEs and PDEs to validate the effectiveness of the proposed WNNIBOA in terms of the accuracy, robustness and convergence.

- The performance of the deep WNNs with only two hidden layer is compared with that of the WNNIBOA with one hidden layer in solving the same PDEs.
- Only an incremental pattern which increases the switch probability value from a minimum to a maximum in the proposed IBOA is investigated.

1.7 Thesis Organization

This chapter discussed a general introduction to DEs and its importance. Subsequently, the evolvement of ANNs in solving DEs was briefly introduced. The motivations of study, problem statements, research objectives, and scope of thesis were given. The outlines of the other six chapters are given as follows.

In Chapter 2, the architecture of an ANN and different training methods for training ANNs models are introduced. Afterwards, the steps of integration of ANNs in solving DEs are explained. Next, the previously proposed ANNs methods made on solving DEs are reviewed. Due to the importance of training methods, this chapter includes a comprehensive review of the existing training methods of ANNs in solving DEs.

Chapter 3 starts with the highlight of the architecture of a WNN and the development of the training methods of WNNs. Besides, the applications of WNNs are discussed. This chapter is followed by reviewing the BOA, including its advantages, limitations, applications, and BOA variants. Then, the proposed IBOA is introduced. Lastly, a section is spent on presenting how the proposed WNNIBOA is integrated in solving DEs.

After presenting the detailed description on the proposed IBOA, the ensemble proposed WNNIBOA is used for solving ODEs in Chapter 4. The mathematical formulations for solving ODEs are described in detail. For numerical simulations, linear and nonlinear ODEs taken from literature are considered and the obtained simulation results are compared with those of the WNNs trained with BOA, PSO and MBP, existing ANNs and numerical methods. At the end of this chapter, the statistical analysis of the MAE values obtained by the WNNs methods is conducted.

Meanwhile, in Chapter 5, the WNNs trained with the four different training methods are employed for solving PDEs. The results of the WNNs are compared and statistically discussed. On the other hand, Chapter 6 proposes a deep WNNs model for solving the PDEs. The numerical results of the deep WNNs are compared with those obtained from Chapter 5. Finally, this thesis comes to a close with conclusions and future research ideas in Chapter 7.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The applications of ANNs scale across different domain of studies. In this thesis, the application of ANNs is mainly focused on solving DEs. In this chapter, an introduction to ANNs and its training algorithms will be presented. The chapter is followed by steps on how ANNs are integrated in solving DEs along with the advantages of using ANNs methods. Next, literature review of ANNs methods in solving DEs will be discussed in detail. Due to the importance of training methods, different training algorithms which have been used for training the ANNs in solving DEs will be reviewed.

2.2 Artificial Neural Networks

The word ANNs is on everyone’s lips. It is a machine learning algorithm which mimics intelligent from biological neural networks and provides learning ability for solving different problems. From the mathematical point of view, the ANN is a mathematical model in which its outputs are expressed as a mathematical function in terms of the adjustable numeric weight parameters. Generally, these weight parameters of an untrained ANN are initialized randomly and then optimized through learning to give optimal outputs to a particular problem.

2.2.1 Architecture of an ANN

The most commonly used ANNs model is MLPs, which are generally composed of three layers, namely, input layer, hidden layer, and output layer, with each layer fully connected to the adjacent layer. Figure 2.1 depicts the architecture of an ANN.

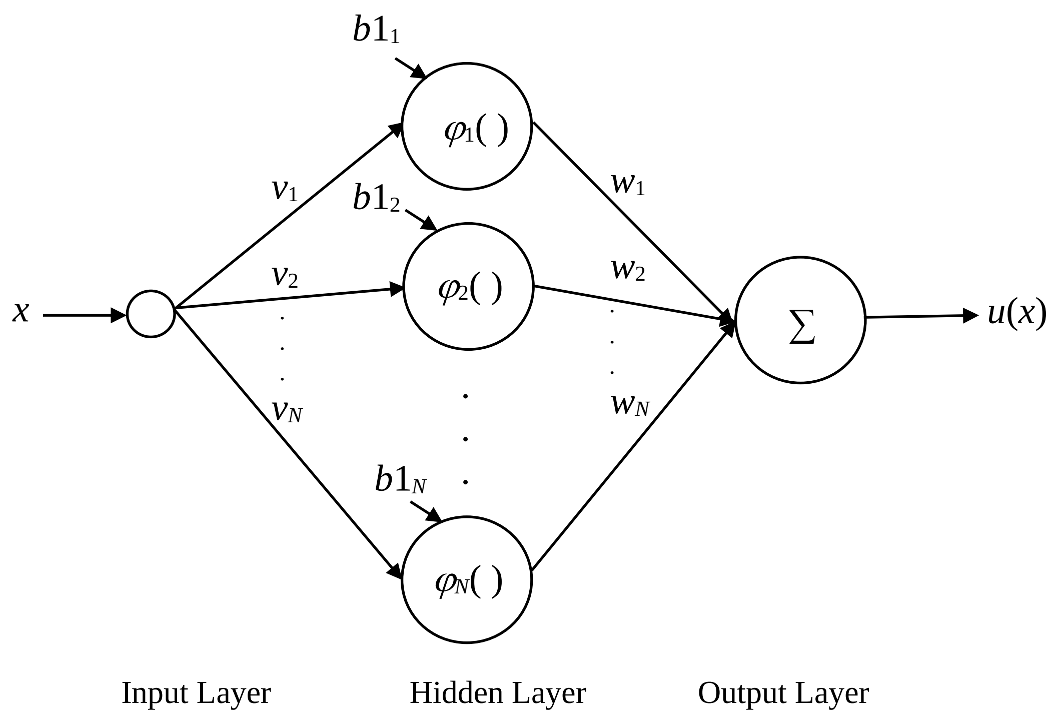

Figure 2.1 Architecture of an ANN with input x and output u. This architecture has one input layer with one input node, one hidden layer with N hidden nodes and one output layer. These nodes are linked by weight values v_n and w_n, for $n = 1, \ldots, N$. The activation functions φ_n and bias term $b1_n$ are only used in the hidden nodes.

The input node in the input layer receives the input variables $x = [x_1, x_2, ..., x_{NI}]$, where NI is the number of training points. In the hidden layer, the accepted input variables are multiplied with the weights between the input layer and the hidden layer v. This weighted sum of inputs with addition of bias $b1$ that pointed to each hidden node will be used as the inputs for the activation functions φ in the hidden nodes. Normally, a log-sigmoid function is used as the activation function in the hidden layer to limit the amplitude of the outputs of the hidden nodes within the range of the log-sigmoid function (Ojha *et al.*, 2017). Subsequently, the outputs of the hidden nodes, which are the outputs of the activation functions, will be sent to the adjacent output layer to be multiplied with the weights between the hidden layer and the output layer w. The output node in the output layer gives outputs, where the mathematical formulation defining the output of the ANN is as follows (Malek and Beidokhti, 2006):

$$u(x)=\sum_{n=1}^{N} w_n \varphi_n (v_n x + b1_n), \tag{2.1}$$

where $b1_n$ is the bias, x is the input, v_n is the weights connecting the input layer and the hidden layer, φ_n is the activation function, w_n is the weights connecting the hidden layer and the output layer, and N is the number of the hidden nodes.

In Equation (2.1), the weight and bias parameters are adjustable through training to epitomize knowledge that can fit to solve problems. Generally, the training paradigm can be classified into supervised training, unsupervised training, and reinforcement training (Ojha *et al.*, 2017). In the supervised training, inputs with target outputs are required. An ANN is trained by comparing the outputs of the ANN with the target outputs and then the weight parameters are adjusted to minimize the differences between them. Some examples of problems based on the supervised training are classification and prediction problems. On the other hand, the unsupervised training consists of only inputs without target outputs. In the case of solving DEs, the target outputs are unknown in prior. Therefore, the problem of solving DEs falls into the category of unsupervised training. In the reinforcement training, the training of ANNs deals with sets of input, output, and grade.

2.2.2 Training Algorithms

Since determining the weights of ANNs is considered as an optimization problem, almost any general-purpose optimization methods are applicable for training ANNs. However, as the fact of the influence of the weights of ANNs on the performance of ANNs, various research efforts have been focused on the training methods of ANNs. In Yang (2010), there are three types of training or optimization algorithms, namely, deterministic algorithms, stochastic algorithms and hybrid algorithms. Figure 2.2 depicts the training algorithms of ANNs.

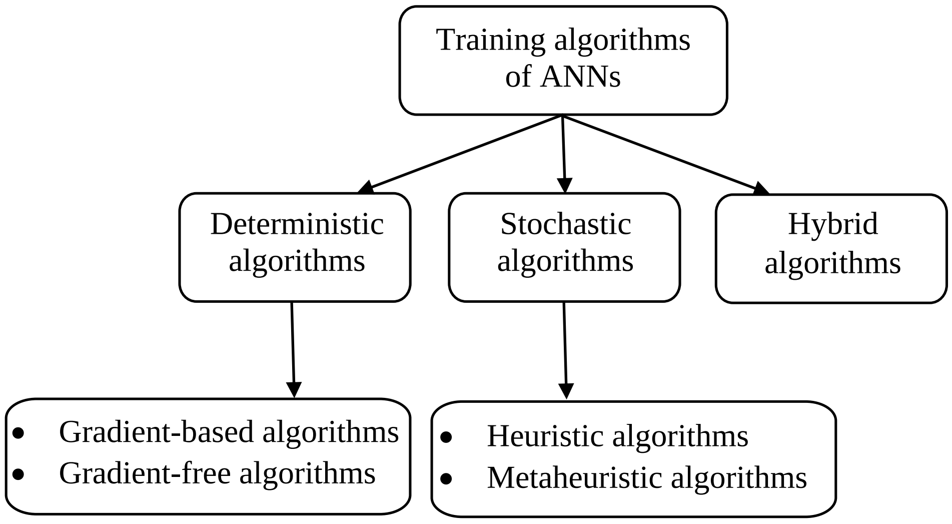
Training algorithms of ANNs
Deterministic algorithms
Stochastic algorithms
Hybrid algorithms
• Gradient-based algorithms
• Gradient-free algorithms
• Heuristic algorithms
• Metaheuristic algorithms

Figure 2.2 Training algorithms of ANNs

2.2.2(a) Deterministic Algorithms

Deterministic algorithms are not based on probabilistic perspectives and can be categorized in two parts, namely, gradient-based algorithms and gradient-free algorithms.

Gradient-based algorithms are performed based on gradient information. BP is an example of gradient-based algorithms which is traditionally used for training ANNs. To minimize network error, the weight parameters are iteratively adjusted as follows:

$$w(t+1) = w(t) + \Delta w(t), \tag{2.2}$$

$$\Delta w(t) = -\eta \frac{\partial E}{\partial w(t)}, \tag{2.3}$$

where η is the learning rate, t is the number of iterations and $\partial E/\partial w$ is the derivative of an error function with respect to weight parameters. Although the BP algorithm has been widely used for training ANNs, this algorithm is prone to slow convergence and might converge to local optimal solutions. The convergence of an algorithm represents its behavior or rate moving towards the global optimum, which is the best solution with the smallest error when considering the entire search space (Mirjalili *et al.*, 2017). On the other hand, the local optimal solutions refer to near-optimal solutions that return a value close to the objective value of the global optimum. Due to the presence of a large number of local optimal solutions in the entire search space, an optimization method might mistakenly terminate the training at local optimal solutions.

To get rid of the shortcomings, a number of variations of the gradient-based algorithms has been developed. MBP is one of the improved versions of BP with incorporation of a momentum term in the standard BP. The momentum term can speed up the convergence rate and prevent the learning process from settling in a local minimum. In the MBP algorithm, the weight update includes the current weight change and the previous weight change, as shown in the following formula:

$$\Delta w(t) = -\eta \frac{\partial E}{\partial w(t)} + \mu \Delta w(t-1), \tag{2.4}$$

where η is the learning rate, μ is the momentum, t is the number of iterations and $\partial E/\partial w$ is the derivative of an error function with respect to weight parameters. Besides, Quasi–Newton and Levenberg–Marquardt methods are the variations of the gradient-based algorithms, which are frequently used for training ANNs (Ranković and Savić, 2011). On the other hand, sequential quadratic programming (SQP) and interior point algorithm (IPA) are examples of gradient-based algorithms used in the training of ANNs. They are powerful local optimization methods for solving constrained nonlinear optimization problems. Despite the usefulness of the existing gradient-based training algorithms for optimizing the weights of ANNs, one of the important issues on gradient-based algorithms is their dependency on gradient information.

In order to overcome the disadvantages of gradient-based algorithms, gradient-free algorithms are introduced by removing the need for computing derivatives. For example, Nelder–Mead method is a well-known gradient-free algorithm. Owing to its simplicity, Nelder–Mead method has been used for training ANNs with good results. However, Nelder–Mead method may converge slowly or may fail to converge to a global minimum and can significantly reduce its efficiency in training ANNs (Wilamowski and Pham, 2012). Another gradient-free algorithm is extreme learning machine (ELM). ELM is not only a gradient-free algorithm but also an iteration-free algorithm. In contrast to iterative-based and gradient-based training algorithms, ELM can offer feasible weights in a much faster and simpler way, by implementing Moore–penrose generalized inverse technique (Wang *et al.*, 2015). However, ELM has a requirement on the ANNs' architectures because ELM is specifically used for training fixed ANNs architectures where only the weights between the hidden layer and the output layer are adjusted.

2.2.2(b) Stochastic Algorithms

Besides deterministic algorithms, a class of algorithms known as stochastic algorithms has been attempted to increase the performance of ANNs. Stochastic algorithms are based on random mechanism to search for solutions with a better objective function value. According to Yang (2010), heuristic and metaheuristic algorithms are two general types of stochastic algorithms. However, no agreed definitions of heuristic and metaheuristic have been rendered in the literature and some of the literature have used both terms interchangeably.

Generally, metaheuristic algorithms are high-level heuristic optimization algorithms which typically are inspired by evolutionary and nature-inspired understanding from biology, natural phenomena, and social systems (Cuevas *et al.*, 2019; Sörensen, 2015). Over the past few decades, the metaheuristic algorithms have gained increasing popularity, due to their flexibility and ease of implementation. These advantages originate from the derivative-free nature of these algorithms which involve two search behaviors, namely, exploration and exploitation, to produce sufficiently good solutions for optimization problems. The exploitation search is used to find other solutions around the current good solution in a short distance while the exploration search is used to find other solutions in new areas in the search space. In addition, compared with gradient-based algorithms, metaheuristic algorithms are considered as more effective and simpler algorithms to optimize the weights of ANNs, because metaheuristic algorithms do not optimize each of the parameters of an ANN separately. Therefore, the calculation of the gradient information of an objective function with respect to each adjustable parameter is avoided. Specifically, updating weight parameters of an ANN using metaheuristic algorithms is done in a single vector containing all the weight parameters.

The metaheuristic algorithms can be categorized into single solution-based algorithms and population-based algorithms according to the number of candidate solutions processed in every iteration (Mirjalili *et al.*, 2017). The single solution-based algorithms contain only one candidate solution which is moved in the search space at each iteration for searching neighboring solution with a better quality. Due to a small number of function evaluations, this kind of algorithm requires low computation efforts. Some examples of algorithms in this category are simulated annealing and tabu search. As opposed to the single solution-based algorithms, the population-based algorithms initiate with a group of candidate solutions called population. From the context of biological ecosystem, the population is a group of one species living in the same habitat. Taking this knowledge into the population-based metaheuristic algorithms, the species represents a group of candidate solutions and the habitat represents the search space within a boundary area which corresponds to the boundary of the candidate solutions. In the context of training ANNs, each candidate solution in the population represents a set of weight parameters of ANNs. Therefore, there are many sets of weight parameters of ANNs in a run, where the best solution set is chosen according to the fitness value. Due to the multiple candidate solutions striking for the best solution, the overall efficiency and performance of the optimization can be improved. PSO and GA are well-known population-based algorithms.

However, no single metaheuristic algorithm can successfully solve all kinds of problems consistently. This fact has been proven by the theorem of No Free Lunch (Wolpert and Macready, 1997). Therefore, the modification of existing metaheuristic algorithms has been observed. Besides, researchers are continuously unearthing new metaheuristic algorithms. A survey on metaheuristic algorithms can be found in Ojha *et al.* (2017).

2.2.2(c) Hybrid Algorithms

Since there is no guarantee that optimal solutions can always be reached by metaheuristic algorithms that are implemented individually, hybrid algorithms have been considered to improve the performance of ANNs. For instance, a gradient-based algorithm can be applied to improve the solutions found by a metaheuristic algorithm. The hybrid algorithms will be further discussed in Section 2.3.2.

2.3 Solving DEs Using ANNs

In recent years, the ANNs have been devoted as an alternative to the numerical methods to solve DEs. The general idea of ANNs approach is to define the solutions of DEs in terms of the adjustable outputs of ANNs and then employs a training method to tune the adjustable weights in the output of ANNs. Eventually, the problem of solving DEs is formulated as an optimization problem, aiming at the minimization of an unsupervised error function.

An ANN starts with receiving a training input vector at the input nodes. Generally, the inputs can be presented to the ANN either through batch training or incremental training. In a batch training, all the inputs are presented to the network and the weights are updated only after all the inputs are received. In an incremental training, each input is presented to the network and the corresponding weights are updated. The number of weight updates will be equal to the number of input entries present in the training input vector. In this thesis, all inputs are presented to the network through the batch training because the error of solving DEs is taken over all the training points for adjusting the parameters.

Then, the output of the ANN can be generated using Equation (2.1) with initialized weight parameters. In this stage, the initialized weights are not able to allow for an error function to converge to zero. Therefore, these weights should be updated to reduce network error. To perform an error minimization, an unsupervised error function is designed by taking the difference between the left-hand side and the right-hand side of a DE since desired targets are supposed to be unknown in prior in the context of solving DEs. One of the aspects to consider when forming the error function is the associated initial and/or boundary conditions of a DE. Therefore, the minimization optimization problem of training ANNs is subjected to a set of constraints. Then, the error function is minimized by applying a training method to adjust the weight parameters.

Generally, when the residue of an error function ceases to zero, the DE can be considered as solved. In general, the employment of the ANNs approach towards solving DEs brings some attractive advantages (Lagaris *et al.*, 1998):

1. The ANN is simple in structure and easy to implement and to modify for solving various kinds of DEs.
2. The ANNs methods require low computational complexity and lesser training points as compared with the numerical methods. Unlike the traditional numerical methods, when the number of training points increases, the computational complexity of the ANNs methods does not increase quickly.
3. The ANNs methods generally provide differentiable solutions in a closed analytic form over the computational domain which are useful for further evaluation.

2.3.1 Existing ANNs Approach for Solving DEs

The building of any ANNs composes of two steps. The first step is to develop the ANN's architecture for representing the solutions of a given DE while the second step is to employ a training method for training the adjustable weight parameters in the ANN. It is clear that the overall performance of an ANN model is highly dependent on both the ANN's architecture and training method. In this subsection, the existing ANN approaches for solving DEs in the literature are reviewed, subsequently the training algorithms which have been used in ANNs for solving DEs will be discussed in next subsection.

Various strategies based on the ANNs have been investigated by many researchers. Generally, solving DEs using ANNs in the literature can be categorized into two approaches, namely, differentiable approach and integrable approach. Figure 2.3 depicts the ANNs approaches for solving DEs.

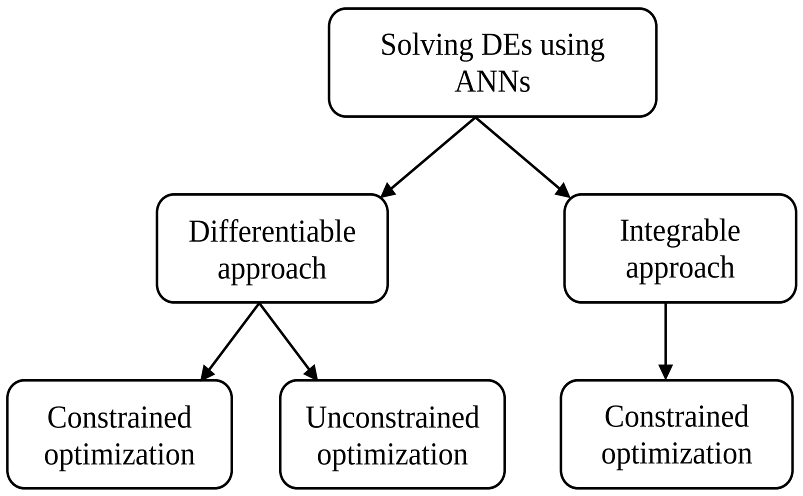

Figure 2.3 ANNs approaches for solving DEs

The differentiable approach is the most frequently used approach in which one uses a differentiable activation function in each hidden node in a network. Therefore, the solutions of DEs based on this approach correspond to differentiable and closed form output of a network. Then, the derivatives of the solutions can be derived through differentiation. In this approach, the network can be trained either through constrained optimization or unconstrained optimization, depending on how the associated conditions of DEs are satisfied. If one can satisfy the conditions of a DE, the optimization is called unconstrained optimization. Therefore, an objective function is formed by using the explicit form of the DE only because the associated conditions are exactly satisfied. For this purpose, a special consideration is required in formulating the approximate solutions for the automatic satisfaction of the associated conditions. The incorporation of trial solution by Lagaris *et al.* (1998) is an example of unconstrained optimization. On the other hand, the constrained optimization is the optimization in which the conditions of a DE are not satisfied exactly but approximately by adding a penalty term to an objective function. Although the constrained optimization is seemed to be easy to implement and able to save the amount of time for formulating the trial solutions, it can cause a low convergence rate and a low accuracy solution (Rudd and Ferrari, 2015). Accordingly, the unconstrained optimization provides a good trade-off in terms of the accuracy. Since this thesis is aiming for a better approximation accuracy, the unconstrained optimization is considered.

In contrast, the integrable approach uses an integrable activation function in each hidden node in a network to generate the output of a network which corresponds to the highest derivative appearing in DEs. Therefore, to obtain the solutions of DEs, the computation is done by integration. Towards this end, it is seen that the property of activation functions in the hidden layer of an ANN is an important feature in order to ensure that the solutions of DEs and its derivative functions are computable and fit the requirements based on the selected approach.

A substantial growth in the study of ANNs based on the differentiable approach in solving different types of DEs is observed. Lee and Kang (1990) firstly introduced a method to solve first order ODE using finite difference methods for the discretization of ODEs and the Hopfield neural network for minimizing error function. However, the emergence of the ANNs as a promising approximator in the context of DEs was first proposed by Lagaris *et al.* (1998). This earliest method is based on the differentiable approach. In this work, the authors proposed the MLPs as a DE's approximator and the trial solution as an approximate solution for the automatic satisfaction of the associated conditions. Using the trial solution, the original problem of solving DEs is reduced from a constrained optimization problem to an unconstrained optimization problem.

Since 1998, such ANNs method with the trial solution has gained increasing popularity and has been frequently used by other researchers. For instance, higher-order ODEs have been solved using ANNs (Malek and Beidokhti, 2006). In particular, the automatic satisfaction of the boundary conditions of the fourth-order ODE has been made using the trial solution. The same authors proposed ANNs with the trial solution for solving mixed BVPs of biharmonic PDEs (Beidokhti and Malek, 2009). In other works, the initial conditions of the first Painlevé equations and the boundary conditions of the Lane–Emden equations have automatically been satisfied using the trial solutions (Raja *et al.*, 2015b; Mall and Chakraverty, 2014; Mall and Chakraverty, 2016). Also, the ANNs approach with trial solution has been employed to solve Stokes problem (Baymani *et al.*, 2010).

However, the trial solution proposed by Lagaris *et al.* (1998) is feasible only for regular domains. For irregular domains, the ANNs have been proposed to solve two-dimensional and three-dimensional PDEs with irregular domain boundaries based on the combination of MLPs and radial basis function networks (RBFNs) in which the RBFNs are used to satisfy the boundary conditions (Lagaris *et al.*, 2000). A similar network combination has also been studied by Hoda and Nagla (2011) in solving the mixed BVPs of PDEs on irregular domains. Although the method proposed for the automatic satisfaction of the mixed boundary conditions on irregular domains is computationally expensive, the solutions of PDEs are fairly encouraging. Besides, the concept of length factor has been introduced in trial solution to satisfy the mixed BVPs conditions automatically for irregular domains (McFall and Mahan, 2009). The length factor is introduced to measure the perpendicular distance from a point to the boundary in order to return zero value on the boundary. Also, based on the length factor, the coupled systems of PDEs with discontinuities and the advection dispersion equations characterizing the mass balance of fluid flow in a chemical reactor have been solved using ANNs (McFall, 2013; Yadav *et al.*, 2018).

On the other hand, numerous ANNs schemes have been proposed to solve DEs based on the constrained differentiable approach. For instance, numerous literatures by Raja *et al.* for solving DEs problems including nonlinear Jeffery–Hamel flow systems (Raja and Samar, 2014a), two-dimensional nonlinear Bratu's problems (Raja and Samar, 2014b), nonlinear boundary value problems (BVPs) governed with pantograph functional differential equations (Raja, 2014), thin film flow of third grade fluids (Raja *et al.*, 2015a), nanotechnology problems based on multi-walled carbon nanotubes (Raja *et al.*, 2016a), nonlinear singularly perturbed BVPs (Raja *et al.*, 2018a), nonlinear Mathieu's systems (Raja *et al.*, 2018b) and Troesch's problem (Raja *et al.*, 2018c) have been observed. These ANNs approaches have shown a promising alternative numerical method in solving DEs. In other works, the hyperbolic and parabolic PDEs have been solved by Rudd and Ferrari (2015) using a constrained integration with ANNs approach. Moreover, a number of ANNs techniques based on the differentiable approach for the solutions to problems governed by PDEs have been developed, including the direct current motor, and a ball and beam system (He *et al.*, 2000), the one-dimensional Kuramoto–Sivashinsky and two-dimensional Navier–Stoke (Smaoui and Al-Enezi, 2004), Burger's equations (Hayati and Karami, 2007), and nonlinear Schrodinger equations (Shirvany *et al.*, 2009). For some complex PDEs, the PDEs of the system model were reduced to a system of nonlinear ODEs and this system was solved by ANNs (Mehmood *et al.*, 2018).

Apart from solving ODEs and PDEs, the applicability of ANNs approach has also been extended to solve FDEs (Raja *et al.*, 2017; Raja *et al.*, 2015c; Lodhi *et al.*, 2019; Raja *et al.*, 2010; Pakdaman *et al.*, 2017; Zúñiga-Aguilar *et al.*, 2017; Jafarian *et al.*, 2018). These methods either use constrained optimization or unconstrained optimization. To solve FDEs by using the ANNs methods, a fractional derivative of activation functions is needed (Raja *et al.*, 2017). For instance, the Maclaurin series expansion of log-sigmoid function and fractional derivatives of exponential functions involving Mittag–Leffler function were used as activation functions in the ANNs (Jafarian *et al.*, 2018; Raja *et al.*, 2010; Raja *et al.*, 2015c; Raja *et al.*, 2017; Lodhi *et al.*, 2019). On the other hand, the ANNs techniques have been applied by Effati and Pakdaman (2010) to solve fuzzy differential equations. In this work, the fuzzy differential equations were replaced by a system of ODEs.

Most of the aforementioned works use the MLPs model. Although the MLPs model has reported significant as the DEs approximator, this model is subject to local minima and slow learning problems (Zainuddin and Ong, 2012; Zainuddin and Pauline, 2011). Additionally, the MLPs model has a lot of weight parameters, i.e., the weights between the input layer and the hidden layer, as well as between the hidden layer and the output layer, to be trained which require considerable computational cost.

Besides the MLPs, other ANNs models, including Legendre neural networks (LENNs), Chebyshev neural networks (CHNNs), and Bernstein neural networks (BENNs) have been proposed to solve DEs. The LENNs, CHNNs and BENNs are a kind of single-layer functional link ANN where its hidden layer is replaced by a functional expansion block based on orthogonal polynomials. Removing the hidden layer and expanding the input pattern by the Legendre, Chebyshev, and Bernstein polynomials in the LENNs, CHNNs and BENNs, respectively, lower computational complexity and a much lesser number of adjustable parameters are achieved as compared with the MLPs. In the literature, Mall and Chakraverty (2016) has developed the LENNs for solving Lane–Emden equations. This ANNs model is capable of producing more accurate solutions with less computational efforts as compared with the MLPs. In another work, the same author has proposed a similar type of ANNs but based