Ibn Al Haitham Journal for Pure and Applied Science

Journal homepage: http://jih.uobaghdad.edu.iq/index.php/j/index

Cascade-Forward Neural Network for Volterra Integral Equation Solution

Shymaa Akram Hantoush Alrubaie
Middle Technical University, Continuous Education Centre, Baghdad-Iraq
Shymmaakram35@mtu.edu.iq

Article history: Received 13 December 2020, Accepted 31 January 2021, Published in 2021.

Doi: 10.30526/34.3.2683

Abstract

The method of solving volterra integral equation by using numerical solution is a simple operation but to require many memory space to compute and save the operation. The importance of this equation appeares new direction to solve the equation by using new methods to avoid obstacles. One of these methods employ neural network for obtaining the solution.

This paper presents a proposed method by using cascade-forward neural network to simulate volterra integral equations solutions. This method depends on training cascade-forward neural network by inputs which represent the mean of volterra integral equations solutions, the target of cascade-forward neural network is to get the desired output of this network. Cascade-forward neural network is trained multi times to obtain the desired output, the training of cascade-forward neural network model terminal when there is no enhancement in result. The model combines all training cascade-forward neural network to obtain the best result. This method proved its successful in training and testing cascade-forward neural network for obtaining the desired output of numerical solution of volterra integral equation for multi intervals. Cascade-forward neural network model measured by calculating MSE to compute the degree of error at each training time.

Keywords: Volterra Integral Equation (VIE), Cascade-Forward Neural Network (CFNN), Artificial Neural Network (ANN).

1. Introduction

Volterra Integral Equation (VIE) represents as a special state in mathematics science and various methods are introduced to solve it, base of this equation depends on it a model of several science field such as in physics, engineering, biology etc. VIE solved by multi numerical methods which are based as a linear or nonlinear converting system of integral equation that have direct or iterative methods solutions. Many different techniques have been presented to solve VIEs of the second kind by adopting the linear property of it. The base

technique of integral equation field depends on numerical quadrature rules, while the trapezoidal rule is the most popular technique to solve numerical integration which gives authenticated solutions [1]. The development in parallel processing affect all fields of knowledge, one of them is mathematics science methods, there are multi parallel techniques employed to take the typical solutions, examples of these are: genetic learning systems, artificial neural network, simulated annealing systems, associative memories, and fuzzy learning systems [2].

The recent papers that are interested of solving the integral equations have been employed some models of artificial neural network (ANN) to take the desired solutions. The comparison between ANN and numerical methods as techniques used to solve VIE appears many features of ANN, the solution by ANN models is characterized as different and continuous [3]. The structure of (ANN) inspired from information processing by getting the form of directed-graph structures. Simple mathematical operations managed in nodes to be processed, while the numerical weights with the links between nodes symbolized to be associated within them to create information. The nodes in one layer connected to nodes in next layer to flow the information through the structure and patterned by using activated function to get the outputs of nervous systems [2]. Cascade Forward Neural Network (CFNN) is a model inspired from Feed Forward Neural Network (FFNN) model. There is a major distinction of CFNN from other FFNN; CFNN contains direct connection between the input layer and the output, in addition to the indirect connection through the hidden layer, which means "in CFNN each neuron in the input layer is attached to each neuron in the hidden layer and each neuron in the output layer" [3]. These connections of CFNN are useful by its adapt nonlinear relationship of input and output and not eliminating the linear relationship between them that depended on the background, and this feature is not available in FFNN that contains only nonlinear relationship of input and output, by this feature CFNN conforms to be suitable to solve problems that requires to build prediction time series data that depends on the probable state with a high level of accuracy [4].

. In this paper, CFNN model focuses on simulation procedure, it is used to predict the time series data of VIE from both the generated data and the real data. The organization of this paper contains six component: literature review in section 2, Theoretical in section 3 & 4, & methodology, performance measure & experimental results in section 5, and conclusions in section 6.

2. Literature Review

Several methods were used to solve VIE and extract the best results, where for each method features which differs from the rest of the methods but the evolution in the use of parallel processors showed new methods to solve the equation. Tahmasbi [5] utilize the power series method to solve linear VIE of the second kind, the method used the exact solution of Taylor expansion of the integral equation computes with approximate solution acceptable, the method proved its effectiveness and appropriately compared with other methods. Isaacson et al. [6] develop collocation methods to solve linear, scalar, VIE of the second kind by employee partitioned quadrature depending on the qualocation framework, with smooth kernels containing sharp gradients and many examples were examined, the method appears the efficient of getting the numerical solution of equation. Mirzaee [7] presents the adaptive simpson's quadrature method for solving linear VIE of the second kind, its considered as simple method and has proved its efficiency and accuracy in obtaining results. Rahman et

al. [8] use the Galerkin weighted residual method to solve a VIE of first and second kind by using a very few "Laguerre polynomial", the approximate results of tested examples present the convergence from the numerical solutions. Kolk et al. [9] utilize suitable smoothing methods and polynomial splines on mildly graded or uniform grids to solve linear VIE of the second kind, the method obtained the appropriate numerical results. Aigo [1] applied the quadrature method to solve linear integral equations of the second kind by utilizing simpson's and trapezoidal quadrature rule to get accurate solution, this method achieved a good degree of accuracy according to obtained results of the illustrative examples. Costarelli et al. [10] develop numerical collocation method depending on approximate of the exact solution of superposition of sigmoidal functions to solve linear and nonlinear VIE of the second kind, the algorithm is characterized by low cost that allowing addition of more number of collocation points N , thereby this method increasing the accuracy of the obtained results.

3. Volterra Integral Equation Solution

The most standard form of Volterra integral equation is given as in equation (1)

$$y(t) + \int_0^t K(t,s)\, y(s) ds = g(t), \qquad t \in [0,T] \qquad (1)$$

When assuming that the solution is required over a finite interval [0, *T*] that t belongs to this interval with $T < \infty$, *K(t,s)* is continuous in $0 \le s \le t \le T$, also g(t) is continuous finite (I). With these assumptions, the solution *y(t)* to (1) exists, is unique and *y(t)* is continuous in [0,*T*] [6].

Trapezoidal Rule is a distinguished case of Newton–Cotes' formulas based on its Formula that states. The trapezoidal rule is considered more saving to compute a sequence of approximations. VIE solving by the Trapezoidal approximation by dividing the interval [a,b] into equal subinterval, then the trapezoidal rules will be applied on equation(2):

$$\int_a^b f(\mathrm{x})\mathrm{dx} = h\left[\frac{f(a)+f(b)}{2} + \sum_{k=2}^{n-1} \mathrm{f}(x_k)\right] \qquad (2)$$

Where h represents equal subinterval of [a,b], *h* calculated by equation ($h = b - a\ /N$) $N \in N^*$, while ($x_k = a + k - 1\ h\ ,\ 1 \le k \le N+1$). [1]

4. Cascade-Forward Neural Network (CFNN)

CFNN structure component of input, output and one or more hidden layers, there is a connection from the input to each layer in the network, and a connection from each layer to the successive layers in the network as shown in **Figure (1)** [11]. This allows the inputs to directly, influence the output nodes by embedding additional information and features to it [12].

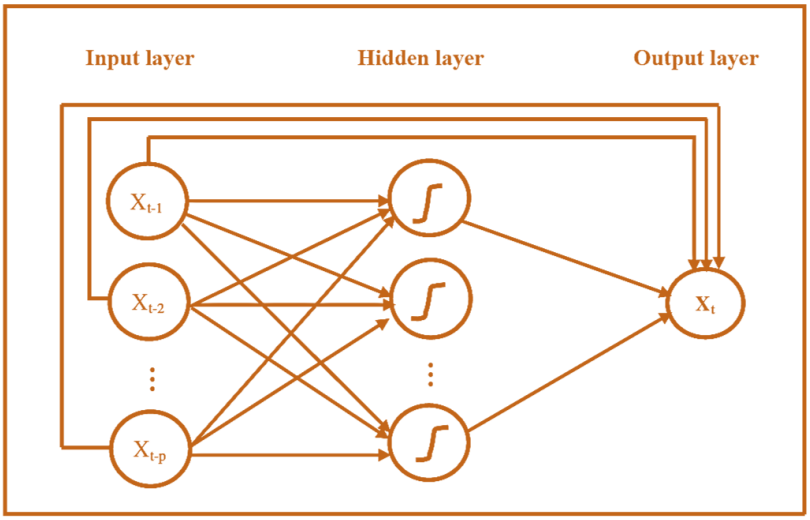


Figure 1.structure of CFNN

CFNN are similar in structure to multilayer feedforward neural network except CFNN which has a direct weighted connection from its input to output layer with two additional connections. CFNNs with more layers might learn complex relationships, while in FFNN the connection formed between input and output is indirect relationship of sample relationships. [11]

CFNN model equation can be formed as equation (3):

$$y = \sum_{i=1}^{n} A^i w_i^i x_i + A^o(\sum_{j=1}^{k} w_j^o A_j^h(\sum_{i=1}^{n} w_{ji}^h x_i)) \qquad (3)$$

Where A^i is the activation function from the input layer to the output layer, x_i is an input sample, w_i^i is weight from the input layer to the output layer, A^o is the activation function of output layer, and the activation function of the hidden layer is A^h. when the bias w^b is added to the input layer and the activation function of each neuron in the hidden layer A^h then equation formed as equation (4) [11]:

$$y = \sum_{i=1}^{n} A^i w_i^i x_i + A^o(w^b + \sum_{j=1}^{k} w_j^o A_j^h(w_j^b + \sum_{i=1}^{n} w_{ji}^h x_i)) \qquad (4)$$

After construct CFNN, the training process of the CFNN will be done, in the train state the samples inputs to the network to obtain the target output. In this state the weight and the bias based to be used, CFNN need inputs, target result, weight and bias to begin training process. There are many types of activation functions used in training process of neural networks, the training activation function "trainlm" considers the fastest backpropagation algorithm in network training function that updates weight and bias values according to Levenberg-Marquardt optimization. The number of hidden layers depend on the size of input samples and the number of neurons in each hidden layer depend on the number of target output [13]. CFNN performance like any neural network can be measured, thus the mean squared error (MSE) is very commonly used as a perfect general purpose error metric for numerical predictions [12].

5. Methodology

The mechanism of the proposed method is in three steps: the first step extracts the exact numerical solution of VIE using trapezoidal quadrature rule and then gets the mean of each solution which represents features that input to CFNN, these features saved in fixed database to use it in the second step. The second step, CFNN model is created then training multi times to obtain the same numerical solutions of the quadrature method. In the third step, the simulation CFNN model test the values to obtain the VIE solutions. The model of this method is illustrated in block diagram as in **Figure (2).**

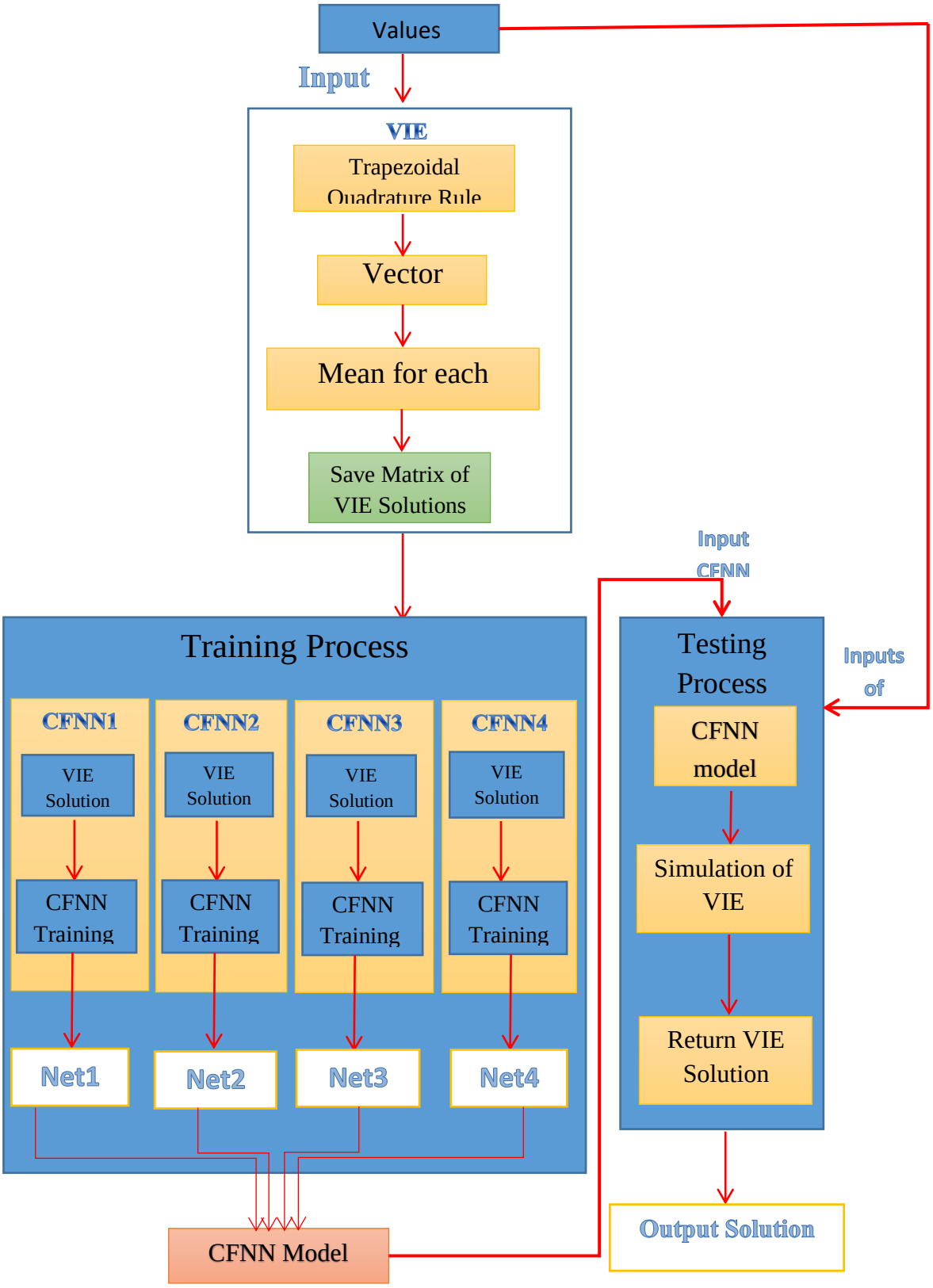


Figure 2. Block Diagram of the Proposed Model

5.1.Design and Implementation

In the proposed method, selected values from 1 to 20 to extract the numerical solution of VIE to be an example to prove the work, solutions of VIE for 20 values are calculated, then

the mean of each of solution is calculated to be a vector of solution for each value. These vectors will saved to be inputs for CFNN, this process is explained in algorithm (1):

Algorithm 1: Trapezoid Rule for Volterra Integral Equations (VIE):
Input: Numerical values
Output: Vectors of Mean of VIE

- **Step1.** Select 20 values as an examples of the proposed method [1 .. 20].
- **Step2.** Calculate VIE by use trapezoid rule.
- **Step3.** Find mean of each VIE vector.
- **Step4.** Save output of VIE.
- **Step5:** Repeat steps (2-4) 20 times.

CFNN building based on the entered inputs to get the target value of the desired solution of VIE, CFNN structure contains one input layer, one hidden layer which contains 20 neurons, and one output layer. In CFNN used defaults parameters for weights and bias values that update by useing "trainlm" as activation function of CFNN, CFNN structure creates in algorithm (2), while figure (3) shows the structure of CFNN which builds in this work.

Algorithm2: Create CFNN
Input: numbers of vectors in Matrix of Mean of VIE
Output: CFNN

- **Step1.** Select network structure type "cascade forward neural network".
- **Step2.** Determine No. of layers depend on numbers of vectors in Matrix of Mean of VIE [one input layer – one hidden layer – one output layer].
- **Step3.**No. of neurons in input layer (No.I) equal the value of vector.
- **Step4.**No. of neurons in hidden layer (No.H) depend on the number of input vectors.
- **Step5.**No. of neurons in output layer (No.O) depend on the number of target vectors.
- **Step6**. Calculate weight and bias values:
  - IW: {2x1 cell} containing 2 input weight matrices.
  - LW: {2x2 cell} containing 1 layer weight matrix.
  - b: {2x1 cell} containing 2 bias vectors.
- **Step7.** Create (CFNN, No of layers, No.I, No.H, No.O, IW, LW,b).

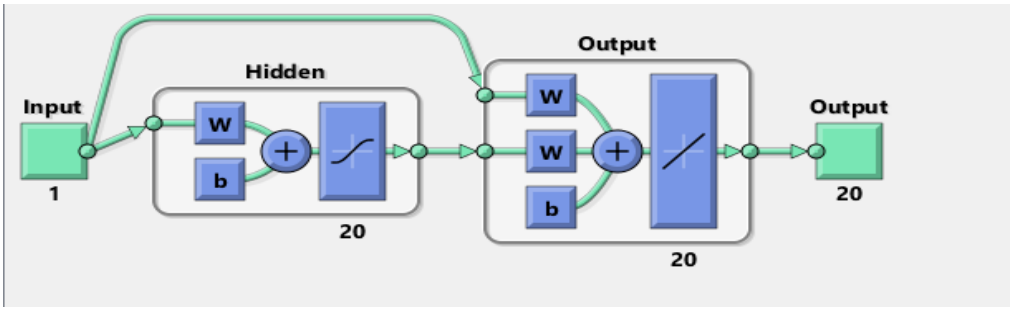


Figure 3. Structure of CFNN

The proposed method of training CFNN explained in algorithm (3), it depends on train CFNN multi times then combine the results of training CFNNs to exceed the errors in training results of CFNN and gets the optimum performance, the training process terminated after there is no enhancement in their performance. In this work the performance of CFNN stopped improving after four training of CFNN, Four CFNN training is combined and saved to be simulated the result in test state. The training process of CFNN can be seen in figure (4) which appears the training progress of CFNN for each time.

Algorithm3: Training process
Input: CFNN, Vectors of Mean of VIE
Output: Cell of CFNNs model

- **Step1.** Call algorithm1 "Trapezoid Rule for Volterra Integral Equations (VIE)" with 20 values.
- **Step2**. Call algorithm2 "create CFNN" with 20 neurons.
- **Step3.** Train CFNN with inputs (vectors of mean of VIE).
- **Step4**. Evaluate network outputs by given inputs.
- **Step5**. Save CFNN with performance value.
- **Step6**. Evaluate the performance of CFNN if there are errors in simulation results repeat the steps from (3 to 6).
- **Step7**. Save CFNN as "net- no attempt"
- **Step8**. After 4 attempts (repeat steps 3 to 7), the best performance would been done by get the target result.
- **Step9**. Save net-1, net-2, net-3 and net-4 as "Cell of CFNNs model".

a. CFNN training1 b. CFNN training2

c. CFNN training3 d. CFNN training4

Figure 4. Training process of CFNNs: a. CFNN training1, b. CFNN training2, c. CFNN training3, d. CFNN training4

The four CFNN training are collected in one programming state to get all the target solutions successfully. Then the final models of CFNN are stimulated their results to get the target solutions of VIE, the algorithm of the proposed model of testing process can be seen in algorithm (4).

Algorithm4: Testing process
Input: Cell of CFNNs Model
Output: Solutions of VIE

- **Step1.** Input any value from [1 .. 20]
- **Step2.** Return the vector of mean of VIE of the same input value
- **Step3.** Call CFNNs Cell.
- **Step4.** Simulate nets of CFNNs Cell with input value as following :
  - If result of net-1 equal the input value, the solution of VIE return and go to the 5 step.
  - If result of net-2 equal the input value, the solution of VIE return and go to the 5 step.
  - If result of net-3 equal the input value, the solution of VIE return and go to the 5 step.

- If result of net-4 equal the input value, the solution of VIE return and go to the 5 step.

- **Step5.** Get the solution of VIE for this value.

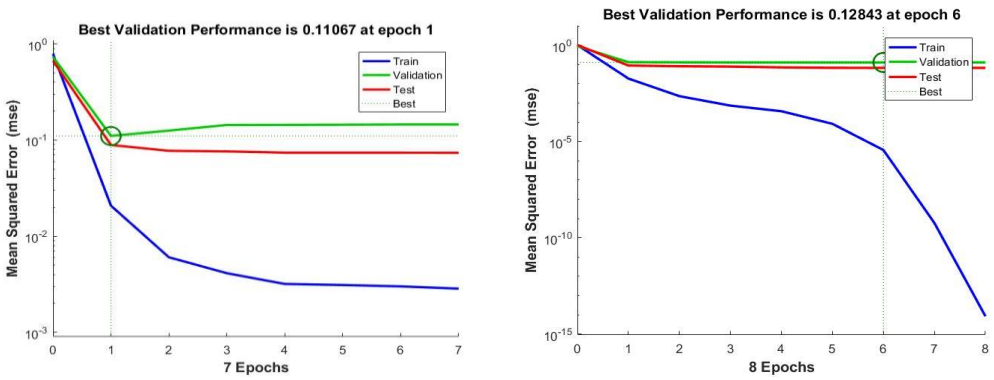
5.2.Performance Measures

This paper shows the efficiency of CFNN in obtain the target solution of VIE by measuring the performance of CFNN model to test their work, MSE used to measure the quality of neural network, the MSE of training CFNN for four training attempts appear in **Table (1).**

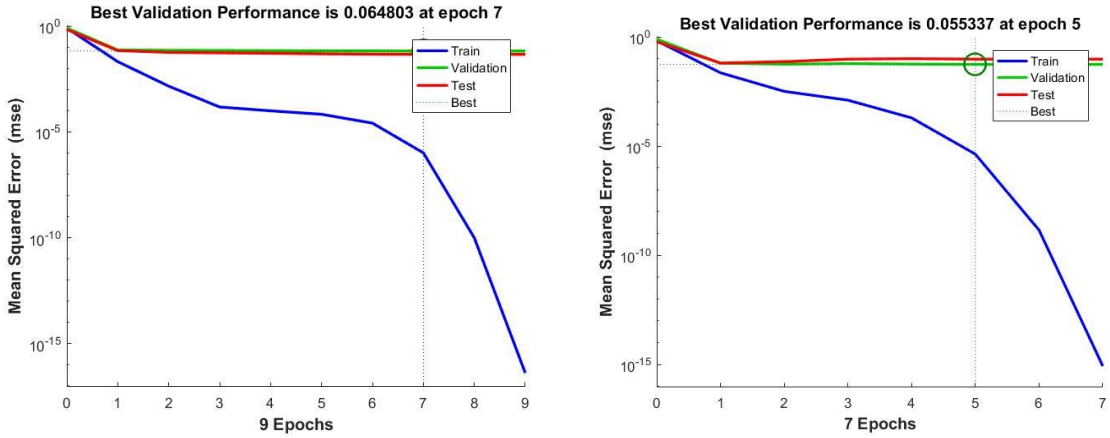
Table (1) The results of training CFNN for four attempts

No. of training attempt	MSE
1	0.0293
2	0.0445
3	0.0165
4	0.0227

This method can calculate the best validation performance of each CFNN training by extracting the MSE for each iterations of CFNN training, this process calculates for four CFNN training, the best value of validation performance obtained for four CFNNs training shown in figure (5), this figure contains chart for each CFNN describes the relation between the number of iteration and MSE for CFNN training. After four CFNN training attempts, all the target output would have been returned exactly, thus the training process will be stopped and save CFNNs.



a. Performance of CFNN1 b. Performance of CFNN2



c. Performance of CFNN3 d. Performance of CFNN4

Figure (5) performance of CFNN for four training networks: a. performance of CFNN1, b. performance of CFNN2, c. performance of CFNN3, d. performance of CFNN4

6. Conclusions

This paper used a proposed technique for getting volterra integral equations (VIE) solutions by useing cascade-forward neural network model. CFNN would have been built to simulate VIE solutions, CFNN represents an appropriate network to solve integral equation because its structures is nonlinear and nonparametric, thus makes it more flexible to get the predictions of time series. In this work, CFNN is trained four times to get the target results with the best performance, then get the results of CFNNs training models to be in one CFNN simulation model for simulating values of VIE solutions. In this method exceeding the training errors in output result, so the simulation process of CFNN with values of VIE would been more reliability. CFNN model that is proposed in this search can simulate more values of VIE that training on it with no errors percentage because CFNN is trained multi times until the target results would be getting, then all the CFNNs training combined to be one CFNN model in simulation process. Thus the results of the proposed method appeared that CFNN model simulated the values of VIE solutions with 100% percentage.

7.Acknowledgment

I would like to thank the university that facilitated the task of researcher prepared for the research and affiliated to it, which are: Middle Technical University (www.mtu.edu.iq).

References

1. Aigo MU. On the numerical approximation of Volterra integral equations of the second kind using quadrature rules, International Journal of Advanced Scientific and Technological Research.. **2013**,*1*, 558-64.
2. Yadav, N.; Yadav, A.; Kumar, M. An introduction to neural network methods for differential equations. Netherlands: Springer. **2015**, E-ISSN 2191-5318.
3. Al-allaf ON. Cascade-forward vs. function fitting neural network for improving image quality and learning time in image compression system. In Proceedings of the world congress on engineering. **2012**, *2*, 4-6.
4. Warsito B, Santoso R, Yasin H. Cascade forward neural network for time series prediction. InJ. Phys. Conf. Ser. **2018**, 1025, 012097.
5. Tahmasbi A. A new approach to the numerical solution of linear Volterra integral equations of the second kind. Int. J. Contemp. Math. Sciences. 2008. 3, 32, 1607-10.
6. Isaacson SA, Kirby RM. Numerical solution of linear Volterra integral equations of the second kind with sharp gradients. Journal of Computational and Applied Mathematics. **2011**, *235, 14*, 4283-301.
7. Mirzaee F. A computational method for solving linear Volterra integral equations. Applied Mathematical Sciences. **2012**, *6, 17*, 807-14.
8. Rahman MA, Islam MS, Alam MM. Numerical solutions of Volterra integral equations using Laguerre polynomials. Journal of scientific research. 2012, 4, 2, 357.
9. Kolk M, Pedas A. Numerical solution of Volterra integral equations with singularities. Frontiers of Mathematics in China. 2013, 8, 2, 239-59.
10. Costarelli D, Spigler R. Solving Volterra integral equations of the second kind by sigmoidal functions approximation. The Journal of Integral Equations and Applications. **2013**, *25, 2*, 193-222.
11. Warsito B, Santoso R, Yasin H. Cascade forward neural network for time series prediction. InJ. Phys. Conf. Ser. **2018**, 1025, 012097.
12. Alkhasawneh MS, Tay LT. A hybrid intelligent system integrating the cascade forward neural network with elman neural network. Arabian Journal for Science and Engineering. **2018**, *43, 12*, 6737-49.
13. Yusoff Y, Zain AM, Sharif S, Sallehuddin R, Ngadiman MS. Potential ANN prediction model for multiperformances WEDM on Inconel 718. Neural Computing and Applications. **2018**, *30, 7*, 2113-27.