Experimental Analysis of Artificial Neural Networks Used for Function Approximation

Akram Mustafa

Computer Science Department, Al al-Bayt University, Mafraq, Jordan-Amman, P.O.BOX 922283, 11192, JORDAN

Abstract

Artificial neural network (ANN) is a powerful mathematical computational model, which is used in many important life applications such as function approximation, data regression, solving classification problem, pattern recognition and much other application. The objective of this paper is to use ANN for function approximation, to find the relationship from a given finite input-output data, that using in the many different application in this time. Different types of ANN will be created, trained and tested; the obtained experimental results will be compared and discussed in order to select the best type of ANN, the best ANN architecture which will minimize the error between the targeted function value and the calculated one.

Key words:

ANN, neuron, FFANN, cascade ANN, Elman ANN, ANN architecture, ANN parameters

1. Introduction

Function approximation means determining one or more output functions depending on the values of the input variables, such as predicting the value of y when we have the values of x1, x2, x3, and x4 as show in table 1:

Table 1: Part of the experimental data

X1	X2	X3	X4	y
0.4658	15.5800	84.9479	75.1238	165.0714
93.3944	95.5501	62.9449	67.8084	212.2650
69.9019	72.7400	20.6862	59.2050	69.9172
98.2575	42.4033	72.1063	43.9034	326.5274
26.3808	63.7372	54.0981	29.4660	121.8527
35.3972	24.4231	39.2436	2.2063	161.8957
52.8335	83.3235	59.3606	40.2054	160.2199
79.5806	0.2929	70.1631	56.1366	313.2211
43.4387	43.1544	40.9139	71.9669	94.4980
93.8172	9.6323	86.1930	59.6668	376.9143

Function approximation is the fundamental problem in a vital majority of real world applications, such as prediction, pattern recognition, data mining and classification, thus finding a tool to predict a function value is a very important task [20].

In this paper we will focus on selecting the appropriate ANN to calculate the function value for a given variables value with minimum error between the targeted and the calculated values of the function.

1.1 Artificial Neural Network

Artificial neural network is a set of fully connected neurons as shown in figure 1, these neurons are organized in layers, one input layer, one output layer and zero or more hidden layers [1], [2].

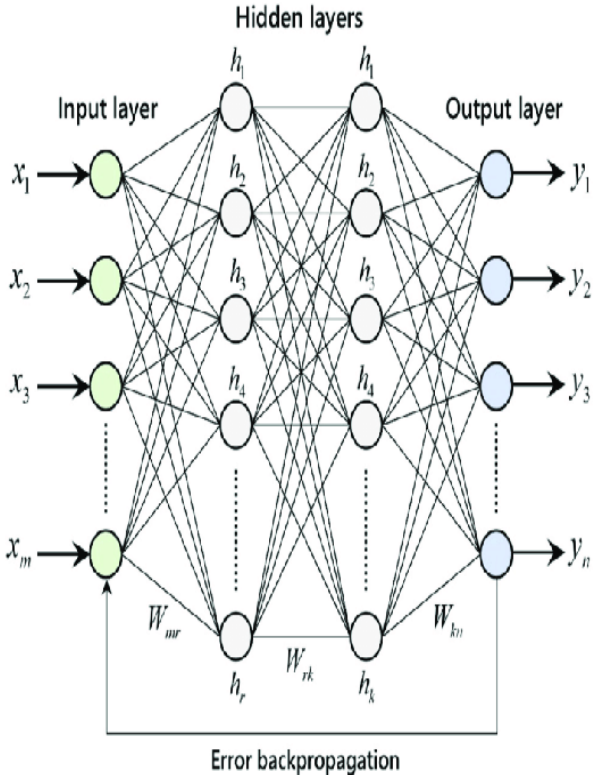


Fig. 1 ANN architecture

ANN must be fed by an input data set in order to calculate the predicted outputs, each neuron has one or more inputs, and each input is associated with a weight [3], figure 2 shows the mathematical model of a neuron which operates applying the following steps:

Manuscript received September 5, 2019
Manuscript revised September 20, 2019

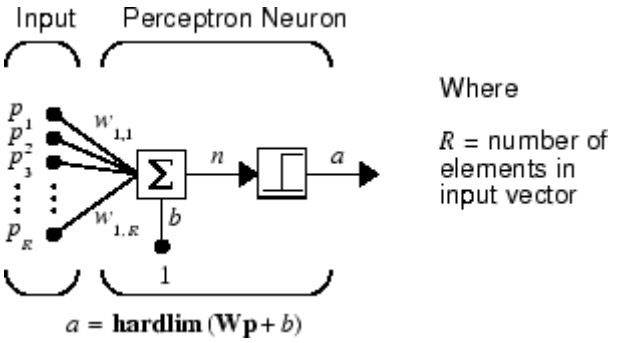


Fig. 2 Neuron mathematical model

- Summation of the products, each product is a multiplication of the input with the associated weight.
- According to the selected activation function for the layer (and neuron in the layer), the neuron calculates the output.

In this paper we will focus on the mostly used activation functions: Linear, logsig, and tansig [4], [5];
Using the linear activation function produces an output equal the sum of products, figures 3 and 4 show how the outputs are calculated using logsig and tansig activation functions [6]:

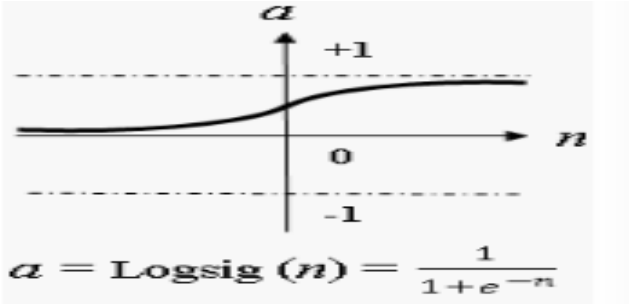


Fig. 3 Using logsig to calculate the output

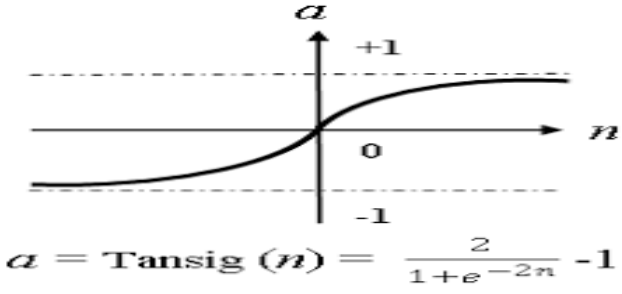


Fig. 4 Using tansig to calculate the output

Figure 5 shows a Simulink model which simulates the neuron operations [7]:

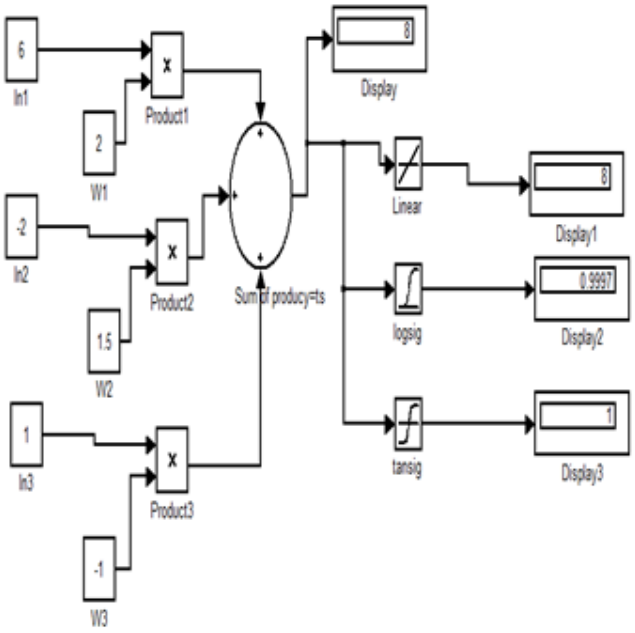


Fig. 5 Neuron Simulink model

Using ANN for function approximation requires paying attention on the following important issues:

1) Selecting an input data set, arranging this set in suitable matrix to be used for feeding ANN.
2) Defining the correct targets (the predicted values of the output function).
3) Selecting ANN type.
4) For the selected ANN type define ANN architecture, which includes:
   a) Number of layers to be used,
   b) Number of neurons in each layer,
   c) Activation functions for layers.
5) Initializing all weights to zero.
6) Defining some parameters for ANN such as: error which must be closed to zero, number of training cycles, and each cycle is used to calculate the outputs in the forward phase, and the propagated error in the back ward phase, if the calculated error matches selected error then the training will be stopped, otherwise the training will be continued.

1.2 Feed Forward ANN with Back Propagation

This type of ANN has the architecture [5] and [7] as shown in figure 6, the input layer neuron are connected directly to the second layer, where the second neuron are connected to third layer and so on.

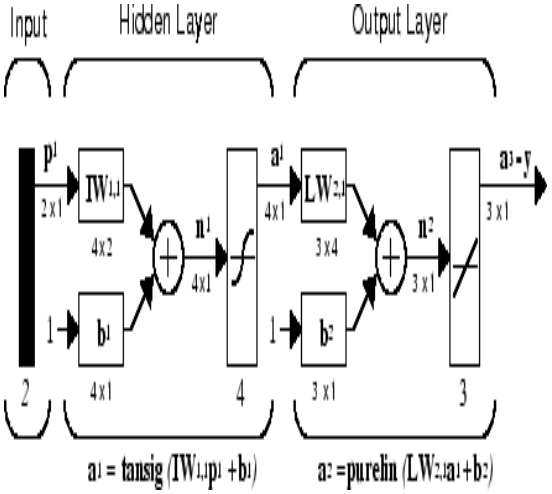


Fig. 6 Feed forward ANN

This type of ANN works as follows (in each training cycle):

- Starting from the input layer find the output of each neuron (feed forward).
- Starting from the output layer calculate the error using back propagation, if the error is accepted stop training, else go back by finding the error and adjusting the weights.

1.3 Cascade FFANN

Cascade feed forward ANN (CFFANN) models are similar to FFANN, but includes a weight connection from the input to each layer and from each layer to the successive layers [8], [9] and [10]. While two-layer feed forward networks can potentially learn virtually any input/output relationship, feed-forward networks with more layers might learn complex relationships more quickly [6]. The function newcf creates cascade-forward networks [17]. For example(see figure 7) , a three layer network has connections from layer 1 to layer 2, layer 2 to layer 3, and layer 1 to layer 3. The three-layer network also has connections from the input to all three layers. The additional connections might improve the speed at which the network learns the desired relationship [9] and [18]. CF artificial intelligence model is similar to feed forward back propagation neural network in using the back propagation algorithm for weights updating, but the main symptom of this network is that each layer of neurons related to all previous layer of neurons [9], [11], [16] and [17].

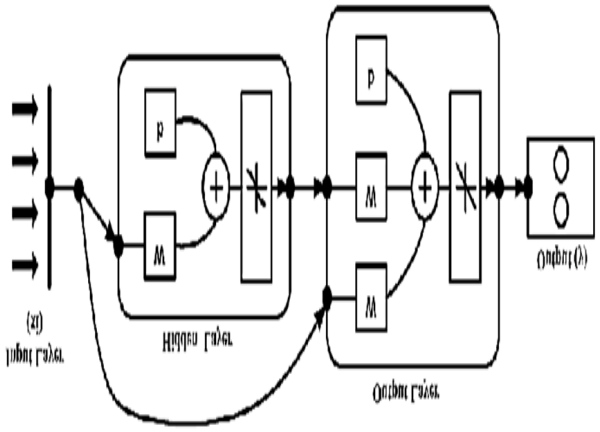


Fig. 7 CFFANN architecture

1.4 Elman FFANN

One of the most known recurrent neural networks is Elman neural network [12], [13] Typical Elman network has one hidden layer with delayed feedback. The Elman neural network is capable of providing the standard state-space representation for dynamic systems. This is the reason why this network architecture is utilized as a recurrent neural equalizer. Generally, this network is considered as a special kind feed-forward network, including additional memory neurons and local feedback [14], [15] and [19]. Typical structure of Elman neural network is depicted in figure 8.

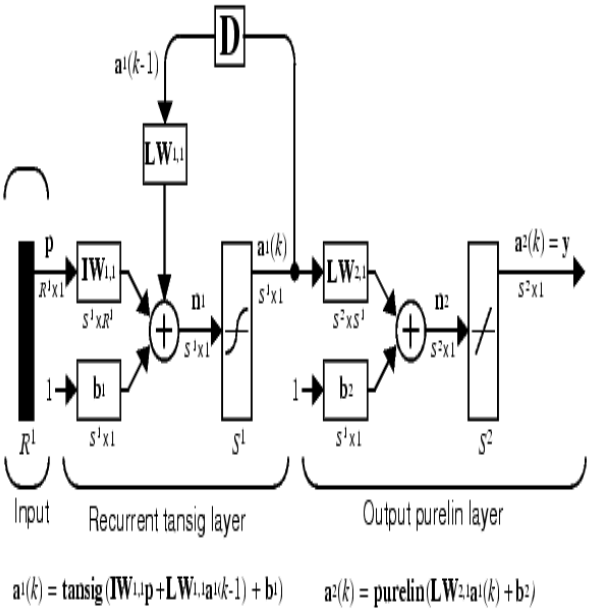


Fig. 8 Elman ANN architecture

2. Implementation and Experimental Results

For the above mentioned types of ANN a matlab codes were written to create, train and test ANN, an input data set of 100 elements were selected such as the data shown in table 1. The maximum value of the selected input data set was around 100, so we do the work in 2 experiments: the first experiment without data normalization (using the input data set to feed ANN), the second by using normalized data by dividing the input data set by 1000 to avoid zero results of the exponent component of the activation function.

Matlab codes were written to implement various types of ANN; each code was implemented applying the following steps:

1) Select the arranged input data set to feed ANN.
2) Select the output targets.
3) Normalize the data (if necessary).
4) Select the type of ANN.
5) Create ANN by selecting the number of layers, the number of neurons in each layer, and the activation function for each layer.
6) Initialize the weights to zero.
7) Set some parameters for ANN (error must be very closed to zero, High number of training cycles).
8) Train ANN.
9) Save ANN if it meets the performance requirements.
10) Run ANN using a selected input set data items.

The following matlab code was implemented for various ANN:

```
clear all, close all, clc
load ex1,load ex2,load ex3,load ex4
y=(2*ex1-ex2+3*ex3-ex4)/1000;
x=[ex1;ex2;ex3;ex4]/1000;
%CFANN Creation
net1=newcf(minmax(x),[4 1],{'purelin','purelin'});
%FFANN Creation
%net1=newff(minmax(x),[4 1],{'purelin','purelin'});
%Elman ANN Creation
%net1=newelm(minmax(x),[4 1],{'purelin','purelin'});
net1=init(net1);
net1.trainParam.goal=0;
net1.trainParam.epochs=8000;
tic
net1=train(net1,x,y);
toc
```

First experiment: Without data normalization

The three types of ANN were created, trained and tested using various architectures and activation functions, tables 2, 3, and 4 show the results of this experiment where below symbols (*, & and #) are meaning:

*- Meet the performance requirement.
&- Acceptable but doesn't real.
#- Not acceptable.

Table 2: Feed forward ANN results

ANN size (byte)	*Input layer neurons and activation function*	*Hidden layer neurons and activation function*	*Output layer neurons and activation function*	Training cycles	Training time	Error
26659*	4 linear*	0*	1 Linear*	6*	0.235000*	0*
26653&	4 logsig&	0&	1 Linear&	8000&	189.481000&	0.0029&
26653&	4 tansig&	0&	1 Linear&	8000&	188.524000&	0.00071942&
33367*	4 linear*	4 linear*	1 Linear*	6*	0.221000*	0*
33367#	4 linear#	4 logsig#	1 Linear#	4#	0.222000#	Fail#
33367#	4 linear#	4 tansig#	1 Linear#	5#	0.248000#	Fail#
33367&	4 logsig&	4 logsig&	1 Linear&	8000&	168.265000&	0.9223&
33367&	4 logsig&	4 tansig&	1 Linear&	8000&	122.017000&	0.9728&
33367&	4 logsig&	4 linear&	1 Linear&	8000&	200.949000&	0.00071942&
33367#	4 linear#	4 logsig#	1 Linear#	4#	0.240000#	Fail#
33367#	4 linear#	4 tansig#	1 Linear#	5#	0.245000#	Fail#
33367&	4 tansig&	4 linear&	1 Linear&	8000&	203.476000&	0.0014&
33367#	4 tansig#	4 logsig#	1 Linear#	70#	0.636000#	Fail#

Table 3: Cascade ANN results

ANN size (byte)	Input layer neurons and activation function	Hidden layer neurons and activation function	Output layer neurons and activation function	Training cycles	Training time	Error
28667*	4 linear*	0#	1 Linear*	6*	0.225000*	0*
28661*	4 logsig*	0#	1 Linear*	4*	0.217000*	0*
28661*	4 tansig*	0#	1 Linear*	4*	0.217000*	0*
39759*	4 linear*	4 linear#	1 Linear*	4*	0.223000*	0*
39753*	4 linear*	4 logsig#	1 Linear*	10*	0.259000*	0*
39753*	4 linear*	4 tansig#	1 Linear*	5*	0.235000*	0*
39747*	4 logsig*	4 logsig#	1 Linear*	5*	0.231000*	0*
39747*	4 logsig*	4 tansig#	1 Linear*	6*	0.239000*	0*
39753*	4 logsig*	4 linear#	1 Linear*	7*	0.237000*	0*
39753*	4 linear*	4 logsig#	1 Linear*	2030*	33.734000*	0*
39753*	4 linear*	4 tansig#	1 Linear*	6*	0.235000*	0*
39753*	4 tansig*	4 linear#	1 Linear*	8*	0.252000*	0*
39747*	4 tansig*	4 logsig#	1 Linear*	6*	0.241000*	0*

Table 4: Elman ANN results

ANN size (byte)	*Input layer neurons and activation function*	*Hidden layer neurons and activation function*	*Output layer neurons and activation function*	Training cycles	Training time	Error
29029&	4 linear&	0&	1 Linear&	8000&	19.987000&	5.9849&
29023#	4 logsig#	0#	1 Linear#	8000#	21.126000#	Fail#
29023#	4 tansig#	0#	1 Linear#	8000#	21.769000#	Fail#
38105&	4 linear&	4 linear&	1 Linear&	8000&	24.757000&	17.9475&
38099#	4 linear#	4 logsig#	1 Linear#	8000#	25.813000#	Fail#
38099#	4 linear#	4 tansig#	1 Linear#	8000#	25.221000#	Fail#
38093#	4 logsig#	4 logsig#	1 Linear#	8000#	26.434000#	Fail#
38093#	4 logsig#	4 tansig#	1 Linear#	8000#	25.968000#	Fail#
38099#	4 logsig#	4 linear#	1 Linear#	8000#	25.536000#	Fail#
38099#	4 linear#	4 logsig#	1 Linear#	8000#	25.846000#	Fail#
38099#	4 linear#	4 tansig#	1 Linear#	8000#	25.076000#	Fail#
38099#	4 tansig#	4 linear#	1 Linear#	8000#	25.360000#	Fail#
38093#	4 tansig#	4 logsig#	1 Linear#	8000#	26.535000#	Fail#

From the obtained experimental results shown above we can conclude the following:

- It is better to use cascade ANN with any architecture to solve the function approximation problem.
- CFFANN with one input layer and output layer gave the best performance (zero error and 0.225000 s training time).
- Only one FFANN architecture gave a good performance.
- No EANN architecture gave an acceptable performance.

Second experiment: With data normalization

The three types of ANN were created, trained and tested using various architectures and activation functions, tables 5, 6, and 7 show the results of this experiment:

Table 5: Feed forward ANN results with data normalization

ANN size (byte)	*Input layer neurons and activation function*	*Hidden layer neurons and activation function*	*Output layer neurons and activation function*	Training cycles	Training time(s)	Error
26659#	4 linear#	0#	1 Linear#	11*	0.253000*	0*
26653#	4 logsig#	0#	1 Linear#	8000&	189.748000&	0.0022&
26653*	4 tansig*	0*	1 Linear*	8000*	115.678000*	0*

33367*	4 linear*	4 linear*	1 Linear*	12*	0.264000*	0*
33367#	4 linear#	4 logsig#	1 Linear#	5#	0.253000#	Fail#
33367&	4 linear&	4 tansig&	1 Linear&	3576&	21.354000&	0.6029&
33367#	4 logsig#	4 logsig#	1 Linear#	3#	0.233000#	Fail#
33367&	4 logsig&	4 tansig&	1 Linear&	8000&	198.120000&	0.0683&
33367*	4 logsig*	4 linear*	1 Linear*	8000*	81.322000*	0*
33367#	4 linear#	4 logsig#	1 Linear#	4#	0.245000#	Fail#
33367&	4 linear&	4 tansig&	1 Linear&	8000&	216.336000&	0.0029&
33367&	4 tansig&	4 linear&	1 Linear&	8000&	39.303000&	0.0151&
33367#	4 tansig#	4 logsig#	1 Linear#	4#	0.271000#	Fail#

Table 6: Cascade ANN results with data normalization

ANN size (byte)	*Input layer neurons and activation function*	*Hidden layer neurons and activation function*	*Output layer neurons and activation function*	Training cycles	Training time(s)	Error
28667*	4 linear*	0*	1 Linear*	13*	0.237000*	0*
28661*	4 logsig*	0*	1 Linear*	5*	0.236000*	0*
28661*	4 tansig*	0*	1 Linear*	22*	0.412000*	0*
39759*	4 linear*	4 linear*	1 Linear*	15*	0.295000*	0*
39753*	4 linear*	4 logsig*	1 Linear*	340*	6.438000*	0*
39753*	4 linear*	4 tansig*	1 Linear*	230*	5.042000*	0*
39747*	4 logsig*	4 logsig*	1 Linear*	8000*	47.111000*	0*
39747*	4 logsig*	4 tansig*	1 Linear*	26*	0.419000*	0*
39753*	4 logsig*	4 linear*	1 Linear*	4880*	126.318000*	0*
39753*	4 linear*	4 logsig*	1 Linear*	1294*	30.826000*	0*
39753*	4 linear*	4 tansig*	1 Linear*	861*	11.960000*	0*
39753*	4 tansig*	4 linear*	1 Linear*	8000*	131.544000*	0*
39747*	4 tansig*	4 logsig*	1 Linear*	5*	0.256000*	0*

Table 7: Elman ANN results with data normalization

ANN size (byte)	*Input layer neurons and activation function*	*Hidden layer neurons and activation function*	*Output layer neurons and activation function*	Training cycles	Training time(s)	Error
29029*	4 linear*	0*	1 Linear*	3589*	9.285000*	0*
29023#	4 logsig#	0#	1 Linear#	8000#	20.989000#	Fail#
29023#	4 tansig#	0#	1 Linear#	8000#	21.088000#	Fail#
38105*	4 linear*	4 linear*	1 Linear*	3872*	11.341000*	0*
38099#	4 linear#	4 logsig#	1 Linear#	8000#	24.828000#	Fail#
38099#	4 linear#	4 tansig#	1 Linear#	8000#	24.959000#	Fail#
38093#	4 logsig#	4 logsig#	1 Linear#	8000#	25.784000#	Fail#
38093#	4 logsig#	4 tansig#	1 Linear#	8000#	25.884000#	Fail#
38099#	4 logsig#	4 linear#	1 Linear#	8000#	24.913000#	Fail#
38099#	4 linear#	4 logsig#	1 Linear#	8000#	25.325000#	Fail#
38099#	4 linear#	4 tansig#	1 Linear#	8000#	25.335000#	Fail#
38099#	4 tansig#	4 linear#	1 Linear#	8000#	24.931000#	Fail#
38093#	4 tansig#	4 logsig#	1 Linear#	8000#	25.776000#	Fail#

From the obtained experimental results shown above we can conclude the following:

- CFFANN still the best type to be selected for function approximation.
- Anny selected CFFANN architecture with any activation function will produce a zero error.
- FFANN can be used for function approximation, but we have to be careful of selecting the appropriate architecture with a suitable activation function
- Elman ANN can be used for function approximation using any architecture, but the activation functions must be linear.
- CFFANN always give a better performance.
- For any selected architecture of CFFANN, the created ANN requires a small size of memory to

be stored in, and ranges from 29 to 40 Kbytes.

Third experiment:
Normalization of both input data set and targets, making the target always less than 1.

Here we took the minimum architecture of ANN (one input layer with 4 neurons, and one output layer with one neuron), tables 8, 9 and 10 show the results of this experiment.

Table 8: Experiment 3 results 1

ANN type	Input layer activation function	Output layer activation function	ANN size (byte)	Training cycles	Training time(s)	Error
FFANN	purelin*	purelin*	26659*	7*	0.236000*	0*
	logsig&	purelin&	26653&	8000&	70.701000&	5.4662&
	tansig&	purelin&	26653&	8000&	65.913000&	5.6223&
CFANN	purelin*	purelin*	28667*	4*	0.238000*	0*
	logsig*	purelin*	28661*	5*	0.253000*	0*
	tansig*	purelin*	28661*	4*	0.230000*	0*
ELANN	purelin*	purelin*	29029*	1987*	5.103000*	0*
	logsig#	purelin#	29023#	8000#	20.704000#	Fail#
	tansig#	purelin#	29023#	8000#	20.927000#	Fail#

Table 9: Experiment 3 results 2

ANN type	Input layer activation function	Output layer activation function	ANN size (byte)	Training cycles	Training time(s)	Error
FFANN	purelin#	logsig#	26653#	16#	0.252000#	Fail#
	logsig#	logsig#	26647#	8000#	28.270000#	Fail#
	tansig#	logsig#	26647#	8000#	92.489000#	Fail#
CFANN	purelin#	logsig#	28661#	17#	0.257000#	Fail#
	logsig&	logsig&	28655&	1859&	13.233000&	11.1820&
	tansig&	logsig&	28655&	8000&	30.586000&	6.3496&
ELANN	purelin#	logsig#	29023#	1720#	4.539000#	Fail#
	logsig#	logsig#	29017#	8000#	20.833000#	Fail#
	tansig#	logsig#	29017#	8000#	21.014000#	Fail#

Table 10: Experiment 3 results 3

ANN type	Input layer activation function	Output layer activation function	ANN size (byte)	Training cycles	Training time(s)	Error
FFANN	purelin#	logsig#	26653#	16#	0.252000#	Fail#
	logsig#	logsig#	26647#	8000#	28.270000#	Fail#
	tansig#	logsig#	26647#	8000#	92.489000#	Fail#
CFANN	purelin#	logsig#	28661#	17#	0.257000#	Fail#
	logsig&	logsig&	28655&	1859&	13.233000&	11.1820&
	tansig&	logsig&	28655&	8000&	30.586000&	6.3496&
ELANN	purelin#	logsig#	29023#	1720#	4.539000#	Fail#
	logsig#	logsig#	29017#	8000#	20.833000#	Fail#
	tansig#	logsig#	29017#	8000#	21.014000#	Fail#

From the obtained results in experiment 3 we can conclude that cascade ANN provides the best performance when the activation function of the output layer was setting to linear.

3. Conclusion

Different types of ANN were implemented tested with a function approximation test data, the obtained experimental results showed that:

- CFFANN is the best type of ANN to be used for

function approximation, with any architecture of CFFANN and with any activation functions we can reach a zero error between the target value of the function and the calculated one by CFFANN.
- FFANN can be also used for function approximation, but here we have to be careful when selecting ANN architecture and when assigning the activation function for each layer.
- Elman ANN can be used for function approximation, but the activation functions must be linear for all ANN layers.
- Data normalization improves the performance of any ANN type.

References

[1] Akram A Moustafa " Performance evaluation of artificial neural networks for spatial data analysis, Contemporary engineering sciences, v 4, issue 4, pp 149-163, 2011.

[2] Khaled M Matrouk, Alasha'ary, Abdullah I. Al-Hasanat, Ziad A. Al-Qadi, Hasan M. Al-Shalabi, Investigation and Analysis of ANN Parameters, European Journal of Scientific Research, v 121, issue 2, pp 217-225, 2014.

[3] Jamil Al-Azzeh, Ziad Alqadi, Mohammed Abuzalata, Performance Analysis of Artificial Neural Networks used for Color Image Recognition and Retrieving, IJCSMC, Vol. 8, Issue. 2, pp 20 – 33, 2019.

[4] Khaled M Matrouk, Alasha'ary, Abdullah I. Al-Hasanat, Ziad A. Al-Qadi, Hasan M. Al-Shalabi, Experimental Investigation of Training Algorithms used in Back propagation Artificial Neural Networks to Apply Curve Fitting, European Journal of Scientific Research, v 121, issue 4, pp 328-335, 2014.

[5] Prof. Mohammed Abu Zalata Dr. Ghazi. M. Qaryouti , Dr.Saleh Khawatreh, Prof. Ziad A.A. Alqadi, Optimal Color Image Recognition System (OCIRS), International Journal of Advanced Computer Science and Technology., v 7, issue 1, pp 91-99, 2017.

[6] Eng. Sameh S. Salma Prof. Ziad A. AlQad, Eng. Ahmad S. AlOthma , Eng. Mahmoud O. Alleddaw ,Eng. Mohammad Ali Al-Hiar , Eng. Osama T. Ghaza, Investigation of ANN Used to Solve MLR Problem, IJCSMC, v 8, issue 5, 2019.

[7] Jamil Al-azzeh Belal Ayyoub, Ahmad Sharadqh, Ziad Alqadi, Simulink based RNN models to solve LPM, International Journal of Research in Advanced Engineering and Technology, v 5, issue 1, pp 49-55, 2019.

[8] Sumit Goyal and Gyandera Kumar Goyal, Cascade and Feed forward Back propagation Artificial Neural Network Models For Prediction of Sensory Quality of Instant Coffee Flavored Sterilized Drink, Canadian Journal on Artificial Intelligence, Machine Learning and Pattern Recognition Vol. 2, No. 6, 2011.

[9] Sumit Goyal and G.K.Goyal,"Radial basis artificial neural network computer engineering approach for predicting shelf life of brown milk cakes decorated with almonds", International Journal of Latest Trends in Computing, vol.2, no.3, pp. 434-438,2011.

[10] Scott E. Fahlman and Christian Lebiere, the Cascade Correlation Learning Architecture. D. S. Touretzky (ed.), Advances in Neural Information Processing Systems 2, pages 524–532, 1990.

[11] Thomas R. Schultz, Francois Rivest, Laszl ´ o Egri, ´ Jean-Philippe Thivierge, and Fred´ eric Dandurand, ´ Could Knowledge-based Neural Learning Be Useful in Developmental Robotics?. The Case of KBCC. International Journal of Humanoid Robotics Vol. 4, No. 2, pages 245–279, 2007.

[12] Chong, S., Rui, S., Jie, L.: Temperature drift modeling of MEMS gyroscope based on Genetic-Elman neural network. In : Mechanical Systems and Signal Processing, 2016, vol. 72, p. 897-905.

[13] Yongchun, L.: Application of Elman neural network in short-term load forecasting. In : Artificial Intelligence and Computational Intelligence (AICI), 2010 International Conference on, IEEE, 2010. p. 141-144.

[14] Storn, R., Price, K.: Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. In: Journal of global optimization, 1997, vol. 11, no 4, p. 341-359.

[15] Nonlinear autoregressive neural network for estimation soil temperature: a comparison of different optimization neural network algorithms Omaima N. A. AL-Allaf #1 and Shahlla A. AbdAlKader Special Issue of ICIT 2011 Conference.

[16] PlantdiseasedetectionandclassificationusingimageprocessingandartificialneuralnetworksMr.SanjayMirchandani,MihirPendse,PrathameshRane,AshwiniVedula, International Research Journal of Engineering and Technology(IRJET)e-ISSN: 2395-0056Volume: 05 Issue: 06 June2018www.irjet.net.

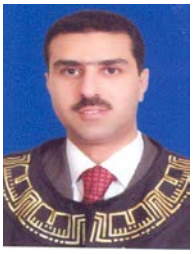
[17] Power System Planning and Operation Using Artificial Neural NetworksAnkita Shrivastava1and Arti Bhandakkar21PG Scholar, Shriram Institute ofTechnology, Jabalpur2Associate Professor, Shriram Institute ofTechnology, Jabalpur.

[18] International Journal of Computer Applications (0975 – 8887)Volume 161 –No 7, March 201741Improved the Prediction of Clinical Data Accuracy using RBF Neural Network ModelDineshKumarSahuSri Satya Sai College of EngineeringBhopal M.P., IndiaRavishKumarSri Satya Sai College of EngineeringBhopal M.P., IndiaAnilRajputCSA, Govt. P.G. CollegeSehore M.P., India.

[19] Elman neural networks in model predictive control

[20] Proceedings 23rd European Conference on Modelling and Simulation ©ECMS Javier Otamendi, Andrzej Bargiela,José Luis Montes, Luis Miguel Doncel Pedrera (Editors)ISBN: 978-0-9553018-8-9 / ISBN: 978-0-9553018-9-6.

[21] International journal of systems applications, engineering & development, Function Approximation Using Artificial Neural Networks, zarita zainuddin & ong pauline.



Akram Aref Mustafa was born in Kuwait, on Jun 23, 1973.He received B.Sc. degree and M.Sc. degree in computer engineering from The National Technical University of Ukraine in 1997 and 1999, respectively. He received the Ph.D. degree in computer engineering from The National Technical University of Ukraine in 2003. From 2001 to 2002, he was an instructor of computer at Yarmouk University, and from 2004 to 2011, he was an Assistant

Professor of computer science department at Al al-Bayt University, Jordan and since November 2011, he has been an Associate Professor of computer science department at Al al-Bayt University, Jordan.
Dr. Mustafa has served as a reviewer for several international journals and conferences; he has published more than 40 scientific papers in computer science field. His research interests include network, image processing, network security, and programming.