Comparison of Learning Parameters Using R-Prop Technique in The Neural Network Training For User Authentication System

A. Joseph, T.L. Jee, D.B.L. Bong, L.C.Kho, D.D.A .Mat

***Abstract*— User authentication is very important in a networked smart environment since that security has been an important issue and concern in the smart home system. Currently, a vast majority of systems use passwords as the means of authentication because it is easier to implement. Although the password-based is very popular in the user authentication, but there are some drawbacks. Thus, in this paper, R-prop technique was embedded in the Multilayer Perceptrons neural network to investigate the performance of network training. The learning parameters in the neural network training for authentication system using R-Prop technique were compared. The comparisons were based on training time, number of epochs and Mean Square Error (MSE) using different transfer functions. Three combinations of transfer function for hidden and output were carried out. First, hidden layer used logsig and output layer used Tansig (combination A). Secondly, Tansig was applied to hidden layer and Purelin was applied to output layer (Combination B). Finally, Logsig and Purelin were applied to hidden layer and output layer respectively (Combination C). Besides, the performances of different number of training sets were also compared for Combination B with 250 hidden neurons. From the results obtained, it is observed that implementing Combination B for 200 hidden neurons yield optimum results. Apart from that, as the training sets increased, the MSE, training time and also number of epochs increased proportionally.**

***Index Terms*— Learning Parameter, Neural Network, R-Prop, User Authentication**

I. INTRODUCTION

Nowadays, there are a lot of method proposed for identifying the login user in the market, likewise, fingerprint, voice recognition, face detection, CCTV and etc. Indeed these systems are very powerful and secure; however they are not widely used due to the price of the product. Password-based user authentication is inexpensive and affordable. Thus, in this paper, the password-based user authentication using neural network was implemented. In the neural network training part, local adaptive technique called R-prop technique was embedded in the Multilayer Perceptrons neural network and comparison of the learning parameter in the training phase was done in order to investigate the performance of the network training for the user authentication system in the smart home. The main objective of this paper is to compare the Mean Square Error (MSE), training time and also number of epochs using different transfer functions and performances of different number of training sets for Combination B with 250 hidden neurons.

Manuscript received October 23, 2009. This work was supported in part by Osaka Gas Grant UNIIMAS/15/01-05.08Jld.11(75).

A. Joseph is with the Faculty of Engineering, Universiti Malaysia Sarawak, 94300 Kota Samarahan. Malaysia. (phone: +60-82-583272; fax: +60-82-583410, email: jannie@feng.unimas.my)

T.L.Jee, D.B.L. Bong, L.C.Kho and D.D.A. Mat are with the Faculty of Engineering, Universiti Malaysia Sarawak, 94300 Kota Samarahan. Malaysia.

II. DATA AND PROCESS

The Multilayer Perceptions neural network has been setup and the data collected was used to train the network. In this paper, training set consists of 200 sets of User ID and Password whereas the password is suggested should at least contain 8 characters, one alphanumeric, one mixed case and at least one special character (not A-Z or 0-9) since password quality refers to the entropy of a password and is necessary to ensure the security of the user's accounts. A good password is password that is impossible to guess by other people [1, 3, 7, 8]. After the training process, 85 sets of User ID and Password were used as testing set in order to measure the simulation output. For the testing set, 55 sets were correct User ID and Password while 15 sets were Wrong User ID with correct password and 15 sets were correct User ID with wrong Password. In this paper, all the information input by the user (User ID and Password) were converted into binary form since the neural network can only recognize the numerical data and the range of input and output value for neural network is 0 and 1. The User ID and Password had to be normalized before the network training and every single data was assigned to 7 bits binary code.

III. NEURAL NETWORK IMPLEMENTATION

The Multilayer Perceptions neural network used in the user authentication is the supervised learning method where the training pattern includes the known input and the expected output. Besides, the training provides the network parameters and weight values where the weights values are adjusted by the training patterns [5, 6]. For the local adaptive technique, R-prop was chosen to be embedded into the neural network training phase. The implementing of neural network training algorithms is shown in figure 1.

ISBN: 978-988-17012-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

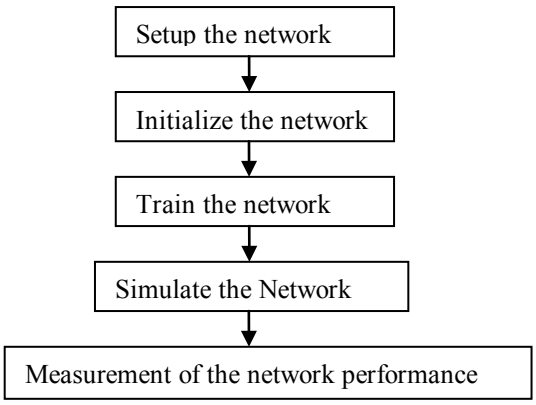

Figure 1: Figure 1 shown the Block diagram of the Neural Network Implementation

From the figure 1, the first step of neural network implementation is to setup the feed-forward back-propagation neural network. Before training the network, the network is initialized in order to returns the neural network net with the weight and bias values updated according to the network initialization function. The network is ready for training once the network weights and biases are initialized. A set of examples of proper network behavior is required for the training process. Simulation of the network output is done after the network had been trained. There are ten simulations were taken for each training process in order to obtain best result since that every training would yield vary simulation result [2]. After the training process, the network was measured by Mean Square Error (MSE) since the network performance shows reliability of the trained network.

IV. RESULTS AND DISCUSSION

In this paper, the performance of training time, number of epochs and also Mean Square Error (MSE) of RPROP learning algorithm were compared base on different transfer functions and different number of training sets for combination B with 250 hidden neurons. Besides, some training parameters of the training algorithm were adjusted in order to obtain the best training network. The appropriate parameters for the algorithm were obtained by using trial and error method.

A. Compare the Performance Using Different Transfer Functions

The overall performance of implementing different transfer functions to the network was observed. 150 and 200 hidden neurons and one hidden layer were applied. The number of training set used to train the network for the comparison was 200 sets. The training time, the number of epoch and MSE was the main concern because training using backpropagation without embedding R-prop technique was slow [1], [3], [4].

Three combinations of transfer function for hidden layer and output layer were carried out. First, hidden layer used Logsig and output layer used Tansig (Combination A). Secondly, Tansig was applied to hidden layer and Purelin was applied to output layer (Combination B). Finally, Logsig and Purelin were applied to hidden and output layer respectively (Combination C).

1. Training Time

The average training time taken to train the 200 sets of data using 150 and 200 hidden neurons were stated in Table 1 and the comparison graph was shown figure 2. The training time was measured in minute and second (min:sec).

Table 1: Training Time with Different Number of Hidden Neurons

Simulation	Transfer Funtion					
	Combination A		Combination B		Combination C	
	150 Hidden Neurons	200 Hidden Neurons	150 Hidden Neurons	200 Hidden Neurons	150 Hidden Neurons	200 Hidden Neurons
1	1:35	1:29	1:57	0:50	2:07	0:55
2	1:30	1:25	2:04	0:50	1:58	0:53
3	1:49	1:20	1:40	0:51	2:11	0:49
4	1:43	1:21	1:52	0:52	2:01	0:49
5	1:41	1:28	1:52	0:51	2:03	0:49
6	1:34	1:26	2:04	0:51	1:50	0:50
7	1:33	1:25	1:54	0:49	2:09	0:57
8	1:42	1:26	2:09	0:49	2:03	0:47
9	1:28	1:18	2:11	0:49	2:04	0:47
10	1:46	1:23	2:14	0:46	2:12	0:53
Average	**1:38**	**1:24**	**1:59**	**0:49**	**2:03**	**0:50**

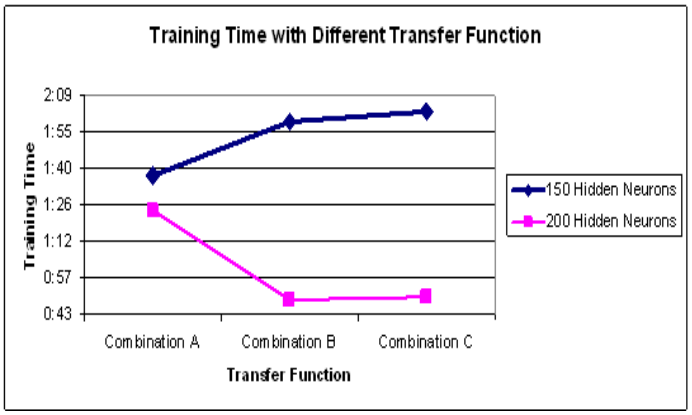

Figure 2: Training Time with Different Number of Hidden Neurons

Table 1 and figure 2 shows the training time of the network with different transfer function. The average time taken to train the network for Combination A was 1 min 38 sec and 1 min 24 sec for using 150 and 200 hidden neurons. Combination B used 1 min 59 sec and 49 sec to train the network for using 150 and 200 hidden neurons respectively. The average time taken by Combination C for using 150 and 200 hidden neurons was 2 min 3 sec and 50 sec respectively.

ISBN: 978-988-17012-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

2. *Number of Epochs*

Number of epochs used to train the network was recorded and shown in table 2 and the average of number of epochs with different transfer function was plotted and shown in figure 3.

Table 2: Number of Epochs with Different Number of Hidden Neurons

Simulation	Transfer Funtion					
	Combination A		Combination B		Combination C	
	150 Hidden Neurons	200 Hidden Neurons	150 Hidden Neurons	200 Hidden Neurons	150 Hidden Neurons	200 Hidden Neurons
1	1067	919	1318	518	1468	577
2	1059	878	1412	504	1420	562
3	1181	817	1148	502	1531	529
4	1169	824	1254	517	1406	530
5	1146	890	1262	505	1441	528
6	1126	889	1417	507	1286	537
7	1115	901	1293	486	1486	602
8	1215	898	1447	486	1421	510
9	1004	825	1467	482	1443	516
10	1210	877	1478	459	1504	583
Average	**1129.2**	**871.8**	**1349.6**	**496.6**	**1440.6**	**547.4**

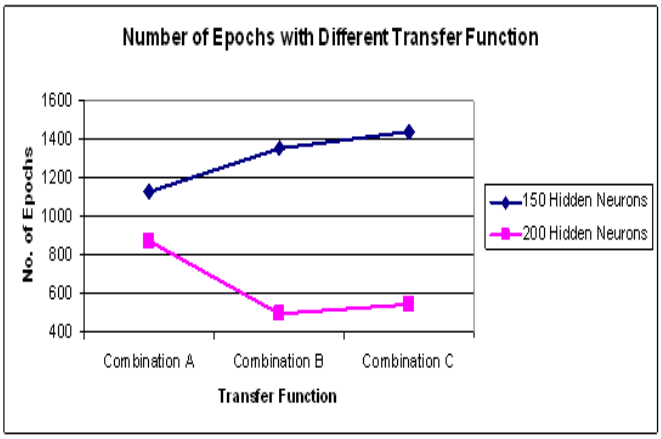

Figure 3: Number of Epochs with Different Number of Hidden Neurons

The training time of the network was dependent to the number of epochs during the training. Hence, the number of epochs had the same scenario as the training time. Combination A stopped training at an average of 1129.2 and 871.8 epochs by using 150 and 200 hidden neurons respectively. An average of 1349.6 and 496.6 epochs were taken to train the network for Combination B using 150 and 200 hidden neurons respectively. Combination C used 1440.6 and 547.4 epochs for training by implementing 150 and 200 hidden neurons respectively.

3. *Mean Square Error (MSE)*

The MSE results using different transfer function in hidden layer and output layer was recorded. Data was shown in table 3 and figure 4.

Table 3: MSE with Different Number of Hidden Neurons

Simulation	Transfer Funtion					
	Combination A		Combination B		Combination C	
	150 Hidden Neurons	200 Hidden Neurons	150 Hidden Neurons	200 Hidden Neurons	150 Hidden Neurons	200 Hidden Neurons
1	0.004800	0.003730	0.001018	0.000975	0.001005	0.001066
2	0.004090	0.003200	0.000875	0.000985	0.001045	0.000951
3	0.005690	0.003820	0.000993	0.000954	0.000983	0.000945
4	0.005330	0.003460	0.001044	0.001017	0.000978	0.00106
5	0.005240	0.003020	0.001028	0.001008	0.000949	0.000982
6	0.004180	0.002580	0.000997	0.000988	0.000996	0.001059
7	0.005240	0.003730	0.001011	0.000989	0.001003	0.000945
8	0.005240	0.003110	0.000956	0.001031	0.000995	0.001056
9	0.004440	0.004180	0.000976	0.001022	0.000976	0.000936
10	0.004440	0.004350	0.000905	0.00096	0.000974	0.001119
Average	**0.004869**	**0.003518**	**0.000980**	**0.000993**	**0.000990**	**0.001012**

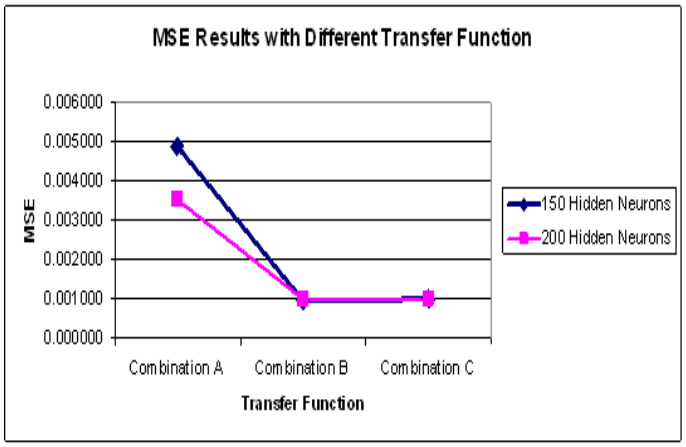

Figure 4: MSE Results with Different Transfer Function

Comparing the MSE results with different transfer function used in hidden and output layer, it was observed that by implementing Combination A in the network gave highest average MSE which was 0.004869 and 0.003518 for 150 hidden neurons and 200 hidden neurons respectively. Combination B had an average MSE 0.000980 for 150 hidden neurons and 0.000993 for 200 hidden neurons. The MSE results for Combination C was 0.000990 (150 hidden neurons) and 0.001012 (200 hidden neurons).

From the results obtained, Combination B gave the best average MSE result, Combination A required less time and less number of epochs to do the training for 150 hidden neurons. However, for 200 hidden neurons, Combination B required less time and number of epochs to train the network. Furthermore, for a security system, the accuracy for the training was the main concern. Hence, comparing the MSE results, training time and number of epochs of the training, it was observed that by using Tansig and Purelin as the transfer

ISBN: 978-988-17012-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

function for hidden and output layer respectively provide the optimum training.

B. Compare the Performance of Different Number of Training Sets

From the results showing in the section A for combination A, B and C, it is observed that Combination B obtained the optimum results for 200 hidden neurons. The transfer function Tansig was used in the hidden layer and Purelin was used in the output layer for Combination B. However, the number of hidden neurons applied was increased to 250 this time. The number of training sets used for the experiment was 50, 100, 150 and 200 sets respectively. The results of MSE from the simulation were shown in table 4 and figure 5.

Table 4: MSE Results with Different Number of Training Sets

Simulation	No. of Training Sets			
	50	100	150	200
1	0.000941	0.000973	0.000976	0.000975
2	0.000848	0.000963	0.000983	0.000944
3	0.000958	0.000961	0.000993	0.001003
4	0.000986	0.000967	0.000987	0.000968
5	0.000973	0.001000	0.000975	0.001026
6	0.000989	0.000983	0.000976	0.000997
7	0.000978	0.000980	0.000995	0.001000
8	0.000973	0.000985	0.000988	0.000959
9	0.000978	0.000967	0.000995	0.000978
10	0.000933	0.000976	0.000993	0.000941
Average	**0.000956**	**0.000975**	**0.000986**	**0.000979**

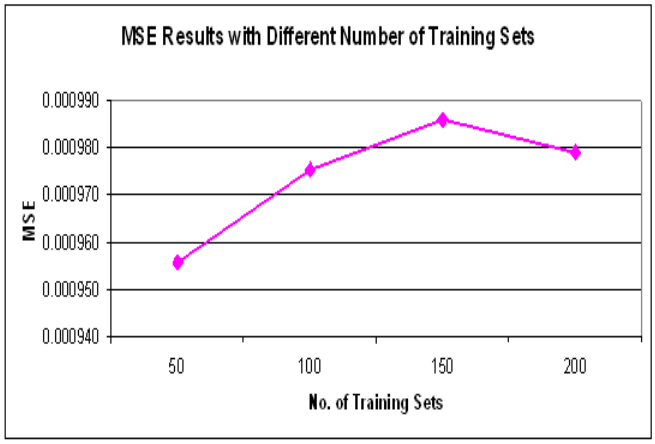

Figure 5: MSE Results with Different Number of Training Sets

The data in table 4 and figure 5 showed the MSE versus different number of training sets used. It was observed that the number of training sets does not affect much on the MSE results. The highest MSE from 150 sets and lowest MSE from 50 sets only have a difference of 0.00003. This was due to the testing sets were extracted from the training sets, hence, as long as the training was good enough, the simulated output would be the identical to the target trained. MSE of 0.001 would still be able to provide the identical simulated output.

Apart from that, simulation also done to compare the training time with different number of training sets and the simulation results were shows in table 5 and figure 6.

Table 5: Training Time with Different Number of Training Sets

Simulation	No. of Training Sets			
	50	100	150	200
1	0:01	0:05	0:12	0:34
2	0:02	0:05	0:13	0:30
3	0:02	0:05	0:12	0:30
4	0:01	0:05	0:12	0:30
5	0:02	0:05	0:12	0:29
6	0:01	0:05	0:12	0:31
7	0:02	0:04	0:12	0:30
8	0:01	0:05	0:12	0:30
9	0:02	0:05	0:13	0:29
10	0:01	0:05	0:13	0:28
Average	**0:01**	**0:05**	**0:12**	**0:30**

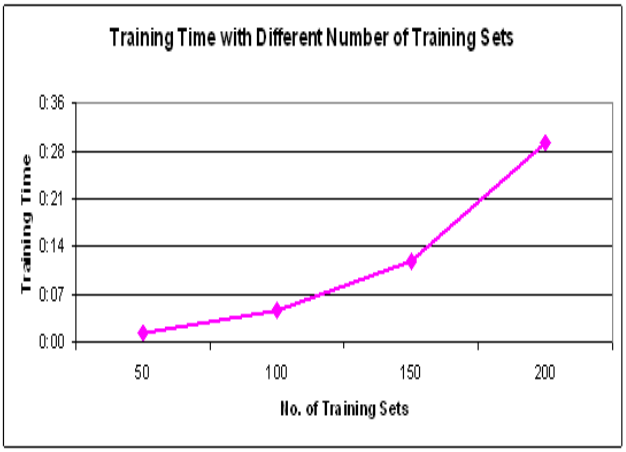

Figure 6: Training Time with Different Number of Training Sets

From the table 5 and figure 6, it is observed that the training time increased as the number of training sets was increased. Besides, to train 50 sets of User ID and password only required 1 sec, 100 sets only required 5 sec, 150 sets required 12 sec and 200 sets used 30 sec to train the network. All the training time from 50 sets to 200 sets was acceptable.

Finally, the simulation was done to compare the number of epochs versus the number of training sets and the simulation results was shows in table 6 and figure 7.

ISBN: 978-988-17012-8-2
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Table 6: Number of Epochs with Different Number of Training Sets

Simulation	No. of Training Sets			
	50	100	150	200
1	35	72	144	297
2	36	69	150	296
3	34	72	143	292
4	34	73	146	289
5	35	71	140	283
6	34	74	144	297
7	36	68	145	290
8	33	74	143	286
9	35	73	149	278
10	33	70	156	276
Average	**34.5**	**71.6**	**146**	**288.4**

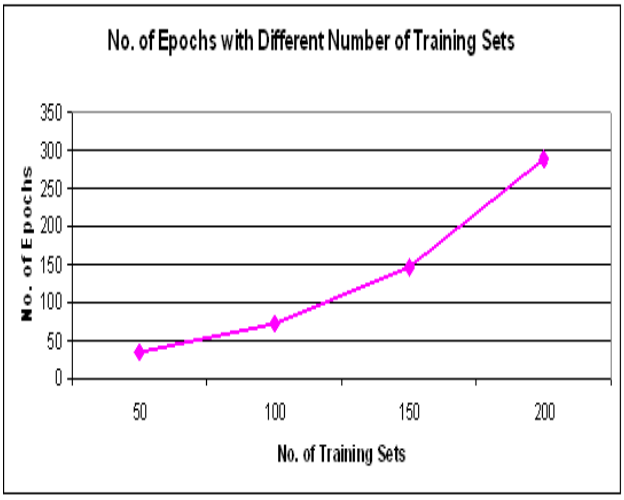

Figure 7: Number of Epochs with Different Number of Training Sets

The Number of Epochs increased as the number of training sets was increased. By using 50 sets of training sets, the number of epochs was 34.5. 100 and 150 training sets had 71.6 and 146 training epochs respectively. Lastly, 200 sets of training sets have 288.4 epochs during the training. The results were shown in table 6 and figure 7.

V. Conclusion

As a conclusion, the objective has been achieved. A local adaptive learning algorithm, R-prop, has been embedded to train the network. Besides, from the comparison results obtained in the section VI, it is observed that, implementing Tansig in the hidden layer and Pureline in the output layer (combination B) with 200 hidden neurons yield the best results for training time and the number of epochs which is 0.49 sec and 496.6 epochs respectively. Apart from that, Comparison results for Training Time, Number of Epochs and MSE based on different number of training sets shows that as the number of training sets increased, the training time and number of epochs to train the network increased proportionally. Furthermore, embedding an R-prop technique in the neural network training accelerates the training time.

ACKNOWLEDGEMENT

The work reported in this paper is supported in part by the Universiti Malaysia Sarawak, Department of Electronic under Osaka Gas Grant UNIMAS/15/01-05.08Jld.11(75).

REFERENCES

[1] I-C. Lin, H-H. Ou, M-S. Hwang, *A User Authentication System using backpropagation Network,* Neural Comput & Applic, 2005.

[2] A. Pavelka, A. Proch'azka, "*Algorithms for Initalization of Neural Network Weights*", Institute of Chemical technology, department of Computing and Control Engineering

[3] S-Z Reyhani, M-Mahdavi, "User Authentication Using Neural Network in Smart Home Networks", *International Journal of Smart Home*, Vol. 1, July 2007, pp147+.

[4] S.Wang, H.Wang, " Password Authentication Using Hopfield Neural Networks", *IEEE Transactions on Systems, man, and Cybernertics*, Vol 38, No 2, March 2008.

[5] R. Berteig, Neuralyst™ User Guide, California: Cheshire Enginnering Corporation 2003

[6] K. Mehrotra, C.K.Mohan, S. Ranka, Elments of *Artificial Neural Networks*, Cambridge: The Mit Press, 1997

[7] U-Manber, *A Simple Scheme to Make Passwords Based on One way Functions Much Harder to Crack*, 2000

[8] M. Curphey, A Guide to Building Secure Web Applications, The Open Web Application Security Project (OWASP), Boston, USA, 2002