AN ENHANCED VERSION OF DELTA-BAR-DELTA ALGORITHM

Mohammed A. Otair
Department of Computer Information System
Jordan University of Science and Technology
Irbed, Jordan
Email Address: otair@just.edu.jo

Walid A. Salameh
Department of Computer Science-RSS
Princess Summaya University for Science and Technology
11941 Al-Jubaiha, Amman, Jordan
Email Address: walid@psut.edu.jo

ABSTRACT

Delta-Bar-Delta (DBD) [7] is an alternative to Back-propagation, which is sometimes more efficient, although it can be more disposed to stick in local minima than Backpropagation. This paper presents the enhanced version of delta-bar-delta (EVDBD) through applying the Delta-Bar-Delta on Optical Backpropagation OBP [10,11 and 13]to adapt its weights rather than standard Backpropagation BP [22] .The feasibility of the proposed algorithm is shown through experiments on a three training problems: Xor, encoder, and optical character recognition with different architectures. A comparative study has been done to solve these problems by using different algorithms, and the performance of the EVDBD is shown.

KEY WORDS

Neural Networks, Backpropagation, Delta-Bar-Delta, Optical Backpropagation .

1. INTRODUCTION

The Backpropagation algorithm [22] is perhaps the most widely used supervised training algorithm for multi-layered feedforward neural networks [20, and 23]. However, in many cases, the standard Backpropagation takes long time to converge [2,3,4, and 6]. The Delta-Bar-Delta network utilizes the same architecture as a back-propagation network. The difference of delta bar delta lies in its unique algorithmic method of learning. Delta-Bar-Delta was developed to improve the convergence rate of standard feedforward, back-propagation networks.

A common approach to avoid slow convergence in the flat directions and oscillations in the steep directions, as well as to exploit the parallelism inherent by the BP algorithm, consists of using different learning rates for each direction in weight space [7, 15, and 21]. However, attempts to find a proper learning rate for each weight usually result in a trade-off between the convergence speed and the stability of the training algorithm. For example, the delta-bar-delta method [7] or the QuickProp method [5] introduce additional highly problem dependent heuristic coefficients to alleviate the stability problem.

Delta-Bar-Delta algorithm uses the information derived from previous weights to determine how large an update can be made without diverging. This typically uses some form of historical information about specific weight's gradient.

In order to speed up the training process, many researches increase each weight update depending on the previous weight update. This effectively increases the learning rate [16]. In [21] the Incremental Delta-Bar-Delta IDBD algorithm is developed for the learning of appropriate biases based on previous learning experience.

The extended-delta-bar-delta (EDBD)[15] algorithm applies heuristic adjustment of the *momentum* term in the DBD-based networks. The momentum is a term, which is added to the weight change and is proportional to the previous weight change. This heuristic is designed to reinforce positive learning trends and dampen the oscillations.

This paper presents the enhanced version of Delta-Bar-Delta with OBP. It can overcome some of the problems associated with standard backpropagation and Delta-Bar-Delta. The experimental results prove that speed of convergence can be improved using EVDBD.

The rest of the paper is organized as follows: The Delta-Bar-Delta is introduced in section 2. Section 3 presents the proposed algorithm. Experimental results and discussion are given in section 4. Section 5 concludes the paper.

2. DELTA BAR DELTA

The Delta-Bar-Delta (DBD) attempts to increase the speed of convergence by applying heuristics based upon the previous values of the gradients for inferring the curvature of the local error surface.
The delta bar delta paradigm uses a learning method where each weight has its own self-adapting coefficient. It also does not use the momentum factor of the back-

propagation networks [17]. The remaining operations of the network, such as feedforward recall, are same to the normal back-propagation networks [8, and 9]. Delta-Bar-Delta is a heuristic approach in training neural networks, because the past error values can be used to infer future calculated error values. Delta-Bar-Delta implements four heuristics regarding gradient decent:

- Every weight should have its own individual learning rate.
- Every individual learning rate should adjust over time.
- If the error derivative has the same sign for several consecutive steps, then increase the learning rate.
- When the sign changes alternatively over a number of steps, then decrease the learning rate: clearly the large rate causes oscillations.

2.1 Technical Details

Weights are updated using the same formula as in Backpropagation, except that momentum is not used, and each weight has its own time-dependent learning rate.

All Learning rates are initially set to the same starting value; subsequently, they are adapted on each epoch using the formula below.

The delta value for each neuron is calculated as:

$$\Delta_{ij} = -\delta_j * x_i \quad \textbf{(1)}$$

Where δ_j is the error at a single output unit is defined as:

$$\delta_j = (Y - O) \bullet f'(\sum w_{ij} x_i) \quad \textbf{(2)}$$

Where Y is the desired output, O is the actual output, Wij are the weights from hidden to the output units and Xi are the input for each output unit.

The bar-delta value for each unit is calculated as:

$$\overline{\Delta}_{ij}(n) = (1-\beta) * \Delta_{ij}(n) + \beta * \overline{\Delta}_{ij}(n-1) \quad \textbf{(3)}$$

$\overline{\Delta}_{ij}(n)$ is the derivative of the error surface, β is the smoothing constant.

The learning rate of each weight is updated using:

$$\Delta\eta_{ij}(n+1) = \begin{cases} \eta_{ij}(n) + k & \text{, If } \overline{\Delta}_{ij}(n-1) * \Delta_{ij}(n) > 0 \quad \textbf{(4.a)} \\ (1-\gamma) * \eta_{ij}(n) & \text{, If } \overline{\Delta}_{ij}(n-1) * \Delta_{ij}(n) < 0 \quad \textbf{(4.b)} \\ \eta_{ij}(n) & \text{, Otherwise} \quad \textbf{(4.c)} \end{cases}$$

Where η is the learning rate, γ is the exponential decay factor and *K* is the linear increment factor.

3. ENHANCED VERSION OF DELTA-BAR-DELTA (EVDBD)

An Enhanced version of Delta-Bar-Delta (EVDBD) algorithm is an extension of the Delta-Bar-Delta algorithm as a natural outgrowth from Jacob's work[3], EVDBD is the same as Delta-Bar-Delta which introduced by Jacobs as outlined in section 2 except that the proposed algorithm uses an Optical Backpropagation OBP rather than BP network.

In [10,11,12,13, and 14], it has been proved that the OBP algorithm improves the performance of the *BP* algorithm. The convergence speed of the training process can be improved significantly by *OBP* through adjusting the error, which is transmitted backward from the output layer to each unit in the hidden layer. So, if the Delta-Bar-Delta applies on OBP, then the rate of convergence can be improved with EVDBD algorithm.

Optical Backpropagation (*OBP*) applies a non-linear function on the error from each output unit before applying the backpropagation phase, using this formula:

$$New\delta_j = (1 + e^{(Y-O)^2}) \bullet f'(\sum w_{ij} x_i) \quad \textbf{(5)}$$

, if *(Y– O) > zero.*

$$New\delta_j = -(1 + e^{(Y-O)^2}) \bullet f'(\sum w_{ij} x_i) \quad \textbf{(6)}$$

, if (Y– O) < zero.

$$New\delta_j = zero$$

, if (Y– O) = zero. **(7)**

The $New\delta_j$ will propagate backward to update the output-layer weights and the hidden-layer weights. (i.e. The deltas of the output layer change, but all other equations of *BP* remain unchanged). This *Newδj* will minimize the errors of each output unit, and the weights on certain units change on large steps from their starting values.

4. EXPERIMENTAL EVALUATION

In this section, seven training algorithms are implemented on a variety of training problems, which are: Backpropagation (BP) [22], Backpropagation with momentum (BPM) [22], QuickProp (QP) [5], Delta-Bar-Delta (DBD) [7], Optical Backpropagation (OBP) [10] , Optical Backpropagation with momentum (OBPM) [13] , and Enhanced Version of Delta-Bar-Delta (EVDBD). Most algorithms have been experimented the following neural network problems:

4.1 XOR problem

The implement of the EVDBD algorithm to solve the XOR problem is very important because the XOR problem requires hidden layers and many other difficult problems involve an XOR as a subproblem.

The XOR problem will be solved using neural network which consists of two input units, two hidden units, and single output unit, with biases for hidden unit and the output unit, without direct connection from input to the output layers, (this network is labeled as 2-2-1). To train this network all initial weights will start as shown in figure 1 for all training processes.

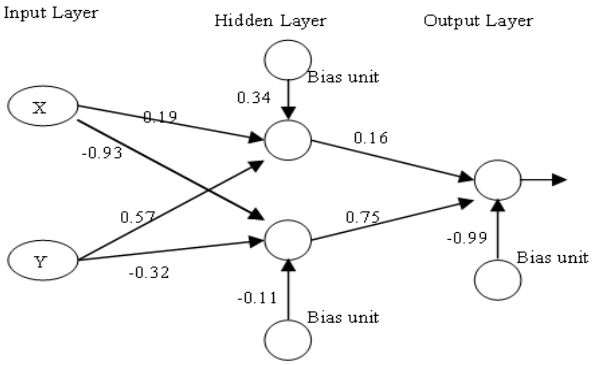


FIGURE 1: *Initial Weights for XOR (2-2-1) problem*

In this experiment, the training process is continued until reaching a mean square error (MSE) less than or equals to 0.001. Different learning rates were used ranged from (0.1 to 1).

In BPM and OBPM the value 0.5 is used for momentum factor, while the DBD and EVDBD use the following parameters (β=0.6, γ=0.001, k=0.001). In addition, in the DBD and EVDBD algorithms there is no constant value for the learning rate, instead the learning rate is initialized with values for each training process as shown in the first column in table 1, and then these learning rates values are adapted through the training epochs. Table 1 shows the results for each training processes using all algorithms in term of number of epochs (this assumption will be associated with all experiments).

TABLE 1: *Solve XOR (2-2-1) problem using Seven Algorithms*

LR	BP	BPM 0.5	QP	DBD	OBP	EVDBD	OBPM 0.5
0.1	21304	13702	8022	981	1640	457	862
0.2	10339	6850	2995	956	911	450	463
0.3	6772	4566	2008	930	713	456	330
0.4	5018	3423	1492	905	656	472	263
0.5	3980	2737	1210	881	659	484	224
0.6	3295	2280	985	857	671	479	199
0.7	2810	1953	870	834	651	454	181
0.8	2448	1708	868	812	593	416	169
0.9	2169	1518	1025	791	521	373	160
1	1947	1365	990	770	443	326	153

Figure 2 represents the previous table. As you can see, the OBPM (with momentum of 0.5) and EVDBD are faster than the OBP. Then, comes the DBD with a close number of epochs to the OBP.

According to the last three algorithms which took a larger number of epochs, they are the QP, BPM and BP respectively.

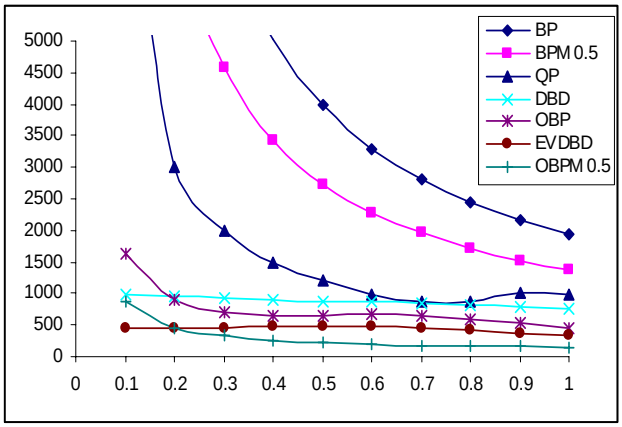


FIGURE 2: *Solve XOR (2-2-1) problem using Seven Algorithms*

4.2 Solve XOR problem (4 bits) using DBD , and EVDBD

The second problem to be described is the XOR problem with 4-bits (*4-16-1*). The network consists of 4 units in the input layer, and only 1 unit in the output layer, and a hidden layer of 16 units, respectively.

Table 2 represents the training process for this problem using DBD, and EVDBD, with different initial values for learning rate between 0.1 and 0.9.

TABLE 2: *Solve XOR problem (4 bits) using EVDBD and DBD*

η	DBD 4-bit	EVDBD 4-bit
0.1	128	56
0.2	120	45
0.3	110	33
0.4	103	27
0.5	98	24
0.6	83	16
0.7	75	13
0.8	66	9
0.9	64	8

As seen in the previous table, the EVDBD algorithm needs less number of epochs in comparing with DBD especially when using a large value for learning rate.

4.3 Encoder Problem

Encoder problem is a feed-forward neural network with *N* input units, *M* hidden unites, and *N* output units (i.e. *N-M-N* networks). Training these networks can be very challenging when *M*<*N* [18]. They are trained so their output values (approximately) match their input values on a training set.

Peter Anderson [1] proposed a new approach(called *Training Wheel)* to train encoder feed-forward neural networks and apply it on many classes of problems such as *N-4-N*. Anderson's approach is to initially train the network with a related, relatively easy-to-learn problem, and then gradually replace the training set with harder problems, until the network learns the problem he originally intended to solve. In several cases, this method allowed us to train networks that otherwise fail to train.

4.3.1 16-M-16 Encoder Problem

A 16-*M-16* problem is to compress a signal of 16 values into one of *M* values. This network is useful for feature extraction. Each network consists of 16 units in both the input layer and output layer, and a hidden layer of *M* (6 to 11) units. In this experiment, the input patterns are equal to the target patterns such as the following sample pattern formats: Note, the size of gab between two ones ≥6 (as shown in table 3).

TABLE 3: Four of the forty training patterns for 16-M-16 network

1 0 0 0 0 0 **1** 0 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 **1** 0 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 **1** 0 0 0 0 0 0 0

1 0 0 0 0 0 0 0 0 **1** 0 0 0 0 0 0

The following table shows the training process of the network 16-M-16, various values of *M* have been applied from 6 to 11. It is clear that the more number of units in the hidden layer, causes the network will be able to escape from local minima, so acceleration the training process.

TABLE 4: *Solve 16-M-16 Encoder using EVDBD with Training Wheel's technique*

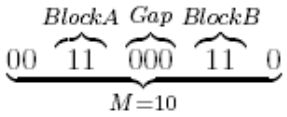
η \ M	11	10	9	8	7	6
0.1	124	142	180	∞	210	327
0.2	105	122	∞	168	202	226
0.3	93	110	∞	120	95	121
0.4	77	90	98	96	122	∞
0.5	64	∞	83	95	∞	152
0.6	61	61	62	72	94	∞
0.7	57	52	54	∞	84	210
0.8	53	51	57	56	74	95
0.9	61	63	55	70	72	∞

From the previous table it can notice that the best values achieved when a hidden layer size equals to 11. Meaning that the large hidden layer size helps to generalize and accelerate the training process.

4.3.2 M-4-M Encoder Problem (Double dots)

This experiment has tested *M-4-M* network to compress a signal of *M* vales into 4 values where *M*=7, 9, 10, 16, and 28 [1]. Each network consists of *M* units in both the input layer and output layer, and a hidden layer of 4 units. In this experiment, the input patterns are equal to the target patterns such as the following sample pattern formats:

TABLE 5: Five of the ten training patterns for 10-4-10 network where gab size = 3 and block size =2



1	1	0	0	0	1	1	0	0	0
0	1	1	0	0	0	1	1	0	0
0	0	1	1	0	0	0	1	1	0
0	0	0	1	1	0	0	0	1	1
1	0	0	0	1	1	0	0	0	1

This experiment has examined M-4-M through Training Wheel's technique. There are more than one architecture were used. Table 6 shows that there are many cases have been tested (for example, in the third column the architecture in this experiment is 9-4-9, the gap between each two ones is 2 and the size of each block is 1).

TABLE 6: *Solve M-4-M Encoder using EVDBD with Training Wheel's technique*

N	7	9	10		16			28
Gap	1	2	3		5			10
Block Size \ η	1	1	2	1	3	2	1	4
0.1	56	180	44	110	96	1123	154	309
0.2	54	182	33	87	67	1072	112	216
0.3	49	112	21	81	63	682	93	203
0.4	42	101	24	73	61	556	112	112
0.5	31	95	27	66	54	442	105	105
0.6	42	94	25	54	56	413	108	84
0.7	43	84	27	71	54	277	106	82
0.8	45	105	33	78	64	457	124	73
0.9	57	121	38	90	91	545	170	85

From the previous table, it can noticeable that the minimum number of epochs was when the architecture

10-4-10 and the value of the gap =3 and the block size is 2 with learning rate equals to 0.3. In addition, the number of epochs with all learning rates in this case is less than all other cases.

In addition, it can be realized that using a medium size of the hidden layer helps in accelerating the training process. Meaning that using a large size of the hidden layer may slow down the training process or may leads the network to overfitting or even falling into local minima [20].

To compare DBD and EVDBD, the best results using EVDBD have been taken from table 6, while the best results for the same experiment using DBD taken from [14]. Take into consideration that using the same parameters with the two algorithms. Table 7 shows this comparison.

TABLE 7: *Comparing between EVDBD and DBD to Solve M-4-M Encoder Problem with Training Wheel's technique*

M-4-M	Gap	Block	DBD	EVDBD
7	1	1	366	33
9	2	1	359	85
		2	450	22
10	3	1	465	56
		3	331	56
		2	363	297
16	5	1	464	96
28	10	4	12859	76

It can see from the previous table that there are great differences between EVDBD and DBD, which surely approaches towards the EVDBD with respect to the convergence time. In addition to that, the ability of the EVDBD to deal with large size of gap and block (e.g. using a gap=10, block =4 with the network architecture of 28-4-28).

4.4. Optical Character Recognition (OCR) Problem

In this experiment, five neural networks are developed and trained to recognize handwritten characters. Two algorithms were tested: DBD and *EVDBD*.

4.4.1 48-8-4 Neural Network to Solve OCR Problem

4.4.1.1 Methodology

To study the performance of the EVDBD and DBD, the neural networks were applied on handwritten English character recognition. Binary images of the characters were represented with bipolar values [-1, 1], and given to the input layer[19]. As shown in figure 3, ten images are used in the training set an *8X6* binary image for the neural network with 48-8-4 architecture.

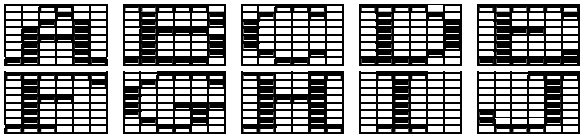


FIGURE 3: *The Training set of OCR (48-8-4 neural Network)*

Binary vectors of size 4 represent the output values.(e.g. character *A* is represented with zero (0000), while *J* is represented with 9 (1001)). Each character image is mapped to its corresponding binary code. To train the network for larger set of characters a large output vector can be used (as will be in sections 4.4.5.2 and 4.4.5.3).The size of the hidden layer for this network is 8 producing *48-8-4* network. Small values for the learning rates were used to avoid local minima, the value ranges from 0.1 to 0.4. In addition, several sets with different random initial weights between –0.5 to +0.5 were used in each training process with various values for the learning rate. The training processes were terminated when the *MSE* (Mean Square Error) is less than or equal to 0.0001(for all the following experiments).

4.4.1.2 Solve OCR problem (48-8-4) Using DBD and EVDBD:

Table 8 shows training summary for the *48-8-4* neural network using the. 20 experiment runs of each learning rate were made with random initial weights setting over the interval [-0.5, 0.5].

TABLE 8: *Solve OCR problem (48-8-4) Using DBD* and EVDBD

η	Algo.	Max	Min
0.1	DBD	6900	5274
	EVDBD	423	380
0.2	DBD	7710	2533
	EVDBD	327	195
0.3	DBD	4028	1706
	EVDBD	355	145

From the previous table, it can see the great difference between the two algorithms which greatly approaches towards EVDBD. Through the EVDBD a minimum number of the epochs reached to 145 while in DBD was 1706 and that was with learning rate of 0.3.

4.4.2 96-16-4 Neural Network to Solve OCR Problem

In this experiment a larger network has been used by using a larger size for each image which each one of them represents one of the capital letters from A-J. Each character represents with 12 rows and 8 columns which means that the number of pixels which represent each character is 96; this number represents the size of the

input layer. 16 units were used to the size of the hidden layer, and the size of the output layer is 4 for all experiments.

In this experiment, the network was trained several times through several learning rates as in table 9. The aim of this test is to compare the algorithms EVDBD and DBD through a network that has a larger architecture from those that were used in the previous section.

TABLE 9: *Solve OCR problem (96-16-4) Using DBD* and EVDBD

η	DBD	EVDBD
0.1	83	472
0.2	31	279
0.3	25	144
0.4	23	191

From the previous table, it can see that EVDBD is still keeping its qualities in speeding up the training process especially with using a larger learning rate. The focus should always be through the number of epochs that gets from each algorithm.

4.4.3 Solve OCR problem (400-L-4) Using DBD and EVDBD

In this test, the architecture of the network was 400-L-4, where L=20, 40 and 60. The aim of this network is to recognize the characters from A-J. Each letter have been represented by 20 rows and 20 columns which makes the input layer size equals to 400. The aim of this test is to know the effect of the size of the input layer on the performance of the proposed algorithm EVDBD, and the effect of maximizing the hidden layer size on speeding up training using this algorithm. Note the learning rate that is used in the remaining experiments is equals to 0.1. The following table shows the results of this test .

TABLE 10: *Solve OCR problem (400-L-4) Using DBD* and EVDBD

Algo. / HLS	DBD	EVDBD
20	139	75
40	111	51
60	107	49

HLS: refers to the Hidden Layer size.

It is clear that the minimum number of epochs was when the size of hidden layer equals to 60 especially when use larger learning rate which indicates that maximizing the size of the hidden layer and the learning rate helps in speeding up the training process. The size of the hidden layer may be maximizing just to a certain size, that is because the network could not generalize(e.g. when used larger than 60 units for the hidden layer size).

4.4.4. Solve OCR problem (900-L-4) Using *DBD* and EVDBD

This network is like other networks were built to recognize the characters from A-J. But here, each character have been represented by 30 rows and 30 columns, which makes the size of the input layer equals to 900 units. Various values for the size of hidden layer have been used such as 25, 50, 75 and 100 units. The training process for this network has given in table 11:

TABLE 11: *Solve OCR problem (900-L-4) Using DBD* and EVDBD

Algo. / HLS	DBD	EVDBD
25	112	81
50	67	45
75	53	30
100	55	31

This experiment has also assured that EVDBD has the ability to generalize even with maximizing the input layer size. From table 11 it can notice that the hidden layer size maximized until reaches to a point where more maximization has no positive effect (may be overfitting). It even might expose the network to a prevention from generalize or memorization (for example, it can see from the previous table that the size of hidden layer equals to 75 and 100 leads to the same performance approximately).

4.4.5. Solve OCR Problem (10000-L-M neural network)

From the previous experiments, different sizes have been examined for input layer, but in this test the size of the input layer has been maximized to be compatible with real problems. Each character has been represented with 100 rows and 100 columns, so that the size of the input layer becomes very large which is 10000 units.

4.4.5.1 Solve OCR problem (10000-L-4) Using DBD *and EVDBD*:

This network has been constructed to recognize the characters from A-J with different sizes of hidden layer 40, 50, and 60 units with many learning rates. The following table represents the results of this network:

TABLE 12: *Solve OCR problem (10000-L-4) Using DBD* and EVDBD

Algo. / HLS	DBD	EVDBD
40	197	61
50	164	50
60	146	37

The row that represents the hidden layer size equals to 60 is the best row but it should say that using hidden layer size greater than 60 will prevent the network from generalization. So using a medium hidden layer size is better for generalization and acceleration of the training process.

4.4.5.2 Solve OCR problem (10000-L-5) Using DBD *and EVDBD*:

The new thing about this experiment is maximizing the output layer and that is to help this network to recognize the characters from A-Z (for example, the target output for the character A is 00000, and for Z is 11001). The number of units (L) in the hidden layer was 30, 40, 50, 60, 70 and 80 units. The following table summarizes the results of this experiment. The number of epochs that have been put in the table gave a better result from the five trails that have been got with each hidden layer.

TABLE 13: *Solve OCR problem (10000-L-5) Using DBD* and EVDBD

Algo. / HLS	DBD	EVDBD
30	150	73
40	114	71
50	83	54
60	80	37
70	58	31
80	54	29

From the previous, it can notice that EVDBD could solve this problem too and through the different sizes of hidden layer. For this test and the previous tests using hidden layer size of 60 and 70 is better because it is close to the results of hidden layer size equals to 80, but it reduces from falling into local minima or overfitting.

4.4.5.3 Solve OCR problem (10000-L-6) Using DBD *and EVDBD*:

The aim of this network is recognizing a large number of symbols. The size of the output layer has been maximized to 6. So this network has the ability to recognize the capital letters from A-Z, small letters from a-z beside the digits from 0-9. According to the size of the hidden layer it was 80, 90 and 100 units as in the following table:

TABLE 14: *Solve OCR problem (10000-L-6) Using DBD* and EVDBD

Algo. / HLS	DBD	EVDBD
80	72	61
90	50	52
100	31	25

From the results of the previous table, the new result is that when the size of the training set (i.e. number of symbols) increases it becomes very important to use a larger size of the hidden layer. In addition, the required number of epochs becomes less.

4. CONCLUSION

This paper introduced a new algorithm *EVDBD* which has been proposed for the training of multilayer neural networks, the EVDBD is an enhanced version of the Delta-Bar-Delta algorithm. The study shows that *EVDBD* is beneficial in speeding up the training process. The experiments results confirmed these observations.

The experimental results show that an EVDBD has the same features of the DBD except that the EVDBD minimizes the time of the training process. In addition, the result of this paper can be generalized to be applied on the multi-layer neural networks.

FUTURE WORK

Use the adaptive momentum strategies should be examined and their results should be compared to those produced by their counterparts.

References

[1] Anderson G. Peter, *Training Wheels for Encoder Networks*, Proc. of the Int. ICSC Symposium on Intelligent Industrial Automation(IIA 96) and Soft Computing (SOCO 96), Reading, U.K. , Academic Press.1996.

[2] Anthony, M., and Bartlett, P.L., *Neural Network Learning: Theoretical Foundations*, Cambridge: Cambridge University Press, ISBN 0-521-57353-X, 1999.

[3] Carling, A., Alison, *Introducing Neural Networks*, 1992, 133-154.

[4] Carsten Peterson, "Artificial Neural Networks", Cambridge University Press, 2004

[5] Fahlman, S. E., Faster-learning variations on Backpropagation: an empirical study. Proceedings of the 1988 Connectionist Models Summer School, 38-51.

[6] Freeman, J. A., Skapura, D. M., Backpropagation. *Neural Networks Algorithm Applications and Programming Techniques*, 1992, 89-125.

[7] Jacobs, R. A., Increased rates of convergence through learning rate adaptation, Neural Networks, 1988,1,169-180

[8] Martin T. Hagan, Howard B. Demuth, Neural Networks Design, 1996, 11.1-12.52

[9] Maureen Caudill, and Charles Butler, Understanding Neural Networks: Computer Explorations, Volume 1,1993,.155-218

[10] M.A. Otair and W.A. Salameh, An Improved Back-Propagation Neural Networks using a Modified Non-linear Function, Proceedings of the IASTED International Conference, 2004,442-447
[11] M.A. Otair and W.A. Salameh," Speeding Up Backpropagation Neural Networks", under preparation, 2004, Accepted for publication in the Journal of Issues in Informing Science and Information Technology, 2005.
[12] M.A. Otair and W.A. Salameh, *Online Handwritten Character Recognition Using An Optical Backpropagation Neural Networks*, Proceedings of The 2004 International Research Conference on Innovations in Information Technology, 2004, 334-341.
[13] M.A. Otair and W.A. Salameh, *Optical Back-Propagation Neural Networks with a Momentum Factor –A Case Study*-, WSEAS Transactions on Circuits and Systems, Issue 9, Volume 3, 2004, p. 2073-2078.
[14] M.A. Otair and W.A. Salameh, *Comparative Study between Different versions of the Backpropagation and Optical Backpropagation,* under review, 2005.
[15] Minai, A.A., Williams, R.D., Acceleration of back-propagation through learning rate momentum adaptation, Proceedings of the International Joint Conference on Neural Networks, 1990, 1676-1679.
[16] M. Hagiwara, "Theoretical derivation of momentum term in back-propagation" , Int. Joint Conf. On Neural Networks, 682-686 ,1992
[17] Negnevitsky, M, *Artificial Intelligence: A Guide to Intelligent Systems*, Boston: Addison Wesley, 2002.
[18] Paul Bakker , Steven Philps, Janet Wiles, *The N-2-N encoder: a matter of representation* , Proc. of The Int. Conf. On Artificial Neural Networks, Amsterdam, The Netherlands, 1993.
[19] R. Plamondon, D.P. Lopresti, L.R.B. Schomaker, R. Shrihari, *Online Handwriting recognition*, Wiley Encyclopedia of Electrical and Electronics Engineering, John G. Webster (editor), vol. 15, John Wiley & Sons, 1999, 123-146.
[20] Robert Callan, The Essence of Neural Networks, Southmpton Institute, 1999, 33-52.
[21] R.S. Sutton , Adapting bias by gradient descent : Incremental Version of Delta-Bar-Delta", Proc. National Conf. On AI , MIT Press ,171-176 ,1992
[22] Rumelhart, D. E., Hinton, G. E., and Williams, R. J., Learning internal representations by error propagation, In D. E. Rumelhart and J. L. McClelland (eds) *Parallel Distributed Processing*, 1986, 318-362.
[23] Simon Hakin , Neural Networks A Comprehensive Foundation ,2nd Edition , 1999 ,161-184