HANDWRITTEN ENGLISH VOWELS RECOGNITION USING HYBRID EVOLUTIONARY FEED-FORWARD NEURAL NETWORK

Manish Mangal[1], Manu Pratap Singh[2]
Department of Computer Science, Institute of Computer & Information Science,
Dr. B.R.Ambedkar University, Khandari Campus, Agra - 282002, Uttar Pradesh, India.
[1]E-mail: manish_mangal@ieee.org, [2]E-mail: manu_p_singh@hotmail.com

ABSTRACT

This paper compares the performance of Backpropagation algorithm with the hybrid evolutionary algorithm (EA) in feed-forward neural networks. The analysis is done with five different samples of handwritten English language vowels. These characters are presented to the neural network for training. The training in the neural network is performed by adjusting the connection strength in it. The evolutionary algorithms evolve the population of weights of the neural network during the training. Using a simulator program, which is designed in C & MATLAB, each algorithm was compared by using five data sets of handwritten English language vowels. The 5 trials indicate significant difference between the two algorithms in the chosen data sets. The results show that the performance of the neural network is much accurate and convergent for the learning with the hybrid evolutionary algorithm.

Keywords: Character recognition, hybrid evolutionary algorithm, multilayer feed-forward neural network, backpropagation algorithm.

1.0 INTRODUCTION

Artificial neural network (ANN) is a technique for creating artificial intelligence in the machine. This is an attempt of the modeling of the human brain in the serial machine for the various pattern recognition tasks [1]. Such a neural network is composed of computer-programming objects called nodes. These nodes closely correspond in both form and function to their organic counterparts called neurons. Nodes are programmed to perform simple mathematical function or to process a small portion of data. A node has other components, called weights, which are an integral part of the neural network. Weights are associated with the neurons in the form of connection strength. By adjusting a weight on a node, the output data is changed, and the behavior of the neural network can be altered and controlled. Careful adjustment of these weights is how the network learns, in which its initial behavior is learned by being exposed to training data. The network processes the data, and a controlling algorithm adjusts each weight to arrive at the correct or final answer(s) for the data. These algorithms or procedures are called learning algorithms.

Neural networks are often used for pattern recognition and classification [2, 3, 4]. Their adaptability and learning capabilities make them excellent choices for tasks requiring comparison of data sets or extracting subtle patterns from complex data. Pattern recognition in neural networks is a very broad field, but a common use for a NN is in handwriting or character recognition [5, 6]. This pattern matching technique enables computers to identify and utilize human handwriting [7, 8].

This paper focuses on the recognition of handwritten English language vowels in its basic form, i.e. individual character recognition. The rationale is to improve the efficiency of neural network for character recognition task. A series of tests were conducted to determine which of the two learning algorithms i.e. Backpropagation [9] or Hybrid Genetic Algorithm (GA) [10], trained a neural network faster and more efficiently. The determination of convergent weights for pattern recognition is also an important observation of this research. The simulated results are determined from the five sets of handwritten characters of English vowels.

Often in neural networks, nodes are organized in a manner called a feed-forward network. In a feed-forward neural network, nodes are organized into layers; each "stacked" on one another. The neural network consists of an input layer of nodes, one or more hidden layers, and an output layer [11]. Each node in the layer has one corresponding node in the next layer, thus creating the stacking effect. The input layer's nodes have output functions that deliver data to the first hidden layer nodes. The hidden layer(s) is/are the processing layer(s), where all of the actual computation takes place. Each node in a hidden layer computes a sum based on its input from the previous layer (either the input layer or another hidden layer). The sum is then "compacted" by a sigmoid function (a logistic curve), which changes the sum to a limited and manageable range. The output sum from the hidden layers is passed

on to the output layer, which produces the final network result. Feed-forward networks may contain any number of hidden layers, but consists of only one input and one output layer. A neural network with a single hidden layer can learn any set of training data that a network with multiple layers can learn [12]. However, neural network with a single hidden layer may take longer to train.

Backpropagation is a learning rule for the training of multi-layer feed-forward neural network. Backpropagation derives its name from the technique of propagating the error in the network backward from the output layer. To train a Backpropagation neural network, it must be exposed to a training data set and the answers or correct interpretations of the set. During a forward pass through the network, the nodes in the network accumulate the changes in weight, and on the backward pass, the weights are altered. The method of weight modification calculation is an integral part of the Backpropagation design.

Algorithms for multi-layer feed-forward neural networks such as Backpropagation, suffer from the intrinsic complications of gradient-descent techniques – predominantly local minima in the error function [3, 4]. Genetic algorithms [15] propose another solution to conventional techniques in which they do not rely on gradient information – they can sample the search space irrespective of where the existing solution is to be found, while remain biased towards good solutions.

A genetic algorithm has four main elements: (i) the genetic code, a concise representation for an individual solution; (ii) the population, a number of individual solutions; (iii) the fitness function, an evaluation of the usefulness of an individual; (iv) the propagation techniques, a set of methods for generating new individuals. In the genetic algorithm, first a population of individuals is generated by randomly selecting different samples (or genes) through mutation and elitism. From these samples, the GA crossover operator is used to generate the combinations of selected samples. The fitness of each individual is then evaluated. The best among all these is selected for further processing. The cycle of evaluation and propagation continues until a satisfactory solution, the optimal solution, is found.

Much work has been done on the evolution of neural networks with GA [13, 14]. The majority of the implementations of the genetic algorithms are derivatives of Holland's innovative specification [15]. Evolution has been introduced in neural networks at three levels: connection weights, architectures and learning rules [16]. The evolution of neural network architectures has been the focus of much of the study in evolutionary computation [17]. The evolution of learning rules has not yet been subjected to such identical study [16], but the literature on the subject is mounting [14]. The evolution of a network's connection weights is an area of curiosity [18] and the center of attention of this work.

The next section presents the simulation design and algorithmic steps of the problem. The experimental results are presented in Section 3. Section 4 provides the description of the results shown in Section 3. Section 5 concludes this study and gives some directions on future research.

2.0 SIMULATION DESIGN AND IMPLEMENTATION DETAILS

This section describes the experiments designed to evaluate the performance of feed-forward neural network when evolved with two different learning algorithms.

2.1 Experiments

Two experiments were conducted, each with the same network architecture (4 neurons in the input layer, 6 neurons in each hidden layers, 2 hidden layers and 5 neurons in the output layer), the same input data (5 different input data sets were used for each experiment) and two different learning algorithms (Backpropagation and hybrid GA). In each experiment, one of the two learning algorithms trained the neural network populations with a standard Backpropagation algorithm and the remaining one trained the neural network populations with hybrid GA algorithm. Five trials were conducted for each experiment. The genetic operators used in each experiment are summarized in Table 1:

Table 1: Genetic operators used in the experiments

Training Algorithms	**Genetic Operators Used**
Backpropagation	None
Hybrid GA	Mutation with probability <= 0.1 and Crossover

Descriptions of these genetic operators can be found in Section 2.3. The parameters used in the experiments are listed in Table 2 and 3.

Table 2: Parameters used for Backpropagation Algorithm

Parameter	**Value**
Backpropagation learning Rate (η)	0.1
Momentum Term (α)	0.9
Doug's Momentum Term $\left(\frac{1}{1-(\alpha)}\right)$	$\left(\frac{1}{1-(\alpha)}\right)$
Adaption Rate (K)	3.0
Minimum Error Exist in the Network $(MAXE)$	0.00001
Initial weights and biased term values	Randomly Generated Values Between 0 and 1

Table 3: Parameters used for Hybrid Genetic Algorithm

Parameter	**Value**
Adaption Rate (K)	3.0
Backpropagation learning Rate (η)	0.1
Momentum Descent Term (α)	0.9
Doug's Momentum Term $\left(\frac{1}{1-(\alpha)}\right)$	$\left(\frac{1}{1-(\alpha)}\right)$
Mutation Population Size	3
Crossover Population Size	2000
Minimum Error Exist in the Network $(MAXE)$	0.00001
Initial weights and biased term values	Randomly Generated Values Between 0 and 1

In all experiments, the neural networks were trained to generate the appropriate classification for the handwritten English language vowels. For this, the scanned images of five different samples of handwritten English language vowels (Fig. 1) were obtained.

After collecting these samples, each English vowel image was partitioned into four equal parts, and the density values of the pixels for each part were calculated. Next, the density values of the central of gravities for these partitioned images of the English vowel were calculated. Consequently four values were obtained from an image of handwritten English language vowel, which were then used as the input for the feed-forward neural network. This procedure was used to present the input pattern to the feed-forward neural network for each of the sample of English language vowels.

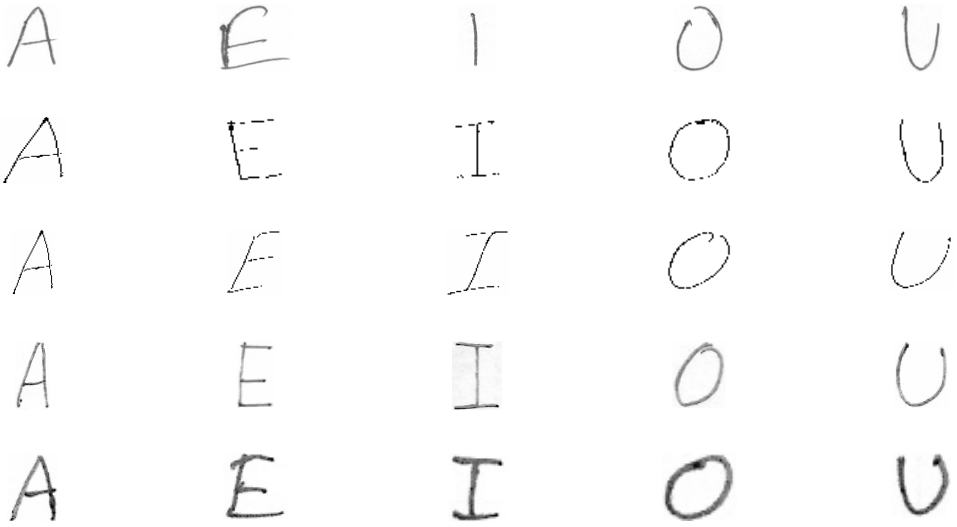

Fig. 1: Scanned images of five different samples of handwritten English language vowels

2.2 Neural Network Architecture

The structural design of the neural networks was based on a fully connected feed-forward multilayer generalized perceptron. Four input units were used, with two hidden layers with six neurons each and five neurons in the output layer. The hidden layers were used to investigate the effects of Backpropagation and hybrid GA on the hyper plane. A hyperplane is a concept in geometry. It is a higher-dimensional generalization of the concepts of a line in Euclidean plane geometry and a plane in 3-dimensional Euclidean geometry. In a one-dimensional space (a straight line), a hyperplane is a point; it divides a line into two rays. In two-dimensional space (such as the xy plane), a hyperplane is a line; it divides the plane into two half-planes. In three-dimensional space, a hyperplane is an ordinary plane; it divides the space into two half-spaces. This concept can also be applied to four-dimensional space and beyond, where the dividing object is simply referred to as a hyperplane.

Each network had a single output unit with the following activation and output functions,

$$A_k^o = \sum_{i=0}^{H} w_{io_k} O_i^h \tag{2.1}$$

$$\int\left(A_k^o\right) = \int\left(\sum_{i=0}^{H} w_{io_k} O_i^h\right) = O_k^o \tag{2.2}$$

where function $\int\left(A_k^o\right)$ is given as,

$$O_k^o = \frac{1}{1+e^{-KA_k^o}} \tag{2.3}$$

Now, similarly, the output and activation value for the neurons of hidden layers and input layer can be written as,

$$A_k^h = \sum_{i=0}^{N} w_{ik} O_i \tag{2.4}$$

$$\text{and} \quad O_k^h = \frac{1}{1+e^{-KA_k^h}} \tag{2.5}$$

$$\text{and} \quad O_k^i = \int\left(A_k^i\right) = A_k^i \tag{2.6}$$

In the Backpropagation learning algorithm, the change in weight populations was done according to the calculated error in the network after each of the iteration of training. The change in weight and error in the network can be calculated as follows,

$$\Delta w_{ho}(s+1) = -\eta \sum_{i=1}^{H} \frac{\partial E}{\partial w_{ho}} + \alpha \Delta w_{ho}(s) + \frac{1}{1-(\alpha \Delta w_{ho}(s))} \tag{2.7}$$

$$\Delta w_{ih}(s+1) = -\eta \sum_{i=1}^{N} \frac{\partial E}{\partial w_{ih}} + \alpha \Delta w_{ih}(s) + \frac{1}{1-(\alpha \Delta w_{ho}(s))} \tag{2.8}$$

$$E = \frac{1}{2} \sum_{p=1}^{P} \left(O^o - T\right)^2 \tag{2.9}$$

where $\left(O^o - T\right)^2$ is the squared difference between the actual output value of the output layer for pattern p and the target output value. Doug's momentum term [19] was used with momentum descent term for calculating the change in weights in equations 2.7 and 2.8. Doug's momentum descent is similar to standard momentum descent with the exception that the pre-momentum weight step vector is bounded so that its length cannot exceed 1.0. After the momentum is added, the length of the resulting weight change vector can grow as high as 1 / (1 - momentum). This change allows stable behavior with much higher initial learning rates, resulting in less need to adjust the learning rate as the training progresses.

The evolutionary algorithms evolve the population of weights using its operators, and select the best population of the weights that minimize the error between the desired output and the actual output of neural network system.

2.3 The Genetic Algorithm Implementation

Fig. 2 shows the standard form of a genetic algorithm. The initial population was generated with randomly assigned values for weights and biases. The values were obtained from the random generator generating values between 0 and 1.

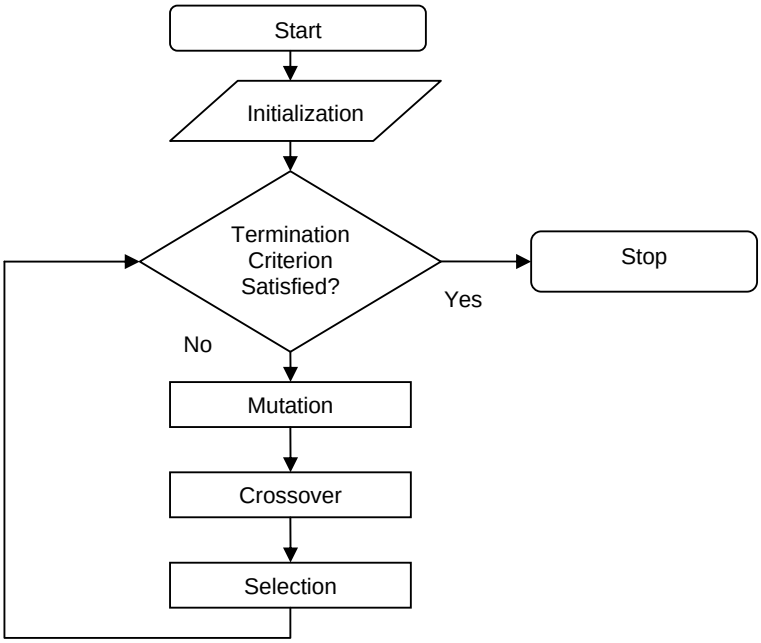

Fig. 2: Flowchart of Genetic Algorithm Implementation

2.3.1 Genetic Representation

After the initial population of weights and biases was created, the problem domain is represented as a chromosome. For that, the set of weight values is represented as a square matrix (Table 4), in which a real number corresponds to the weighted link from one neuron to another neuron. A zero value means that there is no connection between the two given neurons.

Table 4: Weights matrix for neural network

To / From	I_1	I_2	H_{11}	H_{12}	---	H_{16}	H_{21}	H_{22}	---	H_{26}	H_{31}	H_{32}	---	H_{36}	O_1
I_1	0	0	0	0	---	0	0	0	---	0	0	0	---	0	0
I_2	0	0	0	0	---	0	0	0	---	0	0	0	---	0	0
H_{11}	w_{11}	w_{21}	0	0	---	0	0	0	---	0	0	0	---	0	0
H_{12}	w_{12}	w_{22}	0	0	---	0	0	0	---	0	0	0	---	0	0
|	|	|	|	|	|	|	|	|	|	|	|	|	|	|	|
H_{16}	w_{16}	w_{26}	0	0	---	0	0	0	---	0	0	0	---	0	0

H_{21}	0	0	w_{11}	w_{21}	---	w_{61}	0	0	---	0	0	0	---	0	0
H_{22}	0	0	w_{12}	w_{22}	---	w_{62}	0	0	---	0	0	0	---	0	0
|	|	|	|	|	|	|	|	|	|	|	|	|	|	|	|
H_{26}	0	0	w_{16}	w_{26}	---	w_{66}	0	0	---	0	0	0	---	0	0
H_{31}	0	0	0	0	---	0	w_{11}	w_{21}	---	w_{61}	0	0	---	0	0
H_{32}	0	0	0	0	---	0	w_{12}	w_{22}	---	w_{62}	0	0	---	0	0
|	|	|	|	|	|	|	|	|	|	|	|	|	|	|	|
H_{36}	0	0	0	0	---	0	w_{16}	w_{26}	---	w_{66}	0	0	---	0	0
O_1	0	0	0	0	---	0	0	0	---	0	w_{11}	w_{21}	---	w_{61}	0

Each row of this matrix represents a group of incoming weighted links to a single neuron. In total, there are 102 weighted links between neurons and 19 biased values of neurons in the neural network. Thus, a chromosome is a collection of genes representing either a weight or a biased value. A 121-gene chromosome can be represented as

w_{11}	w_{21}	Bias of hidden unit 1 of layer 1	w_{12}	w_{22}	Bias of hidden unit 2 of layer 1	--------	w_{11}	w_{21}	------	w_{61}	Bias of unit 1 of output layer

where some gene correspond to a single weighted link and some correspond to biased values of neurons in the network.

2.3.2 The Mutation Operator

A mutation operator which randomly selects a gene in a chromosome and adds a small random value between –1 and 1 to that particular gene, produces the next generation population of 121-gene chromosomes. The size of the next generated population will be $n+1$, if the mutation operator applied n times over the old chromosome:

$$C^{new} = C^{old} \bigcup_{i=1}^{n} \left[C_{121-\lambda} \bigcup \left(C_{\lambda}^{old} + \in \right) \right] \tag{2.10}$$

where C^{old} symbolizes the old chromosome of 121-gene, $\in$ symbolizes the small randomly generated value between –1 to 1, λ symbolizes the randomly selected gene of C^{old} chromosome for adding the $\in$, and C^{new} symbolizes the next generation population of chromosome, i.e. $C^{new} = \left[C_1, C_2, C_3, -------, C_n, C_{n+1} \right]$. The inner $\bigcup$ operator prepares a new chromosome at each iteration of mutation, and the outer $\bigcup$ operator builds the new population of chromosomes called C^{new}.

2.3.3 Elitism

Elitism was used when creating each generation so that the genetic operators did not lose good solutions. This involved copying the best-encoded network unchanged into the new population as given in equation 2.10, which includes C^{old} for creating C^{new}.

2.3.4 Selection

A chromosome C^{sel} is selected among the mutated population of chromosomes for which the sum of squared errors is minimum for the feed-forward neural network, i.e. iteratively all the chromosome values will be assigned to the network architecture in terms of weights and biased values defined in chromosome. After assigning the values, the network architecture will be able to fabricate output using these assigned values. For each new chromosome C^{new}, the error can be calculated using these fabricated outputs as in equation 2.7. Next, the selection operator will pick a chromosome C^{sel} from C^{new}, which generates minimized error for the network.

2.3.5 Crossover

A crossover operator takes selected chromosome and creates a child for producing the next generation population of 121-gene chromosomes of size $n+1$. This next generation of population is produced by applying the crossover operator n times. The experiments performed the crossover operation by swapping the two randomly selected gene values of the parent chromosome as given in Fig. 3 and equation 2.11:

$$C^{next} = C^{sel} \bigcup_{i=1}^{n} \left[\left(C^{sel} - C_{\alpha}^{sel} - C_{\beta}^{sel} \right) \cup \left(C_{\alpha}^{sel} \xleftrightarrow[v_\alpha \leftrightarrow v_\beta]{} C_{\beta}^{sel} \right) \right] \tag{2.11}$$

where α and β symbolize the randomly generated genes positions in CP_1 and CP_2 in C^{sel} chromosome, and C^{next} is the next generation of size $n+1$.

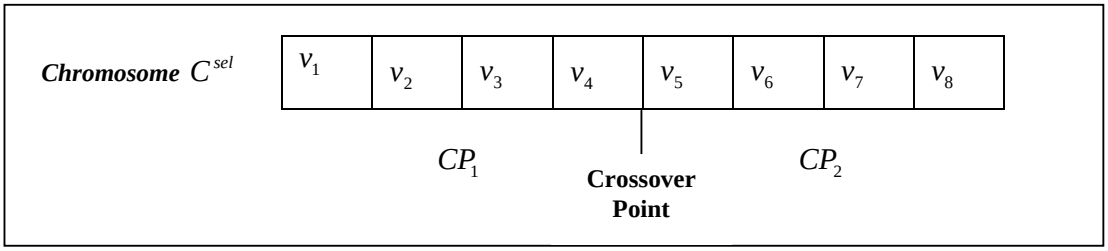

Fig. 3 (a): Chromosome before applying crossover operator

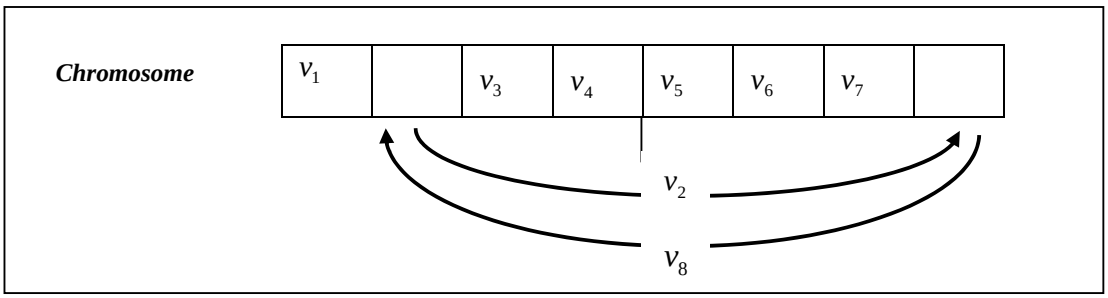

Fig. 3 (b): Applying crossover operator on chromosome

Chromosome	v_1	v_8	v_3	v_4	v_5	v_6	v_7	v_2

Fig. 3 (c): Chromosome after applying crossover operator

2.3.6 Fitness Evaluation Function

A fitness evaluation function defines a function for evaluating the chromosome performance. This function must estimate the performance of weight population of a given feed-forward neural network. A simple function defined based on the proportion of the sum of squared errors is applied. To evaluate the fitness of a given chromosome, each weight and biased value contained in the chromosome is assigned to the respective link and neuron in the network. The training set is then presented to the network, and the sum of squared errors is calculated. The smaller the sum, the higher is the fitness of the chromosome. In other words, the genetic algorithm attempts to find a set of weights and biased values that minimizes the sum of squared errors.

$$\min error = 1.0$$

For all the n+1 chromosomes

$$\text{if } \left(\min error > E_{C_i^{next}}\right) \text{ then}$$

$$\left(\min error = E_{C_i^{next}}\right)$$

$$C^{\min} = C_i^{next}$$

$$\text{else } \left(\min error = \min error\right) \tag{2.12}$$

where $E_{C_i^{next}}$ symbolizes the error calculated for *i*th chromosome among the $n+1$ chromosomes of C^{next} population and $C^{\min}$ symbolizes the chromosome which has minimized error.

3.0 RESULTS

This section presents the results of this study, which will be elaborated in Section 4.

3.1 Results for First Trial of Simulation

Table 5: The results for the classification of handwritten English language vowels using Backpropagation and hybrid GA learning algorithms

S. No.	Characters	Backpropagation Epochs					Hybrid GA Epochs									
		Sample 1	Sample 2	Sample 3	Sample 4	Sample 5	Sample1	No. of Convergence Matrices	Sample 2	No. of Convergence Matrices	Sample 3	No. of Convergence Matrices	Sample 4	No. of Convergence Matrices	Sample 5	No. of Convergence Matrices
1	A	550	0.3	0.4	0.4	0.4	82	22	4	1	4	1	11	1	3	1
2	E	0.2	0.2	0.2	0.4	0.4	2	12	2	28	2	25	2	11	2	26
3	I	0.4	0.2	0.4	0.4	0.4	2	11	2	23	2	24	2	12	2	16
4	O	0.4	0.1	0.4	564	0.4	2	14	2	4	2	11	2	9	2	13
5	U	0.4	0.4	0.4	0.4	0.4	2	12	4	1	3	1	3	2	6	1

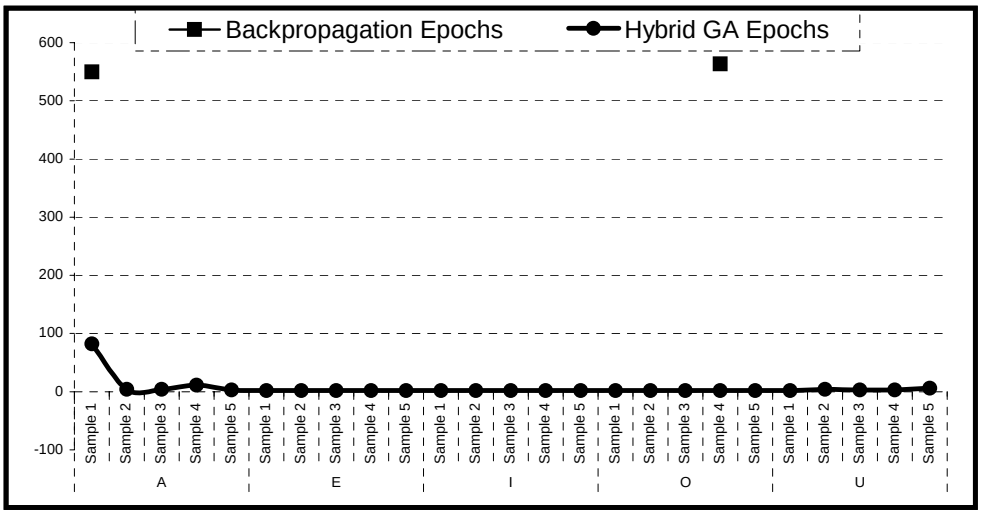

Fig. 4: The comparison chart for handwritten English vowels classification epochs of two learning algorithms

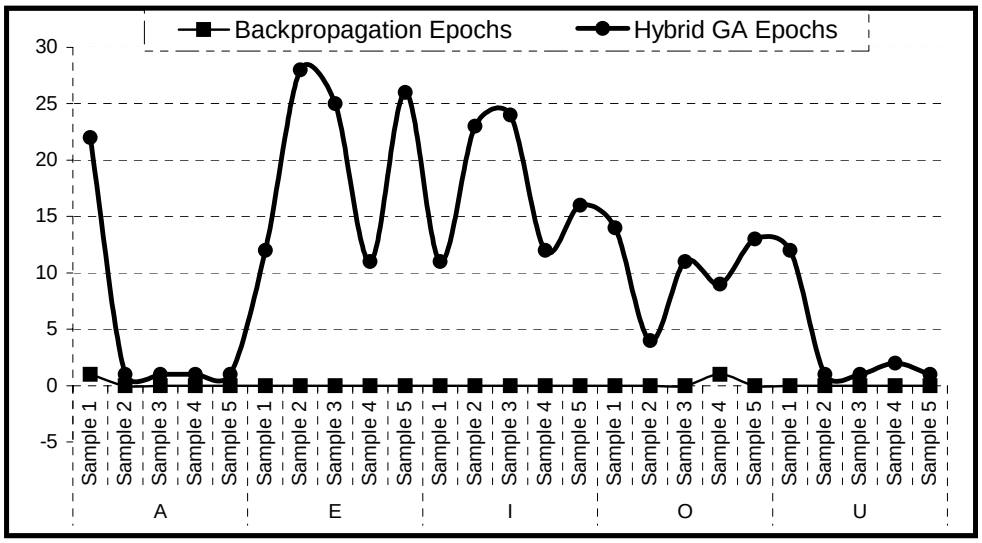

Fig. 5: The comparison chart for the number of convergence matrices for handwritten English vowels classification

3.2 Results for Second Trial of Simulation

Table 6: The results for the classification of handwritten English language vowels using Backpropagation and hybrid GA learning algorithms

S. No.	Characters	Backpropagation Epochs					Hybrid GA Epochs									
		Sample 1	Sample 2	Sample 3	Sample 4	Sample 5	Sample1	No. of Convergence Matrices	Sample 2	No. of Convergence Matrices	Sample 3	No. of Convergence Matrices	Sample 4	No. of Convergence Matrices	Sample 5	No. of Convergence Matrices
1	A	141	0.4	0.4	0.4	0.4	22	1	3	2	9	1	2	4	3	1
2	E	0.4	0.4	0.4	0.4	0.4	3	2	2	8	2	8	2	6	2	1
3	I	0.4	0.4	0.4	0.4	0.4	4	3	2	32	2	12	3	1	3	5
4	O	0.4	0.4	0.4	0.4	0.4	4	2	2	6	2	7	2	6	2	5
5	U	0.4	0.4	0.4	0.4	0.4	544	2	2	1	4	1	706	1	388	3

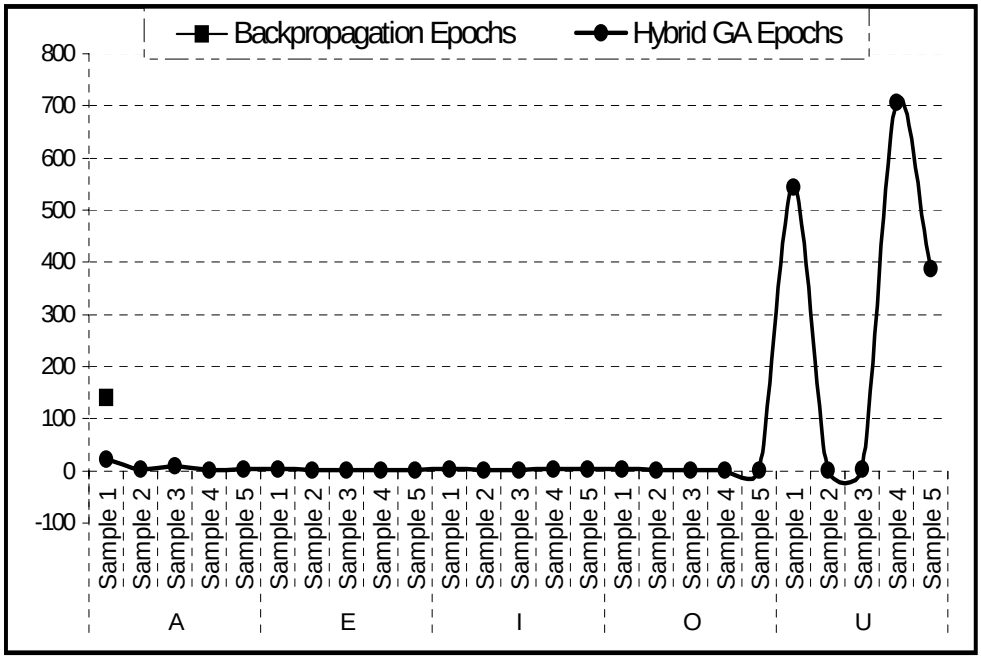

Fig. 6: The comparison chart for handwritten English vowels classification epochs of two learning algorithms

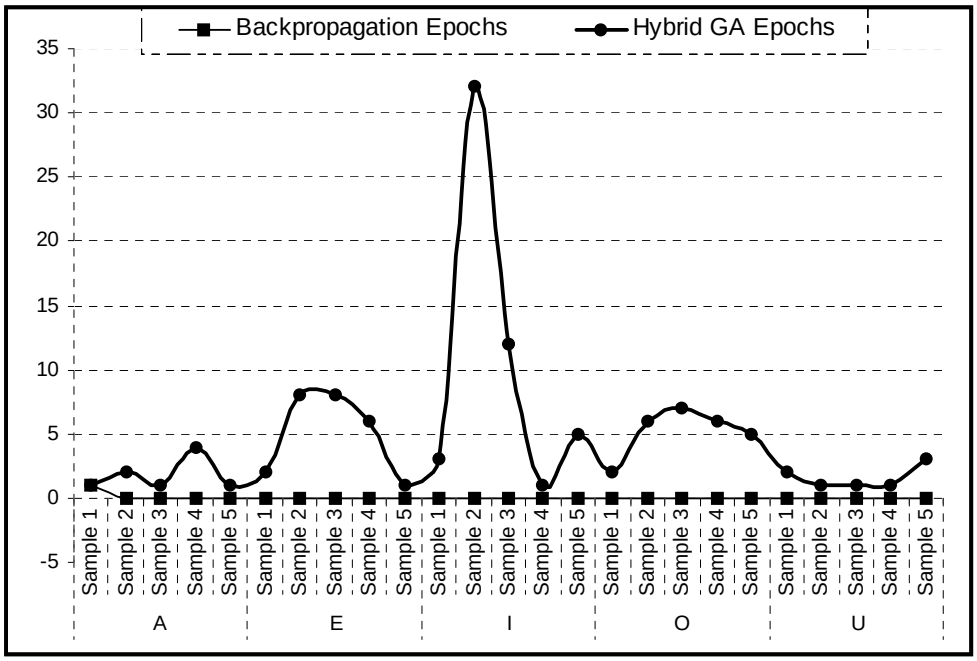

Fig. 7: The comparison chart for the number of convergence matrices for handwritten English vowels classification

3.3 Results for Third Trial of Simulation

Table 7: The results for the classification of handwritten English language vowels using Backpropagation and hybrid GA learning algorithms

S. No.	Characters	Backpropagation Epochs					Hybrid GA Epochs									
		Sample 1	Sample 2	Sample 3	Sample 4	Sample 5	Sample1	No. of Convergence Matrices	Sample 2	No. of Convergence Matrices	Sample 3	No. of Convergence Matrices	Sample 4	No. of Convergence Matrices	Sample 5	No. of Convergence Matrices
1	A	180	0.4	0.4	0.4	0.4	27	1964	2	1	2	1	2	112	2	5
2	E	15905	0.4	0.4	0.4	0.4	2	1	2	1	2	3	2	11	2	7
3	I	8361	0.4	0.4	0.4	0.4	3	2	2	2	2	3	2	38	2	9
4	O	0.4	0.4	0.4	0.4	0.4	3	3	702	1	2	5	2	4	2	3
5	U	0.2	0.4	0.4	0.4	0.4	31	36	4	1	420	32	5	1	9	1

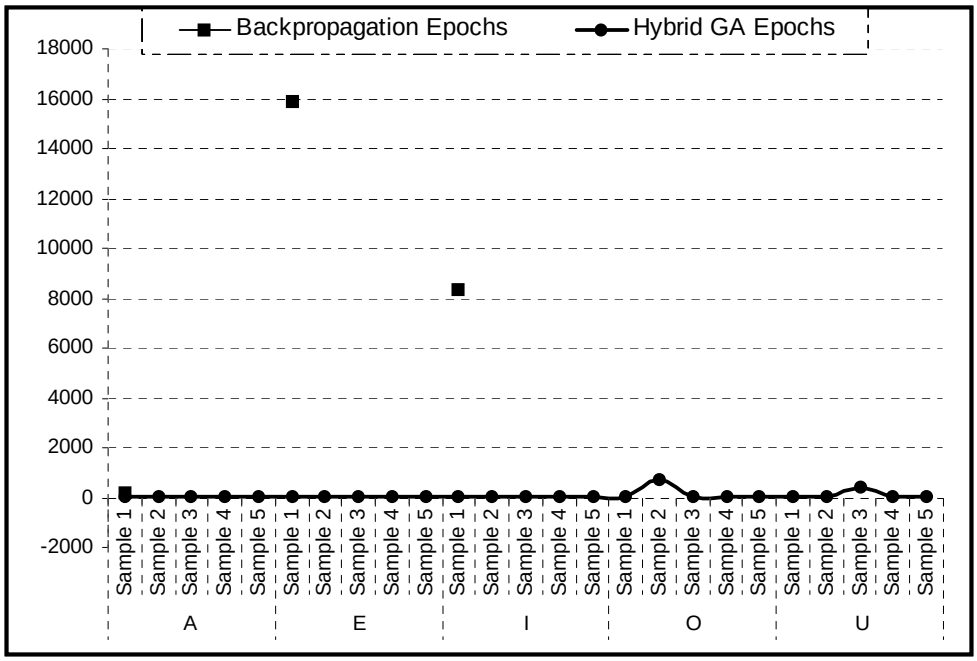

Fig. 8: The comparison chart for handwritten English vowels classification epochs of two learning algorithms

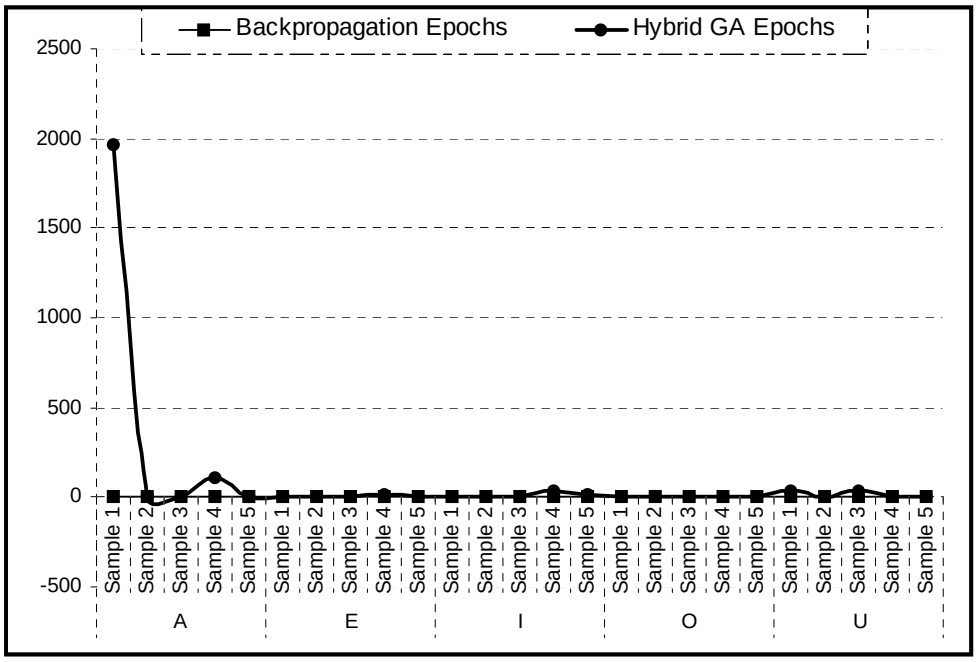

Fig. 9: The comparison chart for the number of convergence matrices for handwritten English vowels classification

3.4 Results for Fourth Trial of Simulation

Table 8: The results for the classification of handwritten English language vowels using Backpropagation and hybrid GA learning algorithms

S. No.	Characters	Backpropagation Epochs					Hybrid GA Epochs									
		Sample 1	Sample 2	Sample 3	Sample 4	Sample 5	Sample1	No. of Convergence Matrices	Sample 2	No. of Convergence Matrices	Sample 3	No. of Convergence Matrices	Sample 4	No. of Convergence Matrices	Sample 5	No. of Convergence Matrices
1	A	1696	0.4	0.4	0.4	0.4	24	28	3	3	3	1	2	2	2	1
2	E	12	0.4	0.4	0.4	0.4	702	1	2	2	2	5	2	1	12	1
3	I	4909	0.4	0.4	0.4	0.4	699	1	2	1	3	4	2	3	3	2
4	O	1391	0.4	0.4	0.4	0.4	2	5	6	1	3	2	2	1	2	1
5	U	147	1848	0.4	2369	0.4	23	1	31	1	10	1	6	1	9	1

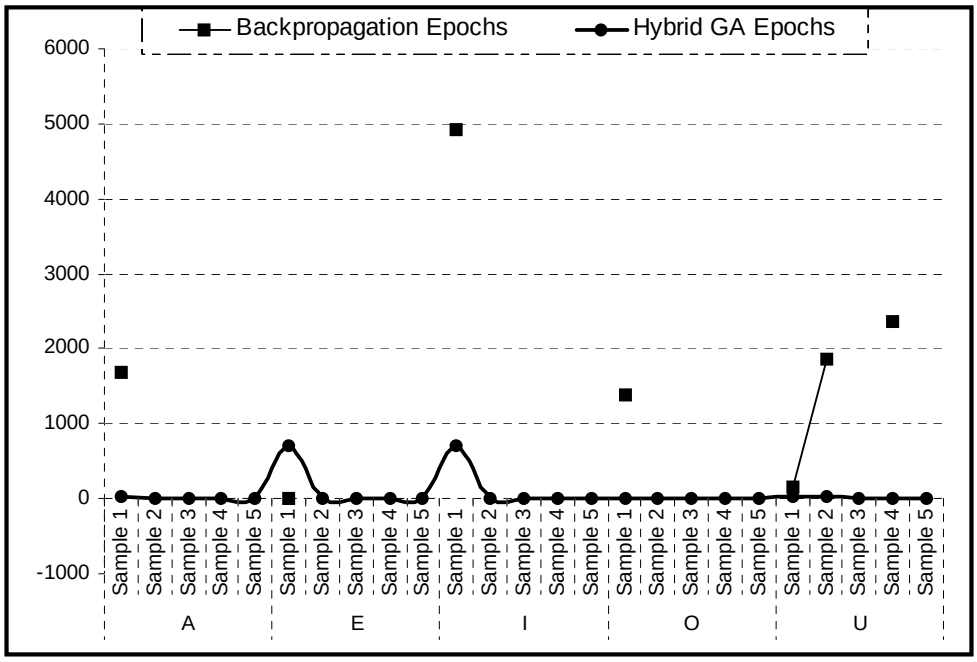

Fig. 10: The comparison chart for handwritten English vowels classification epochs of two learning algorithms

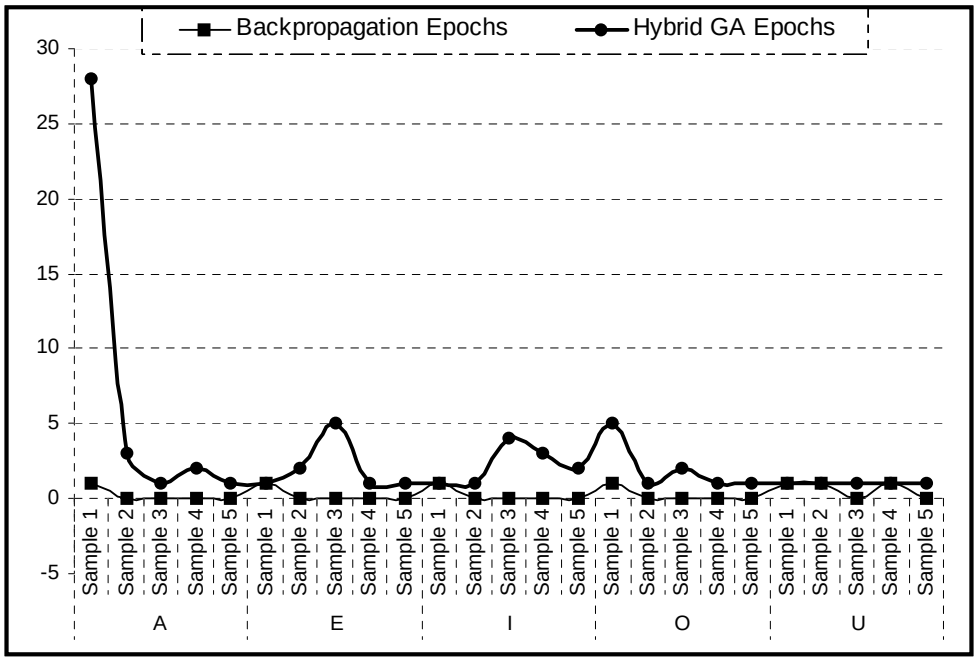

Fig. 11: The comparison chart for the number of convergence matrices for handwritten English vowels classification

3.5 Results for Fifth Trial of Simulation

Table 9: The results for the classification of handwritten English language vowels using Backpropagation and hybrid GA learning algorithms

S. No.	Characters	Backpropagation Epochs					Hybrid GA Epochs									
		Sample 1	Sample 2	Sample 3	Sample 4	Sample 5	Sample1	No. of Convergence Matrices	Sample 2	No. of Convergence Matrices	Sample 3	No. of Convergence Matrices	Sample 4	No. of Convergence Matrices	Sample 5	No. of Convergence Matrices
1	A	222	0.3	0.3	0.5	0.4	23	3	5	150	31	2	4	1	2	1
2	E	0.1	0.1	0.1	0.4	0.4	2	1	6	2	2	2	2	37	2	4
3	I	35728	0.3	0.4	0.3	0.4	3	3	31	2	2	1	2	61	2	3
4	O	0.4	0.3	0.3	0.4	0.4	3	1	2	1	2	2	3	1	2	1
5	U	586	1848	0.4	2369	0.4	31	31	26	1	700	1	28	1	332	7

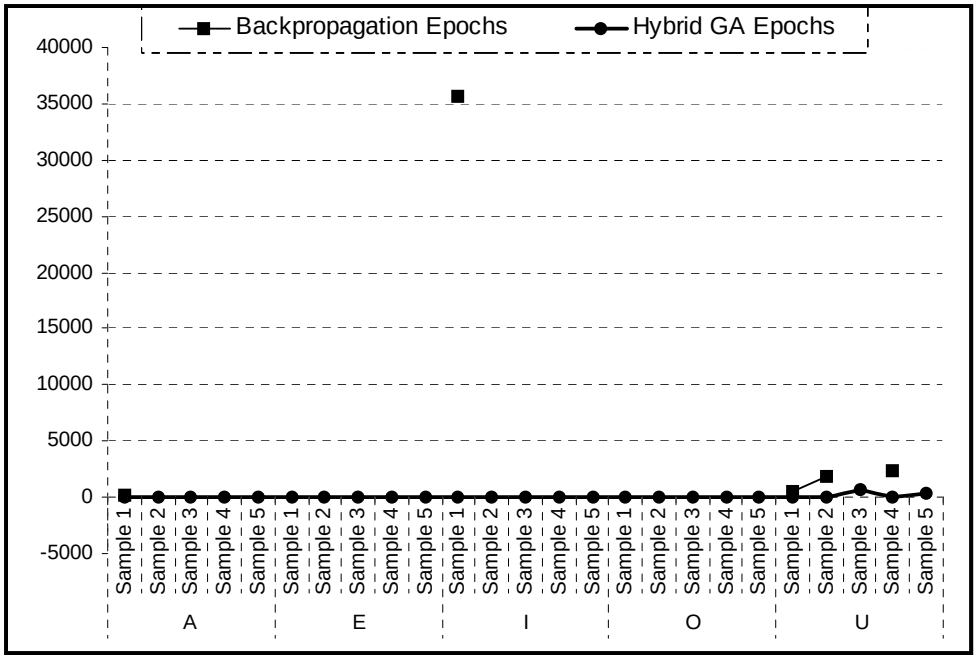

Fig. 12: The comparison chart for handwritten English vowels classification epochs of two learning algorithms

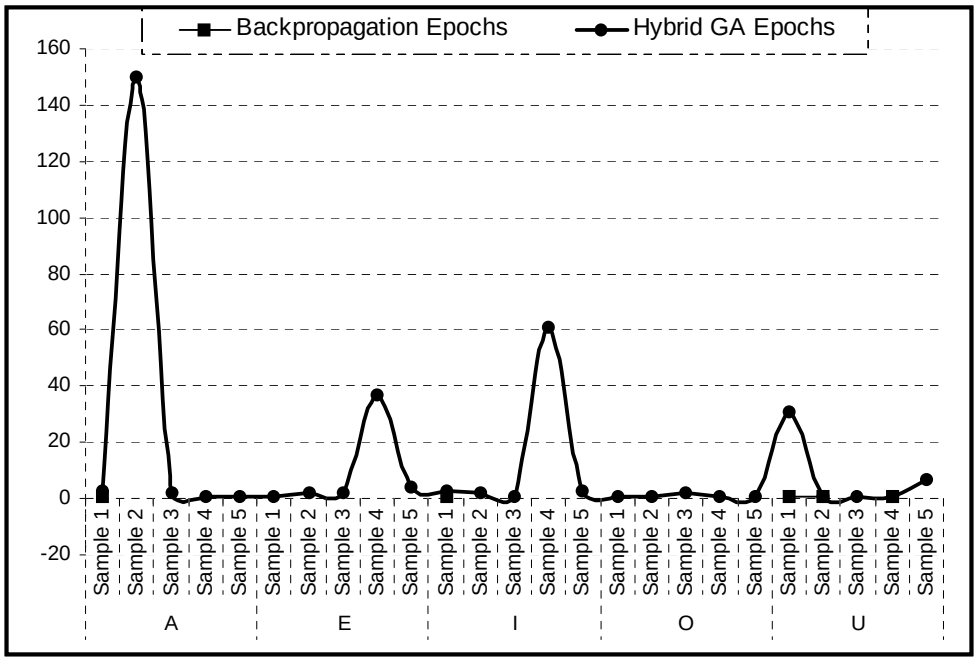

Fig. 13: The comparison chart for the number of convergence matrices for handwritten English vowels classification

4.0 DISCUSSION

The results presented in Section 3 demonstrated that, within the simulation framework presented above, large significant difference exists between the performances of Backpropagation feed-forward neural network and hybrid evolutionary feed-forward neural network for handwritten English language vowels classification problem.

Tables 5, 6, 7, 8 and 9 show the results for handwritten English language vowels classification problem performed (5 times) with the two algorithms up to having the maximum limit of 50000 iterations. All the results take five different types of handwritten samples for each English vowel character. The training has been performed in such a way that if the network is trained with an input sample of a character, then the next training cannot be done with the other input sample of the same character. This input sample will appear for training after other samples of other characters training.

These tables also show the number of convergence matrices for each character recognition process. These values show the number of convergence weight matrices obtained for the particular character by applying the hybrid evolutionary algorithm. For Backpropagation algorithm the same entry is not required, because if the characters are correctly recognized by the network, then only one convergence matrix will be obtained. Therefore, it is required only for hybrid evolutionary algorithm through which multiple number of convergence matrices can be obtained. This shows the higher accuracy of the algorithm for character recognition.

Entries with real numbers in these tables represent the errors existed in the network after executing the simulation program for 50000 iterations, i.e. up to 50000 iterations the algorithm could not converge a sample of a handwritten English language vowel into the feed-forward neural network. Fig. 4, 5, 6, 7, 8, 9, 10, 11, 12 & 13 represent the comparison charts which are based on the values found in Table 5, 6, 7, 8 and 9.

The simulation program, which was developed in MATLAB 6.5, to test the two algorithms for the classification problem of handwritten English language vowels, generates the initial weights randomly through its random generator. As a result, the epochs for the algorithms are different every time, with the same network structure and the same training data set.

5.0 CONCLUSION AND FUTURE WORK

The results described in this paper indicate that, for the handwritten English language vowels classification problem, feed-forward neural network trained with Backpropagation algorithm does not perform better in comparison to feed-forward neural network trained with hybrid genetic algorithm. The performance of hybrid evolutionary algorithm (EA) is efficient and accurate in all the simulations.

It is found that, in each and every case, the hybrid evolutionary feed-forward neural networks gives more than one convergent weight matrices for every input pattern in comparison to the Backpropagation feed-forward neural network. This shows the higher accuracy rate in the pattern recognition with hybrid evolutionary feed-forward neural network. The higher number of convergence weight matrices in the hybrid GA training process suggests that this algorithm may not be trapped in the false minima of the error surface. It may also minimize the possibilities of misclassification for any unknown testing input pattern.

The direct application of hybrid GA to the handwritten character classification has been explored in this research. The aim is to introduce as alternative approach to solve the handwritten character classification problem. The results from the experiments conducted on the algorithm are quite encouraging. Nevertheless, more work need to be done especially on the tests for other complex handwritten characters. This concept can be used for any handwriting classification problem to obtain efficient and consistent results in the future.

REFERENCES

[1] C.M. Bishop, *Neural Networks for Pattern Recognition.* New York, Clarendon, 1997.

[2] Manish Mangal and Manu Pratap Singh, "Patterns Recalling Analysis of Hopfield Neural Network with Genetic Algorithms". Accepted for publication in *International Journal of Innovative Computing, Information and Control*, (JAPAN), 2007.

[3] Manish Mangal and Manu Pratap Singh, "Analysis of Multidimensional X-OR Classification Problem with Evolutionary Feed-forward Neural Networks". Accepted for publication in *International Journal of Artificial Intelligence and Tools*, Word Scientific, 2007.

[4] Manish Mangal and Manu Pratap Singh, "Analysis of Classification for the Multidimensional Parity-Bit-Checking Problem with Hybrid Evolutionary Feed-forward Neural Network." Accepted for publication in *Neurocomputing*, Elsevier Science, 2007.

[5] Y. Le Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbad and L. D. Jackel, "Handwritten digit recognition with a backpropagation network". *Neural Information Processing Systems*, Touretzky (ed.), Vol. 2, pp. 396-404, Morgan Kaufmann Publishers, 1990.

[6] S. Mori, "Historical review of theory and practice of handwritten character recognition". *Fundamentals in Handwriting Recognition, S. Impedovo (*ed.*), NATO AS1 Series F Computer and Systems Sciences*, Vol. 124, pp 43-69, Springer-Verlag, 1994.

[7] S. Impedovo, "Frontiers in handwriting recognition". *Fundamentals in Handwriting Recognition*, S. Impedovo (ed.), NATO ASI Series F Computer and Systems Sciences, Vol. 124, pp 7-39, Springer-Verlag, 1994.

[8] G. Dirnauro, "Digital transforms in handwriting recognition", *Fundamentals in Handwriting Recognition, S. Impedovo (*ed.*), NATO ASI Series F Computer and Systems Sciences*, Vol. 124, pp. 113-146, Springer-Verlag, 1994.

[9] M. Riedmiller, *Rprop - Description and Implementation Details Technical Report*, University of Karlsruhe: W-76128 Karlsruhe,1994.

[10] X. Yao, "Evolving artificial neural networks" in *Proceedings of the IEEE*, Vol. 87, pp.1423 – 1447, 1999.

[11] C. E. Brown, J. Coakley & M. E. Phillips, "Neural networks: nuts and bolts". *Management Accounting (USA)*, Vol. 76 No. 11, pp. 52-54, 1995.

[12] M. Wright, "Designing neural networks commands skill and savvy". *EDN*, Vol. 36 No. 25, pp. 86-87, 1991.

[13] P. A. Castillo, M. G. Arenas, J. J. Castillo-Valdivieso, J. J Merelo., A. Prieto & G. Romero, "Artificial neural networks design using evolutionary algorithms" in *Proceedings of the Seventh World Conference on Soft Computing*, 2002.

[14] X. Yao, "Evolutionary artificial neural networks". *International Journal of Neural Systems*, Vol. 4 No. 3, pp. 203 – 222, 1993.

[15] J. H. Holland, "Genetic algorithms". *Scientific American*, Vol. 267 No. 1, pp. 44 – 50, 1992.

[16] K. Balakrishnan and V. Honavar, "Properties of genetic representations of neural architectures" in *Proceedings of the World Congress on Neural Networks*, pp. 807 – 813, 1995.

[17] R. Krishnan and V. B. Ciesielski, "2DELTA-GANN: Anew approach to training neural networks using genetic algorithms". in *Proceedings of the Fifth Australian Conference on Neural Networks, A. C. Tsoi (Editor)*, pp. 38 – 41, 1994.

[18] D. E.Rumelhart, G. E. Hinton & R. J. William, "Learning internal representation by error propagation", *Parallel Distributed Processing*, D. E. Rumelhart & J. L. McClelland (eds.), 1, MIT Press, pp. 318 – 362, 1986.

[19] P. Christenson, A. Maurer & G. Miner, "Handwriting recognition by neural network". http://csci.mrs.umn.edu/UMMCSciWiki/pub/CSci4555s04/InsertTeamNameHere/handwriting.pdf, 2005.

BIOGRAPHY

Manish Mangal is currently pursuing his Ph.D. in Computer Science from Agra University, Agra, UP, India. He is a senior consultant in T-Systems (India) Pvt. Ltd., Pune, Maharashtra, India. His research interests are neuro-computing, neuroscience, digital signal processing, radio wave propagation, soft computing, etc. He is also an active member of IEEE Computer Society since 2005.

Manu Pratap Singh received his Ph.D. in Computational Physics from Kumaun Univesity, Nanital, UA, India, in 2001. He is currently a senior lecturer in the Department of Computer Science, ICIS, Agra University, Agra, UP, India. His research interests focus on neuro-computing, neuroscience, neuro-informatics, soft computing, etc. Dr. Singh is a member of technical committee of IASTED, Canada since 2004. He is also the referee for various international journals. In 2005, he received young scientist award from International Academy of Physical Sciences, Allahabad, UP, India. He has also published more than 30 papers in various international and national journals.