Handwriting Digit Recognition Using Genetic Algorithm

Nan San San Myint, Dr. Khin Lay Thwin

University of Computer Studies, Hpa-an, Myanmar
sanlai.ytp@gmail.com

Abstract

Handwriting digit recognition is a challenging problem researchers had been research into this area for so long especially in the recent years. In this paper, this system used to recognize the handwritten digits of Kayin language. Genetic algorithm gives an effective Kayin digit to determine the optimal weight parameters that are multiplied by the network outputs. Neural networks have been recognized as a powerful tool for pattern recognition problems. This system checked the digit that is written manually by users and used Genetic Algorithm to produce the output result in order to recognize the Kayin digit. Then, it produces the nearly similar Kayin digit pattern.

Keywords: Handwriting digit recognition, Artificial Neural Network, Genetic Algorithm

1. Introduction

Neural networks are often used for pattern recognition and classification. 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. This pattern matching technique enables computers to identify and utilize human handwriting [9].

This thesis focuses on the handwritten digits of Kayin language. The rationale is to improve the efficiency of neural network for digit recognition task. A series of tests were conducted to determine which of the two learning algorithms i.e. back propagation or Hybrid Genetic Algorithm (GA), 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 ten sets of handwritten digit of Kayin languages. 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. 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 [7].

A neural network with a single hidden layer can learn any set of training data that a network with multiple layers can learn. However, neural network with a single hidden layer may take longer to train. Back propagation is a learning rule for the training of multi-layer feed-forward neural network. Back propagation derives its name from the technique of propagating the error in the network backward from the output layer. To train a Back propagation 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 Back propagation design [8].

2. Artificial Neural Network

Artificial neural networks have emerged as a field of study within AI and engineering via the collaborative efforts of engineers, physicists, mathematicians, computer scientists, and neuroscientists. Although the strands of research are many, there is a basic underlying focus on pattern recognition and pattern generation, embedded within an overall focus on network architectures [6].

Many neural network methods can be viewed as generalizations of classical pattern-oriented techniques in statistics and the engineering areas of signal processing, system identification, optimization, and control theory. There are also ties to parallel processing, VLSI design, and numerical analysis. A neural network is first and foremost a graph, with patterns represented in terms of numerical values attached to the nodes of the graph and transformations between patterns achieved via simple message-passing algorithms. Certain of the nodes in the graph are generally distinguished as being *input* nodes or *output* nodes, and the graph as a whole can be viewed as a representation of a multivariate function linking inputs to outputs. Four input nodes, four output nodes and one hidden layer is used in this system. Numerical values (*weights*) are attached to the links of the graph, parameterizing the input/output function and allowing it to be adjusted via a *learning algorithm*. A broader view of a neural network architecture involves treating the network as a statistical processor, characterized by making particular probabilistic assumptions about data. Patterns appearing on the input nodes or the output nodes of a network are viewed as samples from probability densities, and a network is viewed as a probabilistic model that assigns probabilities to patterns [1].

The problem of learning the weights of a network is thereby reduced to a problem in statistics—that of finding weight values that look probable in the light of observed data. The links to statistics have proved important in practical applications of neural networks. Real-world problems are often characterized by complexities such as missing data, mixtures of qualitative and quantitative variables, regimes of qualitatively different functional relationships, and highly non uniform noise. Neural networks are complex statistical models with the flexibility to address many of these complexities, but as with any flexible statistical model one must take care that the complexity of the network is adjusted appropriately to the problem at hand [2].

A network that is too complex is not only hard to interpret but, by virtue of overfitting the random components of the data, can perform worse on future data than a simpler model. This issue is addressed via statistical techniques such as cross-validation, regularization, and averaging, as well as the use of an increasingly large arsenal of Bayesian methods. Other practical statistical issues that arise include the assignment of degrees of confidence to network outputs ("error bars"), the active choice of data points ("active learning"), and the choice between different network structures ("model selection") [3]. Progress has been made on all these issues by applying and developing statistical ideas. The statistical approach also helps in understanding the capabilities and limitations of network models and in extending their range. Neural networks can be viewed as members of the class of statistical models known as "nonparametric," and the general theory of nonparametric statistics is available to analyze network behavior. It is also of interest to note that many neural network architectures have close cousins in the nonparametric statistics literature; for example, the popular *multilayer perceptron* network is closely related to a statistical model known as "projection pursuit," and the equally popular *radial basis function* network has close ties to kernel regression and kernel density estimation [5].

3. Genetic Algorithms

A genetic algorithm is a heuristic technique used to solve optimization problems. Optimization problems attempt to find the best solution(s) for a given problem that has several parameters (goals or resources) with associated constraints. The most basic tools for solving optimization problems are complete enumeration of all possible choices, calculus, and linear optimization techniques using the simplex algorithm such as Lindo or Excel's solver. However, as I will discuss in the next section, these traditional methods break down when the problem gets very large or complicated. Some of the same issues that affect these tools affect genetic algorithms, but for the most part, GAs are far more robust at handling very complex and non-linear problems. For example, a problem involving selecting the best shipping route for a company that needs to make ten shipments may be solved using an 'exhaustive search' technique[3]. The problem could be defined in the computer, which could go through the 3.6 million different combinations of the cities in a reasonable amount of time. Such a problem is known as a non-deterministic polynomial problem [4].

The basic GA operators are selection, crossover and mutation.

3.1 Fitness Function

A fitness function is a particular type of objective function that prescribes the optimality of a solution (that is, a chromosome) in a genetic algorithm so that that particular chromosome may be ranked against all the other chromosomes. Optimal chromosomes, or at least chromosomes which are more optimal, are allowed to breed and mix their datasets by any of several techniques, producing a new generation that will (hopefully) be even better.

The weighted sum fitness function of the k objectives in (1) is written as follows:

$$fitness(\mathbf{x}) = w_1 f_1(\mathbf{x}) + w_2 \cdot f_2(\mathbf{x}) + \ldots + w_k \cdot f_k(\mathbf{x})$$

where w_i is a non-negative weight value.

One important issue is the specification of the weight vector $\mathbf{w} = (w_1, w_2, \ldots, w_k)$.

3.2 Selection scheme

Selection in GAs aims at giving higher probabilities to fitter individuals in a population so that they can produce hopefully fitter offspring. the genetic is based on the following strategy: after all creatures have been evaluated and assigned a score, a set of elite creatures is directly copied in the next population. This strategy ensures that best creatures are always duplicated in the next generation. The rest of the produced population is then generated by crossover using rank-based selection based on the cumulative probability for each chromosome. The generated offsprings replace their parents in the next generation after the application of our mutation operator.

3.3 Parameter of GA

There are two basic parameters of GA - crossover probability and mutation probability.Crossover probability is that how often will be crossover performed. If there is no crossover, offspring is exact copy of parents. If there is a crossover, offspring is made from parts of parents' chromosome. If crossover probability is 100%, then all offspring is made by crossover. If it is 0%, whole new generation is made from exact copies of chromosomes from old population (but this does not mean that the new generation is the same!). Crossover is made in hope that new chromosomes will have good parts of old chromosomes and maybe the new chromosomes will be better. However it is good to leave some part of population survive to next generation.Mutation probability is that how often will be parts of chromosome mutated. If there is no mutation, offspring is taken after crossover (or copy) without any change. If mutation is performed, part of chromosome is changed. If mutation probability is 100%, whole chromosome is changed, if it is 0%, nothing is changed. Mutation is made to prevent falling GA into local extreme, but it should not occur very often, because then GA will in fact change to random search.

3.4 Crossover

The essence of any crossover operator is to exchange components of the two parents to form new offsprings. In this system one-point crossover is used, it’s realized by cutting the strings at a randomly chosen position. The child is hence generated by taking one segment part from each parent. The genetic operators must produce correct and complete offsprings. The choice of the cross point position is important in order to generate a valid offspring operating on the same interface units (input and output) as their parents.

3.5 Mutation

Often, the mutation is controlled by the mutation rate, which is very low for GAs in comparison with the crossover rate. In contrast, evolutionary programming often uses mutation almost exclusively. In this paper the mutation is performed more frequently than in traditional genetic algorithms.

4. System Design

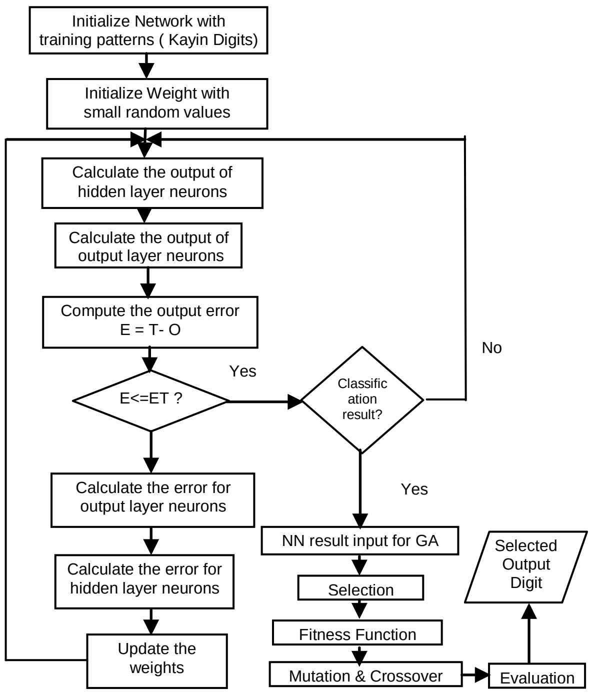

Initialize Network with training patterns (Kayin Digits)
Initialize Weight with small random values
Calculate the output of hidden layer neurons
Calculate the output of output layer neurons
Compute the output error E = T- O
E<=ET ?
Yes
Classific ation result?
No
Yes
Calculate the error for output layer neurons
Calculate the error for hidden layer neurons
Update the weights
NN result input for GA
Selection
Fitness Function
Mutation & Crossover
Evaluation
Selected Output Digit

Figure 1. System flow diagram

In this system, inputs are extracted from the handwriting digit. In this system, about 300 train data(Kayin Digits) is used. The system initializes the weight with small random values, calculate the output of hidden layer neurons and calculate the output of output layer neurons. Then calculate mean square value. If error is greater than minimum value, calculate error for output and hidden layer and update the weight. If error less than or equal with minimum value, stop training generate the classification result. The output values are carried to GA as chromosomes. GA generates these chromosomes and crossover. Finally, GA displays the output handwriting digit.

4.1 Step by Step Calculation of the System

1. Provide initial inputs of sample handwritten digits to train the artificial neural networks.
2. Start the process of training the networks.
3. Store the weight matrices and bias values obtained after training as files.
4. Read the file containing the input matrix.

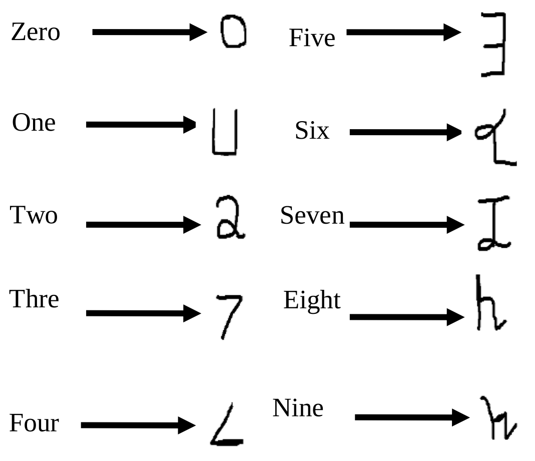
5. Feed this as the input to the neural network.
6. Send the outputs of the neural networks to the Genetic Algorithm.

The system is developed artificial neural networks need to select the best and fittest solution from the set of outputs obtained. This is carried out with the help of Genetic Algorithm which is applied such that it accepts the output of the artificial neural network. The system passes the input column matrix in the neural network and get the output. The output is sent to the Genetic Algorithm and hence making the initial population for the Algorithm.

The following method is applied for applying genetic algorithm to the output of the artificial neural network:

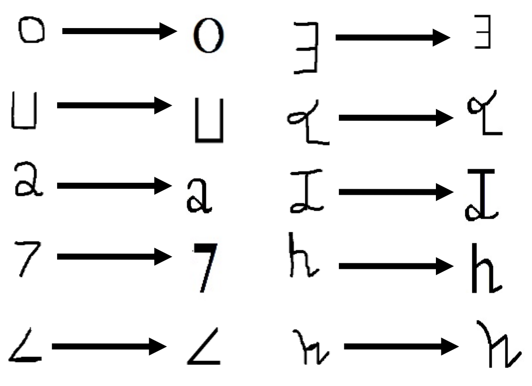
1. **Initialization**: Select the output of the neural network with the. This corresponds to the initial population for the Genetic Algorithm.
2. **Selection**: Select the indexes from the neural network that has minimum number.
3. **Fitness function**: Compute the correlation coefficients of the selected indexes.
4. **Mutation & Crossover**: For the correlation coefficients less than the threshold value 0.50 repeat the step of the fitness function for a different training set. Discard the indexes that have coefficient values less than 0.3.
5. **Evaluation**: Select the index which has the maximum correlation coefficient with the input matrix.
6. Output the selected digits

4.2 Input of handwritten Kayin Digit (0 to 9)

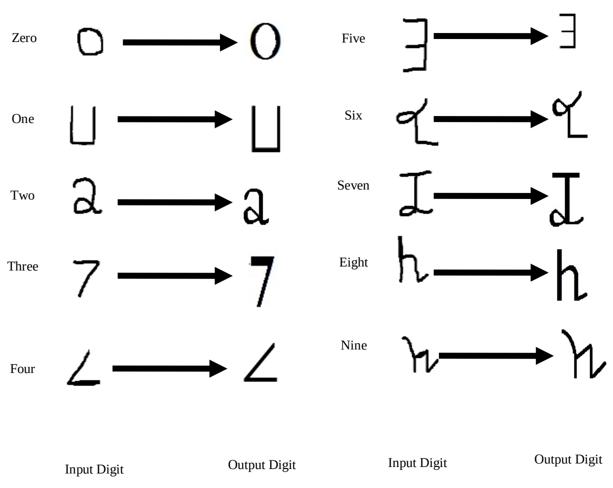


4.2.1 Output of the System (0 to 9)

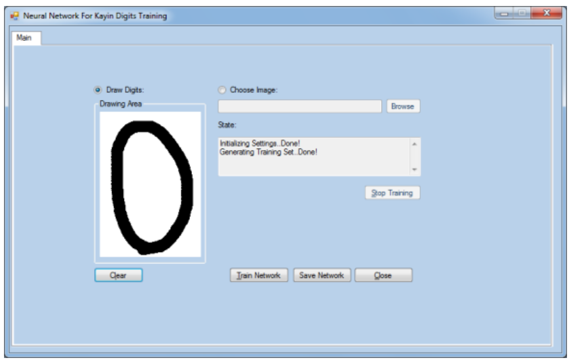
Input (handwritten) output Input (handwritten) output



4.2.2 Kayin Input Digit and Output Digit of the System



5. Implementation



Initializing Settings..Done!
Generating Training Set..Done!

Figure 3. Training Digit

User can train hand written digit or digit image from this page. User draws digit and trains the digit image. Then user saves the digit image.

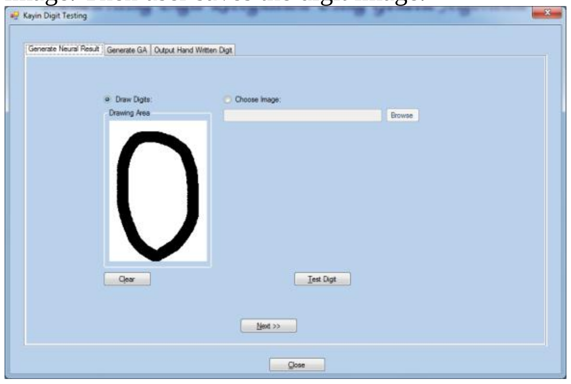
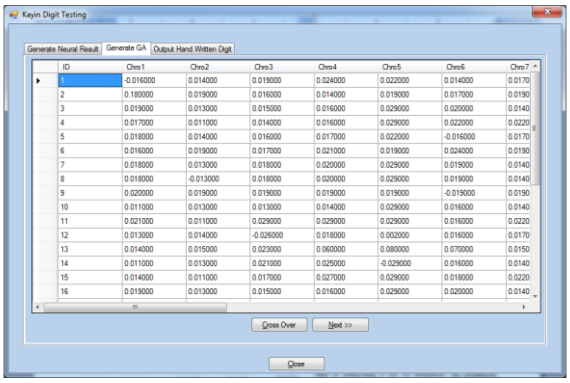


Figure 4 . Testing Digit

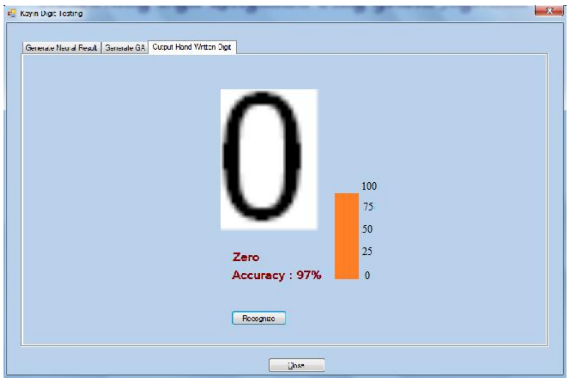
In this page, user can test hand written digit or digit image. User draws digit and click “Test Digit” button. The system generates this image and outputs the result. This output result is set into genetic algorithm (GA) as chromosomes.



	ID	Chro1	Chro2	Chro3	Chro4	Chro5	Chro6	Chro7
▶	1	-0.016000	0.014000	0.019000	0.024000	0.022000	0.014000	0.0170
	2	0.180000	0.019000	0.016000	0.014000	0.019000	0.017000	0.0190
	3	0.019000	0.013000	0.015000	0.016000	0.029000	0.020000	0.0140
	4	0.017000	0.011000	0.014000	0.016000	0.029000	0.022000	0.0220
	5	0.018000	0.014000	0.016000	0.017000	0.022000	-0.016000	0.0170
	6	0.016000	0.019000	0.017000	0.021000	0.019000	0.024000	0.0190
	7	0.018000	0.013000	0.018000	0.020000	0.029000	0.019000	0.0140
	8	0.018000	-0.013000	0.018000	0.020000	0.029000	0.019000	0.0140
	9	0.020000	0.019000	0.019000	0.019000	0.019000	-0.019000	0.0190
	10	0.011000	0.013000	0.013000	0.014000	0.029000	0.016000	0.0140
	11	0.021000	0.011000	0.029000	0.029000	0.029000	0.016000	0.0220
	12	0.013000	0.014000	-0.026000	0.018000	0.002000	0.016000	0.0170
	13	0.014000	0.015000	0.023000	0.060000	0.080000	0.070000	0.0150
	14	0.011000	0.013000	0.021000	0.025000	-0.029000	0.016000	0.0140
	15	0.014000	0.011000	0.017000	0.027000	0.029000	0.018000	0.0220
	16	0.019000	0.013000	0.015000	0.016000	0.029000	0.020000	0.0140

Figure 5. Crossover

GA crossovers these chromosomes and finally generates the best result.



Zero
Accuracy : 97%

Figure 6. Generating Output Digit

The output digit and accuracy are displayed in this page.

6. Experimental Results

Digit	Image	Accuracy
One		97%
Two		92%
Three		92%
Four		94%
Five		89%
Six		93%
Seven		91%
Eight		94%
Nine		86%
Ten		98%

Table (1) Accuracy for Each Digit

Digit	Image	Accuracy	
		Neural Network	Genetic Algorithm
One		89%	97%
Two		82%	92%
Three		83%	92%
Four		86%	94%
Five		81%	89%
Six		85%	93%
Seven		83%	91%
Eight		86%	94%
Nine		80%	86%
Ten		90%	98%

Table (2) Accuracy Comparison for Each Digit of Neural Network and Genetic Algorithm

7. Output of Neural Network and Input of Genetic Algorithm

C1	- 0.016	0.014	0.019	0.024	0.022	0.014	0.017	0.018	0.017	0.024
C2	0.18	0.019	0.016	0.014	0.019	0.017	0.019	0.019	0.019	0.016
C3	0.019	0.013	0.015	0.016	0.029	0.02	0.014	0.017	0.017	0.07
C4	0.017	0.011	0.014	0.016	0.029	0.022	0.022	0.018	0.015	0.012
C5	0.018	0.014	0.016	0.017	0.022	- 0.016	0.017	0.018	0.017	0.024
C6	0.016	0.019	0.017	0.021	0.019	0.024	0.019	0.019	0.019	0.016
C7	0.018	0.013	0.018	0.02	0.029	0.019	0.014	0.017	0.017	0.07
C8	0.018	-0.013	0.018	0.02	0.029	0.019	0.014	0.017	0.017	0.07
C9	0.02	0.019	0.019	0.019	0.019	- 0.019	0.019	0.019	0.019	0.016
C10	0.011	0.013	0.013	0.014	0.029	0.016	0.014	0.017	0.017	0.07

8. Conclusion

Using neural networks and genetic algorithm for handwriting digit recognition is a field that is attracting a lot of attention. As technical advances are made, the benefits of using neural network and genetic algorithm for the purpose of handwriting digit recognition become clearer. This digit recognition starts with acquiring the image to be preprocessed throw a number of steps. As an important point, classification and recognition have to be done to get the outputs that are chromosomes of genetic algorithm. Genetic algorithm has been applied to get the clear selected digit.

8.1 Limitation of the System

There are some limitations in this system because it is only considered for handwritten Kayin digit. This system can be able to recognize not only for user handwritten but also for manual draw Kayin digit. This system can be decided 10 Kayin digits (0 to 9).

7.2 Further Extension

This system can be extended into the capabilities and efficiency of Data Mining System. This system has been tested on Kayin Handwritten digit. This system used feedforward neural network and genetic algorithm for output results. Other measures with the similar properties could be used for further study.

This system can extend other handwritten digit and used different method for produce patterns.

REFERENCES

[1] H. El Fadili, K. Zenkouar and H. Qjidaa, "Lapped Block Image Analysis Via the Method of Legendre Moments," *EURASIP* Journal on Applied Signal Processing, vol. 2003, no.9, pp. 902-913, August 2003.

[2] E. Kussul, T. Baidyk, D. Wunsch, O. Makeyev and A. Martin, 2006. Image recognition systems based on random local descritors. Proceeding of the International Joint Conference on Neural Networks, July 16-21, IEEE Xplore Press, USA., pp:2415-2420.DOI: 0.1109/IJCNN.2006.247067

[3] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no.11, pp. 2278-2324, November 1998.

[4] S. X. Liao and Miroslaw Pawlak, "On image analysis by moments," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 18, no. 3, pp. 254-266, 1996.

[5] G. Miller P. M. Todd and S. U. Hegde, Designing Neural Networks using Genetic Algorithms, Proc. Of the third Intern. Conference on Genetic Algorithms (ICGA), San Mateo (CA), 1989, pp. 379-384.

[6] H. Qjidaa and L. Redouane, "Robust line fitting in a noisy image by the method of moments," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 21, no. 11, pp. 1216-1223, 1999.

[7] C. Rafael Gonzalez, E. Richard Woods and L. Steven Eddins, 2003. Digital Image Processing using MATLAB. 2nd Edn., Prentice Hall, USA, ISBN: 10: 0130085197, pp: 624.

[8]Shashank Mathur, Vaibhav Aggarwal, Himanshu Joshi, Anil Ahlawat ,"Offline Handwriting Recognition using GENETIC ALGORITHM"

[9] W. H. Schiffmann and K. Mecklenburg, "Genetic Generation of Backpropagation Trained Neural Networks," Proc. of Parallel Processing in Neural Systems and Computers(ICNC), Eckmiller R. et al. (Eds.) pp. 205-208, Elsevier, 1990.

[10] J.M Westall and M.S. Narasimhsa, 1997. An evolutionary approach to the use of neural networks in the segmentation of handwritten numerals. Int. J. Patt. Recog. Artifi. Intell. 11: 717-735.