CHARACTER RECOGNITION USING NEURAL NETWORKS

Fakhraddin Mamedov, Jamal Fathi Abu Hasna
fahri@neu.edu.tr, jamalfathi2004@yahoo.com
Near East University, North Cyprus, Turkey via Mersin-10, KKTC

ABSTRACT

This paper presents creating the Character Recognition System, in which Creating a Character Matrix and a corresponding Suitable Network Structure is key. In addition, knowledge of how one is Deriving the Input from a Character Matrix must first be obtained before one may proceed. Afterwards, the Feed Forward Algorithm gives insight into the enter workings of a neural network; followed by the Back Propagation Algorithm which compromises Training, Calculating Error, and Modifying Weights. A problem encountered with a 20 x 20 matrix is time-consuming training, which is conquered with Manual Compression.

1. Introduction

One of the most classical applications of the Artificial Neural Network is the Character Recognition System. This system is the base for many different types of applications in various fields, many of which we use in our daily lives. Cost effective and less time consuming, businesses, post offices, banks, security systems, and even the field of robotics employ this system as the base of their operations. Wither you are processing a check, performing an eye/face scan at the airport entrance, or teaching a robot to pick up and object, you are employing the system of Character Recognition.

One field that has developed from Character Recognition is Optical Character Recognition (OCR). OCR is used widely today in the post offices, banks, airports, airline offices, and businesses. Address readers sort incoming and outgoing mail, check readers in banks capture images of checks for processing, airline ticket and passport readers are used for various purposes from accounting for passenger revenues to checking database records, and form readers are used to read and process up to 5,800 forms per hour. OCR software is also used in scanners and faxes that allow the user to turn graphic images of text into editable documents. Newer applications have even expanded outside the limitations of just characters. Eye, face, and fingerprint scans used in high-security areas employ a newer kind of recognition. More and more assembly lines are becoming equipped with robots scanning the gears that pass underneath for faults, and it has been applied in the field of robotics to allow robots to detect edges, shapes, and colors.

Optical Character Recognition has even advanced into a newer field - Handwritten Recognition, which of course is also based on the simplicity of Character Recognition. The new idea for computers, such as Microsoft's new Tablet Pc, is pen-based computing, which employs lazy recognition that runs the character recognition system silently in the background instead of in real time.

2. Creating the Character Recognition System

The Character Recognition System must first be created through a few simple steps in order to prepare it for presentation into MATLAB. The matrixes of each letter of the alphabet must be created along with the network structure. In addition, one must understand how to pull the Binary Input Code from the matrix, and how to interpret the Binary Output Code, which the computer ultimately produces.

Character Matrixes

A character matrix is an array of black and white pixels; the vector of 1 represented by black, and 0 by white. They are created manually by the user, in whatever size or font imaginable; in addition, multiple fonts of the same alphabet may even be used under separate training sessions.

Creating a Character Matrix

First, in order to endow a computer with the ability to recognize characters, we must first create those characters. The first thing to think about when creating a matrix is the size that will be used. Too small and all the letters may not be able to be created, especially if you want to use two different fonts. On the other hand, if the size of the matrix is very big, their may be a few problems: Despite the fact that the speed of computers double every third year, their may not be enough processing power currently available to run in real time. Training may take days, and results may take hours. In addition, the computer's memory may not be able to handle enough neurons in the hidden layer needed to efficient and accurately process the information. However, the number of neurons may just simply be reduced, but this in turn may greatly increase the chance for error.

A large matrix size of 20 x 20 was created, through the steps as explained above, because it may not be able to process in real time. (See Figure 1) Instead of waiting a few more years until computers have again increased in their processing speed, another solution has been presented which will be explained fully at the end of this paper.

First Font	Second Font
00000000000000000000	00000000000000000000
00000000011000000000	00111111111111111100
00000000111100000000	00111111111111111100
00000001111110000000	00111111111111111100
00000011111111000000	00111000000000011100
00000111000011100000	00111000000000011100
00001110000001110000	00111000000000011100
00011100000000111000	00111000000000011100
00111000000000011100	00111111111111111100
00111000000000011100	00111111111111111100
00111000000000011100	00111000000000011100
00111111111111111100	00111000000000011100
00111111111111111100	00111000000000011100
00111000000000011100	00111000000000011100
00111000000000011100	00111000000000011100
00111000000000011100	00111000000000011100
00111000000000011100	00111000000000011100
00111000000000011100	00111000000000011100
00111000000000011100	00111000000000011100
00111000000000011100	00000000000000000000

Figure 1 Character Matrix A of Different Fonts

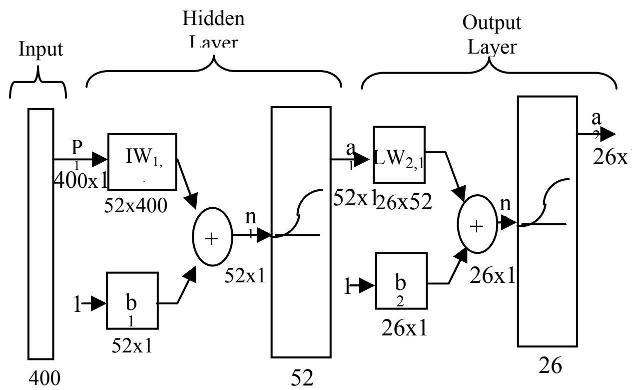
3. Neural Network

The network receives the 400 Boolean values as a 400-element input vector. It is then required to identify the letter by responding with a 26-element output vector. The 26 elements of the output vector each represent a letter. To operate correctly, the network should respond with a 1 in the position of the letter being presented to the network. All other values in the output vector should be 0. In addition, the network should be able to handle noise. In practice, the network does not receive a perfect Boolean vector as input. Specifically, the network should make as few mistakes as possible when classifying vectors with noise of mean 0 and standard deviation of 0.2 or less.

4. Architecture

The neural network needs 400 inputs and 26 neurons in its output layer to identify the letters. The network is a two-layer log-sigmoid/log-sigmoid network. The log-sigmoid transfer function was picked because its output range (0 to 1) is perfect for learning to output Boolean values.

The hidden (first) layer has 52 neurons. This number was picked by guesswork and experience. If the network has trouble learning, then neurons can be added to this layer. The network is trained to output a 1 in the correct position of the output vector and to fill the rest of the output vector with 0's. However, noisy input vectors may result in the network not creating perfect 1's and 0's. After the network is trained the output is passed through the competitive transfer function compet. This makes sure that the output corresponding to the letter most like the noisy input vector takes on a value of 1, and all others have a value of 0. The result of this post-processing is the output that is actually used.



$$a^1=\text{logsig}(IW_{1,1}P^1 + b^1) \quad a^2=\text{logsig}(LW_{2,1}a^1 + b^2)$$

Figure 2. Neural Network Architecture

5. Setting the Weights

There are two sets of weights; input-hidden layer weights and hidden-output layer weights. These weights represent the memory of the neural network, where final training weights can be used when running the network.

Initial weights are generated randomly there, after; weights are updated using the error (difference) between the actual output of the network and the desired (target) output. Weight updating occurs each iteration, and the network learns while iterating repeatedly until a net minimum error value is achieved.

First we must define notion for the patterns to be stored Pattern p. a vector of 0/1 usually binary–valued. Additional layers of weights may be added but the additional layers are unable to adopt. Inputs arrive from the left and each incoming interconnection has an associated weight, w_{ji}. The perception processing unit performs a weighted sum at its input value.

The sum takes the form net $= \sum_{i=1}^{n} O_i W_i$

Weights associated with each inter connection are adjusted during learning .The weight to unit J from unit j from unit I is denoted as w_i after learning is completed the weights are fixed from 0 to 1.

There is a matrix of weight values that corresponds to each layer at inter connections as shown in figure 3.

These matrices are indexed with superscripts to distinguish weights in different layers.

$$Weights\left\{\begin{array}{l}
0.2,0.3,0.4,0.5,0.6,0.7,0.9,0.2,0.4,0.5,0.9,0.7,0.2,0.3,0.2,0.9,0.0,0.4,0.7,0.8\\
0.2,0.3,0.2,0.9,0.0,0.4,0.7,0.8,0.2,0.4,0.5,0.9,0.7.4,0.7,0.8,0.2,0.4,0.0,3,0.2\\
0.4,0.5,0.6,0.7,0.8,0.2,0.4,0.5,0.9,0.9,0.0,0.4,0.7,0.8,0.2,0.4,0.5,0.9,0.7,0.8\\
0.2,0.9,0.0,0.4,0.7,0.8,0.2,0.4,0.5,0.9,0.7,0.4,0.8,0.0,0.7,0.5,0.1,0.2,0.5,0.8\\
0.4,0.7,0.8,0.2,0.4,0.5,0.9,0.7,0.7,0.8,0.2,0.4,0.0,0.3,0.2,0.5,0.6,0.4,0.7,0.9\\
0.4,0.5,0.6,0.7,0.8,0.2,0.4,0.5,0.9,0.9,0.0,0.4,0.7,0.8,0.7,0.5,0.1,0.2,0.5,0.8\\
0.2,0.3,0.2,0.9,0.0,0.4,0.7,0.8,0.2,0.4,0.5,0.9,0.7.4,0.7,0.8,0.2,0.4,0.0,3,0.2\\
0.4,0.5,0.6,0.7,0.8,0.2,0.4,0.5,0.9,0.9,0.0,0.4,0.7,0.8,0.2,0.4,0.5,0.9,0.7,0.8\\
0.4,0.7,0.8,0.2,0.4,0.5,0.9,0.7,0.7,0.8,0.2,0.4,0.0,0.3,0.2,0.5,0.6,0.4,0.7,0.9\\
0.2,0.3,0.4,0.5,0.6,0.7,0.9,0.2,0.4,0.5,0.9,0.7,0.2,0.3,0.2,0.9,0.0,0.4,0.7,0.8\\
0.2,0.3,0.4,0.5,0.6,0.7,0.9,0.2,0.4,0.5,0.9,0.7,0.2,0.3,0.2,0.9,0.0,0.4,0.7,0.8\\
0.4,0.5,0.6,0.7,0.8,0.2,0.4,0.5,0.9,0.9,0.0,0.4,0.7,0.8,0.2,0.4,0.5,0.9,0.7,0.8\\
0.2,0.9,0.0,0.4,0.7,0.8,0.2,0.4,0.5,0.9,0.7,0.4,0.8,0.0,0.7,0.5,0.1,0.2,0.5,0.8\\
0.4,0.5,0.6,0.7,0.8,0.2,0.4,0.5,0.9,0.9,0.0,0.4,0.7,0.8,0.2,0.4,0.5,0.9,0.7,0.8\\
0.2,0.9,0.0,0.4,0.7,0.8,0.2,0.4,0.5,0.9,0.7,0.4,0.8,0.0,0.7,0.5,0.1,0.2,0.5,0.8\\
0.8,0.2,0.4,0.5,0.9,0.9,0.0,0.4,0.7,0.8,0.2,0.4,0.5,0.9,0.9,0.0,0.4,0.7,0.2,0.3\\
0.2,0.3,0.2,0.9,0.0,0.4,0.7,0.8,0.2,0.4,0.5,0.9,0.7,0.7,0.8,0.2,0.4,0.0,0.3,0.2\\
0.7,0.8,0.2,0.9,0.0,0.4,0.7,0.8,0.2,0.4,0.5,0.9,0.7,0.4,0.8,0.0,0.7,0.5,0.1,0.2\\
0.5,0.8,0.4,0.5,0.6,0.7,0.8,0.2,0.4,0.5,0.9,0.9,0.0,0.4,0.7,0.8,0.2,0.4,0.5,0.9\\
0.4,0.7,0.8,0.2,0.4,0.5,0.9,0.7,0.7,0.8,0.2,0.4,0.0,0.3,0.2,0.5,0.6,0.4,0.7,0.9
\end{array}\right\}$$

Figure 3 A Matrix of Weight Values

6. Training

To create a network that can handle noisy input vectors it is best to train the network on both ideal and noisy vectors. To do this, the network is first trained on ideal vectors until it has a low sum-squared error. Then, the network is trained on all sets of ideal and noisy vectors. The network is trained on two copies of the noise-free alphabet at the same time as it is trained on noisy vectors. The two copies of the noise-free alphabet are used to maintain the network's ability to classify ideal input vectors. Unfortunately, after the training described above the network may have learned to classify some difficult noisy vectors at the expense of properly classifying a noise-free vector. Therefore, the network is again trained on just ideal vectors. This ensures that the network responds perfectly when presented with an ideal letter. All training is done using backpropagation with both adaptive learning rate and momentum with the function trainbpx.

6.1 Forward- pass

The forward – pass phase is initiated when an input pattern is presented to the network, each input unit corresponds to an entry in the input pattern vector, and each unit takes on the value of this entry.

Incoming connection to unit J are at the left and originate at units in the layer below.

The function F(x), a sigmoid curve is illustrated as shown in figure 4.

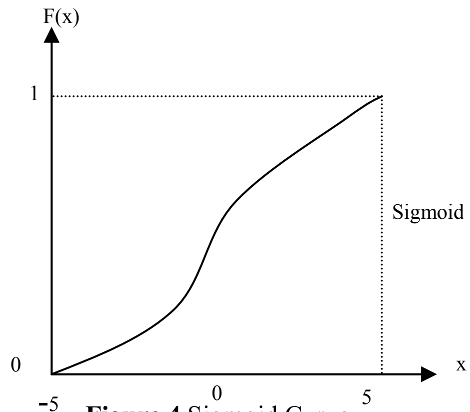


Figure 4 Sigmoid Curve

There is a transition from 0 to 1 that takes place when x is approximately $(-3 < \chi < 3)$ the sigmoid function performs assort at soft threshold that is rounded as shown in figure 5 bellow.

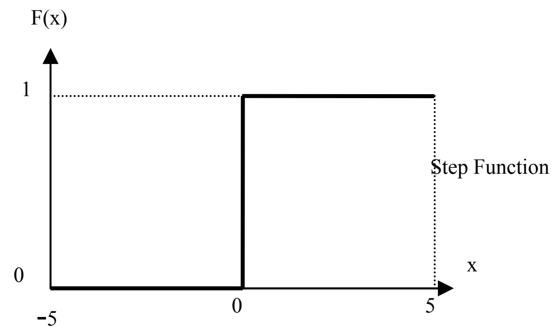


Figure 5 Step Function

The equation for the sigmoid function is

$$f(x) = \frac{1}{1 + e^{\sum_{i=1}^{-n} oi\ \mathrm{wi}}} \qquad (1)$$

a. Input layer (i)

For input we have 26 inputs will be saved by the DAT file. Input Layer at neuron = output layer of neuron $I_i = O_i$

b. Hidden layer (h)

Hidden–Layer input $h = I_k = \sum_i W_{ki} O_i$ as we have suggest that our weight is this and we are taking the value at our input at character is A*I*S. Where A is the input-matrix, I is the hidden-layer input matrix and S is the Sigmoid function matrix as shown in figure 6(a) and (b).

$$
Weights = W_{hi} \left\{ \begin{matrix}
0.2,0.3,0.4,0.5,0.6,0.7,0.9,0.2,0.4,0.5,0.9,0.7,0.2,0.3,0.2,0.9,0.0,0.4,0.7,0.8 \\
0.2,0.3,0.2,0.9,0.0,0.4,0.7,0.8,0.2,0.4,0.5,0.9,0.7.4,0.7,0.8,0.2,0.4,0.0,3,0.2 \\
0.4,0.5,0.6,0.7,0.8,0.2,0.4,0.5,0.9,0.9,0.0,0.4,0.7,0.8,0.2,0.4,0.5,0.9,0.7,0.8 \\
0.2,0.9,0.0,0.4,0.7,0.8,0.2,0.4,0.5,0.9,0.7,0.4,0.8,0.0,0.7,0.5,0.1,0.2,0.5,0.8 \\
0.4,0.7,0.8,0.2,0.4,0.5,0.9,0.7,0.7,0.8,0.2,0.4,0.0,0.3,0.2,0.5,0.6,0.4,0.7,0.9 \\
0.4,0.5,0.6,0.7,0.8,0.2,0.4,0.5,0.9,0.9,0.0,0.4,0.7,0.8,0.7,0.5,0.1,0.2,0.5,0.8 \\
0.2,0.3,0.2,0.9,0.0,0.4,0.7,0.8,0.2,0.4,0.5,0.9,0.7.4,0.7,0.8,0.2,0.4,0.0,3,0.2 \\
0.4,0.5,0.6,0.7,0.8,0.2,0.4,0.5,0.9,0.9,0.0,0.4,0.7,0.8,0.2,0.4,0.5,0.9,0.7,0.8 \\
0.4,0.7,0.8,0.2,0.4,0.5,0.9,0.7,0.7,0.8,0.2,0.4,0.0,0.3,0.2,0.5,0.6,0.4,0.7,0.9 \\
0.2,0.3,0.4,0.5,0.6,0.7,0.9,0.2,0.4,0.5,0.9,0.7,0.2,0.3,0.2,0.9,0.0,0.4,0.7,0.8 \\
0.2,0.3,0.4,0.5,0.6,0.7,0.9,0.2,0.4,0.5,0.9,0.7,0.2,0.3,0.2,0.9,0.0,0.4,0.7,0.8 \\
0.4,0.5,0.6,0.7,0.8,0.2,0.4,0.5,0.9,0.9,0.0,0.4,0.7,0.8,0.2,0.4,0.5,0.9,0.7,0.8 \\
0.2,0.9,0.0,0.4,0.7,0.8,0.2,0.4,0.5,0.9,0.7,0.4,0.8,0.0,0.7,0.5,0.1,0.2,0.5,0.8 \\
0.4,0.5,0.6,0.7,0.8,0.2,0.4,0.5,0.9,0.9,0.0,0.4,0.7,0.8,0.2,0.4,0.5,0.9,0.7,0.8 \\
0.2,0.9,0.0,0.4,0.7,0.8,0.2,0.4,0.5,0.9,0.7,0.4,0.8,0.0,0.7,0.5,0.1,0.2,0.5,0.8 \\
0.8,0.2,0.4,0.5,0.9,0.9,0.0,0.4,0.7,0.8,0.2,0.4,0.5,0.9,0.9,0.0,0.4,0.7,0.2,0.3 \\
0.2,0.3,0.2,0.9,0.0,0.4,0.7,0.8,0.2,0.4,0.5,0.9,0.7,0.7,0.8,0.2,0.4,0.0,0.3,0.2 \\
0.7,0.8,0.2,0.9,0.0,0.4,0.7,0.8,0.2,0.4,0.5,0.9,0.7,0.4,0.8,0.0,0.7,0.5,0.1,0.2 \\
0.5,0.8,0.4,0.5,0.6,0.7,0.8,0.2,0.4,0.5,0.9,0.9,0.0,0.4,0.7,0.8,0.2,0.4,0.5,0.9 \\
0.4,0.7,0.8,0.2,0.4,0.5,0.9,0.7,0.7,0.8,0.2,0.4,0.0,0.3,0.2,0.5,0.6,0.4,0.7,0.9
\end{matrix} \right\}
$$

(a)

$$
A\ is\ Input = O_i \Rightarrow \left\{ \begin{matrix}
00000000000000000000 \\
00000000011000000000 \\
00000000111100000000 \\
00000001111110000000 \\
00000011111111000000 \\
00000111000011100000 \\
00001110000001110000 \\
00011100000000111000 \\
00111000000000011100 \\
00111000000000011100 \\
00111000000000011100 \\
00111111111111111100 \\
00111111111111111100 \\
00111000000000011100 \\
00111000000000011100 \\
00111000000000011100 \\
00111000000000011100 \\
00111000000000011100 \\
00111000000000011100 \\
00111000000000011100
\end{matrix} \right\}
$$

(b)

Figure 6 (a) Sigmoid function matrix, (b) Input Character A.

I_h=∑(0.2x0)+(0.3x0)+(0.4x0)+(0.5x0)+(0.6x0)+(0.7x0)+(0.9x0)+(0.2x0)+(0.4x0)+(0.5x0)+(0.9x0)+(0.7x0)+(0.4x0)+(0.7x0)+(0.8x0)+(0.2x0)+(0.4x0)+(0.0x0)+(0.3x0)+(0.2x0)+(2x0)+(0.3x0)+(0.2x0)+(0.9x0)+(0.0x0)+(0.4x0)+(0.7x0)+(0.8x0)+(0.2x0)+(0.4x1)+(0.5x1)+(0.9x0)+(0.7x0)+(0.4x0)+(0.7x0)+(0.8x0)+(0.2x0)+(0.4x0)+(0.0x0)+(0.2x0)
…………………………………………………………
………………………………………………
(0.4x0)+(0.7x0)+(0.8x1)+(0.2x1)+(0.4x1)+(0.5x0)+(0.9x0)+(0.7x0)+(0.7x0)+(0.8x0)+(0.2x0)+(0.4x0)+(0.0x0)+(0.3x0)+(0.2x0)+(0.5x1)+(0.6x1)+(0.4x1)+(0.7x0)+(0.9x0)}

So after calculation we will get the value of I_h.

Each output at a hidden neuron is calculated using the sigmoid function.

Hidden-layer output $h = o_h = \frac{1}{1+e^{-I_h}}$

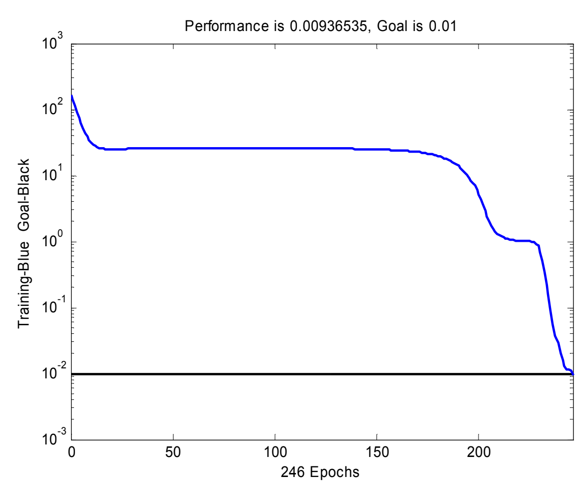
This calculation is for one neuron and the summation for the other output layer (j).

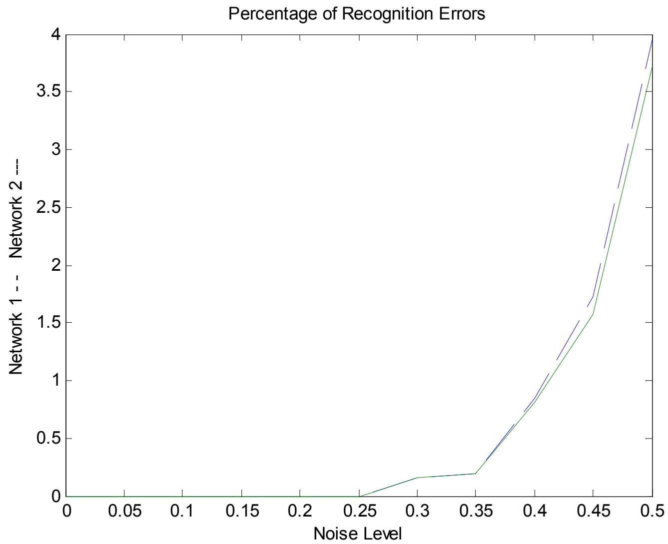
7. System Performance and Results

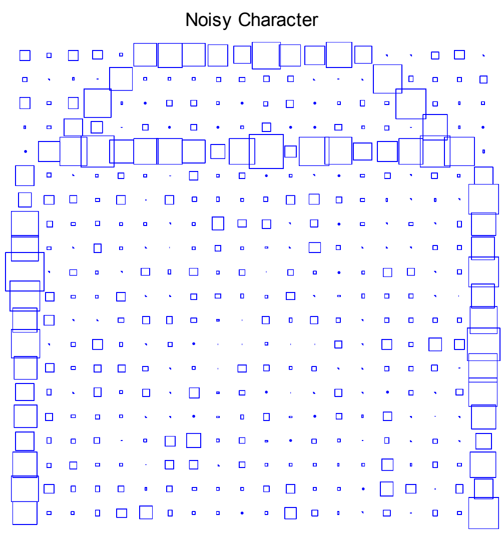
The reliability of the neural network pattern recognition system is measured by testing the network with hundreds of input vectors with varying quantities of noise. The script file appcr1 tests the network at various noise levels, and then graphs the percentage of network errors versus noise. Noise with a mean of 0 and a standard deviation from 0 to 0.5 is added to input vectors. At each noise level, 100 presentations of different noisy versions of each letter are made and the network's output is calculated. The output is then passed through the competitive transfer function so that only one of the 26 outputs (representing the letters of the alphabet), has a value of 1.

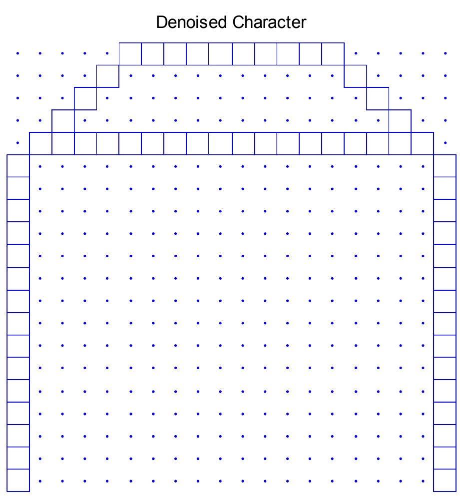
The above input was trained, and tested with the following specifications

Learning Rate	Maximum Epochs	Reached Epochs	Alpha	Goal	Result
0.75	5000	584	0.04	0.01	Performance Goal met
0.75	5000	246	0.03	0.01	Performance Goal met









8. Conclusion

This problem demonstrates how a simple pattern ecognition system can be designed. Note that the raining process did not consist of a single call to a raining function. Instead, the network was trained everal times on various input vectors. In this case, raining a network on different sets of noisy vectors forced the network to learn how to deal with noise, a common problem in the real world.

9. References

[1]. Pratt, William K. Digital Image Processing. New York: John Wiley & Sons, Inc., 1991. p. 634.

[2]. Horn, Berthold P. K., Robot Vision. New York: McGraw-Hill, 1986. pp. 73-77.

[3]. Pratt, William K. Digital Image Processing. New York: John Wiley & Sons, Inc., 1991. p. 633.

[4]. Haralick, Robert M., and Linda G. Shapiro. Computer and Robot Vision, Volume I. Addison-Wesley, 1992.

[5]. Ardeshir Goshtasby, Piecewise linear mapping functions for image registration, Pattern Recognition, Vol 19, pp. 459-466, 1986.

[6]. Ardeshir Goshtasby, Image registration by local approximation methods, Image and Vision Computing, Vol 6, p. 255-261, 1988.

[7]. Jain, Anil K. Fundamentals of Digital Image Processing. Englewood Cliffs, NJ: Prentice Hall, 1989. pp. 150-153.

[8]. Pennebaker, William B., and Joan L. Mitchell. JPEG: Still Image Data Compression Standard. Van Nostrand Reinhold, 1993.

[9]. Gonzalez, Rafael C., and Richard E. Woods. Digital Image Processing. Addison-Wesley, 1992. p. 518.

[10]. Haralick, Robert M., and Linda G. Shapiro. Computer and Robot Vision, Volume I. Addison-Wesley, 1992. p. 158.

[11]. Floyd, R. W. and L. Steinberg. "An Adaptive Algorithm for Spatial Gray Scale," International Symposium Digest of Technical Papers. Society for Information Displays, 1975. p. 36.

[12]. Lim, Jae S. Two-Dimensional Signal and Image Processing. Englewood Cliffs, NJ: Prentice Hall, 1990. pp. 469-476.

[13]. Canny, John. "A Computational Approach to Edge Detection," IEEE Transactions on Pattern Analysis and Machine Intelligence, 1986. Vol. PAMI-8, No. 6, pp. 679-698.

[14]. Lim, Jae S. Two-Dimensional Signal and Image Processing. Englewood Cliffs, NJ: Prentice Hall, 1990. pp. 478-488.
[15]. Parker, James R. Algorithms for Image Processing and Computer Vision. New York: John Wiley & Sons, Inc., 1997. pp. 23-29.
[16]. Gonzalez, Rafael C., and Richard E. Woods. Digital Image Processing. Addison-Wesley, 1992. p. 518.
[17]. Haralick, Robert M., and Linda G. Shapiro. Computer and Robot Vision, Volume I. Addison-Wesley, 1992. p. 158.
[18]. Jain, Anil K. Fundamentals of Digital Image Processing. Englewood Cliffs, NJ: Prentice Hall, 1989. pp. 150-153.
[19]. Pennebaker, William B., and Joan L. Mitchell. JPEG: Still Image Data Compression Standard. New York: Van Nostrand Reinhold, 1993.
[20]. Robert M. Haralick and Linda G. Shapiro, Computer and Robot Vision, vol. I, Addison-Wesley, 1992, pp. 158-205.
[21]. van den Boomgaard and van Balen, "Image Transforms Using Bitmapped Binary Images," Computer Vision, Graphics, and Image Processing: Graphical Models and Image Processing, vol. 54, no. 3, May, 1992, pp. 254-258.
[22]. Kak, Avinash C., and Malcolm Slaney, Principles of Computerized Tomographic Imaging. New York: IEEE Press.
[23]. J. P. Lewis, "Fast Normalized Cross-Correlation", Industrial Light & Magic. http://www.idiom.com/~zilla/Papers/nvisionInterface/nip.html
[24]. Robert M. Haralick and Linda G. Shapiro, Computer and Robot Vision, Volume II, Addison-Wesley, 1992, pp. 316-317.
[25]. Bracewell, Ronald N. Two-Dimensional Imaging. Englewood Cliffs, NJ: Prentice Hall, 1995. pp. 505-537.
[26]. Lim, Jae S. Two-Dimensional Signal and Image Processing. Englewood Cliffs, NJ: Prentice Hall, 1990. pp. 42-45.
[27]. Rein van den Boomgard and Richard van Balen, "Methods for Fast Morphological Image Transforms Using Bitmapped Images," Computer Vision, Graphics, and Image Processing: Graphical Models and Image Processing, vol. 54, no. 3, May 1992, pp. 252-254.
[28]. Rolf Adams, "Radial Decomposition of Discs and Spheres," Computer Vision, Graphics, and Image Processing: Graphical Models and Image Processing, vol. 55, no. 5, September 1993, pp. 325-332.
[29]. Ronald Jones and Pierre Soille, "Periodic lines: Definition, cascades, and application to granulometrie," Pattern Recognition Letters, vol. 17, 1996, 1057-1063.

10. Some Results