Analysis of a Biologically Motivated Neural Network for Character Recognition

M. D. Garris, R. A. Wilkinson, and C. L. Wilson
Advanced Systems Division
National Institute of Standards and Technology,
Gaithersburg, MD 20899

Proceedings: Analysis of Neural Network Applications,
ACM Press, George Mason University, May 1991

Abstract

A neural network architecture for size-invariant and local shape-invariant digit recognition has been developed. The network is based on known biological data on the structure of vertebrate vision but is implemented using more conventional numerical methods for image feature extraction and pattern classification. The input receptor field structure of the network uses Gabor function feature selection. The classification section of the network uses back-propagation. Using these features as neurode inputs, an implementation of back-propagation on a serial machine achieved 100% accuracy when trained and tested on a single font size and style while classifying at a rate of 2 ms per character. Taking the same trained network, recognition greater than 99.9% accuracy was achieved when tested with digits of different font sizes. A network trained on multiple font styles when tested achieved greater than 99.9% accuracy and, when tested with digits of different font sizes, achieved greater than 99.8% accuracy. These networks, trained only with good quality prototypes, recognized images degraded with 15% random noise with an accuracy of 89%. In addition to raw recognition results, a study was conducted where activation distributions of correct responses from the network were compared against activation distributions of incorrect responses. By establishing a threshold between these two distributions, a reject mechanism was developed to minimize substitutional errors. This allowed substitutional errors on images degraded with 10% random noise to be reduced from 2.08% to 0.25%.

1.0 Introduction

Neural network methods show great promise for providing highly accurate, noise-resistant, parallel algorithms and data organization of image recognition. One specific area of image recognition, the conversion of images of hand written and machine print characters to computer representation, has been studied in detail in the past. Both special purpose hardware [1] and software [2] approaches have been used on the character recognition problem with promising results.

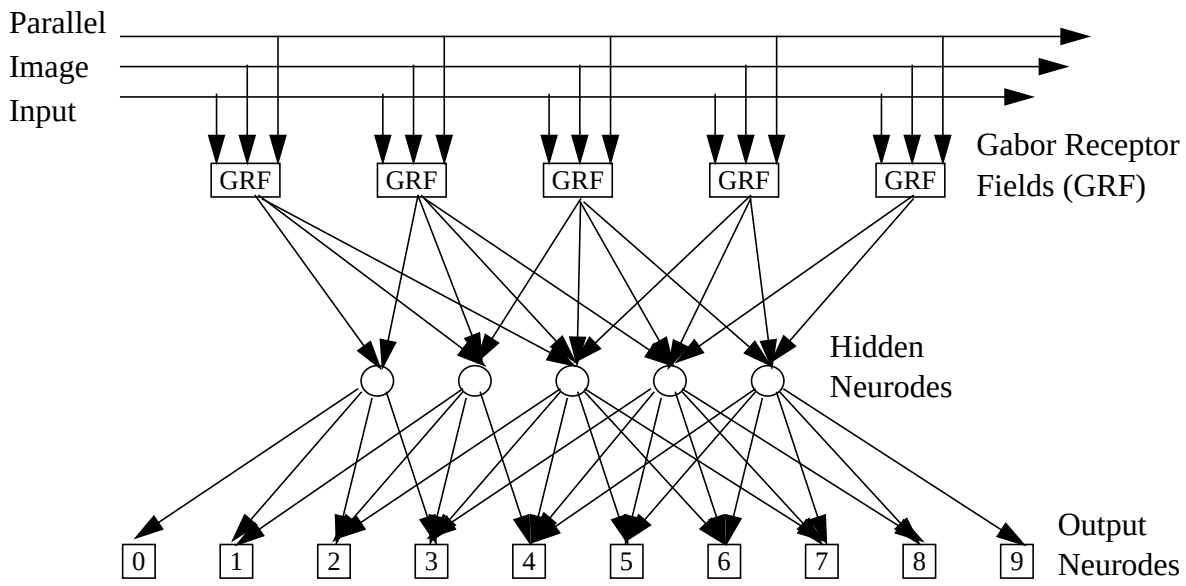

FIGURE 1. Layout of the combined network for feature detection and classification. For clarity, most connections are not shown.

The usual method for designing character recognition systems has been a top down approach. A method of feature extraction is selected and the features produced are used for training and classification by a neural network. In this work, we have taken a different approach. The general form of input receptor fields which are used in tasks such as binocular vision have been modeled using parallel Gabor functions [3]. The output of these receptor fields is coupled to small networks for detection of spatial position [4]. An approximation to this biological model was constructed as shown in Figure 1 and used for character recognition.

The model was not specifically designed for character recognition and could be taught any set of images which could be represented by the available set of Gabor functions. Gabor functions are well suited to character recognition because they allow reasonable quality image reconstruction with small numbers of basis functions. Image reconstruction using Walsh functions [5]of equivalent quality requires approximately 100 basis functions [2]. Any decrease in the basis function size reduces the number of connections needed in the classification network by the product of the basis function difference times the number of hidden nodes in the classification network.

The network in Figure 1 uses parallel processing for the feature extraction process. The image is transmitted to a set of Gabor receptor field (GRF) modules and the combined optimal response is determined by least squares.

This method of feature extraction is not known to be biological but is well understood in terms of conventional numerical processing. The outputs of the GRF's are the inputs to a multi-layered feed-forward network trained using back-propagation learning. The activation of the output nodes of this network are used to classify the input images.

1.1 Previous Work

Both of the methods used in this paper [6,7] have been used for neural network image processing and recognition applications but have not previously been applied together for character recognition applications.

1.2 Gabor Function Image Representation

A set of incomplete nonlinear functions, Gabor functions, is used for input image feature extraction. These functions reduce random image noise and smooth irregularities in image structure by acting as spatially localized low-pass filters. John Daugman [6] has used Gabor functions for image compression and image texture analysis. In his paper [6], the motivation for use of incomplete functions, which add considerable complexity to the representation problem, is discussed in detail. The most important considerations for character feature extraction applications are

that the Gabor functions provide the minimum combination of uncertainty in position and spatial frequency resolution[6], that the profiles of Gabor functions match the visual receptor field profiles of mammalian eyes[6], and that the resulting feature extraction process is well suited to the parallel data mapping of an array processor. For machine printed character recognition applications, an approximate image reconstruction using 16 8-bit Gabor coefficients is sufficient to represent a 1024 pixel image.

The most important of these justifications is the biological one. Characters in all languages have been developed so that they can be distinguished by human vision. The known properties of the vision system can provide vital clues about the recognition process used to distinguish characters. The Gabor function approach provides a mathematically concise method for performing feature extraction on image data.

1.3 Back-Propagation

Back-propagation networks provide a very effective method for performing supervised nonlinear classification [7]. As is common in pattern classification problems, the efficiency of the network is strongly affected by the quality of the input features used to train the network. In back-propagation networks, the number of weights in the fully connected network increases as the product of the number of the input neurodes and the hidden neurodes plus the product of the hidden neurodes and the output neurodes. In the present network, effective character recognition can be performed with as few as five hidden nodes maintaining a reasonable balance between network connectivity and required computation.

1.4 Outline of Paper

The character recognition process presented in this paper uses two steps. First, each character image has features extracted using a Gabor image reconstruction. Next, the feature values are sent to a back-propagation module and the classification of the characters is determined by the activation strength which has been learned from the training images. Finally, a test sample of characters, different from the original character set, is processed and compared to the learned images to measure the effectiveness of the recognition.

In section 2, the construction of adaptive least squares optimal feature extraction, which uses matrices of Gabor functions as independent basis functions, is discussed. Section 3 discusses the construction of the back-propagation network. In section 4, the experimental methods for training and recognition testing are discussed. And finally in section 5, results of character classification experiments with varying amounts of noise and with different combinations of fonts are presented.

2.0 Least Squares Feature Extraction of Character Images

The information presented in this section was originally documented by Charles Wilson [3] and has been incorporated into the work represented in this paper.

2.1 Normalization of Character Images

The majority of segmented character images used in this work were originally produced on a laser printer, and all were digitized at 300 dots per inch binary. Spatial normalization is used to provide limited scale invariance to each character image. Each image initially is represented by a picture area greater than 32 by 32 pixels. In the normalization process the active image area is scaled so that the largest dimension of the image is 32 pixels, and centered so that the image has equal numbers of empty pixels on either side of the image in the other dimension. This process either replicates pixels to enlarge the image or it deletes pixels to reduce image size. The aspect ratio of the image is maintained as accurately as possible.

The SIMD (Single Instruction Multiple Data) architecture used for this study was a Active Memory Technology 510 Distributed Array Processor, DAP510[1]. This machine consists of a 32 by 32 grid of 1 bit processor elements. This processor configuration is capable of performing 10^{10} binary operations per second and is well suited for both vector and matrix operations. By using this highly parallel computer, typical normalization time is 730 µs per character.

2.2 Gabor Functions

The Gabor feature extraction is accomplished using a least squares fit of each image on the same parallel computer that was used for spatial normalization. The kernel functions used are Gabor functions. The least squares fitting of the image reconstruction is necessitated by the non-or-

1. Certain commercial equipment may be identified in order to adequately describe the subject matter of this work. In no case does such identification imply endorsement by the National Institute of Standards and Technology.

thogonal nature of the Gabor functions. Adopting the convention that bold upper case variables represent array processor matrices and bold lower case variables represent array processor vector data types, the Gabor functions are defined as:

$$\mathbf{G}_j(\mathbf{X},\mathbf{Y}) = \exp(-\mathbf{R}^2)\left\{\begin{matrix}\sin(\omega_j\mathbf{X}')\\ \cos(\omega_j\mathbf{X}')\end{matrix}\right. \quad (1)$$

where the matrix variables **R**, **X**', and **Y**' are given by:

$$\mathbf{R}^2 = (\mathbf{X}'^2 + \mathbf{Y}'^2)/\sigma_j^2 \quad (2)$$

$$\begin{bmatrix}\mathbf{X}'\\ \mathbf{Y}'\end{bmatrix} = \mathrm{T}\begin{bmatrix}\mathbf{X}-\mathrm{x}_{0j}\\ \mathbf{Y}-\mathrm{y}_{0j}\end{bmatrix} \quad (3)$$

and T[] is a scalar transformation matrix function. The X and Y matrices in the array processor are row and column expanded in the form:

$$\mathbf{X} = \begin{bmatrix} x_1 & x_2 & \cdots & x_{32}\\ x_1 & x_2 & \cdots & x_{32}\\ & & \cdots & \\ x_1 & x_2 & \cdots & x_{32}\end{bmatrix} \quad (4)$$

$$\mathbf{Y} = \begin{bmatrix} y_1 & y_1 & \cdots & y_1\\ y_2 & y_2 & \cdots & y_2\\ & & \cdots & \\ y_{32} & y_{32} & \cdots & y_{32}\end{bmatrix} \quad (5)$$

A typical scalar transformation applied to each element of the matrix is a rotation of the form:

$$\mathrm{T} = \begin{bmatrix}\cos\theta_j & \sin\theta_j\\ -\sin\theta_j & \cos\theta_j\end{bmatrix} \quad (6)$$

The matrix function $\mathbf{G}_j$ is then expressed as a function of the scalar variables: ω_j, which is the spatial frequency of the function; σ_j, the spatial extent of the function; (x_{0j},y_{0j}), the origin of the function; and θ_j, the orientation of the function.

2.3 Tiling

Since the Gabor basis functions are an infinite set, it is necessary to select a specific subset of them to be used as the image reconstruction elements which cover the character image. This selection process is referred to as tiling the image and the functions defining this process are listed in Table 1.

For the class of image reconstruction discussed here, each set of image origins has twice the sample density of the previous level and the number of directions selected, n_θ, is fixed. This results in an image reconstruction with directional sensitivity and positional sensitivity determined by the choice of the level parameter, *i*. The character images used in this study are 32 by 32 so that using large values of *i* would result in massive over-sampling of the image. The Gabor image reconstruction for the lowest value of *i*, on the other hand, is approximately a directional bar detector and adds little to the image reconstruction's spatial resolution. At each level the frequency and spatial resolutions, ω_i and σ_i, are adjusted to allow small overlaps in extent and provide octave spatial frequency response.

i	x_{0i}	y_{0i}	ω_i	σ_i	θ_i
1	$\frac{d_0}{2}$	$\frac{d_0}{2}$	$\frac{2\pi}{d_0}$	d_0	$\frac{2\pi}{n_\theta}, \frac{4\pi}{n_\theta}, \ldots, 2\pi$
2	$\frac{d_1}{2}, \frac{3d_1}{2}$	$\frac{d_1}{2}, \frac{3d_1}{2}$	$\frac{2\pi}{d_1}$	d_1	$\frac{2\pi}{n_\theta}, \frac{4\pi}{n_\theta}, \ldots, 2\pi$
…					
n	$\frac{d_n}{2}, \frac{3d_n}{2}, \ldots, \frac{(2^n-1)\,d_n}{2}$	$\frac{d_n}{2}, \frac{3d_n}{2}, \ldots, \frac{(2^n-1)\,d_n}{2}$	$\frac{2\pi}{d_n}$	d_n	$\frac{2\pi}{n_\theta}, \frac{4\pi}{n_\theta}, \ldots, 2\pi$

TABLE 1. Table of possible Gabor functions used to tile the image. Each level, *i*, contains $2_i^2 n_\theta$ possible Gabor functions. The values of d_i are obtained by dividing the image as shown in Figure 2; $d_{i+1} = d_i/2$.

After extensive experimentation, it was found that a reasonably good approximation to the image could be obtained by using only level 2 Gabor functions. Reasonable directional selectivity was obtained with four fold symmetry, $n_\theta = 4$. When the even and odd frequency components are included, this results in a Gabor function set with 32 functions. In this study, only the even frequency components generating 16 reconstruction coefficients are used as features in training the neural network.

The results of the experiments are easily explained. For i = 1, the Gabor functions form a directional image reconstruction and provide only field-centered spatial location. As i increases the resolution of the image reconstruction increases. At level 3 the spatial and frequency resolution exceed the stroke size (line width) of the character and provide limited improvement in resolution. All experiments also suggest that, given the complex structure of equations (1)-(5), sampling an image containing less than 16 pixels for each d_i interval is not an efficient use of Gabor functions.

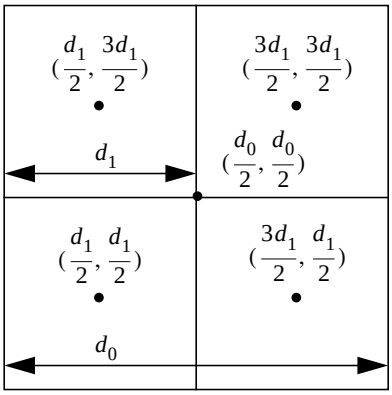

FIGURE 2. Location of the first two levels of tiling points for the Gabor functions. The full set of locations is given in Table 1; in general $d_{i+1} = d_i/2$.

2.4 Feature Extraction by Image Reconstruction

The features needed for neural network input are derived from image reconstruction. Once the Gabor functions are selected, the image reconstruction operation starts by converting the binary image to an 8-bit image, **q**, with a step height between levels of ± 127 and with the sum of the pixels equal to zero (zero mean image). Since the set of Gabor functions is non-orthogonal, the image reconstruction must be performed by least squares optimization. On the small images discussed here, direct methods are far more efficient for this operation than the neural net method proposed for data compression [6]. Given n different $\mathbf{G}_j$'s the image reconstruction operation is based on obtaining a least squares fit to the image **q** by forming the matrix **A**, each component of which is the inner product of the form:

$$a_{ij} = \mathbf{G}_i \cdot \mathbf{G}_j \tag{7}$$

and the vector:

$$b_i = \mathbf{q} \cdot \mathbf{G}_i \tag{8}$$

and solving

$$\mathbf{b} = \mathbf{A}\mathbf{c} \tag{9}$$

for the image reconstruction coefficients, **c**. Since the matrix **A** is the same for any given set of n Gabor functions, the matrix is factored once, and only generation of **b** and back substitution of the factored **A** matrix is required to obtain each **c**. The image is converted to its reconstructed form:

$$\mathbf{q}' = \sum_{j=1}^{n} \mathbf{c}_j \mathbf{G}_j \tag{10}$$

and then thresholded at zero making the image binary again.

FIGURE 3. Gabor Reconstruction of a character image.

The effect of the image reconstruction can be seen in Figure 3. The input image is converted to the gray level image shown in the upper left quadrant of the figure. This input image contains 1024 8-bit elements. Using equations (7)-(10), the reconstructed image **q**', shown in the upper right

quadrant is produced. This image is constructed using 32 8-bit values of c_j. The image is then thresholded at zero to yield the reconstructed image shown in the lower right quadrant of Figure 3. The residual error in the image fit, **q-q'**, is shown in the lower left quadrant of Figure 3. It is interesting to note that the residual output from the Gabor image reconstruction is similar in form to the output of an edge detector. This suggests that the Gabor image reconstruction is a body-detecting image reconstruction.

An additional benefit of the feature enhancement properties of the Gabor image reconstruction can be seen in Figure 4. The input to the image reconstruction is the same as that shown in Figure 3 but with large quantities of random noise added, as shown in the upper left quadrant of Figure 4. The reconstructed image is shown in the upper right quadrant of the figure. The same thresholding procedure is used to produce the output shown in the lower right quadrant of Figure 4. The image in the lower left quadrant contains the residual and is a good image of the random noise in the image.

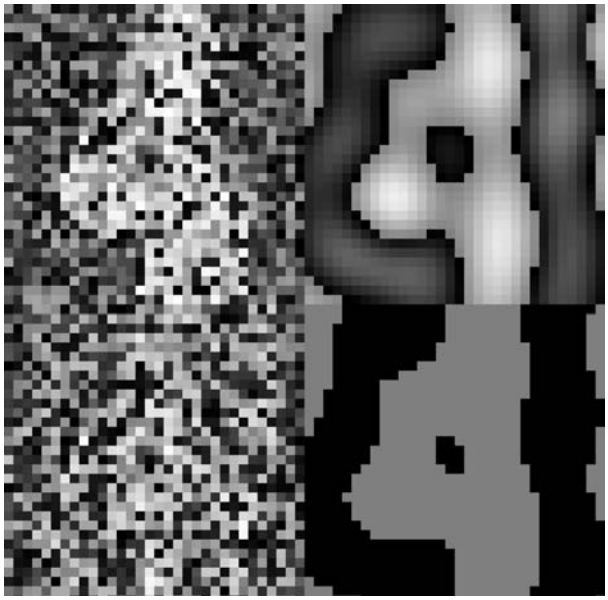

FIGURE 4. Gabor Reconstruction of a character image with severe noise.

The image reconstructions shown in these Figures 3 and 4 use 32 8-bit adaptive coefficients. Note that for the purpose of training the back-propagation network in this paper, only the even frequency components were used, i.e. 16 8-bit adaptive coefficients. Over a set of 1000 character images, 16 Gabor coefficients can be generated on average in 6.8ms per image.

3.0 Back-Propagation Network

Back-propagation networks, with a multilayer network architecture, evolved from the perceptron and utilize supervised learning [7]. A back-propagation network has three or more layers: an input layer, one or more hidden layers, and an output layer. Digit recognition using Gabor features with back-propagation classification is accomplished by presenting the network with a 16-element input vector derived from Gabor image reconstruction. These network inputs are distributed to a fully connected hidden layer and combined into an internal representation. Signals from the hidden layer are transferred using a sigmoid function to the output layer and network activations are produced. The output neurode with the greatest activation is deemed the winner and the character image from which the input pattern was derived is identified as the digit to which the winning output neurode represents.

The training of the network is done through error back-propagation. With each training pattern presented to the network a reference target value is provided with which an error measure can be made between the network's response and the correct target value. The error is then propagated back through the network by updating interconnection weights in order to minimize the error of the network the next time the same or similar input pattern is presented. The implementation used in the following work was taken from the software implementation, "Batchnet" [8]. The following describes the fundamental components of a back-propagation model.

3.1 Three Layer Architecture

The network used in this study has three layers with n_i neurodes in the input layer, n_j neurodes in the hidden layer, and n_l neurodes in the output layer. The subscript variables *i, j,* and *l* reference the input, hidden, and output layers respectively. For this paper $n_i = 16$ and $n_l = 10$ while n_j will vary. Each neurode in a lower layer is fully connected to each neurode in the layer above. The input and hidden layers have a bias neurode, a neurode which is always on. Each connection has a weight associated with it. The weights from the input to the hidden layer are signified by $\mathbf{W}_{ji}$ and from hidden to output layer are $\mathbf{W}_{lj}$ and represent two independent matrices identified by their subscript variable pairs. The inputs to the network are the sixteen Gabor coefficients derived from equations (1)-(5). The network has ten outputs, one for each possible digit class. Figure 5 illustrates this network topology and notation used throughout this paper.

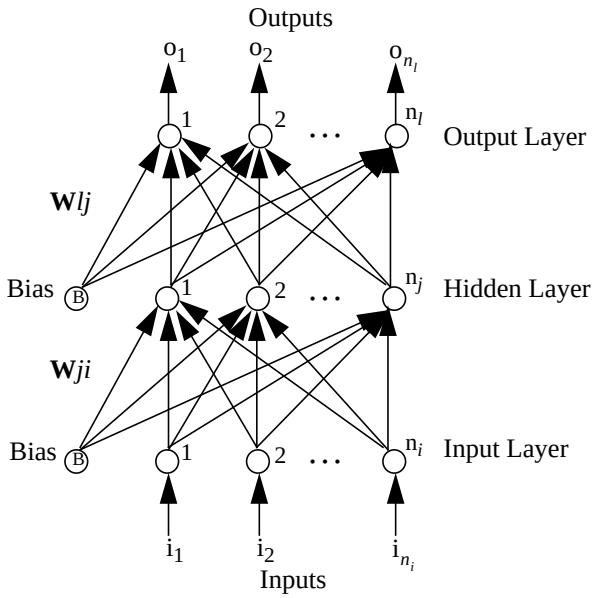

FIGURE 5. Network topology and notation for a back-propagation network.

3.2 Network Training

A back-propagation network requires a set of referenced input prototypes for training. As each input prototype is presented to the network, an error measure is calculated between the network's response and the input's correct target value. The network in this study uses the sum of the squared error which is accumulated over each complete set of prototype presentations or epoch, and is used to modify the connection weights of the network. The training set is repeatedly presented to the network until either the error from the most recent training epoch drops below a threshold or a maximum allowable number of epochs has been exceeded. If the overall error of the network becomes acceptably low, then network learning has converged. This training process can therefore be broken into two stages, feedforward calculations and error back-propagation.

3.2.1 Feedforward Calculations

Given inputs to the network, i_i, the output from the input layer, o_i, is simply o_i=i_i. The input to the hidden layer, i_j, is defined as:

$$i_j = \sum_{i=0}^{n_i} w_{ji} o_i \qquad 1 \le j \le n_j \qquad (11)$$

Note that the bias term from the input layer and hidden layer, o_0, always has a value of 1 and must be included in the calculation of equations (11) and (13). Bias neurodes are used as a threshold mechanism in the layers to which they are connected.

Before any inputs can be presented to the network and outputs calculated, the weights $\mathbf{W}_{ji}$ and $\mathbf{W}_{lj}$ are randomly initialized. Values between ± 0.3 are used. The inputs to the hidden layer are then transferred as output using the sigmoid function which limits values between 0 and 1.

$$o_j = \frac{1}{1 + \exp(-i_j)} \qquad 1 \le j \le n_j \qquad (12)$$

The input to the output layer, i_l, is defined as:

$$i_l = \sum_{j=0}^{n_j} w_{lj} o_j \qquad 1 \le l \le n_l \qquad (13)$$

The output of the output layer, o_l, is defined as:

$$o_l = \frac{1}{1 + \exp(-i_l)} \qquad 1 \le l \le n_l \qquad (14)$$

3.2.2 Error Back-Propagation

The error measure used to monitor the network's performance is the average summed-squared error per output neurode. This error is calculated as the square of the difference between each input's network activation and its target value, accumulated for one complete epoch, and then divided by the number of patterns in the epoch and by the number of output neurodes. This measure determines network convergence during training.

$$E = \left(\sum_{l=1}^{n_l} \sum_{p=1}^{n_p} (t_{pl} - o_{pl})^2 \right) / (n_p n_l) \qquad (15)$$

The error is accumulated for all n_p inputs in the training set, where t_{pl} is input pattern p's target value. A second set of error calculations computes the error signal at the output layer and then at the hidden layer. The error signal is used to update the connection weights in the network in order to minimize the overall error of the network. This error minimization is based on the non-linear, continuously valued, nature of the sigmoid transfer function used in the output equations above, (12) and (14). These characteristics of the sigmoid function are important because the derivation of the error signal involves the first derivative of

the sigmoid function as a component.[7] The error signal at the output layer is defined as:

$$\delta_l = o_l(1 - o_l)\ (t_l - o_l) \qquad 0 \le l \le n_l \qquad (16)$$

where $o_l(1 - o_l)$ is the first derivative of the sigmoid function. The model used in this study updates connection weights upon the completion of each training epoch. In equation (17), δ'_l is the sum of all δ_l for all patterns used in the epoch. Given the accumulated error signals at the output layer, updates to the hidden-to-output weights, $\mathbf{W}_{lj}$, are defined as:

$$w_{lj}(\text{new}) = w_{lj}(\text{old}) + \eta\delta'_l o_j + \alpha\,[\Delta w_{lj}(\text{old})] \qquad 0 \le j \le n_j \qquad (17)$$

where δ'_l is the error signal of an output neurode for the epoch, η is the learning rate, α is the momentum factor, $\Delta w_{lj}(\text{old})$ is the previous epoch's weight change, $\alpha\,[\Delta w_{lj}(\text{old})]$ is the momentum term, and o_0 is the bias term = 1. This network utilizes a momentum term which is a fraction of the previous epoch's weight change. This term is added to try to keep the back-propagation network from falling into local energy minima by maintaining a portion of the networks previous dynamic change.[7]

The error signal at the hidden layer takes into account the error signal from the output layer in the form:

$$\delta_j = o_j(1 - o_j) \sum_{l=0}^{n_l} w_{lj}\delta_l \qquad 0 \le j \le n_j \qquad (18)$$

Note that δ_j in equation (18) is calculated and accumulated on a pattern-by-pattern basis and uses the per pattern calculation of δ_l. Upon completion of an epoch, the accumulated error signal at the hidden layer, δ'_j,is used to update the input-to-hidden weights, $\mathbf{W}_{ji}$.

$$w_{ji}(\text{new}) = w_{ji}(\text{old}) + \eta\delta'_j o_i + \alpha\,[\Delta w_{ji}(\text{old})] \qquad 0 \le i \le n_i \qquad (19)$$

The same values of η and α are used for updating both $\mathbf{W}_{lj}$ and $\mathbf{W}_{ji}$. When a network is trained on inputs normalized to values less than or equal to 1, η is typically assigned a value in the range of 0.5 to 0.001 while α can be value in the range 0 to 1.

3.3 Recall

In order to utilize or test a trained network, the connection weights are frozen and new inputs are presented using just the feedforward calculations (11)-(14). Real application inputs or test patterns are given as input to the network and the output neurode generating the greatest output activation is considered the winner.

4.0 Network Learning

Using back-propagation to solve an application problem often involves defining the following: input parameterization and normalization, output class representation, network parameters and topology, and training prototypes and test sets. The work and results which follow specify each of these categories as they relate to the recognition of machine printed digits using Gabor feature extraction and back-propagation classification.

4.1 Feature Normalization

The features used in this application are the coefficients produced from Gabor reconstruction of segmented character images. There are 16 coefficients produced for each character image, ranging in value from -127 to +127. In order to successfully train the network, four different input representations were analyzed. The first format studied uses no preprocessing so that a network is trained directly on the raw Gabor coefficients. This format will be referred to as the raw bipolar format. To derive the second format, the raw features are translated linearly by adding 127 to each coefficient. This results in feature values ranging from 0 to 254. The second format is named the raw positive format. The remaining formats are normalized versions of the first two. By dividing each feature by the maximum possible feature value, raw bipolar values are normalized to the range ± 1.0 and called normalized bipolar features, while raw positive values are normalized to the range of 0.0 to 1.0 and called normalized positive features.

4.2 Output Representation

There are 10 classes to be recognized by the network. Each class corresponds to a unique digit 0 through 9. Therefore, 10 output neurodes are defined in the output layer, each receiving activation values in the range 0 to 1. The target reference values used with the training prototypes in this study are listed in Table 2.

Class	Target Neurode Values									
0	1	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0
⋮					⋯					
9	0	0	0	0	0	0	0	0	0	1

TABLE 2. List of possible output neurode target values

4.3 Network Parameters

In equations (11)-(19) several network parameters are used which must be specified in order to define a complete back-propagation network. Among these parameters are the learning rate, η, and the momentum factor, α. The values of these two parameters necessary for successful network training will vary according to the input format used. Two different pairs of η and α were tested across the four different input formats defined in Section 4.1. The (η, α) pairs used were (0.01, 0.75) and (0.001, 0.075).

The topology of the networks used in this study are all three-layered architectures with each neurode in one layer fully connected to each neurode in the layer above. The input layer is comprised of 16 neurodes, one for each of 16 Gabor features, and the output layer contains 10 neurodes, one for each digit class 0 through 9. Network topologies with a varied number of hidden neurodes were studied in order to determine the appropriate intermediate structure of a back-propagation model trained with Gabor features. Hidden layers with as few as 3 and as many as 13 neurodes were trained and examined.

A final set of parameters is required to determine convergence of a back-propagation implementation. The first of these termination parameters is a threshold for the average summed-square error of the system defined in equation (15). In addition, a maximum number of epochs is specified to terminate training when the network has not converged, i.e. not reached the desired error threshold. The majority of networks trained in this study used an error threshold of 0.001 and a maximum epoch limit of 1000.

4.4 Training and Test Sets

Training of a back-propagation network requires repeated presentations of a select set of input patterns and their target values. Each output class is represented by multiple prototypes in the training set. By providing a sufficient number of different input examples for each class, the network is able to effectively identify and learn the relational structure between typical inputs and their associated output classes. Network learning can be hindered by too few examples in the training set. Learning can also be inhibited if the set of training prototypes does not accurately reflect the intrinsic structure of the application problem. In light of these considerations, different training set sizes are examined for back-propagation training of Gabor features.

Initially training sets were developed from a single font size and style, 11-point Courier, and consist of 10, 30, and 50 examples of each class. Another training set was developed from multiple font styles of a single font size: 11-point Courier, Helvetica, and Times-Roman. These training sets contain either 10 or 30 examples of each font style per class, which results in 30 or 90 examples per class respectively.

Test sets are used to evaluate how well a trained network learned the input-to-output structure of the application problem. It is possible for a network to converge during training by merely *memorizing* its training prototypes. Even if the training patterns are not explicitly stored within the connection weights, the information embodied by the training set is encoded directly into the network. Any fair test must be conducted using input patterns that are not part of the training set. All results reported in this paper follow this guideline. Test sets were designed to evaluate a trained network's ability to recognize machine printed digits of varying size, style, and random noise. To evaluate size invariance, test sets were created using the Courier font style of three different point sizes; 9, 11, and 14. Test sets of 11-point size were developed from Courier, Helvetica, and Times-Roman font styles to test style invariance. Noise tolerance was also tested using the 11-point Courier test set degraded successively with up to 20% random binary noise.

5.0 Results

In this section, the recognition effectiveness of trained networks is evaluated. These results are documented so as to reveal the method and strategy used to arrive at our network solutions and do not represent an exhaustive investigation. First, the results and knowledge gained from training a back-propagation on Gabor features from a single font style and size are reported. Next, the results from training and testing a network on multiple font styles of the same size are presented. Within these two different training strategies, generalization tests are conducted including style invariance, size invariance, and noise tolerance. Results from a study using activation distributions as a potential classifier are discussed. Classification timings of the network are then reported.

5.1 Training on Single Font Style and Size

In order to demonstrate the feasibility of successfully training a back-propagation network with Gabor features as input, initial studies were conducted on a single font style and size, 11-point Courier. Desired information to be gained from this phase of training and testing included which feature representations and normalizations provide successful network training and testing responses, the learning rate and momentum factors used for each of these feature representations, the number of examples per output class and the number of hidden neurodes needed to adequately train the network, and how well the trained networks generalize when given other inputs.

5.1.1 Raw Feature Training

A survey to find at least one acceptable combination of network parameters for this application was conducted involving the testing and comparing of multiple networks trained with various parameter combinations. Table 3 displays the network parameter and input representation combinations used to train 48 different back-propagation networks on Gabor features from a single font style and size. This table from top to bottom includes the maximum epochs the network was permitted to train, the error threshold used to preempt training if learning converged before the allotted number of epochs were completed, the learning rate and momentum factors, the number of examples per output class included in each training set, the feature representations and normalizations used, and the number of hidden nodes used to train each network. The abbreviations *rbp*, *rps*, *nbp*, and *nps* correspond to the feature normalization formats raw bipolar, raw positive, normalized bipolar, and normalized positive defined in Section 4.1.

Max Epoch	1000
Err Thresh	0.02
η	0.001
α	0.075
Ex/Class	10, 30, 50
Format	rbp,rps,nbp,nps
# Hidden	7, 9, 11, 13

TABLE 3. Training Table: Network parameter combinations used for initial 11-point Courier training studies.

After training, the networks defined in Table 3 were tested with features from 200 digits, 20 input examples per output class, from characters images not included in any of the training sets. Table 4 shows the results from three different tests each conducted on a different network. Table 4 from top to bottom includes the feature format used in training the network and in presenting the test set, the number of hidden nodes and the number of examples per output class used to train the network, the percentage of correct network responses given the test set, the mean activation of the correct responses, the percentage of incorrect network responses given the test set, the mean activation of the incorrect responses, the number of epochs used to the train the network, and the error remaining upon termination of network training. From Table 4, the first network was trained with a training set comprised of 10 examples per output class, the second 30 examples, and the third 50 examples. All three tests exhibit similar recognition percentages, but upon comparison of the magnitude of the mean activations and the error remaining in the network upon termination of training, training sets containing 30 and 50 examples per output class appear to train better than those with only 10 examples per class.

Format	rbp	rbp	rbp
# Hidden	9	9	9
Ex/Class	10	30	50
% Corr	89	90	90
$\bar{\mu}$ Pos Act	.45	.65	.74
% Wrong	11	10	10
$\bar{\mu}$ Neg Act	.14	.31	.39
# Epoch	1000	1000	957
Err Level	0.052187	0.028534	0.020000

TABLE 4. Testing Table: Comparison of networks trained with training sets comprised of 10, 30, and 50 examples per output class. The test set used in each test contained features from 200 digits. $\bar{\mu}$ in this table stands for “mean”.

Table 5 shows results from training and testing on the four different feature representations: raw bipolar, raw positive, normalized bipolar, and normalize positive as defined in Section 4.1. Comparison of the recognition percentages and the error remaining upon training termination shows that raw bipolar is the only input representation which provides useful results using the training parameters listed in Table 3. The networks tested in Tables 4 and 5 were loosely trained for a maximum of 1000 training epochs and an error threshold of 0.02. By changing the number of training epochs, error threshold, learning rate, etc., other successful parameter combinations and results are possible. Therefore, these results serve only as guidelines for further studies.

Format	rbp	rps	nbp	nps
# Hidden	9	9	9	9
Ex/Class	30	30	30	30
% Corr	90	10	21.5	20
$\bar{\mu}$ Pos Act	.65	.10	.113	.102
% Wrong	10	90	78.5	80
$\bar{\mu}$ Neg Act	.31	.10	.107	.102
# Epoch	1000	1000	1000	1000
Err Level	0.028534	0.089998	0.089450	0.089917

TABLE 5. Testing Table: Comparison of networks trained with raw bipolar, raw positive, normalized bipolar, normalized positive feature representations. The test set used in each test contained features from 200 digits.

The raw bipolar representation achieved 90% correct recognition when trained for 1000 epochs. To demonstrate the feasibility of achieving higher recognition rates using Gabor features as input to a back-propagation network, 8 networks were trained for a maximum of 5000 epochs using just the raw bipolar representation. The parameters used for training are displayed in Table 6.

Max Epoch	5000
Err Thresh	0.001
η	0.001
α	0.075
Ex/Class	30, 50
Format	rbp
# Hidden	7, 8, 9, 10

TABLE 6. Training Table: Network parameter combinations used to determine the feasibility of achieving high recognition rates using the raw bipolar Gabor feature representation of 11-point Courier digits.

Table 7 displays the results from one of the tests conducted on the networks trained from Table 6. Upon 5000 training epochs, the error of the network decreased from 0.028534 to 0.004644 and the network recognition increased from 90% to 100% correct.

Format	rbp
# Hidden	9
Ex/Class	30
% Corr	100
$\bar{\mu}$ Pos Act	.86
% Wrong	0
$\bar{\mu}$ Neg Act	None
# Epoch	5000
Err Level	0.004644

TABLE 7. Testing Table: 100% correct recognition achieved on a network trained for 5000 epochs using Gabor features as input. The test set used in this test contained features from 200 digits.

5.1.2 Normalized Feature Training

After demonstrating the feasibility of successfully training and achieving high recognition using raw bipolar Gabor features in a back-propagation network, learning parameters were changed and networks retrained in an attempt to effectively learn normalized input patterns. Very small values of η and α were successful for training the network with raw input features whose values ranged from -127 to +127. In an attempt to train the network on inputs normalized to 1, η and α were increased from 0.001 and 0.075 to 0.01 and 0.75. Based on our experience, as the magnitude of the input values decreases, the learning parameters η and α, necessary to compute a similar change for updating the network's weights, increase. Table 8 lists the parameters and combinations used to train the network on normalized features.

Max Epoch	1000
Err Thresh	0.001
η	0.01
α	0.75
Ex/Class	30, 50
Format	rbp,rps,nbp,nps
# Hidden	3 thru 7, 9, 11, 13

TABLE 8. Training Table: Network parameter combinations used in an attempt to successfully train on normalized Gabor features from 11-point Courier digits.

Listed in Table 9 are results from training networks from Table 8 and testing each of the four feature representations defined in Section 4.1. Normalized features, when used to trained networks with the parameters listed in Table 8, are learned faster than raw input features. Due to increasing the learning rate and momentum factor, the normalized representations achieve a 100% recognition rate. These results are obtained within 1000 training epochs whereas the 100% recognition achieved with raw bipolar features required 5000 training epochs. This can be partially attributed to the range of the randomly initialized weights being in the range -0.3 to +0.3, for all networks used in this study, regardless of the feature representation being used to train. Learning with normalized bipolar features will converge after 735 training epochs and has a higher positive activation mean than normalized positive features which use all 1000 training epochs. Therefore, the feature representation used for the remaining focus of work will be normalized bipolar due to its ability to effectively and efficiently train under the training configuration shown in Table 8.

Format	rbp	rps	nbp	nps
# Hidden	9	9	9	9
Ex/Class	30	30	30	30
% Corr	30	10	100	100
$\bar{\mu}$ Pos Act	.68	.10	.935	.89
% Wrong	70	90	0	0
$\bar{\mu}$ Neg Act	.12	.10	None	None
# Epoch	1000	1000	735	1000
Err Level	0.070216	0.089998	0.001000	0.003475

TABLE 9. Testing Table: Normalized features achieved 100% correct recognition when the learning rate and momentum factor were increased. The test sets used in this test contained features from 200 digits.

5.1.3 Size Invariance Testing

The trained networks described above were trained with Gabor features produced from segmented character images of 11-point Courier digits. One of the networks from Table 8, trained with normalized bipolar features, when presented 4000 feature patterns not used in the training set, achieved 100% correct recognition. Table 10 reports results from two differently trained networks, one trained with 30 examples of each output class, the other with 50 examples of each output class. These two networks were presented input features derived from characters of two different font sizes not used in training, 9-point Courier and 14-point Courier. As can be seen from the recognition results, the networks are very tolerant to font size variations with very little difference in performance between the two networks. This robust characteristic is attributed more to preprocessing than to the network. Each segmented character image is spatially normalized to a 32 by 32 pixel grid before Gabor features are extracted. Gabor reconstruction then determines the minimum combination of uncertainty in position and spatial frequency resolution[6]. This process significantly reduces variations in image space across characters of similar font style and different font size. However, performance was very poor when the networks tested in Table 10 were presented with feature patterns derived from characters of font styles other than Courier.

Style	Courier	Courier	Courier	Courier
Size	9	14	9	14
Format	nbp	nbp	nbp	nbp
# Hidden	9	9	9	9
Ex/Class	30	30	50	50
% Corr	99.90	99.89	99.98	99.81
$\bar{\mu}$ Pos Act	.952	.837	.951	.837
% Wrong	0.10	0.11	0.02	0.19
$\bar{\mu}$ Neg Act	.755	.481	.781	.518

TABLE 10. Testing Table: Normalized bipolar networks trained with 11-point Courier achieved high recognition when tested on 9 and 14-point Courier feature patterns. The 9-point Courier test set contained features from 5200 digits and 14-point Courier test set contained features from 2640 digits.

5.2 Training on Multiple Font Styles of Same Size

A training set strategy was developed in an attempt to train a back-propagation network to accurately recognize Gabor features derived from character images of multiple font styles and sizes. Table 11 lists the network parameters combined to train 8 different networks on 3 combined font styles, Courier, Helvetica, and Times-Roman. Four of these networks were trained with 10 examples from each font style for each output class, totalling 300 example patterns. The other four networks were trained with 30 examples from each font style for each output class, totalling 900 example patterns. The networks were trained on one constant font size, 11-point, due to the size invariant capabilities discovered during the tests described in Table 10.

Style	Courier, Helvetica, Times
Size	11
Ex/Style	10, 30
Max Epoch	1000
Err Thresh	0.001
η	0.01
α	0.75
Format	nbp
# Hidden	5, 7, 9, 11

TABLE 11. Training Table: Network parameter combinations used in an attempt to successfully train on multiple font styles: Courier, Helvetica, and Times-Roman.

5.2.1 Style Invariance Testing

Table 12 displays results of testing a network trained with 10 examples of three font styles per output class. The first three tests in the table correspond directly to the font styles used in training. Unlike the networks trained on a single font style, the network reported in Table 12 demonstrates a network's ability to achieve greater than 99.9% correct accuracy when presented normalized bipolar Gabor features derived from any of the three, Courier, Helvetica, or Times-Roman, font styles. It is interesting to report that the studies conducted on the networks trained from Table 11 showed that networks trained with 10 examples per font style per output class performed as well as networks trained with 30 examples per font style per output class.

Style	Courier	Helvetica	Times	Dot Matrix
Size	11	11	11	10
Format	nbp	nbp	nbp	nbp
# Hidden	9	9	9	9
Ex/Class	10	10	10	10
% Corr	99.93	100	100	99.56
$\bar{\mu}$ Pos Act	.938	.946	.950	.857
% Wrong	0.07	0	0	0.44
$\bar{\mu}$ Neg Act	.406	None	None	.530

TABLE 12. Testing Table: Results from a network trained with multiple font styles. The Courier, Helvetica, and Times-Roman test sets each contained features from 4000 digits. The Dot Matrix test set contained 4800 digits.

The fourth test reported in Table 12 is the most interesting. The test examples used for this test were comprised of character images originally produced by a typical dot matrix printer on moderate-quality traction-feed computer paper. All other test and training sets used in this study were segmented from images originally produced on a laser printer. The font style selected was 10 pitch Courier and was printed using a double density, 9 by 11, matrix. This test set represents characters from a font style *similar* to those trained, but produced by a different printing technology, on a different quality of paper, and of an untrained size. With all these added variations not included in the training set, the network tested in Table 12 performs with a greater than 99.5% accuracy.

5.2.2 Size Invariance Testing

Tests for recognizing feature patterns derived from images of different font sizes were conducted in order to determine if the tolerance to font size variations, which is strongly exhibited by networks trained on a single font, has been compromised when a network is trained with multiple font styles. Table 13 shows the results of testing one of the networks trained from Table 11 with 9, 11, and 14-point font sizes. With correct recognition greater then 99.8%, the network's tolerance to font size variations has not been impaired in any significant way.

Style	Courier	Courier	Courier
Size	9	11	14
Format	nbp	nbp	nbp
# Hidden	9	9	9
Ex/Class	10	10	10
% Corr	99.85	99.93	100
$\bar{\mu}$ Pos Act	.960	.938	.832
% Wrong	0.15	0.07	0
$\bar{\mu}$ Neg Act	.698	.406	None

TABLE 13. Testing Table: A network trained with multiple font styles can recognize test examples of varying font size with high accuracy. The 9-point Courier test set contained features from 5200 digits. The 11-point Courier test set contained features from 4000 digits. The 14-point Courier test set contained features from 2640 digits.

5.2.3 Noise Tolerance Testing

A test to determine how tolerant back-propagation networks trained with Gabor features are to random noise was conducted. The network trained on multiple font styles

and reported in Tables 12 and 13 is further tested here. The results of degrading character images with increasing percentage levels of random noise, deriving Gabor features through image reconstruction, and then testing the trained network with these resulting feature inputs is reported in Table 14.

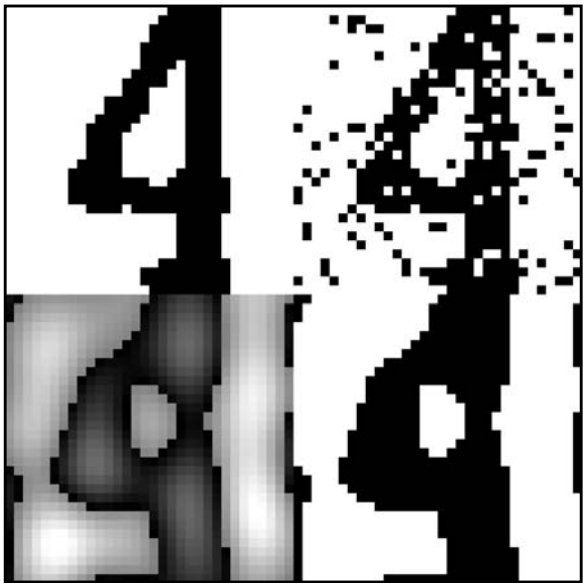

FIGURE 6. Image degraded with 10% random noise.

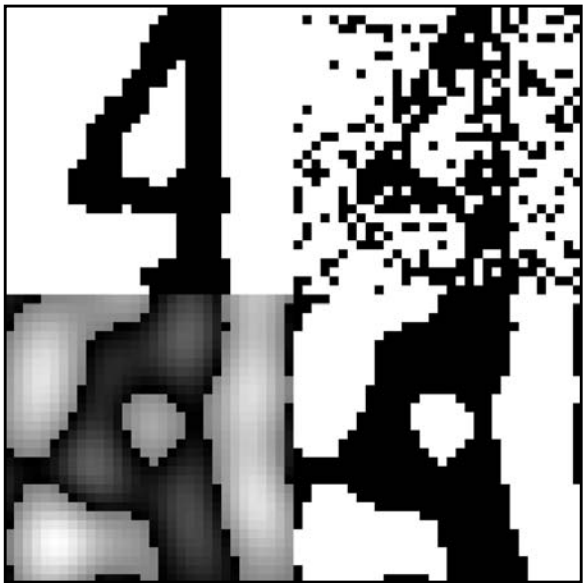

FIGURE 7. Image degraded with 20% random noise.

Figures 6 and 7 illustrate a segmented image of an 11-point Courier digit “4” degraded with 10% random noise in Figure 6 and the same image degraded with 20% random noise in Figure 7. This noise was imposed by flipping the color of a percentage number of pixels chosen at random. Each of the two figures is a composite of 4 quadrant images. The upper-left quadrant is the original segmented and scale-normalized image, the upper-right quadrant illustrates the resulting image after the random noise has been imposed, and the bottom two quadrants show the resulting Gabor image reconstruction. The bottom-left is the resulting gray scale image, and the bottom-right is the resulting binary image after thresholding. Even with the original character image drastically degraded by two different percentage levels of random noise, both image reconstructions produce very similar results. This reinforces the fact that Gabor image representations serve as spatially localized low-pass filters.

The recognition results in Table 14 show that the network, when tested with Gabor features derived from 11-point Courier character images, is very tolerant to random noise degradations in the range 0% to 10%. Beyond this point the network performance decreases substantially. At 15% random noise, which is between the two noise conditions illustrated in Figures 6 and 7, the network maintains a correct recognition rate over 88.9%.

% Noise	0	5	10	15	20
Format	nbp	nbp	nbp	nbp	nbp
# Hidden	9	9	9	9	9
Ex/Class	10	10	10	10	10
% Corr	99.93	99.78	98.50	88.98	61.63
$\bar{\mu}$ Pos Act	.938	.881	.773	.659	.649
% Wrong	0.07	0.22	1.50	11.03	33.38
$\bar{\mu}$ Neg Act	.406	.266	.238	.271	.334

TABLE 14. Testing Table: A network trained with multiple font styles can tolerate over 10% random noise with high accuracy. The 11-point Courier test set used in this test contained features from 4000 digits.

5.3 Activation Distribution Analysis

This section shows initial results on using output activation statistics from the network as a scoring analysis technique. The results reported in the previous tables all include a mean positive activation value and a mean negative activation value. These statistics are derived by first scoring all winning output neurode activations against their target values and then separating them into two distinct categories or distributions; those network responses which are correct and those which are incorrect. Means are simply calculated by summing the activations in each distribution and dividing by the corresponding number of activations in each distribution.

Two general observations can be made from studying the mean activations reported above. As the mean positive activation increases, the network performance increases. Also, as the distance between the mean positive and mean negative activations increases, the network performance increases. If the positive and negative activation distributions were Gaussian in shape, then Figure 8 could be used to visualize these generalizations. In Figure 8, the negative activation distribution is shown on the left with a mean value of $\bar{\mu}_n$, and the positive activation distribution is to the right with a mean value of $\bar{\mu}_p$. In an ideal system, these two distributions would not overlap. However, in our example system, the two distributions intersect each other with the tail of one distribution overlapping with the tail of the other. A threshold, T, is included in Figure 8 which optimally separates one distribution from the other and can be approximated by calculating the midpoint between $\bar{\mu}_n$ and $\bar{\mu}_p$. Replacing the *a priori* knowledge used to generate the two distributions by this threshold results in determining all activations to the right of the threshold as correct classifications and all activations to the left as rejected classifications. The system responses when evaluated fall into one of four categories, true positive, false negative, true negative, and false positive. The term *false* used to define two of these categories implies system classification errors and the significance of these two types of errors depends on the application. These four categories are illustrated in Figure 8.

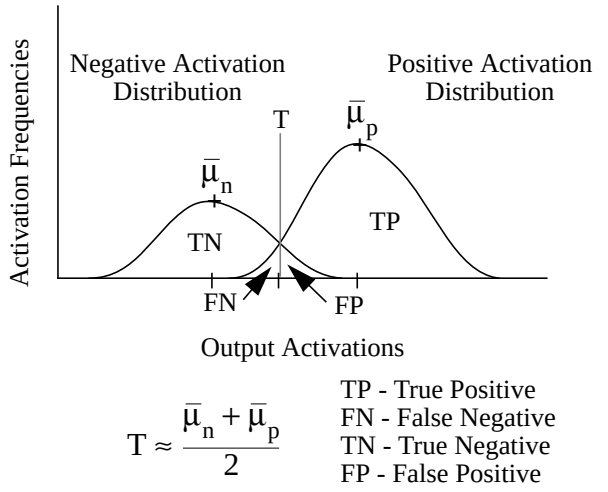

FIGURE 8. Threshold Classifier based on two Gaussian distributions.

Figures 9 and 10 display real positive and negative activation distributions from a network trained on multiple font styles with 7 hidden neurodes. Figure 9 plots activation distributions generated from character images degraded with 15% random noise and Figure 10 plots activation distributions generated from character images degraded with 20% random noise. These distributions are not Gaussian in shape and yet they can be used for scoring analysis. The positive activation distribution in both figures has maximum activation values on the right with activations decreasing from right to left. Comparing the negative activation distributions between the two figures demonstrates that as the uncertainty of the network increases (random noise increases), the negative distribution activations increase with a greater rate on the left and decrease from left to right. This causes the interior tails of the positive and negative distributions to overlap similarly to those of the Gaussian model illustrated in Figure 8.

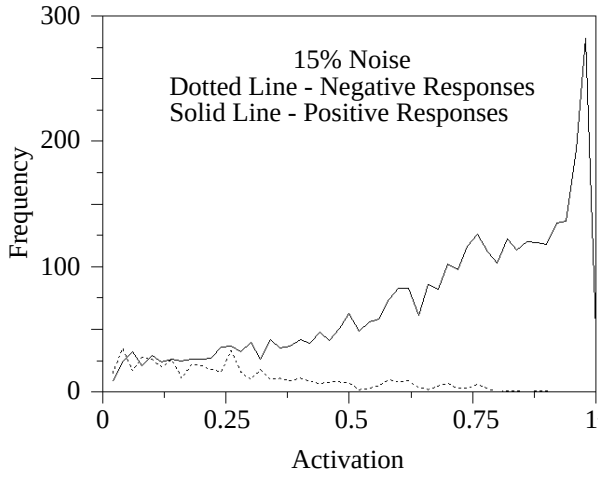

FIGURE 9. Positive and Negative Activation Distributions from a network presented features derived from character images degraded with 15% random noise.

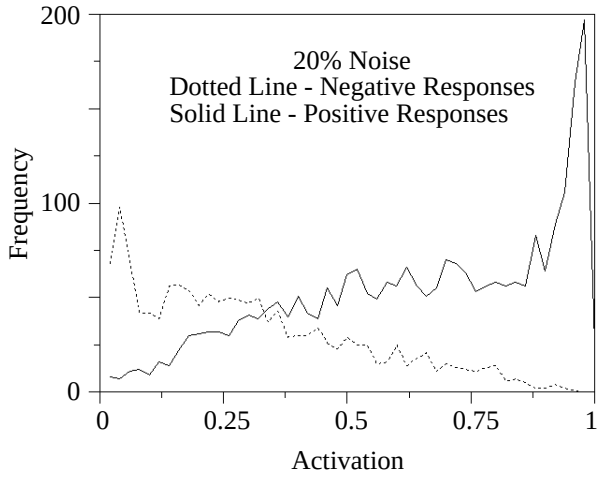

FIGURE 10. Positive and Negative Activation Distributions from a network presented features derived from character images degraded with 20% random noise.

Results from applying the activation classification technique on a network are reported in Table 15. The table includes the random noise imposed for each test, the number of hidden neurodes in the network, the mean positive acti-

vation value and the mean negative activation value, the threshold used, and the four categories of evaluated responses: true positive, true negative, false positive, and false negative. The test using 10% random noise achieves a substitution error rate of 0.25% corresponding to the reported false positive value. The substitutional error rate obtained without using the threshold technique equals 2.08% and can be calculated by adding TN and FP entries from Table 15. Therefore, through using this classification technique, substitutional errors by the system have effectively been reduced from 2.08% to 0.25%.

% Noise	10	15	20
# Hid	7	7	7
$\bar{\mu}_p$	.781	.673	.639
$\bar{\mu}_n$	.250	.267	.297
T	.516	.470	.468
% TP	86.80	69.23	46.58
% TN	1.83	10.00	27.65
% FP	0.25	2.15	8.00
% FN	11.13	18.63	17.78

TABLE 15. Testing Table: Results of reclassification. The test set used in these tests contained features from 4000 11-point Courier digit images degraded with 3 different percentage levels of random noise.

5.4 Classification Timings

The time to classify an input pattern ranged on average from 1.5 milliseconds to 2.5 milliseconds using a serial computer, Sun 3/470. This range represents timings taken from networks with different numbers of hidden neurodes. A network with 7 hidden neurodes takes an average of 1.8 ms to classify while a network with 9 hidden neurodes takes 2.0 ms.

6.0 Conclusions

This study documents the strategies, methods, and results used and gained from training a back-propagation network with Gabor features for character recognition. These networks effectively combine the biological properties of Gabor image reconstruction with the generalization power of back-propagation learning. Tests reported in this paper utilize the parallel system illustrated in Figure 1 for recognition of machine printed digits. This recognition is shown to be invariant to changes in font style and font size while being tolerant to random noise degradations. A network trained with multiple font styles achieved greater than 99.8% accuracy when presented large test sets varying in font style and font size. A test set representing characters from a font style similar to those trained, but produced by a different printing technology, on a different quality of paper, and of an untrained size performed with greater than 99.5% accuracy. This network also maintained greater than 88.9% accuracy when presented features derived from character images degraded with up to 15% random noise and classified inputs with an average time of 2 ms. Through examining the network's winning activations, a simple analytic scoring method was proposed. This technique is based on the relationship between positive and negative activation distributions illustrated in Figure 8 and plotted in Figures 9 and 10. Applying this method to networks exhibiting poor overall results, for example a network presented with very noisy inputs, minimizes substitutional errors. Using this technique, a network presented with features derived from images degraded with 10% random noise had substitutional errors reduced from 2.08% to 0.25%.

References

[1] L. D. Jackel, H. P. Graf, W. Hubbard, J. S. Denker, D. Henderson, and Isabelle Guyon, "An Application of Neural Net Chips: Handwritten Digit Recognition", IEEE International Conference on Neural Networks, San Diego, Vol. II, pp. 107-115, 1988.

[2] A. Rajavelu, M. T. Musavi, and M. V. Shirvaikar, "A Neural Network Approach to Character Recognition", *Neural Networks*, **2**, pp. 387-393, 1989.

[3] C. L. Wilson, "A New Self-Organizing Neural Network Architecture for Parallel Multi-Map Pattern Recognition - FAUST", in preparation.

[4] I. Ohzawa, G. C. DeAngelis, and R. D. Freeman, "Stereoscopic Depth Discrimination in the Visual Cortex: Neurons Ideally Suited as Disparity Detectors", *Science*, **249**, pp. 1037-1041, 1990.

[5] K. G. Beauchamp, *Walsh Functions and Their Applications*, Academic Press, London, pp. 41-65, 1975.

[6] J. G. Daugman, "Complete Discrete 2-D Gabor Transform by Neural Networks for Image Analysis and Compression", *IEEE Trans. on Acoustics, Speech, and Signal Processing*, **36**, pp. 1169-1179, 1988.

[7] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Parallel Distributed Processing, Volume 1: Foundations*, edited by D. E. Rumelhart, J. L. McClelland, et al, MIT Press, Cambridge, pp. 318-362, 1986.

[8] R. C. Eberhart and R. W. Dobbins, *Neural Network PC Tools, A Practical Guide*, Academic Press, San Diego, pp. 36-49, 1990.