Reject Mechanisms for Massively Parallel Neural Network Character Recognition Systems

Michael D. Garris and Charles L. Wilson

National Institute of Standards and Technology
Gaithersburg, Maryland 20899

Presented at SPIE's Conference on
Neural and Stochastic Methods in Image and Signal Processing
San Diego, 1992

ABSTRACT

Two reject mechanisms are compared using a massively parallel character recognition system implemented at NIST. The recognition system was designed to study the feasibility of automatically recognizing hand-printed text in a loosely constrained environment. The first method is a simple scalar threshold on the output activation of the winning neurode from the character classifier network. The second method uses an additional neural network trained on all outputs from the character classifier network to accept or reject assigned classifications. The neural network rejection method was expected to perform with greater accuracy than the scalar threshold method, but this was not supported by the test results. The scalar threshold method, even though arbitrary, is shown to be a viable reject mechanism for use with neural network character classifiers. Upon studying the performance of the neural network rejection method, analyses show that the two neural networks, the character classifier network and the rejection network, perform very similarly. This can be explained by the strong non-linear function of the character classifier network which effectively removes most of the correlation between character accuracy and all activations other than the winning activation. This suggests that any effective rejection network must receive information from the system which has not been filtered through the non-linear classifier.

1. INTRODUCTION

The use of neural networks to recognize well-formed isolated characters has been extensively studied, resulting in recognition rates nearly equal to human performance.[1] This is especially true for digit recognition. A massively parallel character recognition system has been implemented at NIST in which neural network character classification is only one of many functional components required to automatically convert hand-printed data on paper into ASCII computer text.[2] Through the use of this system, it has been shown that well-formed characters can not be guaranteed and other functional components such as character segmentation are error prone. These dynamics introduce ambiguities during classification, degrading the accuracy of the system. Reject mechanisms have been designed to identify ambiguous classifications so that they may be resolved by a human data transcriber rather than have the classifications directly entered into the computer incorrectly. By rejecting ambiguous classifications, the accuracy of the recognized information automatically entered into the computer is increased by minimizing the risk of recognition errors. However, this increase in accuracy is achieved at the expense of reduced system throughput. The more data is rejected, the more data must be hand-keyed, and the demand for hand-keying is exactly what automated recognition technology is designed to reduce.

In order to evaluate recognition system performance, researchers at NIST have developed an automated scoring package which generates statistics on accuracy which are then used in numerous analyses. Using the scoring package in conjunction with the massively parallel character recognition system, results from implementing and testing two different reject mechanisms are presented. The first reject mechanism studied is a scalar threshold method while the second reject mechanism is a neural network method. Section 2 describes the massively parallel character recognition system, Section 3 introduces the automated scoring package, Section 4 discusses various sources of system errors, Section 5 presents the concept of rejection rate versus accuracy, Section 6 defines the two different reject mechanisms and compares their performance, and Section 7 draws conclusions based on the results.

2. MASSIVELY PARALLEL CHARACTER RECOGNITION SYSTEM

The massively parallel character recognition system is comprised of eight separate functional components. These components are shown in Figure 1 and can be divided into three categories: system loading and storing, image processing, and automated recognition. The figure charts the transformation of data from initial image input through ASCII text output. An image of a document is loaded into the system, the hand-printed data is located and preprocessed, a recognition algorithm classifies the isolated hand-printed characters, and the recognition results are retrieved from the system and stored as text. The work presented in this paper focuses on the interactions between the system's recognition module and the reject module.

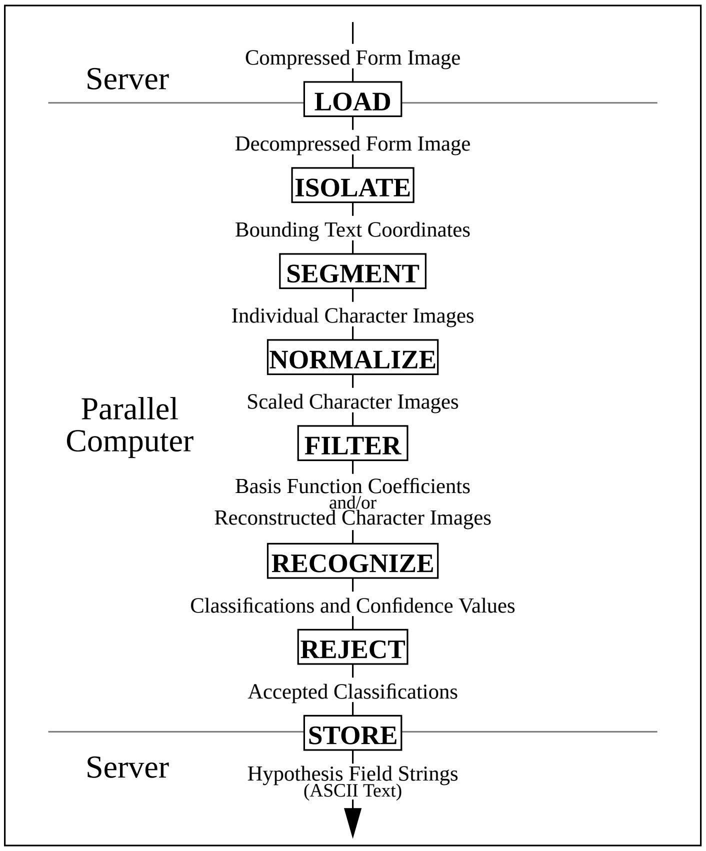

Server
Compressed Form Image
LOAD
Decompressed Form Image
ISOLATE
Bounding Text Coordinates
SEGMENT
Individual Character Images
NORMALIZE
Parallel Computer
Scaled Character Images
FILTER
Basis Function Coefficients and/or Reconstructed Character Images
RECOGNIZE
Classifications and Confidence Values
REJECT
Accepted Classifications
STORE
Server
Hypothesis Field Strings (ASCII Text)

Figure 1. The functional components of the recognition system distributed between the serial server and the massively parallel computer.

2.1 Recognition module

The functionality of the recognition module has been the focus of most published character recognition research.[3-5] When most people speak of character recognition, they are referring to tasks performed by this single module. As can be seen in Figure 1, this highly studied component is just one of many modules necessary to implement a successful automated document processing and data capture system. This module takes input from the filter module which conducts image enhancements and/or feature extraction on segmented character images. The recognition system designed at NIST classifies filtered data via one of several neural networks.

One of the neural networks which has been used in the recognition system is a self-organizing algorithm developed at NIST called Feedforward Association Using Symmetrical Triggering (FAUST).[6] FAUST provides a parallel, multi-map, self-organizing, pattern classification procedure similar to those known to exist in the mid-level visual cortex.[7] This neural network uses a feed-forward architecture which allows multi-map features stored in weights acting as associative memories to be accessed in parallel and to trigger a symmetrically controlled parallel learning process. This method allows features of different data type, such as binary image patterns and multi-bit statistical correlations, to be updated in parallel. The three essential features of FAUST are: 1) Different feature classes use individual association rules for pattern comparison. 2) Different feature classes use individual learning rules for pattern modification. 3) All feature classes contribute symmetrically to learning.

The system configuration studied in this paper uses a multi-layered perceptron (MLP)[8] for the character classifier in the recognition module. The MLP is a more traditional neural network architecture, and it classifies by generating feedforward activations across a network containing an input layer, one hidden layer, and an output layer. The character classifier network used in this study was trained using Scaled Conjugate Gradient (SCG)[9] with Karhunen Loeve (KL)[10] feature vectors extracted from segmented character images by the filter module. Using this network, handprinted digit recognition accuracy of 96% and recognition speeds of 10,100 characters/second can be realized with the massively parallel recognition system[2].

2.2 Reject module

The massively parallel character recognition system has a component responsible for pruning low-confidence classifications from the system. The reject module tags ambiguous classifications as unknown recognitions so that they may be transcribed by a human and not directly entered into the computer. Two different reject mechanisms are presented in this paper. The first reject mechanism is simply implemented as a static threshold applied to the winning activation from the character classifier network in the recognition module. The second reject mechanism is implemented as a neural network which takes the entire output vector from the character classifier network and determines whether the assigned classification should be rejected or accepted using a single output or rejection neurode. By tagging confusable recognitions, postprocessing such as manual keying and context corrections can be applied to the system's hypothesized text. These two reject mechanisms are described in detail in Section 5.

3. SCORING PACKAGE

An automated character recognition system testing methodology has been developed at NIST and is illustrated in Figure 2. The testing of a recognition system requires the input of images from referenced databases for which the textual information contained in each image requiring recognition has been recorded as reference strings in the database. A referenced database can be used as a test for evaluating the performance of a recognition system. The images in the database represent the test material, while the reference strings represent the test's answers. In this way, the database images are presented to the recognition system and hypothesized strings are generated as system output. The scoring package, a third testing component, then receives the system's hypothesized strings and reconciles them to the database's reference strings.

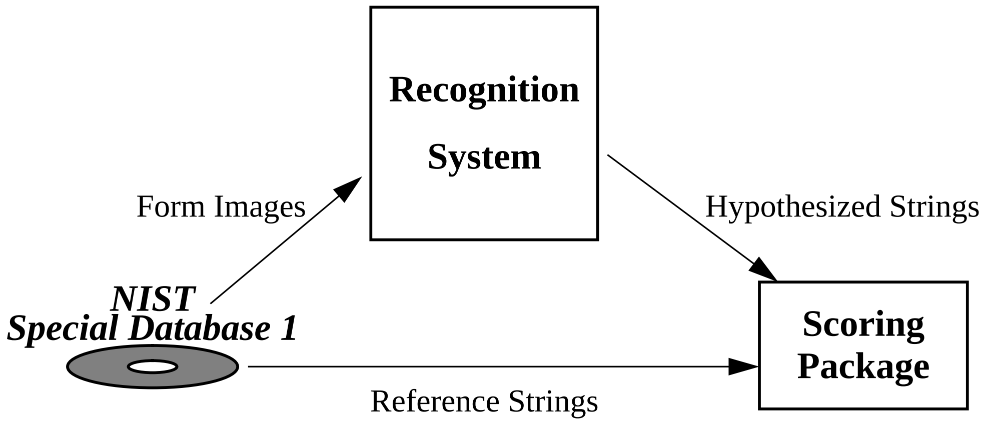


Figure 2. Testing methodology for character recognition systems.

This testing methodology has a two-fold benefit. First, system developers can use this testing scheme as a tool for evaluating the impact different algorithms have on the overall performance of their system. One documented configuration of a system may be used to process the reference database, and the system results then scored. These scores represent a base-line of performance for comparing future system configurations providing a vehicle for testing new algorithms and ideas. A second benefit derived from this testing scheme aids system acquisitions. A potential customer of character recognition technology can specify his application requirements in the form of a referenced database. Then he may request the database be run through a vendor's system, the output from which is automatically scored, allowing comparisons of performance among various vendor products. The power of this testing methodology for the purpose of system comparison is that the buyer of this technology needs no detailed knowledge of the internal workings of the product, which is often proprietary. Yet, the buyer can ensure that the product conforms to a specified level of performance for his specific application.

4. SYSTEM ERROR SOURCES

The scoring package uses dynamic string alignment to reconcile system hypothesis strings to database reference strings. Using string alignment, various types of statistics can be tabulated including system accuracies and the identification of different system error sources. Sources of system errors can be divided into two general categories, ambiguities arising from malformed characters and system component failures.

4.1 Malformed character ambiguities

A familiar system error is a substitution error in which the recognition module assigns an incorrect classification to a segmented character image. For example, a 3 may be incorrectly classified as an 8. If the system only processed correctly isolated characters then the substitution error could be attributed to a malformed ambiguous character. The 3 really does look like an 8 when read by a human. Figure 3 displays some examples of ambiguous character images resulting in substitution errors. The reader is challenged to determine the proper classification of these segmented character examples.

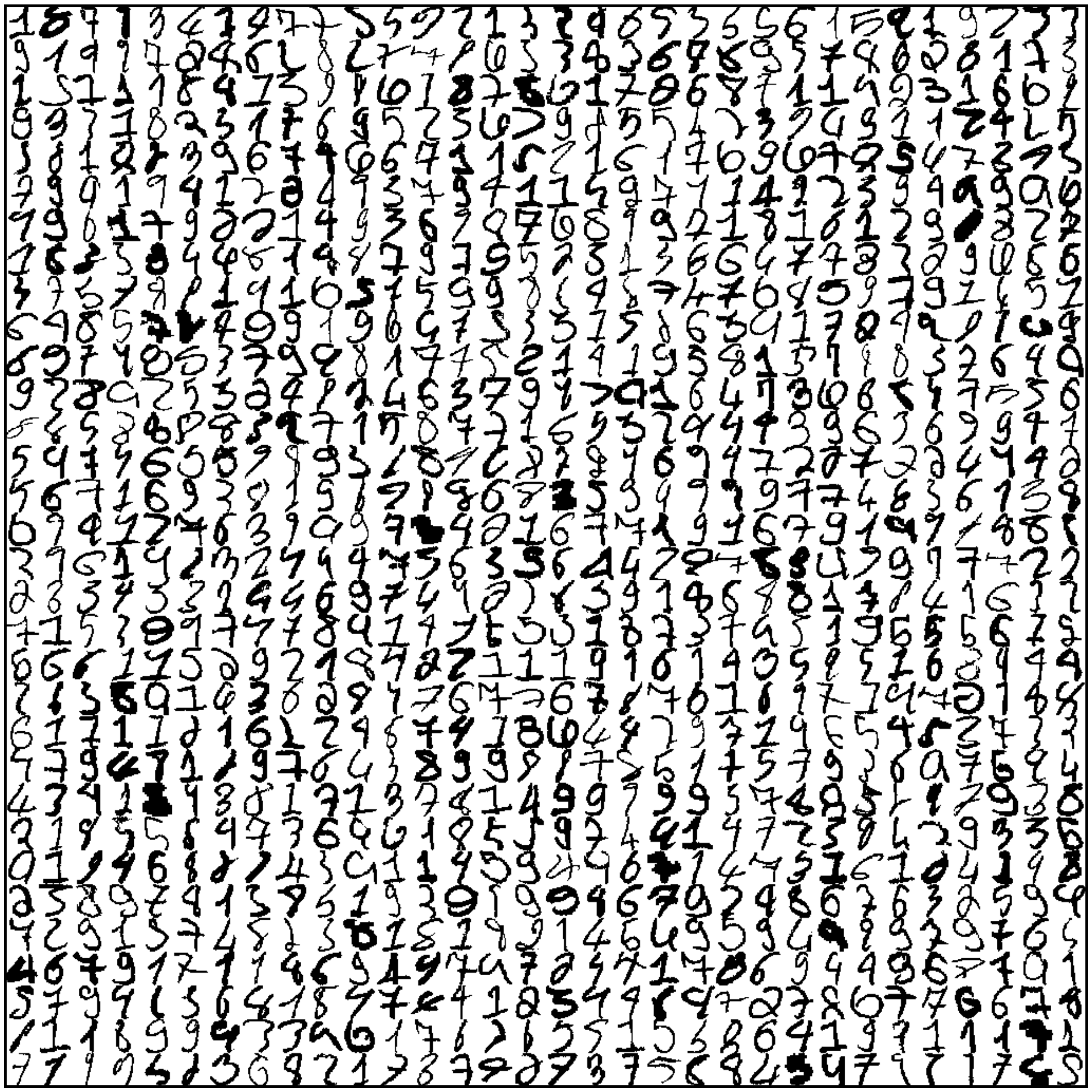


Figure 3. Examples of malformed, ambiguous, character images.

Figure 4 shows an example alignment produced by the scoring package of a substitution error caused by an ambiguous character. The top image is an isolated field containing the five digits 0, 1, 2, 3, and 4. In this example, the hand-printed 3 is unreadable. The second line of images are the result of segmenting the isolated field into separate images, one character per image. The third line in the figure lists the reference string of what truly was printed in the field. The fourth line lists the hypothesis string corresponding to the assigned classifications generated by the recognition system. The last line in the figure marks the substitution errors identified by the scoring package with a 1 representing a substitution error made by the recognition system. As shown in the figure, the segmented character image containing the malformed 3 is classified by the recognition system as an 8 and is identified by the scoring package by reconciling the hypothesis string with the reference string.

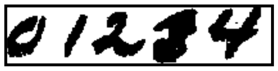
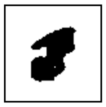
Isolated Field Image					
Segmented Character Images					
Reference String	0	1	2	3	4
Hypothesis String	0	1	2	8	4
Alignment of Substitutions (1= Sub)	0	0	0	1	0

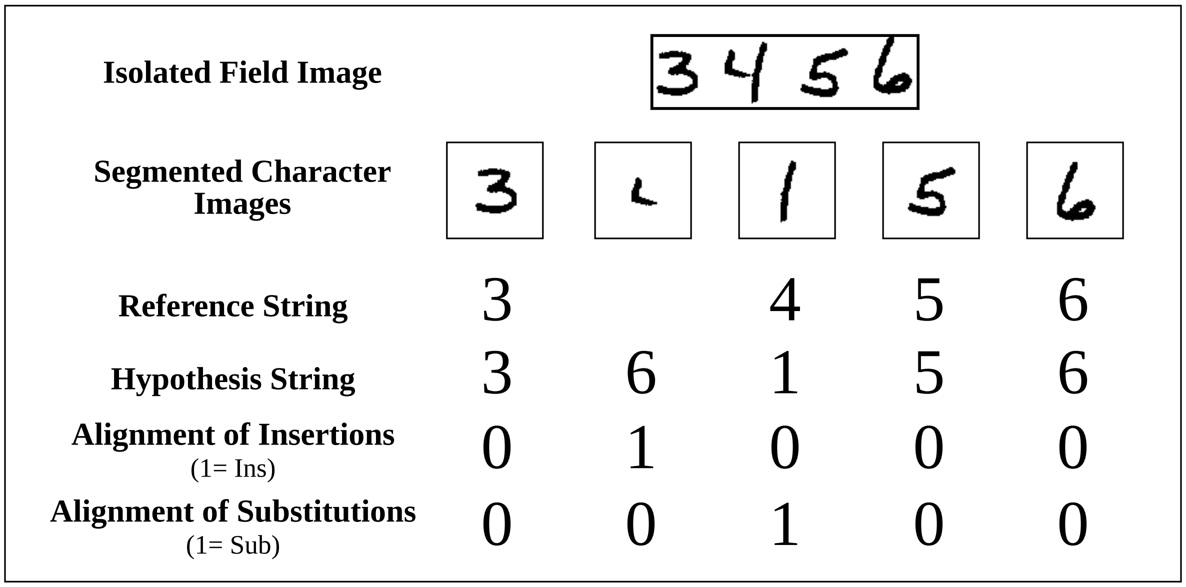
Figure 4. Scoring package alignment of a substitution error caused by a malformed character.

4.2 System component failures

Two system components listed in Figure 1 are largely responsible for introducing errors into the recognition system; the segmentation module and recognition module. Character segmentation, the separation of a multiple character text image into isolate characters, one character per image is currently the topic of much research.[11-14] Most character classifiers are designed to recognize characters one character image at a time. However, in unconstrained hand-print, characters frequently touch and/or overlap making the clean separation of characters most difficult. Unfortunately, isolated characters are not always segmented correctly. This results in isolated images containing partial characters, multiple characters, and/or noise. These segmented images are in turn passed to the recognition module's character classifier. If one system module fails, it introduces errors into the system flow so that every successive module in the system will be potentially influenced by that error. Failure to cleanly segment characters is a source of many errors in a character-based recognition system.

Typical segmentation failures result in the insertion of character-like images into, and the deletion of legitimate character images from, the recognition system. This is demonstrated in the examples shown in Figures 5 and 6. Figure 5 shows an example alignment produced by the scoring package of an insertion error caused by a segmentation failure. The top image is an isolated field containing the four digits 3, 4, 5, and 6. The second line of images is the result of segmenting the isolated field into separate images, which are assumed to be one character per image. Notice the 4 has been incorrectly separated into two pieces resulting in two isolated images with two strokes forming a right angle in the left image and a vertical stroke in the right image. This is an example of a segmentation failure, the splitting of characters. The third line in the figure lists the reference string of what truly was printed in the field. The fourth line lists the hypothesis string corresponding to the assigned classifications generated by the recognition system.

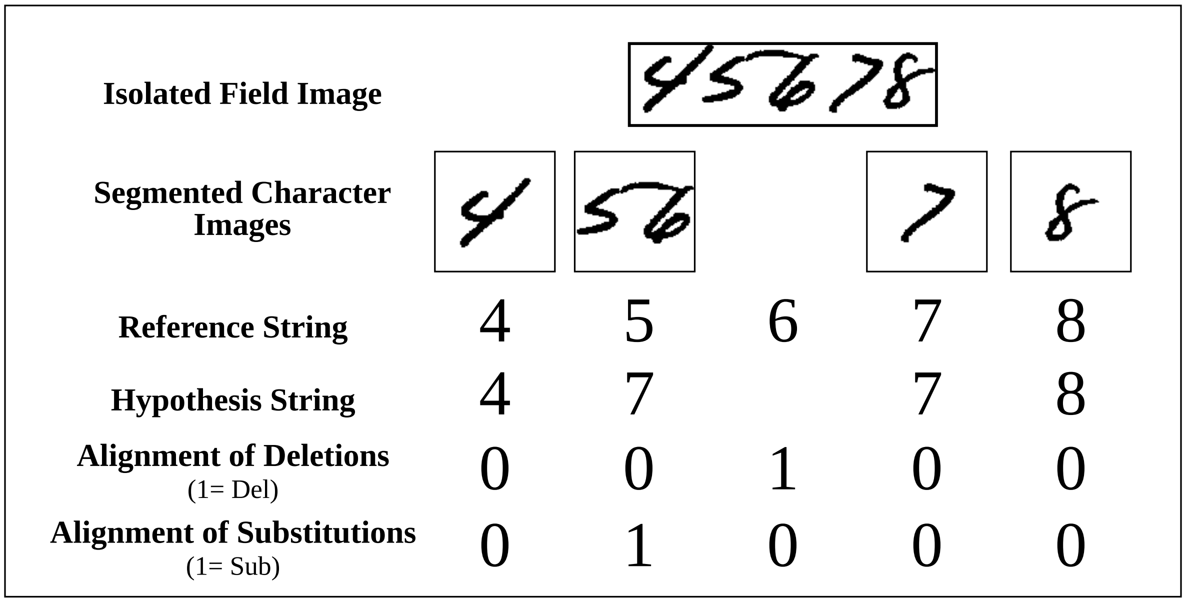
Due to the segmentation failure, the character classifier in the recognition system is presented the two pieces of the 4 rather than one complete character. The result can be seen in the hypothesis string where the first piece of the 4 is classified as a 6 and the second piece of the 4 is classified as a 1. The fifth line in the figure marks the insertion error identified by the scoring package with a 1 representing the inserted classification of a 6. Often, a single segmentation failure introduces multiple errors into the system. This can be seen by the last line in the figure which marks a substitution error at the position of the second piece of the 4, the vertical stroke. If segmentation failures go undetected, then the character classifier assumes the resulting isolated images are correct and the classifier will assign a classification to each isolated image it is permitted to see. By reconciling the reference string to the hypothesis string, the scoring package labeled the second piece of the 4 as a substitution error, knowing that a 4 was truly printed in the field.



Isolated Field Image					
Segmented Character Images					
Reference String	3		4	5	6
Hypothesis String	3	6	1	5	6
Alignment of Insertions (1= Ins)	0	1	0	0	0
Alignment of Substitutions (1= Sub)	0	0	1	0	0

Figure 5. Scoring package alignment of an insertion error caused by a segmentation failure and resulting in a substitution error.

Figure 6 shows an example alignment produced by the scoring package of a deletion error caused by a segmentation failure. The top image is an isolated field containing the five digits 4, 5, 6, 7, and 8. The second line of images is the result of segmenting the isolated field into separate images, which are assumed to be one character per image. Notice the 5 and 6 have been merged into a single isolated image. This is another example of a segmentation failure, the merging of characters. The third line in the figure lists the reference string of what truly was printed in the field. The fourth line lists the hypothesis string corresponding to the assigned classifications generated by the recognition system.



Isolated Field Image					
Segmented Character Images					
Reference String	4	5	6	7	8
Hypothesis String	4	7		7	8
Alignment of Deletions (1= Del)	0	0	1	0	0
Alignment of Substitutions (1= Sub)	0	1	0	0	0

Figure 6. Scoring package alignment of a deletion error caused by a segmentation failure and resulting in a substitution error.

Due to the segmentation failure, the character classifier in the recognition system is presented a single image containing two characters rather than two separate images each containing one character. This is another example of how, if a segmentation failure goes undetected, the character classifier will assign a classification to each isolated image it is permitted to see.The result can be seen in the hypothesis string where the number of assigned classifications is one less than the length of the reference string, and the merged image containing the 5 and 6 is classified as a 7. The fifth line in the figure marks the deletion error identified by the scoring package with a 1 representing the position of the deleted character. The last line in the figure marks the substitution error resulting from the merged character image being incorrectly classified. By reconciling the reference string to the hypothesis string, the scoring package located the position of the deleted character and labeled the classification assigned to the merged character image as a substitution error.

The examples in Figures 5 and 6 demonstrate how undetected component failures introduce errors into the recognition system which are then passed on to other components. These failures have a cascading effect, resulting in multiple errors being introduced into the system at different stages in the system's flow. Given these additional error sources, it becomes increasingly more difficult for the scoring package to attribute substitution errors of the system to any single source.

5. REJECTION RATES VERSUS ACCURACY

Through the use of the recognition system presented in Section 2 and the scoring package presented in Section 3, system dynamics between functional components can be studied. A critical point to be made from Section 4 is that complex recognition systems are prone to many different sources of errors. These contribute to ambiguous classifications resulting in substitution errors that in turn degrade the performance of the system. Errors will occur, so they must be identified and dealt with effectively. This is the motivation for reject mechanisms.

The role of the rejection module is to identify incorrect classifications made by the recognition module, regardless of the error source. If incorrect classifications are not intercepted, the incorrect classification go unnoticed and corrupt the integrity of the data being processed. These incorrect classifications are caused by ambiguities in the system, and being ambiguous, they can not be identified with perfect accuracy. Sometimes ambiguous classifications are actually assigned correctly. Therefore, reject mechanisms must be designed and tuned to deal with trade-offs. The positive effect of reject mechanisms is that ambiguous classifications potentially contributing to undetected system errors are removed from the system, increasing the reliability of the remaining system classifications. Different recognition applications can tolerate different levels of error frequency. A system can be tailored to the specific requirements of a given application by adjusting the amount of data rejected, thereby increasing the accuracy of the system. The rejected data can then be routed to a human who, using provided context, can correctly transcribe the data or choose to reject the data as being illegible. There will never be 100% capture of hand-printed documents because some people's hand-print, or at times samples of their hand-print, is unreadable by humans. Given the current state-of-the-art, the best that machines can hope to do is match the performance of human recognition.

Reject mechanisms come with obvious costs. First, as stated previously, reject mechanisms identify ambiguous classifications that potentially lead to system errors. The ambiguous classifications represent both incorrect and some correct classifications. The challenge facing developers of reject mechanisms is to maximize the detection of incorrect classifications while minimizing the rejection of correct classifications. Second, as the rate of data rejected increases, the rate of effective system throughput decreases. The extreme example is, if all classifications were rejected, there would be no information automatically processed, requiring the hand-keying of all images.

6. REJECT MECHANISMS

This section defines two alternative methods for rejecting ambiguous classifications. Each method was integrated separately into the massively parallel character system for testing. The two resulting system configurations ran, processing all the images in *NIST Special Database 1*.[15] From the system results, performance analyses were generated through the use of the scoring package for comparing the two techniques.

Both rejection techniques were tested in conjunction with exactly the same character classifier. The classifier used was an MLP network trained with KL feature vectors using SCG. This network utilized 48 input features and generated 10 output activations, one for each digit class 0 through 9. Using this classifier in the recognition module, the class of an input feature vector was determined by winner-take-all. In other words, the class associated with the neurode receiving the greatest output activation was chosen to be the assigned classification of the input feature vector.

6.1 Scalar threshold method

The first rejection method studied uses a scalar threshold applied to the output activation generated by the character classifier's winning neurode. If the winning neurode's activation was greater than the threshold, then the classifier was assumed to be sufficiently confident in its choice. Therefore, the assigned classification was accepted and permitted to be directly processed by the computer. If the winning neurode's activation was less than the threshold, then the classification was rejected and flagged for hand-keying by a human. The threshold represents a tunable parameter. The higher the threshold, the more confident a system's classification must be. This minimizes the risk of incorrect classifications going undetected at the expense of reducing effective system throughput.

Figure 7 lists the relationship between the percentage of classifications rejected and the accuracy of the remaining accepted classifications using the scalar threshold method. These values were derived by incrementing the scalar threshold across its dynamic range 0.0 to 1.0. At each level of threshold, the number of classifications rejected along with the accuracy of the accepted classifications were calculated. The scoring package determines whether each character classification made by the system was truly correct or incorrect. This knowledge, assumed to be ground-truth, is required to compute the accuracy of the accepted classifications.

Rejection Rate & Accuracy							
0.0000	0.940678	0.2600	0.988863	0.5200	0.991932	0.7800	0.992744
0.0200	0.952419	0.2800	0.989337	0.5400	0.992071	0.8000	0.992676
0.0400	0.961726	0.3000	0.989751	0.5600	0.992249	0.8200	0.992571
0.0600	0.968880	0.3200	0.990046	0.5800	0.992318	0.8400	0.992910
0.0800	0.974021	0.3400	0.990359	0.6000	0.992469	0.8600	0.993051
0.1000	0.978047	0.3600	0.990621	0.6200	0.992527	0.8800	0.993239
0.1200	0.981019	0.3800	0.990888	0.6400	0.992592	0.9000	0.993615
0.1400	0.983128	0.4000	0.991085	0.6600	0.992631	0.9200	0.993709
0.1600	0.984689	0.4200	0.991192	0.6800	0.992605	0.9400	0.994617
0.1800	0.985928	0.4400	0.991334	0.7000	0.992676	0.9600	0.994836
0.2000	0.986882	0.4600	0.991556	0.7200	0.992676	0.9800	0.995682
0.2200	0.987735	0.4800	0.991715	0.7400	0.992806		
0.2400	0.988331	0.5000	0.991819	0.7600	0.992785		

Figure 7. Percentage of rejected classifications versus the accuracy of accepted classifications using the scalar threshold method.

6.2 Neural network method

A second rejection method was studied in order to determine if a more sophisticated reject mechanism could be obtained through the use of machine learning. A MLP network was trained using SCG to take as input the 10 output activations from the character classifier network and determine through the use of a single output neurode whether the assigned classification was correct or incorrect. The neural network method sought a machine-learned rejection algorithm without imposing any preconceived formulas such as was used in the scalar threshold method. It also had the opportunity to take into account the dynamics among all 10 classifier output activations, whereas the scalar threshold method only analyzed the single winning neurode's activation. It was expected that the reject mechanism based on all 10 activations would yield a more sophisticated solution and better performance.

The rejection network was trained on errors identified by the scoring package from a prior run of the massively parallel character recognition system. The recognition system ran across all 2,100 form images contained in *NIST Special Database 1*, processing the 130 digits distributed across 28 fields on each form for a total potential digit count of 273,000 digits. The scoring package was then invoked on the system output, flagging each segmented image classified by the recognition module as correct or incorrect based on the reference strings included in the database. After scoring, 7,500 output activation vectors produced by the character classifier determined by the scoring package to be incorrect were chosen. Each one of these vectors contained 10 output values, one for each digit class 0 through 9. In addition, 7,500 output activations vectors from the character classifier determined by the scoring package to be correct were chosen. Combining the two sets of activation vectors, a training set of 15,000 prototypes was created. The prototypes were mapped to a single output neurode with vectors representing incorrect character classifications having a target value of 1.0 and vectors representing correct character classifications having a target value of 0.0. The rejection network was then trained using SCG, integrated into the massively parallel recognition system, and tested by running the recognition system across all of *NIST Special Database 1*.

As each character image was segmented from a form, it was classified by the recognition module, the output of which was processed by the rejection network. If the output activation of the single neurode of the rejection network was greater than a scalar threshold then the character image along with its assigned classification was rejected from the system, otherwise the assigned classification was accepted by the system and assumed to be correct.

Figure 8 lists the relationship between the percentage of classifications rejected and the accuracy of the remaining accepted classifications using the neural network method. These values were derived by incrementing the scalar threshold on the output activation uniformly across its dynamic range 0.0 to 1.0. At each level of threshold, the number of classifications rejected along with the accuracy of the accepted classifications were calculated. Once again, the scoring package determines whether each character classification made by the system was truly correct or incorrect. This knowledge, assumed to be ground-truth, is required to compute the accuracy of the accepted classifications.

Rejection Rate & Accuracy							
0.000000	0.940678	0.071972	0.972663	0.108265	0.979853	0.775425	0.990349
0.032566	0.958774	0.074136	0.973252	0.113655	0.980447	0.846312	0.990615
0.040450	0.962158	0.076168	0.973726	0.120311	0.981059	0.918111	0.988028
0.045698	0.964139	0.078294	0.974208	0.131741	0.981869	0.926240	0.989815
0.049905	0.965605	0.080589	0.974728	0.211015	0.983751	0.990797	0.953878
0.053345	0.966904	0.082869	0.975287	0.228106	0.984924	0.996875	0.887019
0.056283	0.967963	0.085510	0.975865	0.237185	0.985745	0.998152	0.857724
0.059040	0.968804	0.088019	0.976396	0.248701	0.986736	0.998764	0.829787
0.061549	0.969549	0.090693	0.976888	0.265183	0.987722	0.999177	0.812785
0.063818	0.970233	0.093623	0.977356	0.296386	0.988789	0.999523	0.787402
0.065985	0.970880	0.096594	0.978005	0.362307	0.989539	0.999823	0.787234
0.068081	0.971528	0.099990	0.978623	0.520958	0.990348		
0.070154	0.972095	0.103934	0.979267	0.661518	0.990623		

Figure 8. Percentage of rejected classifications versus the accuracy of accepted classifications using the neural network method.

6.3 Comparison of results

The results in Figures 7 and 8 were produced by integrating each reject mechanism separately into the massively parallel recognition system and then running each of the two system configurations across all 2,100 form images in *NIST Special Database 1*. These system runs produced character classifications, character classifier output activation vectors, and character rejection decisions to reject or accept each segmented character image. The scoring package was then utilized to determine the accuracy of the two system configurations by reconciling system responses to the reference strings included with the database.

Figure 9 plots the results obtained using the two reject mechanisms. The solid line plots the performance of the massively parallel recognition system using the scalar threshold method for rejecting ambiguous character classifications. The dashed line plots the performance of the recognition system using the neural network method for rejecting ambiguous character classifications. In the figure, the horizontal axis plots the percentage of character classifications rejected, and the vertical axis plots the accuracy of the character classifications accepted by the system. With no rejection and all assigned character classifications accepted by the system, both system configurations achieved 94% accuracy. The two system configurations are identical, using the exact same modules listed in Figure 1 except for the rejection module, and with no rejections resulting at this point, the two system performances are identical. Between 0% and 15% rejection, both systems maintain equivalent performance. At 15% rejection, the systems are 98% accurate, which, compared to the 94% accuracy achieved at 0% rejection, demonstrates the ability to improve the accuracy of the recognition system by rejecting more and more ambiguous character classifications. The improvement in accuracy is negligible from 20% rejection on, with the neural network method consistently performing slightly below the scalar threshold method.

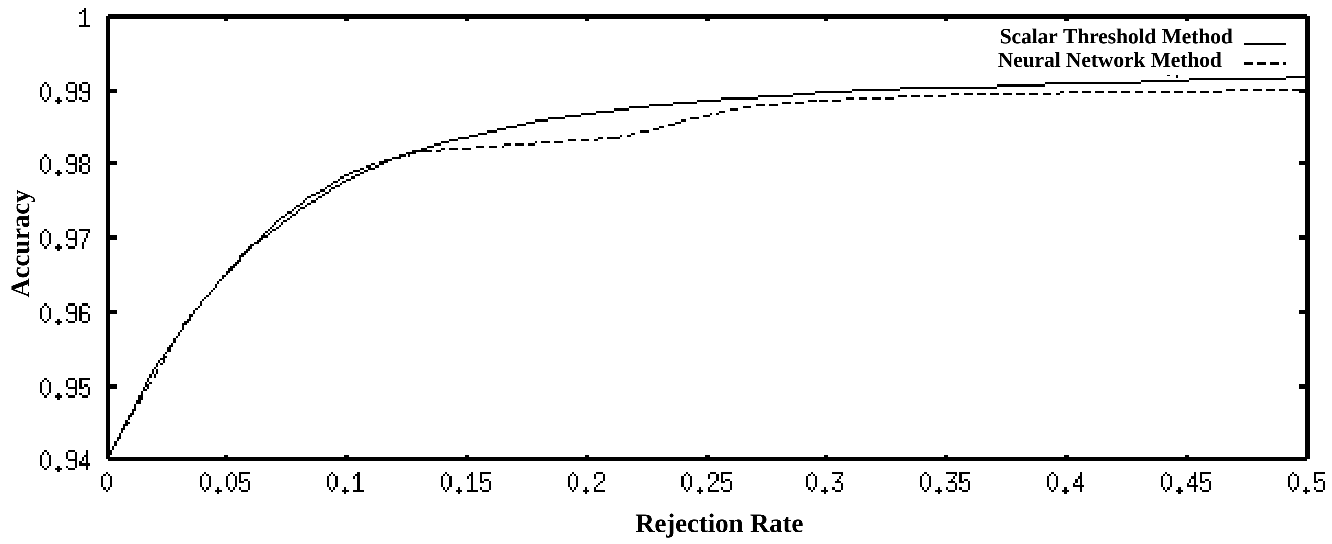


Figure 9. Comparison of recognition system configurations using the scalar threshold method and the neural network method.

6.4 Neural network method analysis

In Section 6.2, it was stated that the neural network method was expected to yield a more sophisticated solution achieving better performance than the scalar threshold method. From the comparison in Figure 9, it can easily be seen that better performance was in fact not achieved. The neural network method based on machine learning matched the performance, or performed slightly worse, than the scalar threshold method based on a preconceived formula. Figures 10 and 11 examine the dynamics between the character classifier network and the rejection network. Both figures contain 1,264 plotted pairs of activations generated by the two neural networks. The data points designated by a plus-sign represent the winning neurode activation from the character classifier network for a given segmented character image. The data points designated by a diamond represent the activation from the single output neurode of the rejection network. The horizontal axis corresponds to segmented character images processed by the system, and the vertical axis corresponds to network activation. For each unit position on the horizontal axis there exist two data points, one plus-sign point and one diamond point. In this way, activation responses between the character classifier network and the rejection network can be compared.

Figure 10 shows activations generated from processing segmented character images plotted in the order of increasing activation from the rejection network, and Figure 11 shows the same activations plotted in the order of increasing maximum neurode activation from the character classifier network. Notice the inverse relationship between the activations of the two neural networks. In general if the rejection network activation is low, the winning neurode activation from the character classifier network is high, and if the rejection network activation is high, the winning neurode activation from the character classifier network is low.

A more interesting pattern is observed in Figure 10. Sporadically along the plotted points of winning neurode activations from the character classifier network (plus-signs), the activity fluctuates. As the fluctuations progress left to right, the magnitude of the distance between the fluctuating points increase. It is observed that at each duration of fluctuation, there is a noticeable step increase in the activations from rejection network (diamonds). This observations shows that the activations generated by the rejection network are strongly correlated to the winning neurode activation from the character classifier network. Remember, there are 10 activations generated by the character classifier network and used as input to the rejection network. However, the activation of the winning neurode from the character classifier network, independent from the other 9 activations, has a large apparent influence on the rejection network. This suggests that the strong non-linear function of the character classifier network effectively removes most of the correlation between character accuracy and the remaining 9 losing activations of the classifier network output. This helps explain why the neural network method of rejection, as defined in this study, in general equalled the performance of the scalar threshold method.

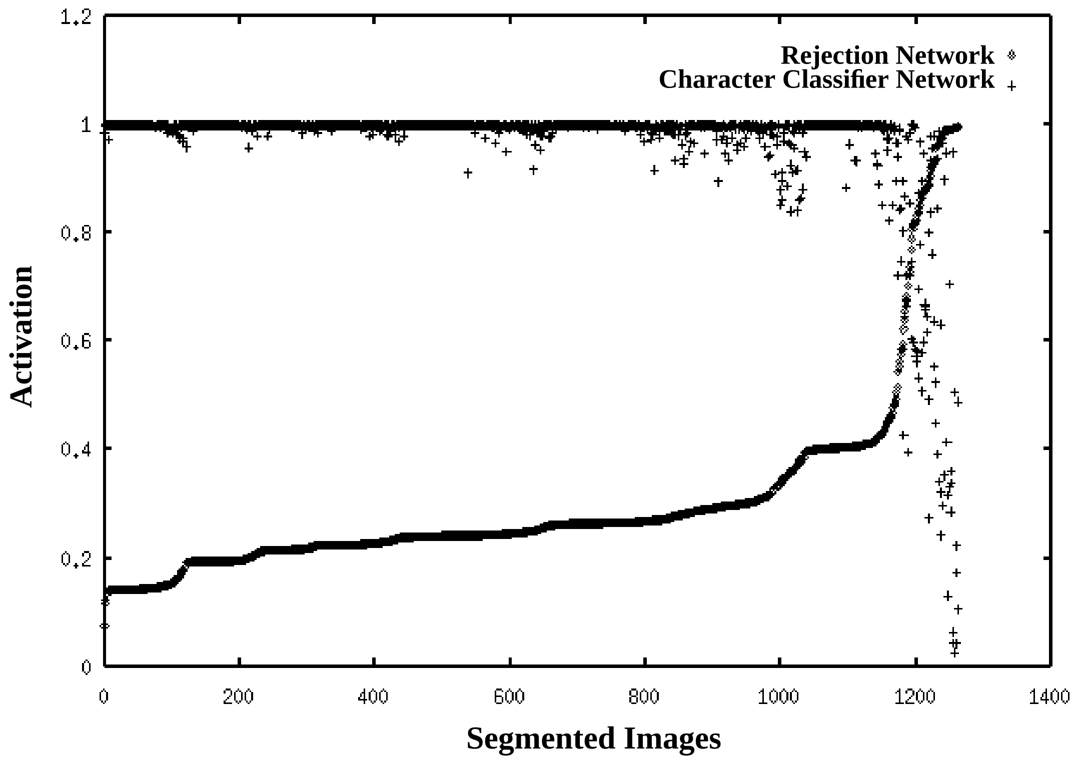


Figure 10. Rejection network activations and corresponding winning neurode activations from the character classifier network plotted in the order of increasing rejection network activation.

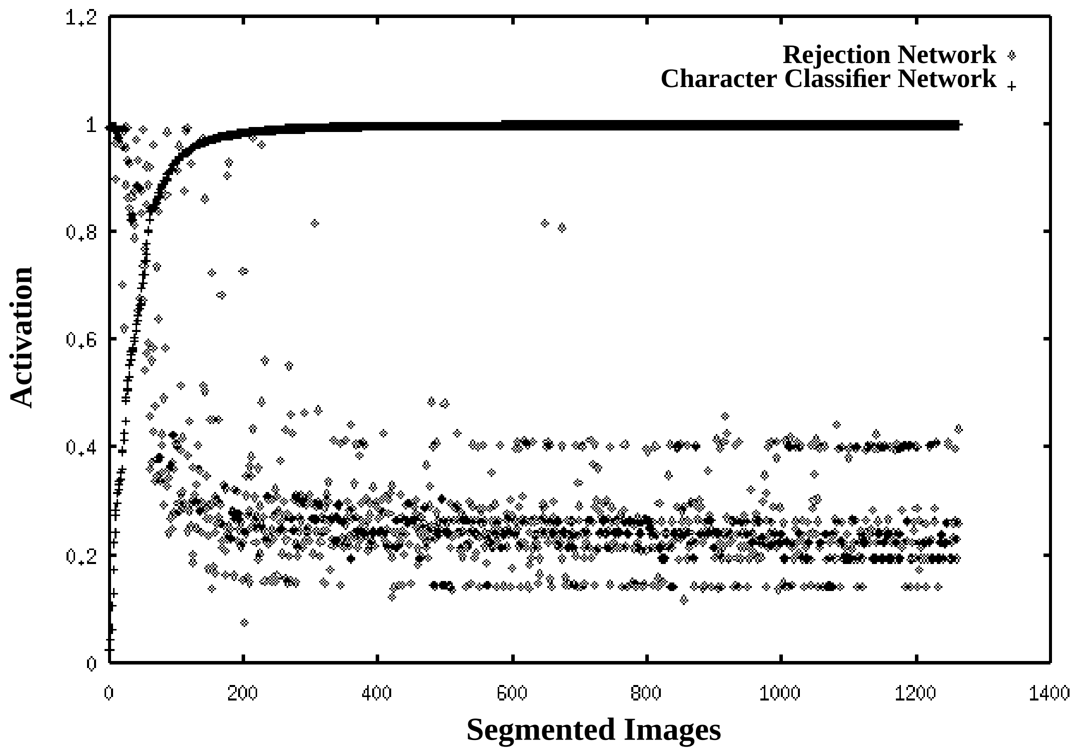


Figure 11. Rejection network activations and corresponding winning neurode activations from the character classifier network plotted in the order of increasing winning neurode activation from the character classifier network.

7. CONCLUSIONS

Half of the prototypes used for training the rejection network contained correct character classification vectors. These vectors contain 10 activations, one for each digit class 0 through 9. When the character classifications are correct, the winning neurode corresponding to the correct digit class has an extremely high activation (close to 1.0); the other 9 activations are extremely small (close to 0.0). This was the motivation for developing the scalar threshold method. Because of this consistent pattern of activations, it is believed that the learning of the rejection network was saturated by these dominant patterns thereby biasing the training. This can be explained by the strong non-linear function of the character classifier network which effectively removes most of the correlation between character accuracy and all activations other than the winning activation. This suggests that, to improve upon the performance of the scalar threshold method, an effective rejection network must receive information from the system which has not been filtered through the non-linear classifier. For example, reject mechanism performance may be improved by training a rejection network on the same inputs used to train the character classifier network. Future work will include experiments in which alternative training strategies for the rejection network are investigated.

This study demonstrates that the neural network rejection method described in this paper performs slightly worse than the scalar threshold method. These two methods may be nearly equal in performance, but in terms of cost they are vastly different. The scalar threshold method is implemented as a single conditional test, whereas generating the activation of the single output neurode using the neural network method requires two matrix multiplications followed by a a conditional test for thresholding.This work has demonstrated that the scalar threshold method, even though arbitrary, is a very viable reject mechanism for use with neural network character classifiers. For reject mechanisms, the scalar threshold method performed slightly better than the neural network method, and it has the added advantage of being computationally less expensive and much easier to implement.

REFERENCES

1. J. Geist, R. A. Wilkinson, C. Burges, R. Creecy, B. Hammond, J. Hull, N. Larsen, T. Vogl, and C. Wilson, “First Census Optical Character Recognition System Conference,” to be published as a NISTIR, 1992.

2. M. D. Garris, C. L. Wilson, J. L. Blue, G. T. Candela, P. Grother, S. Janet, and R. A. Wilkinson, “Massively Parallel Implementation of Character Recognition Systems,” *NISTIR 4750*, January 1992.

3. L. D. Jackel, H. P. Graf, W. Hubbard, J. S. Densker, D. Henderson, and Isabelle Guyon, “An Application of Neural Net Chips: Handwritten Digit Recognition,” *International Joint Conference on Neural Networks*, Vol. II, pp. 107-115, San Diego, 1988.

4. G. L. Martin and J. A. Pittman, “Recognizing Hand-Printed Letters and Digits,” *Advances in Neural Information Processing Systems*, D. S. Touretzky, Vol. 2, 405-414, Morgan Kaufmann, Denver, 1989.

5. C. L. Wilson, R. A. Wilkinson, and M. D. Garris, “Self-Organizing Neural Network Character Recognition on a Massively Parallel Computer,” *International Joint Conference on Neural Networks*, Vol. II, pp. 325-329, San Diego, 1988.

6. C. L. Wilson, “A New Self-Organizing Neural Network Architecture for Parallel Multi-map Pattern Recognition - FAUST,” *Progress in Neural Networks*, Vol. 4, to be published, 1992.

7. A. Rojer and E. Schwatz, “Multi-map Model for Pattern Classification,” *Neural Computation*, 1:104-115, 1989.

8. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning Internal Representations by Error Propagation,” *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D. E. Rumelhart, J. L. McClelland, et al., Volume 1: Foundations, pp. 318-362, MIT Press, Cambridge, 1986.

9. P. J. Grother and J. L. Blue, “Training Feed Forward Neural Networks Using Conjugate Gradients,” *NISTIR 4776*, February 1992.

10. P. J. Grother, “Karhunen Loeve Feature Extraction for Neural Handwritten Character Recognition,” *NISTIR 4824*, April 1992.

11. R. A. Wilkinson, “Segmenting Text Images with Massively Parallel Machines,” SPIE Intelligent Robots and computer vision X, to be published, Boston, 1991.

12. M. D. Garris and C. L. Wilson, “A Neural Approach to Concurrent Character Segmentation and Recognition,” to be published in the proceedings of Southcon, Orlando, 1992.

13. H. P. Graf, C. Nohl, and J. Ben, “Image Segmentation with Networks of Variable Scale,” *Advances in Neural Information Processing Systems*, R. Lippmann, Vol. IV, pp. 480-487, Denver, 1991.

14. G. L. Martin, “Recognizing Overlapping Hand-Printed Characters by Centered-Object Integrated Segmentation and Recognition,” *Advances in Neural Information Processing Systems*, R. Lippmann, Vol. IV, pp. 504-511, Denver, 1991.

15. C. L. Wilson and M. D. Garris, “Handprinted Character Database,” *NIST Special Database 1*, **HWDB**, 1990.