In Proceedings of Neural Networks in Applications, Third International Workshop (NN'98) - Magdeburg, Germany, pp. 39-46, 1998.

A Study on the Combination of Classifiers for Handwritten Digit Recognition

Sven Behnke[1], **Marcus Pfister**[2], **and Raúl Rojas**[1]

[1] Free University of Berlin
Institute of Computer Science
Takustr. 9
14195 Berlin, Germany
{behnke|rojas}@inf.fu-berlin.de

[2] SIEMENS AG
AUT 65
Postfach 4848
90327 Nürnberg, Germany
pfister@scn.de

Abstract

This article presents a case study on the combination of classifiers for the recognition of handwritten digits. Four different classifiers are briefly described and evaluated using the NIST-digits data set. Different parallel and sequential combination schemes are introduced. Furthermore, it is described how to tune the sequential combination using a boosting technique. These combination methods are benchmarked using the NIST-digits. The experimental results indicate that all investigated classifier combinations outperform the best individual classifier. The sequential combination yields slightly better results than the parallel combination and has a much lower computational complexity. In addition, it is possible to improve the performance of the sequential combination by boosting.

1 Introduction

Automatic handwriting recognition has a variety of applications at the interface between man and machine. In this article we will focus on off-line systems that are a key component for some important real world problems, such as mail sorting and check processing.

The pattern recognition task, to tell which digit is represented by a pixel-image of a handwritten numeral, is difficult because of the high variability of the scanned image caused by the peculiar writing style of different persons, the context of the digit, different writing devices and media. The scanned digits are generally of different size and slant, and the strokes that constitute the digits vary in width and shape.

The problem of handwriting recognition has been studied for decades and many methods have been developed. Some use only the pixel-image as input to a powerful statistical or neural classifier. Others preprocess the data in order to extract some features that are fed into a classifier. Structural methods of digit recognition rely on structural information in order to produce a classification decision.

None of those approaches is able to solve the problem perfectly. All classifiers have their particular strengths and weaknesses. Pixel oriented methods, for example, are able to tolerate structural defects much better than structural methods. In contrast, the latter perform well on deformed images that have a typical structure. Thus, it is worth to investigate how to combine the outputs of different classifiers. If the individual classifiers don't make the same mistakes and there is a way to judge the reliability of a classifiers output, the overall classification performance is expected to improve.

There are several methods to combine classifiers. Some combination methods use only the class-labels of the individual classifiers e.g. for a voting mechanism [1]. Other methods incorporate more information from the classifier output, like confidence values for all possible classes and suggestions for rejections [6]. In this case, the function which the combination system has to approximate is usually complex and highly nonlinear. Therefore, the utilization of neural networks for the combination seems promising.

The remainder of the paper is organized as follows. In section 2 the individual classification systems are described. Some approaches for the combination of classifiers are introduced in section 3. The performance of the individual classifiers is reported in section 4 and is compared to the results of the investigated combination methods. The paper concludes with a discussion of the results and an outlook to future work.

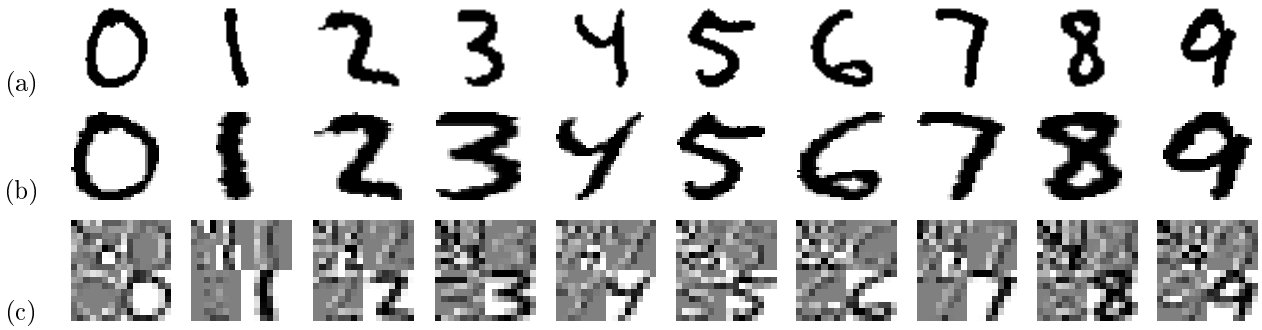

Figure 1: Digits from the NIST-Database: (a) original binary images, (b) slant and size normalized (32×32), (c) wavelet representation (16×16).

2 Individual Classifiers

In order to develop and evaluate a reliable system for the recognition of handwritten digits, large datasets that contain all typical images of handwritten digits are needed. The *NIST Special Databases 19* is such a large writing sample. There are about 180.000 isolated binarized digits available from NIST [3] that have been extracted from handwriting sample forms. These digits were written by 1.500 different persons, distributed across the United States. Some of these digits are shown in Figure 1(a).

The available digits were partitioned as follows. The first 58.646 digits were used for the training of the individual classifiers and the last 30.000 digits were used as an independent validation set. The remaining digits were used to train the neural networks that combine the classifier outputs and for boosting.

2.1 Classification based on Wavelet-Preprocessing

In order to reduce the variance of the images that is not relevant for recognition, some preprocessing is applied prior to the classification. First, the vertical main axis of the digit is estimated and a horizontal shear transformation is used to normalize it to be exactly vertical. In the next step the digit is scaled into a 32×32 box. This scaling may distort the aspect ratio to a limited degree only, centering the digit in the box, if necessary (see figure 1(b)).

The last preprocessing step produces a wavelet representation that contains the digit in different resolutions and decomposed by directed filters. A 2-D Haar-wavelet transformation is applied to the 32×32 pixel image, decomposing it into vertical, horizontal and diagonal detail images of different resolutions (16×16, 8×8, 4×4, 2×2). The diagonal details seem to be irrelevant for recognition and are replaced by sub-sampled versions of the original image. The data is then reduced further by accumulating each 2×2 window of the representation to a single pixel, which also reduces the noise. The resulting images (see Figure 1(c)) are of size 16×16 and contain the information that is relevant for classification.

Two different neural classifiers use the wavelet representation as input. Both have a 256-256-10 feed-forward architecture with one hidden layer. The first network uses hidden units that have a *Radial-Basis-Function (RBF)* characteristic. In contrast to most RBF-networks, the activation function used here is not a Gaussian, but has a hyperbolic shape, defined by $1/(1+r||\mathbf{x}-\mathbf{w_i}||^2)$, where $\mathbf{x}$ is the input vector, $\mathbf{w_i}$ denotes the weight vector of the hidden unit with index i, and r determines the size of the activation function. The second network, referred as BP-classifier, uses Σ-units with a sigmoidal output function in the hidden layer. The output-layer of both networks consists of one sigmoidal Σ-unit for each class. Both networks are trained using the backpropagation algorithm [5, 8].

2.2 Classification based on Structural Features

This hybrid classifier is described in more detail in [2]. Structural information and quantitative features are extracted from the pixel image of the digit and are used for classification. The goal is to preserve the information essential for recognition and to discard the unnecessary details. Figure 2 shows the stages of the recognition process.

In the *vectorization* step the normalized and smoothed digit is skeletonized. Nodes are placed on the skeleton and connected according to the principles of Gestalt psychology to represent strokes. After the graph is simplified, the digit is represented by a line-drawing.

A more abstract representation consisting of strokes which are merged to larger curves is constructed in the *structural analysis* step. A stroke is formed by several consecutive lines, which don't include turning points or sharp angles and don't cross other strokes. A set of strokes is merged to a curve if the strokes form a good continuation of each other. Quantitative features, such as rotation angle and length of the curves and the normalized coordinates of characteristic points are computed. After a simplification step, they are used to represent the digit by an attributed structural graph.

In the *prototype matching* step the structural graph is matched to structural prototypes that have been extracted from the training set. These typical structures represent not only perfect digits, but frequent structural deviations as well. For each prototype there is a neural classifier which is used in the *classification* step to distinguish digits having the same structure based on the extracted quantitative features. Feed-forward networks are trained using the Cascade-Correlation [8] algorithm on the digits of the training set that match the corresponding structural prototype. The training is terminated and a rejection parameter is determined based on the performance on a test set.

2.3 Classification using Time-Delayed Neural Networks

The *Time-delayed neural network (TDNN)* classification system described here is another approach to construct a robust classification system. Some of the variance of the digits is removed by preprocessing techniques, such as size and slant normalization as described in section 2.1. In order to be additionally invariant to nonlinear deformations, the system has to detect those characteristic features of the digit, which also help us humans to discriminate and 'classify' it (see Figure 3). These features may be shifted horizontally (Figure 3 a), vertically, or in both directions (Figure 3 b), depending on the writer, the digit and the preconditioning (slant removal) of the digit. Attractive and typical features of TDNNs, such as weight sharing receptive fields or the inclusion of perceptron activations of previous time-steps into actual computations have proofed successful in the detection of distinct patterns in larger context. Thus, the use of a TDNN to scan the normalized pixel image of the digit to perform a shift-invariant detection of its features seems appropriate.

The general architecture of TDNNs is shown in Figure 4 (for more details see e.g. [8]). Each group of input nodes (called the *receptive fields* with shared weights) 'sees' only a small window of the input stream, which 'marches' through the windows one step further in each time-step. The output of the hidden layer is also covered with receptive windows using shared weights. The network output consists of the sum of squares of the different time-steps of the output neurons.

The *input* of the TDNN consists of the gray-scale image of an isolated digit, which is scaled to a fixed height P_H and a fixed width P_W. There are R_1 receptive fields, each one 'sees' R_0 columns of the picture. During the scanning process the columns are moved through the input window of the receptive fields. Best results have been obtained with field-sizes $R_0 \approx P_W$. The different windows are shifted by a fixed amount of columns of the pixel-image. The *hidden layer* consists of N_H hidden nodes in R_1 time-states. This is realized by connecting each group of N_H hidden nodes with the corresponding input window. The *output layer* of the network consists of 10 nodes, each one representing one class of the digits '0' to '9', which are fully connected to *all* hidden nodes. The output layer thus works without receptive fields. This modification of the standard TDNN is motivated by two ideas. First of all, it accelerates and simplifies the learning

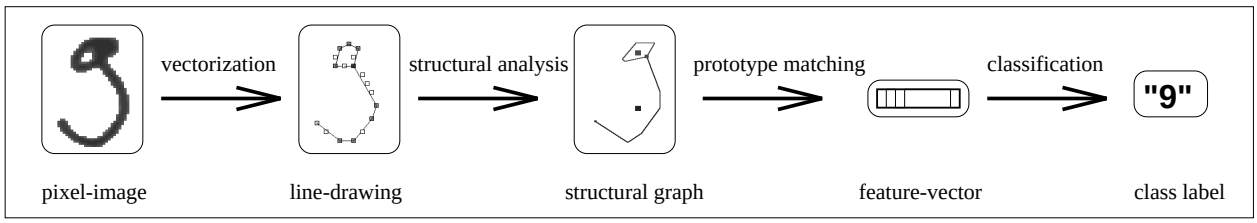

Figure 2: Stages of Recognition.

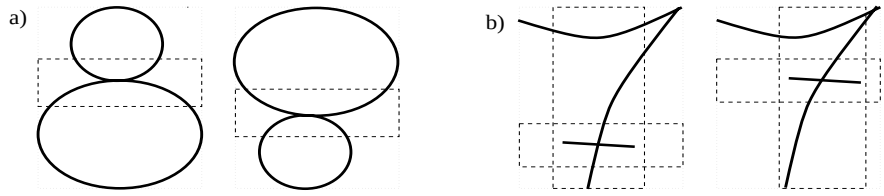

Figure 3: Locations of characteristica of the same digit depend on the writer.

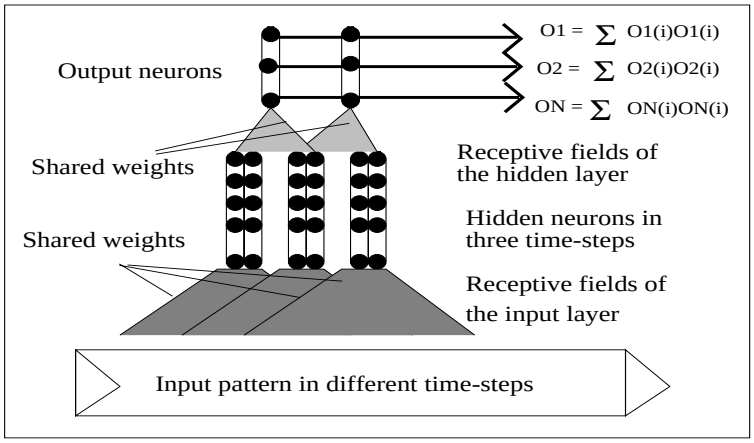

Figure 4: Receptive fields with shared weights in different layers of the TDNN.

algorithm of the TDNN. Secondly, working in this manner, the output layer gets a 'full view' over all time states of all hidden nodes, which also significantly improves the networks performance.

The TDNN is trained using a simple online gradient descent algorithm, similar to backpropagation [7, 8]. The only difference is, that during the weight-update step the corrections for the different receptive fields have to be averaged to guarantee the identity of the weights of the different receptive fields.

3 Combining Classifiers

For the combination of the available N classification experts $e^1, \ldots, e^N$ we assume that each of the experts e^i outputs a confidence-vector $\mathbf{o^i}$ with one entry per class when asked to classify a digit. These confidence-values $o^i_j, i \in \{1, \ldots, N\}, j \in \{1, \ldots, M\}$ are restricted to the interval $[0, 1]$ and constitute a generalized one-out-of-M-coding, where $M = 10$ is the number of possible classes. Furthermore, we assume that the classifier e^i indicates that it is not able to reliably classify the presented digit by a reject suggestion r^i. This binary signal can either be generated internally by the classifier or externally by inspecting the confidence values of its output-vector $\mathbf{o^i}$. In the latter case, we choose to set $r^i = \mathsf{TRUE}$, iff $o^i_{max} < o^i_{sec} + R$, where o^i_{max} and o^i_{sec} are the maximal and second largest confidences and the reject parameter $R \in [0, 1]$ determines the reject rate.

3.1 Parallel Combination

The parallel combination of classifiers described here is depicted in figure 5(a). In the parallel combination all N experts are consulted. The output vectors $\mathbf{o^1}, \ldots, \mathbf{o^N}$ of the individual experts are concatenated to form the input of the combination network. The length of this input vector MN grows with the number of experts. The task of the combination network c is to produce an output-vector $\mathbf{o^c}$ that codes the correct class of the presented digit. Several strategies are possible to achieve this. One could add or multiply the corresponding confidence values or use maximal/minimal values. Furthermore, one could apply voting or ranking schemes. All these approaches assume a certain unique interpretation of the confidence values, for

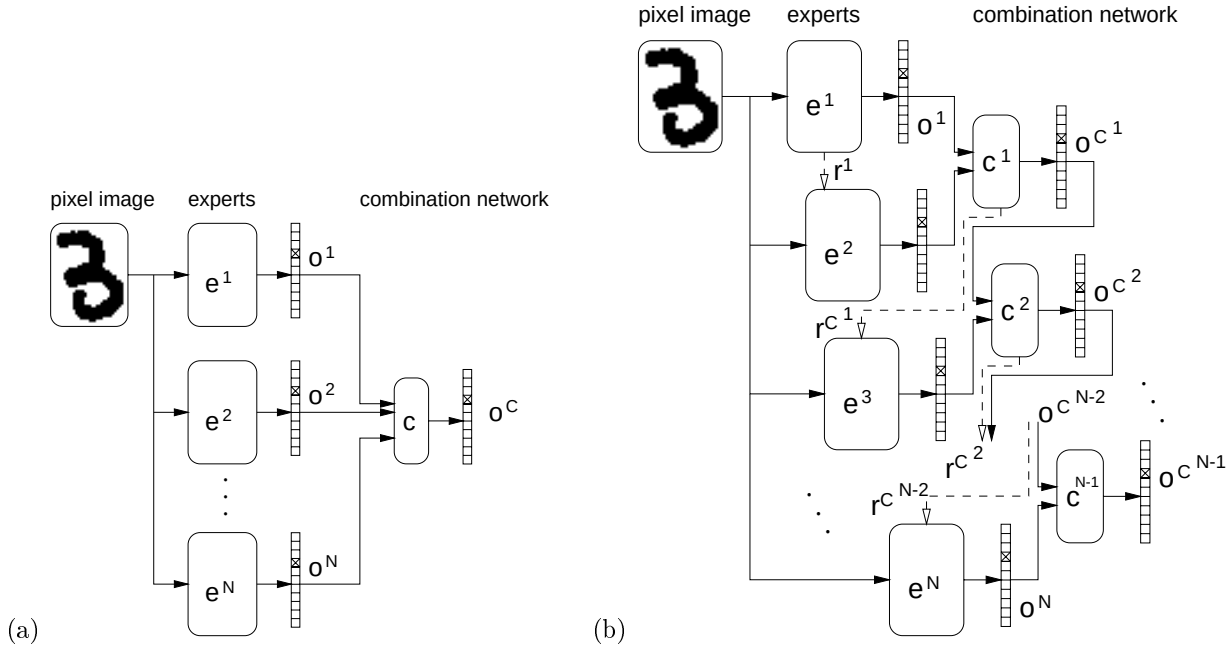

Figure 5: Classifier combination: (a) parallel, (b) sequential.

instance as a posteriori probabilities. Due to the specific characteristics of the individual classification experts this is not the case. For instance the structural classifier signals by a confidence value of 0.9 that it is almost sure about the classification decision, while the same confidence value at the output of the TDNN would signal uncertainty. Thus, the function that the combination network has to approximate is highly nonlinear. Feed-forward neural networks (FFNN) are a suitable tool for such function approximation problems. We use here a simple FFNN with one hidden layer. The training is done on a set of digits that is independent of the training set that was used for the training of the individual classifiers. An independent test set is used to stop the training. The performance of the parallel combination network is evaluated on an independent validation set.

3.2 Sequential Combination

In order to produce a classification decision, the parallel combination has to evaluate all experts. This is not necessary for most of the digits, since in typical cases the experts agree on the classification decision. These digits could as well be classified when only a subset of the experts is consulted. The sequential combination that is shown in figure 5(b) takes advantage of this situation. The classification experts are arranged in a chain where the individual classifiers are evaluated in sequential order. A classifier e^i in this chain is only consulted if the combined output $\mathbf{o^{c^{i-2}}}$ of the previous classifiers suggests to reject the digit. The combination network consists of a sequence of simple 2M-H-M FFNNs c^i that combine the output vector $\mathbf{o^{i+1}}$ of the corresponding expert e^{i+1} and the output vector $\mathbf{o^{c^{i-1}}}$ of the previous FFNN c^{i-1}. The output of the entire combination network is the output $\mathbf{o^{c^k}}$ of the FFNN c^k that first suggests via r^{c^k} to accept the presented digit.

With this design the combination network can not recover from substitutions. A wrong classification decision of a combination network c^k can only be modified by experts $e^l, l > k+1$, later in the chain, if the digit has not been accepted before. Consequently, one has to place at the beginning of the chain the classifiers that are able to produce very low substitution rates when allowed to reject a larger portion of the digits. At the end of the chain this property gets less important. Here a low substitution rate for the zero-rejection case is crucial.

The second factor that has to be considered when determining the sequence of classifiers is their running times. Since the first expert is consulted for every digit, it is useful to pick the fastest classifier for the first

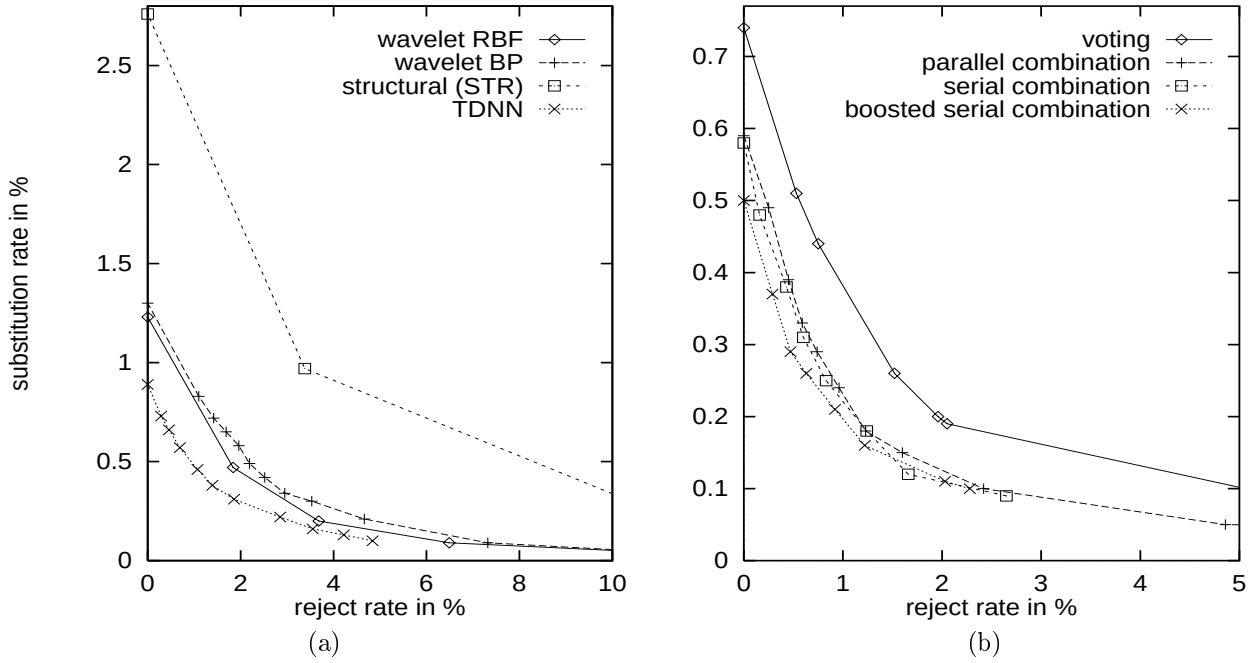

Figure 6: Performance of (a) individual classifiers, (b) classifier combinations.

position. The experts placed later in the chain receive smaller and smaller fractions of the digits for detailed inspection. Therefore, they can allocate longer running times without causing a significant increase of the average running time. In fact, if we are able to design a fast first classifier, the sequential combination of experts will be faster than most of the incorporated individual classifiers.

The sequential combination acts like a filter that passes only the digits that are hard to recognize to the classification experts at the end of the chain. Thus, these experts operate on a dataset that is very different from the original training set. Boosting techniques [9] are a way to take advantage of that fact. The classification experts are presented resampled versions of the training set in order to increase the classification expertise for the hard digits at the expense of the classification results for the easy digits. This competence decrease for the easy digits does not impact the overall performance, since these digits are recognized earlier in the chain. Unlike most boosting techniques, we do not focus on the substitutions from earlier experts, but on rejections, since the later expert is not consulted for substitutions. We use the digits that have been rejected from the combined preceding experts as a training set for classifier modification. Since this set is not large, the training does not start from scratch, but the parameters of the classifier are initialized using the parameters that have been learned from the original training set. In order to resemble the distribution of the arriving digits, it is necessary to extract them from a set of digits that has not been used for the training of the preceding combined classifiers.

4 Experimental Results

4.1 Individual Classifiers

Results for the individual classifiers on an independent validation set of 30.000 digits are shown in figure 6(a) where substitution rates are plotted against the rejection rates. The curves show that there is a tradeoff between reliability and recognition rate. The TDNN has the best performance of the four individual classifiers. When no digits are rejected, the TDNN classifies 99.11% correctly. The reliability of the recognition can be improved to 99.90% when rejecting 4.84% of the digits. The classifiers that use the wavelet-preprocessing substitute slightly more digits than the TDNN. At zero-rejection the RBF-classifier has a substitution rate of 1.23% and the BP-classifier substitutes 1.30% of the digits. This is a good result when considering the

9:	4:	9:	9:	9:	9:	8:	8:	1:	3:
5 (.85),	8 (.76),	7 (.79),	5 (.62),	4 (.61),	0 (.77),	3 (.84),	3 (.90),	2 (.80),	5 (.77),
9 (.19)	4 (.16)	9 (.19)	9 (.35)	9 (.55)	9 (.14)	8 (.02)	8 (.38)	1 (.08)	3 (.29)

Figure 7: The first ten substitutions from the validation set of the boosted sequential classifier combination. Correct label: best class (confidence), second best class (confidence).

fact that the TDNN has a much higher algorithmic complexity than the wavelet based methods. In addition, the classifiers that use the wavelet-preprocessing yield lower substitution rates than the TDNN when a larger portion of the digits is rejected. The RBF-classifier, for example, substitutes only 0.01% of the digits when it is allowed to reject 20%. The structural classifier yields the highest substitution rates. Most of its substitutions are due to structural deficits of the digits and can be avoided when allowing the classifier to reject ambiguous digits. For a rejection rate of 11.55% a substitution rate of 0.19% is observed. Although this is still higher than the respective rates of the other classifiers, the structural classifier is included in the combination because the structural approach is almost orthogonal to the pixel-oriented methods and therefore substitutes different digits.

4.2 Combined Classifiers

The performance of different methods for classifier combination is shown in figure 6(b). The first curve shows for comparison the results of a voting scheme. For this curve the three most confident classes of each classifier were given four, two and one vote, respectively. The normalized number of total votes was used as output confidence. This voting scheme resulted in a zero-rejection substitution rate of 0.74%. When rejecting 5.06% of the digits 0.10% of the digits were substituted.

The second curve shows the results of the parallel combination. FFNNs with different numbers of hidden units have been trained and benchmarked. The resulting zero-rejection substitution rates range from 0.61% for 16 hidden units, to 0.60% for 32 hidden units, to 0.59% for 64 hidden units. The latter FFNN was used to draw the curve. In order to reduce the substitutions to 0.10% a fraction of 2.42% of the digits had to be rejected.

In the third curve the performance of the sequential combination of the four classification experts is shown. The experts were arranged in the sequence: RBF, BP, STR, TDNN. When no rejections were allowed, the combination network substituted 0.58% of the digits. In this case, the RBF-classifier rejected 19.98% of the digits and substituted only four. The BP-classifier was adjusted to a rejection rate of 31.44% of the remaining digits. From the 4110 digits that were accepted by the combination of the first two classifiers only 18 were substituted. A number of 62 substitutions was caused by the combination of the STR-classifier with the former two. At this stage 14.01% of the arriving 1885 digits were rejected. For these 264 digits the TDNN was consulted and 91 were substituted by the last combination network. When not the zero-rejection rate, but the reduction of the substitutions is interesting, more digits need to be rejected by the FFNNs in the chain in order to be able to reject a larger percentage of the digits by the last combination network. About 2.35% of the digits need to be rejected in order to reduce the substitution rate to 0.1%. In this case the TDNN was consulted for 944 digits.

The last curve shows the results of the sequential combination after the TDNN was retrained on a set of 984 digits that were rejected by the combined preceding classifiers from a set of 30727 independent digits. This boosting improves the classification performance. The zero-rejection substitution rate drops to 0.50% and only 2.28% of the digits need to be rejected in order to reach a substitution rate of 0.1%. These results compare favorably to the results published by NIST [3, 4]. Figure 7 illustrates the difficulties of recognition. The first ten zero-rejection substitutions from the validation set are shown.

5 Conclusions

Four different classifiers for handwritten digits were briefly described in this paper and evaluated using the NIST-digits dataset. It was investigated how the outputs of individual classifiers can be combined. A parallel as well as a sequential combination scheme that utilize small feed-forward neural networks were introduced and benchmarked on the NIST data against a voting method.

The experimental results show that all investigated classifier combinations outperform the best individual classifier. Furthermore, the two proposed methods perform significantly better than voting. The sequential combination yields slightly better results than the parallel combination and has a much lower computational complexity. The average running time of the sequential combination is lower than the running times of most of the incorporated individual classifiers.

In addition, the performance of the sequential combination can be boosted when the classifiers later in the chain are modified using resampled versions of the training set. For this boosting a small subset of hard digits is selected from the original digit source. Since these digits need to be independent of the digits used to train the preceding experts and their combination network, the boosting consumes a significant part of the available digits.

Clearly, if one could provide a larger amount of interesting digits, the overall performance would benefit. Since the manual labeling of huge amounts of digits is not practical, two options seem promising. The first is to distort labeled digits in order to produce more examples. This approach requires the definition of a group of transformations for that the classification should be invariant. The second approach is to automatically select the hard digits from a source of unlabeled digits. These digits can then be labeled and used for boosting. The automatic selection could be done by using the digits that are rejected by a sequential combination of experts. We plan to investigate both options in future work.

References

[1] R. Battiti and A. M. Colla, "Democracy in Neural Nets: Voting Schemes for Classification," *Neural Networks*, vol. 7, no. 4, pp. 691–707, 1994.

[2] S. Behnke, M. Pfister, and R. Rojas, "Recognition of Handwritten Digits using Structural Information," *Proceedings ICNN'97–Houston*, vol. 3, pp. 1391–1396, June 1997.

[3] M. D. Garris *et al NIST Form-Based Handprint recognition System (Release 2.0).* NIST Internal Report 5959, 1997.

[4] P. J. Grother and G. T. Candela, "Comparison of Handprinted Digit Classifiers", Technical Report NISTIR 5209, NIST, 1993.

[5] N. B. Karayiannis, "Gradient Descent Learning of Radial Basis Neural Networks," *Proceedings ICNN'97–Houston*, vol. 3, pp. 1391–1396, June 1997.

[6] D-S. Lee and S. N. Srihari, "A Theory of Classifier Combination: The Neural Network Approach", *Proceedings ICDAR'95–Montreal* vol. 1, pp. 42–45, August 1995.

[7] M. Pfister. *Learning Algorithms for Feed-forward Neural Networks – Design, Combination and Analysis.* PhD Thesis, FU Berlin, 1995.

[8] R. Rojas. *Neuronal Networks.* Springer, New York, 1996.

[9] R. E. Schapire, Y. Freund, P. Bartlett and W. S. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods.", *In Machine Learning: Proceedings of the Fourteenth International Conference*, 1997.