IEEE TRANSACTIONS ON NEURAL NETWORKS, in press (1992).

Handwritten Digit Recognition by Neural Networks with Single-Layer Training

S. KNERR, L. PERSONNAZ, G. DREYFUS, Senior Member, IEEE
Ecole Supérieure de Physique et de Chimie Industrielles
de la Ville de Paris,
Laboratoire d'Electronique
10, rue Vauquelin
75005 PARIS, FRANCE

ABSTRACT

We show that neural network classifiers with single-layer training can be applied efficiently to complex real-world classification problems such as the recognition of handwritten digits. We introduce the STEPNET procedure, which decomposes the problem into simpler subproblems which can be solved by linear separators. Provided appropriate data representations and learning rules are used, performances which are comparable to those obtained by more complex networks can be achieved. We present results from two different data bases: a European data base comprising 8,700 isolated digits, and a zip code data base from the U.S. Postal Service comprising 9,000 segmented digits. A hardware implementation of the classifier is briefly described.

1 Introduction

1.1 The task

Optical Character Recognition is a typical field of application of automatic classification methods. In addition to its practical interest (zip code recognition, automatic reading of bank checks, etc.), it exhibits all the typical problems encountered when dealing with classification: choice of the data representation, choice of a classifier of suitable type and structure, and supervised training of the classifier using a set of examples. In this paper, we focus on the recognition of isolated handwritten digits, a task which is known to be difficult and which still lacks a technically satisfactory solution. Two "real world" data bases are used throughout the paper: a European data base of 8,700 isolated digits, and a data base of 9,000 segmented digits, originating from 2,000 zip codes provided by the U.S. Postal Service.

1.2 The approach

Since the criticism of single-layer perceptrons, mainly for their limitation to building linear separation surfaces [1], more powerful neural network classifiers have been developed. The most popular network is the Multilayer Perceptron (MLP) trained by the backpropagation algorithm [2]. It has been shown in various papers that MLP's with a single hidden layer are universal classifiers, in the sense that they can approximate decision surfaces of arbitrary complexity, provided the number of neurons in the hidden layer is large enough (see for instance [3]). However, there is no simple rule which indicates how many hidden units are required for learning a given task. Moreover, limitations on hardware requirements or computation time may influence the choice of the classifier and favor classifiers with simpler structures and faster training than MLP's.

Classically, the recognition process is divided into preprocessing steps and subsequent classification. Within our approach, the preprocessing operations do not involve learning.

In an earlier paper, we described a procedure, hereinafter termed the STEPNET procedure, whereby any classification problem defined in R^N can be decomposed into subproblems which are efficiently solved by a classifier having a single layer of trainable connections [4]. The network is built and trained simultaneously and automatically, without user's intervention. In this paper, we show, using two data bases, the efficiency of the procedure when applied to "real world" problems such as the recognition of handwritten digits. We compare the performances of two classifiers of identical sizes and structures, resulting from the above procedure, operating on the same data bases, with two different data representations: a simple pixel representation and a more elaborate feature representation. While the first data representation results from normalization as the only preprocessing, the second is obtained after feature extraction and normalization.

2 Data bases, preprocessing and performance estimation

2.1 Data bases

In this paper we use two different data bases: a European data base and an American zip code data base. The European data base consists of 8,700 digits from 13 different writers from our laboratory. The writers were told to draw the numerals into prepared square boxes in order to facilitate segmentation. As can be seen in Figure 1a, there is a great variety of sizes and writing styles. The numerals were digitized by a scanner, which had the drawback of erasing some of

the thin lines: the resulting black and white (binary) digits are thus sometimes disconnected.
The second data base consists of 9,000 digits from the U.S. Postal Service OAT Handwritten Zip Code Data Base (1987). In addition to variations in size and writing style, the segmentation problem is made more difficult by the existence of overlapping numerals, postmarks, horizontal bars and marks on the envelope. Therefore, many digits are cut in part, include extraneous marks or parts from other digits, and some digits could not be segmented automatically at all. Figure 1b shows some examples of zip codes.
The training set and the test set of both data bases contain highly ambiguous examples which are hardly recognizable by humans: some examples were even assigned to the wrong class during segmentation.

2.2 Choice of a data representation

The choice of an appropriate data representation is a crucial point when solving a classification task, either with a trainable or with a non-trainable classifier. If a relatively low-level data representation is used, one might need a very large training set in order to get satisfactory results, because, in general, the performance of the classifier on the test set is quite sensitive to the particularities of the training set; see for instance the recent paper by Geman et al. [5]. Therefore, the original input representation is usually transformed into a higher-level data representation by using human expertise for designing appropriate preprocessing operations. However, besides mere recognition rates, there are other factors which may influence the choice of the data representation, such as the need for high computation speed, or hardware limitations, favouring data representations which do not require complicated and time-consuming preprocessing. In many cases, this precludes the use of structured data representations. Therefore, the first data representation we use for the subsequent classification is a simple pixel representation.
For a given data representation, an optimal hypersurface separating the classes in the N-dimensional input space in the best possible way might be found if the underlying probability distributions were either known or estimated accurately; this hypersurface might be approximated by training some neural network classifier on the examples of the training set. Suppose the classifier has a set of parameters, the weights in the case of a neural network, which must be determined. The question then is: how many training samples does it take to achieve a given degree of accuracy for the approximation? While the complete answer to this question certainly depends on the complexity of the classification problem at hand, it is clear that the number P of samples needed to achieve a

given degree of accuracy grows exponentially with the dimension N of the input space; this is often called the "curse of dimensionality", which is particularly troublesome for real world vision problems such as the digit recognition task discussed herein.

In the case of handwritten digits, it seems therefore natural to use some a priori knowledge about the recognition task in order to transform the low level information of the pixel images into a data representation of higher level. Two possibilities arise: either the a priori knowledge is used to design preprocessing operations which are carried out explicitly before classification, or it is used to constrain the general architecture of the classifier, which results in a classifier specialized in the recognition task at hand. We have chosen the first possibility. Digits, whether handwritten or typed, are essentially line drawings, i.e. one-dimensional structures in a two-dimensional space. Therefore, local detection of line segments seems to be an adequate preprocessing. For each location in the image, information about the presence of a line segment of a given direction is stored in a feature map. Thus, the second data representation used for the subsequent classification consists of one feature map for each detected direction of line segments. The hope is that, with this data representation, the examples of a given class are less dispersed in input space than with the simple pixel representation.

2.3 Preprocessing

In this section we briefly outline the preprocessing leading to the two data representations. All the preprocessing steps, shown on Figure 2, involve no training.

The most difficult part of the preprocessing chain is the automatic segmentation of the zip codes into isolated digits, especially if no a priori knowledge about the characters and no feedback from the subsequent classification of the segmented parts is used. The dilemma is the following: if the class of the character to be segmented was known, one could most probably apply the appropriate segmentation technique in order to isolate the character; but of course in order to classify the character, it has to be segmented. In our approach, the zip codes are first thresholded, and connected parts are subsequently extracted. This procedure works automatically, but from all the extracted parts (including postmarks, bars and extraneous marks), those corresponding to one of the ten digit classes are sorted by hand; class labels are also assigned by hand (not without errors). At this stage of preprocessing, the data base consists of isolated binary pixel images of variable size.

The first data representation used for classification is a simple pixel representation. All isolated binary digits are normalized in size to smoothed 16 by 16 pixel images, using a transformation which is approximately linear, thereby preserving the shape of the digits. The normalized image has 16 gray levels per pixel which results in 256 4-bit inputs to the classifier.

The second data representation incorporates more a priori knowledge on the recognition problem by performing some hand-designed edge detection and feature extraction. The image is scanned by the 4 pairs of Kirsch masks (3x3 pixels) [6], resulting in 4 graded feature maps, coding for the presence of horizontal, vertical or diagonal edges as shown in Figure 2. There are no masks for detecting end-stops, curved lines, line crossings or other more complex features. As a final preprocessing step, the four feature maps are normalized to an 8 by 8 format using the same transformation as for the first data representation. The final data representation corresponds to 4x8x8 = 256 4-bit inputs, which is the same format as for the pixel representation.

Note that all preprocessing steps can be carried out by simple mask operations, i.e. weighted sums and comparisons. Thus, a digital signal processor, or a special-purpose chip such as described in [7] seems to be a good choice for a future implementation of the preprocessing steps.

2.4 Performance estimation

For subsequent classification, the resulting data bases are randomly partitioned into training set and test set. Because of the limited size of the data bases, one can only estimate the performance that would be obtained on the set of all possible patterns; in order to compute a meaningful estimation, we performed several random partitions of the data bases into training set and test set, and averaged the recognition rates measured on the various test sets. Furthermore, both sets contain examples with debatable or erroneous class labels, so that the result thus found is a pessimistic estimate of the actual performance.

3 Neural network classifier with single-layer training

3.1 The STEPNET procedure, network architecture

As was shown in an earlier paper [4], any classification problem defined in R^N can be decomposed into classification problems involving linear separation surfaces. The proposed procedure consists of three steps of increasing complexity, in the spirit of a "divide and conquer" strategy. In a first step, linear separation of each class from all others using a single neuron per class is attempted. In a second step, pairwise linear separation of the classes which were not separated during the previous step is tried. This corresponds to a transformation of the output coding in the network during training: while the first step uses the so-called "grand-mother" coding (each neuron codes for one of the ten digit classes), each neuron used in the second step discriminates between two classes only; therefore, in the second step, a maximum of 45 separations must be performed. Once all neurons have been trained, the ten final outputs of the network are obtained by appropriately ANDing the outputs of the neurons; this is explained in more detail in section 3.3. In a third step, pairs of classes which are still not separated, can be separated by piecewise linear decision surfaces; these are implemented by single-layer subnetworks and can be constructed using a recursive partitioning procedure in the spirit of binary decision trees (see for instance [8]). Throughout the three steps of the procedure, all neurons are trained independently. Since each neuron has only N=257 weights which are trained on P examples of the training set, it is rather easy to have an appropriate ratio P/N: for instance, in the case of the U.S. Postal Service data base, we use about 1400 examples for training each neuron in the second step.

In addition to the advantage of solving linearly separable subproblems instead of a complicated nonlinear problem, this procedure, as others proposed in the same spirit [9], generates automatically a network structure tuned to the complexity of the initial classification problem. Such procedures circumvent the problem of determining by trial and error a satisfactory network structure for Multilayer Perceptrons.

For both data bases and both data representations, the STEPNET procedure stopped after the second step, thereby indicating that the ten classes of the digit recognition problem are pairwise linearly separable. The resulting network is shown in Figure 3: a single layer of 45 neurons is fully connected to the 257 inputs; the final decision is made by ten AND gates. The network has 11,565 trainable weights.

3.2 Learning Rules

Each neuron of the discussed network is trained separately on two classes out of ten; the weight vector of each neuron defines a linear decision surface. As pointed out by other authors [10], the behaviour of linear classifiers depends heavily on the learning rule used. Whereas Perceptron-type rules only find solutions for linearly separable sets of examples, the Delta rule and the Generalized Delta rule minimize a Mean Square Error criterion (MSE) and converge to unique solutions for non-linearly separable sets as well. However, the Delta rule trains neurons with linear transfer functions and does not necessarily converge to a separating solution, even if such a solution exists. The

Generalized Delta rule trains neurons using sigmoidal transfer functions and is guaranteed to find a separating hyperplane, if such a hyperplane exists [4]. There are no local minima in the MSE criterion landscape and, by initializing the weights to zero and controlling the learning rate carefully, a gradient method always finds the minimum, i.e. the best hyperplane with respect to the MSE criterion. In the case of the Generalized Delta rule, this hyperplane is positioned so as to maximize the distance to the marginal examples of the classes. We chose the latter rule to train the 45 neurons of the network using the stochastic gradient method. The learning rate, which is the only free parameter of the training procedure, is fixed before training and is not changed thereafter.

When training neurons on the handwritten character recognition problem, we did not observe any effect of overspecialization of the classifier to the training data. While the MSE criterion was minimized, the recognition rate on the training set increased and came close to 100%, whereas the recognition rate on the test set reached a maximum after some 30 passes through the training set and then stayed roughly constant. Therefore, the stopping criterion is not critical with respect to the classification performances on the test set, but the training time can be greatly decreased by monitoring the recognition rate on a validation set. Figure 4 displays the Mean Square Error, the performance on the training set and the performance on the test set, as a function of the number of passes through the training set. Note that the use of a simple validation set or the more elaborate use of resampling techniques, such as random subsampling or cross-validation, raises the question of the significance of the validation sets, which is questionable for relatively small data bases; see for instance the critical remarks by Y. Chauvin [11].

Once the network is trained, the sigmoidal transfer functions of the neurons are replaced by simple threshold functions which provide binary inputs to the ten subsequent boolean operations.

3.3 Performance measure and rejection mechanism

In order to assess the performance of a classifier, three figures of merit must be considered: the number of well classified items, the number of errors (misclassified items), and the number of rejected items. For many applications, it is more important to minimize the number of errors than to maximize the number of well classified items, the price being a higher rejection rate; e.g. it is cheaper to sort a rejected letter by hand than to send it to a wrong city. Therefore, a realistic recognizer should implement a flexible rejection mechanism. In order to achieve this with our network, the values of the weighted sums (potentials) are taken into account for the final decision made by the AND gates: a small magnitude of a potential indicates an ambiguous situation. Figure 5 illustrates the neuron (i/j), separating class i from class j and the rejection mechanism: each neuron compares its potential $v_{(i/j)}$ to a common threshold θ; the two binary outputs $s_{(i/j)j}$ and $s_{(i/j)i}$ are then:

- if $v_{(i/j)}<-\theta$, then $s_{(i/j)j}=1$ and $s_{(i/j)i}=0$;
- if $v_{(i/j)}>\theta$, then $s_{(i/j)j}=0$ and $s_{(i/j)i}=1$;
- otherwise $s_{(i/j)j}=s_{(i/j)i}=0$, indicating an ambiguous input pattern.

The final decision of the network is as follows:

- if $s_{(i/j)i}=1$ for all j, the output of the AND gate "i" is one, and the input pattern is assigned to class i ;
- if all AND gates have zero outputs, the input pattern is rejected.

A low threshold results in a high percentage of well classified examples, whereas a high threshold yields a low error rate. Note that this rejection mechanism uses a single parameter in order to set the error rate to a prescribed percentage.

4 Results and comparison with other work

For both data bases, the single-layer network was trained using the simple pixel representation and the more elaborate feature representation. Since the network structure used for the two data representations is the same, we really compare data representations, not classifiers nor training procedures. The two data bases being of comparable size, we used a somewhat larger training set for the U.S. Postal Service data base than for the European data base: the first consisted of 80 % of the data base whereas the second consisted of only 50 % of the data base. This difference in the choice of the training set size reflects the fact that the U.S. Postal Service data base has even more variations in writing styles than the European data base. Tables 1 and 2 show the simulation results obtained on the European data base and on the U.S. Postal Service data base respectively. All results are averaged over 5 different partitions of the data base into training set and test set, with a standard deviation of approximately 1%.

As indicated by the performances on the training sets, the network learned the training set almost perfectly in all four cases; as pointed out in the previous section, the ten classes represented by the training sets are pairwise linearly separable: therefore the recognition rate on the training set would have reached 100 % if the training process had been continued. However, the performances on the test sets are quite different, especially when the misclassification rate is brought down to 1 %.

The results on the European data base are almost equally satisfactory for both data representations. When the error rate is further reduced to 0.1 %, we achieve a

19 % rejection rate. The recognition rates on the U.S. Postal Service data base vary strongly with respect to the data representation and are only satisfactory when the feature representation is used. This demonstrates impressively the importance of an appropriate data representation, especially when the data base shows a lot of variations in writing styles and when the size of the training set is limited. It also shows that performance comparisons based on results from different data bases should be taken carefully! Both data bases look rather difficult to a human, but, since the data representation used by the classifier is certainly different from the one used by humans, digits which seem to be easy to recognize for humans might cause difficult problems for the classifier and vice versa.

Our results on the U.S. Postal Service data base using the feature representation appear to be at the level of the present state of the art, which is roughly a 10 % rejection rate for a 1 % error rate for the recognition of handwritten digits without constraints on writing style. In comparison to other work, however, our classifier is simple and the size of the data base is still modest. Figure 6 shows the 18 examples (1 % of the test set) from the U.S. zip code data base which were misclassified when using the feature representation. For some of the misclassified examples we have a good explanation; e.g. for the *zero*, which is the only one in the data base tilted to the left, and therefore not represented in the training set, or for the first *three*, which looks more like a five because of the low resolution of the input image, or for the last *eight*, which is classified as a *one* and which in fact is a *one* (this is one of the cases for which a wrong class label was assigned during segmentation). Some other examples seem to be misclassified because their writing style is not sufficiently well represented by the training set. Considering all the variations of the writing styles in the data base, the latter is likely to be not large enough to be representative. By increasing the size of the data base, it would certainly be possible to further improve the performance.

We also trained Multilayer Perceptrons (MLP) on the two data bases. From Table 3 it can be seen that simulation results obtained with an unconstrained MLP with a single hidden layer and an optimized number of hidden units are not significantly better than the results obtained by our single-layer network. The MLP was trained using the standard stochastic back-propagation algorithm. The recognition rate on the test set was monitored (for simplicity we did not use a validation set) in order to stop the training process: after approximately 20-30 passes through the training set, the recognition rate reached a maximum and stayed roughly constant thereafter. Despite the relatively small number of training passes, training times for the MLP were one order of magnitude longer than for our network with single-layer training, which can be trained on the complete data base in less than 30 minutes on an Apollo DN10000 workstation. The rejection criterion used in order to set the error rate to 1 % was that the difference between the two highest outputs should exceed a given threshold.

Other authors incorporated a priori knowledge about the recognition task in the network architecture. The hope is that the first layers of the MLP will learn to perform preprocessing operations in the spirit of local feature detection, an objective which cannot be achieved without explicit help from the network designer. For character recognition, shift invariance can be implemented by the use of local receptive fields and the weight sharing technique [2]. A rejection rate of 9 % for a 1 % error rate was reported on the U.S. Postal Service data base [12]; these results are very similar to ours, which were obtained with a much simpler network, at the expense of a separate preprocessing (which is not necessary in the case of the European data base for instance). Therefore, at the present time, the decision as to whether the preprocessing should be performed by the network, or should be performed separately, relies mainly on implementation issues.

It has been argued by Martin et al. [13] that the impact of the data representation and of the network architecture is due basically to the fact that the data bases are of limited size. If the data base is large enough and the network has a minimum size in order to be able to learn the task, the performance of an unbiased classifier approximates the best possible for the given task. This is known as consistency in the statistical inference literature ([5] and references therein). For instance, it might well be that, due to consistency, our simple network classifier with single-layer training performs as well using the pixel representation as using the feature representation, provided it is trained on a very large data base.

To summarize, from the point of view of "real world" applications, mere recognition rates are not the only criterion for the choice of a digit recognizer. Whereas the performance of a classifier is measured in terms of its generalization ability, its cost can be measured in terms of complexity of the network (number and type of units, number of trainable connections), of training time and of classification time. We believe that the proposed network with single-layer training does not necessarily have the best recognition rates, but that it has an advantageous performance-to-cost ratio.

5 Hardware Implementation

An integrated circuit implementing the network described in Section 3 is in the test stage at the Laboratoire de Conception de Systèmes Intégrés (INPG, Grenoble) [14]. It uses standard 1.2 µm CMOS technology, 24 neurons being implemented on a single chip. Training is performed on a host computer; the 11,565 weights and the rejection threshold θ of the network can be loaded onto the chips. Table 4 shows simulation results obtained on the European data base using the pixel representation when weights are stored on 32 bits (floating-point arithmetics), 6 bits and 4 bits (integers) respectively. The threshold θ was chosen to set the misclassification rate to 1 %. Clearly, there is no substantial decrease in performance when the precision of the weights is brought down to 6 bits. The resulting moderate memory requirements, and the use of binary neurons, facilitate greatly the implementation of the network. Classification time for a 256-input pattern is estimated conservatively to 130 µs. This classification speed makes the circuit attractive when very fast classification is mandatory; this might be the case when performing automatic segmentation, whereby a large number of segmentation hypotheses are suggested by the recognizer and must be checked very quickly.

In a first version of our digit recognizer, all the preprocessing steps (segmentation, feature extraction, normalization) are performed by the host computer. In future versions, digital signal processors or dedicated template matching chips, such as the chip described by Graf et al. [7], will perform the multiplications and accumulations of the mask operations.

6 Conclusion

We have shown that our network, resulting from the STEPNET building and training procedure and using an appropriate data representation, leads to very satisfactory recognition rates on two moderately sized data bases of handwritten digits. A priori knowledge about the classification task is used to design explicitly the preprocessing steps; the classifier itself is very simple, featuring 45 binary neurons and a few logic gates. The performances of our classifier on "real world" data bases appear to be at the standard level of present-day recognizers, i.e. roughly a 10% rejection rate for a 1% error rate in the recognition of handwritten digits without constraints on writing style. Yet, in comparison to other networks, e.g. Multilayer Perceptrons, the structure of the discussed network is simpler, and it is generated automatically. In addition, the STEPNET procedure gives some insight into the difficulty of the classification problem: in the case of handwritten digits, it indicated that, surprisingly enough, the sets of examples were pairwise linearly separable. Both training times and classification times compare favorably with respect to Multilayer Perceptrons, and the simple structure of the network as well as the use of binary neurons facilitate greatly hardware implementation. An integrated circuit, performing the classification of a digit in 130 µs, has been designed and fabricated; its speed makes it a good candidate for performing the classification task necessary for the automatic segmentation of zip codes.

Acknowledgments

We thank the U.S. Postal Service for providing us with the OAT Handwritten Zip Code Data Base (1987). This work was supported in part by EEC BRAIN contract ST2000422.

References

[1] M. Minsky, S. Papert, *Perceptrons*, MIT Press, Cambridge MA, 1969.

[2] D.E. Rumelhart, G.E. Hinton, R.J. Williams, "Learning Internal Representations by Error Propagation", in *Parallel Distributed Processing,* Vol. 1, MIT Press, 1986.

[3] K. Hornik, "Multilayer Feedforward Networks are Universal Approximators", *Neural Networks*, Vol. 2, pp.359-366, 1989.

[4] S. Knerr, L. Personnaz, G. Dreyfus, "Single-layer Learning Revisited: A Stepwise Procedure for Building and Training a Neural Network", NATO Workshop on Neurocomputing, Les Arcs, France (February 1989); in *Neurocomputing*, F. Fogelman, J. Hérault, eds , Springer, 1990.

[5] S. Geman, E. Bienenstock, R. Doursat, "Neural Networks and the Bias/Variance Dilemma", preprint, 1991.

[6] W.K. Pratt, *Digital Image Processing,* Wiley, 1978.

[7] H.P. Graf, R. Janow, D. Henderson, R. Lee, "Reconfigurable Neural Net Chip with 32K Connections", in *Neural Information Processing Systems*, R.P. Lippmann, J.E. Moody, D.S. Touretzky, eds. , Morgan Kaufmann, 1991.

[8] L. Breiman, J. H. Friedman, R.A. Olshen, C.J. Stone, *Classification and Regression Trees*, Wadsworth International, 1984.

[9] M. Mézard, J.P. Nadal, "Learning in Feedforward Layered Networks: the Tiling Algorithm", *J. Phys.* A 22, 2191-2203, 1989.
T.D. Sanger, "A Tree-Structured Algorithm for Reducing Computation in Networks with Separable Basis Functions", *Neural Computation* 3, 67-78, 1991.
G.Z. Sun, Y.C.Lee, H.H. Chen, "A Novel Net that Learns Sequential Decision Process", in *Neural Information Processing Systems*, D.Z. Anderson, ed., American Institute of Physics, 1988.
P.E. Utgoff, "Perceptron Trees: A Case Study in Hybrid Concept Representations", *Connection Science* 1, 377-391, 1989.

[10] E. Barnard, D. Casasent, "A Comparison between Criterion Functions for Linear Classifiers, with an Application to Neural Nets", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 19, No. 5, 1989.
B. Wittner, J. Denker, "Strategies for Teaching Layered Networks Classification Tasks", in *Neural Information Processing Systems*, D.Z. Anderson, ed., American Institute of Physics, 1988.

[11] Y. Chauvin, "Generalization Dynamics in LMS Trained Linear Networks", in *Neural Information Processing Systems*, R.P. Lippmann, J.E. Moody, D.S. Touretzky, eds., Morgan Kaufmann, 1991.

[12] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, "Backpropagation Applied to Handwritten Zip Code Recognition", *Neural Computation* 1, 541-551, 1989.

Y. LeCun, L.D. Jackel, B. Boser, J.S. Denker, H.P. Graf, I. Guyon, D. Henderson, R.E. Howard, W. Hubbard, "Handwritten Digit Recognition: Applications of Neural Net Chips and Automatic Learning", in *Neurocomputing*, F. Fogelman, J. Hérault, eds. , Springer, 1990.

B. Boser, E. Säckinger, J. Bromley, Y. LeCun, R. Howard, L. Jackel, "An Analog Neural Network Processor and its Application to High-speed Character Recognition", in *Proceedings of IJCNN'91*, I-415 - I-420, 1991.

[13] G.L. Martin, J.A. Pittman, "Recognizing Hand-Printed Letters and Digits Using Backpropagation Learning", *Neural Computation* 3, 258-267, 1991.

[14] P.Y. Alla, G. Saucier, S. Knerr, L. Personnaz, G. Dreyfus, "Design and Implementation of a Dedicated Neural Network for Handwritten Digit Recognition", in *Silicon Architectures for Neural Networks*, M.G. Sami, ed., Elsevier, 1991.

FIGURE CAPTIONS

Figure 1: Examples of isolated digits from the European data base, original and normalized (a) and examples of zip codes from the U.S. Postal Service data base (b).

Figure 2: The preprocessing steps including feature extraction and normalization.

Figure 3: Network architecture.

Figure 4: Mean square error (MSE) with respect to the training set and recognition rates as a function of the passes through the training set.

Figure 5: Neuron (i/j) and rejection mechanism.

Figure 6: The 18 examples from the U.S. Postal Service data base which were misclassified by the single-layer network using the feature representation. The upper index gives the real class, the lower index the class associated by our classifier.

Table 1: Results on the European data base using pixel and feature representation; w.c. = well classified, rej. = rejected, m.c. = missclassified.

Table 2: Results on the U.S. Postal Service data base using pixel and feature representation.

Table 3: Results on both data bases obtained with an unconstrained MLP with 50 hidden units.

Table 4: Simulation results from the European data base with restricted precision of the weights.