* In Proceedings of the 4th USPS advanced Technology Conference, Washington D.C., pp 1003-1011, November 1990.

Handwritten Character Recognition Using Neural Network Architectures*

O. Matan, R. K. Kiang, C. E. Stenard, B. Boser,
J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard,
L. D. Jackel, and Y. Le Cun
AT&T Bell Laboratories, Holmdel, N. J. 07733

Abstract

We have developed a neural-network architecture for recognizing handwritten digits. This network has 1% error rate with about 7% reject rate on handwritten zipcode digits provided by the U.S. Postal Service. In this paper, we discuss implementing this architecture in a real-world character recognition system. The main issue is the tradeoff between cost and benefits such as accuracy and speed. A method for combining independently trained networks to achieve higher performance at relatively low cost is presented.

Accurate estimates of the probability of correct recognition, as well as runner-up probabilities, are of ever-increasing importance as recognition systems move out of the lab into the real world. Per-character probabilities give us the information necessary for calculating per-field or multi-field probabilities. We discuss a method for normalizing output activations levels, thus providing a normalized score, which (for our network at least) is a good estimate of the probability. We also find that using this normalized score as a rejection threshold gives similar performance to previous rejection schemes.

We also discuss the important and complex relationship between rejection threshold, average number of errors, and the cost of errors.

1 Introduction

We have built a feed-forward multi-layer neural network architecture for recognizing handwritten digits (Le Cun et al., 1990). This network has shown good generalization performance. This was accomplished by combining "learning from examples" with careful engineering; we find that even

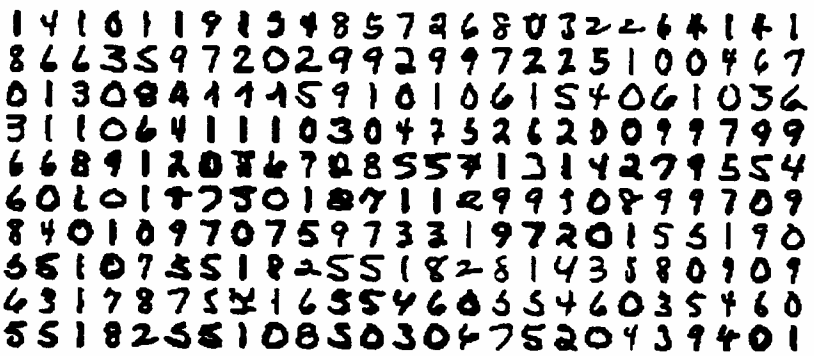

Figure 1: Examples of normalized digits from the testing set.

tens of thousands of examples will not suffice unless considerable *a priori* knowledge of the problem is incorporated into the architecture. The input to the network is a size-normalized pixel image and the output consists of 10 units (one for each class). The target output for a pattern belonging to class i is $+1$ for the ith unit and -1 for all the other output units. This network has been trained using back-propagation (Rumelhart, Hinton and Williams, 1986) on zipcode digits supplied by the U.S. Postal Service plus additional digits from other sources. Some examples of digits are shown in figure 1.

We have implemented several versions of this architecture, varying parameters such as the size of the input field. On normalized input sizes of 16 by 16 grey-level pixels, our best result is 1% error rate with 9% reject rate; when the input was 20 by 20 grey-level pixels, we achieved a 1% error rate with 7% reject rate. For a more detailed account of this work see (Le Cun et al., 1990).

This paper will discuss two ways of improving the performance of such networks in real-world applications: In section 2 we will present a method for combining outputs of two independently trained networks, thus decreasing the reject rate and/or error rate. In section 3 we discuss a method for normalizing the activity levels of the output units; this does not improve the recognition of individual digits, but provides us a reasonable estimate of the probabilities of correct classification. In section 4 we discuss the importance of coupling the rejection variable with a cost function, in order to determine the rejection threshold that measures the overall performance.

Network	0.5%	1%	2%
A	11.60	6.50	3.02
B	9.44	6.82	3.54
C	14.90	8.63	4.09
A+B	9.00	5.13	2.36
A+C	9.51	5.97	2.47
B+C	9.63	6.42	3.06
A+B+C	8.15	5.16	2.58

Table 1: Rejection percentage rates for 0.5%, 1% and 2% error rates on 2711 handwritten and printed characters. Results are shown for individual networks and for combinations.

2 Multi-Network Decisions

Let us consider combining two or more networks. At one extreme, it would be a waste of time to evaluate multiple networks if each one produced the same output for corresponding inputs. Fortunately, we find that different networks make different mistakes; therefore the combination will provide more information than a single network. Note that the multiple networks can easily be evaluated in parallel in cases where speed is important.

In table 1 are presented recognition performance rates for three independently trained networks of the 16 by 16 input architecture described in (Le Cun et al., 1990). The three networks A,B,C were all trained and tested on exactly the same data. Before training, a few of the weights (connection strengths) in each network were set to values that were expected *a priori* to be approximately correct, but the vast majority of the weights were *independently* set to random initial values. In network C, all the initial weights were random. In all cases, all weights were free to change during training.

The database consisted of 9298 segmented numerals extracted from handwritten zipcodes supplemented by 3349 machine printed digits from several fonts. The training set was composed of 7291 handwritten digits and 2645 printed digits. The remaining 2007 handwritten and 704 printed digits constituted the test set. The reject rate necessary for a certain error rate is presented for each individual network. One can see that the performance for network C is considerably worse. At the bottom of table 1 the results for various combinations of two networks are shown. Combining the two

Network	0.5%	1%	2%
A	16.14	9.66	4.98
B	13.79	9.61	6.23
C	18.92	12.75	7.61
A+B	12.25	8.70	4.73
A+C	15.04	8.61	5.18
B+C	13.20	9.72	6.12
A+B+C	12.81	8.17	4.93

Table 2: Rejection percentage rates for 0.5%, 1% and 2% error rates on 2007 handwritten characters. Results are shown for individual networks and for combinations.

networks was done by averaging the activation level of both output layers and treating the result as the output of the combined network.

One should not attribute too much precision to the tabulated numbers, since the total number of errors made by any of the networks is so small that there is significant "quantization noise" in the measurements — the difference between ten errors and eleven errors cannot be tremendously significant.

The combined network performance exceeds, in most cases, the performance of the individual networks. It is interesting that this is also the case for B+C where C is by far an inferior performer. The results for only the handwritten data are presented in table 2.

Discussion

There are two possible reasons that contribute to the explanation of why networks trained on the same data would make different mistakes. For one, we know that the training method, back-propagation, is imperfect. It is subject to getting stuck in local minima. (There exist methods for escaping from local minima, but the computation involved in finding the absolute best point in weight space is usually prohibitive.) More fundamentally, though, there may be no unique "best" place in weight space, but rather many large regions that are equally consistent with the training data. The training set is extremely limited in size, and we believe that it leaves the network *underdetermined.*

In either case, the trained network will retain vestiges of its (random) initial conditions. Because of the high dimensionality of weight space, there are, in effect, a vast number of ways in which the different networks can make different mistakes. This gives us grounds for hoping that multi-net voting will be effective, but there is no guarantee.

For highest accuracy, one would apply multiple networks to *every* input. Combining two networks requires twice the amount of computing power, but for many applications, high accuracy is so important that the added complexity is well worth it. It is possible that by using custom analog VLSI neural-net chips (Jackel et al., 1990; Graf and Henderson, 1990), the price/performance ratio will drop to the point that the cost of multiple-network recognition will be immaterial.

In any case, there is a way of achieving considerable improvement with *less* than a doubling of the computation, by using "multi-level rejection" as follows: One passes to the second network only the cases that were assigned low confidence by the first network. The computation increase is just the fraction of cases that need to be passed on for "a second opinion." For case A+B that appears in table 1, one needs to pass only 14% percent of the digits to a second recognition to maintain the same level of performance for the 1% and 2% error rate. In order to maintain the performance for 0.5% error rate, one has to pass 39% to a second recognizer.

3 Classification Probability Estimation

The output of a network is the set of activation levels of the output layer. In the previous section, the task was simply to choose, according to these values, one of eleven possible outcomes: rejection, or classification into one of the ten classes. No attempt was made to assign probabilities to the possible outcomes. In this section we present a method to normalize the output activation levels, and show that it forms a good estimator of these probabilities.

The ability to estimate the probability of the input belonging to class i is of great theoretical and practical importance. Given this ability we are able to model the classifier's performance. Combining this with a cost model we can tune the system to minimize cost. This is not a possibility when dealing with activity levels of the output layer. For the case of character recognition, knowledge of the probability of single character recognition makes it possible to calculate the probability for a multiple character recognition and to reject

on a per-field basis.

John Bridle recently proposed an normalization scheme for classifiers with N mutually-exclusive outcomes (Bridle, 1989). This scheme, which he has named Softmax, is applied as a post-processor to the "raw outputs" of the neural network. Softmax's appeal is that its outputs are positive and sum to 1, thus satisfying the axioms of probability theory. Each Softmax output is an increasing function of the corresponding raw output (when the other outputs are held constant) and preserves the ordering of the classes. There are other interesting theoretical qualities of Softmax, such as its connection to the entropy of the system (Bridle, 1989). The form of Softmax is the following:

$$S_i = \frac{e^{\beta O_i}}{\sum_k e^{\beta O_k}}$$

Where O_i is the activation level of output unit i, and S_i is the Softmax score for class i. We have slightly modified this function by adding an additional term to the denominator:

$$S_i = \frac{e^{\beta O_i}}{e^{\alpha} + \sum_k e^{\beta O_k}}$$

The term involving α essentially represents the activation level of an artificial $N+1$st category, the "none of the above" category. It will cause reduction of the score when the highest active unit has a low absolute value.

Softmax considers competition between the most-active unit and all the others (whereas the simple rejection scheme used in the previous section considers only competition between the top two classes). Tests we have conducted show that using Softmax with appropriate values of β gave essentially identical rejection performance to the simpler scheme. This indicates that there are relatively few cases where there is a many-way tie for second place.

For a discrete ("go/no-go") single-digit rejection decision, α is immaterial; its main importance is that it enables us to tune Softmax to be a reliable probability estimator for the most probable candidate classes returned by the neural network classifier. In figure 2 the Softmax score for the "winner" class is plotted versus the percentage of the cases that this class was in fact the correct answer. The distribution of the Softmax scores is also plotted. Since the raw recognition rate is well over 90% most of the population is concentrated in a small part of the histogram. Even so, one can see that the

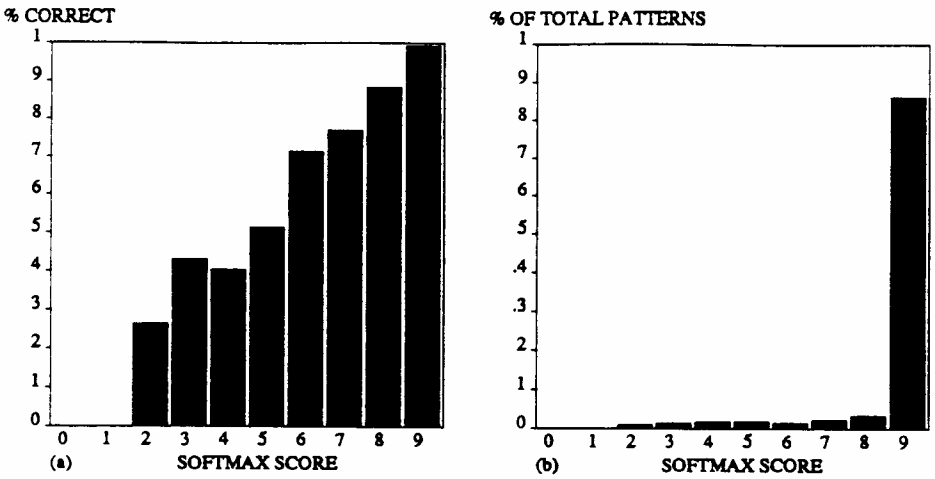

Figure 2: (a) Percentage of correct classification vs. the Softmax score supplied by the classifier for the highest scoring candidate ($\beta = 3.75, \alpha = 5$). The population distribution of this score on all the test set is plotted in (b).

Softmax score gives a good indication of the probability of correctness of the classification. In figure 3 the same functions are plotted for the second highest scoring class ("runner up"). In this case most of the population is concentrated in the low scoring values. Here, as well, Softmax seems to give a good estimation of the probability for this class to be the correct answer.

4 Cost, Rejection and Errors

In the previous sections we have discussed methods to process the output layer of a neural network in order to enhance the productivity of our classifying system. This post-processing stage is of great practical importance in making acceptance/rejection decisions. An issue of that is not addressed the full extent of its importance is *cost* of acceptance/rejection/error. The parameters that are usually minimized in the lab are the rejection rate at a constant error rate or vice versa. These are good measures for assessing the technology and measuring one's technical advancement. However in a real-world system, the true parameter one wants to minimize is the cost, therefore, the cost model must be incorporated into the rejection parameter. A typical example of this are the digits in a zip code. An error in the first digit is much more costly than an error in the last digit. This information must be encoded into the rejection value for the digit in question.

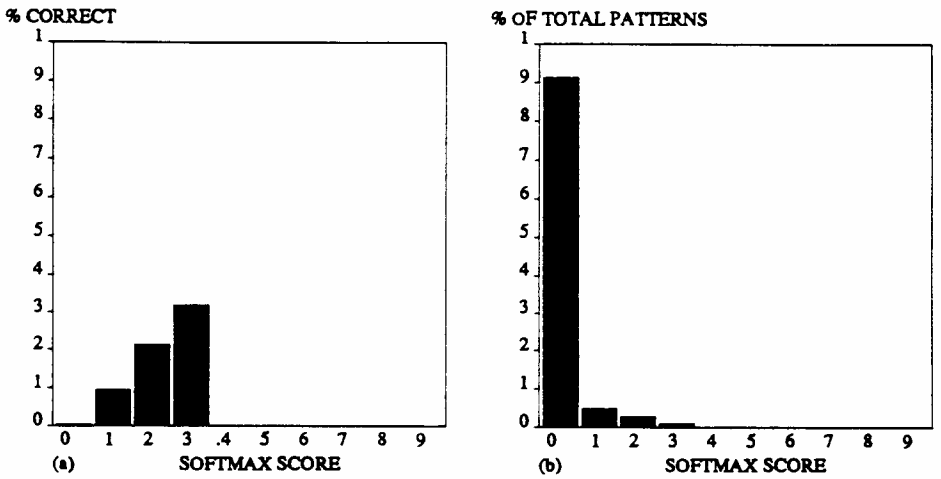

Figure 3: (a) Percentage of correct classification vs. the Softmax score supplied by the classifier for the second highest scoring candidate ($\beta = 3.75, \alpha = 5$). The population distribution of this score for the second candidate on all the test set is plotted in (b).

The conclusion is that even though the recognition engine would be the same, the acceptance/rejection decision would have to be tuned to the local problem. Knowledge that the majority of the mail stream is local can be incorporated into the rejection scheme to enhance the cost/performance on typical zipcodes.

Summary

We have presented two techniques for enhancing neural network classification and have applied them to a zipcode digit recognition problem. The results presented are far from a systematic study, but are promising enough to encourage further work.

This and other work (Denker and Le Cun, 1990) has brought us to view the propagation through a neural network as the *first stage* of a classification process. The output from the network is then fed into a post-processing stage that uses knowledge about the neural network and about the problem we are trying to solve. The two stage process generates superior performance to that one would expect from a stand-alone neural-network.

Acknowledgements

We thank the US Postal Service and its contractors for providing us with the zipcode database. We thank Henry Baird for providing the printed font database. We are grateful to Rodolfo Milito for comments on this manuscript.

References

Bridle, J. S. (1989). Probabilistic Interpretation of Feedforward Classification Network Outputs with Relationships to Statistical Pattern Recognition. In Fougelman-Soulie, F. and Hérault, J., editors, *Neurocomputing: algorithms, architectures and applications*. Springer-Verlag.

Denker, J. S. and Le Cun, Y. (1990). Unpublished.

Graf, H. P. and Henderson, D. (1990). A Reconfigurable CMOS Neural Network. In *ISSCC Dig. Tech. Papers*. IEEE Int. Solid-State Circuits Conference.

Jackel, L. D., Graf, H. P., Boser, B., Denker, J. S., Le Cun, Y., Howard, R. E., Matan, O., and Baird, H. S. (1990). Hardware Considerations for Neural-net Character Recognition Systems. In Simon, J., editor, *From Pixels to Features II*, (Bonas, France 1990).

Le Cun, Y., Matan, O., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., Jackel, L. D., and Baird, H. S. (1990). Handwritten Zip Code Recognition with Multilayer Networks. In *Proceedings of the 10th International Conference on Pattern Recognition*, (Atlantic City, NJ, 1990). IEEE Computer Society Press.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In *Parallel distributed processing: Explorations in the microstructure of cognition*, volume I, pages 318–362. Bradford Books, Cambridge, MA.