Communicated by Dana Ballard

Backpropagation Applied to Handwritten Zip Code Recognition

Y. LeCun
B. Boser
J. S. Denker
D. Henderson
R. E. Howard
W. Hubbard
L. D. Jackel
AT&T Bell Laboratories, Holmdel, NJ 07733 USA

The ability of learning networks to generalize can be greatly enhanced by providing constraints from the task domain. This paper demonstrates how such constraints can be integrated into a backpropagation network through the architecture of the network. This approach has been successfully applied to the recognition of handwritten zip code digits provided by the U.S. Postal Service. A single network learns the entire recognition operation, going from the normalized image of the character to the final classification.

1 Introduction

Previous work performed on recognizing simple digit images (LeCun 1989) showed that good generalization on complex tasks can be obtained by designing a network architecture that contains a certain amount of a priori knowledge about the task. The basic design principle is to reduce the number of free parameters in the network as much as possible without overly reducing its computational power. Application of this principle increases the probability of correct generalization because it results in a specialized network architecture that has a reduced entropy (Denker *et al.* 1987; Patarnello and Carnevali 1987; Tishby *et al.* 1989; LeCun 1989), and a reduced Vapnik–Chervonenkis dimensionality (Baum and Haussler 1989).

In this paper, we apply the backpropagation algorithm (Rumelhart *et al.* 1986) to a real-world problem in recognizing handwritten digits taken from the U.S. Mail. Unlike previous results reported by our group on this problem (Denker *et al.* 1989), the learning network is directly fed with images, rather than feature vectors, thus demonstrating the ability of backpropagation networks to deal with large amounts of low-level information.

2 Zip Codes

2.1 Data Base. The data base used to train and test the network consists of 9298 segmented numerals digitized from handwritten zip codes that appeared on U.S. mail passing through the Buffalo, NY post office. Examples of such images are shown in Figure 1. The digits were written by many different people, using a great variety of sizes, writing styles, and instruments, with widely varying amounts of care; 7291 examples are used for training the network and 2007 are used for testing the generalization performance. One important feature of this data base is that both the training set and the testing set contain numerous examples that are ambiguous, unclassifiable, or even misclassified.

2.2 Preprocessing. Locating the zip code on the envelope and separating each digit from its neighbors, a very hard task in itself, was performed by Postal Service contractors (Wang and Srihari 1988). At this point, the size of a digit image varies but is typically around 40 by 60 pixels. A linear transformation is then applied to make the image fit in a 16 by 16 pixel image. This transformation preserves the aspect ratio of the character, and is performed after extraneous marks in the image have been removed. Because of the linear transformation, the resulting image is not binary but has multiple gray levels, since a variable number of pixels in the original image can fall into a given pixel in the target image. The gray levels of each image are scaled and translated to fall within the range -1 to 1.

3 Network Design

3.1 Input and Output. The remainder of the recognition is entirely performed by a multilayer network. All of the connections in the network are adaptive, although heavily constrained, and are trained using backpropagation. This is in contrast with earlier work (Denker *et al.* 1989) where the first few layers of connections were hand-chosen constants implemented on a neural-network chip. The input of the network is a 16 by 16 normalized image. The output is composed of 10 units (one per class) and uses place coding.

3.2 Feature Maps and Weight Sharing. Classical work in visual pattern recognition has demonstrated the advantage of extracting local features and combining them to form higher order features. Such knowledge can be easily built into the network by forcing the hidden units to combine only local sources of information. Distinctive features of an object can appear at various locations on the input image. Therefore it seems judicious to have a set of feature detectors that can detect a particular

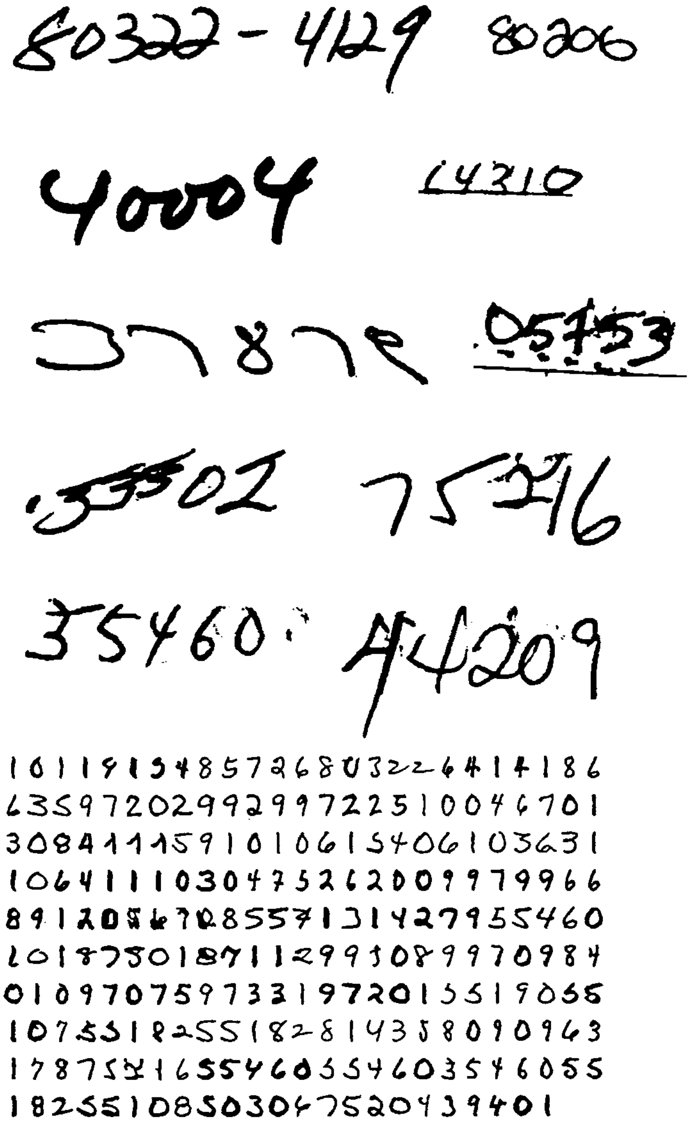


Figure 1: Examples of original zip codes (top) and normalized digits from the testing set (bottom).

instance of a feature anywhere on the input plane. Since the *precise* location of a feature is not relevant to the classification, we can afford to lose some position information in the process. Nevertheless, *approximate* position information must be preserved, to allow the next levels to detect higher order, more complex features (Fukushima 1980; Mozer 1987).

The detection of a particular feature at any location on the input can be easily done using the "weight sharing" technique. Weight sharing was described in Rumelhart *et al.* (1986) for the so-called T-C problem and consists in having several connections (links) controlled by a single parameter (weight). It can be interpreted as imposing equality constraints among the connection strengths. This technique can be implemented with very little computational overhead.

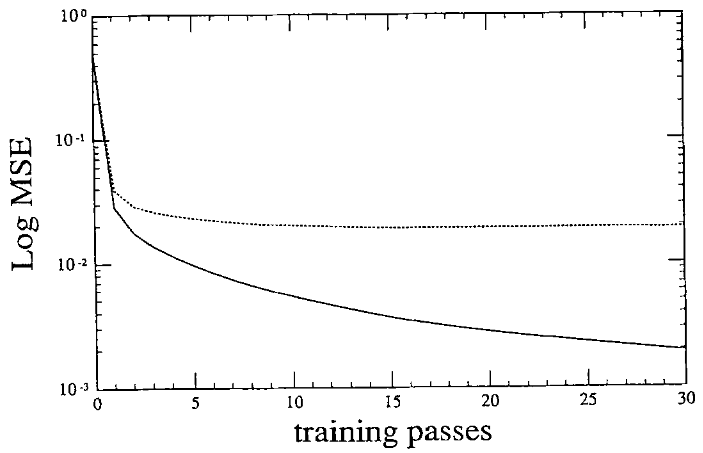
Weight sharing not only greatly reduces the number of free parameters in the network but also can express information about the geometry and topology of the task. In our case, the first hidden layer is composed of several planes that we call *feature maps*. All units in a plane share the same set of weights, thereby detecting the same feature at different locations. Since the exact position of the feature is not important, the feature maps need not have as many units as the input.

3.3 Network Architecture. The network is represented in Figure 2. Its architecture is a direct extension of the one proposed in LeCun (1989). The network has three hidden layers named H1, H2, and H3, respectively. Connections entering H1 and H2 are local and are heavily constrained.

H1 is composed of 12 groups of 64 units arranged as 12 independent 8 by 8 feature maps. These 12 feature maps will be designated by H1.1, H1.2, ..., H1.12. Each unit in a feature map takes input on a 5 by 5 neighborhood on the input plane. For units in layer H1 that are one unit apart, their receptive fields (in the input layer) are two pixels apart. Thus, the input image is *undersampled* and some position information is eliminated. A similar two-to-one undersampling occurs going from layer H1 to H2. The motivation is that high resolution may be needed to detect the presence of a feature, while its exact position need not be determined with equally high precision.

It is also known that the kinds of features that are important at one place in the image are likely to be important in other places. Therefore, corresponding connections on each unit in a given feature map are constrained to have the same weights. In other words, each of the 64 units in H1.1 uses the same set of 25 weights. Each unit performs the same operation on corresponding parts of the image. The function performed by a feature map can thus be interpreted as a nonlinear subsampled convolution with a 5 by 5 kernel.

Of course, units in another map (say H1.4) share *another* set of 25 weights. Units do not share their biases (thresholds). Each unit thus has 25 input lines plus a bias. Connections extending past the boundaries of the input plane take their input from a virtual background plane whose



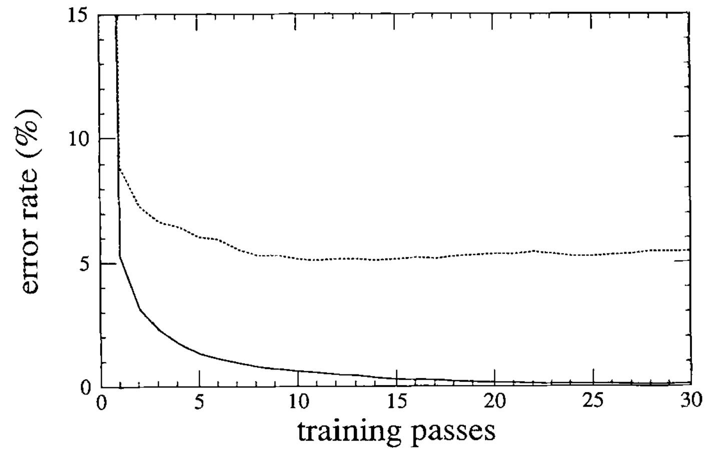


Figure 2: Network architecture.

state is equal to a constant, predetermined background level, in our case −1. Thus, layer H1 comprises 768 units (8 by 8 times 12), 19, 968 connections (768 times 26), but only 1068 free parameters (768 biases plus 25 times 12 feature kernels) since many connections share the same weight.

Layer H2 is also composed of 12 features maps. Each feature map contains 16 units arranged in a 4 by 4 plane. As before, these feature maps will be designated as H2.1, H2.2, . . . , H2.12. The connection scheme between H1 and H2 is quite similar to the one between the input and H1, but slightly more complicated because H1 has multiple two-dimensional maps. Each unit in H2 combines local information coming from 8 of the 12 different feature maps in H1. Its receptive field is composed of eight 5 by 5 neighborhoods centered around units that are at identical positions within each of the eight maps. Thus, a unit in H2 has 200 inputs, 200 weights, and a bias. Once again, all units in a given map are constrained to have identical weight vectors. The eight maps in H1 on which a map in H2 takes its inputs are chosen according a scheme that will not be described here. Connections falling off the boundaries are treated like as in H1. To summarize, layer H2 contains 192 units (12 times 4 by 4) and there is a total of 38,592 connections between layers H1 and H2 (192 units times 201 input lines). All these connections are controlled by only 2592 free parameters (12 feature maps times 200 weights plus 192 biases).

Layer H3 has 30 units, and is fully connected to H2. The number of connections between H2 and H3 is thus 5790 (30 times 192 plus 30 biases). The output layer has 10 units and is also fully connected to H3, adding another 310 weights. In summary, the network has 1256 units, 64,660 connections, and 9760 independent parameters.

4 Experimental Environment

All simulations were performed using the backpropagation simulator SN (Bottou and LeCun 1988) running on a SUN-4/260.

The nonlinear function used at each node was a scaled hyperbolic tangent. Symmetric functions of that kind are believed to yield faster convergence, although the learning can be extremely slow if some weights are too small (LeCun 1987). The target values for the output units were chosen within the quasilinear range of the sigmoid. This prevents the weights from growing indefinitely and prevents the output units from operating in the flat spot of the sigmoid. The output cost function was the mean squared error.

Before training, the weights were initialized with random values using a uniform distribution between $-2.4/F_i$ and $2.4/F_i$ where F_i is the number of inputs (fan-in) of the unit to which the connection belongs. This technique tends to keep the total inputs within the operating range of the sigmoid.

During each learning experiment, the patterns were repeatedly presented in a constant order. The weights were updated according to the so-called stochastic gradient or "on-line" procedure (updating after each presentation of a single pattern) as opposed to the "true" gradient procedure (averaging over the whole training set before updating the weights). From empirical study (supported by theoretical arguments), the stochastic gradient was found to converge much faster than the true gradient, especially on large, redundant data bases. It also finds solutions that are more robust.

All experiments were done using a special version of Newton's algorithm that uses a positive, diagonal approximation of the Hessian matrix (LeCun 1987; Becker and LeCun 1988). This algorithm is not believed to bring a tremendous increase in learning speed but it converges reliably without requiring extensive adjustments of the parameters.

5 Results

After each pass through the training set, the performance was measured both on the training and on the test set. The network was trained for 23 passes through the training set (167,693 pattern presentations).

After these 23 passes, the MSE averaged over the patterns and over the output units was 2.5×10^{-3} on the training set and 1.8×10^{-2} on the test set. The percentage of misclassified patterns was 0.14% on the training set (10 mistakes) and 5.0% on the test set (102 mistakes). As can be seen in Figure 3, the convergence is extremely quick, and shows that backpropagation *can* be used on fairly large tasks with reasonable training times. This is due in part to the high redundancy of real data.

In a realistic application, the user usually is interested in the number of rejections necessary to reach a given level of accuracy rather than in the raw error rate. We measured the percentage of test patterns that must be rejected in order to get 1% error rate on the *remaining* test patterns. Our main rejection criterion was that the difference between the activity levels of the two most active units should exceed a given threshold.

The percentage of rejections was then 12.1% for 1% classification error on the remaining (nonrejected) test patterns. It should be emphasized that the rejection thresholds were obtained using performance measures on the *test set*.

Some kernels synthesized by the network can be interpreted as feature detectors remarkably similar to those found to exist in biological vision systems (Hubel and Wiesel 1962) and/or designed into previous artificial character recognizers, such as spatial derivative estimators or off-center/on-surround type feature detectors.

Most misclassifications are due to erroneous segmentation of the image into individual characters. Segmentation is a very difficult problem, especially when the characters overlap extensively. Other mistakes are due to ambiguous patterns, low-resolution effects, or writing styles not present in the training set.

Other networks with fewer feature maps were tried, but produced worse results. Various fully connected, unconstrained networks were also tried, but generalization performances were quite bad. For example, a fully connected network with one hidden layer of 40 units (10,690 connections total) gave the following results: 1.6% misclassification on the

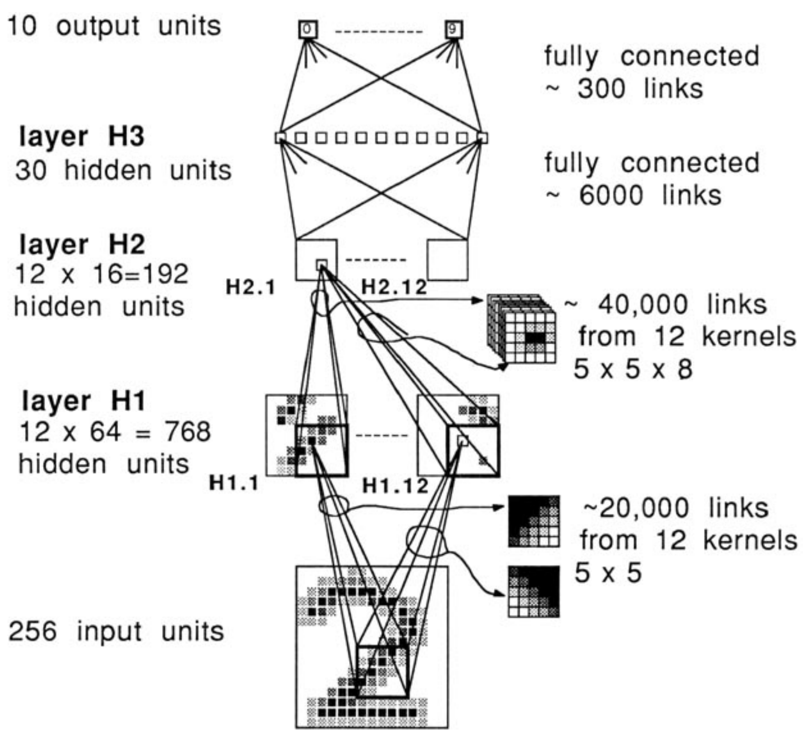


Figure 3: Log mean squared error (MSE) (top) and raw error rate (bottom) versus number of training passes.

training set, 8.1% misclassifications on the test set, and 19.4% rejections for 1% error rate on the remaining test patterns. A full comparative study will be described in another paper.

5.1 Comparison with Other Work. The first several stages of processing in our previous system (described in Denker *et al.* 1989) involved convolutions in which the coefficients had been laboriously hand designed. In the present system, the first two layers of the network are constrained to be convolutional, but the system automatically learns the coefficients that make up the kernels. This "constrained backpropagation" is the key to success of the present system: it not only builds in shift-invariance, but vastly reduces the entropy, the Vapnik–Chervonenkis dimensionality, and the number of free parameters, thereby proportionately reducing the amount of training data required to achieve a given level of generalization performance (Denker *et al.* 1987; Baum and Haussler 1989). The present system performs slightly better than the previous system. This is remarkable considering that much less specific information about the problem was built into the network. Furthermore, the new approach seems to have more potential for improvement by designing more specialized architectures with more connections and fewer free parameters. [1]

Waibel (1989) describes a large network (but still small compared to ours) with about 18,000 connections and 1800 free parameters, trained on a speech recognition task. Because training time was prohibitive (18 days on an Alliant mini-supercomputer), he suggested building the network from smaller, separately trained networks. We did not need such a modular construction procedure since our training times were "only" 3 days on a Sun workstation, and in any case it is not clear how to partition our problem into separately trainable subproblems.

5.2 DSP Implementation. During the recognition process, almost all the computation time is spent performing multiply accumulate operations, a task that digital signal processors (DSP) are specifically designed for. We used an off-the-shelf board that contains 256 kbytes of local memory and an AT&T DSP-32C general purpose DSP with a peak performance of 12.5 million multiply add operations per second on 32 bit floating point numbers (25 MFLOPS). The DSP operates as a coprocessor; the host is a personal computer (PC), which also contains a video acquisition board connected to a camera.

The personal computer digitizes an image and binarizes it using an adaptive thresholding technique. The thresholded image is then scanned and each connected component (or segment) is isolated. Components that are too small or too large are discarded; remaining components are sent to the DSP for normalization and recognition. The PC gives a variable sized pixel map representation of a single digit to the DSP, which performs the normalization and the classification.

The overall throughput of the digit recognizer including image acquisition is 10 to 12 classifications per second and is limited mainly by the normalization step. On normalized digits, the DSP performs more than 30 classifications per second.

6 Conclusion

We have successfully applied backpropagation learning to a large, real-world task. Our results appear to be at the state of the art in digit recognition. Our network was trained on a low-level representation of

[1]A network similar to the one described here with 100,000 connections and 2600 free parameters recently achieved 9% rejection for 1% error rate. That is about 30% better than the best of the hand-coded-kernel networks.

data that had minimal preprocessing (as opposed to elaborate feature extraction). The network had many connections but relatively few free parameters. The network architecture and the constraints on the weights were designed to incorporate geometric knowledge about the task into the system. Because of the redundant nature of the data and because of the constraints imposed on the network, the learning time was relatively short considering the size of the training set. Scaling properties were far better than one would expect just from extrapolating results of backpropagation on smaller, artificial problems.

The final network of connections and weights obtained by backpropagation learning was readily implementable on commercial digital signal processing hardware. Throughput rates, from camera to classified image, of more than 10 digits per second were obtained.

This work points out the necessity of having flexible "network design" software tools that ease the design of complex, specialized network architectures.

Acknowledgments

We thank the U.S. Postal Service and its contractors for providing us with the data base. The Neural Network simulator SN is the result of a collaboration between Léon-Yves Bottou and Yann LeCun.

References

Baum, E. B., and Haussler, D. 1989. What size net gives valid generaliztion? *Neural Comp.* **1**, 151–160.

Becker, S., and LeCun, Y. 1988. *Improving the Convergence of Back-Propagation Learning With Second-Order Methods*. Tech. Rep. CRG-TR-88-5, University of Toronto Connectionist Research Group.

Bottou, L.-Y., and LeCun, Y. 1988. Sn: A simulator for connectionist models. In *Proceedings of NeuroNimes 88*, Nimes, France.

Denker, J., Schwartz, D., Wittner, B., Solla, S. A., Howard, R., Jackel, L., and Hopfield, J. 1987. Large automatic learning, rule extraction and generalization. *Complex Syst.* **1**, 877–922.

Denker, J. S., Gardner, W. R., Graf, H. P., Henderson, D., Howard, R. E., Hubbard, W., Jackel, L. D., Baird, H. S., and Guyon, I. 1989. Neural network recognizer for hand-written zip code digits. In D. Touretzky, ed., *Advances in Neural Information Processing Systems*, pp. 323–331. Morgan Kaufmann, San Mateo, CA.

Fukushima, K. 1980. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol. Cybernet.* **36**, 193–202.

Hubel, D. H., and Wiesel, T. N. 1962. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *J. of Physiol.* **160**, 106–154.

LeCun, Y. 1987. Modèles connexionnistes de l'apprentissage. Ph.D. thesis, Université Pierre et Marie Curie, Paris, France.

LeCun, Y. 1989. Generalization and network design strategies. In *Connectionism in Perspective,* R. Pfeifer, Z. Schreter, F. Fogelman, and L. Steels, eds. North-Holland, Amsterdam.

Mozer, M. C. 1987. Early parallel processing in reading: A connectionist approach. In *Attention and Performance, XII: The Psychology of Reading,* M. Coltheart, ed., Vol. XII, pp. 83–104. Erlbaum, Hillsdale, NY.

Patarnello, S., and Carnevali, P. 1987. Learning networks of neurons with boolean logic. *Europhys. Lett.* **4**(4), 503–508.

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,* D. E. Rumelhart and J. L. McClelland, eds., Vol. I, pp. 318–362. Bradford Books, Cambridge, MA.

Tishby, N., Levin, E., and Solla, S. A. 1989. Consistent inference of probabilities in layered networks: Predictions and generalization. In *Proceedings of the International Joint Conference on Neural Networks,* Washington DC.

Waibel, A. 1989. Consonant recognition by modular construction of large phonemic time-delay neural networks. In *Advances in Neural Information Processing Systems,* D. Touretzky, ed., pp. 215–223. Morgan Kaufmann, San Mateo, CA.

Wang, C. H., and Srihari, S. N. 1988. A framework for object recognition in a visually complex environment and its application to locating address blocks on mail pieces. *Int. J. Computer Vision* **2**, 125.

Received 7 July 1989; accepted 12 September 1989.