Y. Le Cun, L. D. Jackel, B. Boser, J. S. Denker, H. P. Graf, I. Guyon, D. Henderson, R. E. Howard, and W. Hubbard. Handwritten digit recognition: Applications of neural net chips and automatic learning. *IEEE Communication*, pages 41–46, November 1989. invited paper.

Handwritten Digit Recognition: Applications of Neural Network Chips and Automatic Learning

Y. Le Cun
L. D. Jackel
B. Boser
J. S. Denker
H. P Graf
I. Guyon
D. Henderson
R. E. Howard
W. Hubbard

THIS ARTICLE DESCRIBES TWO NEW METHODS for achieving handwritten digit recognition. The task of handwritten digit recognition was chosen for investigation not only because it has considerable practical interest, but because it is relatively well-defined and is sufficiently complex to constitute a meaningful test of connectionist methods.

Simple classification techniques applied to pixel images do not provide high recognition rates because systems based on these techniques contain little prior knowledge about the topology of the task. Knowledge can be built into the system by changing the representation of a digit from a pixel image to a predefined feature description. The first of our methods implements this idea by performing feature extraction with a neural network chip The feature representation can then be used by a relatively simple classifier, consisting of a two-layer network trained with back-propagation.

Finding the proper feature representation for a particular problem is a very complex task. To circumvent this, our second method incorporates sufficient knowledge of the task topology into the classifier so that it automatically generates an appropriate change of representation. This is done by using an adaptive network that has constraints on its architecture that capture the two-dimensional topology of the data. The network is trained on pixel images directly using the back-propagation algorithm

The first section describes the database and shows some examples The following section discusses the first method based on a neural network chip that performs line-thinning and feature extraction using local template matching. The section after that discusses the second method, which is implemented on a digital signal processor, and makes extensive use of constrained automatic learning Some concluding remarks are made in the final section.

The Database

A recurring problem in evaluating character recognition systems is the lack of a reliable measure of task difficulty. In particular, for digit recognition the performance of the system is highly test-set dependent. A system may successfully recognize 99% of test data consisting of well-formed digits but score only 80% when confronted with the poorly-formed digits that are both routinely produced and easily recognized by people. We choose to perform our experiments on a rather difficult data set: isolated handwritten digits that were taken from postal zip codes. The zip code images were collected by the U.S. Postal Service from envelopes that passed through the Buffalo, NY Post Office. A postal service contractor converted the original zip code images to binary images, and segmented the digits; that is, disaggregated them into five disjointed images, one for each digit of the zip code. The resulting database consists of 9,298 binary images of isolated digits, 7,291 of which are used as the training set, while the remaining 2,007 are used as the test set. Most of the images are fairly clean; however, a significant fraction are very blotchy or incomplete. The latter defect is quite common among "5s," in which the top horizontal stroke is often missing. It should be stressed that such mistakes in the segmentation are inevitable. High-level, contextual information is needed to perform a perfectly reliable segmentation on connected character sequences, and no such information is available at the early preprocessing stage without any feedback from the recognition stages.

In our laboratory, we performed some additional preprocessing to normalize the shape and the skew (tilt) of the digits [2] [9]. We scaled the bit maps to fit into either a 32 × 32 or 16 × 16 pixel window. Typical scaled and "de-skewed" examples taken from this data set are shown in Figure 1. As can be seen, many of the digits are of poor quality.

Digit Recognition Using a Neural Network Template Matching Chip

For our first attack on this problem [3], extensive preprocessing was done beyond the simple scaling and deskewing described above. The object here was to change the representation of the digit from a bit map to a feature map with the expectation that the classes will be more tightly clustered in

0163-6804/89/0011-0041 $01 00 © 1989 IEEE

plates exceeds a preset threshold, the center pixel is deleted. Examples of some of the templates used for skeletonization are shown in Figure 2: a) – d) shows typical templates used for thinning in the upper left of each box, e) shows a scaled binarized image, and f) shows the result after skeletonization using one pass with 20 templates. Black is excitation, grey is inhibition, and white is don't care. The pixels deleted by each template are shown in grey on the large image in each box. These templates were chosen in a systematic, but *ad hoc*, manner.

It should be noted that, although skeletonization is performed in a raster scan fashion, it can be performed entirely in parallel with a sufficient amount of special-purpose hardware

Feature Extraction

In the feature extraction process, the skeletonized image is raster scanned with a 7 × 7 pixel window. The templates for feature extraction, which are loaded on the chip after skeletonization is complete, were inspired by results from experimental neurobiology [5], but their precise shapes were fine-tuned by hand [3], in part to conform to the constraints imposed by the chip The templates check for the presence of oriented lines, oriented line end-stops, and arcs. Examples of some of the 49 templates are shown in Figure 3

Whenever a feature template match exceeds the preset threshold, a 1 (for the corresponding feature) is set in the map. Thus, there is a feature map for every template. Some of the templates search for the same feature, but on a different size scale The maps for such features are logically "OR"ed together after the scan is completed.

The feature extraction process generates 49 32 × 32 maps from the 32 × 32 normalized image. Some of these maps are combined by logical operations to produce 18 32 × 32 feature maps To reduce the amount of data, each feature map is coarse-blocked into a 3 × 5 array by simply "OR"ing neighboring bits in each feature map. The 18 3 × 5 feature maps make up a 270-bit vector which is used for digit classification. The coarse blocking also has the effect of building a fair amount of translation and rotation invariance into the processing

Classification and Results of Chip-Based Systems

The back-propagation algorithm was used to train the final classification network. The network had 270 input units, corresponding to the coarse-blocked feature maps; 40 hidden units, fully connected to the input layer; and 10 output units, one for each digit class, fully connected to the hidden units Thus, there were about 11,000 weights for the classification network The raw generalization performance on the 2,007 test examples was around 94% and was obtained after about 15 learning passes through the training set.

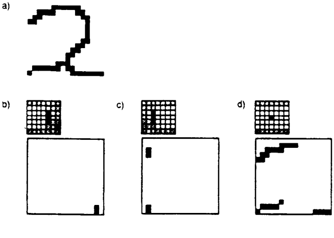

Fig 3 Feature extraction process

In a realistic application, the user is less interested in the raw error rate than in the number of rejections necessary to reach a given level of accuracy. In our case, we measured the percentage of test patterns that must be rejected in order to get 1% error rate on the remaining test patterns.

Our rejection criterion was based on three conditions, the activity level of the most active output unit should be larger than a given threshold, t_1; the activity level of the second most active unit should be smaller than a given threshold, t_2, and finally, the difference between the activity levels of these two units should be larger than a given threshold, t_d.

In a realistic application, the user is less interested in the raw error rate than in the number of rejections necessary to reach a given level of accuracy.

It should be emphasized, however, that the rejection thresholds were obtained using performance measures on the test set. Thus, we measure performance by setting output unit activation criteria, which must be attained in order to accept a classification. For activations below this level we reject the digit as unclassifiable. We find that to obtain a misclassification rate no higher than 1% we must reject 13% of the digits. We expect that a patient human could achieve the same error rate by rejecting about 5% of the digits.

Time Budget

As stated earlier, the chip has enough computing power to evaluate all the templates at one window location in one microsecond Thus, a few milliseconds should be required for the skeletonization and feature extraction across the whole 32 × 32 image. Actual processing times were about two orders of magnitude slower. Although the chip-host interface costs an order of magnitude in throughput, the bottleneck in the process is the speed at which the host computer formats the pixel-window data to be sent to the chip. The bottleneck and the input/output problem could be eliminated if the chip were incorporated into a special-purpose image processing system.

Degrees of Freedom in the Network

We can interpret the combination of skeletonization, feature extraction, and classification as one huge feed-forward network. In that case, about 2 million connections must be evaluated to perform the three skeletonization passes and an additional 2 million are required for the feature extraction Thus, 99 5% of the connection evaluations involve a "hand crafted" change of representation from bit maps to feature maps, the final classification requires only 0 5% of the connections. We note, however, that relatively few bits are required to specify the weights for the skeletonization and feature extraction There are 25 skeletonization kernels of 25 coefficients each Each coefficient has one of three values (≈ 1.6 bits). Most of the kernels are rotations or reflections of other kernels. Furthermore, most of the kernels have symmetry axes. We estimate that once these geometric constraints are imposed, about 150 bits are required to specify the skeletonization kernels, and by similar arguments, about 400 bits are required to specify the feature extraction kernels. Thus, the network of 4 million connections can be parametrized by only several hundred bits

These arguments suggest that by imposing sufficient constraints on the weights in a multilayered network, high recognition accuracy might be achieved by learned convolution kernels Such a network is discussed in the following sections.

Digit Recognition Using Constrained Automatic Learning

The experiments described above suggest that a large network is probably needed to perform accurate digit recognition If we were to design a large unconstrained network, it is very unlikely that it could be trained on a database even as large as ours with any hope of achieving good generalizing ability (i.e , good scores on the test data). In fact, we only expect the network to perform well if we include some of our knowledge about the problem into the network design

Classical work in visual pattern recognition has demonstrated the advantage of extracting local features and combining them to form higher-order features.

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 easily be built into the network by forcing the hidden units to combine only local sources of information. Le Cun [7] has shown how to incorporate prior knowledge as constraints on the network weights during back-propagation learning. We know that the digit recognition task is a two-dimensional geometric problem, so we should incorporate this knowledge into the network design One way this can be accomplished is by forcing the early layers in the network to perform two-dimensional convolutions over the image. Following this approach, we designed the network shown in Figure 4. This method is independent of the one in the previous section.

Architecture

The input to the network is a 16 × 16 gray-scale image that is formed by normalizing the raw image The image is gray-scale rather than binary since a variable number of pixels in the raw image can fall into a given pixel in the normalized image. It should be emphasized that no further processing (such as skeletonization) was performed.

All of the connections in the network are adaptive, although heavily constrained, and are trained using back-propagation. This is in contrast with the chip-based method where the first few layers of connections were hand-chosen constants. The architecture is a direct extension of the one proposed in [7]. In addition to the input and output layer, the network has three hidden layers, labeled H1, H2, and H3, respectively. Connections entering H1 and H2 are local and heavily constrained.

H1 is composed of 12 groups of 64 units arranged as 12 independent 8 × 8 feature maps These twelve feature maps will be designated by H1 1 through H1 12. Each unit in a feature map takes input from a 5 × 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 in the process. A similar two-to-one undersampling occurs going from layer H1 to H2

This design is motivated by the consideration that high resolution may be needed to detect whether a feature of a certain shape appears in an image, while the exact position where that feature appears need not be determined with equally high precision. It is also known that the sort of features that are important at one place in the image are likely to be important in other places.

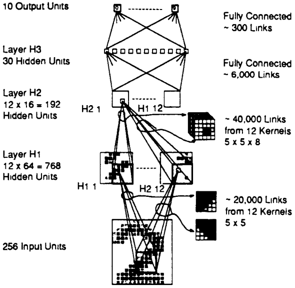

Fig 4 Network architecture

Therefore, corresponding connections on each unit in a given feature map are constrained to have the same weights. In other words, all of the 64 units in H1.1 use the same set of 25 weights. This is taken into account in the appropriate back-propagation formula. 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 generalized convolution[1] with a 5 × 5 kernel.

Of course, units in another map (say H1 4) share another set of 25 weights. It is worth mentioning that units do not share their biases (thresholds). Therefore, each unit 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 state is equal to a constant, predetermined background level, in our case, − 1. Thus, layer H1 comprises 768 units (8 × 8 × 12), 19,968 connections (768 × 26), but only 1,068 free parameters (768 biases + 25 × 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 × 4 plane. As before, these feature maps will be designated H2.1 through 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 eight of the 12 different feature maps in H1. Its receptive field is composed of eight 5 × 5 neighborhoods centered around units that are at identical positions within each of the eight maps Therefore, a unit in H2 has 200 inputs, 200 weights, and a bias. Of course, 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 to the following scheme: there are four maps in the first hidden layer (namely H1.9 to H1.12) that are connected to all maps in the next layer and are expected to compute coarsely-tuned features.

Connections between the remaining eight maps and H2 are as shown in the first eight rows of Table I. The idea behind this

[1] Because of undersampling and non-linear saturating unit functions the total effect cannot be expressed as a convolution in the strict sense, although the spirit is the same.

Table I. Connections between H1 and H2.

	1	2	3	4	5	6	7	8	9	10	11	12
H1.1	X	X	X							X	X	X
H1.2	X	X	X							X	X	X
H1.3	X	X	X	X	X	X						
H1.4	X	X	X	X	X	X						
H1.5				X	X	X	X	X	X			
H1.6				X	X	X	X	X	X			
H1.7							X	X	X	X	X	X
H1.8							X	X	X	X	X	X
H1.9	X	X	X	X	X	X	X	X	X	X	X	X
H1.10	X	X	X	X	X	X	X	X	X	X	X	X
H1.11	X	X	X	X	X	X	X	X	X	X	X	X
H1.12	X	X	X	X	X	X	X	X	X	X	X	X

scheme is to introduce a notion of functional contiguity between the eight maps. Because of this architecture, H2 units in consecutive maps receive similar error signals and are expected to perform similar operations. As in the case of H1, connections falling off the boundaries of H2 maps take their input from a virtual plane whose state is constantly equal to 0. To summarize, layer H2 contains 192 units (12 × 4 × 4) and there is total of 38,592 connections between layers H1 and H2 (192 units × 201 input lines). All these connections are controlled by only 2,592 free parameters (12 feature maps × 200 weights + 192 biases)

Layer H3 has 30 units and is fully connected to H2. The number of connections between H2 and H3 is therefore 5,790 (30 × 192 + 30 biases). The output layer has 10 units and is also fully connected to H3, adding another 310 weights. In summary, the network has 1,256 units, 64,660 connections, and 9,760 independent parameters.

Experimental Setup

All simulations were performed using the BP simulator SN [4] running on a SUN-4/260. 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. The output cost function is the usual Mean Squared Error (MSE):

$$MSE = \frac{1}{OP} \sum_p \sum_o \frac{1}{2} (d_{op} - x_{op})^2 \qquad (1)$$

where P is the number of patterns, O is the number of output units, d_{op} is the desired state for output unit o when pattern p is presented on the input, and x_{op} is the state of output unit o when pattern p is presented.

During each learning experiment, the patterns were presented in a constant order and the training set was presented 28 times. 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). Experiments show that the stochastic gradient converges significantly faster than the true gradient on highly redundant data sets such as ours. We used a variation of the back-propagation algorithm that computes a diagonal approximation to the Hessian matrix to optimally set the learning rate This "pseudo-Newton" procedure is not believed to bring a considerable increase in learning speed, but produces a reliable result without requiring extensive adjustments of the parameters [6]

Results

The MSE on the test set reached a minimum after 23 learning passes through the training set The network was then saved and retrained for 5 passes using a dataset that had undergone a slightly different preprocessing. The total number of training passes was therefore 28. The MSE was then 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). The percentage of rejections would be 12.1% to achieve 1% error on what is left in the test set. To get to this point took three days on a SUN-4 workstation. Figure 5 shows learning curves for the error rate and the log MSE on the test and training set. Due to the highly redundant nature of the data, the best performance on the test set is obtained very quickly.

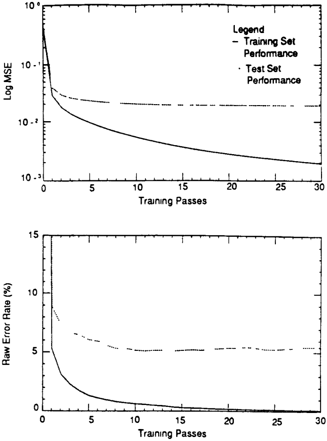

Fig 5 Learning curves

Figure 6 shows all 17 of the images in the test set that were misclassified by the network. In most cases there is a reasonable explanation, or at least an apology for the mistake. The main problem, accounting for six of the cases, is erroneous segmentation. Segmentation is a very difficult problem, especially when the characters overlap extensively. Improving the segmentation would greatly improve the overall performance of the system, but would require considerable effort. In four cases, the image is ambiguous even to humans. This contribution to the error rate obviously cannot be eliminated by any method In three cases, the raw image is unambiguous, but the 16 × 16 image (at the input of the network) is ambiguous because of its low resolution. In two cases, the characters have an unusual style which is not represented in the training set, a modest increase in training set size should reduce the number of such failures. We have no good explanation for the two remaining cases.

Digital Signal Processor Implementation

During the classification process, almost all of the computation time is spent performing multiply/accumulate operations. A Digital Signal Processor (DSP) is therefore a natural choice for implementing the neural network. because of its efficiency in performing multiply/accumulate operations. 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 on 32-bit floating point numbers (25MFLOPS). 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 PC digitizes an image and binarizes it using an adaptive thresholding technique Next. the thresholded image is scanned and each connected component (or segment) is isolated Components that are too small or too large are discarded. Finally the remaining components are sent to the DSP which performs the normalization (including deskewing) and classification steps

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

Conclusions

In past years. much was learned about neural networks by studying small test problems. To make further progress. more can be learned by attacking large. real-world tasks. especially if the tasks have been attempted in the past with other methods. In particular. real applications contain the surprises and secrets of the natural world. Just as in other types of research and engineering. there are aspects of the system that would not appear in a idealized model. and must be discovered through experiments on real systems. We believe that our experiments on digit recognition. described in this article. uncover aspects of real data that cannot be inferred from small problems.

We have successfully applied neural network methods to a large. real-world task. Our results appear to be the state of the art in digit recognition. We demonstrated that a general-purpose neural network chip can be incorporated as an accelerator in a large network. We found that real problems with regularity scale well.

We also showed that a network can be trained on a low-level representation of data that has minimal preprocessing (as opposed to elaborate feature extraction).

Perhaps the most important lesson is the importance of constrained adaptation We used a complex network (capable of handling a complex problem) where many connections were specified by relatively few free parameters (which could therefore be determined from relatively small amounts of training data) 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 back-propagation on smaller. artificial problems.

The final network of connections and weights obtained by back-propagation 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

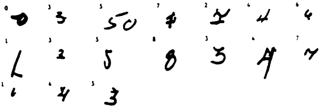

Fig 6 Misclassified images

Acknowledgments

We would like to thank the U S. Postal Service and its contractors for providing us with the database

The neural network simulator SN is the result of a collaboration between Leon-Yves Bottou and Yann Le Cun.

References

[1] L Y Bottou and Y Le Cun. "SN A Simulator for Connectionist Models." *Proc of NeuroNimes '88*. Nimes. France. 1988

[2] R G Casey. "Moment Normalization of Handprinted Characters " *IBM J of Res and Dev* . vol. 14 pp 548–557. Sept 1970

[3] J S Denker *et al* . "Neural Network Recognizer for Handwritten Zip Code Digits." *Advances in Neural Information Processing Systems*. D Touretzky. ed. pp 323–331. Morgan Kaufmann. 1989

[4] H P Graf W. Hubbard. L D Jackel. and P G. de Vegvar. "A CMOS Associative Memory Chip." *Proc IEEE 1st Int'l Conf on Neural Networks* vol. III, pp 461–468 San Diego. CA 1987

[5] D H Hubel and T N. Wiesel. "Receptive Fields. Binocular Interaction and Functional Architecture in the Cat's Visual Cortex." *J. of Physiology*. vol 160. pp. 106–154. 1962

[6] Y Le Cun. "Modeles Connexionnistes de l'Apprentissage." Ph D thesis. Universite Pierre et Marie Curie. Paris. France. 1987

[7] Y Le Cun "Generalization and Network Design Strategies." *Connectionism in Perspective*. R Pfeifer. Z Schreter. F Fogelman. and L Steels. eds North Holland. Zürich Switzerland pp 143–155. 1989

[8] N J Naccache and R Shingal. "SPTA: A Proposed Algorithm for Thinning Binary Patterns." *IEEE Trans Systems. Man. and Cybernetics* vol SMC-14 pp. 409–418. 1964

[9] W C. Naylor. "Some Studies in the Interactive Design of Character Recognition Systems." *IEEE Transaction on Computers*. vol 20. pp 1,075–1,088. Sept 1971

[10] D E. Rumelhart. G E Hinton. and R J Williams. "Learning Internal Representations By Error Propagation " *Parallel Distributed Processing Explorations in the Microstructure of Cognition*. vol I. pp 318–362 Bradford Books. Cambridge. MA. 1986

Biography

The Holmdel Neural Network Group of AT&T Bell Laboratories was founded in 1984 by Richard E Howard. Head of the Microelectronics Research Department and Lawrence D Jackel. Head of the Device Structure Research Department Both Mr Howard and Mr Jackel have Ph D s in physics and recent experience in superconductivity. microscience. and microfabrication Current members of the group are. **Bernhard E. Boser**. received an electrical engineering diploma from E T H in Zürich and an electrical engineering Ph D from Stanford University. **John S. Denker**: received a B.S degree from the California Institute of Technology and a Ph.D in physics from Cornell University. **Hens-Peter Graf**: received a Ph D. in physics from E T H in Zürich; **Wayne Hubbard** received an electrical engineering degree. **Don Henderson** received a B S degree in Computer Science from Monmouth College. **Lawrence D. Jackel** received a B A. degree in physics from Brandeis University and a Ph D. from Cornell University; **Yann Le Cun**: received an electrical engineering diploma from E.S.I.E E in Paris and a computer science Ph.D. from the University of Paris VI; **Isabelle Guyon**. received an engineering diploma from E S P C.I. in Paris and a physical sciences Ph D from the University of Paris VI. and **Richard E Howard**. received a B S.c degree in physics from the California Institute of Technology and a Ph D. in applied physics from Stanford University