COMPARISON OF LEARNING ALGORITHMS FOR HANDWRITTEN DIGIT RECOGNITION

Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes,
J. Denker, H. Drucker, I. Guyon, U. Müller,
E. Säckinger, P. Simard, and V. Vapnik
Bell Laboratories, Holmdel, NJ 07733, USA
Email: yann@research.att.com

Abstract

This paper compares the performance of several classifier algorithms on a standard database of handwritten digits. We consider not only raw accuracy, but also rejection, training time, recognition time, and memory requirements.

COMPARISON OF LEARNING ALGORITHMS FOR HANDWRITTEN DIGIT RECOGNITION

Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Müller, E. Säckinger, P. Simard, and V. Vapnik
Bell Laboratories, Holmdel, NJ 07733, USA
Email: yann@research.att.com

1 Introduction

The simultaneous availability of inexpensive powerful computers, powerful learning algorithms, and large databases, have caused rapid progress in handwriting recognition in the last few years. This paper compares the relative merits of several classification algorithms developed at Bell Laboratories and elsewhere for the purpose of recognizing handwritten digits. While recognizing individual digits is only one of many problems involved in designing a practical recognition system, it is an excellent benchmark for comparing shape recognition methods. Though many existing method combine a handcrafted feature extractor and a trainable classifier, this study concentrates on adaptive methods that operate directly on size-normalized images.

2 Database

The database used to train and test the systems described in this paper was constructed from the NIST's Special Database 3 and Special Database 1 containing binary images of handwritten digits. Our training set was composed of 30,000 patterns from SD-3, and 30,000 patterns from SD-1. Our test set was composed of 5,000 patterns from SD-3 and 5,000 patterns from SD-1. The 60,000 pattern training set contained examples from approximately 250 writers. We made sure that the sets of writers of the training set and test set were disjoint. All the images were size normalized to fit in a 20x20 pixel box (while preserving the aspect ratio). For some experiments, the 20x20 images were deslanted using moments of inertia before being presented. For other experiments they were only centered in a larger input field using center of mass. Grayscale pixel values were used to reduce the effects of aliasing. Two methods (LeNet 1 and Tangent Distance) used subsampled versions of the images to 16 by 16 pixels.

3 The Classifiers

In this section we briefly describe the classifiers used in our study. For more complete descriptions readers may consult the references.

Baseline Linear Classifier: Possibly the simplest classifier that one might consider is a linear classifier. Each input pixel value contributes to a weighted sum for each output unit. The output unit with the highest sum (including the contribution of a bias constant) indicates the class of the input character. For this experiment, we used deslanted 20x20 images. The network has 4010 free parameters. The deficiencies of the linear classifier are well documented (Duda

& Hart 73) and it is included here simply to form a basis of comparison for more sophisticated classifiers. The test error rate is 8.4%. Various combinations of sigmoid units, linear units, gradient descent learning, and learning by directly solving linear systems gave similar results.

Baseline Nearest Neighbor Classifier: Another simple classifier is a K-nearest neighbor classifier with a Euclidean distance measure between input images. This classifier has the advantage that no training time, and no brain on the part of the designer, are required. However, the memory requirement and recognition time are large: the complete 60,000 twenty by twenty pixel training images (about 24 Megabytes at one byte per pixel, or 12 megabytes at 4 bits/pixel) must be available at run time. Much more compact representations could be devised with modest increase in recognition time and error rate. As in the previous case, deslanted 20x20 images were used. The test error for $k = 3$ is 2.4%. Naturally, a realistic Euclidean distance nearest-neighbor system would operate on feature vectors rather than directly on the pixels, but since all of the other systems presented in this paper operate directly on the pixels, this result is useful for a baseline comparison.

Pairwise Linear Classifier: A simple improvement of the basic linear classifier was tested (Guyon et al. 89). The idea is to train each unit of a single-layer network to classify one class from one other class. In our case this layer comprises 45 units labelled 0/1, 0/2,...0/9, 1/2....8/9. Unit i/j is trained to produce +1 on patterns of class i, -1 on patterns of class j, and is not trained on other patterns. The final score for class i is the sum of the outputs all the units labelled i/x minus the sum of the output of all the units labelled y/i, for all x and y. Error rate on the test set was 7.6%, only slightly better than a linear classifier.

Principal Component Analysis and Polynomial Classifier: Following (Schürmann 78), a preprocessing stage was constructed which computes the projection of the input pattern on the 40 principal components of the set of training vectors. To compute the principal components, the mean of each input component was first computed and subtracted from the training vectors. The covariance matrix of the resulting vectors was then computed, and diagonalized using Singular Value Decomposition. The 40-dimensional feature vector was used as the input of a second degree polynomial classifier. This classifier can be seen as a linear classifier with 821 inputs, preceded by a module that computes all products of pairs of input variables. Error on the test set was 3.3%.

Radial Basis Function Network: Following (Lee 91), an RBF network was constructed. The first layer was composed of 1,000 Gaussian RBF units with 400 inputs (20x20), the second layer was a simple 1000-10 linear classifier. The RBF units were divided into 10 groups of 100. Each group of units was trained on all the training examples of one of the 10 classes using the adaptive K-means algorithm. The second layer weights were computed using a regularized pseudo-inverse method. Error rate on the test set was 3.6%

Large Fully Connected Multi-Layer Neural Network: Another classifier that we tested was a fully connected multi-layer neural network with two layers of weights (one hidden layer). The network trained with various numbers of hidden units. Deslanted 20x20 images were used as input. The best result was 1.6% on the test set, obtained with a 400-300-10 network (approximately 123,300 weights). It remains somewhat of a mystery that networks with such a large number of free parameters manage to achieve reasonably low testing errors. We conjecture that the dynamics of gradient descent learning in multi-

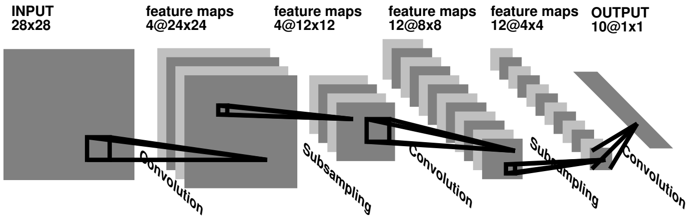

Figure 1: Architecture of LeNet 1. Each plane represents a feaure map, i.e. a set of units whose weights are constrained to be identical. Input images are sized to fit in a 16 x 16 pixel field, but enough blank pixels are added around the border of this field to avoid edge effects in the convolution calculations.

layer nets has a "self-regularization" effect. Because the origin of weight space is a saddle point that is attractive in almost every direction, the weights invariably shrink during the first few epochs (recent theoretical analysis seem to confirm this (Sara Solla, personal communication)). Small weights cause the sigmoids to operate in the quasi-linear region, making the network essentially equivalent to a low-capacity, single-layer network. As the learning proceeds, the weights grow, which progressively increases the effective capacity of the network. A better theoretical understanding of these phenomena, and more empirical evidence, are definitely needed.

LeNet 1: To solve the dilemma between small networks that cannot learn the training set, and large networks that seem overparameterized, one can design *specialized* network architectures that are specifically designed to recognize two-dimensional shapes such as digits, while eliminating irrelevant distortions and variability. These considerations lead us to the idea of *convolutional network* (LeCun et al. 90). In a convolutional net, each unit takes its input from a local "receptive field" on the layer below, forcing it to extract a local feature. Furthermore, units located at different places on the image are grouped in planes, called *feature maps*, within which units are constrained to share a single set of weights. This makes the operation performed by a feature map shift invariant, and equivalent to a convolution, followed by squashing functions. This *weight-sharing* technique greatly reduces the number of free parameters. A single layer is formed of multiple feature maps, extracting different features types.

Complete networks are formed of multiple convolutional layers, extracting features of increasing complexity and abstraction. Sensitivity to shifts and distortions can be reduced by using lower-resolution feature maps in the higher layers. This is achieved by inserting subsampling layers between the convolution layers. It is important to stress that *all* the weights in such a network are trained by gradient descent. Computing the gradient can be done with a slightly modified version of the classical backpropagation procedure. The training process causes convolutional networks to automatically synthesize their own features. One of our first convolutional network architecture, LeNet 1, shown in Figure 3, was trained on the database. Because of LeNet 1's small input field, the images were down-sampled to 16x16 pixels and centered in the 28x28 input layer. Although about 100,000 multiply/add steps are required to evaluate LeNet 1, its convolutional nature keeps the number of free parameters to only about 3000. The LeNet 1 architecture was developed using our own version of the USPS database and its size was tuned to match the available data. LeNet 1 achieved 1.7% test error.

LeNet 4: Experiments with LeNet 1 made it clear that a larger convolutional network was needed to make optimal use of the large size of the training set. LeNet 4 was designed to address this problem. It is an expanded version of LeNet 1 that has a 32x32 input layer in which the 20x20 images (not deslanted) were centered by center of mass. It includes more feature maps and an additional layer of hidden units that is fully connected to both the last layer of features maps and to the output units. LeNet 4 contains about 260,000 connections and has about 17,000 free parameters. Test error was 1.1%. In previous experiments with ZIP code data, replacing the last layer of LeNet with a more complex classifier improved the error rate. We replaced the last layer of LeNet4 with a Euclidean Nearest Neighbor classifier, and with the "local learning" method of Bottou and Vapnik, in which a local linear classifier is retrained each time a new test pattern is shown. Neither of those improve the raw error rate, although they did improve the rejection.

LeNet 5: LeNet 5, has an architecture similar to LeNet 4, but has more feature maps, a larger fully-connected layer, and it uses a distributed representation to encode the categories at the output layer, rather than the more traditional "1 of N" code. LeNet 5 has a total of about 340,000 connections, and 60,000 free parameters, most of them in the last two layers. Again the non-deslanted 20x20 images centered by center of mass were used, but the training procedure included a module that distorts the input images during training using small randomly picked affine transformations (shift, scaling, rotation, and skewing). It achieved 0.9% error.

Boosted LeNet 4: Following theoretical work by R. Schapire, Drucker et al. (Drucker et al 93) developed the "boosting" method for combining multiple classifiers. Three LeNet 4 are combined: the first one is trained the usual way. the second one is trained on patterns that are filtered by the first net so that the second machine sees a mix of patterns, 50% of which the first net got right, and 50% of which it got wrong. Finally, the third net is trained on new patterns on which the first and the second nets disagree. During testing, the outputs of the three nets are simply added. Because the error rate of LeNet 4 is very low, it was necessary to artificially increase the number of training samples with random distortions (like with LeNet 5) in order to get enough samples to train the second and third nets. The test error rate was 0.7%, the best of any of our classifiers. At first glance, boosting appears to be three times more expensive as a single net. In fact, when the first net produces a high confidence answer, the other nets are not called. The cost is about 1.75 times that of a single net.

Tangent Distance Classifier (TDC): The Tangent Distance classifier (TDC) is a nearest-neighbor method where the distance function is made insensitive to small distortions and translations of the input image (Simard et al. 93). If we consider an image as a point in a high dimensional pixel space (where the dimensionality equals the number of pixels), then an evolving distortion of a character traces out a curve in pixel space. Taken together, all these distortions define a low-dimensional manifold in pixel space. For small distortions, in the vicinity of the original image, this manifold can be approximated by a plane, known as the tangent plane. An excellent measure of "closeness" for character images is the distance between their tangent planes, where the set of distortions used to generate the planes includes translations, scaling, skewing,

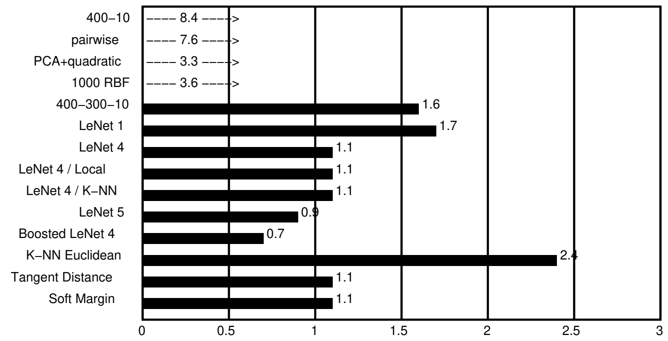

Figure 2: error rate on the test set (%). The uncertainty in the quoted error rates is about 0.1%.

squeezing, rotation, and line thickness variations. A test error rate of 1.1% was achieved using 16x16 pixel images. Prefiltering techniques using simple Euclidean distance at multiple resolutions allowed to reduce the number of necessary Tangent Distance calculations. The figure for storage requirement assumes that the patterns are represented at multiple resolutions at one byte per pixel.

Optimal Margin Classifier (OMC): Polynomial classifiers are well-studied methods for generating complex decision surfaces. Unfortunately, they are impractical for high-dimensional problems, because the number of product terms is prohibitive. A particularly interesting subset of decision surfaces is the ones that correspond to hyperplanes that are at a maximum distance from the convex hulls of the two classes in the high-dimensional space of the product terms. Boser, Guyon, and Vapnik (Boser et al. 92) realized that any polynomial of degree k in this "maximum margin" set can be computed by first computing the dot product of the input image with a subset of the training samples (called the "support vectors"), elevating the result to the k-th power, and linearly combining the numbers thereby obtained. Finding the support vectors and the coefficients amounts to solving a high-dimensional quadratic minimization problem with linear inequality constraints. Using a version of the procedure, known as Soft Margin Classifier (Cortes & Vapnik 95) that is well suited for noisy problems, with a 4-th degree decision surface, a test error of 1.1% was reached. The number of support vectors obtained was around 25,000.

4 Discussion

A summary of the performance of our classifiers is shown in Figures 2 to 5. Figure 2 shows the raw error rate of the classifiers on the 10,000 example test set. Boosted LeNet 4 is clearly the best, achieving a score of 0.7%, closely followed by LeNet 5 at 0.9%. This can be compared to our estimate of human performance, 0.2%.

Figure 3 shows the number of patterns in the test set that must be rejected to attain a 0.5% error. In many applications, rejection performance is more significant than raw error rate. Again, Boosted LeNet 4 has the best score. The enhanced versions LeNet 4 did better than the original LeNet 4, even though

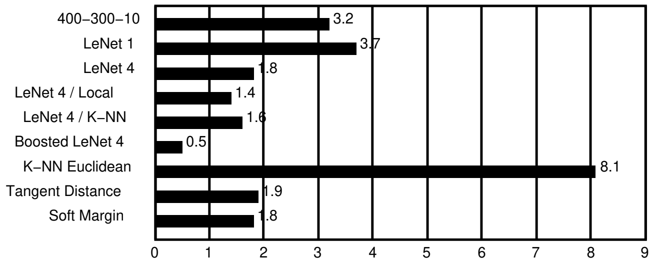

Figure 3: Percent of test patterns rejected to achieve 0.5% error on the remaining test examples for some of the systems.

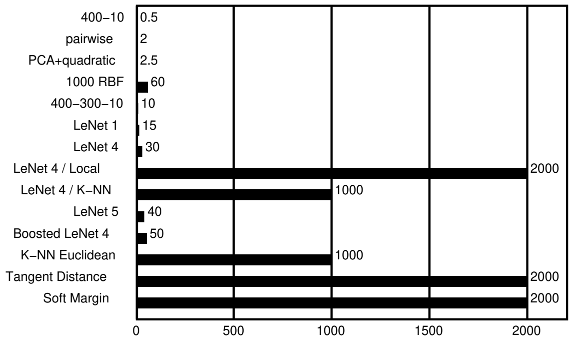

Figure 4: Time required on a Sparc 10 for recognition of a single character starting with a size-normalized image (in milliseconds).

the raw accuracies were identical.

Figure 4 shows the time required on a Sparc 10 for each method to recognize a test pattern, starting with a size-normalized image. Expectedly, memory-based method are much slower than neural networks. Single-board hardware designed with LeNet in mind performs recognition at 1000 characters/sec (Säckinger & Graf 94). Cost-effective hardware implementations of memory-based techniques are more elusive, due to their enormous memory requirements.

Training time was also measured. K-nearest neighbors and TDC have essentially zero training time. While the single-layer net, the pairwise net, and PCA+quadratic net could be trained in less than an hour, the multilayer net training times were expectedly much longer: 3 days for LeNet 1, 7 days for the fully connected net, 2 weeks for LeNet 4 and 5, and about a month for boosted LeNet 4. Training the Soft Margin classifier took about 10 days. However, while the training time is marginally relevant to the designer, it is totally irrelevant to the customer.

Figure 5 shows the memory requirements of our various classifiers. Figures are based on 4 bit per pixel representations of the prototypes for K-Nearest Neighbors, 1 byte per pixel for Soft Margin, and Tangent Distance. They should be taken as upper bounds, as clever compression of the data and/or elimination of redundant training examples can reduce the memory requirements of some

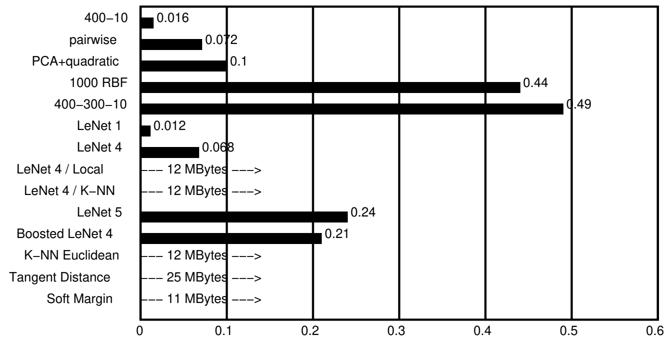

Figure 5: Memory requirements for classification of test patterns (in MBytes). Numbers are based on 4 bit/pixel for K-NN, 1 byte per pixel for Soft Margin, and Tangent Distance, 4 byte per pixel for the rest.

of the methods. Memory requirements for the neural networks assume 4 bytes per weight (and 4 bytes per prototype component for the LeNet 4 / memory-based hybrids), but experiments show that one-byte weights can be used with no significant change in error rate. Of the high-accuracy classifiers, LeNet 4 requires the least memory.

5 Conclusions

This paper is a snapshot of ongoing work. Although we expect continued changes in all aspects of recognition technology, there are some conclusions that are likely to remain valid for some time.

Performance depends on many factors including high accuracy, low run time, and low memory requirements. As computer technology improves, larger-capacity recognizers become feasible. Larger recognizers in turn require larger training sets. LeNet 1 was appropriate to the available technology five years ago, just as LeNet 5 is appropriate now. Five years ago a recognizer as complex as LeNet 5 would have required several months' training, and more data than was available, and was therefore not even considered.

For quite a long time, LeNet 1 was considered the state of the art. The local learning classifier, the optimal margin classifier, and the tangent distance classifier were developed to improve upon LeNet 1 – and they succeeded at that. However, they in turn motivated a search for improved neural network architectures. This search was guided in part by estimates of the capacity of various learning machines, derived from measurements of the training and test error as a function of the number of training examples. We discovered that more capacity was needed. Through a series of experiments in architecture, combined with an analysis of the characteristics of recognition errors, LeNet 4 and LeNet 5 were crafted.

We find that boosting gives a substantial improvement in accuracy, with a relatively modest penalty in memory and computing expense. Also, distortion models can be used to increase the effective size of a data set without actually taking more data.

The optimal margin classifier has excellent accuracy, which is most remarkable, because unlike the other high performance classifiers, it does not include *a priori* knowledge about the problem. In fact, this classifier would do just as well if the image pixels were permuted with a fixed mapping. It is still much slower and memory hungry than the convolutional nets. However, improvements are expected as the technique is relatively new.

Convolutional networks are particularly well suited for recognizing or rejecting shapes with widely varying size, position, and orientation, such as the ones typically produced by heuristic segmenters in real-world string recognition systems (see article by L. Jackel in these proceedings).

When plenty of data is available, many methods can attain respectable accuracy. Although the neural-net methods require considerable training time, trained networks run much faster and require much less space than memory-based techniques. The neural nets' advantage will become more striking as training databases continue to increase in size.

References

B. E. Boser, I. Guyon, and V. N. Vapnik, *A Training Algorithm for Optimal Margin Classifiers,* in Proceedings of the Fifth Annual Workshop on Computational Learning Theory **5** 144-152, Pittsburgh (1992).

L. Bottou and V. Vapnik, *Local Learning Algorithms,* Neural Computation **4**, 888-900 (1992).

C. Cortes and V. Vapnik, *The Soft Margin Classifier,* Machine Learning, to appear (1995).

R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, Chapter 4, John Wiley and Sons (1973).

H. Drucker, R. Schapire, and P. Simard, *Boosting Performance in Neural Networks,* International Journal of Pattern Recognition and Artificial Intelligence **7** 705-720 (1993).

I. Guyon, I. Poujaud, L. Personnaz, G. Dreyfus, J. Denker, and Y. LeCun, *Comparing Different Neural Net Architectures for Classifying Handwritten Digits,* in Proc. 1989 IJCNN II 127-132, Washington DC. IEEE, (1989).

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, R. Hubbard, and L. D. Jackel, *Handwritten digit recognition with a back-propagation network*, in D. Touretzky (ed), Advances in Neural Information Processing Systems **2**, Morgan Kaufman, (1990).

Yuchum Lee, *Handwritten Digit Recognition using K-Nearest Neighbor, Radial-Basis Functions, and Backpropagation Neural Networks*, Neural Computation, **3**, 3, (1991).

E. Säckinger and H.-P. Graf, *A System for High-Speed Pattern Recognition and Image Analysis*, Proc of the fourth International Conference on Microelectronics for Neural Networks and Fuzzy Systems, IEEE (1994).

J. Schürmann, *A Multi=Font Word Recognition System for Postal Address Reading*, IEEE Trans., G27, **3** (1978).

P. Simard, Y. LeCun, and J Denker, *Efficient Pattern Recognition Using a New Transformation Distance,* Neural Information Processing Systems **5**, 50-58, Morgan Kaufmann (1993).