Communicated by Gerald Tesauro

Boosting and Other Ensemble Methods

Harris Drucker*
Corinna Cortes
L. D. Jackel
Yann LeCun
Vladimir Vapnik
AT&T Bell Laboratories, Holmdel, NJ 07733 USA

We compare the performance of three types of neural network-based ensemble techniques to that of a single neural network. The ensemble algorithms are two versions of boosting and committees of neural networks trained independently. For each of the four algorithms, we experimentally determine the test and training error curves in an optical character recognition (OCR) problem as both a function of training set size and computational cost using three architectures. We show that a single machine is best for small training set size while for large training set size some version of boosting is best. However, for a given computational cost, boosting is always best. Furthermore, we show a surprising result for the original boosting algorithm: namely, that as the training set size increases, the training error decreases until it asymptotes to the test error rate. This has potential implications in the search for better training algorithms.

Introduction

There is interest in using a committee of learning machines to improve performance over that of a single learning machine. Perrone and Cooper (1992) did a study of what they termed ensemble methods (and others call committee machines), and give extensive references to other publications. In addition, there has been work done on boosting by Drucker *et al.* (1993a,b), and other work on committees by Srihani (1992), Suen *et al.* (1992), Benediksson and Swain (1992), and Hansen and Salamon (1990). In many of these cases, the committee members are neural networks trained on the same data, but initialized with different weights. The expectation is that the different networks converge to different local minima and by somehow combining the outputs, performance can be improved. Boosting is different in that each learning machine in an en-

*Corresponding address: Monmouth College, West Long Branch, NJ 07764 USA.

semble is trained sequentially on patterns that have been filtered by the previously trained members of the ensemble. Most ensemble methods have a training set with identical statistics for each member of the ensemble while each member in the ensemble of a boosting machine has a training set whose statistics are different because they have been altered by filtering them through the previously trained machines.

After training an ensemble, one usually has to decide how to weight the outputs. One may use a straight voting scheme, which leads to problems in other than the two-class problem. (How does one make a decision in a committee of three if each member is voting for a different class?) Alternatively, one may use a weighted linear addition of the outputs of each machine. However, this question usually comes up after the networks in the ensemble have been trained and is not explicitly part of the training process. This is not true in boosting. Moreover, in recent work on "mixtures of experts" done by Jordan and Jacobs (1992) based on the expectation–maximization algorithm, both the networks and what they call a gating network are trained. The basic idea is to build up a committee of experts, each of which is good on a subset of the problem and then gate the experts together. However, these techniques have not yet been applied to OCR problems.

In assessing performance on learning machines, one usually has a training and test set. The test set performance is also termed the generalization ability and a plot of test error rate versus training set size is usually called the learning curve. For a finite capacity network (Vapnik 1982), one expects the difference between the test and training error rate to approach zero as training set size is increased.

We have to carefully distinguish between the size of the training set and the actual number of patterns trained on. Assume we have a training set of size N, which may consist of what we may call (but hard to define) typical patterns, hard patterns, easy patterns, mislabeled patterns, and outliers (not all mutually exclusive). By judiciously picking a subset of size N_c to actually train on, we may get better generalization performance than if we had trained on all N patterns. Of course one has to have some procedure to pick this subset but if we have some "smart" procedure we obtain two advantages: generalization is better and time to train is less. Of course, there is some computational cost in examining patterns and then discarding them, but that cost is typically much less than using them in training, which requires a forward propagation, backward propagation, and weight updates for each pattern for each epoch. There are a number of procedures that either discard data or pick data intelligently including the references above concerning boosting and other work done by Sueng *et al.* (1992), Matic *et al.* (1992), Freund (1990), Baum and Lang (1991), and Atlas *et al.* (1990). To distinguish between N and N_c we shall call the former the training set size and the latter the computational cost. When we do not explicitly train on all N patterns, the computational cost is a fair method to compare algorithms.

The interest here is in comparing ensemble learning algorithms on an OCR problem as a function of the training set size and the computational cost. We therefore experimentally answer the following two questions for the algorithms investigated: given a training set size, which algorithm is best? Also, for a given compute time, which algorithm is best? We have also determined that the training error curve (plotted against training set size) for the boosting algorithm has a surprising characteristic that has implications in the search for better algorithms.

1 Theory of Boosting

Boosting is an algorithm invented by Robert Schapire (1990) that, under certain conditions, allows one to improve the performance of any learning machine and was first designed in the context of the so-called *probably approximately correct (PAC)* learning model. In the standard PAC model (sometimes called the *strong* learning model), the learner must be able to produce a hypothesis with error at most ε, for arbitrarily small positive values of ε. Since the learner is receiving random examples, there is always the chance that the learner will receive a highly unrepresentative sample that prevents it from learning anything meaningful about the target concept. We therefore ask only that the learner succeed in finding a good approximation to the target concept with probability at least $1-\delta$, where δ is an arbitrarily small positive number. In a variation of the PAC model called the *weak learning model*, this requirement is relaxed dramatically. In this model, which was introduced by Kearns and Valiant (1989), the learner is required only to produce hypotheses with error rates slightly less than $1/2$. Since the hypothesis that guesses entirely at random on every example will achieve an error rate of exactly $1/2$, the weak learning model is essentially asking that the learner be able to produce hypotheses that perform only slightly better than random guessing.

The main result of Schapire's paper was a proof that the strong and weak learning models are actually equivalent. That is, Schapire gave a provably correct technique for converting any learning algorithm that performs only slightly better than random guessing into one that produces hypotheses with arbitrarily small error rates. The technique is described below (under the Procedure section) for creating a composite or ensemble hypothesis from three subhypotheses that were trained on three different distributions. Schapire proves that if these three weak subhypotheses have an error rate of $\alpha < 1/2$ (with respect to the distribution on which they were trained), then the resulting ensemble hypothesis at each successive iteration will have an error rate of $3\alpha^2-2\alpha^3$, which is significantly less than α. Applying this technique recursively, Schapire was able to also show how the error rate can be made arbitrarily small.

2 Procedure

A database of 120,000 handwritten and segmented digits from the National Institute of Standards and Technology (NIST) was used for evaluation: one set of 2000 was used for testing and randomly chosen subsets of the remaining 118,000 used for training. The pixel arrays were subsampled to give a 10 by 10 size input space so the algorithms could run in a reasonable time. For each of the four algorithms, each of the three architectures, and a particular training set size, the training and test performance were evaluated 10 times (with different initialization of weights and different sets of training patterns) to obtain a mean for that training set size and computational cost. The average test and training performance were then plotted as a function of the training set size and computational cost. The objective here was not to get the best performance by choosing good architectures but to compare the algorithms. The choices of architecture and input space size were also dictated by the size of the database. We wanted to be sure that, for some training set size less than 118,000, error rates on both the test and training set would be approximately equal indicating that the best performance had been reached within the limits of the capacity of the network (Vapnik 1982).

Training of a single machine is as follows: N_1 patterns are chosen randomly and the weights randomized to small values. The network is trained using backpropagation (with stochastic gradient descent) until it reaches a minimum of the mean square error. The minimum is defined to be that epoch where the relative difference between the mean square error of the present epoch and the previous epoch is less than 0.25%. The test performance is then evaluated. This is repeated 10 times for the same size training set (but randomly chosen patterns and initial weights). We then iterate for different training set sizes until the test and training error rate are approximately equal. Here $N = N_1 = N_c$.

For what we term a parallel machine, we randomize three machines and train each with a different training set, each of size N_1. Each is trained separately. In previous work (Drucker *et al.* 1993a), we determined that addition of the respective outputs (digit 0 outputs of three machines added, etc.) gives better performance than simple voting (verified here also). Thus, to evaluate the training performance we present the entire $N = 3N_1$ patterns to each of the three machines and add the respective 10 outputs to get one set of 10 outputs. The maximum output corresponds to the chosen label. Both the training set size and computational cost is $3N_1$. We also tried a committee with five members.

In the original form of the boosting algorithm, a first machine is trained with N_1 patterns. We then use the first machine to filter another set of training patterns in the following manner: Flip a fair coin. If heads, pass *new* patterns through the first machine and discard correctly classified patterns until the first machine misclassifies a pattern. That misclassified pattern is added to the training set for the second ma-

chine. If tails, pass new patterns through the first machine and discard incorrectly classified patterns until the first machine correctly classifies a pattern and add this pattern to the training set. Iterate this procedure until there are a total N_1 examples that become the training set for the second machine in the ensemble. The coin flipping ensures that the second set of N_1 training examples is such, that if passed through the first machine, would have a 50% error rate. Thus these new N_1 patterns have a different distribution than the set used to train the first machine. This is critically important in the proof of Schapire's algorithm because now the second machine must learn a different distribution and should be contrasted to a parallel machine where each network learns the same distribution. Now define N_2 as the number of examples that must be filtered to obtain N_1 patterns to train the second machine. Note that N_1 is fixed and N_2 depends on the generalization error rate of the first machine.

Once the second machine is trained in the usual manner, a third training set is formed as follows: Pass a new pattern through the first two machines. If the two machines agree, discard the data. If the two machines disagree, that pattern becomes a part of the training set for the third machine. Continue until there are a total of N_1 patterns. Say there are N_3 patterns that have to be examined to find the N_1 patterns that form the training set for the third machine. The total size of the training set is $N = N_1 + N_2 + N_3$ because these many patterns have to be examined (although not all are part of the computational cost) where the computational cost $N_c = 3N_1$ because these are the actual patterns trained on. The algorithm is therefore "smart" in the sense that there is a large pool of training examples but test performance is better if one selectively uses a subset (of size $3N_1$) for the actual training. In the original theoretical derivation of the algorithm, evaluation of the test performance was as follows: present a test pattern to the three machines. If the first two machines agree, use that label, otherwise use the labeling assigned by the third machine. As stated before, we have experimentally shown that adding of the three sets of outputs is much better than voting and that is the approach we take.

The training performance for the boosting machine (which is called the original boosting algorithm here) is determined by presenting the N patterns to the three machines and adding the respective outputs. The test performance is evaluated by presenting the 2000 test patterns to the ensemble. In obtaining learning curves for this algorithm we increment N_1 but plot training and test error rate versus both the training set size N and computational cost ($3N_1$). Because the boosting machine uses so many patterns it was not possible to construct an ensemble with more than three members within the constraint of the size of the NIST database.

The original boosting algorithm always uses new examples for each machine. If the error rate on the first machine is low then many patterns may have to be examined until N_1 patterns are obtained to train the

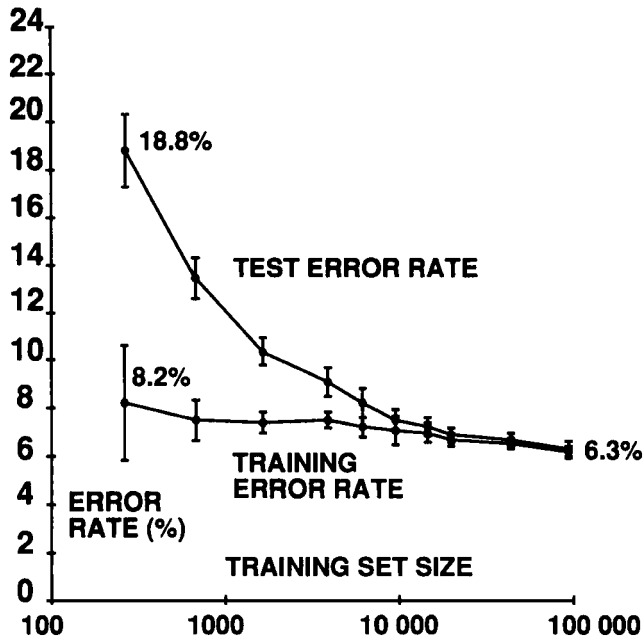

Figure 1: Training and test error rate for a boosting network using three 100-10-10 networks.

second machine. If a large training set is not available, then one might consider a modified algorithm. The modified boosting algorithm starts out with a first training set of size N_4. After training the first machine, we *reuse* the N_4 patterns and pass all these old patterns through the first machine, using a subset of them such that the new training set consists of patterns, 50% of which are incorrectly classified by the first machine and 50% classified correctly. Call this training set size N_5. Note that here N_4 is fixed and N_5 will depend on the training error of the first machine. If the error rate of the first machine is very low, then there may be very few examples for the second machine to train on. After the second machine is trained, the N_4 patterns are reused by presenting them to both trained machines. If the two machines disagree, add the pattern to the training set, otherwise discard. Call the number of patterns that the two machines disagree on N_6. The total training set size is fixed at N_4 but the computational cost is $N_4 + N_5 + N_6$.

We first (Fig. 1) show the training and test performance for a boosted network using the original boosting algorithm (each constituent network is 100-10-10). Error bars show plus and minus one standard deviation.

The negative slope for the boosted training error rate is somewhat surprising because a single 100-10-10 network has a training error rate that monotonically increases until it asymptotes to the test error rate. The explanation is as follows: recall that the training set of size N consists of a subset of size $3N_1$ actually used to train on; the remainder are examined but filtered out. As an example, suppose the training set for the first network (N_1) is of size 100. The training performance is very good (approximately 2%), but the generalization is very poor (23%). This means that the trained first network has to sort through only approximately 217 (50/0.23) examples never seen before to find 50 patterns in error and approximately 65 (50/0.77) examples to find 50 patterns classified correctly obtaining an $N_2 = 217 + 65 = 282$ examples. The 182 examples examined but filtered out have an approximate zero error rate. Furthermore, since the second network will have poor generalization based on the 100 training examples, not many samples will have to be examined to find 100 examples that the first two networks disagree on. When the entire training set (including ones filtered out) are presented to the three networks to find the overall training rate, the performance is expected to be poor because you really have three networks with poor generalization.

Now consider, what happens when we increase the training size. The first network has worse training error but the generalization is better. Therefore more examples have to be sorted through to find the second and third training sets. As a consequence, the training set includes proportionately more examples filtered out and not used for actual training. These filtered out examples have close to zero error rate and tend to drive the training error rate [which includes both the error rate on the filtered out examples and the samples used for actual training (the computational cost)] down. In summary, as the training set size increases (Fig. 2), both N and N_1 increase but not at the same rate so the ratio $(3N_1)/N$ decreases. In the limit, this algorithm uses only 20% of the training set to actually train on.

The parallel machines and the single machine use all the patterns in the training set to train on. Therefore the ratio of computational cost to training set size is 1 for these algorithms. The modified boosting algorithm is more expensive: the computational cost is larger than the training set size.

Figure 3 shows the test performance of the four algorithms. Generally, each algorithm has a range where it is superior. For small training set size a single machine is better while for large training set size, both the modified boosting and the original boosting algorithm give similar results (with differences statistically insignificant at the largest training set size). However, the modified boosting algorithm is better over a broader range. We did not want to clutter this figure with error bars for each curve. However, we do report the smallest test error rate and standard deviation in Table 1 that occurs when the training and test performance are approximately equal.

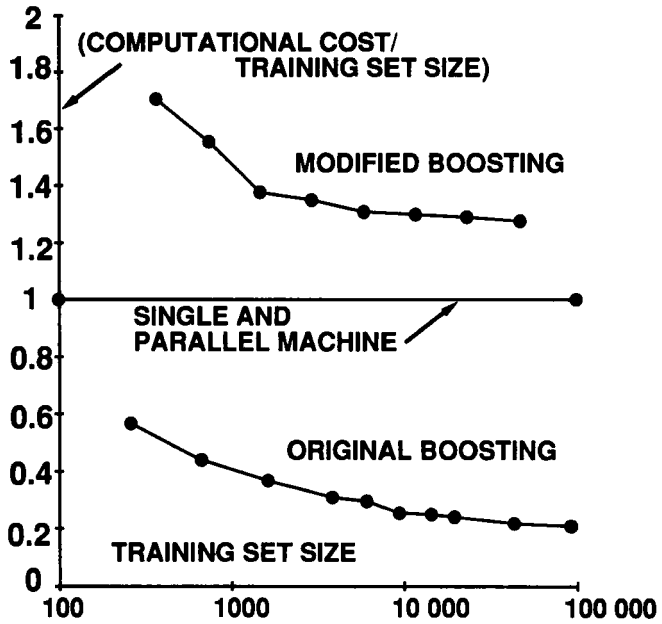

Figure 2: (Computational cost)/(training set size) versus training set size. For single and parallel machines, this ratio is 1. For the original boosting algorithm, not all the patterns in the training set are used and the ratio is less than 1. For the modified boosting algorithm the ratio is greater than 1.

For the 100-10-10 network, we ask which algorithm is better if one has a given time to train. This is shown as a plot of test error rate versus computational cost (Fig. 4). This clearly shows that the original boosting algorithm is better. This is not unexpected considering Figure 2, which shows that a smaller and smaller part of the training set contributes to the computational cost as the training set size increases.

We performed the same experiments on a single layer fully connected 100-10 network and a weight sharing network (LeCun *et al.* 1990) with 5400 connections but only 1014 free weights. The associated curves have the same general appearance of those in Figs. 3 and 4 with asymptotic characteristics in Table 1. An unusual characteristic of the 100-10 network is that the parallel and single machine have the same training and test performance for large training set size. This is not surprising in light of the fact that single layer networks have one minimum and all the machines in the parallel set of machines eventually obtain the same decision boundary. For this network, a single machine is as good as a

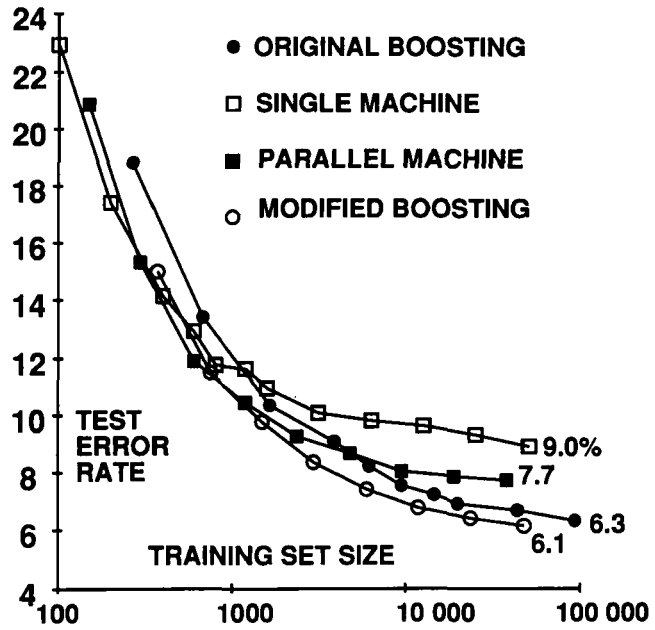

Figure 3: Test performance of four algorithms using a 100-10-10 network.

parallel machine. The characteristics of the training error curves for all these architectures would have shown the following: a negative slope for the original boosting algorithm, an initial positive slope and then turning flat or negative for the modified algorithm, and positive slopes for the remaining algorithms.

As pointed out previously, increasing the number of members in the ensemble is useless for a single layer network. For the original boosting algorithm, there are not enough potential training patterns in the NIST database and it is too computationally expensive to use the modified boosting algorithm. We therefore examined the possibility of using five members in the parallel machine for the 100-10-10 and weight sharing networks. In Table 1, we show the best results. As can be seen, this does not give statistically significant improvement in performance. The study by Perrone and Cooper (1993) shows that beyond five members in an ensemble, there is little improvement in what they term "figure-of-merit," which includes both the error rate and the ability to reject patterns (i.e., make no decisions).

Table 1: Minimum Test Error Rate and Standard Deviation (in Percent) for the Three Networks and Four Algorithms

Network	Algorithm	Mean	Standard Deviation
100-10-10	Single network	9.0	0.61
	Original boosting	6.3	0.34
	Modified boosting	6.1	0.28
	Parallel machine (3 members)	7.7	0.31
	Parallel machine (5 members)	7.6	0.38
100-10	Single network	12.1	0.37
	Original boosting	8.9	0.47
	Modified boosting	8.8	0.50
	Parallel machine (3 members)	11.9	0.23
Structured network	Single	8.9	0.55
	Original boosting	5.2	0.87
	Modified boosting	5.2	0.28
	Parallel machine (3 members)	7.1	0.51
	Parallel machine (5 members)	6.8	0.58

3 Conclusions

Figures 3 and 4 allow us to conclude that, for a given computational cost, the original form of boosting is best, although both boosting algorithms achieve the same asymptotic performance. It may seem that using a subset of the training set for the actual training is "throwing away data," but by using the networks to select the data to be trained on, one can do better than training on all the data.

In this investigation we constrained the architectures and input space so that the networks could be trained to their best performance within the constraints of the size of the database. One can do much better if one is willing to synthetically enlarge the database. One then can use all the training data and "new data" arise from using random deformations

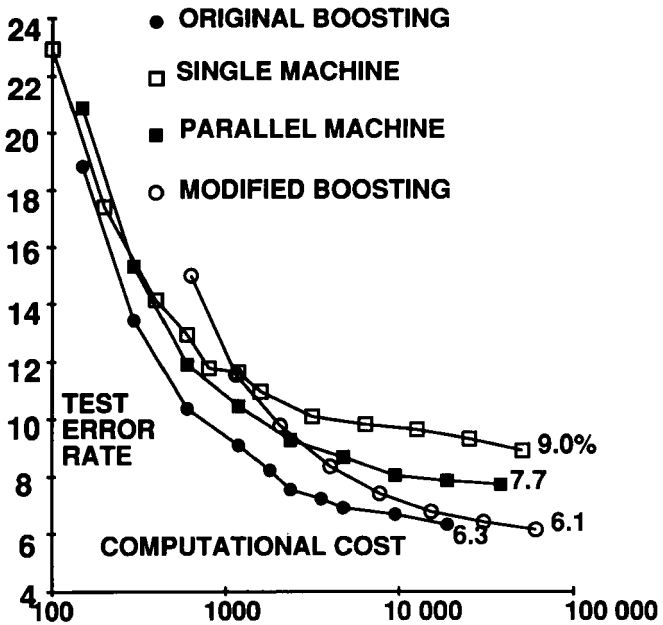

Figure 4: Test error rate versus computational cost for a 100-10-10 network.

of the original data. Deforming data (sometimes called image defect modeling) have been used extensively to improve the performance of non-neural-based, printed document readers (Baird 1993a,b). In Drucker *et al.* (1993a,b), we detail the deformation procedures.

Using deformations, 60,000 training examples, a full 28×28 input space, 10,000 test examples, and a very large shared-weight network with approximately 13,000 neurons, 17,000 weights, and 260,000 connections we are able to obtain a test error rate of 0.70% using boosting. For comparison purposes, results were obtained for the following networks (in increasing size of error rate): the single network shared-weight network with a 1.13% error rate, a fully connected 784-300-10 network with a 1.6% error rate, and a single-layer 784-10 network with an 8.4% test error rate.

An important observation made in this study is that the training error curve for the original boosting algorithm has negative slope until it asymptotes to some fixed value of training error. We make the following conjectures: (1) The capacity (in some sense which we cannot define yet) increases with increasing training set size until it reaches an asymptotic value. (2) "Good" algorithms will have this same negative

slope characteristic. We know that the difference of the test and training error rate should go to zero as we increase the size of the training set. Therefore, it is highly desirable that the training error curve has this negative slope so that the test error rate follows the training error rate to some small value. (3) Constructive algorithms such as that of cascade architectures (Littman and Ritter 1993), cascade-correlation (Fahlman and Lebiere 1990), and tiling (Mezard and Nadal 1989) that build networks incrementally and algorithms referenced above that use queries, filtering, hierarchical structures, or in general adapt the structure of the network or discard data intelligently have (or should have) this same negative slope.

We are attempting to prove or disprove these conjectures.

References

Atlas, L., Cohn, D., Ladner, R., El-Sharkawi, M. A., Marks, II, R. J., Aggoune, M. E., and Park, D. C. 1990. Training connectionist networks with queries and selective sampling. In *Advances in Neural Information Processing* 2, D. Touretzky, ed., pp. 566-573. Morgan Kaufmann, San Mateo, CA.

Baird, H. S. 1993a. Calibration of document image defect models. *Proc. 2nd Ann. Sympos. Document Anal. Inform. Retrieval* (available from Information Science Research Center, University of Nevada, Las Vegas).

Baird, H. S. 1993b. Document image defect models and their uses. *Proc. IAPR 2nd Int. Conf. Document Anal. Recognition*, Tsukuba Science City, Japan (available from Computer Society Press), 20–22.

Baum, E. B., and Lang, K. J. 1991. Constructing hidden units using examples and queries. In *Advances in Neural Information Processing 3*, R. Lippman, J. Moody, and D. Touretzky, eds., pp. 904–910. Morgan Kaufmann, San Mateo, CA.

Benediktsson, J. A., and Swain, P. H. 1992. Consensus theoretic classification methods. *IEEE Trans. Syst. Man Cybern.* **22**(4), 688–704.

Drucker, H., Schapire, R., and Simard, P. 1993a. Improving performance in neural networks using a boosting algorithm. In *Neural Information Processing Systems 5*, S. J. Hanson, J. D. Cowan, and C. L. Giles, eds., pp. 42–49. Morgan Kaufmann, San Mateo, CA.

Drucker, H., Schapire, R., and Simard, P. 1993b. Boosting performance in neural networks. *Int. J. Pattern Recognition Artif. Intelligence* **7**(4), 704–709.

Fahlman, S. E., and Lebiere, C. 1990. The cascade-correlation learning architecture. In *Advances in Neural Information Processing Systems* 2, E. S. Touretsky, ed., pp. 524–532. Morgan Kaufmann, San Mateo, CA.

Freund, Y. 1990. Boosting a weak learning algorithm by majority. *Proc. Third Annu. Workshop Comp. Learning Theory*, 202–206.

Hansen, L. K., and Salamon, P. 1990. Neural network ensembles. *IEEE Trans. Patterns Anal. Machine Intelligence* **12**(10), 993–1001.

Jordan, M. I., and Jacobs, R. A. 1992. Hierarchies of adaptive experts. In *Advances in Neural Information Processing Systems 4*, J. Moody, S. Hanson, and R. Lippman, eds., pp. 985–993. Morgan Kaufmann, San Mateo, CA.

Kearns, M., and Valiant, L. G. 1989. Cryptographic limitations on learning Boolean formulae and finite automata. *Proc. Nineteenth Annu. ACM Symp. Theory Computing*, 443–444.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. 1990. Handwritten digit recognition with a back-propagation network. In *Neural Information Processing Systems 2*, D. Touretzsky, ed., pp. 396–404. Morgan Kaufmann, San Mateo, CA.

Littman, E., and Ritter, H. 1993. Generalization abilities of cascade network architectures. In *Advances in Neural Information Processing Systems 5*, S. J. Hanson, J. D. Cowan, and C. L. Giles, eds., pp. 188–195. Morgan Kaufmann, San Mateo, CA.

Matic, N., Guyon, I., Bottou, L., Denker, J., Vapnik, V. 1992. Computer aided cleaning of large databases for character recognition. In *Proceedings of the 11th International Conference on Pattern Recognition*, vol. 2, pp. 330–333. IEEE Computer Society Press, Los Angeles, CA.

Mezard, M., and Nadal, J.-P. 1989. Learning in feedforward layered networks: The tiling algorithm. *J. Phys. A* **22**, 2191–2204.

Perrone, M. P., and Cooper, L. N. 1993. When networks disagree: Ensemble methods for hybrid neural networks. In *Neural Networks for Speech and Image Processing*, R. J. Mammone, ed. Chapman-Hall, London.

Schapire, R. 1990. The strength of weak learnability. *Machine Learn.* **5**(2), 197–227.

Srihari, S. 1992. High-performance reading machines. *Proc. IEEE* **80**(7), 1120–1132.

Sueng, H. S., Opper, M., and Sompolinsky, H. 1992. Query by committee. In *Proceedings of the 1992 Conference on Learning Theory*, ACM 0-89791-498-/8/92/0007/0287, 287–294.

Suen, C. Y. *et al.* 1992. Computer recognition of unconstrained handwritten numerals. *Proc. IEEE* **80**(7), 1162–1180.

Vapnik, V. 1982. *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, Berlin.

Received November 5, 1993; accepted March 15, 1994.