LETTER Communicated by Nicolas Brunel

Attractor Networks for Shape Recognition

Yali Amit
Massimo Mascaro
Department of Statistics, University of Chicago, Chicago, IL 60637, U.S.A.

We describe a system of thousands of binary perceptrons with coarse-oriented edges as input that is able to recognize shapes, even in a context with hundreds of classes. The perceptrons have randomized feed-forward connections from the input layer and form a recurrent network among themselves. Each class is represented by a prelearned attractor (serving as an associative hook) in the recurrent net corresponding to a randomly selected subpopulation of the perceptrons. In training, first the attractor of the correct class is activated among the perceptrons; then the visual stimulus is presented at the input layer. The feedforward connections are modified using field-dependent Hebbian learning with positive synapses, which we show to be stable with respect to large variations in feature statistics and coding levels and allows the use of the same threshold on all perceptrons. Recognition is based on only the visual stimuli. These activate the recurrent network, which is then driven by the dynamics to a sustained attractor state, concentrated in the correct class subset and providing a form of working memory. We believe this architecture is more transparent than standard feedforward two-layer networks and has stronger biological analogies.

1 Introduction

Learning and recognition in the context of neural networks is an intensively studied field. The leading paradigm from the engineering point of view is that of feedforward networks (to cite only a few examples in the context of character recognition, see LeCun et al., 1990; Knerr, Personnaz, & Dreyfus, 1992; Bottou et al., 1994, Hussain & Kabuka, 1994). Two-layer feedforward networks have proved to be very useful as classifiers in numerous other applications. However, they lack a certain transparency of interpretation relative to decision trees, for example, and their biological relevance is quite limited. Learning is not Hebbian and local; rather, it employs gradient descent on a global cost function. Furthermore, these nets fail to address the issue of sustained activity needed to deal with working memory.

On the other hand, much work has gone into networks in which learning is Hebbian, and recognition is represented through the sustained activity of attractor states of the network dynamics (see Hopfield, 1982, Amit, 1989, Amit & Brunel, 1995; Brunel, Carusi, & Fusi, 1998). In this field however research has primarily focused on the theoretical and biological modeling aspects. The input patterns used in these models are rather simplistic, from the statistical point of view. The prototypes for each class are assumed uncorrelated with approximately the same coding level. The patterns for each class are assumed to be simple noisy versions of the prototypes, with independent flips of low probability at each feature. These models have so far avoided the issue of how the attractors can be used for classification of real data, for example, image data, where the statistics of the input features are more complex. Indeed, if the input features of visual data are assumed to be some form of local edge filters, or even local functions of these edges, there is a significant variability in conditional distributions of these features given each class. This is described in detail in section 4.1.

In this article we seek to bridge the gap between these two rather disjoint fields of research yet trying to minimize the assumptions we make regarding the information encoded in individual neurons or in the connecting synapses, as well as staying faithful to the idea that learning is local. Thus, the neurons are all binary, and all have the same threshold. Synapses are positive with a limited range of values and can be effectively assumed binary. The network can easily be modified to have multivalued neurons; the performance would only improve. However, since it is still unclear to what extent the biological system can exploit the fine details of neuronal firing rates or spike timing, we prefer to work with binary neurons, essentially representing high and low firing rates. Similar considerations hold for the synaptic weights; moreover, the work in Petersen, Malenka, Nocoll, and Hopfield (1998) suggests that potentiation in synapses is indeed discrete.

The network consists of a recurrent layer A with random feedforward connections from a visual input layer I. Attractors are trained in the A layer, in an unsupervised way, using K classes of easy stimuli: noisy versions of K uncorrelated prototypes represented by subsets $A_c, c = 1, \ldots, K$, in which the neurons are on. The activities of the attractors are concentrated around these subsets. Each visual class is then arbitrarily associated with one of the easy stimuli classes and is hence represented by the subset or population A_c. Prior to the presentation of visual input from class c to layer I, a noisy version of the easy stimulus associated with class c is used to drive the recurrent layer to an attractor state so that activity concentrates in the population A_c. Following this, the feedforward connections between layers I and A are updated. This stage of learning is therefore supervised.

The easy stimuli are not assumed to reach the A layer via the visual input layer I. They can be viewed, for example, as reward-type stimuli arriving from some other pathway (auditory, olfactory, tactile) as described in Rolls (2000), potentiating cues described in Levenex and Schenk (1997), or mnemonic hooks introduced in Atkinson (1975). In testing, only visual input is presented at layer I. This leads to the activation of certain units in A, and the recurrent dynamics leads this layer to a stable state, which hopefully concentrates on the population corresponding to the correct class.

All learning proceeds along the Hebbian paradigm. A continuous internal synaptic state increases when both pre- and postsynaptic neurons are on (potentiation) and decreases if the presynaptic neuron is on but the postsynaptic neuron is off (depression). The internal state translates through a binary or sigmoidal transfer function to a synaptic efficacy. We add one important modification. If at a given presentation, the local input to a postsynaptic neuron is above a threshold (higher than the firing threshold), the internal state of the afferent synapses does not increase. This is called field-dependent Hebbian learning. This modification allows us to avoid global normalization of synaptic weights or the modification of neuronal thresholds. Moreover, field-dependent Hebbian learning turns out to be very robust to parameter settings, avoids problems due to variability in the feature statistics in the different classes, and does not require a priori knowledge of the size of the training set or the number of classes. The details of the learning protocol and its analysis are presented in section 4. Note that we are avoiding real-time dynamic architectures involving noisy integrate-and-fire neurons and real-time synaptic dynamics, as in Amit and Brunel (1995), Mattia and Del Giudice (2000), and Fusi, Annunziato, Badoni, Salamon, and Amit (1999), and are not introducing inhibition. In future work we intend to study the extension of these ideas to more realistic continuous time dynamics.

The visual input has the form of binary-oriented edge filters with eight orientations. The orientation tuning is very coarse, so that these neurons can be viewed as having a flat tuning curve spanning a range of angles, as opposed to the classical bell-shaped tuning curves. Partial invariance is incorporated by using complex type neurons, which respond to the presence of an edge within a range of orientations anywhere in a moderate-size neighborhood. This type of partially invariant unit, which responds to a disjunction (union) or a local MAX operation, has been used in Fukushima (1986), Fukushima and Wake (1991), Amit and Geman (1997, 1999), Amit (2000), and Riesenhuber and Poggio (1999).

From a purely algorithmic point of view, the attractors are unnecessary. Activity in the A layer can be clamped to the set A_c during training, and testing can simply be based on a vote in the A layer. In fact, the attractor dynamics applied to the visual input leads to a decrease in performance, as explained in section 3.2. However, from the point of view of neural modeling, the attractor dynamics is appealing. In the training stage, it is a robust way to assign a label to a class without pinpointing a particular individual neuron. In testing, the attractor dynamics cleans the activity in layer A; it is concentrated within a particular population, which again corresponds to the label. This could prove useful for other modules that need to read out the activity of this system for subsequent processing stages. Finally, the attractor is a stable state of the dynamics and is hence a mechanism for maintaining a sustained activity representing the recognized class, working memory, as in Miyashita and Chang (1988) and Sakai and Miyashita (1991).

We have chosen to experiment with this architecture in the context of shape recognition. We work with the NIST handwritten digit data set and a data set of 293 randomly deformed Latex symbols, which are presented in a fixed-size grid, although they exhibit significant variation in size as well as other forms of linear and nonlinear deformation. (In Figure 10 we display a sample from this data set.)

With one network with 2000 neurons in the *A* layer, we achieve a 94% classification rate on NIST (see section 5), a very encouraging result given the constrained learning and encoding framework we are working with. Moreover, a simple boosting procedure, whereby several nets are trained in sequence, brings the classification rate to 97.6%. Although far from the best reported in the literature, which is over 99% (see Wilkinson et al., 1992; Bottou et al., 1994; Amit, Geman, & Wilder, 1997), this is definitely comparable to many experimental papers on this subject (e.g., Hastie, Buja, & Tibshirani, 1995; Avimelelch & Intrator, 1999) where boosting is also employed. The same holds for the Latex database where, the only comparison is with respect to decision trees studied in Amit and Geman (1997). Boosting consists of training a new network on training examples that are misclassified by the combined vote of the existing networks. The concept of boosting is appealing in that hard examples are put aside for further training and processing. In this article, however, we do not describe explicit ways to model the actual neural implementation of boosting. Some ideas are offered in the discussion section.

In section 2 we discuss related work, similar architectures, learning rules, and the relation to certain ideas in the machine learning literature. In section 3 the precise architecture is described, as well as the learning rule and the training procedure. In section 4 we analyze this learning rule and compare it to standard Hebbian learning. In section 5 we describe experimental results on both data sets and study the sensitivity of the learning rule to various parameter settings.

2 Related work

2.1 Architecture. The architecture we propose consists of a visual input layer feeding into an attractor layer. The main activity during learning involves updating the synaptic weights of the connections between the input layer and the attractor layer. Such simple architectures have recently been proposed by Riesenhuber and Poggio (1999) and in Bartlett and Sejnowski (1998). In Riesenhuber and Poggio (1999) the second layer is a classical output layer with individual neurons coding for different classes. No recurrent activity is modeled during training or testing, but training is supervised. The input layer codes large-range disjunctions, or MAX operations, of complex features, which in turn are conjunctions of pairs of complex edge-type features. In an attempt to achieve large-range translation and scale invariance, the range of disjunction is the entire visual field. This means that objects are recognized based solely on the presence or absence of the features, entirely ignoring their location. With sufficiently complex features, this may well be possible, but then the combinatorics of the number of necessary features appears overwhelming. It should be noted that in the context of character recognition studied here, we find a significant drop in classification rates when the range of disjunction or maximization is on the order of the image size. This trade-off between feature complexity, combinatorics, and invariance is a crucial issue, which has yet to be systematically investigated.

In the architecture proposed in Bartlett and Sejnowski (1998), the second layer is a recurrent layer. The input features are again edge filters. Training is semisupervised, not directly through the association of certain neurons to a certain class but through the sequential presentation of slowly varying stimuli of the same class. Hebbian learning of temporal correlations is used to increase the weight of synapses connecting units in the recurrent layer responding to these consecutive stimuli. At the start of each class sequence, the temporal interaction is suspended. This is a very appealing alternative to supervision, which employs the fact that objects are often observed by slowly rotating them in space.

This network employs continuous-valued neurons and synapses, which in principle can achieve negative values. A bias input is introduced for every output neuron, which corresponds to allowing for adaptable thresholds, and depends on the size of the training set. Finally, the feedforward synaptic weights are globally normalized. It would be interesting to study whether field-dependent Hebbian learning can lead to similar results without the use of such global operations on the synaptic matrix.

2.2 Field-Dependent Hebbian Learning.

The learning rule employed here is motivated on one hand by work on binary synapses in Amit and Fusi (1994), Amit and Brunel (1995), and Mattia and Del Giudice (2000), where the synapses are modified stochastically only as a function of the activity of the pre- and postsynaptic neurons. Here, however, we have replaced the stochasticity of the learning process with a continuous internal variable for the synapse. This is more effective for neurons with a small number of synapses on the dendritic tree.

On the other hand we draw from Diederich and Opper (1987), where Hopfield-type nets with positive- and negative-valued multi-state synapses are modified using the classical Hebbian update rule, but modification stops when the local field is above or below certain thresholds.

There exists ample evidence for local synaptic modification as a function of the activities of pre and postsynaptic neurons (Markam, Lubke, Frotscher, & Sakmann, 1997; Bliss & Collingridge, 1993). However, at this point we can only speculate whether this process can be controlled by the depolarization or the firing rate of the postsynaptic neuron. One possible model in which such controls may be accommodated can be found in Fusi, Badoni, Salamon, and Amit (2000).

2.3 Multiple Classifiers. From the learning theory perspective, the approach we propose is closely related to current work on multiple classifiers. The idea of multiple randomized classifiers is gaining increasing attention in machine learning theory (Amit & Geman, 1997; Breiman, 1998, 1999; Dietterich, 2000; Amit, Blanchard, & Wilder, 1999). In the present context, each unit in layer A associated with a class c is a randomized perceptron classifying class c against the rest of the world, and the entire network is an aggregation of multiple randomized perceptrons. Population coding among a large number of such linear classifiers can produce very complex classification boundaries. It is important to note, however, that the family of possible perceptrons available in this context is very limited due to the constraints imposed on the synaptic values and the use of uniform thresholds.

A complementary idea is that of boosting, where multiple classifiers are created sequentially by up-weighting the misclassified data points in the training set. Indeed in our context, to stabilize the results produced using the attractors in layer A and to improve performance, we train several nets sequentially using a simplified version of the boosting proposed in Schapire, Freund, Bartlett, and Lee (1998) and Breiman (1998).

3 The Network

The network is composed of two layers: layer I for the visual input and a recurrent layer A where attractors are formed and population coding occurs. Since it is still unclear to what extent biological systems can exploit the fine details of neuronal firing rates or spike timing, we prefer to work with binary 0/1 neurons, essentially representing high and low firing rates.

The I layer has binary units $i = (x, e)$ for every location x in the image and every edge type $e = 1, \ldots, 8$ corresponding to eight coarse orientations. Orientation tuning is very coarse so that these neurons can be viewed as having a flat tuning curve spanning a range of angles, as opposed to the classical bell-shaped tuning curves. The unit (x, e) is on if an edge of type e has been detected anywhere in a moderate-size neighborhood (between 3×3 and 10×10) of x. In other words, a detected edge is spread to the entire neighborhood. Thus, these units are the analogs of complex orientation-sensitive cells (Hubel, 1988). In Figure 1 we show an image of a 0 with detected edges and the units that are activated after spreading. We emphasize that recognition does not require careful normalization of the image to a fixed size. Observe, for example, the significant variations in size and other shape parameters of the samples in the bottom panel of Figure 10. Therefore, the spreading of the edge features is crucial for maintaining some invariance to deformations and significantly improves the generalization properties of any classifier based on oriented edge inputs.

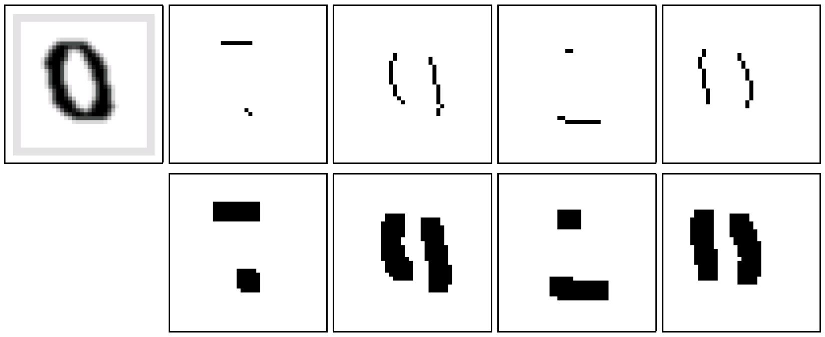

Figure 1: (Top, from left to right): Example of a deformed Latex 0; Detected edges of angle 0, edges of angle 90, edges of angle 180, edges of angle 270 (note that the edges have a polarity). (Bottom, left to right): Same edges spread in a 5×5 neighborhood.

The attractor layer has a large number of neurons N_A—on the order of thousands—with randomly assigned recurrent connections allocated to each pair of neurons with probability p_{AA}. The feedforward connections from I to A are also independently and randomly assigned with probability p_{IA} for each pair of neurons in I and A, respectively. This system of connections is described in Figure 2.

Each neuron $a \in A$ receives input of the form

$$h_a = h_a^I + h_a^A = \sum_{i=0}^{N_I} E_{ia} u_i + \sum_{a'=0}^{N_A} E_{a'a} u_{a'}, \tag{3.1}$$

where u_i, u_a represent the states of the corresponding neurons and $E_{ia}, E_{a'a}$ are the efficacies of the connecting synapses. The efficacy is assumed 0 if no synapse is present. The input h_a is also called the local field of the neuron and decomposes into the local field due to feedforward input—h_a^I—and the local field due to the recurrent input—h_a^A. The neurons in A respond to their input field through a step function:

$$u_a = \Theta(h_a - \theta), \tag{3.2}$$

where

$$\Theta(x) = \begin{cases} 1 & \text{for } x > 0 \\ 0 & \text{for } x \le 0. \end{cases} \tag{3.3}$$

All neurons are assumed to have the same firing threshold θ; they are all excitatory and are updated asynchronously. Inhibitory neurons could be

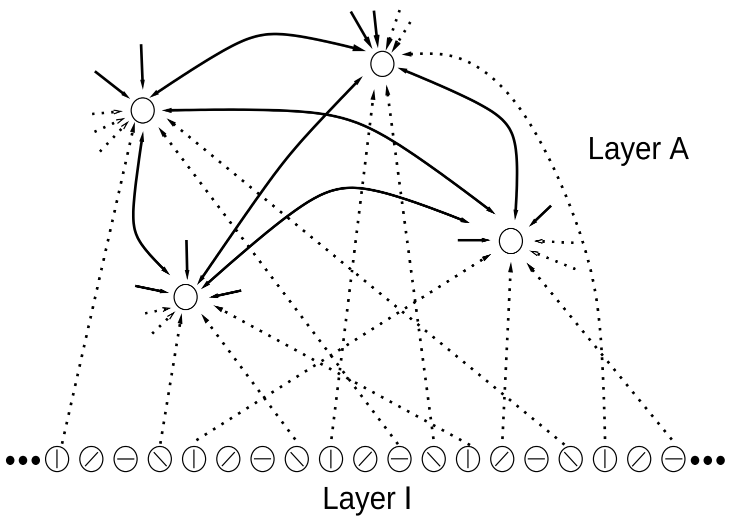

Figure 2: Overview of the network. There are two layers, denoted *I* and *A*. Units in layer *A* receive inputs from a random subset of layer *I* and from other units in layer *A* itself. Dashed arrows represent input from neurons in *I*, solid ones from neurons in *A*.

introduced in the attractor layer to perhaps stabilize or create competition among the attractors.

Each synapse is characterized by an internal state variable S. This state has two reflecting barriers, forcing it to stay in the range [0, 255]. The efficacy E is a deterministic function of the state. This function is the same for all synapses of the net:

$$\begin{cases} E(S) = 0 \quad \text{for} \quad S \leq L \\ E(S) = \frac{S-L}{H-L} J_m \quad \text{for} \quad L < S < H \\ E(S) = J_m \quad \text{for} \quad S \geq H. \end{cases} \tag{3.4}$$

The parameter L defines the minimal state at which the synapse is activated, and H defines the state above which the synapse is saturated. Varying L and H produces a broad range of synapse types, from binary synapses when $L = H$, to completely continuous ones in the allowed range ($L = 0$, $H = 255$). The parameter J_m represents the saturation value of the efficacy. A graphical explanation of these parameters is provided in Figure 3.

3.1 Learning. Given a presynaptic unit s and a postsynaptic unit t, the synaptic modification rule is of the form

$$\Delta S_{st} = C_+(h_t) u_s u_t - C_-(h_t) u_s (1 - u_t), \tag{3.5}$$

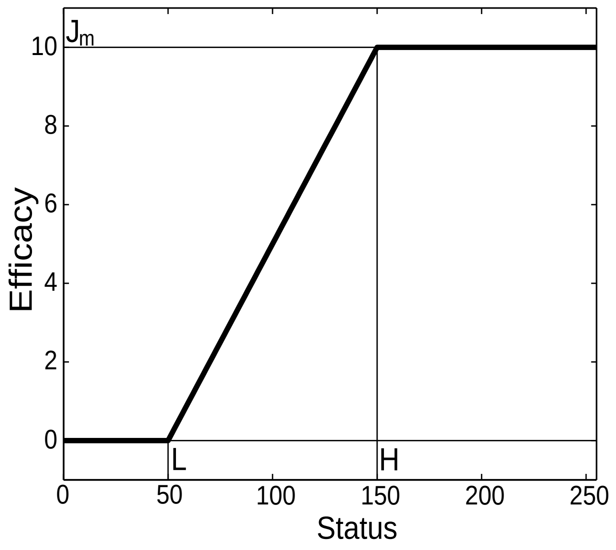

Figure 3: The transfer function for converting the internal state of the synapse to its efficacy. The function is 0 below L, and saturates at J_m above H. In between it is linear. 0 and 255 are reflecting barriers for the status. In this example, L $= 50$, H $= 150$, $J_m = 10$.

where

$$C_+(h_t) = C_p\Theta(\theta(1 + k_p) - h_t), \tag{3.6}$$

$$C_-(h_t) = C_d\Theta(h_t - \theta(1 - k_d)). \tag{3.7}$$

Typically k_p ranges between .2 and .5, and $k_d \leq 1$. In our experiments, C_d is fixed to 1, while C_p ranges from 4 to 10.

In regular Hebbian learning, whenever both the pre- and postsynaptic neurons are on, the internal state increases, and whenever the presynaptic neuron is on and the postsynaptic neuron is off, the internal state decreases. In other words, the functions C_+ and C_- do not depend on the local field and are held constant at C_p and C_d, respectively. The effect of the proposed learning rule is to stop synaptic potentiation when the input to the postsynaptic neuron is too high (higher than $\theta(1 + k_p)$) or to stop synaptic depression when the input is too low (less than $\theta(1 - k_d)$). Of course, once a different input is presented, the field changes, and potentiation or depression may resume.

The parameters k_p (potentiation) and k_d (depression) regulate to what degree the field is allowed to differ from the firing threshold before synaptic modification is stopped. These parameters may vary for different types of synapses—for example, the recurrent synapses versus the feedforward ones. The parameters C_p and C_d regulate the amount of potentiation and depression applied to the synapse when the constraints on the field are satisfied. The transfer from internal synaptic state to synaptic efficacy needs to be computed at every step of learning in order to calculate the local field h_a of the neuron in terms of the actual efficacies.

This learning rule bears some similarities to the perceptron rule (Rosenblatt, 1962), the field-dependent rules proposed by Diederich and Opper (1987), and the penalized weighting in Oja (1989). However, it is implemented in the context of binary 0/1 neurons and nonnegative synapses. In some experiments, synapses have been constrained to be binary with a small loss in performance.

The main purpose of field constraints on synaptic potentiation is to keep the average local field for any unit $a \in A_c$ to be approximately the same during the presentation of an example from class c, irrespective of the specific distribution of feature probabilities for that class. This enables the use of a fixed threshold for all neurons in layer A. In contrast to regular Hebbian learning, this form of learning is not entirely local at the synaptic level due to the dependence on the field of the neuron. However, if continuous spiking units were employed, the field constraints could be reinterpreted as a firing-rate constraint. For example, when the postsynaptic neuron fires at a rate above some level, which is significantly higher than the rate determined by the recurrent input alone, potentiation of any incoming synapses ceases. It should also be noted that the role of the internal synaptic state is to slow down synaptic modifications similar to the idea of stochastic learning in Brunel et al. (1998) and Amit and Brunel (1995). A more detailed discussion of the properties of this learning rule is provided in section 4.

3.1.1 Learning the Recurrent Efficacies. We assume that the A layer receives noisy versions of K uncorrelated prototypes from some external source. These are well separated, "easy" stimuli, which are associated with the different visual classes. Although they are well separated, we still expect some noise to occur in their activation. Specifically, we include each unit in A independently with probability p_A in each of K subsets A_c. Each of these random subsets defines a prototype: all neurons in the subset A_c are on, and all in its complement are off. The subsets belonging to each class will be of different size, fluctuating around an average of $p_A N_A$ units with a standard deviation of $\sqrt{p_A(1 - p_A)N_A}$. A typical case for a small NIST classifier, as in section 5, would be $N_A = 200$ and $p_A = .1$, so that the number of units in each class will be 20 ± 4. Random overlaps—namely units belonging to more than one subset—are significant. For example, in the case above, about 60% of the units of a subset will also belong to other subsets. Given a noise level f (say, 5–10%), a random stimulus from class c will have each neuron in A_c on with probability $1 - f$ and each neuron in the complement of A_c on with probability f. Thus, the sets A_c correspond to prototype patterns, and the actual stimuli are noisy versions of these prototypes. The recurrent synapses

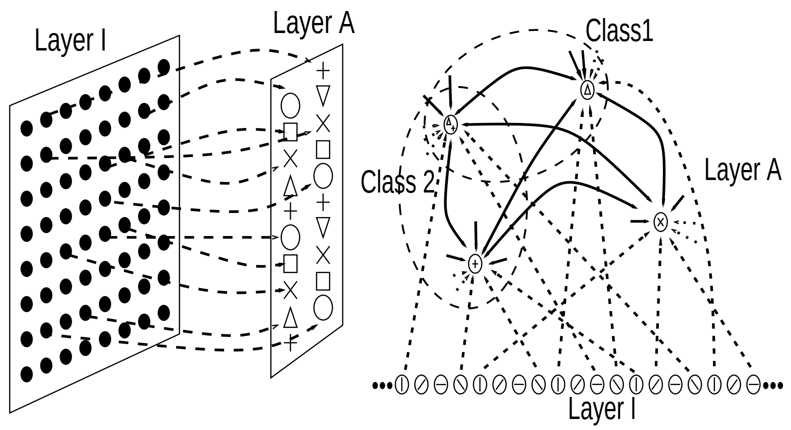

Figure 4: (Left) Global view of layers I and A, emphasizing that the subsets in A are random and unordered. Each class is represented by a different symbol. (Right) Close-up showing the input features extracted in I, random connections between the layers, recurrent connections in A, and possible overlaps between subsets in A.

in A are modified by presenting random stimuli from the different classes in random order. Modification proceeds as prescribed in equation 3.5.

After a sufficient number of presentations, this network will develop attractors corresponding to the subsets A_c with certain basins of attraction. Moreover, whenever the net is in one of these attractor states, the average field on each neuron in a subset A_c will be approximately $\theta(1 + k_p)$. Note that the fluctuations in the size of the sets A_c typically reduce the maximum memory capacity of the network, (see Brunel, 1994). We use subsets of several tens of neurons so that p_A is very small. Figure 4 provides a graphic description of the way the different populations are organized. We reemphasize that direct local learning of attractors on the visual inputs of highly variable shapes is impossible due to the complex structure of the statistics of the input features. Large overlaps exist between classes, and coding levels vary significantly.

3.1.2 Learning the Feedforward Efficacies. Once the attractors in layer A have been formed, learning of the feedforward connections can begin. Preceding the presentation of the visual input of a data point from class c to layer I, the corresponding external input (again with noise) is presented to A and the associated attractor state emerges, with activity concentrated around the subset A_c. Learning then proceeds by updating the synaptic state variables S_{ia} between layers I and A according to the field learning rule.

Note that the field at a unit $a \in A$ is composed of h_a^I and h_a^A. However, since we assume that at this stage of learning the A layer is in an attractor state, the field h_a^A is approximately $\theta(1 + k_p)$, and we rewrite the learning rule for the feedforward synapses only in terms of the feedforward local field h_a^I,

$$\Delta S_{ia} = C_+(h_a^I)u_a u_i - C_-(h_a^I)(1 - u_a)u_i, \tag{3.8}$$

where $a \in A$ has a connection from $i \in I$, and

$$C_+(h_a^I) = C_p \Theta(\theta(1 + k_p) - h_a^I), \tag{3.9}$$

$$C_-(h_a^I) = C_d \Theta(h_a^I - \theta(1 - k_d)), \tag{3.10}$$

using the same parameters k_p, k_d as in the recurrent synapses.

This same rule can be written in terms of the entire field h_a, if the value of k_p on the feedforward connections is modified to $k_p' = 2k_p + 1$. Whereas for recurrent synapses potentiation stops when the local field is above $\theta(1+k_p)$, for feedforward synapses potentiation stops when the local field is above $\theta(1 + k_p') = \theta(2 + 2k_p)$.

3.2 Recognition Testing and the Role of the Attractors. After training, the feedforward and recurrent synapses are held fixed, and testing can be performed. A pattern is presented only at the input level I. The activity of the neurons in this layer creates direct activation in some of the neurons in layer A. First, it is possible to carry out a simple majority vote among the classes—how many neurons were activated in each of the subsets A_c. This yields a classification rule, which corresponds to a population code for the class. The architecture is then nothing but a large collection of randomized perceptrons, each trained to discriminate between one class or several classes (in the case where a unit in layer A happens to be selective to several classes,) against the "rest of the world"—all other classes merged together as one. Recall that synaptic connections from layer I to A are chosen randomly. For each pair $i \in I, a \in A$ the connection is present independently with probability p_{IA}, (which may vary between 5% and 50%). This randomization ensures that units assigned to the same class in layer A produce different classifiers based on different subsets of features.

The performance of this scheme is surprisingly good given the simplicity of the architecture and the training procedure. For example, on the feedforward nets with 50,000 training and 50,000 test samples on the NIST database, we have observed a classification rate of 94%.

When recurrent dynamics are applied to layer A, initialized by the direct activation produced by the feedforward connections, we obtain convergence to an attractor state that serves as a mechanism for working memory. Moreover, the activity is concentrated on the recognized class alone. This is a nontrivial task since the noise around the actual subset A_c produced by the bottom-up input can be substantial and is not at all the same as the independent f-level noise used to train the attractors. The classification rates with the attractors on an individual net are therefore often lower than voting based on the direct activation from the feedforward connections. In some cases, the response to a given stimulus is lower than usual, and not enough neurons are activated in any of the classes to reach an attractor. Alternatively, a stimulus may produce a strong enough response in more than one class, producing a stable state composed of a union of attractors. In both cases, the outcome after the net converges to a stable state could be wrong even if the initial vote was correct. One more source of noise on the attractor level lies in the overlap between the populations A_c in the A layer.

3.3 Boosting. A popular method to improve the performance of any particular form of classification is boosting, (see, e.g., Schapire et al., 1998). The idea is to produce a sequence of classifiers using the same architecture and the same data. At each stage, the sequence of classifiers is aggregated and produces an outcome based on voting among the classifiers. Furthermore, at each stage, those training data points that remain misclassified are up-weighted. A new classifier is then trained using the new weighting of the training set. In the extreme case, each new classifier is constructed only from the misclassified training points of the current aggregate. We employed this extreme version for simplicity.

More precisely, assume k nets $A_1, \ldots, A_k$ have been trained, each one with a separate attractor layer, and synaptic connections to the the same input layer I. The aggregate classifier of the k nets is the vote among all neurons in the k attractor layers. Only the misclassified training points of this aggregate net are used to train the $k+1$ net. After net A_{k+1} is trained, in terms of both the recurrent connections and the feedforward connections, it joins the pool of the k existing networks for voting. Note that voting is not weighted; all neurons in all A_k layers have equal vote. Examples of classification with boosting are given in section 5.

4 Field Learning

In this section we study the properties of field learning on the feedforward connections in greater detail. Henceforth, the field refers only to the contribution of the input layer, h_a^I. Recall that each feature $i = (x, e)$ is on if an edge of type e is present somewhere in a neighborhood of x. For each i and class c, let $p_{i,c}$ denote the probability that $u_i = 1$ for images in class c and $q_{i,c}$ the probability that $u_i = 1$ for images not in class c. Let $P(c), c = 1, \ldots, K$ denote the prior on class c, namely, the proportions of the different classes, and let $\tilde{p}_{i,c} = p_{i,c}P(c)$, $\tilde{q}_{i,c} = q_{i,c}(1 - P(c))$. Finally let

$$\rho_{i,c} = \frac{\tilde{p}_{i,c}}{\tilde{q}_{i,c}}. \tag{4.1}$$

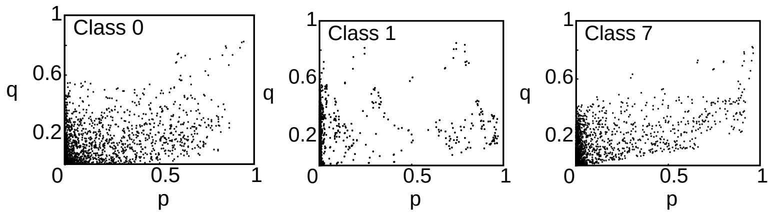

Figure 5: *pq* scatter plots for classes 0, 1, and 7 in the NIST data set. Each point corresponds to one of the N_I features. The horizontal axis represents the on-class *p*-probability of the feature, and the vertical axis is the off-class *q*-probability.

For the type of features we are employing in this context and in many other real-world examples, the distribution of $p_{i,c}, q_{i,c}, i = 1, \ldots, N_I$, is highly variable among different classes. In Figure 5 we show the two-dimensional scatter plots of the p versus q probabilities for classes 0, 1, and 7 from the NIST data set. Each point corresponds to a feature $i = 1, \ldots, N_I$. Henceforth, we denote these scatter plots the *pq* distributions. Note, for example, that there are many features common to 0s and other classes such as 3s, 6s, or 8s, especially due to the spreading we implement on the precise edge locations. Thus, many features have high $p_{i,0}$ and $q_{i,0}$ probabilities. On the other hand, only a small number of features have high probability on 1s, but these typically also have a low probability on other classes.

For simplicity, consider the units in layer A that belong to only one subset, $A_c \subset A$. Each is a perceptron trained to classify class c against the rest of the classes. We assume all neurons in layer A have the same firing threshold θ. Imagine now performing Hebbian learning of the form suggested in Amit and Brunel (1995) or Brunel et al. (1998), where C_p, C_d do not depend on the field. Calculating the mean synaptic change, one easily gets for $a \in A_c$,

$$\langle \Delta S_{ia} \rangle = C_p \tilde{p}_{i,c} - C_d \tilde{q}_{i,c}, \tag{4.2}$$

so that assuming a large number of training examples for each class, S_{ia} will move toward one of the two reflecting barriers 0 or 255 according to whether $\frac{\tilde{p}_{i,c}}{\tilde{q}_{i,c}}$ is less than or greater than the fixed threshold $\tau = \frac{C_d}{C_p}$. The resulting synaptic efficacies will then be close to either 0 or J_m, respectively. In any case, the final efficacy of each synapse will not depend at all on the *pq* distribution of the particular class.

However, since these *pq* distributions vary significantly across different classes, using a fixed threshold τ will be problematic. If τ is high, some classes will not have a sufficient number of synapses potentiated, causing the neuron to receive a small total field. The corresponding neurons will rarely fire, leading to a large number of false negatives for that class. Lowering τ will cause other classes to have too many synapses potentiated, and the

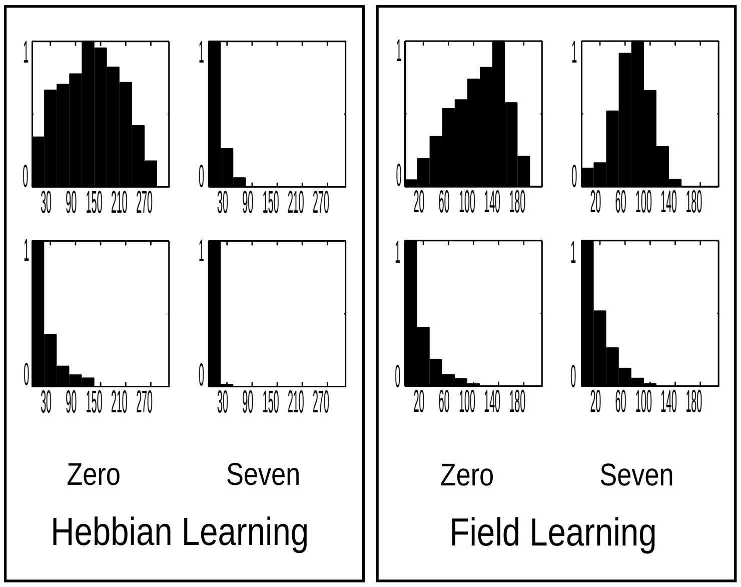

Figure 6: (Left) Hebbian learning. (Right) Field learning. In each panel, the top two histograms are of fields at neurons belonging to correct class of that column (0 on the left, 7 on the right) when data points from the correct class are presented. The bottom row in each panel shows histograms of fields at neurons belonging to classes 0 and 7 when data points from other classes are presented. The firing threshold was always 100.

corresponding neurons will fire too often because their field will often be very high, leading to a large number of false positives.

One solution could be to maintain Hebbian learning but adapt the threshold of the neurons of each class (and possibly each neuron). This would require a very wide range of thresholds and a sophisticated threshold adaptation scheme (see the left panel of Figure 6). The field learning rule is an alternative, which uses field-dependent thresholds on synaptic potentiations as in equation 3.8. The field is calculated in terms of the activated features in the current example and the current state of the synapses. Potentiation occurs only if the postsynaptic neuron is on, that is, the example is from the correct class, *and* the field is below $\theta(1 + k_p)$. Depression occurs only if the postsynaptic neuron is off, that is, the example is from the wrong class, *and* the field is above $\theta(1-k_d)$. Hence, asymptotically in the number of training examples, the mean field at a neuron of class c, over the data points of that class, will be close to $\theta(1+k_p)$, irrespective of the particular type of pq distribution associated with that class. Furthermore the mean field at that neuron, over data points outside class c, will be around $\theta(1 - k_d)$.

This is illustrated in the two panels on the right of Figure 6, where we show the distribution of the field for a collection of neurons of classes 0 and 7 when the correct and incorrect class is presented.

The stability in the value of the mean field across the different classes allows us to use a fixed threshold for all neurons. For comparison, on the left of Figure 6, we show the histograms of the fields when Hebbian learning is employed, all else remaining the same. Note the large variation in the location of these histograms, excluding the use of a fixed threshold for classification. Of course, the distributions of the histograms around the means are very different and depend on the class, meaning that for some classes, the individual perceptrons will be more powerful classifiers than for others. Note also that these histograms in no way represent the classification power of voting among the neurons in layer A. The field for a data point of class c is not constant over all neurons of that class; it may be below threshold in some and above threshold in others.

4.1 Field Learning and Feature Statistics. The question is, Which synapses does the learning tend to potentiate? For Hebbian learning, the answer is simple: all synapses connecting features for which $\rho_{i,c} = \frac{\tilde{p}_{i,c}}{\tilde{q}_{i,c}}$ is greater than $\frac{C_d}{C_p}$. If there are many such synapses, due to the constraint on the field, not all such synapses can be potentiated when field-dependent learning is implemented. There is some form of competition over synaptic resources. On the other hand, if there is only a small number of such synapses, the learning rule will try to employ more features with a lower ratio.

In the simplified case where no constraints are imposed on depression, that is, $k_d = 1$, the synapses connected to features with a higher $\rho_{i,c}$ ratio have a much larger chance of being potentiated asymptotically in the number of presentations of training data. Those with higher $p_{i,c}$ may get potentiated sooner, but asymptotically, synapses with low $p_{i,c}$ but large $\rho_{i,c}$ gradually get potentiated as well.

To provide a better understanding of this phenomenon, we constrain ourselves to binary synaptic efficacies: the transfer function of equation 3.4 has $0 < L = H < 255$. We generate a synthetic feature distribution, consisting of three groups of features with the following on-class and off-class conditional probabilities: (1) $.7 \leq p \leq 0.8$, $0.1 \leq q \leq 0.2$, (2) $0.1 \leq p \leq 0.5$, $q \approx 0.05$, and (3) $0 \leq p \leq 0.6$, $0.4 \leq q \leq 0.8$. The features are assumed conditionally independent given class.

The idea is that groups 1 and 2 contain informative features with a relatively high $\frac{p}{q}$ ratio. However in group 1, both p and q are high, and it is interesting to see how the learning chooses among these two sets. Group 3 is introduced as a control group to verify that uninformative features are not chosen. In Figure 7 we show which synapses the field learning has chosen to potentiate in the pq plane when $k_d = 1$ at two stages in the training process.

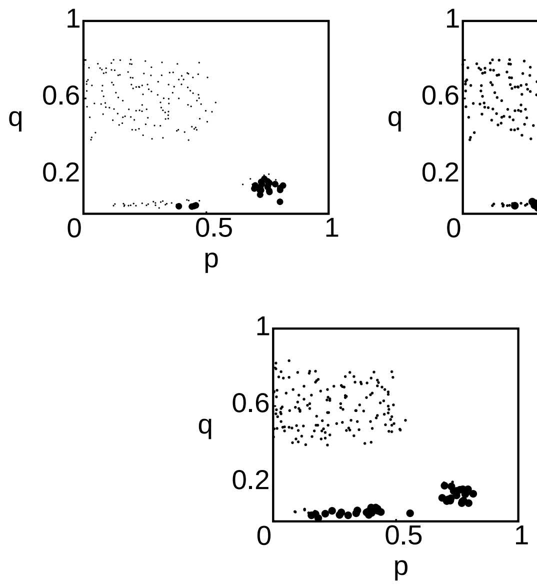

Figure 7: (Top, left to right) Features corresponding to potentiated synapses at two stages (epochs 1 and 10, left to right) of the training process for the synthetic data set ($k_d = 1$), overlaid on the *pq* scatter plot. (Bottom) Features selected by the synthetic learning rule. Horizontal axis: on-class *p*-probabilities. Vertical axis: off-class *q*-probabilities.

Note that initially features with large p_i and smaller ρ_i are potentiated, but gradually the features with higher ρ_i and lower p_i take over.

The outcome of the potentiation rule can be synthetically mimicked as follows. Sort all features according to their ρ_i ratio. Let $p_{(i)}$ denote the sorted on-class probabilities in decreasing order of ρ. Pick features from the top of the list so that

$$\sum_{(i)=1}^{Q} J_m p_{(i)} \sim \theta(1 + k_p). \tag{4.3}$$

See Figure 7. The value of Q will depend on the *pq* statistics of the neuron one is considering. In fact, using this synthetic potentiation rule, we obtain very similar classification results on a small network applied to the NIST data set, (see Table 1).

From the purely statistical point of view, the conditional independence of the features allows us to provide an explicit formula for the Bayes classifier, which is linear in the inputs (see, e.g., Duda & Hart, 1973), and in fact employs all features. This linear classifier can be implemented with one

Table 1: Performance of a System of 500 Neurons with Field Learning and Binary Synapses.

Rule Type	Rate Observed
Field learning	88%
Synthetic rule	86%

perceptron with the appropriate weights and threshold. However, due to the assumption of a constant-threshold, positive-bounded synaptic efficacies, it is not possible to obtain this optimal performance with any one of the individual perceptrons defined here.

Another point to be emphasized is that the heuristic explanation provided here regarding the outcome of field learning is only in terms of the marginal distributions of the features in each class. In reality, the features are not conditionally independent, and certain correlations are very informative. It is not clear at this point whether the field learning is picking up any of this information.

The situation when $k_d < 1$ is more complex. It appears that in this case, there is some advantage for features with large p_i and lower ρ_i. Due to the limitation on depression, these synapses tend to persist in the potentiated state despite the relatively larger q_i parameters. Recall that depression depends on the magnitude of the fields when the wrong class is presented. If these are relatively low, no depression occurs, even if features corresponding to potentiated synapses have been activated by the current data point. Moreover, with less depression taking place, during the presentation of an example of the correct class, the field will be high enough even without the low p_i–high ρ_i features disabling potentiation and thus not allowing their synapses to strengthen. In Figure 8 we show the potentiated features in the

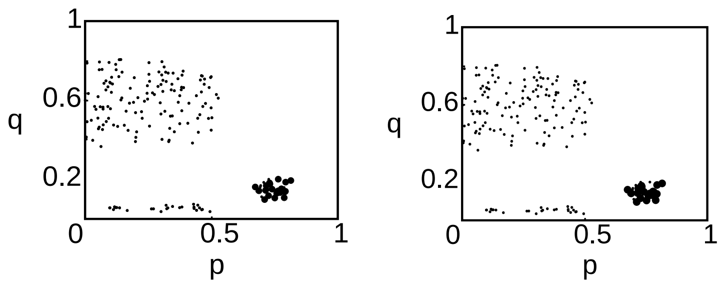

Figure 8: (Left to right) Features corresponding to potentiated synapses at two stages in the training process for the synthetic data set ($k_d = .2$), overlaid on the *pq* scatter plot. Horizontal axis: on-class *p*-probabilities. Vertical axis: off-class *q*-probabilities.

Table 2: Summary of Parameters Used in the System.

J_m	Maximum value for the synaptic strength
L	Lower bound on the linear part of the efficacy
H	Upper bound on the linear part of the efficacy
k_p	$(1 + k_p)\theta$ is upper threshold for potentiation
k_d	$(1 - k_d)\theta$ is lower threshold for depression
C_p	Synaptic status increase following potentiation
C_d	Synaptic status decrease following depression
p_{IA}	Probability of existence of a synapse between a unit in layer I and a unit in layer A
p_{AA}	Probability of existence of a synapse between two units in layer A
p_A	Probability of a unit in A to be selected to a subset A_c
N_A	Total number of neurons in each net in layer A
Spread	Spread of each feature in layer I
Iter	Learning iterations per data point

pq plot for $k_d = .2$ at two stages of the training process. Observe that even at the end, features with higher p_i and lower ρ_i remain potentiated.

5 Experimental Results

In Table 2 we provide a brief summary of the parameters introduced to help in reading the experimental results in the following sections. In order to make the experiments more time efficient, we have reduced the size of the A layer to more or less accommodate the number of classes involved in the problem. Thus, for the NIST data set with 10 classes, we used $N_A = 200$ neurons with $p_A = .1$ and for the Latex data set with 293 classes, we used $N_A = 6000$ with $p_A = 0.01$. As mentioned in section 3, ultimately layer A should have tens of thousands of neurons to accommodate thousands of recognizable classes. Keeping fixed the average number of neurons per class, a larger A for a fixed number of classes can only improve classification results because the subsets remain the same size and will therefore be virtually disjoint, further stabilizing the outcome of the attractors.

5.1 Training the Recurrent Layer A. The original description of the training procedure for the feedforward synapses was as follows. Some noisy version of the prototype A_c is presented to layer A. This layer converges to the attractor state concentrated around A_c, and finally a visual input from

Table 3: Base parameters for the parameter scan. Only parameters below the line are modified during the scan.

$L = 100$	$H = 100$	$\theta = 100$	$N_A = 100$	Spread 3×3
Iter = 2	$p_A = .1$	$C_d = 1$	$p_{AA} = 1$	
$k_p = .2$	$k_d = .2$	$C_p = 4$	$p_{IA} = .1$	$J_m = 10$

class c is presented to layer I, after which the feedforward synapses are updated. In the experiments below, we skip the first step, which involves dynamics in the A layer and is very time-consuming on large networks. If the noise level f is sufficiently low, we can assume that only units in A_c will be activated after the attractor dynamics converges. Thus, we directly turn on all units in A_c and keep the others off prior to presenting a visual stimulus from class c. This shortcut has no effect on the final results. If, however, the attractor dynamics is not implemented in layer A, so that the label activation is noisy during the training of the feedforward connections, performance decreases. It is clear that the attractors ensure a coherent internal representation of the labels during training.

For the stable formation of attractors in the A layer, there is a critical noise level f_c above which the system destabilizes. This level depends on the size of the network, the size of the sets A_c, and other parameters. For the system used for NIST, $f_c \sim 0.2$. For the 293-class Latex database with $N_A = 6000$, we found f_c to be around .05.

5.2 Parameter Stability. In order to test the stability of field learning on the feedforward connections with respect to the various parameters, we performed a fast survey with a single net. Here the goal was not to optimize the recognition rate or the performance of our system, but merely to check the robustness of our algorithm. The simulations were based on a simple net composed of only 100 neurons, (on average, 10 per class), trained on 10,000 examples of handwritten digits taken from the NIST data set. The images in the NIST data set of handwritten digits are normalized to 16×16 gray-level images with a primitive form of slant correction, after which edges are extracted.

We start with the system described by the parameters in Table 3. The recognition rate (89%) is quite high considering the size of the system. We then vary one parameter at a time, trying two additional values. This does not provide a full sample of the parameter space, but since we are not interested in optimization, this is not crucial.

In Table 4 we report the results. Note that despite the large variability, the final rates remain well above the random choice 10% rate. Moreover even for parameters yielding low classification rates with a small number of neurons (100), it is possible to achieve great improvement by increasing the number of neurons in layer A. In one experiment, the parameters were

Table 4: Recognition Rates for Parameter Scan on the NIST Data Set.

Parameter Name	First Change	Second Change
J_m	$20.0 \rightarrow 86\%$	$5.0 \rightarrow 74\%$
k_d	$0.50 \rightarrow 84\%$	$1.0 \rightarrow 51\%$
k_p	$0.50 \rightarrow 90\%$	$1.0 \rightarrow 82\%$
C_p	$10.0 \rightarrow 87\%$	$2.0 \rightarrow 89\%$
p_{IA}	$.05 \rightarrow 74\%$	$0.2 \rightarrow 91\%$

Note: Each cell contains the new parameter value and the resulting rate.

fixed for the worst rate in Table 4—$k_d = 1$ and all else the same—and the number of neurons was gradually increased. The results are reported in Table 5, where they are compared to those obtained by the best parameter set, $p_{IA} = 0.2$, and all else the same. The point here is that in a system of thousands of neurons, the values of these quantities are not critical. The use of boosting further increases the asymptotic value achievable and usually requires many fewer neurons (see section 5.4).

5.3 Attractors. After presentation, the visual stimulus is immediately removed, an initial state is produced in layer A due to the feedforward connections, and the net in layer A evolves to a stable state. A random asynchronous updating scheme is used.

The A layer has one attractor for each class, which is concentrated on the original subset selected for the class. However, since no inhibition is present in the system, the self-sustained state can involve a union of such subsets: multiple attractors can emerge simultaneously. Final classification is done again with a majority vote. However, now, after convergence, the state of the net will be "clean," as can be seen from Figure 9, where the residual activity for class 0 on NIST is shown before and after the net reaches a stable state. The residuals are the difference between the number of active neurons in the correct population and the number of neurons active in the most active of all other populations. The histogram is over all data points of class 0.

Table 5: Recognition Rate as a Function of the Number of Neurons for Two Parameter Sets

Neurons	100	200	500	1000	2000	5000
Worst	51	63	69	75	81	85
Best	91	92	93	93	94	94

Note: The parameter sets are based on those of Table 3 but with $k_d = 1$ (Worst) and $p_{IA} = 0.2$ (Best).

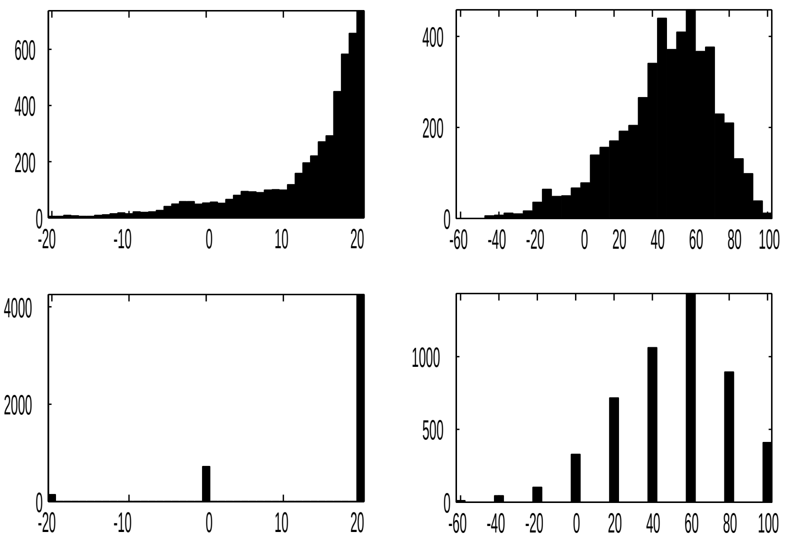

Figure 9: (Left panels) Histogram of the residuals of activity of a network composed of 200 neurons (here for simplicity the subsets are forced to be disjoint) for class 0 on NIST. The upper graph is the preattractor situation, and the lower graph is the postattractor situation. The total number of examples presented for this class was 5000, and the overall performance of the net was 90.1% without attractors and 85.1% with attractors. (Right panels) Same data for five nets produced with boosting. Classification rate 93.7% without attractors and 90.3% with attractors.

Note that most of the examples that generated a positive residual from the feedforward connections converged to the attractor state. This is evident in the high concentration on the peak residual, which is 20 neurons—precisely the size of the population in this example. Those examples with a low initial residual end up with 0 residuals at the stable state, causing a decrease in the classification rate with respect to the preattractor situation—for example, from 90.1% to 85.1% on one net for the NIST data set. Note that a 0 residual is reached when none of the neurons is active at the stable state or because two attractors are activated simultaneously.

The situation with more than one net is somewhat more complicated. In the final state, one might have only the attractors for the right class in all the nets, or all the attractors of some other class, or anything in between. This is in part captured in the right panel of Figure 9, where five nets have been trained with boosting. The histogram is of the aggregated residuals. The final rate of this system is 93.7% without attractors and 90.3% with attractors, showing that the gap between the two is greatly reduced. The reason is that it is improbable that all of the nets have a small residual after

Table 6: NIST Experiments.

a. Parameters

$L = 0$	$H = 120$	$\theta = 100$	$N_A = 200$	Spread 3×3
Iter $= 2.0$	$J_m = 10.0$	$k_p = 0.2$	$k_d = 0.2$	$C_p = 4.0$
$C_d = 1.0$	$p_{IA} = 0.1$	$p_A = 0.1$	$p_{AA} = 1.0$	

b. Results

Number of Nets	1	5	10	20	50
Train rate, no attractors	92.7	95.4	97.2	98.9	99.2
Test rate, no attractors	89.2	92.9	95.5	97.4	97.6
Test rate, with attractors	84.1	89.4	94.2	96.7	97.4

Notes: (a) Parameter set for each net of the experiment on NIST described in section 5.4. The notation used is explained in section 2. (b) Recognition rates as a function of the number of nets on training and on test sets without attractors and on the test set with attractors.

the presentation of a certain stimulus. As the number of nets increases, the gap decreases further; see Table 6.

5.4 NIST. The maximum performance on the NIST data set has been obtained using an ensemble of nets, each with the parameters given in Table 6. Training has been done using the boosting procedure described in section 3.2 to improve the performance of a single net.

The net has been trained on 50,000 examples and tested on another 50,000. In Table 6 we show the recognition rate as a function of the number of nets, on both the training set and the test set. Although performance with attractors is lower than without, the gap between the two tends to decrease with the number of nets.

5.5 Latex. Ultimately one would want to be able to learn a new object sequentially by simply selecting a new random subset for that object and updating the synaptic weights in some fashion, using presentations of examples from the new object and perhaps some refreshing of examples from previously learned objects. Here we test only the scalability of the system: Can hundreds of classes be learned within the same framework of an attractor layer with several thousands of neurons and using the same range of parameters? We work with a data set of 293 synthetically deformed Latex symbols, as in Amit and Geman (1997), there are 50 samples per class in the training set and 32 samples per class in the test set.

In Figure 10 we show the original 293 prototypes alongside a random sample of deformed ones. Below that we show 32 test samples of the 2 class

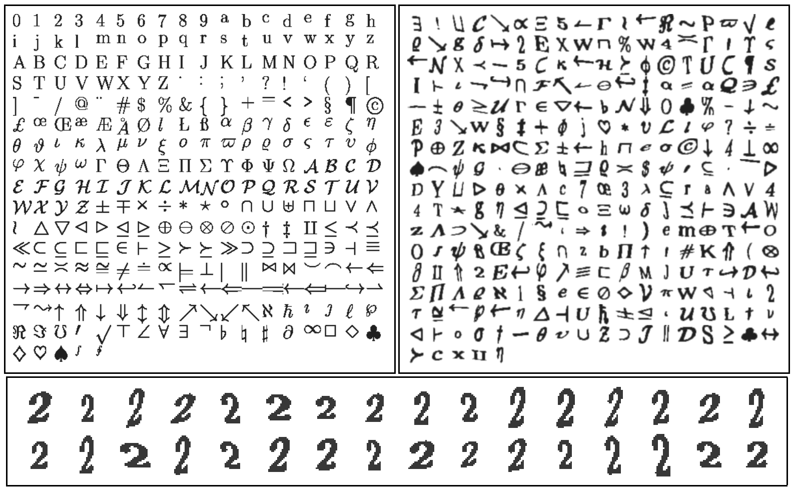

Figure 10: (Left) Prototypes of Latex figures. (Right) Random sample of deformed symbols. (Bottom) Sample of deformed symbols of class 2.

where significant variations are evident. Since the data are not normalized or registered, the spreading of the edge inputs is essential. The images are 32×32 and are more or less centered. The parameters employed in this experiment and the classification rates as a function of the number of nets

Table 7: LaTeX Experiments.

a. Parameters

$L = 0$	$H = 120.0$	$\theta = 100.0$	$N_A = 6000$	Spread 7×7
Iter $= 10.0$	$J_m = 10.0$	$k_p = 0.5$	$k_d = 0.2$	$C_p = 10.0$
$C_d = 1.0$	$p_{IA} = 0.05$	$p_A = 0.01$	$p_{AA} = 1.0$	

b. Results

293 classes, $N_A = 6000$

Num. Nets	1	5	10	15
Train rate, no attractors	60.2	91.0	94.2	94.6
Test rate, no attractors	42.4	74.9	78.6	79.1
Train rate, attractors	52.3	76.6	84.3	88.6
Test rate, attractors	37.7	57.9	63.2	68.5

Notes: (a) Parameters used in the LaTeX experiment. (b) Results with $N_A = 6000$ and $p_A = 0.01$.

are presented in Table 7. Note that the spread parameter is larger due to the larger images.

The main modification with respect to the NIST experiment is in C_p. This has to do with the fact that the ratio $\tilde{p}_i/\tilde{q}_i$ is now much smaller because $P(c) = 0.003$ as opposed to 0.1 in the case of NIST. This is a rather minor adjustment. Indeed, the experiments with the original C_p produce comparable classification rates. We show the outcome for the 293 class problem with up to 15 nets. With one net of 6000 neurons, we achieve 52% before attractors and 37% with attractors. This is definitely a respectable classification rate for one classifier on a 293-class problem. With 15 nets, we achieve 79.1% without attractors and 68.5% with attractors. Note that using multiple decision trees on the same input, we can achieve classification rates of 91%. Although we have not achieved nearly the same rates, it is clear that the system does not collapse in the presence of hundreds of classes. One clear advantage of decision trees over these networks is the use of "negative" information: features that are not expected to be present in a certain region of a character of a specific class. In the terminology of section 4, these are features with very low p and moderate or high q. There is extensive information in such features, which is entirely ignored in the current setting.

6 Discussion

We have shown that a simple system of thousands of perceptrons with coarse-oriented edge features as input is able to recognize images of characters, even in a context with hundreds of classes. The perceptrons have randomized connections to the input layer and among themselves. They are highly constrained as classifiers in that the synaptic efficacies are positive and essentially binary. Moreover, all perceptrons have the same threshold. There is no need for "grandmother cells"; each new class is represented by a small, random subset of the population of perceptrons. Classification manifests itself through a stable state of the dynamics of the network of perceptrons concentrated on the population representing a specific class.

The attractors in the A layer serve three distinct purposes. The first is during learning, where they maintain a stable activity of a specific subset of the A layer corresponding to the class of the presented data point. The second is cleaning up the activity in the A layer after recognition, and the third is maintaining a sustained activity at the correct attractor, thus providing a mechanism for working memory.

Learning is performed using a modified Hebbian rule, which is shown to be very robust to parameter changes and has a simple interpretation in terms of the statistics of the individual input features on and off class. The learning rule is not entirely local on the synapse, although it is local on each neuron, since potentiation and depression are suspended as a function of the field. We believe that in a system with continuous time dynamics and integrate-and-fire neurons, this suspension can depend on the firing rate of the neuron. Indeed in future work, we intend to study the implementation of this system in terms of real-time dynamics.

In the context of this article, boosting has allowed us to improve classification rates. It is very sensible to have a procedure in which more resources are used to train on more difficult examples; however, here this is implemented in an artificial manner. One can imagine boosting evolving naturally in the framework of a predetermined number of several interconnected recurrent A networks, all receiving feedforward input from the I layer. The basic idea would be that if in one network a visual input of class c produces the "correct" activity in the A layer, the field on the units in the A_c subsets in other nets is strong enough so that no potentiation occurs on synapses connecting to these units. On the other hand, if the visual input produces incorrect activity, the local field on the other A_c subsets is smaller, and potentiation of feedforward connections can occur.

It is of interest to develop methods for sequential learning. A new random subset is selected in the attractor layer and synapses are modified using examples of the new object, while maintaining the existing recognition capabilities of the system through some refreshing mechanism. It is also important to investigate the possibility that this learning mechanism, or a variation thereof, can learn correlations among features. Up to this point, features have been selected essentially according to their marginal probabilities. There is abundant information in feature conjunctions, which does not seem to be exploited in the current set-up. Finally, it will be of interest to investigate various feedback mechanisms from the A layer to the input I layer that would produce template-type visual inputs to replace the original noisy input.

Acknowledgments

We thank Daniel Amit, Stefano Fusi, and Paolo Del Giudice for numerous insightful suggestions. Y. A. was supported in part by the Army Research Office under grant DAAH04-96-1-0061 and MURI grant DAAH04-96-1-0445.

References

Amit, D. J. (1989). *Modelling brain function: The world of attractor neural networks*. Cambridge: Cambridge University Press.

Amit, Y. (2000). A neural network architecture for visual selection. *Neural Computation, 12*, 1059–1082.

Amit, Y., Blanchard, G., & Wilder, K. (1999). *Multiple randomized classifiers: MRCL* (Tech. Rep.). Chicago: Deptartment of Statistics, University of Chicago.

Amit, D., & Brunel, N. (1995). Learning internal representations in an attractor neural network with analogue neurons. *Network, 6*, 261.

Amit, D. J., & Fusi, S. (1994). Dynamic learning in neural networks with material synapses. *Neural Computation, 6*, 957.

Amit, Y., & Geman, D. (1997). Shape quantization and recognition with randomized trees. *Neural Computation*, *9*, 1545–1588.

Amit, Y., & Geman, D. (1999). A computational model for visual selection. *Neural Computation*, *11*, 1691–1715.

Amit, Y., Geman, D., & Wilder, K. (1997). Joint induction of shape features and tree classifiers. *IEEE Trans. on Patt. Anal. and Mach. Intel.*, *19*, 1300–1306.

Atkinson, R. (1975). Mnemotechnics in second-language learning. *American Psycholigsts*, *30*, 821–828.

Avimelelch, R., & Intrator, N. (1999). Boosted mixture of experts: An ensemble learning scheme. *Neural Computation*, *11*, 483–497.

Bartlett, M. S., & Sejnowski, T. J. (1998). Learning viewpoint–invariant face representations from visual experience in an attractor network. *Network: Comput. Neural. Syst.*, *9*, 399–417.

Bliss, T. V. P., & Collingridge, G. L. (1993). A synaptic model of memory: Long term potentiation in the hippocampus. *Nature*, *361*, 31.

Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Jackel, L. D., LeCun, Y., Muller, U. A., Sackinger, E., Simard, P., & Vapnik, V. (1994). Comparison of classifier methods: A case study in handwritten digit recognition. In *Proc. IEEE Inter. Conf. on Pattern Recognition* (pp. 77–82).

Breiman, L. (1998). Arcing classifiers (with discussion). *Annals of Statistics*, *26*, 801–849.

Breiman, L. (1999). *Random forests, random features* (Tech. Rep.). Berkeley: University of California, Berkeley.

Brunel, N. (1994). Storage capacity of neural networks: Effect of the fluctuations of the number of active neurons per memory. *J. Phys. A.: Math. Gen.*, *27*, 4783–4789.

Brunel, N., Carusi, F., & Fusi, S. (1998). Slow stochastic Hebbian learning of classes of stimuli in a recurrent neural network. *Network*, *9*, 123–152.

Diederich, S., & Opper, M. (1987). Learning correlated patterns in spin-glass networks by local learning rules. *Phys. Rev. Lett.*, *58*, 949–952.

Dietterich, T. G. (2000). An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting and randomization. *Machine Learning*, *40*, 139–157.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. New York: Wiley.

Fukushima, K. (1986). A neural network model for selective attention in visual pattern recognition. *Biol. Cyber.*, *55*, 5–15.

Fukushima, K., & Wake, N. (1991). Handwritten alphanumeric character recognition by the neocognitron. *IEEE Trans. Neural Networks*, *1*.

Fusi, S., Annunziato, M., Badoni, D., Salamon, A., & Amit, D. J. (1999). *Spike-driven synaptic plasticity: Theory, simulation, VLSI implementation* (Tech. Rep.). Rome: University of Rome, La Sapienza.

Fusi, S., Badoni, D., Salamon, A., & Amit, D. (2000). Spike-driven synaptic plasticity: Theory, simulation, VLSI implementation. *Neural Computation*, *12*, 2305–2329.

Hastie, T., Buja, A., & Tibshirani, R. (1995). Penalized discriminant analysis. *Annals of Statistics*, *23*, 73–103.

Hopfield, J. J. (1982). Neural networks and physical systems with emergent computational abilities. *PNAS*, *79*, 2554.

Hubel, H. D. (1988). *Eye, brain, and vision*. New York: Scientific American Library.

Hussain, B., & Kabuka, M. R. (1994). A novel feature recognition neural network and its application to character recognition. *IEEE Trans. PAMI*, *16*, 99–106.

Knerr, S., Personnaz, L., & Dreyfus, G. (1992). Handwritten digit recognition by neural networks with single-layer training. *IEEE Trans. Neural Networks*, *3*, 962–968.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., & Jackel, L. D. (1990). Handwritten digit recognition with a back-propagation network. In D. S. Touretsky (Ed.), *Advances in neural information*, *2*. San Mateo, CA: Morgan Kaufmann.

Levenex, P., & Schenk, F. (1997). Olfactory cues potentiate learning of distant visuospatial information. *Neurobiology of Learning and Memory*, *68*, 140–153.

Markram, H., Lubke, J., Frotscher, M., & Sakmann, B. (1997). Regulation of synaptic efficacy by coincidence of postsynaptic AP's and EPSP's. *Science*, *375*, 213.

Mattia, M., & Del Giudice, P. (2000). Efficient event-driven simulation of large networks of spiking neurons and dynamical synapses. *Neural Comp.*, *12* 2305–2329.

Miyashita, Y., & Chang, H. S. (1988). Neuronal correlate of pictorial short term memory in the primate temporal cortex. *Nature*, *68*, 331.

Oja, E. (1989). Neural networks, principle components, and subspaces. *International Journal of Neural Systems*, *1*, 62–68.

Petersen, C. C. H., Malenka, R. C., Nocoll, R. A., & Hopfield, J. J. (1998). All-or-none potentiation at CA3-CA1 synapses. *Proc. Natl. Acad. Sci.*, *95*, 4732.

Riesenhuber, M., & Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature Neuroscience*, *2*, 1019–1025.

Rolls, E. T. (2000). Memory systems in the brain. *Annu. Rev. Psychol.*, *51*, 599–630.

Rosenblatt, F. (1962). *Principles of neurodynamics: Perceptrons and the theory of brain mechanisms*. Washington, DC: Spartan.

Sakai, K., & Miyashita, Y. (1991). Neural organization for the long-term memory of paired associates. *Nature*, *354*, 152–155.

Schapire, R. E., Freund, Y., Bartlett, P., & Lee, W. S. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods. *Annals of Statistics*, *26*, 1651–1686.

Wilkinson, R. A., Geist, J., Janet, S., Grother, P., Gurges, C., Creecy, R., Hammond, B., Hull, J., Larsen, N., Vogl, T., & Wilson, C. (1992). *The first census optical character recognition system conference* (Tech. Rep. No. NISTIR 4912). Gaithersburg, MD: National Institute of Standards and Technology.

Received November 9, 1999; accepted September 8, 2000.