

Sparse RNNs can support high-capacity classification

Denis Turcu and L. F. Abbott

The Mortimer B. Zuckerman Mind, Brain and Behavior Institute
 Department of Neuroscience
 Department of Physiology and Cellular Biophysics
 Columbia University
 New York NY 10027 USA

Abstract

Feedforward network models performing classification tasks rely on highly convergent output units that collect the information passed on by preceding layers. Although convergent output-unit like neurons may exist in some biological neural circuits, notably the cerebellar cortex, neocortical circuits do not exhibit any obvious candidates for this role; instead they are highly recurrent. We investigate whether a sparsely connected recurrent neural network (RNN) can perform classification in a distributed manner without ever bringing all of the relevant information to a single convergence site. Our model is based on a sparse RNN that performs classification dynamically. Specifically, the interconnections of the RNN are trained to resonantly amplify the magnitude of responses to some external inputs but not others. The amplified and non-amplified responses then form the basis for binary classification. Furthermore, the network acts as an evidence accumulator and maintains its decision even after the input is turned off. Despite highly sparse connectivity, learned recurrent connections allow input information to flow to every neuron of the RNN, providing the basis for distributed computation. In this arrangement, the minimum number of synapses per neuron required to reach maximum memory capacity scales only logarithmically with network size. The model is robust to various types of noise, works with different activation and loss functions and with both backpropagation- and Hebbian-based learning rules. The RNN can also be constructed with a split excitation-inhibition architecture with little reduction in performance.

Introduction

Binary classification is a basic task that involves dividing stimuli into two groups. Machine-learning solutions to this task typically use single- or multi-layer perceptrons [1] in which, almost invariably (but see [2]), the output that delivers the network’s decision comes from a unit that collects information from all of the units in the previous layer. Collecting all of the evidence in one place (i.e. in one unit) is an essential element in the design of these networks. In humans and other mammals, tasks like this are likely performed by neocortical circuits that have a recurrent rather than feedforward architecture, and where there are no obvious highly convergent ‘output’ neurons. Instead, all of the principal neurons are sparsely connected to a roughly equal degree. This raises the question of whether a network can classify effectively if the information needed for the classification remains dispersed across the network rather than being concentrated at a single site. Here we explore how and how well recurrent networks with sparse connections and no convergent output unit can perform binary classification.

We study sparse RNNs that reach decisions dynamically. Despite their sparse connectivity, these networks are able to compute distributively by propagating information across their units. To add biological realism, we also constrain our sparse RNN to have a split excitation-inhibition architecture. The model maintains high performance despite this constraint. To investigate capacity and accuracy, networks were trained by back-propagation through time (BPTT). With extensive training, these models can categorize up to two input patterns per plastic synapse, matching the proven limit of the perceptron [3]. The model is robust to different types of noise, training methods and activation functions. The number of recurrent connections per neuron needed to reach high performance scales only logarithmically with network size. To investigate biologically plausible learning, we also constructed networks using both one-shot and iterative Hebbian plasticity. Although performance is significantly reduced compared to BPTT, capacity is still proportional to the number of plastic synapses in the RNN.

Results

The Model

We built a sparse RNN [Figure 1A] and evaluated its performance on a typical binary classification task. The RNN consists of N units described by a vector $\mathbf{x}$ that evolves in time according to

$$\tau \frac{d\mathbf{x}}{dt}(t) = -\mathbf{x}(t) + \mathbf{J}^s \phi(\mathbf{x}(t)) + \mathbf{i}(t), \quad (1)$$

where τ is a time constant and $\mathbf{i}$ is the external input being classified. The response function $\phi(\cdot)$ is, in general, nonlinear, monotonic, non-negative and bounded; we use

either a shifted, scaled and rectified hyperbolic tangent (Methods) or a squared rectified linear function [Figure 1B]. Both response functions performed well; we use the modified hyperbolic tangent for all the results we report. Importantly, the connection matrix $\mathbf{J}^s$ is sparse with only fN^2 randomly placed non-zero elements, for $0 < f < 1$. Sparsity is enforced by a mask that also constrains the sparseness during training Methods.

Categorization requires the RNN to correctly match inputs with associated binary labels. Inputs are N -dimensional vectors, called patterns, with each component chosen randomly and independently from a uniform distribution between -1 and 1. The patterns are represented by P N -component vectors ξ^μ with elements ξ_i^μ for $i = 1, 2, \dots, N$ and $\mu = 1, 2, \dots, P$. The label for pattern μ , q^μ , is chosen randomly for each pattern to be either -1 or 1, with equal probability. The category assigned by the RNN is determined by averaging the activity of $f_{\text{out}}N$ randomly chosen units in the RNN, a quantity denoted by $z(t)$ [Figure 1A]. We typically set $f_{\text{out}} = f$, except in Figure 1B. The sparse average over neurons, $z(t)$, measured at a specified time t_{OUT} , reports the decision of the network. Specifically, the category determined by the RNN is defined as -1 if $z(t_{\text{OUT}})$ is less than a threshold θ , and 1 otherwise. The fraction of input-label pairs that are categorized correctly quantifies the accuracy of the sparse RNN, and its capacity is the number of patterns that can be categorized to a given level of accuracy.

Each categorization run starts at time 0 with the initial RNN activity set to zero, $\mathbf{x}(0) = \mathbf{0}$, and $\mathbf{i}$ set to one of the input patterns, $\mathbf{i} = \xi^\mu$ for a randomly chosen μ value. The input remains on at a constant level for a duration t_{ON} and then $\mathbf{i}$ is set to zero. The trial terminates at time t_{F} . After training, the network's decision can be read out at any time $t_{\text{ON}} < t_{\text{OUT}} < t_{\text{F}}$ with minimal effect on performance.

Training was used to adjust only the recurrent connections; the input connections are one-to-one with unit weights and the output connections are sparse and fixed at 0 or $1/(f_{\text{OUT}}N)$ [Figure 1A]. Training did not include noise in the sparse RNN dynamics, but two types of noise were incorporated for testing (see below). For our initial studies, the weights of the connections, given by $\mathbf{J}^s$, were set by backpropagation through time (BPTT), with the goal of generating the correct associated label for each input pattern. This training method can incorporate the sparseness constraints imposed on $\mathbf{J}^s$ and, as discussed later, can also impose a split excitation-inhibition architecture (Methods).

To train the network, label-specific target dynamics T were imposed on the output $z(t)$ at certain times [Figure 1C]. The target dynamics consist of constant target values $T_{+(-)}$ (dashed black curves in [Figure 1C]). The target values for the ± 1 categories satisfy $T_+ > T_-$ for all times. The loss function penalizes the RNN at all times $t_{\text{ON}} < t < t_{\text{F}}$ when the output is below (+1 category) or above (-1 category) the target in proportion to the absolute difference between $z(t)$ and $T_{+(-)}$. This loss function implicitly defines a discrimination threshold $\theta = \frac{T_+ + T_-}{2}$ (dashed green curve in [Figure 1C]) for computing the decision (Methods). The targets were chosen so that the threshold was small, $0.1 \leq \theta \leq 0.6$, to take advantage of the supralinear behavior of the neural response function near 0, which enhanced performance.

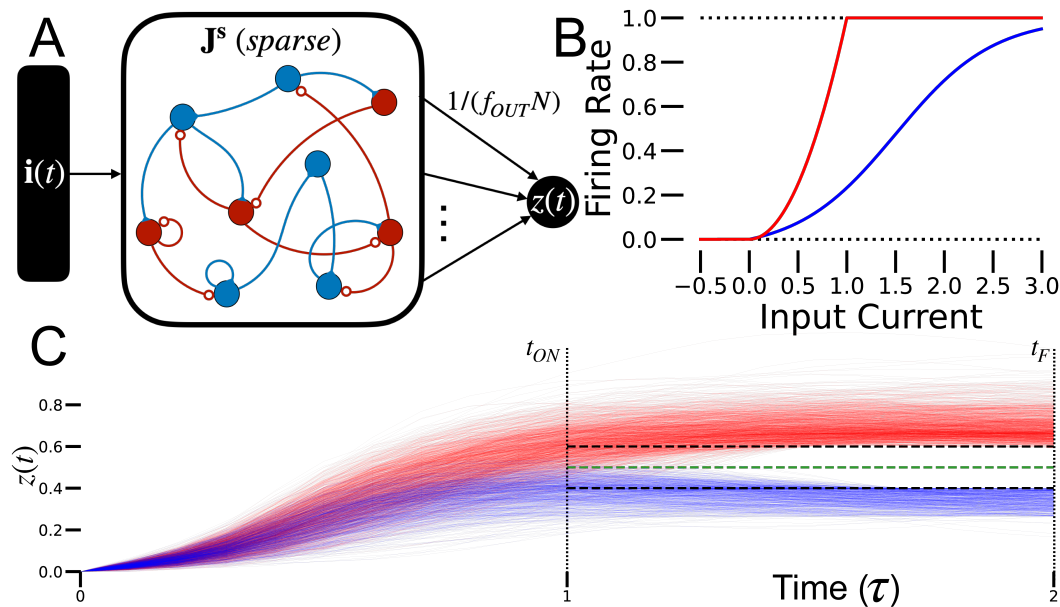


Figure 1: (A) The sparse RNN. Recurrent connections are plastic, but input and output connections are fixed and uniform. We consider both mixed and split excitation-inhibition networks (the split architecture is shown here). (B) Examples of activation functions used. Red line is squarified rectified linear function, bounded by 1. Blue line is a shifted, scaled and rectified hyperbolic tangent function. (C) Illustration of target dynamics training method. Readout activity of trained RNN is shown in red for +1-labeled inputs and in blue for -1-labeled inputs. The corresponding targets T_+ and T_- are shown as dashed black lines. The threshold θ is the dashed green line.

The targets and threshold were constant in time to allow the network to act as an evidence accumulator.

Training length varied depending on the convergence criteria and to maintain reasonable running times. We ran 5000 BPTT epochs for results in [Figure 2D], and typically at most 500 epochs for all other results. We found that performance improvement was minimal beyond 500 epochs, especially for small networks, and in some cases, particularly in which fewer pattern-label pairs were shown, 300, 200 or even 150 epochs sufficed to reach good performance, demonstrate the results and gain negligible improvement from epoch to epoch.

RNNs can be trained to perform categorization across a range of the temporal parameters t_{ON} , t_{OUT} and t_F . Networks performed well for trial durations in the range $\tau < t_F < 15\tau$ with $t_{ON} = t_F/2$ when sampled within the range $3t_F/4$ to t_F . Performance was poor when the trial duration was very small, i.e. $t_F < 0.2\tau$. For the results reported below, we set $t_{ON} = \tau$, $t_F = 2\tau$ and sampled the RNN with $1.5\tau < t_{OUT} < 2\tau$.

Note that the readout is not learned and is as sparsely connected as the units of the RNN. Thus there is no special convergence of information onto the read out. In addition, each RNN unit receives only one component of the input vector and units are sparsely interconnected.

Thus, there is no locus where information converges. Instead, the network units must solve the classification task collectively.

Network Performance

Sparse RNNs trained with BPTT can memorize a number of pattern-label pairs proportional to the number of plastic synapses, $P = \alpha f N^2$ [Figure 2A]. To verify this, we determined the slopes of lines of constant performance on a plot of $\log P$ versus $\log N$ [Figure 2A]. To avoid excessive training time, all networks in Figure 2A were trained for 200 epochs. Even with limited training, the R^2 of the regressions is 0.95 ± 0.03 and the slope is 1.91 ± 0.07 . We observed that training converges faster for smaller networks, which could affect the comparison of networks across many sizes. Removing networks smaller than 115 neurons from this analysis yielded a similar regressions value, $R^2 = 0.93 \pm 0.03$ and a slope of 2.08 ± 0.05 . These results support that $P \sim N^2$, that is, sufficiently large sparse RNNs can memorize a number of pattern-label pairs proportional to the square of their neuron numbers or to the first power of their plastic synapse counts.

For given values of α and f , the categorization accuracy for $P = \alpha f N^2$ patterns is independent of network size, for large N [Figure 2B]. Recall that the network output is obtained by averaging the activity of $f_{\text{out}} N$ units. This level of output sparsity does not impair performance as long as f_{out} is comparable to f [Figure 2C]. We use $f_{\text{out}} = f$ for all further results.

Well trained sparse RNNs reach high memory capacity and do not manifest "blackout catastrophe". Often they can perfectly memorize one input-label pair for each plastic synapse ($\alpha = 1$) if trained for a sufficient time. Alternatively, they can classify up to two patterns for each plastic synapse with accuracy $> 90\%$ [Figure 2D]. Accuracy decreases as α increases, but the decrease in accuracy is gradual, even for $\alpha > 2$. Thus, sparse RNNs do not completely forget previously learned patterns when pushed above a critical value of α . Instead, their performance decreases gradually for increasing numbers of patterns.

Minimum sparseness

Sparse RNNs categorize by distributing the computation across all the elements of the network. Because all the information is not brought together at a single locus, that information must flow freely through the network to generate a correct categorization. The importance of information propagation is evidenced by the reduced performance of RNNs at extreme sparsity levels [Figure 3A].

Signal propagation in networks can be characterized by their adjacency graphs. Randomly generated undirected graphs are disconnected at extreme sparsity levels according to the Erdős-Rényi model [4]. However, as in biological networks, our sparse RNNs are based

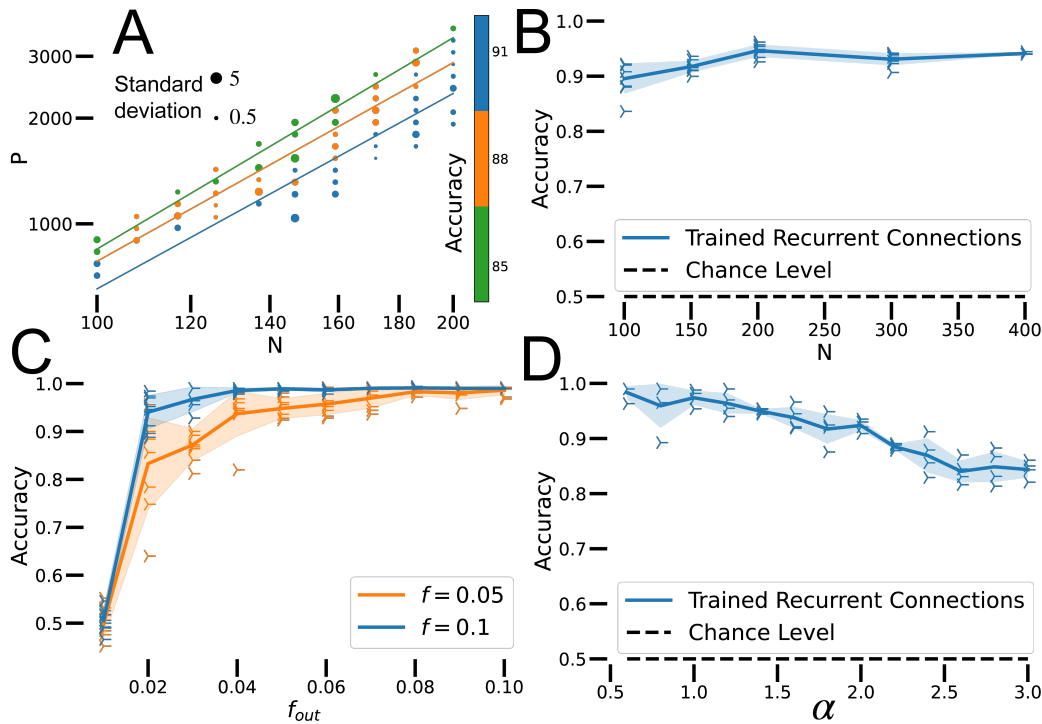


Figure 2: (A) Accuracy for various combinations of N and P . Colors represent accuracy, grouped in bins spanning 3%. Lines are fitted to every group of binned accuracy values. $f = 0.1$, # epochs = 200, $n = 4$. (B) Binary classification performance of sparse RNNs of various sizes. Each RNN classifies $P = \alpha f N^2$ inputs. $f = 0.1$, $\alpha = 1$, # epochs = 500, $n = 4$. (C) Sparse RNN performance as a function of the readout sparsity for RNNs with two different levels of recurrent sparsity. $N = 100$, $\alpha = 0.5$, # epochs = 500, $n = 6$. (D) Accuracy of sparse RNNs as a function of $\alpha = P/(fN^2)$. Runs were truncated at 99% accuracy. $N = 100$, $f = 0.05$, # epochs = 5000, $n = 4$.

on directed graphs, meaning we cannot fully apply the analytic results developed by [4]. A directed graph of size N with connection sparsity f consists of a large strongly connected sub-graph of size $N - k$ and k other nodes with some probability $P_k^{N,f}$ (Methods). We assumed the main sub-graph of size $N - k$ performs the task on all $\alpha f N^2$ inputs without help or interference from the other k nodes, and computed the expected accuracy of the sparse RNNs estimating $P_k^{N,f}$ empirically. We also assumed that the network correctly classifies $\alpha f (N - k)^2 (N - k) / N$ inputs and performs at chance level on the rest, and we computed the estimated accuracy of a network of size N with only $N - k$ functional units, A_k^N . Thus, the expected accuracy of the sparse RNN with N neurons and sparsity f is $\sum_{k=0}^{N-1} A_k^N P_k^{f,N}$, shown as dashed lines in Figure 3A.

The performance of our models roughly fits the predictions based on the empirical analysis of random directed graphs [Figure 3A]. In particular, the shape of the performance drop qualitatively matches our expectations and the location of the initiation of the performance drop appears to shift logarithmically with network size, as expected from our empirical

analysis of directed graphs. Discrepancies between simulations and empirical analysis predictions are apparent for large network sizes. One reason for the increasing discrepancy with network size may lie in the difficulty of training extremely sparse large networks, which require more training epochs. Running time per epoch scales like N^4 in our task, since both the number of plastic synapses and the number of inputs scale like N^2 , making it very difficult to train large networks for many more epochs. Another reason for the discrepancy may lie in our assumptions in computing A_k^N . We assumed that the k separated nodes would not interfere with network decisions. Though we accounted for the likelihood that not all fN readout neurons will be part of the main connected sub-graph through the $(N - k)/N$ factor from A_k^N , we suspect that reading out from one of the k separated neurons introduces further noise and impairs performance. This effect is intensified at extreme sparsity values, where large networks still perform above chance level.

This evidence suggests that it is sufficient for our sparse RNNs to have a strongly connected adjacency graph to perform well. For undirected graphs, Erdős-Rényi results provide a basis for analytically computing the number of connections each node needs for a graph to be connected with probability near 1 [4]. Specifically, a random graph of size N with fN^2 connections is fully connected with probability $\exp(-\exp(-2fN + \log N))$, in the limit of large graph sizes. If we set that probability to $1 - \varepsilon$ for $\varepsilon \ll 1$, the average number of synapses per neuron is $fN^2/N = fN = \log(N)/2 - \log[\log(1/(1 - \varepsilon))]$, which scales as $\log N$ for fixed ε . Moreover, the distribution of the number of synapses per neuron follows a binomial distribution, according to the definition of the adjacency graph. For directed graphs, such as our sparse RNN model, we observe the same logarithmic scaling in our empirical analysis of their adjacency graphs.

To test this hypothesis, we simulated sparse RNNs of various sizes N and having the extreme sparsity level $f = \log(N)/N$. We generated the sparse RNNs randomly and rejected all networks whose strongly connected component was smaller than $0.95N$. We found that these networks performed the task well and at a constant level across network size under the same training conditions [Figure 3A]. The line fit to these data points has a slope of $0.00003 \approx 0$. The results summarized in Figures 3A,B support the need for collective decision making across multiple units.

The Effects of Noise

We tested the robustness of sparse RNNs to two types of noise, injected after training. ‘Input noise’ is added to the input of the RNN whenever $\mathbf{i}(t) \neq 0$. This noise changes for every trial but remains constant once the input is on in each trial. ‘Dynamic noise’ is added to each RNN unit at all times during each trial and changes at every time step. Both noise sources are generated randomly and independently for each unit (and time for dynamic noise) from a Gaussian distribution. Trained sparse RNNs of various sizes, sparsities and capacities maintain high performance over a wide range of noise values [Figure 3C]. Larger

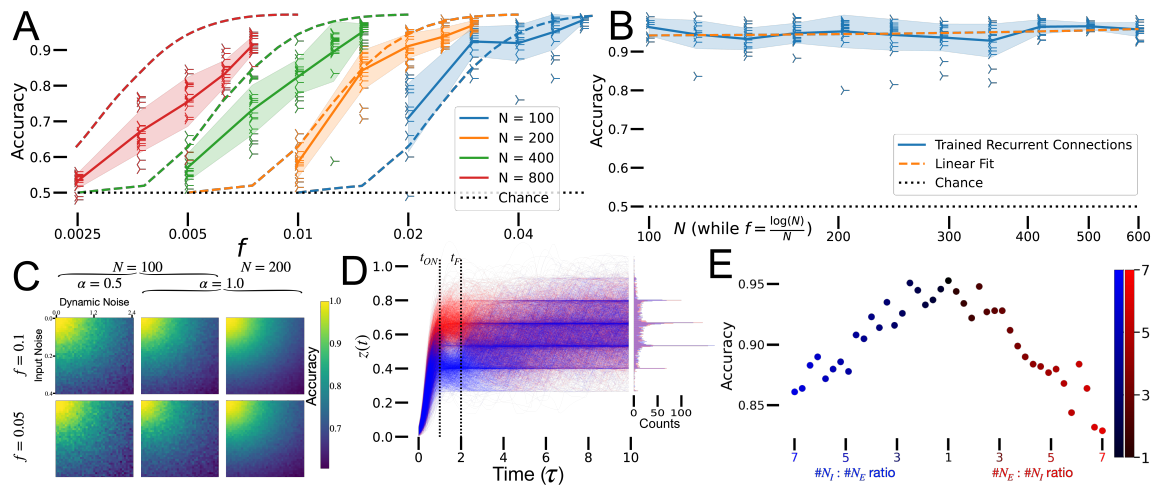


Figure 3: (A) Performance of sparse RNNs at extreme sparsity levels as a function of model size (scatter points; solid lines are averages and shaded areas are standard deviations). Predictions based on empirical analysis of random directed graphs (dashed lines). $\alpha = 0.5$, # epochs = 500, $n = 16$. (B) Performance of nearly fully connected sparse RNNs at the extreme sparsity level $f = \log(N)/N$ for various networks sizes – the size of the strongly connected component is at least $0.95N$ (blue scatter points; blue solid lines are averages and blue shaded areas are standard deviations). Linear fit of all the points is shown as orange dashed line. $\alpha = 0.5$, # epochs = 1000, $n = 16$. (C) Noise robustness heat maps for sparse RNNs. The horizontal axis indicates the level of dynamic noise, defined as the standard deviation of the dynamic noise divided by the mean of the activity in the RNN, and the vertical axis indicates the standard deviation of input noise. Sparse RNNs maintain high accuracy over a wide range of noises levels. # epochs = 150. (D) Output of an example sparse RNN over time. Model is trained for 2τ , and input is on for τ . $N = 300$, $f = 0.05$, $\alpha = 1$, # epochs = 500. (E) Performance of the E/I sparse RNNs as a function of the E/I ratio. $N = N_E + N_I = 100$, $f = 0.1$, $\alpha = 1$, # epochs = 200.

sparse RNNs respond to noise more predictably than smaller models (note the smoother heat maps). Sparse RNNs trained to memorize fewer input patterns are more noise resistant (note the yellow color covers a larger area in Figure 3C).

Output Dynamics

We examined the dynamics of network outputs before, during and after the time of readout [Figure 3D]. BPTT-trained sparse RNNs develop their decisions incrementally in time, acting as evidence accumulators, and they maintain their decisions for a period of time following the readout time. Initially, the models integrate input information while pattern input is present, allowing the sparse RNN to separate into two label-dependent output distributions. After the input is turned off, the model maintains the decision, typically for the same duration as it was trained. During this time, the output distributions remain separated. At longer times, the distributions slowly mix and discrimination between the two categories is lost. The sparse RNNs do not construct or reach two fixed points that correspond to the two labels [5]. Instead, although the activity is sustained for a while,

	$\mathcal{L}^{\text{targets}}$	$\mathcal{L}^{\text{threshold}}_{2\tau}$	$\mathcal{L}^{\text{threshold}}_{\tau}$	$\mathcal{L}^{\text{threshold}}_{0.1\tau}$
Modified tanh(x)	100 $\pm$ 0.0	97.3 $\pm$ 0.4	98.2 $\pm$ 0.4	96.3 $\pm$ 0.1
Bounded ReLU(x)	99.9 $\pm$ 0.1	97.9 $\pm$ 0.1	98.7 $\pm$ 0.1	94.3 $\pm$ 0.7
Bounded ReLU(x^2)	100 $\pm$ 0.0	96.7 $\pm$ 0.9	97.4 $\pm$ 0.8	96.3 $\pm$ 0.7
Bounded ReLU(x^3)	96.9 $\pm$ 0.3	97.4 $\pm$ 0.4	95.9 $\pm$ 0.5	87.3 $\pm$ 2.3

Table 1: Accuracy for combinations of response nonlinearities and loss functions. The hyperbolic tangent response non-linearity was shifted, rectified and scaled to 1. The polynomial response nonlinearities were rectified and bounded by 1. The subscript on the threshold loss function indicates the amount of time the network was trained on. Best results are in green and worst in red. $N = 100$, $f = 0.1$, $\alpha = 1$, # epochs = 300, $n = 8$.

eventually the separability of the input classes is lost [Figure 3D, histogram].

E-I Architecture

In cortical networks, excitatory and inhibitory neurons separately provide positive and negative inputs to each neuron. A split excitation-inhibition architecture can be enforced on sparse RNNs during training by constraining the connections matrix $\mathbf{J}^s$ such that elements in individual columns of the sparse matrix all have the same sign. The E or I identity is assigned randomly for each column, with possible bias towards one of the two identities. In our investigations, the ratio between the number of E neurons, N_E , and the number of I neurons, N_I , where $N_E + N_I = N$, took a range of values.

Split excitation-inhibition architecture does not significantly impair the performance of sparse RNNs performing classification [Figure 3E]. A balanced ratio of 1 between the number of excitatory and inhibitory neurons works best and achieves performance similar to the unconstrained architecture [compare to Figure 2D]. Interestingly, a ratio skewed towards more excitatory neurons appears to be more detrimental than a bias in the inhibitory direction.

Effect of response nonlinearity and loss function

Sparse RNNs can categorize using different activation and loss functions, although certain of these are better than others [Table 1]. The activation functions that provided best results had a slight supralinear behavior for small and positive inputs and were bounded by above and below. The supralinear behavior helps separate the output distributions more rapidly by keeping low activity low while pushing high activity disproportionately higher.

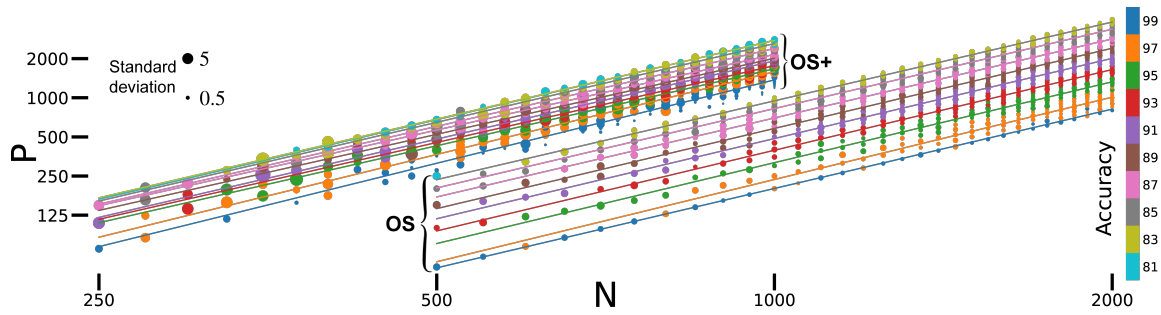


Figure 4: Accuracy for OS and OS+ methods at various combinations of N and P . Colors represent accuracy, grouped in bins spanning 2%. Lines are fitted to every group of binned accuracies, with all slopes > 2 and regression $r > 0.98$. Size of the scatter points is proportional to the standard deviation (inset). $f = 0.1$. OS: $n = 10$. OS+: $n = 5, \epsilon = 0.02, \# \text{ epochs} = 150$.

In addition to the loss function $\mathcal{L}^{\text{targets}}$ described above, we trained networks with a loss $\mathcal{L}^{\text{threshold}}$ that depends on the quantity $-q^\mu(z(t) - \theta)$ summed from time $t_F - \Delta t$ to time t_F . A nonlinear mapping assured that positive values, i.e. categorization miss-matches, contributed large penalties to the loss, whereas negative values contributed little (Methods). This loss function worked well, but produced accuracy values about 5% worse than the results we report using the previously described loss [Table 1].

Hebbian Learning

Although it is useful for constructing networks for study, BPTT is not a biologically realistic way to train networks to perform categorization. It is possible to construct sparse RNNs that perform binary classification, albeit with less capacity, using a closed expression for the connection matrix $\mathbf{J}^s$. This connection matrix can be interpreted as the result of sequentially applying a Hebbian rule to the connections of the sparse RNN learning the patterns, with one exposure each. For this reason, we called it one-shot (OS). Specifically, we set the elements of $\mathbf{J}^s$ to

$$J_{ij} = \frac{g}{\sqrt{NP\sqrt{f}}} M_{ij} \sum_{\mu=1}^P q^\mu \xi_i^\mu \xi_j^\mu, \quad (2)$$

where $\mathbf{M}$, with elements equal to either 0 or 1, is the sparsity mask matrix and g is set to the value 5.62 to optimize performance. We found this value of g by using a grid search and chose the value that performed best.

Sparse RNNs initialized using the OS method memorize a number of pattern-label pairs proportional to the number of plastic synapses, $P = \alpha_{\text{OS}} f N^2$, similarly to BPTT-trained sparse RNNs, except $\alpha_{\text{OS}} \approx \alpha_{\text{BPTT}}/1000 \approx 0.001$ [Figure 4OS]. We show the performance of many sparse RNNs of various sizes being presented different numbers of pattern-label pairs. We divided the achieved accuracy into bins spanning 2% and fit lines to all networks

that fall in one bin. The R^2 of the 9 line fits is 0.9936 ± 0.0050 . The average slope of those lines is 2.047 ± 0.030 . These results suggest that the number of pattern-label pairs memorized up to a given accuracy by the sparse RNNs scales proportionally with the number of plastic synapses, in particular $P \sim N^2$.

We also considered an extension from OS to OS+ that includes plasticity of the recurrent connections as the sparse RNN reacts to input patterns multiple times. This plasticity is a form of gated Hebbian synaptic modification; if pattern μ is categorized incorrectly, a term $\epsilon q^\mu \xi_i^\mu \xi_j^\mu$ is added to $\mathbf{J}^s$, with ϵ the learning rate. This makes small corrections to the output of the sparse RNN for incorrectly categorized patterns without significantly interfering with correctly matched patterns. Allowing ϵ to decrease with epoch number enhances performance.

Sparse RNNs trained using the OS+ method improve the capacity of OS-initialized networks and maintain the same asymptotic performance [Figure 4OS+]. We find that $\alpha_{\text{OS+}} \approx 10\alpha_{\text{OS}} \approx 0.01$, still 100 times smaller than the performance of BPTT-trained networks. We binned the results and fitted lines as for the OS method and found the R^2 of the 10 line fits to be 0.993 ± 0.004 and the slope 2.00 ± 0.07 . These results were obtained by training sparse RNNs for 150 epochs using the OS+ method. We found that when perfect accuracy was not achieved within 150 epochs, forgetting of previous patterns emerged and accuracy dropped close to chance level within 500 – 1000 epochs.

Discussion

We presented a biologically plausible neural network architecture that solves the binary classification task with high capacity. This architecture is based on a sparse RNN that solves the task dynamically. Our main purpose was to show that categorization can be achieved in a truly distributed way without the convergence of information onto any single locus. Sparse RNNs can categorize roughly two patterns per plastic synapse, matching classic perceptron performance [3]. These networks are robust to various types of noise and across training methods and activation functions. Our approach supports separate populations of excitatory and inhibitory neurons, and the resulting E/I networks perform well.

The performance of sparse RNNs for categorization, with scaling of the number of patterns proportional to the number of synapses is in keeping with results from other network studies. In sparse Hopfield-type models, the number of stored bits scales with the number of synapses [6]. In Hopfield-style models of recognition memory, the number of patterns than can be identified as familiar or novel scales with the number of synapses [7], a result that also holds for feedforward networks [8] and for various plasticity rules [8,9].

Categorization by sparse networks has been considered previously by Kushnir and Fusi [2], who used a committee-machine-like readout on a recurrent network with a fixed number

of recurrent connections per unit. Fixed here means a number of connections per neuron that was independent of the number of neurons (N), as opposed to the case we studied with sparse connections proportional to N . In addition, in their study [2], recurrent connections were not learned. Nevertheless, Kushnir and Fusi showed that the recurrent connections play a crucial role in maintaining classification accuracy with sparse readouts. They proved that their model can classify a number of inputs proportional to the number of plastic synapses, as reported by other studies [3, 10, 11] and in ours. Because of the fixed number of connections per neuron, in the regime of large numbers of neurons, their RNN eventually becomes disconnected.

Our results suggest that it is possible to generate decisions in a dynamic and distributed manner in RNNs. This is almost certainly closer than perceptron models to how the bulk of decisions made by biological networks are computed. We suggest the following model: motor or premotor circuits are held in a state of readiness during a go/nogo task, but are not activated until the decision is made. Relevant information is conveyed to the neurons in this circuit, much like the patterns are conveyed to the RNNs we studied. If these inputs are appropriate for a nogo decision, the motor/premotor circuit may be perturbed, but it does not make a transition to a fully activated state. This is analogous to the blue curves in Figure 1C. If, on the other hand, the evidence favors action, the motor/premotor circuit reacts more strongly, analogous to the red curves in Figure 1C, and the motor action is initiated.

Acknowledgements

Research supported by NSF NeuroNex Award DBI-1707398, the Gatsby Charitable Foundation, and Boehringer Ingelheim Fonds.

References

- [1] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408, 1958.
- [2] Lyudmila Kushnir and Stefano Fusi. Neural classifiers with limited connectivity and recurrent readouts. *Journal of Neuroscience*, 38(46):9900–9924, 2018.
- [3] Thomas M. Cover. Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition. *IEEE Transactions on Electronic Computers*, EC-14(3):326–334, 1965.
- [4] P Erdős and A Rényi. On random graphs I. *Publicationes Mathematicae*, 6:290–297, 1959.

- [5] Xiao Jing Wang. Decision Making in Recurrent Neuronal Circuits. *Neuron*, 60(2):215–234, oct 2008.
- [6] Matthias Löwe and Franck Vermet. The Hopfield Model on a Sparse Erdős-Renyi Graph. *Journal of Statistical Physics* 2011 143:1, 143(1):205–214, mar 2011.
- [7] Rafal Bogacz, Malcolm W Brown, and Christophe Giraud-Carrier. Model of Familiarity Discrimination in the Perirhinal Cortex. *Journal of Computational Neuroscience*, 10:5–23, 2001.
- [8] Danil Tyulmankov, Guangyu Robert Yang, and L.F. Abbott. Meta-learning synaptic plasticity and memory addressing for continual familiarity detection. *Neuron*, dec 2021.
- [9] Rafal Bogacz, Malcolm W Brown, and Christophe Giraud-Carrier. Model of Familiarity Discrimination in the Perirhinal Cortex. *Journal of Computational Neuroscience*, 10:5–23, 2001.
- [10] Nicolas Brunel, Vincent Hakim, Philippe Isope, Jean Pierre Nadal, and Boris Barbour. Optimal information storage and the distribution of synaptic weights: Perceptron versus Purkinje cell. *Neuron*, 43(5):745–757, sep 2004.
- [11] Jasmine Collins, Jascha Sohl-Dickstein, and David Sussillo. Capacity and trainability in recurrent neural networks. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, pages 1–17, 2017.
- [12] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

Methods

BPTT Simulations We simulated sparse RNNs with discrete time steps according to the dynamics in Equation 1. For all BPTT simulations we used custom RNN architectures and code developed in Python using PyTorch [12]. Code is available at github.com/DenisTurcu/SparseRNN.

Response function The shifted, scaled and rectified hyperbolic tangent response function we used for all reported results is $\max[(\tanh(x + x_0) - \tanh(x_0))/(1 - \tanh(x_0)), 0]$, where x_0 represents

the shift amount. We used $x_0 < 0$, typically $x_0 = -0.5$, such that the derivative at small positive values is supralinear.

Sparsity Mask We imposed a sparsity mask $\mathbf{M}$ on the recurrent connections $\mathbf{J}$ such that the effective recurrent connections $\mathbf{J}^s \equiv \mathbf{J} \odot \mathbf{M}$ are sparse. The sparsity mask $\mathbf{M}$ is an $N \times N$ matrix with elements equal to either 0 or 1 and $\sum_{i,j} \mathbf{M}_{ij} = fN^2$. All the 1s are randomly placed in $\mathbf{M}$. This ensures that only fN^2 of the N^2 recurrent connections are used by the sparse RNN.

Decision Threshold The simulated sparse RNNs are judged based on their output being above or below a certain threshold. For all results reported, except for part of [Table 1](#), we trained the networks using a label-specific dynamic target. The targets for the ± 1 categories [[Figure 1C](#)] implicitly define a discrimination threshold $\theta = \frac{T_+ + T_-}{2}$. We chose the targets to start at time t_{ON} such that the sparse RNNs process their input while it is presented and then make a decision between t_{ON} and t_F . The targets are constant at all times $t_{ON} < t < t_F$, such that the sparse RNNs are required to maintain their decision for some amount of time and act as evidence accumulators.

Split E-I Architecture Excitation-inhibition constraints were imposed at every gradient step during training for all simulations with such restrictions. The constraints ensure that all elements of any given column of $\mathbf{J}^s$ have the same pre-assigned sign. We defined a vector $\mathbf{v}$ of size N with elements equal to either 1, for excitatory, or -1 , for inhibitory identity. We defined N_E as the number of 1s and N_I as the number of -1 s in $\mathbf{v}$ such that $N_E + N_I = N$. We updated the effective recurrent connections after each optimization step according to $\mathbf{J}_{ij}^s \leftarrow \mathbf{J}_{ij}^s \text{sgn}(\mathbf{J}_{ij}^s) \mathbf{v}_j$, which ensures that all recurrent connections going out from neuron j have the same identity as $\mathbf{v}_j$.

Noise Description No noise was introduced during training the sparse RNNs, yet they are robust to noise at test time. We presented two types of noise after training the sparse RNNs, input noise and dynamic noise. Input noise changes the input pattern presented to sparse RNN. Dynamic noise alters the recurrent state of the sparse RNN at each time step. We incorporated these noise types into [Equation 1](#):

$$\tau \frac{d\mathbf{x}}{dt}(t) = -\mathbf{x}(t) + \mathbf{J}^s \phi(\mathbf{x}(t)) + [\mathbf{i}(t) + \mathbf{z}_{INPUT} \Theta(t_{ON} - t)] + \mathbf{z}_{DYNAMIC}(t), \quad (3)$$

where $\mathbf{z}_{INPUT}$ is Gaussian, fixed for every trial and turns off with the input at t_{ON} as noted by the step-function, and $\mathbf{z}_{DYNAMIC}(t)$ is Gaussian and changes independently at every time step.

Loss Functions The primary loss function we used for all reported results, except part of [Table 1](#), was the loss function based on label-specific dynamic targets. This targets-based loss function penalizes the sparse RNNs when their output is below (for $+1$ category) or above (for -1 category) the respective target, in proportion to the absolute value difference between $z(t)$ and $T_{+(-)}$. To avoid instructing the sparse RNNs how to perform the task via this target-based training, we also trained networks using a threshold-based loss function. For this loss function, we set the discrimination threshold, θ , constant in time, similar to before. The threshold-based loss function depends on a nonlinear mapping applied to the quantity $-q^\mu(z(t) - \theta)$ and summed from time $t_F - \Delta t$ to time t_F . This quantity is positive at a time step when there is a categorization miss-match, e.g. $z(t) < \theta$ but $q^\mu = +1$, and negative otherwise. Therefore, we used a nonlinear mapping which is large for positive values, and small for negative values. Specifically, this mapping was $\max[x\beta, 0] - \tanh(\max[-x\beta, 0])$ for some slope parameter β , typically set to 1.

BPTT Speed Up BPTT training was sped up with the introduction of “PyTorch’s sparse” module. We started using this module once it reached a stable development version and operations on sparse tensors were properly differentiable. This work replaces our version of using a sparse recurrent connections matrix via a sparsity mask and we used for all results reported, except from [Figure 3D] and Table 1. Code is available at github.com/DenisTurcu/SparseRNN.

Hebbian Learning All Hebbian learning results reported were generated using custom Matlab code (Mathworks, Natick, MA). Code is available at github.com/DenisTurcu/SparseRNN

Directed Graph Percolation We estimated the probability that a randomly generated directed graph with N nodes and connection probability f from node i to node j consists of a strongly connected sub-graph of size $N - k$ and k other nodes empirically. We call this probability $P_k^{N,f}$. We ran these simulations in Python using Kosaraju’s algorithm and adapted the implementation of the algorithm by Neelam Yadav. Code is available at github.com/DenisTurcu/SparseRNN