PLOS COMPUTATIONAL BIOLOGY

RESEARCH ARTICLE

Sparse RNNs can support high-capacity classification

Denis Turcu*, L. F. Abbott

The Mortimer B. Zuckerman Mind, Brain and Behavior Institute, Department of Neuroscience, Columbia University, New York, New York, United States of America

* dt2626@cumc.columbia.edu

Citation: Turcu D, Abbott LF (2022) Sparse RNNs can support high-capacity classification. PLoS Comput Biol 18(12): e1010759. https://doi.org/10.1371/journal.pcbi.1010759

Editor: Alireza Soltani, Dartmouth College, UNITED STATES

Received: May 20, 2022

Accepted: November 24, 2022

Published: December 14, 2022

Data Availability Statement: The authors confirm that all data underlying the findings are fully available without restriction. All relevant data are within the paper and its Supporting information files. Code is available at github.com/DenisTurcu/SparseRNN.

Funding: DT and LFA are supported by NSF NeuroNex Award DBI-1707398 and the Gatsby Charitable Foundation GAT3708. DT is additionally supported by Boehringer Ingelheim Fonds. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

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.

Author summary

Binary classification is a decision task that requires splitting stimuli into two groups. Animals perform many such decisions on a daily basis, but the neural mechanisms of these computations are not well understood. In this work, we present a biologically plausible computational mechanism that can divide large numbers of stimuli into two groups. In standard computational models of this task, all information flows in a single direction and converges onto a single site, which does not match biological architectures. In humans and other mammals, tasks such as binary classification are likely performed by neocortical circuits that process the evidence recurrently and have no 'readout' neurons where information converges. Our computational model accumulates evidence when a stimulus is

Competing interests: The authors have declared that no competing interests exist.

present, makes a decision based on that accumulated evidence without convergence, and maintains the decision for a short period of time after it has been made. This demonstrates that it is possible to generate decisions in a dynamic and distributed manner closer to how decisions are made by biological circuits.

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 [Fig 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

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

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 (Materials and methods) or a squared rectified linear function [Fig 1B]. Both response functions performed well; we use the modified hyperbolic tangent for all the results we report. Importantly, the connection matrix $\mathbf{J}^{\mathbf{s}}$ is sparse with only

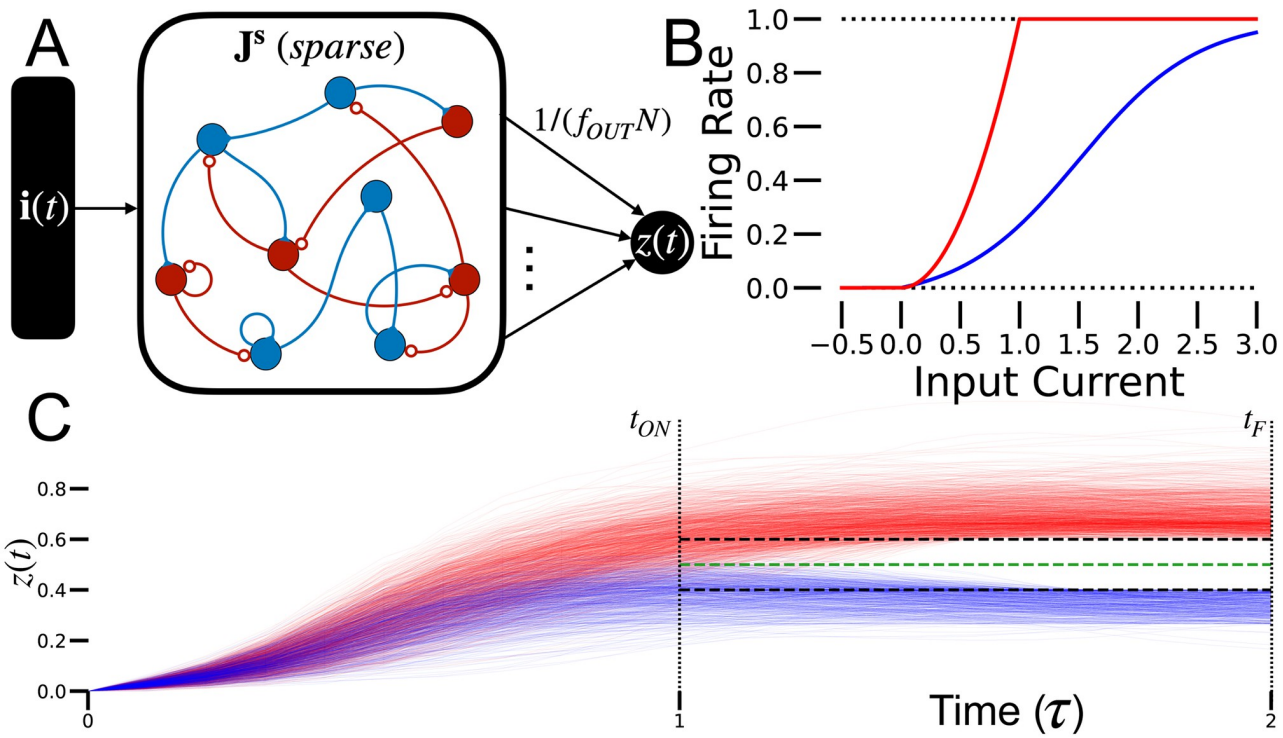

Fig 1. Model description. A: The sparse RNN. Recurrent connections are plastic, but input and output connections are fixed and uniform. We consider both mixed and split excitation-inhibition networks (the split architecture is shown here). B: Examples of activation functions used. Red line is squared 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.

https://doi.org/10.1371/journal.pcbi.1010759.g001

fN^2 randomly placed non-zero elements, for $0 < f < 1$. Sparsity is enforced by a mask that also constrains the sparseness during training (Materials and 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 $\boldsymbol{\xi}^\mu$ with elements ξ_i^μ for $i = 1, 2, \ldots, N$ and $\mu = 1, 2, \ldots, P$. 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 $z(t)$ [Fig 1A]. We typically set $f_{\text{out}} = f$, except in Fig 2C. 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, $\mathbf{i} = \boldsymbol{\xi}^\mu$ 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 $t_{\text{OUT}} < t_{\text{ON}} < t_{\text{F}}$ 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 $1/f_{\text{OUT}}N$

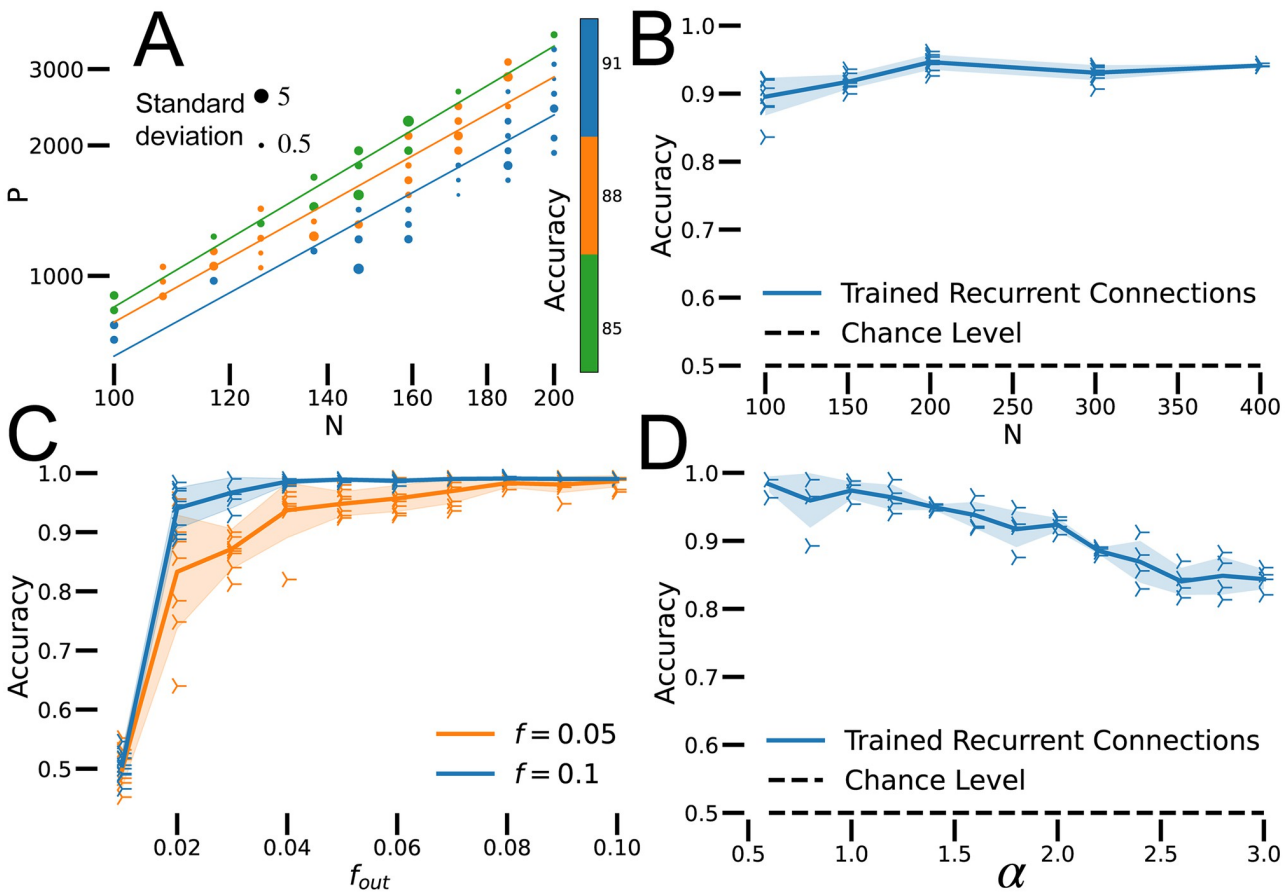

Fig 2. Performance. A: Accuracy for various combinations of N and P. Colors represent accuracy, grouped in bins spanning 3%. Lines are fitted to every group of binned accuracy values. $f = 0.1$, # epochs = 200 and number of trained networks (n): $n = 4$. B: Binary classification performance of sparse 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, $n = 4$.

https://doi.org/10.1371/journal.pcbi.1010759.g002

[Fig 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 $\mathbf{J}^{\mathbf{s}}$, 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 $\mathbf{J}^{\mathbf{s}}$ and, as discussed later, can also impose a split excitation-inhibition architecture (Materials and methods).

To train the network, label-specific target dynamics T were imposed on the output $z(t)$ at certain times [Fig 1C]. The target dynamics consist of constant target values $T_{+(-)}$ (dashed black curves in [Fig 1C]). The target values for the ±1 categories satisfy $T_+ > T_-$ for all times. The loss function penalizes the RNN at all times $t_{\mathrm{ON}} < t < t_{\mathrm{F}}$ 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 [Fig 1C]) for computing the decision (Materials and 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. 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 Fig 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, $P = \alpha f N^2$ [Fig 2A]. To verify this, we determined the slopes of lines of constant performance on a plot of log P versus log N [Fig 2A]. To avoid excessive training time, all networks in Fig 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 [Fig 2B]. Recall that the network output is obtained by averaging the activity of $f_{out}N$ units. This level of output sparsity does not impair performance as long as f_{out} is comparable to f [Fig 2C]. We use $f_{out} = f$ for all further results.

Well trained sparse RNNs reach high memory capacity and do not manifest "blackout catastrophe" [4]. 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% [Fig 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 [Fig 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 Erdös-Rényi model [5]. However, as in biological networks, our sparse RNNs are based on

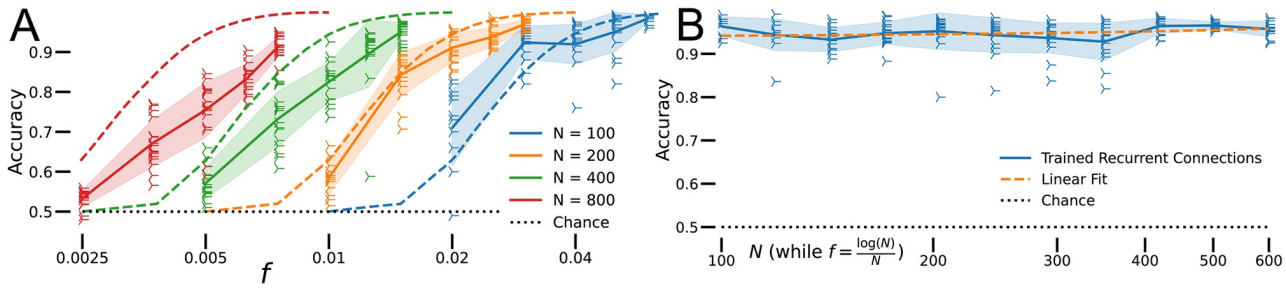

Fig 3. Extreme sparsity. A: Performance of sparse RNNs at extreme sparsity levels as a function of model size (scatter points; solid lines are averages and shaded areas are standard deviations). Predictions based on empirical analysis of random directed graphs (dashed lines). $\alpha = 0.5$, # epochs = 500, $n = 16$. B: Performance of nearly strongly connected sparse RNNs at the extreme sparsity level $f = \log(N)/N$ for various networks sizes—the size of the strongly connected sub-network is at least $0.95N$ for each simulation shown (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, $n = 16$.

https://doi.org/10.1371/journal.pcbi.1010759.g003

directed graphs, meaning we cannot fully apply the analytic results developed by [5]. 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 $P_k^{N,f}$ (Materials and methods). For directed graphs, "strongly connected" means that there is a directed path between any two nodes. 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 $\alpha f(N-k)^2 \cdot (N-k)/N$ 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 (Materials and methods). 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 Fig 3A.

For undirected graphs, Erdös-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 [5]. Specifically, a random graph of size N with fN^2 connections is connected with probability $\exp(-\exp(-2fN + \log N))$, in the limit of large graph sizes. For undirected graphs, "connected" means that there is a path between any two nodes. 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.

The performance of our models roughly fits the predictions based on the empirical analysis of random directed graphs [Fig 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. A reason for the increasing discrepancy with network size may lie in the difficulty of training extremely sparse large networks [6], 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. As such, this evidence suggests that it is sufficient for our sparse RNNs to have a strongly connected adjacency graph to perform well.

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 largest strongly connected component had fewer than $0.95N$ nodes. We found that these networks performed the task well and at a constant level across network size under the same training conditions [Fig 3B]. The line fit to these data points has a slope of $0.00003 \approx 0$. The results summarized in Fig 3A and 3B 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 [Fig 4A]. Larger 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 Fig 4A.

Output dynamics. We examined the dynamics of network outputs before, during and after the time of readout [Fig 4B]. 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 RNNs to separate their readout into two label-dependent output distributions. The readout is the average activity of the recurrent units connected to the output. 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, although the readout is sustained, the output distributions slowly mix and their separability is eventually lost [Fig 4B, histogram]. We found that classification could not be performed based on the activity of the recurrent units not connected to the output, regardless of the readout time. The sparse RNNs did not construct or reach two fixed points that correspond to the two labels, in agreement with other work [7]. Instead, the recurrent units transition back and forth, according to different irregular time scales, between their active (1) and inactivate (0) states. These states appear to have slow dynamics, meaning that the recurrent units exhibit smoothed-discrete activity. The readout tends to cluster in discrete-like states [Fig 4B, histogram], which mark the average smoothed-discrete activity of the readout units.

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 $\mathbf{J}^{\mathbf{s}}$ 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 [Fig 4C]. 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 Fig 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

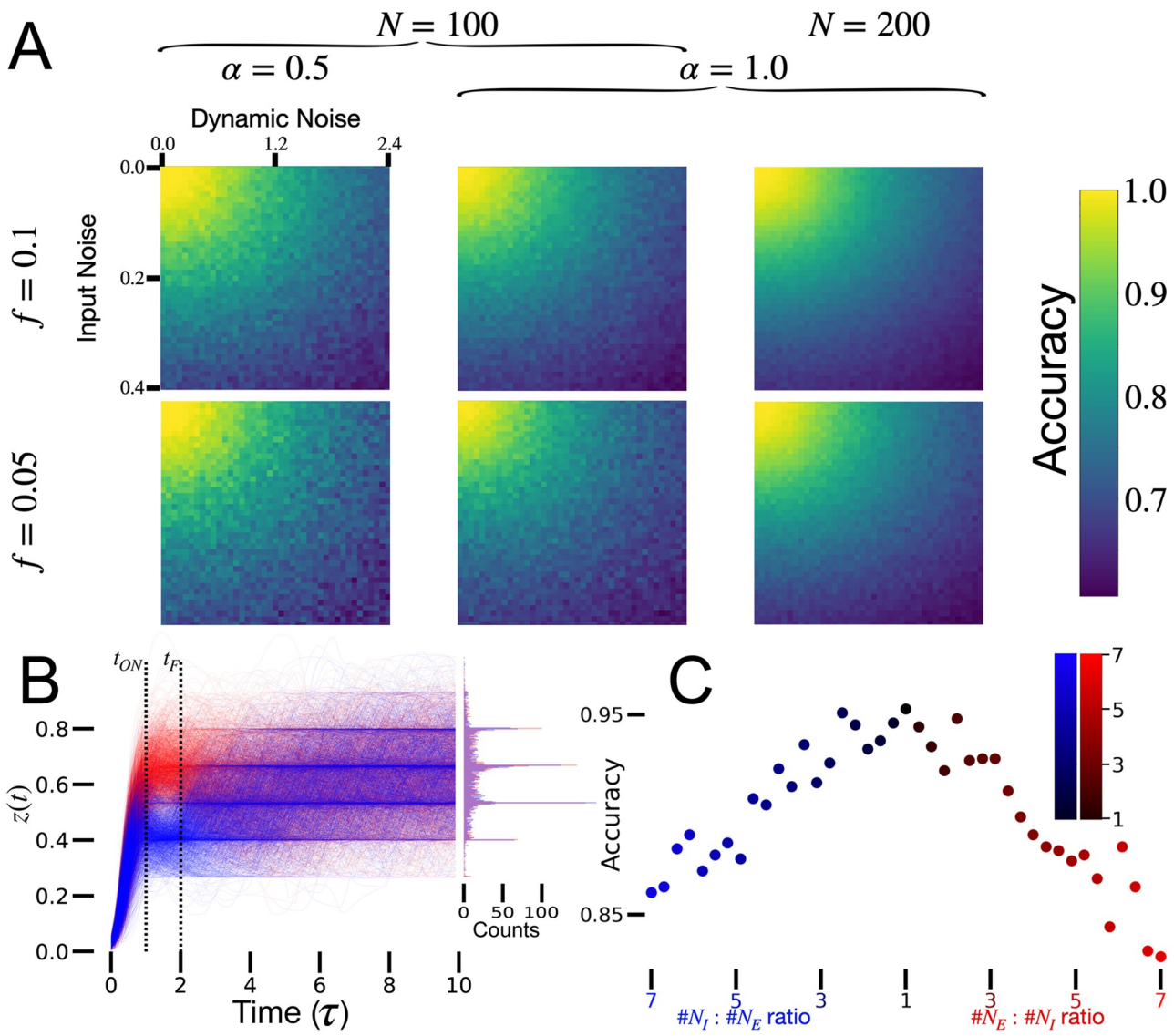

Fig 4. Robustness. A: Noise robustness heat maps for sparse RNNs. The horizontal axis indicates the level of dynamic noise, defined as the standard deviation of the dynamic noise divided by the mean of the activity in the RNN, and the vertical axis indicates the standard deviation of input noise. Sparse RNNs maintain high accuracy over a wide range of noises levels. # epochs = 150. B: 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. C: 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.

https://doi.org/10.1371/journal.pcbi.1010759.g004

separate the output distributions more rapidly by keeping low activity low while pushing high activity disproportionately higher.

In addition to the loss function $\mathcal{L}^{\text{targets}}$ previously described, we trained networks with a loss $\mathcal{L}^{\text{threshold}}_{\Delta t}$ that nonlinearly depends on the quantity $q^{\mu}(\theta - z(t))$ from time $t_{\text{F}} - \Delta t$ to time t_{F}. The nonlinear mapping assured that positive values, i.e. categorization miss-matches, contributed large penalties to the loss, whereas negative values contributed little (Materials and 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].

Table 1. Accuracy for different training methods.

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

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 and worst results are in bold. $N = 100$, $f = 0.1$, $\alpha = 1$, #epochs = 300, $n = 8$.

https://doi.org/10.1371/journal.pcbi.1010759.t001

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 $\mathbf{J}^{\mathbf{s}}$. 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 $\mathbf{J}^{\mathbf{s}}$ to

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

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 $\alpha_{\text{OS}} \approx \alpha_{\text{BPTT}}/1000 \approx 0.001$ [Fig 5A, OS]. 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 ±

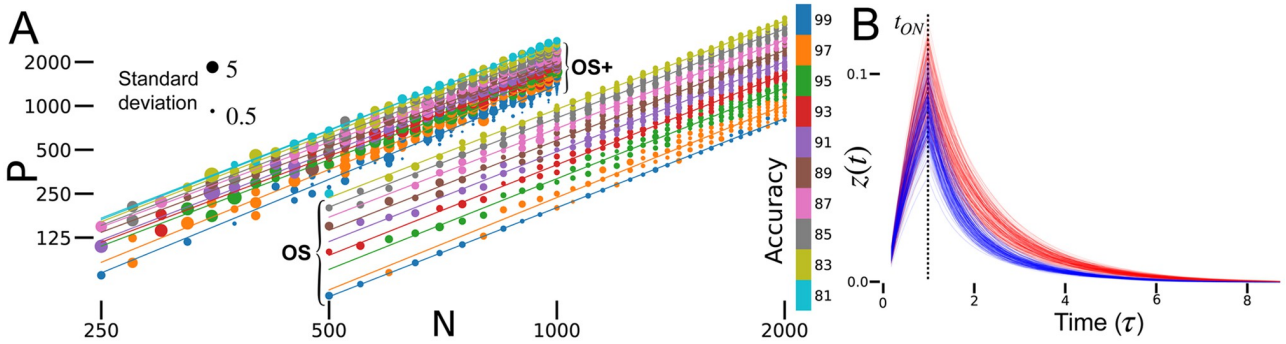

Fig 5. Hebbian-trained performance. A: Accuracy for OS and OS+ methods at various combinations of N and P. Colors represent accuracy, grouped in bins spanning 2%. Lines are fitted to every group of binned accuracies, with all slopes > 2 and regression $r > 0.98$. Size of the scatter points is proportional to the standard deviation (inset). $f = 0.1$. OS: $n = 10$. OS+: $n = 5$, $\epsilon = 0.02$, # epochs = 150. B: Output of a Hebbian-based sparse RNN over time. Input is on for τ. $N = 1000$, $f = 0.1$, $\alpha = 0.002$.

https://doi.org/10.1371/journal.pcbi.1010759.g005

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 $P \sim N^2$.

The solution found by the Hebbian learning mechanism for this task differs in many significant ways from the BPTT solution, yet the basis for classification is the amplified and non-amplified responses in both cases [Fig 5B]. Note that, similarly to the BPTT solution, Hebbian-based sparse RNN readout accumulates evidence while the stimulus is on and amplifies the response when a +1-labeled input is shown. Compared to the BPTT solution from [Fig 4B], the Hebbian solution differs in a few keys ways. First, assuming a fixed classification threshold, the readout must be read out at a precise time because the activity decays as soon as the input is turned off. This solution is not robust to readout time variations. Second, assuming a dynamic classification threshold which decays like the network's response, performance has a peak at $\sim 1.3\tau$ after the input is turned off and drops after that. Moreover, the signal to noise ratio of the classification suffers at longer times because the readout decays. Finally, the Hebbian solution is not as dynamically complex as the BPTT solution, does not sustain activity for as long periods of time and does not drive the activity as high.

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 [Fig 5A, OS+]. We find that $\alpha_{\mathrm{OS+}} \approx 10\alpha_{\mathrm{OS}} \approx 0.01$, 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 150–200 epochs, forgetting of previous patterns emerged and accuracy dropped close to chance level within 500–1000 epochs [S2 Fig].

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, 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 [8]. 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 [9], a result that also holds for feed-forward networks [10] and for various plasticity rules [10, 11].

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 per neuron (or log N in Minimum sparseness). 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, 12, 13] 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. However, the largest connected subnetwork scales linearly with N as long as $f > \frac{1}{N}$ [14, 15], a reasonable assumption which requires that, on average, every neuron have at least 1 synapse and which was used in many network dilution studies [2, 16, 17]. Though the Kushnir and Fusi results hold in light of the largest connected component, some neurons in the RNN cannot contribute to the computation. These disconnected neurons consume resources but do not help solving the task. Our results, particularly from Minimum sparseness, suggest that it suffices to have $\sim \log N$ connections per neuron to have all neurons contribute distributively to solve the task. This is only a small price to pay, compared to the case of fixed number of connections per neuron, for having the RNN be fully efficient, since $\log(10^{11}) < 26$.

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/no-go 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 no-go 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 Fig 1C. If, on the other hand, the evidence favors action, the motor/premotor circuit reacts more strongly, analogous to the red curves in Fig 1C, and the motor action is initiated.

Materials and methods

BPTT simulations

We simulated sparse RNNs with discrete time steps according to the dynamics in Eq (1). For all BPTT simulations we used custom RNN architectures and code developed in Python using PyTorch [18].

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 $\mathbf{J}^{\mathbf{s}} \equiv \mathbf{J} \odot \mathbf{M}$ 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 [Fig 1C] implicitly define a discrimination threshold $\boldsymbol{\theta} = 2$. 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 $t_{\mathrm{ON}} < t < t_{\mathrm{F}}$, 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 $\mathbf{J}^{\mathbf{s}}$ 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 −1s in $\mathbf{v}$ such that $N_E + N_I = N$. We updated the effective recurrent connections after each optimization step according to $\mathbf{J}^{\mathbf{s}}{}_{ij} \leftarrow \mathbf{J}^{\mathbf{s}}{}_{ij}\,\mathrm{sgn}(\mathbf{J}^{\mathbf{s}}{}_{ij})\mathbf{v}_j$, 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 the sparse RNN. Dynamic noise alters the recurrent state of the sparse RNN at each time step. We incorporated these noise types into Eq (1):

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

where $\mathbf{z}_{\mathrm{INPUT}}$ is Gaussian, fixed for every trial and turns off with the input at t_{ON} as noted by the step-function, and $z_{\mathrm{DYNAMIC}}(t)$ 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 $T_{+(-)}$. The expression for this loss function is:

$$\mathcal{L}^{\mathrm{targets}} = \sum_{\mu=1}^{P}\sum_{t=t_{\mathrm{ON}}}^{t_{\mathrm{F}}} \max[0,\ q^{\mu}(T_{q^{\mu}} - z^{\mu}(t))] \tag{4}$$

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 nonlinearly on $q^{\mu}(\theta - z(t))$ from time $t_{\mathrm{F}} - \Delta t$ to time t_{F}. The nonlinear mapping ensures that this quantity is positive at a time step when there is a categorization mismatch, e.g. $z(t) < \theta$ but $q^{\mu} = +1$, and negative otherwise. We used $g(x) \equiv \max(x\beta, 0) - \tanh[\max(-x\beta, 0)]$ with slope parameter β typically set to 1 for the nonlinear mapping. The

expression for this loss function is:

$$\mathcal{L}_{\Delta t}^{\text{threshold}} = \sum_{\mu=1}^{P} \sum_{t=t_F-\Delta t}^{t_F} g[q^{\mu}(\theta - z^{\mu}(t))] \tag{5}$$

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 it for all results reported, except for Fig 4 and Table 1.

Hebbian learning

All Hebbian learning results reported were generated using custom Matlab code (Mathworks, Natick, MA).

Directed graph percolation

We empirically 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 subgraph of size $N - k$ and k other nodes. 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. We also computed the estimated accuracy of a network of size N with only $N - k$ functional units, sparsity f and fN readout neurons. We call this estimated accuracy A_k^N and compute it as follows. A network of size $N - k$ will correctly classify $\alpha f(N-k)^2$) inputs for a given accuracy level. However, of the fN readout neurons, only $f(N - k)$ will be connected to the functional units, on average, so the number of correctly classified inputs will scale by $(N - k)/N$. Therefore, we estimate the accuracy as follows:

$$\begin{aligned} A_k^N &= \frac{\overbrace{\alpha f(N-k)^2}^{\text{default performance for the largest sub-graph}} \cdot \overbrace{\frac{N-k}{N}}^{\text{account for missing readouts}}}{\underbrace{\alpha f N^2}_{\text{total number of inputs}}} + \\ &+ \frac{\overbrace{\alpha f N^2 - \alpha f(N-k)^2 \cdot \frac{N-k}{N}}^{\text{chance level on remaining patterns, 50\%}}}{\underbrace{\alpha f N^2}_{\text{total number of inputs}}} = \\ &= \frac{2(N-k)^3 + [N^2 - (N-k)^3]}{2N^3} = 1 - \frac{k(3N^2 - 3NK + k^2)}{2N^3} \end{aligned} \tag{6}$$

Supporting information

S1 Fig. Input sparseness. In addition to recurrent and output sparsity, we explored the effects of input connection sparsity. In this scenario, the number of patterns stored with good

performance scales with the input sparsity as well, i.e. $P = \alpha f_{in} f N^2$ instead of the result $P = \alpha f N^2$ we report throughout the rest of the paper. We report sparse RNN performance as a function of input sparsity (f_{in}) for RNNs with two different levels of recurrent and readout sparsity (f). We have used $P = \alpha f_{in} f N^2$. $N = 100$, $\alpha = 0.5$, # epochs = 500, $n = 160$.
(TIF)

S2 Fig. OS+ failure. The OS+ Hebbian training method fails to maintain previously stored patterns when pushed beyond maximum capacity. We report sparse RNN performance as a function of the training epoch when trained with the OS+ method. Thick lines are averages and thin lines are individual simulations. $f = 0.1$, $\alpha = 0.007$, $n = 30$.
(TIF)

Acknowledgments

We thank Stefano Fusi for helpful discussions.

Author Contributions

Conceptualization: Denis Turcu, L. F. Abbott.

Data curation: Denis Turcu.

Formal analysis: Denis Turcu, L. F. Abbott.

Funding acquisition: Denis Turcu, L. F. Abbott.

Investigation: Denis Turcu, L. F. Abbott.

Methodology: Denis Turcu.

Project administration: Denis Turcu, L. F. Abbott.

Resources: L. F. Abbott.

Software: Denis Turcu, L. F. Abbott.

Supervision: L. F. Abbott.

Validation: Denis Turcu, L. F. Abbott.

Visualization: Denis Turcu.

Writing – original draft: Denis Turcu, L. F. Abbott.

Writing – review & editing: Denis Turcu, L. F. Abbott.

References

1. Rosenblatt F. The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review. 1958; 65(6):386–408. https://doi.org/10.1037/h0042519 PMID: 13602029
2. Kushnir L, Fusi S. Neural classifiers with limited connectivity and recurrent readouts. Journal of Neuroscience. 2018; 38(46):9900–9924. https://doi.org/10.1523/JNEUROSCI.3506-17.2018 PMID: 30249794
3. Cover TM. Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition. IEEE Transactions on Electronic Computers. 1965; EC-14(3):326–334. https://doi.org/10.1109/PGEC.1965.264137
4. Nadal JP, Toulouse G, Changeux JP, Dehaene S. Networks of formal neurons and memory palimpsests. Epl. 1986; 1(10):535–542. https://doi.org/10.1209/0295-5075/1/10/008
5. Erdös P, Rényi A. On random graphs I. Publicationes Mathematicae. 1959; 6:290–297.
6. Dauphin YN, Bengio Y. Big Neural Networks Waste Capacity;.

7. Wang XJ. Decision Making in Recurrent Neuronal Circuits. Neuron. 2008; 60(2):215–234. https://doi.org/10.1016/j.neuron.2008.09.034 PMID: 18957215

8. Löwe M, Vermet F. The Hopfield Model on a Sparse Erdös-Renyi Graph. Journal of Statistical Physics 2011 143:1. 2011; 143(1):205–214.

9. Bogacz R, Brown MW, Giraud-Carrier C. Model of Familiarity Discrimination in the Perirhinal Cortex. Journal of Computational Neuroscience. 2001; 10:5–23. https://doi.org/10.1023/A:1008925909305 PMID: 11316340

10. Tyulmankov D, Yang GR, Abbott LF. Meta-learning synaptic plasticity and memory addressing for continual familiarity detection. Neuron. 2021. https://doi.org/10.1016/j.neuron.2021.11.009 PMID: 34861149

11. Bogacz R, Brown MW. Comparison of computational models of familiarity discrimination in the perirhinal cortex. Hippocampus. 2003; 13(4):494–524. https://doi.org/10.1002/hipo.10093 PMID: 12836918.

12. Brunel N, Hakim V, Isope P, Nadal JP, Barbour B. Optimal information storage and the distribution of synaptic weights: Perceptron versus Purkinje cell. Neuron. 2004; 43(5):745–757. https://doi.org/10.1016/S0896-6273(04)00528-8 PMID: 15339654

13. Collins J, Sohl-Dickstein J, Sussillo D. Capacity and trainability in recurrent neural networks. 5th International Conference on Learning Representations, ICLR 2017—Conference Track Proceedings. 2017; p. 1–17.

14. Erdos P, Rényi A. On the evolution of random graphs; 1960.

15. Bollobs B. In: The Evolution of Random Graphs—the Giant Component. 2nd ed. Cambridge Studies in Advanced Mathematics. Cambridge University Press; 2001. p. 130–159.

16. Derrida B, Gardner E, Zippelius A. An exactly solvable asymmetric neural network model. Epl. 1987; 4 (2):167–173. https://doi.org/10.1209/0295-5075/4/2/007

17. Amit DJ. In: Robustness—Getting Closer to Biology. Cambridge University Press; 1989. p. 345–386.

18. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: Wallach H, Larochelle H, Beygelzimer A, d'Alché-Buc F, Fox E, Garnett R, editors. Advances in Neural Information Processing Systems 32. Curran Associates, Inc.; 2019. p. 8024–8035. Available from: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.