Neuron

Article

Meta-learning synaptic plasticity and memory addressing for continual familiarity detection

Highlights

- Meta-learning is used to discover network architectures and plasticity rules
- Anti-Hebbian plasticity emerges as the mechanism for encoding familiarity
- Strong feedforward synapses emerge as an addressing function for storage and retrieval
- Experimental features such as repetition suppression are reproduced

Authors

Danil Tyulmankov,
Guangyu Robert Yang, L.F. Abbott

Correspondence

dt2586@columbia.edu (D.T.),
yanggr@mit.edu (G.R.Y.),
lfa2103@columbia.edu (L.F.A.)

In brief

Tyulmankov et al. use meta-learning to build neural network models for continual familiarity detection. They show that anti-Hebbian plasticity is the preferred mechanism for optimizing memory capacity and propose strong feedforward weights as an explicit addressing mechanism for selecting memory locations during storage and retrieval.

Tyulmankov et al., 2022, Neuron *110*, 544–557
February 2, 2022
https://doi.org/10.1016/j.neuron.2021.11.009

CellPress

CellPress

Neuron

Article

Meta-learning synaptic plasticity and memory addressing for continual familiarity detection

Danil Tyulmankov,[1,4,*] Guangyu Robert Yang,[1,2,*] and L.F. Abbott[1,3,*]
[1]Zuckerman Mind Brain Behavior Institute, Department of Neuroscience, Columbia University, New York, NY 10027, USA
[2]Department of Brain and Cognitive Sciences, Department of Electrical Engineering and Computer Science, Center for Brains, Minds, and Machines, McGovern Institute for Brain Research, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
[3]Department of Physiology and Cellular Biophysics, Columbia University, New York, NY 10027, USA
[4]Lead contact
*Correspondence: dt2586@columbia.edu (D.T.), yanggr@mit.edu (G.R.Y.), lfa2103@columbia.edu (L.F.A.)
https://doi.org/10.1016/j.neuron.2021.11.009

SUMMARY

Over the course of a lifetime, we process a continual stream of information. Extracted from this stream, memories must be efficiently encoded and stored in an addressable manner for retrieval. To explore potential mechanisms, we consider a familiarity detection task in which a subject reports whether an image has been previously encountered. We design a feedforward network endowed with synaptic plasticity and an addressing matrix, meta-learned to optimize familiarity detection over long intervals. We find that anti-Hebbian plasticity leads to better performance than Hebbian plasticity and replicates experimental results such as repetition suppression. A combinatorial addressing function emerges, selecting a unique neuron as an index into the synaptic memory matrix for storage or retrieval. Unlike previous models, this network operates continuously and generalizes to intervals it has not been trained on. Our work suggests a biologically plausible mechanism for continual learning and demonstrates an effective application of machine learning for neuroscience discovery.

INTRODUCTION

Every day, a continual stream of sensory information and internal cognitive processing causes lasting synaptic changes in our brains that alter our responses to future stimuli. It remains a mystery how neural activity and local synaptic updates coordinate to support distributed storage and readout of information and, in particular, how ongoing synaptic changes due to either new memories or homeostatic mechanisms do not interfere with previously stored information.

Familiarity detection—identifying whether a stimulus has been previously encountered—is a simple and ubiquitous form of memory that serves as a useful testbed for addressing these issues. Classical studies have demonstrated that human recognition memory capacity for images is "almost limitless," retention following a power law as a function of the number of items viewed (Standing, 1973). Theoretical work has shown that the number of memories stored by a familiarity detection network depends on the synaptic plasticity rule and can scale proportionally to the number of synapses (Bogacz and Brown, 2003). More recent behavioral work has further demonstrated an impressive capacity in a continual setting, the error rate as a function of the number of intervening items exhibiting a "power law of forgetting" (Brady et al., 2008) and theoretical studies showing that this is achievable by synapses with metaplasticity (Fusi et al., 2005; Ji-An et al., 2019). Neural signals of visual familiarity have been observed as reductions in responses to repeated presentations of a stimulus, a phenomenon known as repetition suppression (Grill-Spector et al., 2006; Meyer and Rust, 2018; Miller et al., 1991; Xiang and Brown, 1998). At the timescales relevant for this task—one-shot memorization on the order of seconds and long-term forgetting on the order of days—this is plausibly caused by depression of excitatory synapses or potentiation of inhibitory ones (Lim et al., 2015).

Previous modeling work on recognition memory used a predesigned architecture and plasticity rule and both empirical and analytic evaluation of performance (Androulidakis et al., 2008; Bogacz and Brown, 2003; Norman and O'Reilly, 2003; Sohal and Hasselmo, 2000). An emerging approach uses a machine learning technique known as "meta-learning," or "learning how to learn" (Thrun and Pratt, 2012), that uses optimization tools to rapidly search for mechanisms that artificial neural networks can use to solve a learning/memory task (Confavreux et al., 2020; Gu et al., 2019; Jordan et al., 2021; Lindsey and Litwin-Kumar, 2020; Metz et al., 2019; Najarro and Risi, 2021). In contrast to hand-designed models, meta-learning enables unbiased exploration of a large family of architectures and plasticity rules. Importantly, it is possible to impose constraints that ensure biological plausibility (Bengio et al., 1991).

In this work, we investigate not only "how" memories are stored—the synaptic plasticity rule—but also "where"—the mechanism for addressing the storage and retrieval locations.

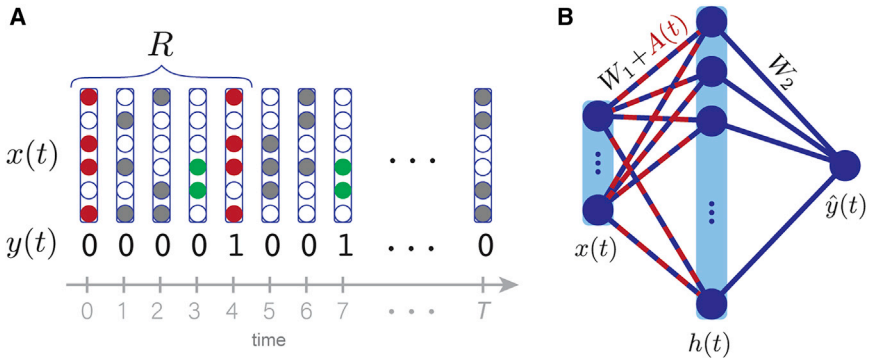

Figure 1. Continual familiarity detection task and HebbFF model

(A) The continual familiarity detection task. Given a continual stream of stimuli $x(t)$, the desired output is $y(t) = 1$ if the stimulus has appeared previously and $y(t) = 0$ otherwise. For a given dataset, repeat stimuli always appear at an interval R after their first presentation. Although the task is continual, for the purposes of network training, we use a finite-duration trial of length $T \gg R$.

(B) The HebbFF network architecture. A feedforward layer is endowed with ongoing Hebbian plasticity, the parameters of which are optimized using stochastic gradient descent. The hidden units are linearly read out to produce the network's estimate of familiarity $\hat{y}(t)$.

Classical models of memory rely on "content-based addressing" (Hopfield, 1982), whereby a partial cue elicits recall of the full memory through recurrent dynamics, but do not explicitly select which synapses store the memory. On the other hand, "key-value" memory networks in machine learning (Graves et al., 2014, 2016) store values in a memory matrix indexed explicitly by keys, analogous to the addressing in a computer's random-access memory, although such models lack a biological interpretation (but see Tyulmankov et al., 2021). Our model includes both a synaptic plasticity rule and an explicit addressing mechanism.

Positing that the answer to "when" plasticity should occur is "always," we consider a simple version of "what" to remember: familiarity. We construct a family of models that recognize previously experienced stimuli and, importantly, learn and operate continuously without separate learning and testing phases, avoiding catastrophic forgetting, a phenomenon in which a network renders stored information unreadable (Beaulieu et al., 2020; Parisi, 1986). The capacity of these networks remains constant over time, so they can be continually fed new inputs with no reduction in steady-state memory performance.

We use a feedforward network architecture with ongoing synaptic plasticity, parameters of which are meta-learned using gradient descent to optimize continual familiarity detection. Unlike similar models (Ba et al., 2016; Miconi et al., 2019), to isolate synaptic plasticity as the unique memory mechanism, we avoid recurrent connectivity that could store memory through maintained neuronal activations. This architecture, unlike recurrent networks, generalizes naturally over a range of repeat intervals even if trained on a single interval. We show that an anti-Hebbian plasticity rule (co-activated neurons cause synaptic depression) enables repeat detection over longer intervals than a Hebbian rule and leads to experimentally observed features such as repetition suppression in the hidden-layer neurons. Furthermore, an addressing function emerges through strong static feedforward weights, selecting a unique neuron to index the synapses for storage of a novel stimulus and detection of a familiar one.

RESULTS

Continual familiarity detection task

In our task, a continuous stream of stimuli is presented to the network (Figure 1A). With probability $1 - p$, the stimulus at time t is a randomly generated binary vector $\boldsymbol{x}(t)$, where each component is either $+1$ or -1. With probability p, the stimulus is a copy of the stimulus presented R time steps ago. The output of the network should be $y(t) = 0$ if $\boldsymbol{x}(t)$ is novel and $y(t) = 1$ if it is familiar (i.e., has appeared previously). We begin by considering familiarity detection for uncorrelated stimuli, but in later sections we generalize to a task that requires simultaneous classification and familiarity detection and to a dataset of images (see STAR Methods).

HebbFF network architecture

To investigate the effectiveness of synaptic plasticity for solving this task, we use a feedforward neural network with a single hidden layer and activity-dependent ongoing Hebbian plasticity to implement the memory function (HebbFF) (Figure 1B). We do not include any recurrent connections, to ensure that memory cannot be stored through persistent neuronal activity, thus isolating synaptic plasticity as the only memory mechanism.

In the HebbFF network, a group of hidden-layer neurons with firing rates given by an N-dimensional vector $\boldsymbol{h}(t)$, receives a d-dimensional input $\boldsymbol{x}(t)$ at time t. The input to each hidden-layer neuron is weighted by its corresponding synaptic strength and then transformed into a firing rate through a nonlinear activation function $\sigma(\cdot)$. The synaptic strength between the postsynaptic neuron with rate $h_i(t)$ and the presynaptic neuron carrying the input $x_j(t)$ is the (i,j) component of an N-by-d matrix that is the sum of a static matrix $\boldsymbol{W}_1$ and a plastic matrix $\boldsymbol{A}(t)$. Thus, the firing rate of the hidden layer is given by

$$\boldsymbol{h}(t) = \sigma((\boldsymbol{W}_1 + \boldsymbol{A}(t))\boldsymbol{x}(t) + \boldsymbol{b}_1),$$

where σ is the logistic function applied element-wise and $\boldsymbol{b}_1$ is a vector representing baseline currents into the hidden layer. The matrix $\boldsymbol{W}_1$ is fixed after training, and its unconstrained values are set though optimization. Its structure serves an addressing function by imposing a unique baseline activity pattern in the hidden layer for each input. The plastic matrix $\boldsymbol{A}(t)$ is updated at every time step: its (i,j) component decays by a factor $0<\lambda<1$ and is incremented by a Hebbian product of the pre- and postsynaptic activities, $h_i(t)x_j(t)$. A plasticity rate parameter $-\infty<\eta<\infty$ controls the sign and magnitude of this increment. In matrix form, the synaptic update rule is

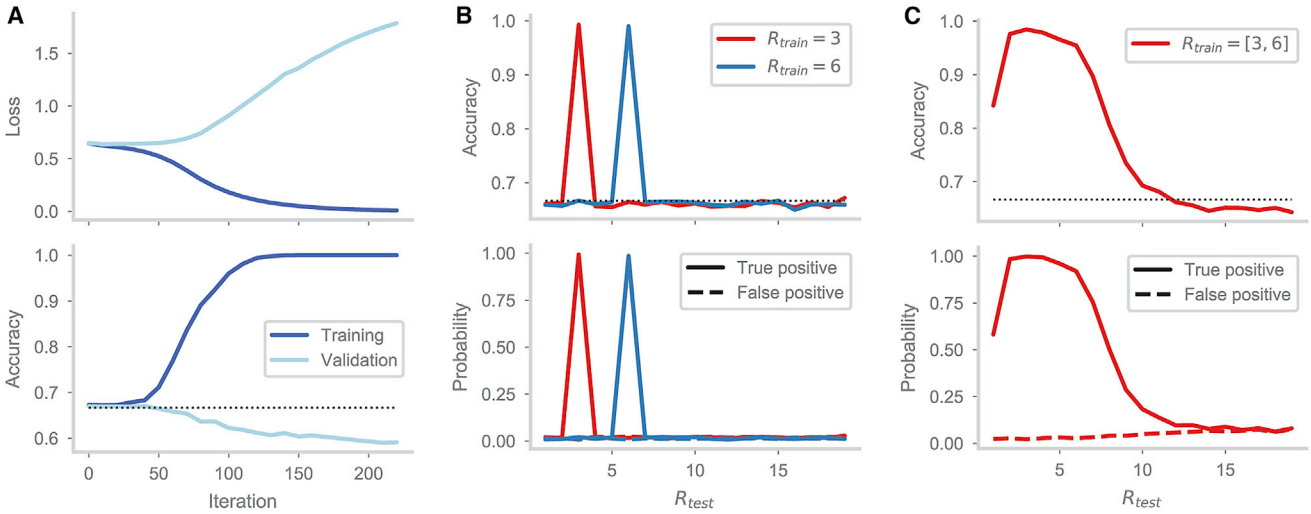

Figure 2. RNN performance on continual familiarity detection

(A) Training an RNN ($d = 100$ input dimension, $N = 100$ recurrent units) on a single familiarity detection dataset ($T = 500$ stimulus presentations, repeat interval $R = 3$). Although the loss (top) approaches 0 and accuracy (bottom) approaches 1 on training data (red), performance on a validation data (blue) with the same parameters fails to generalize, even when tested in distribution with the same R.

(B) RNN trained with "infinite data," $R = 3$ (red) or $R = 6$ (blue). Accuracy (top) and true and false positive probabilities (bottom) shown as a function of the repeat interval on validation data. RNNs perform well in distribution on datasets with the same repeat interval as used during training but fail to generalize out of distribution to other repeat intervals.

(C) RNN trained on "infinite data" with both intervals $R = 3$ and $R = 6$ interpolates between the intervals but fails to extrapolate.

$$\boldsymbol{A}(t+1) = \lambda \boldsymbol{A}(t) + \eta \boldsymbol{h}(t)\boldsymbol{x}(t)^{\mathrm{T}}.$$

Finally, the output of the network $\hat{y}(t)$ is a linear readout of the hidden layer, and, because the target $y(t)$ is binary, we bound the readout with the logistic function,

$$\hat{y}(t) = \sigma(\boldsymbol{W}_2 \boldsymbol{h}(t) + \boldsymbol{b}_2).$$

The response of the network is "familiar" if $\hat{y}(t) > 1/2$ and "novel" otherwise. Although in the general case $\boldsymbol{W}_2$ is unconstrained, to simplify analysis we later consider a uniform readout in which all entries of $\boldsymbol{W}_2$ are equal, with no appreciable change in performance.

To construct the network, we use backpropagation through time (BPTT) to meta-learn the parameters $\boldsymbol{W}_1$, $\boldsymbol{b}_1$, $\boldsymbol{W}_2$, $\boldsymbol{b}_2$, λ, η, which are fixed once training is completed (STAR Methods). The continual familiarity detection task—the "learning" task—is then performed by the ongoing synaptic dynamics of $\boldsymbol{A}(t)$, controlled by the fixed parameters. These dynamics are a biologically plausible mechanism for solving the continual memory task, but BPTT is simply used as an optimization tool to find suitable parameters of the network.

The HebbFF network generalizes both in and out of distribution

As a benchmark for comparing HebbFF performance, we train a long short-term memory (LSTM) network (Hochreiter and Schmidhuber, 1997)—a recurrent neural network (RNN) architecture well suited for memory performance—on the continual familiarity detection task. Unlike HebbFF, which stores its input history in the plastic synaptic matrix $\boldsymbol{A}(t)$, an RNN uses ongoing neuronal activity.

If we train the RNN using a single dataset with $T = 500$ image presentations (STAR Methods) and a repeat interval of $R = 3$, it successfully learns the training set but fails to generalize to new test sets with the same R (Figure 2A). To fix this, we use an "infinite data" approach in which we generate a new dataset for every iteration of BPTT, each with the same value of $R = 3$. Trained in this way, the RNN now generalizes in distribution across datasets with $R = 3$ (i.e., to datasets drawn from the same distribution as the training data, which is parameterized by R) but fails to generalize out of distribution to data with any other value of R (i.e., to datasets from a different distribution) (Figure 2B). The same result holds with $R = 6$ (Figure 2B). We can further train the RNN with items spaced at intervals of both $R = 3$ and $R = 6$ (the value of R is chosen randomly for each familiar stimulus rather than being fixed). Although the network can interpolate between the trained values, it does not extrapolate well to larger or smaller ones (Figure 2C). Although it is likely possible to train the RNN to perform well for multiple values of R with more complex training schedules, we believe that poor out-of-distribution generalization is a bottleneck of the RNN approach.

In contrast, the HebbFF network exhibits both in- and out-of-distribution generalization. Even when trained on a single dataset with a fixed repeat interval R, the network generalizes to new test sets with the same R (Figure 3A) and even to those with different R values. Critically, the training procedure is the same as for the RNN above, but HebbFF successfully learns a qualitatively different solution because of its inductive bias. Trained with "infinite data" (the scheme we use in general), HebbFF generalizes

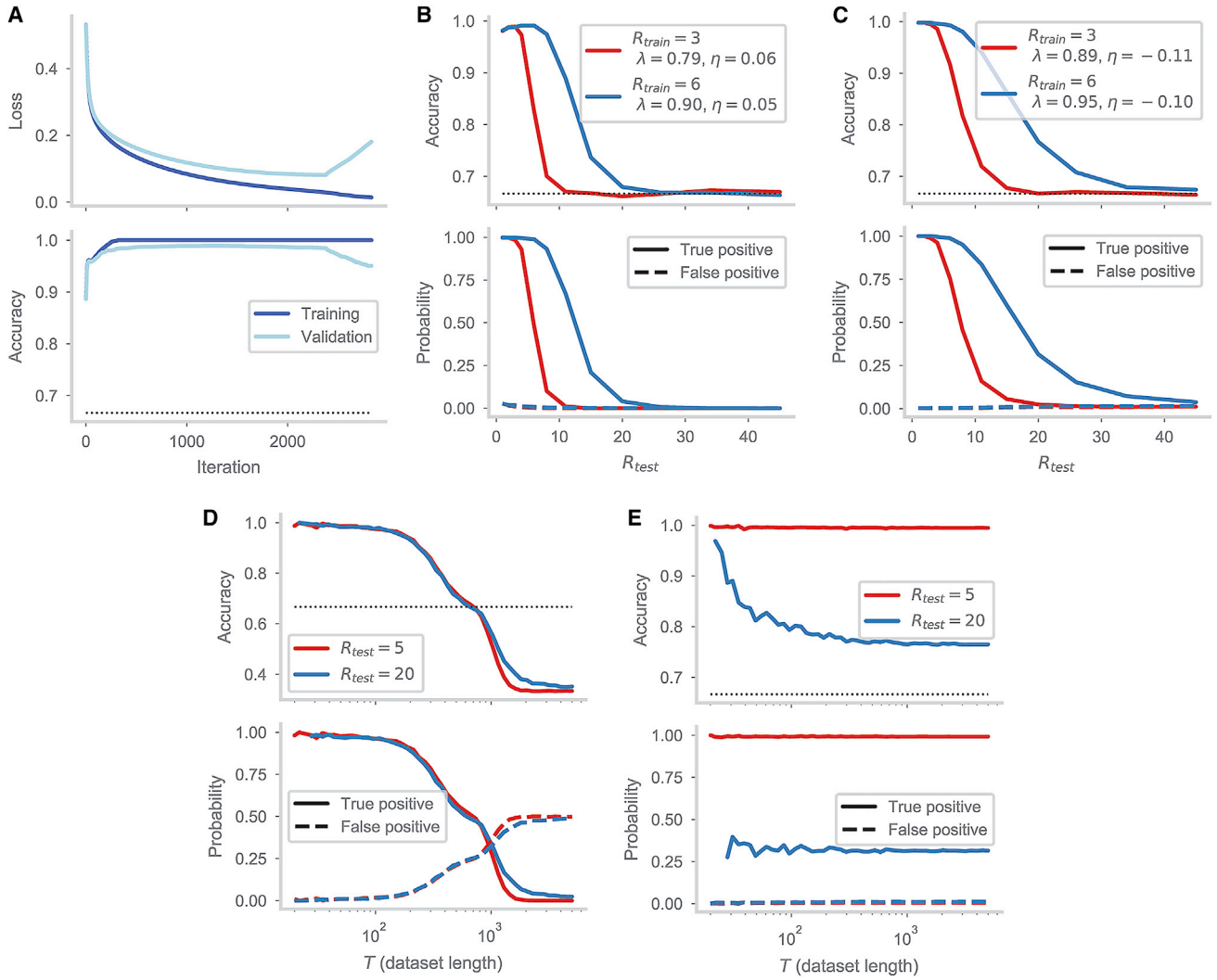

Figure 3. Hebbian versus anti-Hebbian plasticity and continual operation

(A) Training the HebbFF network ($d = N = 100$), as in Figure 2A. Both training and validation loss decrease, indicating in-distribution generalization. Over many iterations, overtraining occurs because of the use of a single dataset, increasing the final validation loss.

(B) HebbFF, trained as in Figure 2B, shows not only in-distribution generalization to datasets with the same R but also out-of-distribution generalization to data with any smaller and slightly larger R.

(C) HebbFF with a different initialization converges to an anti-Hebbian learning rule (see also Figure S8) with generalization over a longer R values than Hebbian.

(D) Model from Bogacz and Brown (2003), evaluated on the continual familiarity detection task, varying the length T of the trial. Accuracy (top) is near perfect regardless of the repeat interval R (blue versus red curve) until the model reaches its capacity ($P^* \approx 100$ for network size $d = N = 100$) because the model reliably stores the first P^* patterns. Accuracy rapidly drops below chance for $T > P^*$ as the model begins to report familiar stimuli as novel (see Figure S2B).

(E) HebbFF network operates continuously. Accuracy is consistent with the generalization curve from (C), with near perfect performance for $R_{test} = 5$ and above 80% for $R_{test} = 20$ for any trial length. True and false probabilities (bottom) are better representations as accuracy is artificially higher for small T because of the low proportion of familiar stimuli.

to datasets with smaller and larger R (Figure 3B). Matching the number of dynamic variables rather than the number of hidden neurons, HebbFF still shows superior generalization compared with the RNN (Figure S1). This qualitative difference in performance suggests that Hebbian plasticity provides a more "natural" mechanism for familiarity detection.

The generalization performance of HebbFF is due to the fact that the memory representation of an item does not change over time, other than being scaled by a factor. A stimulus $\boldsymbol{x}(t)$ is initially stored as the outer product of $\boldsymbol{h}(t)$ and $\boldsymbol{x}(t)$, multiplied by the plasticity rate η. The plastic component of the connectivity matrix also contains terms arising from previously stored memories, which for the purposes of this particular stimulus act as additive noise ε:

$$\boldsymbol{A}(t+1) = \eta \boldsymbol{h}(t)\boldsymbol{x}(t)^{\mathsf{T}} + \varepsilon$$

As subsequent stimuli are presented, the representation of $\boldsymbol{x}(t)$ maintains the same form, so that k time steps later it is still stored as the outer product of $\boldsymbol{h}(t)$ and $\boldsymbol{x}(t)$, scaled by a factor λ^k:

$$\boldsymbol{A}(t+k) = \lambda^k \eta \boldsymbol{h}(t)\boldsymbol{x}(t)^{\mathsf{T}} + \lambda^k \varepsilon + \varepsilon'$$

where further additive noise ε' arises from stimuli presented after $\boldsymbol{x}(t)$.

Unlike HebbFF, RNNs are poor at generalizing across intervals R because the dynamics of their units allow the memory representation of a stimulus to change arbitrarily over time. The RNN only generates the appropriate representation at the time when a query is expected, namely, after a delay equal to the value of R used during training. This makes it difficult to generalize across intervals.

The "how" of synaptic plasticity: storage via an anti-Hebbian rule

The plasticity rate η in HebbFF can be positive or negative, resulting in either Hebbian or anti-Hebbian plasticity. For the Hebbian solution with $\eta > 0$, synapses are potentiated in response to a stimulus. When it is repeated, the hidden-layer activity is higher because of the increased strength of the synapses storing the memory. For anti-Hebbian plasticity, $\eta < 0$, synapses are depressed when a memory is stored. In this case, the hidden-layer activity is lower for a familiar stimulus, consistent with experimental results of repetition suppression (Grill-Spector et al., 2006; Meyer and Rust, 2018; Xiang and Brown, 1998). Furthermore, the meta-learning algorithm is more likely to converge to the anti-Hebbian solution, especially when trained with a relatively large repeat interval, even if the initial value of η is positive, and almost always when the initial value is negative (Figure S8).

Anti-Hebbian plasticity enables successful familiarity detection over considerably longer intervals than a Hebbian rule (Figure 3C). To understand this, note that the memory of a stimulus is degraded in two ways: plasticity events obscure existing memories, and plastic weights decay over time. With an anti-Hebbian plasticity rule, the hidden layer activation $\boldsymbol{h}(t)$ is close to zero for a familiar stimulus. As a result, the plasticity update $\boldsymbol{h}(t)\boldsymbol{x}(t)^{\mathsf{T}}$ when the stimulus is repeated is negligible, as if a stimulus were not presented at that time step. This effectively reduces the number of plasticity events, and the disruption of existing memories. As a secondary effect, the smaller number of plasticity events allows a larger λ (slower decay rate) to be used while still controlling the amplitude of plastic weights (Figure S8), further extending the lifetime of the memory. Because of their superior performance and consistency with experimental results, we consider only anti-Hebbian solutions throughout the following sections.

The "when" of synaptic plasticity: continual learning without catastrophic forgetting

Previous modeling work using anti-Hebbian plasticity mechanisms for familiarity detection (Bogacz and Brown, 2003) focused on a paradigm used in classic studies of recognition memory (Standing, 1973) in which subjects are serially presented an entire dataset and later asked to identify which stimulus is familiar in a two-alternative forced-choice (2AFC) test. Analogously, this previous modeling work used explicit "learning" and "testing" phases and demonstrated an impressive capacity for recognition memory (Bogacz and Brown, 2003) (Figure S2A). When evaluated on the continual memory task that we use, the Bogacz-Brown model has near perfect performance if the number of stimuli T in the dataset is smaller than the model's capacity P^*, independent of the value of the repeat interval R (Figure 3D), as the model successfully stores all $T<P^*$ stimuli. As the dataset size increases, however, the model performance declines because of catastrophic interference (Figures 3D and S2B; STAR Methods). To store additional memories, the old memories must be removed by resetting the synaptic weights.

In real-world scenarios, an organism typically does not experience a dedicated "learning" phase. The answer to "when" synaptic plasticity should occur is "always." As such, the HebbFF model operates continually rather than using separate learning and evaluation phases. Its performance is independent of the length of the dataset, and it can operate continuously without any need to reset the synaptic weights. For example, a HebbFF network trained with $R = 5$ operates at near perfect performance irrespective of the duration of the trial T when tested with $R = 5$ (Figure 3E). Similarly, when tested with $R = 20$, it operates continually at near 80% accuracy (Figure 3E), as expected from the generalization curve in Figure 3C (note that for small T the accuracy [Figure 3E, top, blue] is transiently elevated because the fraction of novel stimuli is more than $\frac{2}{3}$). In other words, the model has a moving window in time within which it can successfully detect a familiar stimulus and forgets old stimuli gracefully without suffering from catastrophic interference.

The "where" of synaptic plasticity: addressing via strong feedforward weights

In the HebbFF network, the hidden layer plays a dual role. On one hand, it must produce a reliable familiarity signal for the readout to decode. On the other, it must create a robust representation of the input stimulus during the Hebbian plasticity update. The hidden activity is controlled by the fixed parameters $\boldsymbol{W}_1$ and $\boldsymbol{b}_1$, as well as the plastic matrix $\boldsymbol{A}(t)$. Here, we investigate how $\boldsymbol{W}_1$, $\boldsymbol{b}_1$, and $\boldsymbol{A}(t)$ influence these two aspects of the familiarity detection task.

To simplify this analysis, we restrict $\boldsymbol{W}_2$ to be a scaled 1 -by- N matrix of ones, $\boldsymbol{W}_2 = \alpha_2[1, \ldots, 1]$, where α_2 is a trained scalar. Similarly, we restrict $\boldsymbol{b}_1 = \beta_1[1, \ldots, 1]^{\mathsf{T}}$. As the hidden units now contribute equally to the readout, they are statistically identical (although not necessarily independent). Therefore, the rows of $\boldsymbol{W}_1$ and $\boldsymbol{A}(t)$ are statistically identical, allowing us to meaningfully plot histograms of the corresponding input currents. This choice of output weights does not affect the performance or memory mechanism (Figure S3).

Networks trained with larger R have sparser hidden unit activity (Figures 4A–4C): the sparser the activity, the less plasticity is evoked and thus the longer memories can be retained without overwriting. In the limiting case we might expect that exactly one neuron is active for a novel stimulus, and none are active for familiar stimuli. Associated with this increased sparsity in activity, $\boldsymbol{W}_1$ is also sparser for larger R (Figures 4D–4F and S8).

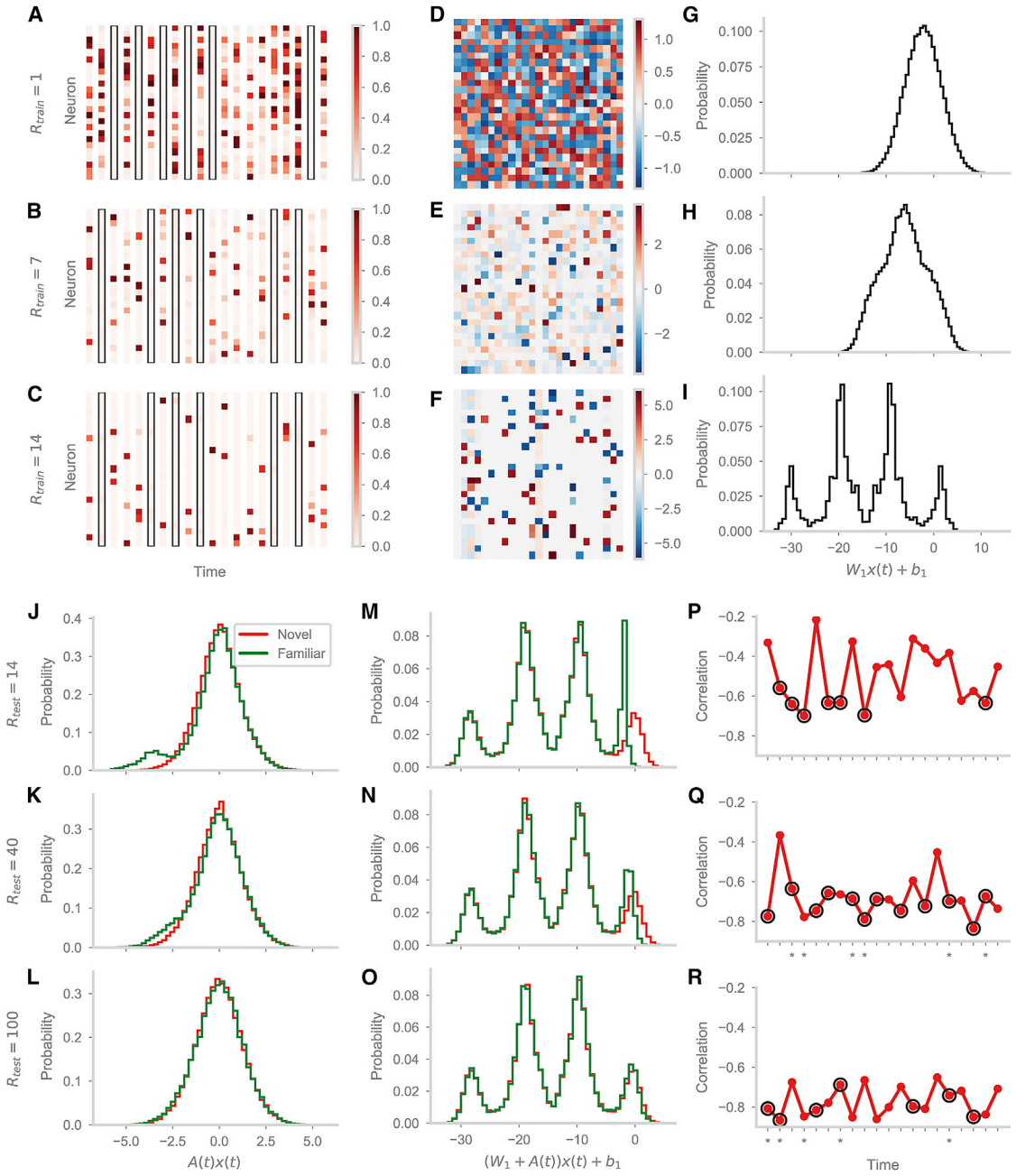

Figure 4. Storage and readout mechanism

(A–C) Hidden-layer activity $\boldsymbol{h}(t)$ over 20 consecutive time steps for networks with input dimension $d = 25$ and $N = 25$ hidden units, trained on datasets with $R =$ 1, 7, or 14, respectively. Familiar stimuli (black rectangles) cause silencing (i.e., repetition suppression of hidden-layer activity). Activity for novel stimuli becomes sparser for networks trained with larger R.

(D–F) Static weight matrix $\boldsymbol{W}_1$ of the networks from (A)–(C). The weight matrix becomes sparser (Figure S8) and individual weight magnitudes increase for networks trained with larger R, enabling sparser activity in the hidden layer for novel stimuli.

(G–I) Distributions of hidden-layer input current due to the static component of the synapses ($\boldsymbol{W}_1$, $\boldsymbol{b}_1$) for the networks from (A)–(C). For networks trained with larger R, the distribution becomes multi-modal, with the number of modes equal (approximately) to the number of high-magnitude values per row of $\boldsymbol{W}_1$, plus one. Because of the bias, only the rightmost mode has the potential to produce firing rates that are significantly above zero.

(J–L) Distributions of input current into the hidden layer due to the plastic component of the synapses $\boldsymbol{A}(t)$, for novel (red) and familiar (green) stimuli. We consider only the trained network from (C), (F), and (I) and evaluate its behavior on test sets with $R = 14$, 40, or 100, corresponding to perfect, intermediate, and chance accuracy. The large central mode occurs because of stored stimuli uncorrelated with the input stimulus $\boldsymbol{x}(t)$. In the novel case, the input is uncorrelated with all the stored stimuli by definition, and thus there is only one mode. Similarly, in the familiar case with a long delay interval $R = 100$, the stored stimulus has decayed

(legend continued on next page)

To isolate the effect of $\boldsymbol{W}_1$ on hidden unit activity, we compute a histogram of the input current into the hidden layer due to the non-plastic synapses, $\boldsymbol{W}_1\boldsymbol{x}(t)+\boldsymbol{b}_1$, across units and across time (Figures 4G–4I). As R increases, the distribution becomes multimodal as a result of the combinatorial structure of the rows in $\boldsymbol{W}_1$ (more evident in the idealized model; see below and STAR Methods). In general, the number of peaks in this distribution depends on the number of large-magnitude values of $\boldsymbol{W}_1$ per row. Critically, because of the logistic function nonlinearity, only the rightmost peak in Figure 4I is large enough to elicit appreciable activity in the hidden layer. This peak drives the small number of hidden units that are significantly activated by a novel stimulus. In other words, the $\boldsymbol{W}_1$ matrix acts like an addressing function to select a small subset of hidden units to store the memory of a given stimulus.

We next consider the effect of $\boldsymbol{A}(t)$, focusing on the network trained to maximum capacity (Figures 4C, 4F, and 4I) (see Curriculum training and empirical capacity). For a novel stimulus, the distribution of the input current due to the plastic synapses $\boldsymbol{A}(t)\boldsymbol{x}(t)$ is unimodal and symmetric about zero (Figures 4J–4L). For a familiar stimulus, however, there is an additional peak at approximately $\lambda^{R-1}\eta d$. This peak is due to the dot product of the input vector $\boldsymbol{x}(t-R)$ (stored in the matrix $\boldsymbol{A}(t)$ as $\lambda^{R-1}\eta\boldsymbol{h}(t-R)\boldsymbol{x}(t-R)^{\mathrm{T}}$), and the familiar input vector $\boldsymbol{x}(t)=\boldsymbol{x}(t-R)$. Importantly, the neurons that exhibit this behavior are the same ones active due to $\boldsymbol{W}_1$ when the stimulus was novel. Thus, again, $\boldsymbol{W}_1$ provides addressing functionality (now indirectly through its effect on $\boldsymbol{A}(t)$), allowing the system to probe the same neurons not only during storage but also during recall.

Finally, the total hidden-layer input current is the sum of these two components, $(\boldsymbol{W}_1+\boldsymbol{A}(t))\boldsymbol{x}(t)+\boldsymbol{b}_1$ (Figures 4M–4O). Comparing Figures 4I and 4O, we see that the large central symmetric mode of the $\boldsymbol{A}(t)\boldsymbol{x}(t)$ distribution does not significantly affect the total hidden-layer input current. Rather, the familiarity signal arises because the smaller peak of the $\boldsymbol{A}(t)\boldsymbol{x}(t)$ distribution pushes the rightmost peak of the $\boldsymbol{W}_1\boldsymbol{x}(t)+\boldsymbol{b}_1$ distribution below zero (Figure 4M). Anti-correlation between the two input currents for familiar stimuli (Figures 4P–4R) indicates that this shift is caused by the input current from the plastic component of the synapse canceling the input current from the fixed component, resulting in lower activation (i.e., repetition suppression).

Curriculum training and empirical capacity

A randomly initialized HebbFF network may fail to find a solution if directly trained with a large value of R (Figure S8). Instead, we use a curriculum training procedure to bootstrap the optimized solution. First, the network is trained on data with $R=1$, using the "infinite data" regime. Once the accuracy is above 99%, R is incremented by 1, and training continues on data with $R=2$. This process continues until R becomes large enough that the network cannot find a solution with accuracy above 99% (i.e., if R is not incremented for at least 2 million iterations) (Figure 5A). We thus define the memory capacity $R_{\max}$ as the largest value of R for which the familiarity detection accuracy is above 99%.

We curriculum-train networks of different sizes and plot the capacity $R_{\max}$ for each one (Figure 5B). For consistency and ease of training, we restrict the networks to the anti-Hebbian solution and use the uniform readout as above. We find that the capacity depends primarily on the number of synapses, rather than on the number of pre- or postsynaptic neurons (Figures 5C and 5D), consistent with previous familiarity detection results (Bogacz and Brown, 2003). To estimate the scaling, we compute a linear least-squares fit of $\log(R_{\max})$ as a function of $\log(Nd)$. Empirically, we find that the capacity of the network scales as

$$R_{\max}\approx 0.10(Nd)^{0.79}$$

which is sublinear in the number of plastic input synapses to the hidden layer, Nd.

In contrast, the model of Bogacz and Brown (2003) for the non-continual task has a capacity that is linear in the number of synapses. To determine whether the difference between the empirical performance of HebbFF and the Bogacz-Brown model reflects a fundamental limitation in the feedforward architecture, we developed an idealized version of the model (Figure 6A) that we could study analytically (STAR Methods).

Idealized model and theoretical capacity

We noted above that the limiting behavior of the network at maximum capacity appears to have $\boldsymbol{W}_1$ activate just a single unit for memory storage. We build this limiting behavior into the idealized model through a specific choice of $\boldsymbol{W}_1$ and $\boldsymbol{b}_1$, set by design rather than through a training procedure. Specifically, we use the first $n \ll d$ components of $\boldsymbol{x}(t)$ as an identifier by choosing the first n columns of $\boldsymbol{W}_1$ so that a unique hidden unit is activated by each possible n -bit combination of these components (STAR Methods) and set the remaining columns of $\boldsymbol{W}_1$ to zero. To simplify the model, we do not allow plasticity to operate on the inputs from these bits and set the first n columns of $\boldsymbol{A}(t)$ to zero (Figure 6A). This isolates the addressing function of the fixed matrix from the memory storage. Furthermore, instead of a sigmoid nonlinearity for the hidden units, we use a Heaviside step function $\Theta(\cdot)$. Thus, the hidden layer in the idealized model is governed by

$$\boldsymbol{h}(t)=\Theta((\boldsymbol{W}_1+\boldsymbol{A}(t))\boldsymbol{x}(t)+\boldsymbol{b}_1)$$

sufficiently that its signal is lost. In the case of familiar stimuli presented at shorter delay intervals, $R=14$ or 40, there is an additional mode due to the correlation between the input $\boldsymbol{x}(t)$ and its copy $\boldsymbol{x}(t-R)$ previously stored in the plastic matrix $\boldsymbol{A}(t)$.

(M–O) Distributions of the total input current into the hidden layer on test sets with $R=14$, 40, or 100. Only the values above zero cause high firing rates after applying the logistic sigmoid nonlinearity. As all the input currents are low for familiar stimuli (green) for small values of R, there is repetition suppression.

(P–R) Correlation between the input current into the hidden layer from static and plastic synapse components at each of 20 consecutive time points. Asterisks indicate output response errors. For sufficiently small R, the input currents are more anti-correlated for familiar stimuli (black circles) than for novel. Combined with the distributions of input currents, this indicates that the units receiving positive input current from the static synapses receive negative input current from the plastic synapses.

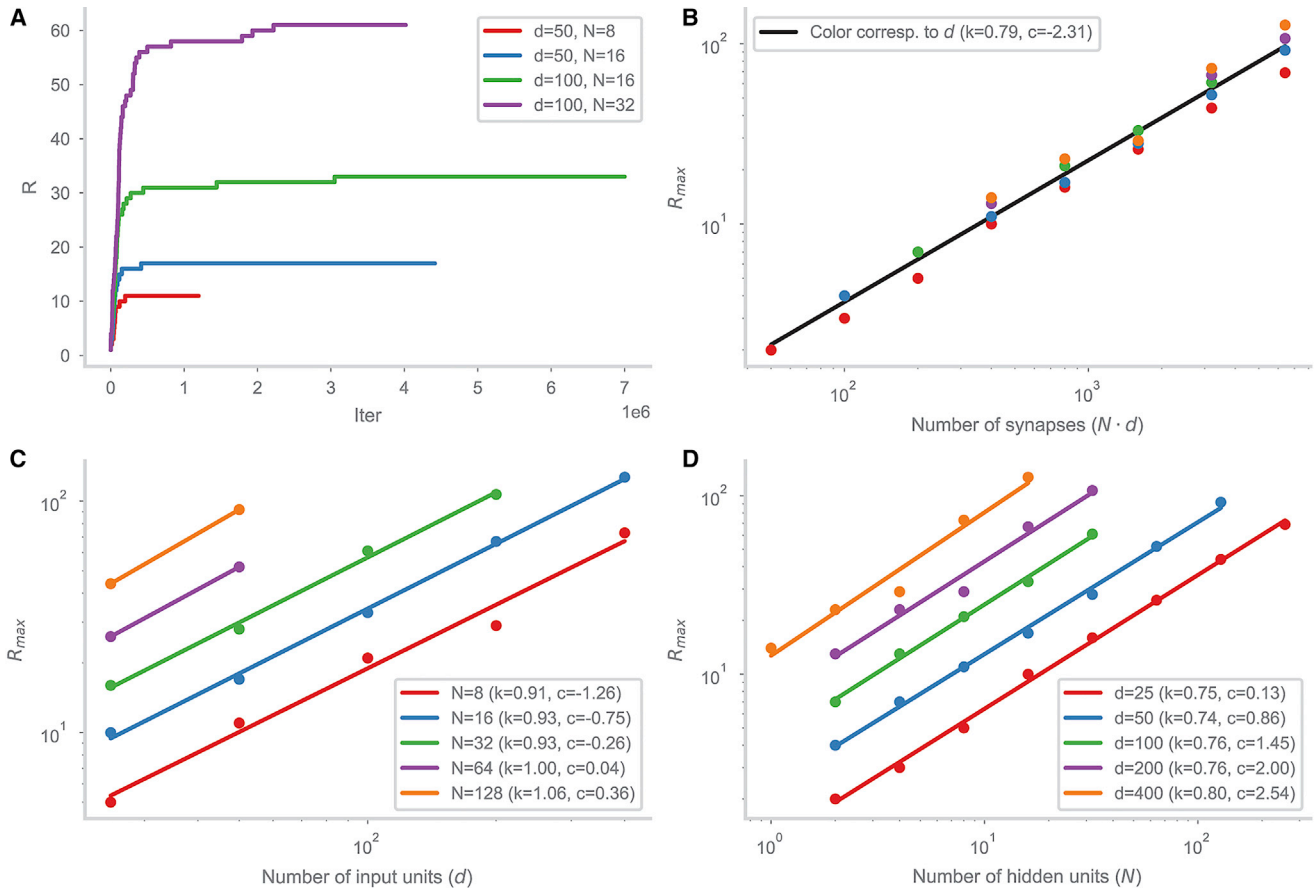

Figure 5. Curriculum training and empirical capacity

(A) The value of R used over the course of curriculum training for four different network sizes.

(B) The final value of R after curriculum training (i.e., network capacity) as a function of the number of plastic synapses in the network, plotted on a log-log scale. The color corresponds to the number of input units as in (D). The least-squares fit (slope k, bias c) indicates that the empirical network capacity scales sublinearly with the number of synapses.

(C and D) Capacity plotted as a function of the input dimension d and hidden layer size N, respectively, holding the other one constant. It primarily depends only on the number of synapses, rather than on the hidden or input layer sizes.

For the nonzero entries of $\boldsymbol{A}(t)$, plasticity is the same as in the trained model, but because the Heaviside function does not depend on the scale of the input, we can set the plasticity rate to $\eta = -1$ without loss of generality. The optimal synaptic decay rate λ is computed analytically. A stimulus is considered familiar if all hidden unit activities are zero and novel otherwise (STAR Methods).

This idealized model exhibits similar behavior to HebbFF. We can fit the analytic functional form of the true and false positive probabilities computed from the idealized model (Figure 6B) to the corresponding probabilities of HebbFF (Figure 6C). Furthermore, the histograms of inputs to the hidden layer are qualitatively similar: $\boldsymbol{W}_1\boldsymbol{x}(t) + \boldsymbol{b}_1$ has the same multi-modal distribution with more prominent peaks in the middle (Figures 4I and 6D; STAR Methods), a bimodal distribution of $\boldsymbol{A}(t)\boldsymbol{x}(t)$ with a large symmetric central peak and a smaller one corresponding to the familiarity signal (Figures 4J and 6E), and a similar distribution of the total input current $(\boldsymbol{W}_1 + \boldsymbol{A}(t))\boldsymbol{x}(t) + \boldsymbol{b}_1$ (Figures 4O and 6F). From this, we conclude that the memory storage and readout mechanisms are analogous in the meta-learned HebbFF network and the idealized model.

Finally, the memory capacity of the idealized model can be computed analytically (STAR Methods). As in the Bogacz-Brown model (2003), the capacity is proportional to the number of synapses Nd. There are several possible reasons for the discrepancy between the analytic capacity, as well as that of the Bogacz-Brown model, relative to the empirical capacity for HebbFF. First, the idealized HebbFF model uses a dedicated set of synapses through the fixed $\boldsymbol{W}_1$ matrix, and the Bogacz-Brown model selects the units that have the highest input current implicitly through inhibitory competition. Both of these are dedicated addressing functions for the hidden layer, but meta-learned HebbFF must multiplex this functionality with memory storage, leading to correlations between the hidden-layer input currents from the plastic and fixed synapse components (Figure S4A).

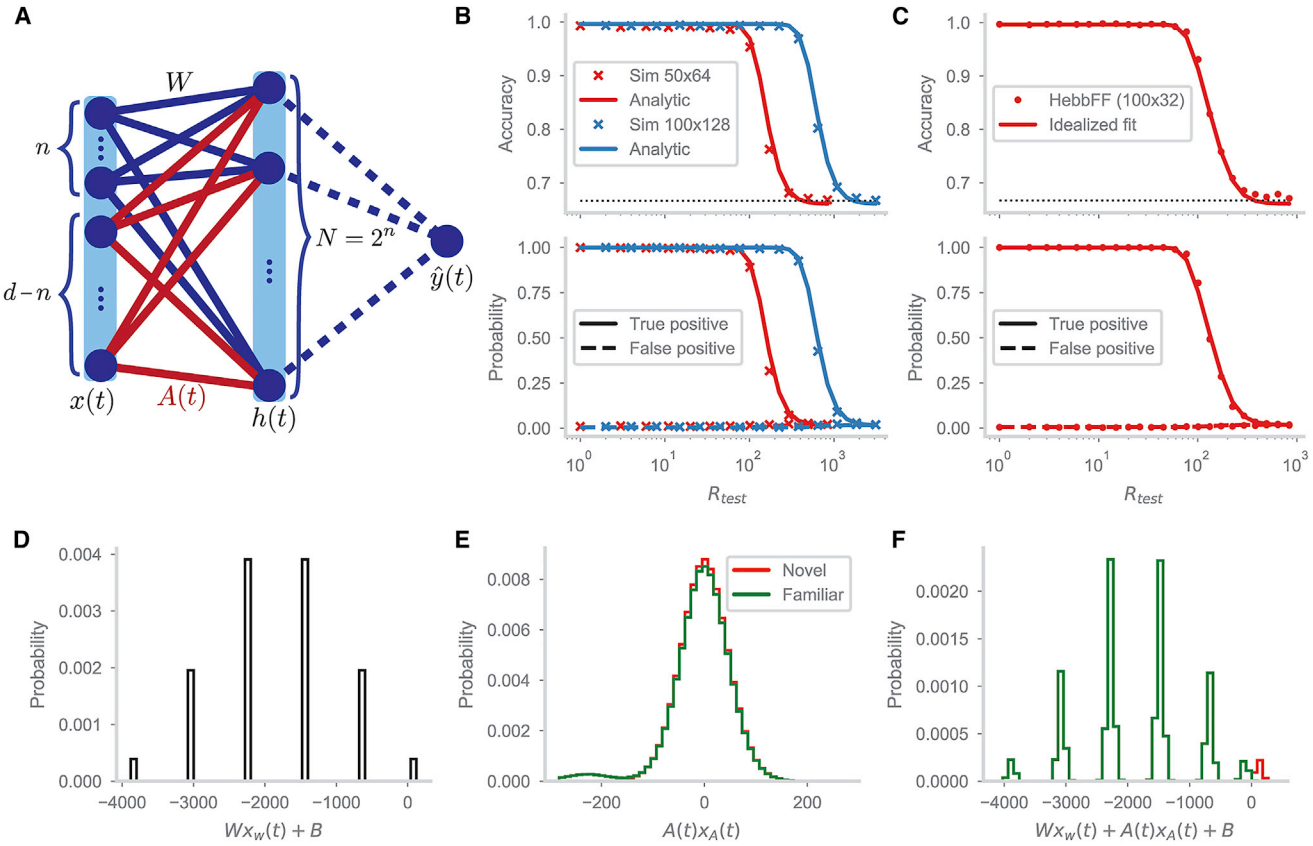

Figure 6. Idealized model

(A) The idealized HebbFF network architecture. The input $\boldsymbol{x}(t)$ is effectively split into two sections of size n and $D = d - n$ that serve as inputs into separate static and plastic synaptic matrices $\boldsymbol{W}_1$ and $\boldsymbol{A}(t)$, respectively (STAR Methods). The hidden layer size is $N = 2^n$ and output is $\hat{y}(t) = 1$ whenever any of the hidden units is active .

(B) The analytic calculation of network performance (solid line) matches simulation results for the idealized network (x's), shown for two different network sizes (red, blue).

(C) A least-squares fit of the analytic performance curve of the idealized network to a trained HebbFF network of the same size for two network sizes. The idealized network has similar performance to the HebbFF model if its decay rate and bias are scaled appropriately: $\lambda \approx 0.986$, $b_1 \approx -4.771$ (for all units) for $d = 200$, $N = 32$, and $\lambda \approx 0.993$, $b_1 \approx -4.771$ for $d = 200$, $N = 32$.

(D–F) Same as Figures 4L, 4J, and 4M but for the idealized network ($D = 400$, $N = 32$, $R = 300$).

In addition, replacing the logistic function with a Heaviside function means that familiar stimuli truly generate no plasticity in the idealized model, reducing overwriting at the cost of not reinforcing partially decayed memories (Figures S4B and S4C). For the same reason, in contrast to HebbFF, the idealized model achieves maximal plasticity for any suprathreshold level of input to a hidden layer unit.

Finally, training the HebbFF model may lead to specialized solutions for small d and N that have better performance than that predicted by the asymptotic analysis. Similarly, training may not converge to the optimal solution for large d and N because it requires the use of very long repeat intervals R. This means the dataset size T must be very large to include a sufficient number of familiar examples, which may lead to practical issues such as vanishing gradients. Thus, the empirical capacity may scale sublinearly with the number of synapses because of over-performance at low R, under-performance at high R, or both.

HebbFF recapitulates neural data from inferotemporal cortex

We next compare the optimized HebbFF model with experimental results. Meyer and Rust (2018) recorded neurons from the inferotemporal (IT) cortex of monkeys performing familiarity detection and compared the quality of two decoders in predicting behavior from neural data as a function of neural population size. The authors considered a "spike count classifier" (SCC) decoder, which amounts to comparing a simple average of neuronal firing rates to a threshold, as well as a Fisher linear discriminant (FLD), which instead considers a weighted average, with weights computed from the data.

We perform a similar analysis. We first construct an FLD decoder of the hidden unit firing rates and rank the units in reverse order of their FLD readout weights (i.e., units with the most negative weights are top ranked; STAR Methods). We then consider decoders that use increasingly larger subsets of hidden units, adding them according to their ranking. As in the

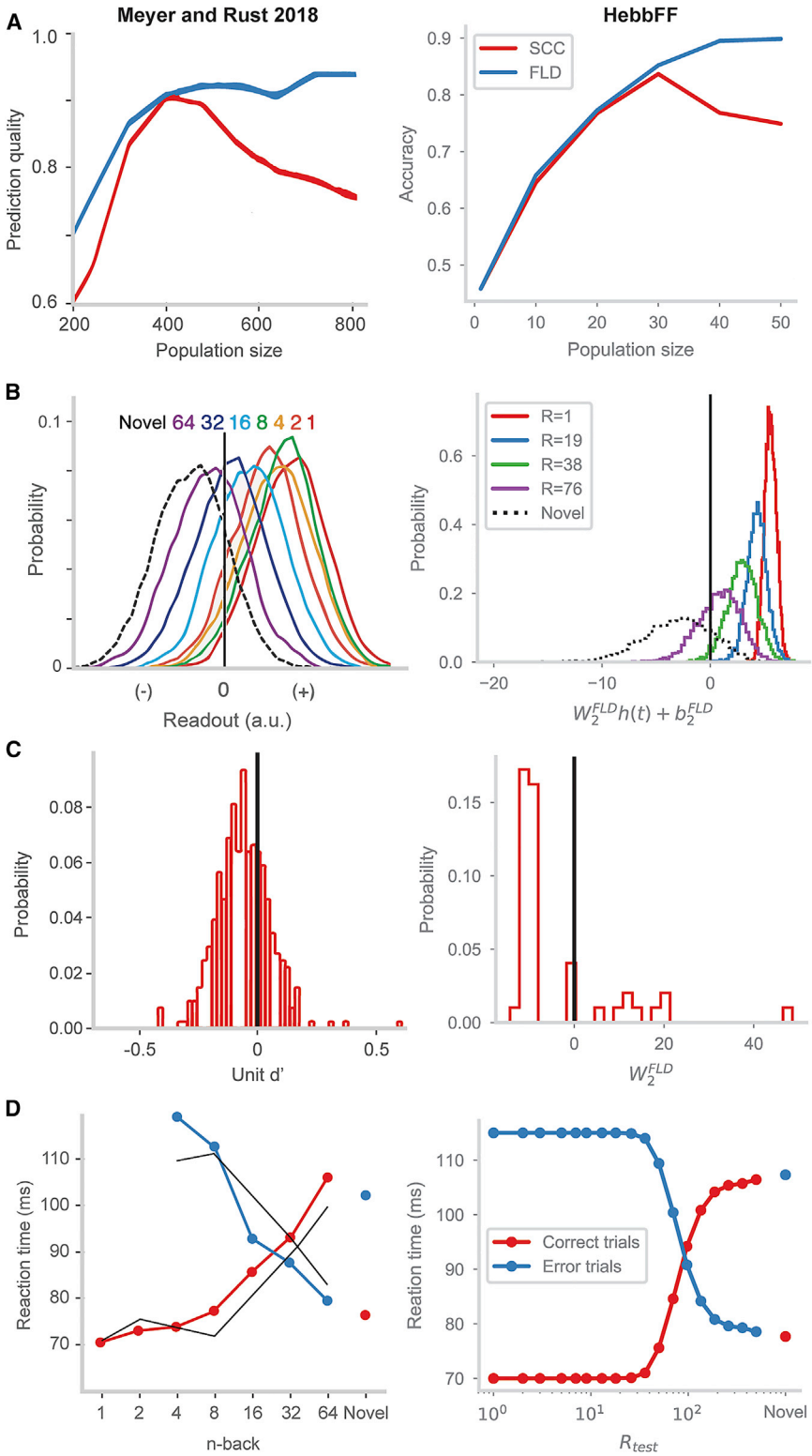

Figure 7. Comparison with IT cortex data

(A) Left: neurons from the IT cortex used to predict the behavioral outputs of a monkey performing continual familiarity detection, decoded using the Fisher linear discriminant (FLD; blue) or spike count classifier (SCC; red). Right: units from the hidden layer of a trained HebbFF network (trained with unconstrained $\boldsymbol{W}_2$) used to decode familiarity with SCC or FLD. In both cases, the number of neurons/units available to the decoder was varied, added in order of increasing FLD weight. While the FLD decoder accuracy saturates, the SCC decoder accuracy peaks and declines as more neurons/units are included in the decoder.

(B) Distribution the FLD decoder output for IT cortex neurons (left) and HebbFF hidden units (right) for familiar stimuli at varying delay intervals. In both cases, the distribution shifts toward lower values as delay interval increases. For HebbFF, the distribution gets narrower for shorter delay intervals because of saturation in the hidden layer units.

(C) Distribution of the FLD decoder weights for decoding IT cortex data (left) or HebbFF hidden unit activity (right). In both cases, the majority of output weights are negative.

(D) Left: measured reaction time as a function of delay interval for correct and error trials (red, blue curves) in monkeys performing the continual familiarity detection task. Black lines indicate reaction times predicted using strength theory analysis. Right: HebbFF predicted reaction times using analogous strength theory analysis (STAR Methods). Both result in a qualitatively similar x-shaped pattern. Plots on the left side of (A)–(D) adapted from Meyer and Rust (2018).

experimental data, performance saturates for the FLD and declines for the SCC readout beyond a certain number of decoded units (Figure 7A), as some of the units do not provide a reliable signal of familiarity. Including them hurts performance of the SCC decoder, but because the FLD readout weight for these units is close to zero, they do not alter its familiarity detection performance.

The unreliable units occur in HebbFF because of suboptimal training. In the IT cortex, they are possibly due to performing an unrelated task. We explicitly consider the latter scenario by training our network to perform binary classification in parallel with familiarity detection, reading out both signals from the same set of hidden units (STAR Methods). As a result, two sub-populations of hidden units emerge: one for classification and one for familiarity detection (Figure S5E), the classification units degrading the SCC readout as expected. All other results in this section remain unchanged (Figures S5A–S5D).

Comparing the experimental and model distributions of readout activity shows a qualitatively similar pattern for outputs to novel and familiar stimuli (Figure 7B). Both distributions shift

toward smaller values as R increases and familiar stimuli begin to appear novel. The fact that the distribution of outputs becomes narrower for HebbFF as R decreases, unlike in the data, may be due to repetition suppression causing hidden units to have near zero responses for highly familiar (low R) stimuli, thus causing the readout distribution to cluster around its minimal value. On the other hand, biological neurons that exhibit repetition suppression may never be fully silenced, for example, if it takes multiple repetitions to achieve maximal familiarity or if neurons are multiplexed with another task that requires a baseline level of activity. Furthermore, as in the data, the distribution of readout weights is biased toward negative values (Figure 7C).

Finally, using a similar "strength theory" analysis as in the experimental results (Meyer and Rust, 2018; Murdock, 1985), which suggests that reaction times are inversely proportional to the distance of the readout from the threshold, we can qualitatively reproduce the x-shaped reaction time curves seen experimentally. We used the same proportionality constant determined experimentally to compute network "reaction times" (Figure 7D).

Familiarity detection of images

To validate the HebbFF model in a more realistic scenario, we evaluate its performance on images. As a stand-in for the processing done by the visual stream, we use a pre-trained convolutional neural network and down-sample its output, either by subsampling and binarizing (Figures 8A and 8B) or by introducing a trainable intermediate layer (Figure S6A). These two down-sampling approaches change the statistics of the inputs to HebbFF, either by introducing correlations (Figure 8C) or by having them be real-valued vectors (Figure S6B). Interestingly, in the latter case, the network learns an uncorrelated representation automatically (Figure S6C), which further supports storing uncorrelated stimuli.

These networks have similar features to those trained on uncorrelated binary vectors. For binarized inputs, the $\boldsymbol{W}_1$ matrix has a similar structure (Figures 4F and 8D), the hidden-layer activity is sparse (Figures 4C and 8E), and the hidden unit input current distributions have similar shapes (Figures 4I, 4J, 4M, 7B, and 8F–8I), but there is a slight drop in performance due to correlations (Figure 8J). For real-valued inputs, although its structure is different, the $\boldsymbol{W}_1$ matrix still acts as a addressing function to select a unique neuron in the hidden layer (Figure S6E) that is then suppressed for a familiar stimulus through the $\boldsymbol{A}(t)$ matrix (Figures S6G and S6H). The network maintains its generalization performance (Figure S6J). See STAR Methods for details.

DISCUSSION

In answer to the question of "how" memories are stored, we find that anti-Hebbian plasticity, in which neuronal co-activation causes synaptic depression (this may be also interpreted as potentiation of inhibitory synapses; Schulz et al., 2020), is a better storage mechanism for familiarity detection than Hebbian. An anti-Hebbian rule generalizes better, has a larger capacity, and is discovered by meta-learning more frequently and reliably. Although this result is consistent with previous work (Bogacz and Brown, 2003), the underlying reasons are different. Bogacz and Brown (2003) showed that in a non-continual version of the familiarity detection task, an anti-Hebbian plasticity rule leads to a larger storage capacity, although this advantage held only in the case of correlated inputs. In their case, the anti-Hebbian rule automatically suppresses common input features, effectively storing only the uncorrelated components, leading to an increased capacity. In contrast, anti-Hebbian HebbFF shows an advantage even for uncorrelated inputs in the continual task. This is due to an effective decrease in the number of plasticity events; a synaptic update is weak for a familiar stimulus because the postsynaptic activity is low, leading to smaller updates that are less disruptive to stored memories.

Equally important is the question of "where" memories are stored. HebbFF explicitly selects storage locations through an addressing function implemented by strong feedforward weights $\boldsymbol{W}_1$, independent of the previously stored memories $\boldsymbol{A}(t)$. By inducing hidden-layer activity (typically a single active neuron), $\boldsymbol{W}_1$ selects only those afferents for storing a novel memory. This is in contrast to implicit addressing through recurrent inhibition in a previous anti-Hebbian model (Bogacz and Brown, 2003) which selects 50% of hidden-layer neurons. Although much experimental and theoretical work has been devoted to elucidating the plasticity rules used in memory storage, our work highlights the equal importance of studying the addressing functions of neuronal circuits as well.

Critically, unlike classical models, these answers emerged from meta-learning. The architectural features were not due to decisions made by the modeler but rather discovered through optimization. Although our particular meta-learning algorithm, BPTT, does not easily map onto a biological mechanism, we can nevertheless interpret it as a stand-in for structural changes over long timescales: an addressing function developing in a newborn's brain over the first years of her life or a plasticity rule emerging within a species across generations. Evolutionary strategies for meta-learning (Confavreux et al., 2020; Jordan et al., 2021; Najarro and Risi, 2021) imply the latter interpretation. In contrast, the plasticity rule itself is a biologically realistic mechanism for learning over short timescales: seconds or minutes to store a memory that may be retrieved throughout a lifetime.

Thus, the HebbFF model predicts that there should be two populations of synapses: a small set of slow-varying or fixed synapses for addressing the memory neurons (the hidden layer of HebbFF) and a larger set of highly plastic synapses for encoding memories.

We also make a more quantitative experimental prediction. Although it is obvious that the true positive rate should decrease with longer delay intervals R, we also observe that the false positive rate slightly increases (Figure S7A). Neither the Hebbian mechanism nor the RNN trained on a single R show this behavior (Figures S7B and S7C). If biological networks implement familiarity detection through an anti-Hebbian plasticity mechanism, we expect to see the same effect. Note, however, that anti-Hebbian plasticity is merely sufficient, not necessary, for this result, so the converse may not be true (Figure S7D).

There are experimental results that the HebbFF model does not capture. For example, data from human subjects show a very slow decrease in performance as a function of R that begins

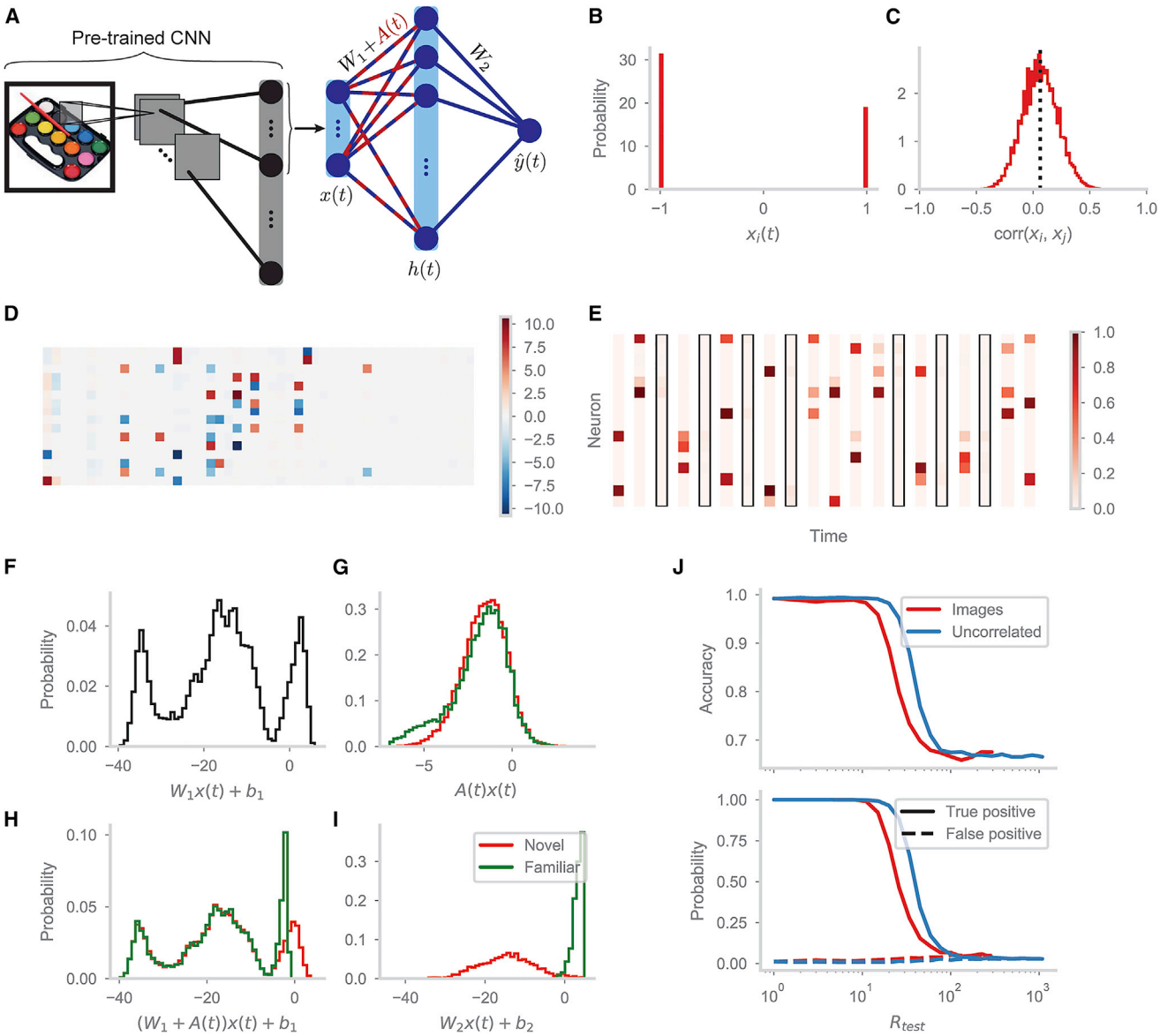

Figure 8. HebbFF performance on real-world images

(A) Network architecture for familiarity detection of real-world images. The activity of the penultimate layer of a convolutional neural network (ResNet18, pre-trained on ImageNet) is down-sampled and passed to the HebbFF network (d = 50, N = 16) for familiarity detection. Only the HebbFF portion of this network is trained, via curriculum training.

(B) Distribution of inputs $\boldsymbol{x}(t)$ to HebbFF. After down-sampling by extracting the first 50 units of the CNN, the activity is centered at zero and binarized.

(C) Histogram of the correlations between all pairs of input stimuli $\boldsymbol{x}(t)$. On average (vertical dashed line) the correlation is slightly positive.

(D–H) Same as Figures 4F, 4C, 4I, 4J, and 4M, respectively (R_{train} = R_{test} = 12).

(I) Distribution of network outputs $\hat{y}(t)$. for novel (red) and familiar (green) stimuli

(J) Generalization performance, compared with a network of the same size trained on uncorrelated binary random vectors, is lower because of correlations in the input images.

at relatively small value (Brady et al., 2008). In contrast, HebbFF has near perfect performance for all $R<R_{max}$, and then performance drops off quickly. However, it is likely that errors in the experiments do not reflect limitations on recognition memory but rather are due to factors such as fatigue and lack of attention that were not included in the model.

Finally, along with other recent applications of this technique, our work demonstrates the utility of meta-learning as a tool for neuroscience discovery. We used meta-learning to optimize a network architecture and plasticity rule that solves the continual familiarity detection task, contrasted it with an alternative suboptimal solution, and subsequently used analytic

methods to understand its mechanism. A similar approach can be used for other networks, plasticity rules, datasets, and tasks.

STAR★METHODS

Detailed methods are provided in the online version of this paper and include the following:

- KEY RESOURCES TABLE
- RESOURCE AVAILABILITY
  - Lead contact
  - Materials availability
  - Data and code availability
- METHOD DETAILS
  - Continual familiarity detection task
  - HebbFF and RNN training
  - Bogacz-Brown (Bogacz and Brown, 2003) model implementation
  - Training FLD and SCC decoders
  - Simultaneous classification and familiarity detection
  - Idealized model analytic capacity derivation
  - CNN preprocessing for familiarity detection of images

SUPPLEMENTAL INFORMATION

Supplemental information can be found online at https://doi.org/10.1016/j.neuron.2021.11.009.

ACKNOWLEDGMENTS

We thank Stefano Fusi, Ken Miller, Dmitriy Aronov, James Murray, Marcus Benna, SueYeon Chung, Juri Minxha, Taiga Abe, and Denis Turcu for helpful discussions. This research was supported by the NIH (T32-NS064929), National Science Foundation (NSF) NeuroNex award DBI-1707398, the Gatsby Charitable Foundation (GAT3708), and the Simons Collaboration for the Global Brain. G.R.Y. was additionally supported by the Simons Foundation. We acknowledge computing resources from Columbia University's Shared Research Computing Facility project, which is supported by NIH Research Facility Improvement Grant 1G20RR030893-01, and associated funds from the New York State Empire State Development, Division of Science Technology and Innovation (NYSTAR), contract C090171, both awarded April 15, 2010.

AUTHOR CONTRIBUTIONS

Conceptualization, D.T., G.R.Y., and L.F.A.; Methodology, D.T., G.R.Y., and L.F.A.; Software, D.T.; Investigation, D.T.; Writing – Original Draft, D.T.; Writing – Review & Editing, D.T., G.R.Y., and L.F.A.; Visualization, D.T.; Supervision, G.R.Y. and L.F.A.; Funding Acquisition, L.F.A.

DECLARATION OF INTERESTS

The authors declare no competing interests. L.F.A. serves on the advisory board of *Neuron*.

Received: April 7, 2021
Revised: August 24, 2021
Accepted: November 10, 2021
Published: December 2, 2021

REFERENCES

Androulidakis, Z., Lulham, A., Bogacz, R., and Brown, M.W. (2008). Computational models can replicate the capacity of human recognition memory. Network *19*, 161–182.

Ba, J., Hinton, G., Mnih, V., Leibo, J.Z., and Ionescu, C. (2016). Using fast weights to attend to the recent past. arXiv, arXiv:1610.06258 https://arxiv.org/abs/1610.06258.

Beaulieu, S., Frati, L., Miconi, T., Lehman, J., Stanley, K.O., Clune, J., and Cheney, N. (2020). Learning to continually learn. arXiv, arXiv:2002.09571 https://arxiv.org/abs/2002.09571.

Bengio, Y., Bengio, S., and Cloutier, J. (1991). Learning a synaptic learning rule. In IJCNN-91-Seattle International Joint Conference on Neural Networks (Piscataway, NJ: Institute of Electrical and Electronics Engineers).

Bogacz, R., and Brown, M.W. (2003). Comparison of computational models of familiarity discrimination in the perirhinal cortex. Hippocampus *13*, 494–524.

Brady, T.F., Konkle, T., Alvarez, G.A., and Oliva, A. (2008). Visual long-term memory has a massive storage capacity for object details. Proc. Natl. Acad. Sci. U S A *105*, 14325–14329.

Confavreux, B., Agnes, E.J., Zenke, F., Lillicrap, T., and Vogels, T.P. (2020). A meta-learning approach to (re)discover plasticity rules that carve a desired function into a neural network. bioRxiv. https://doi.org/10.1101/2020.10.24.353409.

Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Li, F.-F. (2009). ImageNet: a large-scale hierarchical image database. In 2009 IEEE Conference on Computer Vision and Pattern Recognition (Piscataway, NJ: Institute of Electrical and Electronics Engineers), pp. 248–255.

Fusi, S., Drew, P.J., and Abbott, L.F. (2005). Cascade models of synaptically stored memories. Neuron *45*, 599–611.

Graves, A., Wayne, G., and Danihelka, I. (2014). Neural Turing machines. arXiv, arXiv:1410.5401 https://arxiv.org/abs/1410.5401.

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S.G., Grefenstette, E., Ramalho, T., Agapiou, J., et al. (2016). Hybrid computing using a neural network with dynamic external memory. Nature *538*, 471–476.

Grill-Spector, K., Henson, R., and Martin, A. (2006). Repetition and the brain: neural models of stimulus-specific effects. Trends Cogn. Sci. *10*, 14–23.

Gu, K., Greydanus, S., Metz, L., Maheswaranathan, N., and Sohl-Dickstein, J. (2019). Meta-learning biologically plausible semi-supervised update rules. bioRxiv. https://doi.org/10.1101/2019.12.30.891184.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. arXiv, arXiv:1512.03385 https://arxiv.org/abs/1512.03385.

Hochreiter, S., and Schmidhuber, J. (1997). Long short-term memory. Neural Comput. *9*, 1735–1780.

Hopfield, J.J. (1982). Neural networks and physical systems with emergent collective computational abilities. Proc. Natl. Acad. Sci. U S A *79*, 2554–2558.

Ji-An, L., Stefanini, F., Benna, M.K., and Fusi, S. (2019). Face familiarity detection with complex synapses. bioRxiv. https://doi.org/10.1101/854059.

Jordan, J., Schmidt, M., Senn, W., and Petrovici, M.A. (2021). Evolving to learn: discovering interpretable plasticity rules for spiking networks. arXiv, arXiv:2005.14149 https://arxiv.org/abs/2005.14149.

Kazanovich, Y., and Borisyuk, R. (2021). A computational model of familiarity detection for natural pictures, abstract images, and random patterns: combination of deep learning and anti-Hebbian training. Neural Netw. *143*, 628–637.

Kingma, D.P., and Ba, J. (2017). Adam: a method for stochastic optimization. arXiv, arXiv:1412.6980 https://arxiv.org/abs/1412.6980.

Lehky, S.R., and Tanaka, K. (2016). Neural representation for object recognition in inferotemporal cortex. Curr. Opin. Neurobiol. *37*, 23–35.

Lim, S., McKee, J.L., Woloszyn, L., Amit, Y., Freedman, D.J., Sheinberg, D.L., and Brunel, N. (2015). Inferring learning rules from distributions of firing rates in cortical neurons. Nat. Neurosci. *18*, 1804–1810.

Lindsey, J., and Litwin-Kumar, A. (2020). Learning to learn with feedback and local plasticity. arXiv, arXiv:2006.09549 https://arxiv.org/abs/2006.09549.

Lueschow, A., Miller, E.K., and Desimone, R. (1994). Inferior temporal mechanisms for invariant object recognition. Cereb. Cortex *4*, 523–531.

Metz, L., Maheswaranathan, N., Cheung, B., and Sohl-Dickstein, J. (2019). Meta-learning update rules for unsupervised representation learning. arXiv, arXiv:1804.00222 https://arxiv.org/abs/1804.00222.

Meyer, T., and Rust, N.C. (2018). Single-exposure visual memory judgments are reflected in inferotemporal cortex. eLife *7*, e32259.

Miconi, T., Rawal, A., Clune, J., and Stanley, K.O. (2019). Backpropamine: training self-modifying neural networks with differentiable neuromodulated plasticity. arXiv, arXiv:2002.10585 https://arxiv.org/abs/2002.10585.

Miller, E.K., Li, L., and Desimone, R. (1991). A neural mechanism for working and recognition memory in inferior temporal cortex. Science *254*, 1377–1379.

Murdock, B.B., Jr. (1985). An analysis of the strength-latency relationship. Mem. Cognit. *13*, 511–521.

Najarro, E., and Risi, S. (2021). Meta-learning through Hebbian plasticity in random networks. arXiv, arXiv:2007.02686 https://arxiv.org/abs/2007.02686.

Norman, K.A., and O'Reilly, R.C. (2003). Modeling hippocampal and neocortical contributions to recognition memory: a complementary-learning-systems approach. Psychol. Rev. *110*, 611–646.

Parisi, G. (1986). A memory which forgets. J. Phys. Math. Gen. *19*, L617–L620.

Rutishauser, U., Ye, S., Koroma, M., Tudusciuc, O., Ross, I.B., Chung, J.M., and Mamelak, A.N. (2015). Representation of retrieval confidence by single neurons in the human medial temporal lobe. Nat. Neurosci. *18*, 1041–1050.

Schulz, A., Miehl, C., Berry, M.J., and Gjorgjieva, J. (2020). The generation of cortical novelty responses through inhibitory plasticity. bioRxiv. https://doi.org/10.1101/2020.11.30.403840.

Sohal, V.S., and Hasselmo, M.E. (2000). A model for experience-dependent changes in the responses of inferotemporal neurons. Network *11*, 169–190.

Standing, L. (1973). Learning 10,000 pictures. Q. J. Exp. Psychol. *25*, 207–222.

Thrun, S., and Pratt, L. (2012). Learning to Learn (New York: Springer Science & Business Media).

Tyulmankov, D., Fang, C., Vadaparty, A., and Yang, G.R. (2021). Biological learning in key-value memory networks. arXiv, arXiv:2110.13976 https://arxiv.org/abs/2110.13976.

Xiang, J.-Z., and Brown, M.W. (1998). Differential neuronal encoding of novelty, familiarity and recency in regions of the anterior temporal lobe. Neuropharmacology *37*, 657–676.

STAR★METHODS

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Software and algorithms		
Python	Python Software Foundation	https://www.python.org/
PyTorch	Facebook, Inc.	https://pytorch.org
Custom code	This paper	https://github.com/dtyulman/hebbff (https://doi.org/10.5281/zenodo.5659610)

RESOURCE AVAILABILITY

Lead contact

Further information and requests should be directed to and will be fulfilled by the lead contact, Danil Tyulmankov (dt2586@columbia.edu)

Materials availability

This study did not generate new unique materials.

Data and code availability

All original data in this work was programmatically generated. The code for data generation, as well as the network and analysis can be found at https://github.com/dtyulman/hebbff (https://doi.org/10.5281/zenodo.5659610). Any additional information required to reanalyze the data reported in this paper is available from the lead contact upon request.

METHOD DETAILS

Continual familiarity detection task

We consider a continual familiarity detection task (Figure 1A) in which a stream of stimuli is presented to a network. With probability $1-p$, the stimulus at time t is chosen as a randomly generated d -dimensional binary vector $\boldsymbol{x}(t)$, where each component is either $+1$ or -1 (note that for sufficiently large d, spurious chance repeats are extremely unlikely). With probability p, the stimulus is a copy of the stimulus presented R time steps ago, so that $\boldsymbol{x}(t) = \boldsymbol{x}(t-R)$. However, we ensure that a stimulus is repeated at most once so, if $\boldsymbol{x}(t-R)$ is already a repeat, i.e., $\boldsymbol{x}(t-R) = \boldsymbol{x}(t-2R)$, a new $\boldsymbol{x}(t)$ is generated. As a result, the fraction of novel stimuli, which we call f, is not equal to $1-p$, but rather $f = (1/1+p)$. We use $p = (1/2)$, so $f = (2/3)$. The output of the network should be $y(t) = 0$ if $\boldsymbol{x}(t)$ is novel and $y(t) = 1$ if it is familiar, i.e., has appeared previously.

The accuracy of the network (P_{correct}, the probability of correctly responding to a stimulus) depends on two factors: the true positive rate (P_{TP}, the probability of correctly reporting a repeated stimulus as "familiar"), and the false positive rate (P_{FP}, the probability of incorrectly reporting a novel stimulus as "familiar"). These two factors are weighted by the fraction of novel stimuli f, so that $P_{\text{correct}} = (1-f)P_{TP} + f(1-P_{FP})$. Through our choice of loss function (next section), we are effectively training the networks to maximize accuracy, so the "chance" level performance is f (for $f > (1/2)$), which a network can achieve by reporting all stimuli as novel ($P_{TP} = P_{FP} = 0$).

In our paradigm, a given dataset has a single repeat interval R, which differs slightly from previously studied experimental paradigms (Brady et al., 2008; Meyer and Rust, 2018). However, we evaluate performance on multiple datasets with various values of R. For testing, this is analogous to evaluating a single dataset with multiple repeat intervals and computing accuracy for each interval separately. We use this approach because it allows us to test generalization by training on one value of R and testing on others. It also allows us to train the network to its maximal capacity by gradually increasing R during "curriculum training," and simplifies analytic calculations.

HebbFF and RNN training

To set the fixed HebbFF parameters $\boldsymbol{W}_1, \boldsymbol{b}_1, \boldsymbol{W}_2, \boldsymbol{b}_2, \lambda, \eta$, as well as the RNN weight and bias matrices, we use the PyTorch implementation of the Adam optimizer with the suggested default hyperparameters (Kingma and Ba, 2017). For a single trial, we use a dataset containing T stimuli, with familiar ones appearing at a repeat interval R. We present stimuli to the network sequentially, and compute the binary cross-entropy loss

$$\mathcal{L} = \frac{1}{T}\sum_{t=1}^{T} y(t)\log\hat{y}(t) + (1-y(t))\log(1-\hat{y}(t))$$

Since this is a dynamic task (the state of the network at time $t+1$ depends on the state at time t, either through recurrent activity in the RNN or through ongoing plasticity in HebbFF), backpropagation through time is used to compute the gradient of the loss with respect to the parameters.

For each trial, we either use the same pre-generated length- T dataset, or we generate a new length-T dataset using the same repeat interval R. We refer to the latter case as the "infinite data" training regime since the sample space is much larger than the network would explore during training. Note that in the infinite data regime, we do not consider a validation dataset, since the training set is new every time and the training accuracy is therefore the same as the validation accuracy. In both cases, one trial corresponds to one step of gradient descent. To train the HebbFF network, the plastic matrix $\boldsymbol{A}(t)$ is reset to a matrix of zeros at the start of each trial. Similarly, when training the RNN, hidden unit activity is reset to zero. In practice, the plastic matrix of HebbFF reaches its steady state distribution quickly and the transient does not contribute significantly to the gradient, so any reasonable initialization can be used.

Bogacz-Brown (Bogacz and Brown, 2003) model implementation

To validate it on the (non-continual) two-alternative-forced-choice (2AFC) familiarity detection task, we implement the anti-Hebbian model as described by Bogacz and Brown (Bogacz and Brown, 2003), with the exception that the distribution of weights in the plastic weight matrix must be normalized such that its variance is equal to $(1/N)$, rather than unit variance as stated in the paper. In the encoding phase, the network is presented a sequence of P random patterns. In the testing phase, it is shown the original P patterns, as well as P novel ones. Critically, there are no plastic updates in the testing phase. A stimulus is reported as "familiar" if the output unit activity is below the mean across all $2P$ test patterns and "novel" otherwise. We see that this model performs well on the 2AFC task with a range of plasticity rates η (Figure S2A), so we arbitrarily choose $\eta = 0.7$ to test its performance on the continual task.

The continual task, unlike the 2AFC task, does not have an equal proportion of novel and familiar stimuli since we ensure that a stimulus is repeated at most once. So, we set the readout threshold such that an item is considered novel if it is in the f^{th} quantile of output unit activity for that trial, where f is the fraction of novel stimuli in the trial. This ensures that the fraction of stimuli reported as "novel" is equal to the true fraction of novel stimuli. In the case of equal proportions of novel and familiar stimuli, this reduces to the threshold being equal to the mean of the output unit activity for that trial.

Finally, note that unlike in the 2AFC task (Figure S2A), the performance of this model does not go to chance levels for large dataset sizes T in the continual task (Figure 3D). Rather, the true positive rate goes to zero and the false positive rate is ≈ 0.5, so accuracy is ≈ 0.33. The reason for this difference is that the second presentation of a stimulus in the continual task causes an additional plasticity event, unlike the 2AFC task where the test phase is offline. As a result, for datasets much larger than the network capacity $T \gg P^*$, the output unit activity for familiar stimuli becomes larger than the activity for novel stimuli (Figure S2B).

Training FLD and SCC decoders

To construct the Fisher linear discriminant (FLD) and spike count classifier (SCC) decoders, we first generate a dataset of length $T = 1000$. To better match the experimental dataset (Meyer and Rust, 2018), we use multiple values of R in this single stream. For each familiar stimulus, the value of R is drawn uniformly at random from 34 unique values, log-spaced from 1 to 100 (in practice, the results are qualitatively the same regardless of the number of items, the range, or whether the spacing is linear or logarithmic). We evaluate the trained network on this dataset and use the firing rates of the hidden layer to perform analyses analogous to those reported in (Meyer and Rust, 2018).

We compute the readout weight and bias terms for the FLD decoder as

$$\boldsymbol{W}_2^{\text{FLD}} = \boldsymbol{\Sigma}^{-1}(\bar{\boldsymbol{h}}_{\text{nov}} - \bar{\boldsymbol{h}}_{\text{fam}}),\ b_2^{\text{FLD}} = -\boldsymbol{W}_2^{\text{FLD}} \cdot \frac{1}{2}(\bar{\boldsymbol{h}}_{\text{nov}} + \bar{\boldsymbol{h}}_{\text{fam}})$$

where $\bar{\boldsymbol{h}}_{\text{nov}}$ and $\bar{\boldsymbol{h}}_{\text{fam}}$ are the average firing rates of the hidden layer for novel and familiar stimuli, respectively, and the mean covariance matrix is calculated as

$$\boldsymbol{\Sigma} = \frac{\boldsymbol{\Sigma}_{\text{fam}} + \boldsymbol{\Sigma}_{\text{nov}}}{2}$$

where $\boldsymbol{\Sigma}_{\text{fam}}$ and $\boldsymbol{\Sigma}_{\text{nov}}$ are the covariance matrices of the firing rates of the hidden layer for familiar and novel stimuli, respectively. The SCC decoder is a simple weighted average

$$\boldsymbol{W}_2^{\text{SCC}} = \frac{1}{N}(\bar{\boldsymbol{h}}_{\text{nov}} - \bar{\boldsymbol{h}}_{\text{fam}}),\ b_2^{\text{SCC}} = -\boldsymbol{W}_2^{\text{SCC}} \cdot \frac{1}{2}(\bar{\boldsymbol{h}}_{\text{nov}} + \bar{\boldsymbol{h}}_{\text{fam}})$$

To get the ranking of the units for both decoders, we sort their readout weights and consider the most negative weights as the highest ranked. Note that for both decoders, the sign of the weights is flipped compared to (Meyer and Rust, 2018), and high-ranked units have the most negative weights rather than positive. This is due to the fact that we ask the network to label familiar stimuli as $y(t) = 1$, whereas (Meyer and Rust, 2018) readout a familiar stimulus as $y(t) = 0$. The two cases are symmetric and this does not change the results.

Simultaneous classification and familiarity detection

IT cortex encodes object identity as well as familiarity (Lehky and Tanaka, 2016; Lueschow et al., 1994). To match this dual functionality, we augment familiarity detection with object classification. We first create a large pool of random vectors and randomly assign a binary label to each one. We then generate a familiarity detection dataset as before, except that each novel input is drawn from this pool (without replacement) rather than being generated anew. In addition to the scalar readout of familiarity, the network must now report the class of the stimulus through a second binary output. Critically, both outputs are read out from the same hidden layer activity.

To train the HebbFF network on the augmented familiarity detection/object classification task, we simply sum the cross-entropy losses from the classifier and familiarity output units:

$$\mathcal{L} = \frac{1}{2T}\sum_{t=1}^{T}\sum_{a=1}^{2} y_a(t)\log\widehat{y}_a(t) + (1-y_a(t))\log(1-\widehat{y}_a(t))$$

For every trial, we draw a new dataset from the pre-generated pool of stimuli. The class of each stimulus remains the same across datasets, but the ordering and repeats are chosen randomly each time. Although the network will have seen all of the stimuli during training in order to learn their classes, we can test generalization performance on the familiarity subtask by varying R and generating previously unseen permutations of the stimuli.

The augmented task could be solved by having all the neurons multiplexed to encode both familiarity and object identity (Meyer and Rust, 2018). Alternatively, the neurons could split into two subpopulations, one of which detects familiarity and the other classifies objects (Rutishauser et al., 2015). We find that the HebbFF model converges to this second solution, an even split between familiarity and classifier units, as evident from inspecting the $\boldsymbol{W}_1$ matrix (Figure 7E). Consistent with this, the capacity of the classifier-augmented HebbFF with 50 hidden units ($R_{\max} \approx 13$) is approximately the same as the original network with 25 units ($R_{\max} \approx 14$). In accord with this split, SCC decoder performance peaks in the split-task network when half of the top-ranked units are included (Figure S5D) because including units responsible for object identity but not familiarity degrades the familiarity readout. The other similarities to experimental results discussed in the previous section also hold for the task-augmented network (Figure S5).

Idealized model analytic capacity derivation

For notational simplicity, we only consider the nonzero submatrices of $\boldsymbol{W}_1$ and $\boldsymbol{A}(t)$, each of which acts on its corresponding subset of the input vector $\boldsymbol{x}(t)$. Thus, equivalently, input layer of the idealized network is a d -dimensional vector split into two parts $\boldsymbol{x}(t) = [\boldsymbol{x}_{\boldsymbol{W}}(t), \boldsymbol{x}_{\boldsymbol{A}}(t)]$, of dimension n and D respectively ($d = n+D$). The firing rate of the hidden layer is given by

$$\boldsymbol{h}(t) = \Theta(\boldsymbol{W}_1\boldsymbol{x}_{\boldsymbol{W}}(t) + \boldsymbol{A}(t)\boldsymbol{x}_{\boldsymbol{A}}(t) + \boldsymbol{b}_1)$$

for an $N\times n$ matrix $\boldsymbol{W}_1$, an $N\times D$ matrix $\boldsymbol{A}(t)$, and an $N\times 1$ vector $\boldsymbol{b}_1$. In other words, the firing rate of the i^{th} hidden unit is

$$h_i(t) = \Theta\left(\sum_{j=1}^{n} W_{ij}x_j(t) + \sum_{k=1}^{D} A_{ik}(t)x_{n+k}(t) + b\right) \qquad \text{(Equation 1)}$$

for $i = 1, \ldots, N$, where $\Theta(\cdot)$ is the Heaviside step function, i.e., $\Theta(z) = 0$ for $z<0$ and 1 for $z \geq 0$. We fix the value of b to be the same for all i. As before, the elements of $\boldsymbol{x}(t)$ are $+1$ or -1 with equal probability. We would like to specify the network parameters such that exactly one hidden neuron is active for a novel stimulus and none for familiar, which will serve as the familiarity readout mechanism.

The $N\times n$ matrix $\boldsymbol{W}_1$ is designed such that the vector $\boldsymbol{W}_1\boldsymbol{x}_{\boldsymbol{W}}(t)$ has exactly one maximal entry given any such $\boldsymbol{x}(t)$. Importantly, this matrix must act like a hash function such that different values of $\boldsymbol{x}_{\boldsymbol{W}}(t)$ result in different entries of $\boldsymbol{W}_1\boldsymbol{x}_{\boldsymbol{W}}(t)$ attaining the maximum value. One such $\boldsymbol{W}_1$ is one whose rows enumerate all of the binary length- n strings consisting of entries $+1$ and -1. This sets the number of rows N to be equal to the total number of such strings, $N = 2^n$. To set the overall scale of the input current (the term inside the nonlinearity), we scale this matrix by a factor K, to be determined later. For example, if $n = 3$,

$$\boldsymbol{W}_1 = K\begin{bmatrix} +1 & +1 & +1 \\ +1 & +1 & -1 \\ +1 & -1 & +1 \\ +1 & -1 & -1 \\ -1 & +1 & +1 \\ -1 & +1 & -1 \\ -1 & -1 & +1 \\ -1 & -1 & -1 \end{bmatrix}$$

Thus, we have $\sum_{j=1}^{n} W_{ij}x_j(t) = Kn$ for exactly one value of i, specifically the row where $W_{ij} = x_j(t)$ for all j. This is the unique maximal value and will correspond to a different row for each instance of $\boldsymbol{x_W}(t)$. Subsequently, $\sum_{j=1}^{n} W_{ij}x_j(t) = K(n-2)$ for n values of i, specifically those where $W_{ij} \neq x_j(t)$ for exactly one j, and so on. This structure explains the multi-modal distribution of $\boldsymbol{W}_1\boldsymbol{x_W}(t) + \boldsymbol{b}_1$ in Figure 6D and by extension that of $\boldsymbol{W}_1\boldsymbol{x}(t) + \boldsymbol{b}_1$ in Figures 4G–4I.

Assuming that the vector $\boldsymbol{A}(t)\boldsymbol{x_A}(t)$ is zero-mean with sufficiently small variance (this will be made rigorous shortly), we can now choose the scalar offset b such that exactly one element of $\boldsymbol{h}(t)$ is equal to 1 and all others are zero.

The $N \times D$ matrix $\boldsymbol{A}(t)$ is updated at every timestep by $\boldsymbol{A}(t+1) = \lambda\boldsymbol{A}(t) - \eta\boldsymbol{h}(t)\boldsymbol{x_A}(t)^{\mathsf{T}}$, where the plasticity rate η is now restricted to be positive, corresponding to an anti-Hebbian learning rule. Considering one entry in this matrix and unrolling this recurrence, we find that

$$\begin{aligned} A_{ik}(t+1) &= \lambda A_{ik}(t) - \eta h_i(t) x_{n+k}(t) \\ &= \lambda^{t+1} A_{ik}(0) - \eta \sum_{t'=0}^{t} \lambda^{t-t'} h_i(t') x_{n+k}(t') \\ &\approx -\eta \sum_{t'=0}^{t} \lambda^{t-t'} h_i(t') x_{n+k}(t') \end{aligned}$$

where the last equality holds if we assume that the network is in steady-state, so t is large, i.e., $t \to \infty$, and therefore $\lambda^{t+1} A_{ik}(0) \to 0$.

We can now consider the middle term of Equation 1, which we denote by $\varepsilon_i(t)$. We consider it as a random variable and compute its mean and variance. By definition, we have

$$\varepsilon_i(t) = \sum_{k=1}^{D} A_{ik}(t) x_{n+k}(t) = \sum_{k=1}^{D} \left(-\eta \sum_{t'=0}^{t-1} \lambda^{t-1-t'} h_i(t') x_{n+k}(t') \right) x_{n+k}(t)$$

$$= -\eta \sum_{t'=0}^{t-1} \lambda^{t-1-t'} h_i(t') \sum_{k=1}^{D} x_{n+k}(t') x_{n+k}(t) \qquad \text{(Equation 2)}$$

In the case where $\boldsymbol{x}(t)$ is novel, $x_{n+k}(t')$ and $x_{n+k}(t)$ are independent Bernoulli random variables that take on values ± 1 with probability $\frac{1}{2}$. Thus, $X_k(t') = x_{n+k}(t')x_{n+k}(t)$ is also a Bernoulli random variable with the same distribution, zero mean and unit variance, so

$$\varepsilon_i(t) = -\eta \sum_{t'=0}^{t-1} \lambda^{t-1-t'} h_i(t') \sum_{k=1}^{D} X_k(t')$$

Since the entries of $\boldsymbol{x}(t)$ are independent by definition, the $X_k(t')$ are also independent across k, so summing over these indices, the variances add. Therefore, $X(t') = \sum_{k=1}^{D} X_k(t')$ is a random variable with mean 0 and variance D, and

$$\varepsilon_i(t) = -\eta \sum_{t'=0}^{t-1} \lambda^{t-1-t'} h_i(t') X(t')$$

Next, we need the statistics of the term $h_i(t')$. Since it is a function of the random variable $\boldsymbol{x}(t)$, we also consider it as a random variable. Let f_{eff} denote the fraction of stimuli reported as "novel" by the network. Note that there are two ways for a network to report a stimulus as "novel" – by correctly identifying a novel stimulus ("true negative"), or incorrectly identifying a familiar one ("false negative") – so if we let f denote the true fraction of novel input stimuli, we have

$$f_{\text{eff}} = P_{TN} f + P_{FN}(1-f) = (1-P_{FP}) f + (1-P_{TP})(1-f)$$

where P_{TN}, P_{FN}, P_{TP} and P_{FP} are the true negative, false negative, true positive, and false positive rates, respectively. Since by design there is exactly one hidden unit active for a novel stimulus, we have $h_i(t') = 1$ with probability $\frac{f_{\text{eff}}}{N}$, and $h_i(t') = 0$ with probability $1 - (f_{\text{eff}}/N)$. So, $h_i(t')$ is a Bernoulli random variable with mean (f_{eff}/N) and variance $(f_{\text{eff}}/N)(1 - (f_{\text{eff}}/N))$. Now, we let $H_i(t') = h_i(t')X(t')$, so

$$\varepsilon_i(t) = -\eta \sum_{t'=0}^{t-1} \lambda^{t-1-t'} H_i(t')$$

Although $h_i(t')$ is, in principle, a function of $\boldsymbol{x}(t')$, we assume they are independent. Since $X(t')$ is zero-mean, the mean of $H_i(t')$ is also zero. Using the identity $\text{var}(XY) = \text{var}(X)\text{var}(Y) + \text{var}(X)\mathbb{E}^2[Y] + \text{var}(Y)\mathbb{E}^2[X]$, which holds for independent random variables X and Y,

we have that the variance of $H_i(t')$ is $\frac{f_{\text{eff}}D}{N}$. Finally, for convenience we can rewrite this as

$$\varepsilon_i(t) = -\eta\sqrt{\frac{f_{\text{eff}}D}{N}}\sum_{t'=0}^{t-1}\lambda^{t-1-t'}\xi_i(t')$$

where $\xi_i(t')$ is a zero-mean, unit-variance random variable. Furthermore, we now see that by the Central Limit Theorem $\varepsilon_i(t)$ is a Gaussian random variable since we are considering the steady-state performance at large t, so we can take $t\to\infty$.

We can now compute the mean and variance of $\varepsilon_i(t)$. First, since $x_{n+k}(t)$ is zero-mean and independent of $A_{ik}(t)$,

$$\mathbb{E}[\varepsilon_i(t)] = \mathbb{E}\left[\sum_{k=1}^{D}A_{ik}(t)x_{n+k}(t)\right] = 0$$

To compute the variance,

$$\text{var}(\varepsilon_i(t)) = \mathbb{E}\left[\varepsilon_i^2(t)\right] - \mathbb{E}^2[\varepsilon_i(t)] = \mathbb{E}\left[\varepsilon_i^2(t)\right]$$

$$= \mathbb{E}\left[\left(-\eta\sqrt{\frac{f_{\text{eff}}D}{N}}\sum_{t'=0}^{t-1}\lambda^{t-1-t'}\xi_i(t')\right)^2\right]$$

$$= \mathbb{E}\left[\left(-\eta\sqrt{\frac{f_{\text{eff}}D}{N}}\sum_{t'=0}^{t-1}\lambda^{t-1-t'}\xi_i(t')\right)\left(-\eta\sqrt{\frac{f_{\text{eff}}D}{N}}\sum_{t''=0}^{t-1}\lambda^{t-1-t''}\xi_i(t'')\right)\right] = \eta^2\frac{f_{\text{eff}}D}{N}\sum_{t'=0}^{t-1}\sum_{t''=0}^{t-1}\lambda^{t-1-t'}\lambda^{t-1-t''}\mathbb{E}[\xi_i(t')\xi_i(t'')]$$

In general, we have $\mathbb{E}[\xi_i(t')\xi_i(t'')] = 1$ for $t' = t''$ since $\xi_i(t')$ is a zero-mean, unit-variance random variable. For $t'\neq t''$, we again make a simplifying independence assumption. In principle, $\xi_i(t'')$ is not independent of $\xi_i(t')$ since $h_i(t'')$ depends on $h_i(t')$ for $t''>t'$ through the memory stored in the $\boldsymbol{A}(t)$ matrix. This dependence, however, is sufficiently weak, so we let $\mathbb{E}[\xi_i(t')\xi_i(t'')] = 0$ for $t'\neq t''$. As a result, the double-sum collapses and we have

$$\text{var}(\varepsilon_i(t)) = \eta^2\frac{f_{\text{eff}}D}{N}\sum_{t'=0}^{t-1}\lambda^{2(t-1-t')} = \eta^2\frac{f_{\text{eff}}D}{N}\frac{1-\lambda^{2t}}{1-\lambda^2}$$

where the second equality comes from the standard geometric series. As before, since we are considering the steady-state with $t\to\infty$, we have $\gamma^{2t}\to 0$, so

$$\text{var}(\varepsilon_i(t)) = \eta^2\frac{f_{\text{eff}}D}{N}\frac{1}{1-\lambda^2}$$

Thus, for a novel input $\boldsymbol{x}(t)$ we can write

$$\varepsilon_i(t) = \xi_i\eta\sqrt{\frac{f_{\text{eff}}D}{N(1-\lambda^2)}} \qquad \text{(Equation 3)}$$

for all i, where ξ_i is a zero-mean, unit-variance Gaussian random variable, since $\varepsilon_i(t)$ is Gaussian.

For a familiar stimulus, where $\boldsymbol{x}(t) = \boldsymbol{x}(t-R)$, clearly $x_{n+k}(t')$ and $x_{n+k}(t)$ are no longer independent for $t' = t-R$. Thus, we consider this term separately, rewriting the sum in Equation 2 as

$$\varepsilon_i(t) = -\eta\lambda^{t-1-(t-R)}h_i(t-R)\sum_{k=1}^{D}x_{n+k}(t-R)x_{n+k}(t) - \eta\sum_{\substack{t'=0\\ t'\neq t-R}}^{t-1}\lambda^{t-1-t'}h_i(t')\sum_{k=1}^{D}x_{n+k}(t')x_{n+k}(t)$$

Assuming no errors, by design, $h_i(t-R) = 1$ for exactly one neuron i, since the stimulus at time $t-R$ was guaranteed to be novel (we enforce that a stimulus is repeated at most once in this task). We consider the statistics of $\varepsilon_i(t)$ for this particular neuron. In the first term, the sum $\sum_{k=1}^{D}x_{n+k}(t-R)x_{n+k}(t) = D$ since by assumption $x_{n+k}(t) = x_{n+k}(t-R)$ for all k. The second term has the same distribution as the one for a novel input since we have only removed one term from the sum and t is large. Thus, for a familiar stimulus we can write

$$\varepsilon_i(t) = -\eta\lambda^{R-1}D + \xi_i\eta\sqrt{\frac{f_{\text{eff}}D}{N(1-\lambda^2)}}$$

for exactly one value of i, where ξ_i is a zero-mean, unit-variance random variable as before. For all other values of i, Equation 3 holds.

Having established the statistics of the hidden layer input currents for a novel and a familiar stimulus, we can now write down the conditions for the model to work, use them to find the optimal values of the parameters and calculate the true positive and false positive probabilities, and compute the capacity – the largest value of R for which the error is below a predetermined threshold. First, to ensure that exactly one unit is active for a novel stimulus (true negative), since we are using a step function nonlinearity, we must have the largest input current take on a positive value (since ξ_i is an identically distributed standard normal random variable for every neuron, for simplicity we suppress the index i),

$$Kn + \xi\eta\sqrt{\frac{f_{\text{eff}}D}{N(1-\lambda^2)}} + b > 0$$

and second-largest to be below zero,

$$K(n-2) + \xi\eta\sqrt{\frac{f_{\text{eff}}D}{N(1-\lambda^2)}} + b < 0$$

Second, to ensure there are no units active for a familiar stimulus (true positive),

$$Kn - \eta\lambda^{R-1}D + \xi\eta\sqrt{\frac{f_{\text{eff}}D}{N(1-\lambda^2)}} + b < 0$$

For sufficiently large R, i.e., if $\eta\lambda^{R-1}D<2K$, the third of these conditions implies the second. Since we are interested in maximizing R, we only consider the first and third conditions. Furthermore, note that these conditions are overparameterized. If we divide all three equations by η (e.g., let $k = \frac{K}{\eta}$, $B = \frac{b}{\eta}$), we can eliminate this free parameter. In other words, for any value of η we can scale K and b proportionally to satisfy the conditions, so for simplicity we choose $\eta = 1$. Similarly, the term $Kn+b$ can be replaced by a single parameter since for any choice of K we can rescale b to keep this sum constant. To ensure that the condition $\eta\lambda^{R-1}D<2K$ holds for all R, we can choose $K = D$. For convenience, we also let $b = \beta D$ and $\sqrt{(f_{\text{eff}}D/N(1-\lambda^2))} = \alpha_\lambda D$, the subscript indicating explicit dependence on λ. Dividing both inequalities by D, the conditions simplify to

$$n + \alpha_\lambda\xi + \beta > 0$$
$$n + \beta + \alpha_\lambda\xi - \lambda^{R-1} < 0$$

The accuracy, i.e., probability of a correct response, is given by $P_{\text{correct}} = (1-f)P_{TP} + fP_{TN}$. For convenience, we compute the false positive instead of the true negative rate, noting that $P_{TN} = 1 - P_{FP}$. The false positive and true positive rates are given by

$$P_{FP} = \mathbb{P}\left[\xi < -\frac{n+\beta}{\alpha_\lambda}\right]$$
$$P_{TP} = \mathbb{P}\left[\xi < -\frac{n+\beta-\lambda^{R-1}}{\alpha_\lambda}\right]$$

Since ξ is a standard Normal random variable, $\mathbb{P}[\xi < z] = (1/2)\text{erfc}(-z/\sqrt{2})$, so

$$P_{FP} = \frac{1}{2}\text{erfc}\left(\frac{n+\beta}{\alpha_\lambda\sqrt{2}}\right)$$
$$P_{TP} = \frac{1}{2}\text{erfc}\left(\frac{n+\beta-\lambda^{R-1}}{\alpha_\lambda\sqrt{2}}\right)$$

We would now like to set the optimal values of λ and β which maximize R, given a desired true positive and false positive probability P^*_{FP}, P^*_{TP}. Note that fixing these probabilities also fixes $f_{\text{eff}} = f^* = (1-P^*_{FP})f + (1-P^*_{TP})(1-f)$. Rearranging the previous equations, we get

$$\frac{n+\beta}{\alpha_\lambda} = \sqrt{2}\text{erfc}^{-1}(2P^*_{FP})$$
$$\frac{n+\beta-\lambda^{R-1}}{\alpha_\lambda} = \sqrt{2}\text{erfc}^{-1}(2P^*_{TP})$$

The first equality sets the value for β. To determine λ, we substitute β into the second equality to get

$$\sqrt{2}\text{erfc}^{-1}(2P^*_{FP}) - \frac{\lambda^{R-1}}{\alpha_\lambda} = \sqrt{2}\text{erfc}^{-1}(2P^*_{TP})$$

For notational convenience, let $E = \sqrt{2}[\text{erfc}^{-1}(2P^*_{FP}) - \text{erfc}^{-1}(2P^*_{TP})]$. Using the definition of α_λ and $f_{\text{eff}} = f^*$, we have $\lambda = \sqrt{1 - (f^*/\alpha_\lambda^2 ND)}$. Rearranging, we have

$$\left(1 - \frac{f^*}{\alpha_\lambda^2 ND}\right)^{\frac{R-1}{2}} = \alpha_\lambda E$$

Assuming N and D are large (so λ is close to 1), we can use the first-order Taylor expansion $\exp(-z) \approx 1 - z$ for the term in parentheses (this will be necessary to get a closed-form expression for the optimal λ) and solve for R

$$\exp\left(-\frac{f^*}{\alpha_\lambda^2 ND} \cdot \frac{R-1}{2}\right) = \alpha_\lambda E \Rightarrow R = 1 + \frac{2ND\alpha_\lambda^2}{f^*} \ln\left(\frac{1}{\alpha_\lambda E}\right)$$

Setting $(dR/d\lambda) = 0$ and solving for λ gives the optimum

$$\lambda = \sqrt{1 - (eE^2 f^*/ND)}$$

Thus, the capacity of the optimized network is

$$R_{\max} = 1 + \frac{ND}{eE^2 f^*}$$

where f^* and E are constants that depend on P^*_{FP} and P^*_{TP} (f^* also depends on the true fraction of novel stimuli f). For instance, if we impose that $P^*_{FP} = 0.01$ and $P^*_{TP} = 0.99$, with our value of $f = (2/3)$, we get

$$\begin{aligned} R_{\max} &= 1 + \frac{ND}{e \cdot 2 \cdot \left[\text{erfc}^{-1}(2P^*_{FP}) - \text{erfc}^{-1}(2P^*_{TP})\right]^2 \cdot \left[(1 - P^*_{FP})f + (1 - P^*_{TP})(1 - f)\right]} \\ &= 1 + \frac{ND}{2e[1.645 - (-1.645)]^2[0.98f + 0.01]} \\ &= 1 + \frac{0.017ND}{0.98f + 0.01} = 1 + 0.026ND \end{aligned}$$

It is clear that that the capacity scales in proportion to the number of plastic synapses in the network. Furthermore, since $d = n + D$, i.e., $D = d - \log_2(N)$, the capacity scales in proportion to the total number of synapses d, as long as $D \gg n$.

$$R_{\max} = O(ND) = O(N(d - n)) = O(Nd - N\log N)) = O(Nd)$$

Finally, note that the equations for P_{FP} and P_{TP} are a function of f_{eff} due to the α_λ parameter, and therefore recursively depend on P_{FP} and P_{TP}. We cannot compute the closed-form solution for these, but we can approximate the values with arbitrary accuracy by iterating through this recurrence until convergence to the fixed point. As the initial value for the recurrence, we use P_{FP} and P_{TP} computed using $f_{\text{eff}} = f$, i.e., assuming no errors.

CNN preprocessing for familiarity detection of images

We consider the dataset used by Brady et al. (2008) to study familiarity detection in humans. As a stand-in for the processing done by the visual stream before the inferotemporal or perirhinal cortices, we use a pre-trained convolutional neural network (CNN), and extract the activity in its penultimate layer (before the final classification step). We use the ResNet18 network (He et al., 2015), although any CNN could, in principle, be used (see also Kazanovich and Borisyuk, 2021). This activity is a 512 -dimensional vector, which, if used as the HebbFF input dimension d, would lead to the capacity $R_{\max}$ being prohibitively large for training purposes. To keep the performance in a reasonable range, we downsample to $d = 50$, either by partial sampling (Figure 8A) or by introducing an intermediate layer (Figure S6A). We use the uniform readout $\boldsymbol{W}_2$ for simplicity of training and analysis, although the results are similar for the unconstrained readout.

As the first method of downsampling, we truncate the output of the CNN (Figure 8A). To keep the same input datatype as in previous sections, we also shift the inputs to zero mean and binarize them by taking the sign of each input component (Figure 8B). Unlike in previous sections, however, the inputs to HebbFF now have correlations that tend to be positive (Figure 8C). Nevertheless, this network has qualitatively similar features as the networks trained on uncorrelated vectors. The $\boldsymbol{W}_1$ matrix has a similar structure (Figures 4F and 8D), the hidden layer activity is sparse (Figures 4C and 8E), and the hidden unit input current distributions have similar shapes (Figures 4I, 4J, 4M, 7B, and 8F–8I). Due to the added correlations, however, there is a decline in performance compared to a network of the same size trained and evaluated on uncorrelated binary random vectors (Figure 8J).

As another way to downsample, we add a trainable linear layer that transforms the CNN output to a 50 -dimensional real-valued vector (Figure S6A). After training, the resulting inputs to HebbFF are no longer binary, but they are zero-mean (Figure S6B) and have zero-mean correlations Figure S6C). Interestingly, the network learns to generate this representation automatically to optimize

familiarity detection over long intervals, which further supports storing uncorrelated stimuli. Although the $\boldsymbol{W}_1$ matrix (Figure S6D) and the distribution of input currents from the fixed component of the synapses (Figure S6F) have a different structure compared to the original network, the operating principle remains the same: the $\boldsymbol{W}_1$ matrix acts as a addressing function to select a unique neuron in the hidden layer (Figure S6E) that is then suppressed for a familiar stimulus through the $\boldsymbol{A}(t)$ matrix (Figures S6G and S6H). The network maintains its generalization performance across repeat intervals R, and across permutations of the sequence of images (Figure S6J). However, it does not generalize well to images it has not been trained on. It is possible that this difficulty is due to the relatively small number of images used during training and may be addressed by using a much larger dataset such as ImageNet (Deng et al., 2009).