# Reducing Catastrophic Forgetting With Associative Learning: A Lesson From Fruit Flies

**Yang Shen**
*yashen@cshl.edu*
*Cold Spring Harbor Laboratory, Simons Center for Quantitative Biology,*
*Cold Spring Harbor, NY 11724, U.S.A.*

**Sanjoy Dasgupta**
*dasgupta@eng.ucsd.edu*
*Department of Computer Science and Engineering, University of California,*
*San Diego, La Jolla, CA 92093, U.S.A.*

**Saket Navlakha**
*navlakha@cshl.edu*
*Cold Spring Harbor Laboratory, Simons Center for Quantitative Biology,*
*Cold Spring Harbor, NY 11724, U.S.A.*

**Catastrophic forgetting remains an outstanding challenge in continual learning. Recently, methods inspired by the brain, such as continual representation learning and memory replay, have been used to combat catastrophic forgetting. Associative learning (retaining associations between inputs and outputs, even after good representations are learned) plays an important function in the brain; however, its role in continual learning has not been carefully studied. Here, we identified a two-layer neural circuit in the fruit fly olfactory system that performs continual associative learning between odors and their associated valences. In the first layer, inputs (odors) are encoded using sparse, high-dimensional representations, which reduces memory interference by activating nonoverlapping populations of neurons for different odors. In the second layer, only the synapses between odor-activated neurons and the odor's associated output neuron are modified during learning; the rest of the weights are frozen to prevent unrelated memories from being overwritten. We prove theoretically that these two perceptron-like layers help reduce catastrophic forgetting compared to the original perceptron algorithm, under continual learning. We then show empirically on benchmark data sets that this simple and lightweight architecture outperforms other popular neural-inspired algorithms when also using a two-layer feedforward architecture. Overall, fruit flies evolved an efficient continual associative learning algorithm, and circuit mechanisms from neuroscience can be translated to improve machine computation.**

*Neural Computation* **35, 1797–1819** (2023)
https://doi.org/10.1162/neco_a_01615

## 1 Introduction

Catastrophic forgetting, when neural networks inadvertently overwrite old memories with new memories, remains a long-standing problem in machine learning (Parisi et al., 2019). Here, we studied how fruit flies learn continuously to associate odors with behaviors and discovered a circuit motif capable of alleviating catastrophic forgetting.

While modern machine learning algorithms excel at learning complex and discriminating representations (LeCun et al., 2015), an equally challenging problem in continual learning is finding good ways to preserve associations between these representations and output classes. Indeed, the performance of deep artificial neural networks is considerably degraded when classes are learned sequentially (one at a time), as opposed to being randomly interleaved in the training data (Goodfellow et al., 2013). The effect of this simple change is profound and has warranted the search for new mechanisms that can preserve input-output associations over long periods of time. In addition, catastrophic forgetting has been shown to effect deeper layers of neural networks more than feature extraction layers (Ramasesh et al., 2021). This finding highlights the importance of preserving good associations for reducing catastrophic forgetting.

Since learning in the natural world often occurs sequentially, the past few years have witnessed an explosion of brain-inspired continual learning models. These models can be divided into three categories: (1) regularization models, where important weights (synaptic strengths) are identified and protected (Hinton & Plaut, 1987; Fusi et al., 2005; Benna & Fusi, 2016; Kirkpatrick et al., 2017; Zenke et al., 2017; Douillard et al., 2020; Peng et al., 2021); (2) experience replay models, which use external memory to store and reactivate old data (Lopez-Paz & Ranzato, 2017) or use generative models to generate new data from prior experience (van de Ven et al., 2020; Tadros et al., 2020, 2022; Shin et al., 2017) and (3) complementary learning systems (McClelland et al., 1995; Roxin & Fusi, 2013), which partition memory storage into multiple subnetworks, each subject to different learning rules and rates. Importantly, these models often take inspiration from mammalian memory systems, such as the hippocampus (Wilson & McNaughton, 1994; Rasch & Born, 2007) or the neocortex (Qin et al., 1997; Ji & Wilson, 2007), where detailed circuit anatomy and physiology are still lacking. Fortunately, continual learning is also faced by simpler organisms, such as insects, where supporting circuit mechanisms are understood at synaptic resolution (Takemura et al., 2017; Zheng et al., 2018; Li et al., 2020).

Most brain-inspired algorithms use backpropagation-based supervised learning, whose existence in the brain is still controversial. On the other hand, associative learning (a local learning scheme that strengthens connections between representation neurons and output neurons) plays an important role for learning in the brain, though its role toward reducing catastrophic forgetting in artificial neural networks has not been carefully analyzed.

Here, we developed an associative continual learning algorithm inspired by the fruit fly olfactory system. This algorithm tackles an important yet underappreciated subproblem within continual learning: after good representations are learned, how do you best preserve associations between representation neurons and output classes in a class-incremental learning framework? The algorithm we propose stitches together two well-known computational ideas—sparse coding (Maurer et al., 2013; Ruvolo & Eaton, 2013; Ororbia et al., 2019; Ahmad & Scheinkman, 2019; Rapp & Nawrot, 2020; Hitron et al., 2020) and perceptron-like associative learning (Hinton & Plaut, 1987; Fusi et al., 2005; Benna & Fusi, 2016; Kirkpatrick et al., 2017; Zenke et al., 2017; Minsky & Papert, 1988)—in a unique and effective way, which we show effectively reduces catastrophic forgetting under a simple feedforward network architecture. The fruit fly circuit uses a perceptron-like architecture, which we prove theoretically helps reduce catastrophic forgetting compared to the original perceptron algorithm. We also show empirically that the fruit fly circuit outperforms alternative perceptron-like circuits in design space (e.g., replacing sparse coding with dense coding, associative learning (freezing synapses) with supervised learning (modifiable synapses), which provides biological insight into the function of these evolved circuit motifs and how they operate together in the brain to sustain memories.

## 2 Methods

### 2.1 Circuit Mechanisms for Continual Learning in Fruit Flies.

How do fruit flies associate odors (inputs) with behaviors (classes) such that behaviors for odors learned long ago are not erased by newly learned odors? We first review the basic anatomy and physiology of two layers of the olfactory system that are relevant to the exposition here (Modi et al., 2020).

The two-layer neural circuit we study takes as input an odor after a series of preprocessing steps have been applied, including gain control (Root et al., 2008; Gorur-Shandilya et al., 2017), noise reduction (Wilson, 2013), and normalization (Olsen et al., 2010; Stevens, 2015). After these steps, odors are represented by the firing rates of $d = 50$ types of projection neurons (PNs), which constitute the input to the two-layer network motif described next.

*2.1.1 Sparse Coding.* The goal of the first layer is to convert the dense input representation of the PNs into a sparse, high-dimensional representation (Cayco-Gajic & Silver, 2019) (see Figure 1A). This is accomplished by a set of about 2000 Kenyon cells (KCs), which receive input from the PNs. The matrix connecting PNs to KCs is sparse and approximately random (Caron et al., 2013); each KC randomly samples from about 6 of the 50 projection

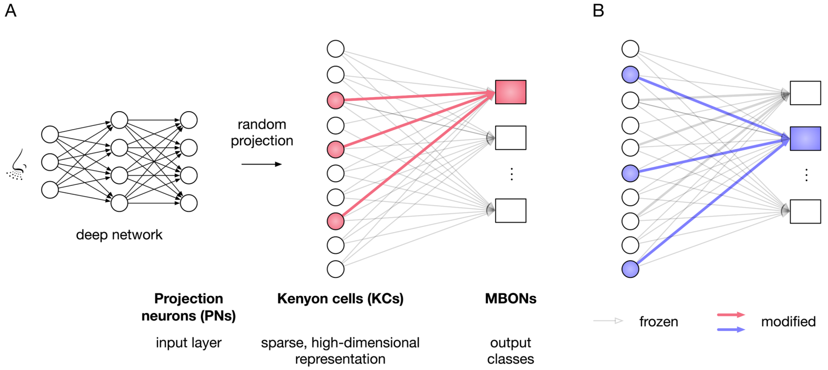

Figure 1: A two-layer circuit for continual learning in the fruit fly olfactory system. (A) An input (odor) is received by the "nose" and preprocessed via a series of transformations. In the fruit fly, this preprocessing includes noise reduction, normalization, and gain control. In a deep network, preprocessing is similarly used to generate a suitable representation for learning. After these transformations, the dimensionality of the preprocessed input (PNs) is expanded via a random projection and is sparsified via winner-take-all thresholding. This leaves only a few Kenyon cells active per odor (indicated by red shading). To associate the odor with an output class (MBON), only the synapses connecting the active Kenyon cells to the target MBON are modified. The rest of the synapses are frozen. (B) A second example with a second odor, showing different Kenyon cells activated, associated with a different MBON.

neurons and sums up their firing rates. Next, each KC provides feedforward excitation to a single inhibitory neuron, called APL. In return, APL sends feedback inhibition to each KC. The result of this loop is that approximately 95% of the lowest-firing KCs are shut off, and the top 5% remain firing in what is often referred to as a winner-take-all (WTA) computation (Turner et al., 2008; Lin et al., 2014; Stevens, 2015). Thus, an odor initially represented as a point in $\mathbb{R}_+^{50}$ is transformed, via a 40-fold dimensionality expansion followed by WTA thresholding, to a point in $\mathbb{R}_+^{2000}$, where only approximately 100 of the 2000 KCs are active (i.e., nonzero) for any given odor.

This transformation was previously studied in the context of similarity search (Dasgupta et al., 2017, 2018; Papadimitriou & Vempala, 2018; Ryali et al., 2020), compressed sensing (Stevens, 2015; Zhang & Sharpee, 2016), and pattern separation for subsequent learning (Babadi & Sompolinsky, 2014; Litwin-Kumar et al., 2017; Dasgupta & Tosh, 2020).

*2.1.2 Associative Learning.* The goal of the second layer is to associate odors (sparse points in high-dimensional space) with behaviors.

The main locus of associative learning lies at the synapses between KCs and a set of mushroom body output neurons called MBONs (Aso et al., 2014), which encode behaviorally relevant odor information important for decision making (see Figure 1).

During training, say the fly is presented with a naive odor (odor A) that is paired with a punishment (e.g., an electric shock). How does the fly learn to avoid odor A in the future? Initially, the synapses from KCs activated by odor A to both the "approach" MBON and the "avoid" MBON have equal weights. When odor A is paired with punishment, the KCs representing odor A are activated around the same time that a punishment-signaling dopamine neuron fires in response to the shock. The released dopamine causes the synaptic strength between odor A KCs and the approach MBON to decrease, resulting in a net increase in the avoidance MBON response.[1] Eventually the synaptic weights between odor A KCs and the approach MBON are sufficiently reduced to reliably learn the avoidance association (Felsenberg et al., 2018).

Importantly, the only synapses that are modified in each associative learning trial are those from odor A KCs to the approach MBON. All synapses from odor A KCs to the avoid MBON are frozen (i.e., left unchanged), as are all weights from silent KCs to both MBONs. Thus, the vast majority of synapses are frozen during any single odor-association trial.

To summarize, associative learning in the fly is driven by dopamine signals that only affect the synapses of sparse odor-activated KCs and a target MBON that drives behavior.

### 2.2 The FlyModel.

We now introduce an associative continual learning algorithm based on the two-layer olfactory circuit above.

As input, we are given a $d$-dimensional vector, $x = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$. As in the fly circuit, we assume that $x$ is preprocessed to remove noise and encode discriminative features. Biologically, this is often accomplished by peripheral sensory circuitry that is separate from learning-related circuitry; computationally, the representations extracted from models trained on other data sets, as often done in transfer learning, could serve a similar role. To emphasize, our goal here is not to study the complexities of learning good representations but rather to disentangle representation learning from associative learning and focus exclusively on the latter.

The first layer computes a sparse, high-dimensional representation of $x$. This layer has $m$ units, where $m \approx 40d$. The input layer and the first layer are connected by a sparse, binary random matrix, $\Theta$, of size $m \times d$. Each row of $\Theta$ contains about $0.1d$ ones in random positions, and the

[1] Curiously, approach behaviors are learned by decreasing the avoid MBON response, as opposed to increasing the approach MBON response, as may be more intuitive.

rest of the positions in the row are set to zero. The initial representation $\psi(x) = (\psi_1, \psi_2, \ldots, \psi_m) \in \mathbb{R}^m$ is computed as

$$\psi(x) = \Theta x. \tag{2.1}$$

After this dimensionality expansion, a winner-takes-all process is applied, so that only the top $l$ most active units remain on and the rest of the units are set to 0. This produces a sparse representation $\phi(x) = (\phi_1, \phi_2, \ldots, \phi_m) \in \mathbb{R}^m$, where

$$\phi_i = \begin{cases} \psi_i & \text{if } \psi_i \text{ is one of the } l \text{ largest positive units of } \psi(x) \\ 0 & \text{otherwise.} \end{cases} \tag{2.2}$$

For computational convenience, a min-max normalization is applied to $\phi(x)$ so that each unit has a value between 0 and 1. The matrix $\Theta$ is fixed and not modified during learning; that is, there are no trainable parameters in the first layer. The winner-takes-all competition could be implemented in alternative ways (Holca-Lamarre et al., 2017) besides the direct inhibition we show here. But direct inhibition has the benefit of easily specifying the number of neurons remaining active per input.

The second layer is an associative learning layer, which contains $k$ output class units, $y = \{y_1, y_2, \ldots, y_k\}$. The first and second layers are connected with all-to-all synapses. If an input $x$ is to be associated with target $y_j$, the only weights that are modified are those between the active units in $\phi(x)$ and $y_j$. No other weights, including those from the active units in $\phi(x)$ to the other $k - 1$ units in $y$—are modified. We refer to this as "partial freezing" of weights during learning.

Formally, let $w_{ij} \in [0, 1]$ be the weight between $\phi_i$ and $y_j$. Then, for all $i \in [1 \ldots m]$, $j \in [1 \ldots k]$, the weight update rule after each input $x$ is

$$w_{ij} = \begin{cases} (1-\alpha)w_{ij} + \beta\phi_i & \text{if } j = \text{target} \\ (1-\alpha)w_{ij} & \text{otherwise.} \end{cases} \tag{2.3}$$

Here, $\beta$ is the learning rate, and $\alpha$ is a very small forgetting term that mimics slow, background memory decay. In our experiments, we set $\alpha = 0$ to minimize forgetting and to simplify the model. The problem of weight saturation arises when $\alpha = 0$, since weights can only increase and never decrease. However, despite thousands of training steps, most weights did not saturate (see Figure S1) since most $\phi_i$ are inactive for most classes (sparse coding) and only a small fraction of the active $\phi_i$ weights are modified during learning (partial freezing). Nonetheless, in practice, some small, nonzero $\alpha$ may be desired to avoid every synapse from eventually saturating.

Finally, biological synapses have physical bounds on their strength, and here we mimic these bounds by capping weights to [0, 1].

## 3 Theoretical Results

The fruit fly associative learning algorithm (partial freezing) resembles a well-known supervised learning algorithm: the perceptron (Rosenblatt, 1958). On an instance $\phi$, the multiclass perceptron predicts a label in $[1 \ldots k]$ and then performs an update if this prediction is incorrect:

$$w_{ij} = \begin{cases} w_{ij} + \phi_i & \text{if } j = \text{target} \\ w_{ij} - \phi_i & \text{if } j = \text{prediction} \\ w_{ij} & \text{otherwise.} \end{cases} \tag{3.1}$$

In contrast, the partial freezing algorithm always updates and modifies only the weights of the target class. We now see how these differences affect catastrophic forgetting. (See supplement for details.)

**3.1 Even the Perceptron with Linearly Separable Data Suffers from Catastrophic Forgetting.** It is well known that if the data points $(x, y)$ presented to the perceptron algorithm have linear margin $\gamma > 0$ and have lengths bounded by $\|x\| \leq R$, then the algorithm will make at most $2kR^2/\gamma^2$ wrong predictions over its entire lifetime, where $k$ is the number of classes. This is true regardless of the order in which the data are presented.

Nonetheless, catastrophic forgetting is possible. One way this can happen is if the data are not perfectly separable. In that case, the linear margin condition does not hold and the perceptron will not converge. But forgetting can occur even when the data are separable, as we now show.

Suppose we introduce one class at a time, or a couple of classes at a time, as is commonly done in continual learning. The guarantees of the perceptron need to be interpreted carefully in this setting. After new classes are introduced, the algorithm will, in general, need to see more examples of earlier classes and tweak their weight vectors further. It is thus never "done" with a particular class.

To make this more concrete, suppose that we introduce one class at a time, and the margin so far, after seeing the first $j$ classes is $\gamma_j$. That is, $\gamma_j$ is the linear margin of classes $\{1, 2, \ldots, j\}$, which can only get smaller as $j$ grows: $\gamma_1 \geq \gamma_2 \geq \gamma_3 \geq \cdots$ Upon introducing the $k$th class, the mistake bound goes up by $2kR^2/\gamma_k^2 - 2(k-1)R^2/\gamma_{k-1}^2 \geq 2R^2/\gamma_k^2$; we can think of this quantity as the additional number of mistakes we might make in accommodating the $k$th class. Crucially, in order to get convergence, some of these mistakes may need to be made on classes that have been seen earlier.

We now construct a concrete example of this phenomenon.

**Lemma 1.** *Pick any positive integer $d$ that is a power of two. Then there exists a set of $d$ −1 vectors $x_1, \ldots, x_{d-1} \in \{0, 1\}^d$ with the following two properties: (1) each vector $x_i$ has exactly d/2 ones and (2) any pair of vectors has a dot product exactly d/4.*

**Proof.** Start with the $d \times d$ Hadamard matrix and remove the row that is all ones. In the resulting $(d - 1) \times d$ matrix, replace every −1 with a 0, and take $x_1, \ldots, x_{d-1}$ to be the rows of the matrix. □

Now consider a situation with $d - 1$ classes, where class $j$ is supported on the single point $x_j$. The ideal classifiers are $w_j = x_j$:

$$w_j \cdot x_i = \begin{cases} d/2 & \text{if } i = j \\ d/4 & \text{otherwise.} \end{cases}$$

Suppose the classes are introduced one at a time, in order:

- Class 1 is introduced: $w_1$ is set to $x_1$, which is perfect.
- Class 2 is introduced: point $x_2$ is misclassified as coming from class 1. Thus, $w_2$ is set to $x_2$, which is perfect, but $w_1$ is changed to $x_1 - x_2$, which no longer correctly classifies class 1:

  $$\begin{aligned} w_1 \cdot x_1 &= (x_1 - x_2) \cdot x_1 = d/2 - d/4 = d/4, \\ w_2 \cdot x_1 &= d/4. \end{aligned}$$

  Moreover, for any $j > 2$, we have $w_1 \cdot x_j = (x_1 - x_2) \cdot x_j = 0$. Thus, all such points $x_j$ will be classified as class 2.
- Class 3 is introduced: point $x_3$ is misclassified as coming from class 2. Thus $w_3$ is set to $x_3$, but now $w_2$ becomes $x_2 - x_3$, suffering a similar fate to $w_1$.

And so on. By the time $x_{d-1}$ has arrived, all of $w_1, \ldots, w_{d-2}$ have been corrupted to the point of uselessness, even though they were originally set perfectly.

Thus, even in the case where the perceptron algorithm is known to fare best (i.e., when the data are linearly separable), catastrophic forgetting occurs under continual learning.

**3.2 The Partial-Freezing perceptron Does Not Suffer from Catastrophic Forgetting.** The partial-freezing algorithm in equation 2.3 is an associative version of the perceptron. We now show that even in a continual learning framework, this algorithm will provably learn to correctly distinguish classes if the classes satisfy a *separation condition* that says, roughly, that dot products between points within the same class are, on average, greater than between classes. We will then show that adding sparse coding enhances the separation of classes (Babadi & Sompolinsky, 2014), making associative learning easier.

**Definition 1.** *Let $\pi_1, \ldots, \pi_k$ be distributions over $\mathbb{R}^d$, corresponding to k classes of data points. We say the classes are $\gamma$-separated, for some $\gamma > 0$, if for any pair of classes $j \neq j^{'}$ and any point $x_o$ from class j,*

$$\mathbb{E}_{X \sim \pi_j}[x_o \cdot X] \geq \gamma + \mathbb{E}_{X' \sim \pi_{j'}}[x_o \cdot X'].$$

*Here, $\mathbb{E}_{X \sim \pi}$ refers to expected value under a vector X drawn at random from distribution $\pi$.*

Under $\gamma$-separation, the labeling rule,

$$x \mapsto \arg\max_j \, w_j \cdot x,$$

is a perfect classifier if the $w_j$ (i.e., the KC → MBON weight vector for class *j*) are the means of their respective classes, that is, $w_j = \mathbb{E}_{X \sim \pi_j}[X]$. This holds even if the means are only approximately accurate, within $O(\gamma)$ (in the supplement, see theorem 8). The partial freezing algorithm, in turn, provably produces such mean estimates (in the supplement, see theorem 5).

### 3.3 Sparse Coding Provably Creates Favorable Separation for Continual Learning.

The separation condition of definition 1 is quite strong and might not hold in the original data space. But we will show that subsequent sparse coding can nonetheless produce this condition, so that the partial freezing algorithm, when run on the sparse encodings, performs well.

To see a simple model of how this can happen, suppose that there are $N$ prototypical inputs, denoted $p_1, \ldots, p_N \in \mathcal{X}$, where $\mathcal{X} \subset \mathbb{R}^d$, that are somewhat separated from each other: for some $\xi \in [0, 1)$,

$$\frac{p_i \cdot p_j}{\|p_i\| \|p_j\|} \leq \xi.$$

Each $p_i$ has a label $y_i \in [1 \ldots k]$. Let $C_j \subset [N]$ be the set of prototypes with label *j*. Since the labels are arbitrary, these classes will in general not be linearly separable in the original space (see Figure S2).

Suppose the sparse coding map $\phi : \mathcal{X} \rightarrow \{0, 1\}^m$ generates *k*-sparse representations with the following property: for any $x, x' \in \mathcal{X}$,

$$\phi(x) \cdot \phi(x') \leq k \, f\left(\frac{x \cdot x'}{\|x\| \|x'\|}\right),$$

where $f$: $[-1, 1] \to [0, 1]$ is a function that captures how the coding process transforms dot products. Some earlier work (Dasgupta et al., 2018) has characterized $f$ for two types of random mappings: a sparse binary matrix (inspired by the fly's architecture) and a dense gaussian matrix (common in engineering applications). In either case, $f(s)$ is a much shrunken version of $s$; in the dense gaussian case, for instance, it is roughly $(k/m)^{1-s}$.

We can show that for suitable $\xi$, the sparse representations of the prototypes—that is, $\phi(p_1), \ldots, \phi(p_N) \in \{0, 1\}^m$—are then guaranteed to be separable, so that the partial freezing algorithm will converge to a perfect classifier.

**Theorem 1.** *Let $N_o = \max_j |C_j|$. Under the assumptions above, the sparse representation of the data set, $\{(\phi(p_1), y_1), \ldots, (\phi(p_N), y_N)\}$, is $(1/N_o - f(\xi))$-separated in the sense of definition 1.*

**Proof.** This is a consequence of theorem 9 in the supplement, a more general result that applies to a broader model in which observed data are noisy versions of the prototypes. □

## 4 Experimental Evaluation

**4.1 Testing Framework and Problem Setup.** We tested each algorithm (see below) on two benchmark data sets using a class-incremental learning setup (Farquhar & Gal, 2019; van de Ven et al., 2020), in which the training data were ordered and split into sequential tasks. For the MNIST-20 data set (a combination of regular MNIST and Fashion MNIST; see the supplement), we used 10 nonoverlapping tasks, where each task is a classification problem between two classes. For example, the first task is to classify between digits 0 and 1, the second task is to classify digits 2 and 3, and so on. Similarly, the CIFAR-100 data set is divided into 25 nonoverlapping tasks, where each task is a classification problem among four classes.

Testing is performed after the completion of training of each task and is quantified using two measures. The first measure, the accuracy for classes trained so far, assesses how well classes from previous tasks remain correctly classified after a new task is learned. Specifically, after training task $i$, we report the accuracy of the model tested on classes from all tasks $\leq i$. The second measure, memory loss, quantifies forgetting for each task separately. We define the memory loss of task $i$ as the accuracy of the model when tested (on classes from task $i$ only) immediately after training on task $i$ minus the accuracy when tested (again, on classes from task $i$ only) after training on all tasks, that is, at the end of the experiment. For example, say the immediate accuracy of task $i$ is 0.80, and the accuracy of task $i$ at the end of the experiment is 0.70. Then the memory loss of task $i$ is 0.10. A memory loss of zero means memory of the task was perfectly preserved despite learning new tasks.

*4.1.1 Comparison to Other Algorithms.* There are, of course, many heavy-duty continual learning algorithms in the literature, and our intention here is not to perform an exhaustive comparison to them. Instead, we compared the FlyModel with three neurally plausible methods that are popular in the literature and represent the broad strategies of brain-inspired continual learning (e.g., synapse protection, memory replay) outlined in section 1. All of these methods use backpropagation-based supervised learning as opposed to associative learning. Moreover, none of these methods have provable convergence guarantees, as ours does:

1. *Elastic weight consolidation* (EWC; Kirkpatrick et al., 2017) uses the Fisher information criterion to identify weights that are important for previously learned tasks and then introduces a penalty if these weights are modified when learning a new task.
2. *Gradient episodic memory* (GEM; Lopez-Paz & Ranzato, 2017) uses a memory system that stores a subset of data from previously learned tasks. These data are used to assess how much the loss function on previous tasks increases when model parameters are updated for a new task.
3. *Brain-inspired replay* (BI-R; van de Ven et al., 2020) protects old memories by using a generative model to replay activity patterns related to previously learned tasks.
4. *Vanilla* is a standard fully connected neural network that does not have any explicit continual learning mechanism. This is used as a lower bound on performance.
5. *Offline* is a standard fully connected neural network, but instead of learning classes sequentially, for each task, it is retrained from scratch on all classes (current and previously seen) together, presented in a random order. This is used as an upper bound on performance.

For a fair comparison, all five methods (except BI-R; see the supplement) use the same architecture as the FlyModel—the same number of layers, the same number of units per layer ($m$ units in the first layer, $k$ units in the second layer)—and they all use the same hidden unit activation function (ReLU). In addition, all methods, including the FlyModel, use the same representation for each input. Thus, the primary difference among methods is how learning mechanisms store and preserve memories.

See the supplement for full details on data sets, preprocessing, network architectures, and parameters.

**4.2 The FlyModel Outperforms Existing Methods in Class-Incremental Learning.** The FlyModel reduced catastrophic forgetting compared to all four continual learning methods tested. For example, on the MNIST-20 data set (see Figure 2A), after training on 5 tasks (10 classes), the accuracy of the FlyModel was 0.86 ± 0.0006 compared to 0.77 ± 0.02 for BI-R, 0.69 ± 0.02 for GEM, 0.58 ± 0.10 for EWC, and 0.19 ± 0.0003 for Vanilla. At the end

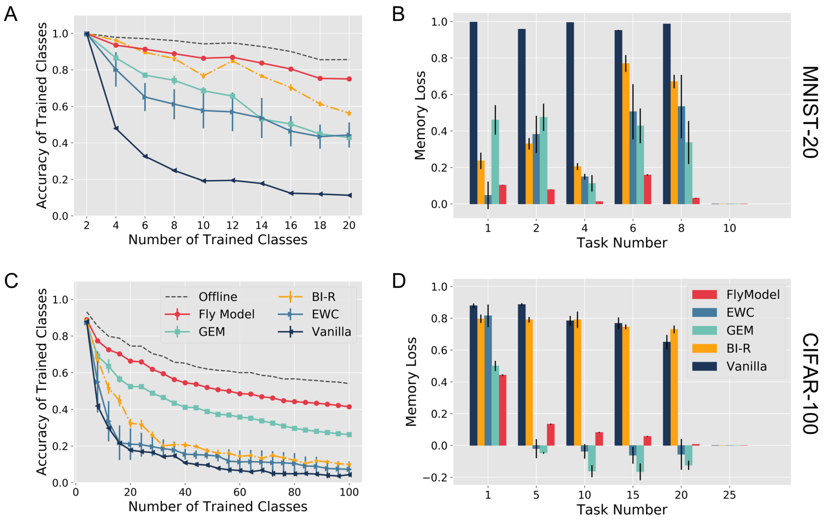

Figure 2: The FlyModel outperforms existing continual learning methods in class-incremental learning. (A) The *x*-axis is the number of classes trained on, and the *y*-axis is the classification accuracy when testing the model on the classes trained on thus far. The offline method (dashed black line) shows the optimal classification accuracy when classes are presented together, instead of sequentially. Error bars show standard deviation of the test accuracy over five random initializations for GEM, BI-R, EWC, and Vanilla, or over five random matrices (Θ) for the FlyModel. (B) The *x*-axis is the task number during training and the *y*-axis is the memory loss of the task. (A–B) MNIST-20 data set. (C–D) CIFAR-100. The memory loss of all tasks is shown in Figure S3.

of training (10 tasks, 20 classes trained), the test accuracy of the FlyModel was at least 0.19 higher than any other method and only 0.11 lower than the optimal offline model.

Next, we used the memory loss measure to quantify how well the "memory" of an old task is preserved after training new tasks (see Figures 2B and S3). As expected, the standard neural network (Vanilla) preserves almost no memory of previous tasks; it has a memory loss of nearly one for all tasks except the most recent task. While GEM, EWC, and BI-R perform better—memory losses of 0.24, 0.27, and 0.42, respectively, averaged across all tasks—the FlyModel has an average memory loss of only 0.07. This means that the accuracy of task *i* was only degraded on average by 7% at the end of training when using the FlyModel.

Similar trends were observed on a second, more difficult data set (CIFAR-100; see Figures 2C and 2D), where the FlyModel had an accuracy that was

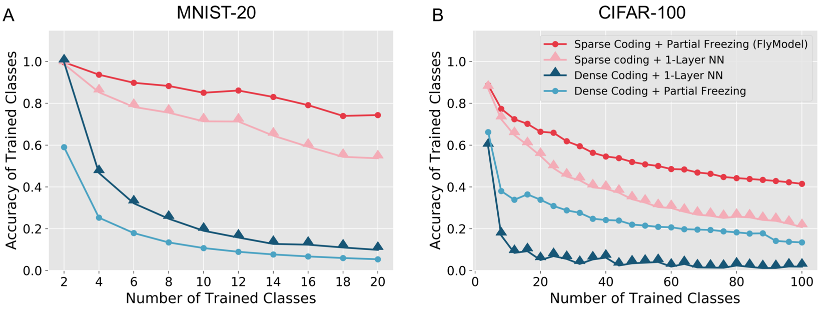

Figure 3: Sparse coding and partial freezing are both required for continual learning. Axes are the same as those in Figure 2A. Both sparse coding methods outperform both dense coding methods. When using sparse coding, partial freezing outperforms logistic regression (one-layer neural network). (A) MNIST-20. (B) CIFAR-100.

at least 0.15 greater than all continual learning methods and performed only 0.13 worse than the offline model.

**4.3 Sparse Coding and Partial Freezing Are Both Required for Continual Learning.** An important challenge in theoretical neuroscience is to understand why circuits may be designed the way they are. Quantifying how evolved circuits fare against putative, alternative circuits in design space could provide insight into the biological function of observed network motifs. We explored this question in the context of the two core components in the FlyModel: sparse coding of representations in the first layer and partial freezing of synaptic weights in the associative learning layer. Are both of these components required, or can good performance be attained with only one or the other?

We piecemeal explored the effects of replacing sparse coding with dense coding and replacing partial freezing with a traditional single-layer neural network (i.e., logistic regression), where every weight can change for each input. This gave us four combinations to test. The dense code was calculated in the same way as the sparse code, minus the winner-take-all step. In other words, for each input $x$, we used $\psi(x)$ (see equation 2.1, with min-max normalization) as its representation, instead of $\phi(x)$ (see equation 2.2). For logistic regression, the associative layer was trained using backpropagation.

Both sparse coding variants (with partial freezing or with logistic regression) performed better than the two dense coding variants on both data sets (see Figures 3A and 3B). For example, on MNIST-20, at the end of training, the sparse coding models had an average accuracy of 0.64 versus 0.07 for the two dense coding models. Furthermore, sparse coding with partial freezing (i.e., the FlyModel) performed better than sparse coding with logistic regression: 0.75 versus 0.54 on MNIST-20 and 0.41 versus 0.21 on CIFAR-100.

Hence, on at least the two data sets used here, both sparse coding and partial freezing are needed to optimize continual learning performance.

**4.4 Comparison of the FlyModel with the Perceptron.** In our theoretical analysis, we highlighted two important differences between the perceptron-supervised learning algorithm and the FlyModel associative learning algorithm. Next, we studied how the four combinations of these two differences affect continual learning.

The first combination (Perceptron v1) is the classic perceptron learning algorithm, where weights are modified only if an incorrect prediction is made, by increasing weights to the correct class and decreasing weights to the incorrectly predicted class. The second combination (Perceptron v2) also learns only when a mistake is made, but it increases weights only to the correct class (i.e., it does not decrease weights to the incorrect class). The third combination (Perceptron v3) increases weights to the correct class regardless of whether a mistake is made, and it decreases weights to the incorrect class when a mistake is made. Finally, the fourth combination (Perceptron v4) is equivalent to the FlyModel; it simply increases weights to the correct class regardless of whether a mistake is made. All models start with the same sparse, high-dimensional input representations in the first layer.

**Perceptron v1** (Original Perceptron)

```
for $x$ in data do
    if predict $\neq$ target then
        weight[target] $+= \beta x$
        weight[predict] $-= \beta x$
    end if
end for
```

**Perceptron v2**

```
for $x$ in data do
    if predict $\neq$ target then
        weight[target] $+= \beta x$

    end if
end for
```

**Perceptron v3**

```
for $x$ in data do
    if predict $\neq$ target then
        weight[target] $+= \beta x$
        weight[predict] $-= \beta x$
    else
        weight[target] $+= \beta x$
    end if
end for
```

**Perceptron v4** (FlyModel)

```
for $x$ in data do
    if predict $\neq$ target then
        weight[target] $+= \beta x$

    else
        weight[target] $+= \beta x$
    end if
end for
```

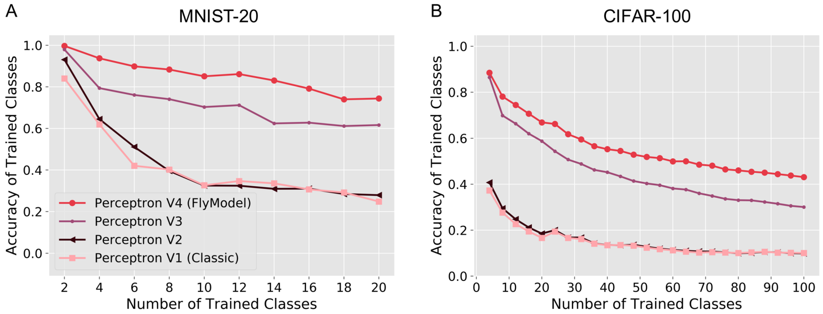

Figure 4: Continual learning performance for the four perceptron variants. Axes are the same as those in Figure 2A. Compared to the classic perceptron learning algorithm (Perceptron v1), the FlyModel (Perceptron v4) learns regardless of whether a mistake is made, and it does not decrease weights to the incorrect class when mistakes are made. These two changes significantly improve continual learning performance. (A) MNIST-20. (B) CIFAR-100.

Overall, we find a striking difference in continual learning with these two tweaks, with the FlyModel performing significantly better than the other three models on both data sets (see Figures 4A and 4B). Specifically, learning regardless of whether a mistake is made (v3 and v4) works better than mistake-only learning (v1 and v2), and decreasing the weights to incorrectly predicted class hurts performance (v4 compared to v3; no major difference between v2 and v1).

As we showed analytically, decreasing weights to the incorrect class (v1 and v3) suffers from catastrophic forgetting when inputs from different classes are overlapping. While this feature of the perceptron algorithm is believed to help create a larger boundary (margin) between the predicted incorrect class and the correct class, it also causes shared weights to be hijacked by recent classes observed. This leads to more catastrophic forgetting, albeit faster initial learning. The FlyModel, on the other hand, avoids this issue because the shared neurons are split between both classes and thus cancel each other out. As a result, the weight vectors in the associative layer converge to the mean of its class inputs, scaled by a constant (see the supplement, lemmas 3 and 4 and theorems 5 and 8). See supplement figures S4 and S5 for an empirical demonstration of this result.

## 5 Discussion

While learning mechanisms in the brain have been the source of inspiration for many continual learning algorithms, one commonly used neural learning mechanism (associative learning) has been largely overlooked. Here, we developed a simple and lightweight associative continual learning algorithm that reduces catastrophic forgetting, inspired by how fruit flies learn odor-behavior associations. The FlyModel outperformed three popular class-incremental continual learning algorithms on two benchmark data sets (MNIST-20 and CIFAR-100), despite not using external memory, generative replay, or backpropagation. The fly's associative learning algorithm is strikingly similar to the classic perceptron algorithm but for two modifications that we show are critical for retaining old memories. Indeed, alternative circuits in design space suffered more catastrophic forgetting than the FlyModel, potentially shedding new light on the biological function and conservation of this circuit motif. Finally, we grounded these ideas theoretically by proving that associative layer weight vectors in the FlyModel converge to the mean representation of its class and that sparse coding further reduces memory interference by better separating classes compared to the conventional perceptron algorithm, which we proved suffers under the continual learning scenario even when classes are linearly separable.

Given the same architecture, the FlyModel requires less memory and has comparable training efficiency compared to alternative methods (see the supplement, Figures S6 and S7). If the input layer is $N$-dimensional and the hidden layer undergoes a 40 times dimensionality expansion, given $m$ inputs and $t$ tasks, the number of parameters (weights) the FlyModel needs to store is of $O(N^2 + tN)$ across the two layers, and the total computational complexity for training is $O(mN)$, since for each input, we only update a few ($5\% \times 40N = 2N$) weights in the second layer. In addition, since FlyModel makes no distinction between tasks, the computational complexity is independent of $t$. On the other hand, EWC and GEM require additional storage of weights or data from previous tasks.

The two main features of the FlyModel—sparse coding (Kanerva, 1988, 2009; Babadi & Sompolinsky, 2014) and associative learning (i.e., partial synaptic freezing; Kirkpatrick et al., 2017; Zenke et al., 2017)—are well appreciated in both neuroscience and machine learning. For example, sparse, high-dimensional representations have long been recognized as central to neural encoding (Kanerva, 1988), hyperdimensional computing (Kanerva, 2009), and classification and recognition tasks (Babadi & Sompolinsky, 2014). Similarly, the notion of freezing certain weights during learning has been used in both classic perceptrons and modern deep networks (Kirkpatrick et al., 2017; Zenke et al., 2017), but these methods are still subject to interference caused by dense representations. However, the benefits of such features toward continual learning have not been well quantified. Indeed, the fly circuit evolved a unique combination of common computational ingredients that work effectively in practice.

The FlyModel performs associative rather than supervised learning. In associative learning, the same learning rule is applied regardless of whether the model makes a mistake. In supervised learning, changes are made to weights only when the model makes a mistake, and the changes are applied to weights for both the correct and the incorrect class labels. In other words, the FlyModel learns each class independently compared to supervised methods, and hence is flexible about the total number of classes to be learned; the network is easily expandable to more classes if necessary. Supervised methods focus on discrimination between multiple classes at a time, which we showed is particularly susceptible to interference, especially when class representations are overlapping. Thus, our results suggest that some traditional benefits of supervised classification may not carry over to continual learning (Hand, 2006) and that association-like models may better preserve memories when classes are learned sequentially.

However, associative learning alone without good representations is not sufficient to achieve good continual learning performance. While conventional backpropagation-based continual learning algorithms try to solve both representation learning and association learning at the same time, it could be argued that the brain takes a different approach by separating these two into different network layers. The associative learning component of continual learning has not been well studied in the literature, even though, as we showed, this seemingly simple problem can have important consequences on reducing catastrophic forgetting.

Previous studies share conceptual similarities with some features of FlyModel. For example, PackNet (Mallya & Lazebnik, 2018) uses weight pruning to free up redundant weights while keeping important weights fixed. This approach is similar to partial freezing, but instead of pruning less important weights, partial freezing only modifies relevant weights during learning and requires no computation to determine the importance of weights in retrospect. Partial freezing also resembles another well-known continual learning method, iCaRL (Rebuffi et al., 2017). iCaRL selects a few prototypes per class, stores these, and then, at prediction time, averages the prototypes for each class (using the current representation) and chooses the one nearest the query vector. FlyModel maintains one linear function per class and, at prediction time, takes the one with the highest value.

There are four additional features of the fruit fly mushroom body (MB) that remain underexplored computationally. First, instead of using one output neuron (MBON) per behavior, the mushroom body contains multiple output neurons per behavior, with each output neuron learning at a different rate (Hige et al., 2015; Aso & Rubin, 2016). This simultaneously provides fast learning with poor retention (large learning rates) and slow learning with longer retention (small learning rates), which is reminiscent of complementary learning systems (Parisi et al., 2019). Second, the MB contains mechanisms for memory extinction (Felsenberg et al., 2018) and reversal learning (Felsenberg et al., 2017; Felsenberg, 2021), which are used to update inaccurate memories. Third, there is evidence of memory replay in the MB, which is required for memory consolidation (Yu et al., 2005; Haynes et al., 2015; Cognigni et al., 2018). Fourth, there exists feedback from the MB to the input layer that could tune representations during learning (Hu et al., 2010). We hope our model can be used as a stepping stone as circuit mechanisms controlling these computations are discovered. Moreover, although we only evaluated continual learning performance using simple architectures, follow-up work has already successfully implemented some variants of FlyModel (Robinson et al., 2023; Bricken et al., 2023), suggesting that better performance can indeed be achieved using more sophisticated architectures.

Finally, a motif similar to that of the fruit fly olfactory system also appears in the mouse olfactory system, where sparse representations in the piriform cortex project to other learning-related areas of the brain (Komiyama & Luo, 2006; Wang et al., 2020). In addition, the visual system uses many successive layers to extract discriminative features (Riesenhuber & Poggio, 1999; Tacchetti et al., 2018), which are then projected to the hippocampus, where a similar sparse, high-dimensional representation is used for memory storage (Olshausen & Field, 2004; Wixted et al., 2014; Lodge & Bischofberger, 2019). Thus, the principles of learning studied here may help illuminate how continual learning is implemented in other brain regions and species.

In all, our work exemplifies how understanding detailed neural anatomy and physiology in a tractable model system can be translated into efficient architectures for use in artificial neural networks.

## References

Ahmad, S., & Scheinkman, L. (2019). How can we be so dense? The benefits of using highly sparse representations. *CoRR*, abs/1903.11257.

Aso, Y., Hattori, D., Yu, Y., Johnston, R. M., Iyer, N. A., Ngo, T.-T., . . . Rubin, G. M. (2014). The neuronal architecture of the mushroom body provides a logic for associative learning. *eLife*, *3*, e04577.

Aso, Y., & Rubin, G. M. (2016). Dopaminergic neurons write and update memories with cell-type-specific rules. *eLife*, *5*. 10.7554/eLife.16135

Babadi, B., & Sompolinsky, H. (2014). Sparseness and expansion in sensory representations. *Neuron*, *83*(5), 1213–1226. 10.1016/j.neuron.2014.07.035

Benna, M. K., & Fusi, S. (2016). Computational principles of synaptic memory consolidation. *Nature Neuroscience*, *19*(12), 1697–1706. 10.1038/nn.4401

Bricken, T., Davies, X., Singh, D., Krotov, D., & Kreiman, G. (2023). *Sparse distributed memory is a continual learner*. arXiv:2303.11934.

Caron, S. J., Ruta, V., Abbott, L., & Axel, R. (2013). Random convergence of olfactory inputs in the Drosophila mushroom body. *Nature*, *497*(7447), 113–117. 10.1038/nature12063

Cayco-Gajic, N. A., & Silver, R. A. (2019). Re-evaluating circuit mechanisms underlying pattern separation. *Neuron*, *101*(4), 584–602. 10.1016/j.neuron.2019.01.044

Cognigni, P., Felsenberg, J., & Waddell, S. (2018). Do the right thing: Neural network mechanisms of memory formation, expression and update in Drosophila. *Current Opinion in Neurobiology*, *49*, 51–58. 10.1016/j.conb.2017.12.002

Dasgupta, S., Sheehan, T. C., Stevens, C. F., & Navlakha, S. (2018). A neural data structure for novelty detection. *Proceedings of the National Academy of Sciences USA*, *115*(51), 13093–13098. 10.1073/pnas.1814448115

Dasgupta, S., Stevens, C. F., & Navlakha, S. (2017). A neural algorithm for a fundamental computing problem. *Science*, *358*(6364), 793–796. 10.1126/science.aam9868

Dasgupta, S., & Tosh, C. (2020). *Expressivity of expand-and-sparsify representations*. arXiv:2006.03741

Douillard, A., Cord, M., Ollion, C., Robert, T., & Valle, E. (2020). Podnet: Pooled outputs distillation for small-tasks incremental learning. In *Proceedings of the IEEE European Conference on Computer Vision*.

Farquhar, S., & Gal, Y. (2019). *Towards robust evaluations of continual learning*. arXiv:1805.09733v3.

Felsenberg, J. (2021). Changing memories on the fly: The neural circuits of memory re-evaluation in *Drosophila melanogaster*. *Current Opinion in Neurobiology*, *67*, 190–198. 10.1016/j.conb.2020.12.003

Felsenberg, J., Barnstedt, O., Cognigni, P., Lin, S., & Waddell, S. (2017). Reevaluation of learned information in Drosophila. *Nature*, *544*(7649), 240–244. 10.1038/nature21716

Felsenberg, J., Jacob, P. F., Walker, T., Barnstedt, O., Edmondson-Stait, A. J., Pleijzier, M. W., . . . Waddell, S. (2018). Integration of parallel opposing memories underlies memory extinction. *Cell*, *175*(3), 709–722.

Fusi, S., Drew, P. J., & Abbott, L. F. (2005). Cascade models of synaptically stored memories. *Neuron*, *45*(4), 599–611. 10.1016/j.neuron.2005.02.001

Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., & Bengio, Y. (2013). *An empirical investigation of catastrophic forgetting in gradient-based neural networks*. arXiv:1312.6211.

Gorur-Shandilya, S., Demir, M., Long, J., Clark, D. A., & Emonet, T. (2017). Olfactory receptor neurons use gain control and complementary kinetics to encode intermittent odorant stimuli. *eLife*, *6*. 10.7554/eLife.27670

Hand, D. J. (2006). Classifier technology and the illusion of progress. *Statistical Science*, *21*(1), 1–14.

Haynes, P. R., Christmann, B. L., & Griffith, L. C. (2015). A single pair of neurons links sleep to memory consolidation in *Drosophila melanogaster*. *eLife*, *4*. 10.7554/eLife.03868

Hige, T., Aso, Y., Modi, M. N., Rubin, G. M., & Turner, G. C. (2015). Heterosynaptic plasticity underlies aversive olfactory learning in Drosophila. *Neuron*, *88*(5), 985–998. 10.1016/j.neuron.2015.11.003

Hinton, G. E., & Plaut, D. C. (1987). Using fast weights to deblur old memories. In *Proceedings of the 9th Annual Conference of the Cognitive Science Society* (pp. 177–186). Erlbaum.

Hitron, Y., Lynch, N., Musco, C., & Parter, M. (2020). Random sketching, clustering, and short-term memory in spiking neural networks. In T. Vidick (Ed.), *11th Innovations in Theoretical Computer Science Conference*, *151* (pp. 23:1–23:31). Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Holca-Lamarre, R., Lücke, J., & Obermayer, K. (2017). Models of acetylcholine and dopamine signals differentially improve neural representations. *Frontiers in Computational Neuroscience*, *11*, 54. 10.3389/fncom.2017.00054

Hu, A., Zhang, W., & Wang, Z. (2010). Functional feedback from mushroom bodies to antennal lobes in the Drosophila olfactory pathway. *Proceedings of the National Academy of Sciences*, *107*(22), 10262–10267. 10.1073/pnas.0914912107

Ji, D., & Wilson, M. A. (2007). Coordinated memory replay in the visual cortex and hippocampus during sleep. *Nature Neuroscience*, *10*(1), 100–107. 10.1038/nn1825

Kanerva, P. (1988). *Sparse distributed memory*. MIT Press.

Kanerva, P. (2009). Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation*, *1*(2), 139–159. 10.1007/s12559-009-9009-8

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., . . . Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences USA*, *114*(13), 3521–3526. 10.1073/pnas.1611835114

Komiyama, T., & Luo, L. (2006). Development of wiring specificity in the olfactory system. *Current Opinion in Neurobiology*, *16*(1), 67–73. 10.1016/j.conb.2005.12.002

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. 10.1038/nature14539

Li, F., Lindsey, J. W., Marin, E. C., Otto, N., Dreher, M., Dempsey, G., . . . Rubin, G. M. (2020). The connectome of the adult Drosophila mushroom body provides insights into function. *eLife*, *9*.

Lin, A. C., Bygrave, A. M., de Calignon, A., Lee, T., & Miesenböck, G. (2014). Sparse, decorrelated odor coding in the mushroom body enhances learned odor discrimination. *Nature Neuroscience*, *17*(4), 559–568. 10.1038/nn.3660

Litwin-Kumar, A., Harris, K. D., Axel, R., Sompolinsky, H., & Abbott, L. F. (2017). Optimal degrees of synaptic connectivity. *Neuron*, *93*(5), 1153–1164. 10.1016/j.neuron.2017.01.030

Liu, X., & Davis, R. L. (2009). The GABAergic anterior paired lateral neuron suppresses and is suppressed by olfactory learning. *Nature Neuroscience*, *12*(1), 53–59. 10.1038/nn.2235

Lodge, M., & Bischofberger, J. (2019). Synaptic properties of newly generated granule cells support sparse coding in the adult hippocampus. *Behavioural Brain Research*, *372*, 112036. 10.1016/j.bbr.2019.112036

Lopez-Paz, D., & Ranzato, M. (2017). Gradient episodic memory for continual learning. In I. Guyon, Y. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems*, *30* (pp. 6467–6476). Curran.

Mallya, A., & Lazebnik, S. (2018). Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 7765–7773).

Maurer, A., Pontil, M., & Romera-Paredes, B. (2013). Sparse coding for multitask and transfer learning. In *Proceedings of the 30th International Conference on Machine Learning*.

McClelland, J. L., McNaughton, B. L., & O'Reilly, R. C. (1995). Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. *Psychological Review*, *102*(3), 419–457. 10.1037/0033-295X.102.3.419

Minsky, M. L., & Papert, S. A. (1988). *Perceptrons* (Exp. ed.). MIT Press.

Modi, M. N., Shuai, Y., & Turner, G. C. (2020). The Drosophila mushroom body: From architecture to algorithm in a learning circuit. *Annual Review of Neuroscience*, *43*, 465–484. 10.1146/annurev-neuro-080317-0621333

Olsen, S. R., Bhandawat, V., & Wilson, R. I. (2010). Divisive normalization in olfactory population codes. *Neuron*, *66*(2), 287–299. 10.1016/j.neuron.2010.04.009

Olshausen, B. A., & Field, D. J. (2004). Sparse coding of sensory inputs. *Current Opinion in Neurobiology*, *14*(4), 481–487. 10.1016/j.conb.2004.07.007

Ororbia, A., Mali, A., Kifer, D., & Giles, C. L. (2019). Lifelong neural predictive coding: Sparsity yields less forgetting when learning cumulatively. *CoRR*, abs/1905.10696.

Papadimitriou, C. H., & Vempala, S. S. (2018). Random projection in the brain and computation with assemblies of neurons. In A. Blum (Ed.)., *10th Innovations in Theoretical Computer Science Conference*, *124* (pp. 57:1–57:19). Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., & Wermter, S. (2019). Continual lifelong learning with neural networks: A review. *Neural Networks*, *113*, 54–71. 10.1016/j.neunet.2019.01.012

Peng, J., Tang, B., Jiang, H., Li, Z., Lei, Y., Lin, T., & Li, H. (2021). Overcoming long-term catastrophic forgetting through adversarial neural pruning and synaptic consolidation. *IEEE Transactions on Neural Networks and Learning Systems*, *33*(9), 4243–4256. 10.1109/TNNLS.2021.3056201

Qin, Y. L., McNaughton, B. L., Skaggs, W. E., & Barnes, C. A. (1997). Memory reprocessing in corticocortical and hippocampocortical neuronal ensembles. *Philosophical Transactions of the Royal Society B: Biological Sciences*, *352*(1360), 1525–1533. 10.1098/rstb.1997.0139

Ramasesh, V. V., Dyer, E., & Raghu, M. (2021). Anatomy of catastrophic forgetting: Hidden representations and task semantics. In *Proceedings of the International Conference on Learning Representations*.

Rapp, H., & Nawrot, M. P. (2020). A spiking neural program for sensorimotor control during foraging in flying insects. *Proceedings of the National Academy of Sciences*, *117*(45), 28412–28421. 10.1073/pnas.2009821117

Rasch, B., & Born, J. (2007). Maintaining memories by reactivation. *Current Opinion in Neurobiology*, *17*(6), 698–703. 10.1016/j.conb.2007.11.007

Rebuffi, S.-A., Kolesnikov, A., Sperl, G., & Lampert, C. H. (2017). iCaRL: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2001–2010).

Riesenhuber, M., & Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nature Neuroscience*, *2*(11), 1019–1025. 10.1038/14819

Robinson, B. S., Joyce, J., Norman-Tenazas, R., Vallabha, G. K., & Johnson, E. C. (2023). *Informing generative replay for continual learning with long-term memory formation in the fruit fly*. bioRxiv:2023–01.

Root, C. M., Masuyama, K., Green, D. S., Enell, L. E., Nässel, D. R., Lee, C. H., & Wang, J. W. (2008). A presynaptic gain control mechanism fine-tunes olfactory behavior. *Neuron*, *59*(2), 311–321. 10.1016/j.neuron.2008.07.003

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386. 10.1037/h0042519

Roxin, A., & Fusi, S. (2013). Efficient partitioning of memory systems and its importance for memory consolidation. *PLOS Computational Biology*, *9*(7), e1003146. 10.1371/journal.pcbi.1003146

Ruvolo, P., & Eaton, E. (2013). ELLA: An efficient lifelong learning algorithm. In S. Dasgupta & D. McAllester (Eds.), *Proceedings of the 30th International Conference on Machine Learning*.

Ryali, C., Hopfield, J., Grinberg, L., & Krotov, D. (2020). Bio-inspired hashing for unsupervised similarity search. In *Proceedings of the 37th International Conference on Machine Learning*.

Shin, H., Lee, J. K., Kim, J., & Kim, J. (2017). *Continual learning with deep generative replay*. arXiv:1705.08690v3.

Stevens, C. F. (2015). What the fly's nose tells the fly's brain. *Proceedings of the National Academy of Sciences*, *112*(30), 9460–9465. 10.1073/pnas.1510103112

Tacchetti, A., Isik, L., & Poggio, T. A. (2018). Invariant recognition shapes neural representations of visual input. *Annual Review of Vision Science*, *4*, 403–422. 10.1146/annurev-vision-091517-034103

Tadros, T., Krishnan, G. P., Ramyaa, R., & Bazhenov, M. (2020). Biologically inspired sleep algorithm for reducing catastrophic forgetting in neural networks. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence* (pp. 13933–13934).

Tadros, T., Krishnan, G. P., Ramyaa, R., & Bazhenov, M. (2022). Sleep-like unsupervised replay reduces catastrophic forgetting in artificial neural networks. *Nature Communications*, *13*(1), 7742. 10.1038/s41467-022-34938-7

Takemura, S. Y., Aso, Y., Hige, T., Wong, A., Lu, Z., Xu, C. S., . . . Scheffer, L. K. (2017). A connectome of a learning and memory center in the adult Drosophila brain. *eLife*, *6*.

Turner, G. C., Bazhenov, M., & Laurent, G. (2008). Olfactory representations by Drosophila mushroom body neurons. *Journal of Neurophysiology*, *99*(2), 734–746. 10.1152/jn.01283.2007

van de Ven, G. M., Siegelmann, H. T., & Tolias, A. S. (2020). Brain-inspired replay for continual learning with artificial neural networks. *Nature Communications*, *11*(1), 4069. 10.1038/s41467-020-17866-2

Wang, P. Y., Boboila, C., Chin, M., Higashi-Howard, A., Shamash, P., . . . Axel, R. (2020). Transient and persistent representations of odor value in prefrontal cortex. *Neuron*, *108*(1), 209–224. 10.1016/j.neuron.2020.07.033

Wilson, M. A., & McNaughton, B. L. (1994). Reactivation of hippocampal ensemble memories during sleep. *Science*, *265*(5172), 676–679. 10.1126/science.8036517

Wilson, R. I. (2013). Early olfactory processing in Drosophila: Mechanisms and principles. *Annual Review of Neuroscience*, *36*, 217–241. 10.1146/annurev-neuro-062111-150533

Wixted, J. T., Squire, L. R., Jang, Y., Papesh, M. H., Goldinger, S. D., Kuhn, J. R., . . . Steinmetz, P. N. (2014). Sparse and distributed coding of episodic memory in neurons of the human hippocampus. *Proceedings of the National Academy of Sciences*, *111*(26), 9621–9626. 10.1073/pnas.1408365111

Yu, D., Keene, A. C., Srivatsan, A., Waddell, S., & Davis, R. L. (2005). Drosophila DPM neurons form a delayed and branch-specific memory trace after olfactory classical conditioning. *Cell*, *123*(5), 945–957.

Zenke, F., Poole, B., & Ganguli, S. (2017). Continual learning through synaptic intelligence. *Proceedings of Machine Learning Research*, *70*, 3987–3995.

Zhang, Y., & Sharpee, T. O. (2016). A robust feedforward model of the olfactory system. *PLOS Computational Biology*, *12*(4), e1004850.

Zheng, Z., Lauritzen, J. S., Perlman, E., Robinson, C. G., Nichols, M., Milkie, D., . . . Bock, D. D. (2018). A complete electron microscopy volume of the brain of adult *Drosophila melanogaster*. *Cell*, *174*(3), 730–743.

Received February 4, 2023; accepted July 14, 2023.