A Systems Neuroscience Approach to Continual Learning

Nand Chandravadia
Computer Science
Columbia University
ndc2136@columbia.edu

Nabil Imam
College of Computing
Georgia Institute of Technology
nimam6@gatech.edu

Abstract

How can we endow artificial neural networks with the capacity for life-long (i.e., continual) learning? Recently, algorithmic insights into the olfactory (neural) circuit of the *Drosophila Melanogaster* have been discovered that endows it with the ability to acquire novel information, while preserving old memories. In this work, we leverage these algorithmic insights, notably sparse coding via a random projection, a k-winner-take-all computation, (partial) synaptic freezing, and associative learning, to build an artificial system with the ability of continual learning. [1]

1 Introduction

Continual learning - the ability to acquire novel information, while preserving old memories - is a hallmark attribute of intelligent agents, including humans and insects. In the real-world, these biological agents learn tasks in a sequential (e.g., continual) life-long manner, avoiding destructive interference (“forgetting") of past learned information.

A long-standing problem in machine learning, however, is the inability of artificial neural networks to consolidate and retain information learned from the past when learning a novel task - this is known as “catastrophic forgetting" [Fre99]. Although artificial neural networks have achieved superhuman results in the classical setting (e.g., supervised training on a single episode/task), in the sequential learning regime, where the artificial neural network has to learn multiple episodes in sequence, the problem becomes much more difficult, leading to catastrophic interference. The problem arises due to the weights of the network being optimized to each episode individually, yielding poor transfer of learned information among episodes.

Nature has found a solution to the sequential learning problem, endowing biological agents with the capability of life-long learning [Zel+06]. Thereby, can we leverage insights from nature to endow artificial agents with a similar capability? This is the central goal of this work. In particular, we leverage algorithmic insights from the olfactory neural circuit of the *Drosophila Melanogaster* to alleviate catastrophic forgetting in artificial agents [She21].

2 Circuit Motif: inspiration from the *Drosophila Melanogaster*

Throughout its lifetime, the *Drosophila Melanogaster* must learn to navigate noisy, uncertain environments, while simultaneously preserving past information learned to ensure its continual survival, mapping ‘averse’ odors to avoidance behaviors, while mapping ‘benign’ odors to approach behaviors. The olfactory (neural) circuit accomplishes this feat in a robust and life-long manner. What are the key features of this circuit that enable this life-long learning?

[1] Code available at https://github.com/nandchandravadia/ASystemsNeuroscienceApproachtoContinualLearning

Due to recent advances in systems and experimental neuroscience, a detailed landscape of the neural architecture of the olfactory system has been catalogued at the synaptic and connectomic level. In fact, the connectome of the *Drosophila Larva* has been fully reconstructed, offering structural and computational insights into the underlying processes that subserve its myraid of behaviors [Win22].

In particular, we focus on a two-layer (olfactory) circuit motif that alleviates the "catastrophic forgetting" problem - by capitalizing on the following algorithmic insights: sparse coding via a random projection, a k-winner-take-all computation, (partial) synaptic freezing, and associative learning. How does the *Drosophila* map sensory inputs into outputs that enable life-long learning?

Circuit Motif Briefly, we detail the two-layer neural circuit in the *Drosophila Melongoster* in a biological context. Then, we provide a (generalized) computational formalization (in Section 3) of this circuit motif.

Sensory Projection First, inputs to the fly olfactory circuit come in the form of odors, represented as $\vec{x}$ in some d-dimensional space. These inputs odors, $\vec{x}$, undergo *sensory processing* (e.g., gain control, noise reduction, and normalization), yielding $\vec{y} \in \mathbb{R}^{50}$, which represents the firing rates of 50 projection neurons (PNs) such that $\vec{y}_i \sim Exponential(\lambda)$.

Sparse Coding Then, these 50 PNs get randomly projected (via a sparse, random dimensionality expansion) onto 2000 Kenyon Cells (KCs), yielding $\vec{c} \in \mathbb{R}^{2000}$. Each c_i sends an excitatory signal to an inhibitory anterior paired lateral (APL) neuron, which, in turns, sends a proportional inhibitory signal to each c_i, yielding a sparse, high-dimensional representation of our original input, $\vec{x}$.

Associative Learning Finally, similar odors $\vec{x}$'s get mapped onto similar sparse, high-dimensional representations $\vec{c}$'s such that input-output behavioral pairing(s) are learned.

Thereby, we capitalize on the above computational strategies of the olfactory (neural) circuit to endow an artificial system with similar capabilities.

3 FlyModel

The first layer (Fig. 1) in our circuit computes a sparse, high-dimensional representation of our original input $\vec{\hat{x}}$ as follows:

$$\vec{c} = W^{(1)}\vec{\hat{x}}$$

where $W^{(1)} \in \mathbb{R}^{m \times d}$, $\vec{c} \in \mathbb{R}^{m \times 1}$, and $m \approx 40d$. What is the purpose of this dimensionality expansion, $m \approx 40d$? Recall, in the original fly circuit, we had $m = 50$ and $d = 2000$. As such, this represents a 40-fold dimensionality expansion. $W^{(1)}$ is a sparse, binary random matrix, where the sparsity is defined as $0.1d$; that is, each row of $W^{(1)}$ is a sparse, high dimensional vector with 1's in random positions. What is the purpose of defining the sparsity as $0.1d$? Recall, in the original fly olfactory circuit, each KC, c_i, samples from 6 of the 50 PN's (or approximately 12% of the PNs).

Further, we apply a k-winner-take-all process (also known as a k-winner-take-all computation) on the initial representation in our circuit, $\vec{c}$, which yields the top k active $\vec{c}_i$'s to stay on ($= 1$), while the rest are turned off ($= 0$). Our k-winner-take-all computation is thereby defined by the following sparsification scheme:

$$\vec{c}_{max_k} = \text{the top } k \text{ entries of } \vec{c}$$

$$\vec{c}^{\,sparse} = sparsify(\vec{c}) = \begin{cases} c_i & \text{if } c_i \in \vec{c}_{max_k} \\ 0 & \text{otherwise} \end{cases}$$

where the sparsification parameter, $k \approx 0.05m$. How do we determine k, the sparsification parameter? Recall, in the fly olfactory ciruit, the top 5% of the KC's remained active, while the

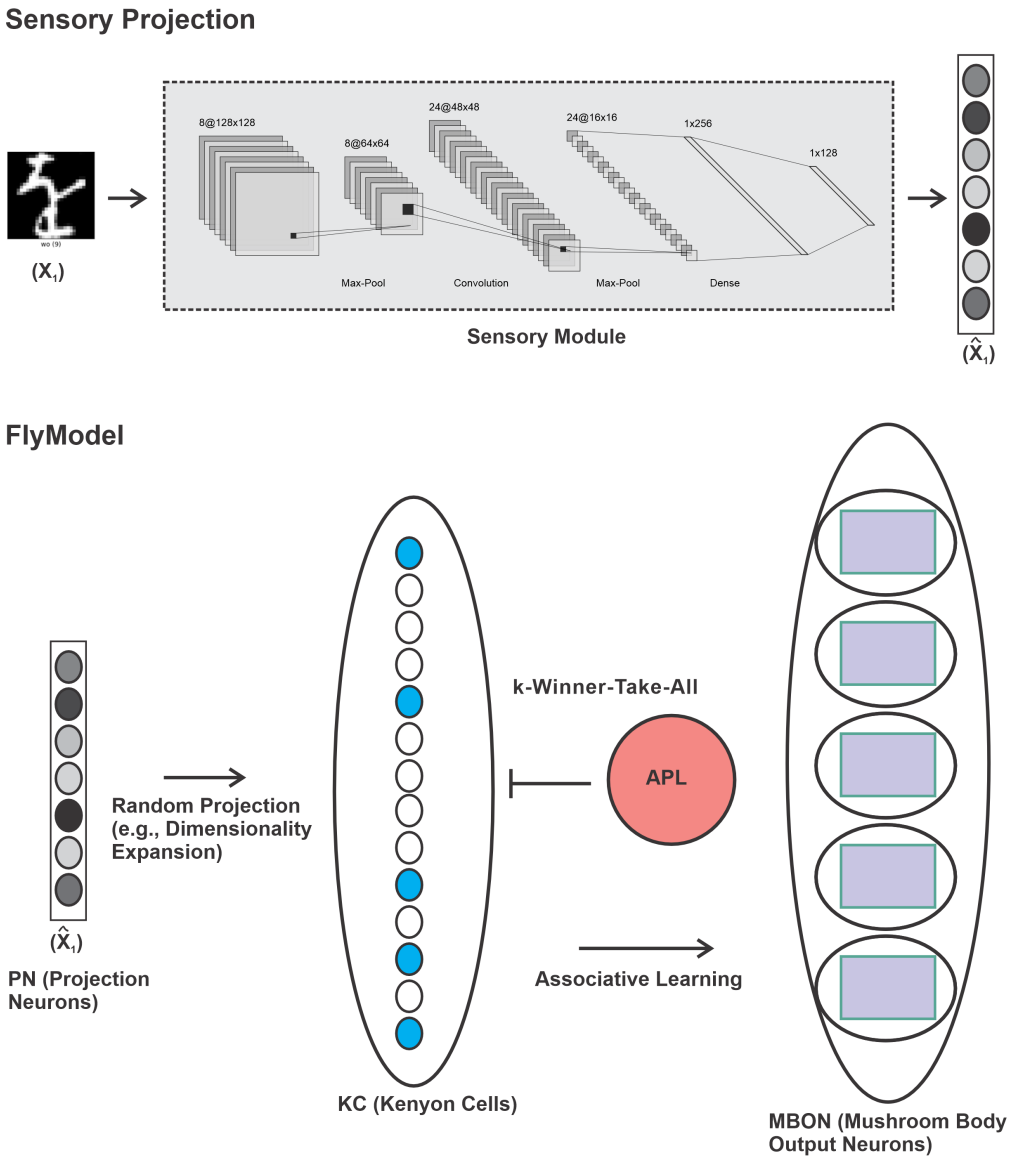

Figure 1: **FlyModel**. Top (*Sensory Projection*): Each input x is fed into a sensory module (LeNet5), yielding a representation $\vec{\hat{x}}$. The representation, $\vec{\hat{x}}$, is from the penultimate layer of the sensory module (in this case, $\vec{\hat{x}} \in \mathbb{R}^{84}$). Bottom (*FlyModel*): Each $\vec{\hat{x}}$ is randomly projected, yielding a sparse, high-dimensional representation (KCs). Blue = On; White = Off. Each KC sends an excitatory signal to an inhibitory anterior paired lateral (APL) neuron, which sends a proportional inhibitory signal to each KC. Computationally, this is represented as a k-WTA (k-winner-take-all) process. The connections between a KC and each MBON is all-to-all, and paired via associative learning.

bottom 95% turned off; that is, only 100 out of the 2000 KC's remained active after stimulus input, which is 5% of all KC's.

Moreover, we apply a min-max normalization such that the values $\in sparsify(\vec{c})$ are between $[0, 1]$:

$$\vec{c}^{\,sparse}_{max} = max(\vec{c}^{\,sparse})$$

$$\vec{c}^{\,sparse}_{min} = min(\vec{c}^{\,sparse})$$

$$\hat{c}_i = \frac{\vec{c}^{\,sparse}_i - \vec{c}^{\,sparse}_{min}}{\vec{c}^{\,sparse}_{max} - \vec{c}^{\,sparse}_{min}}$$

Thereby, as a result of the computation in our first layer, we yield:

$$\hat{\vec{c}} \in \mathbb{R}^m_{[0,1]}$$

The second layer in our circuit is the associative learning layer, where the goal is to learn the association between an input-output, such as $(\vec{x}, y)$. We have N output class units, $\vec{o} = \{o_1, \ldots, o_N\}$. Each output class unit, o_i corresponds to an MBON. The connections between our c_i's and o_j's are all-to-all (e.g., fully-connected), meaning every synapse between c_i and o_j is unique. Thereby, we have:

$$\vec{o} = W^{(2)}\hat{\vec{c}}$$

where $W^{(2)} \in \mathbb{R}^{m \times N}_{[0,1]}$. Unlike $W^{(1)}$, the weights in this layer, $W^{(2)}$, are learned (i.e., trainable) via a biologically plausible, associative learning-rule. That is, on each training step t, we update our weights, $w^{(t)}_{i,j}$, according to:

$$w^{(t+1)}_{i,j} = \begin{cases} (1-\alpha)w^{(t)}_{i,j} + \beta\hat{c}_i & \text{if } j = \text{target class} \\ (1-\alpha)w^{(t)}_{i,j} & \text{otherwise} \end{cases}$$

where α = forgetting term (which mimics slow, background memory decay) and β = learning rate. Since biological synapses have a threshold on the value $w_{i,j}$ assumes, we cap the weights $w_{i,j} \in [0, 1]$. Typically, we let $\alpha = 0$ (no memory decay).

4 Methods

4.1 Continual Learning Framework

The generalized continual learning problem is framed as follows: a singular neural network must sequentially learn a series of n tasks $t_i \in T$ for $i = [n]$, where only data X_i from the current task is available. In particular, we employ a class-incremental learning setup (class-IL), albeit various continual learning scenarios have been proposed each with varying levels of difficulty, such as task-incremental learning (task-IL) or domain-incremental learning (domain-IL) [Ven19].

In class-IL, the goal is to sequentially learn each task t_i seen thus far without the task-ID provided. Formally, during training, on each task i, the network is presented with (X_i, y_i). During testing, the network is only presented with (X_i), inferring the task-ID.

In particular, we employ 'Split-MNIST,' a continual learning variant of the MNIST dataset (Fig. 2). In Split-MNIST, we have $n = 5$ non-overlapping episodes, where each episode consists of data from $k = 2$ classes. For instance, in episode t_1, we have data from classes $\{0, 1\}$; in episode t_2, we have data from classes $\{2, 3\}$; in episode t_3, we have data from classes $\{4, 5\}$; in episode t_4, we have data from classes $\{6, 7\}$; in episode t_5, we have data from classes $\{8, 9\}$. Although we instantiated a simple version of Split-MNIST, where $n = 5$ and $k = 2$, one may modify these parameters to alter the task structure.

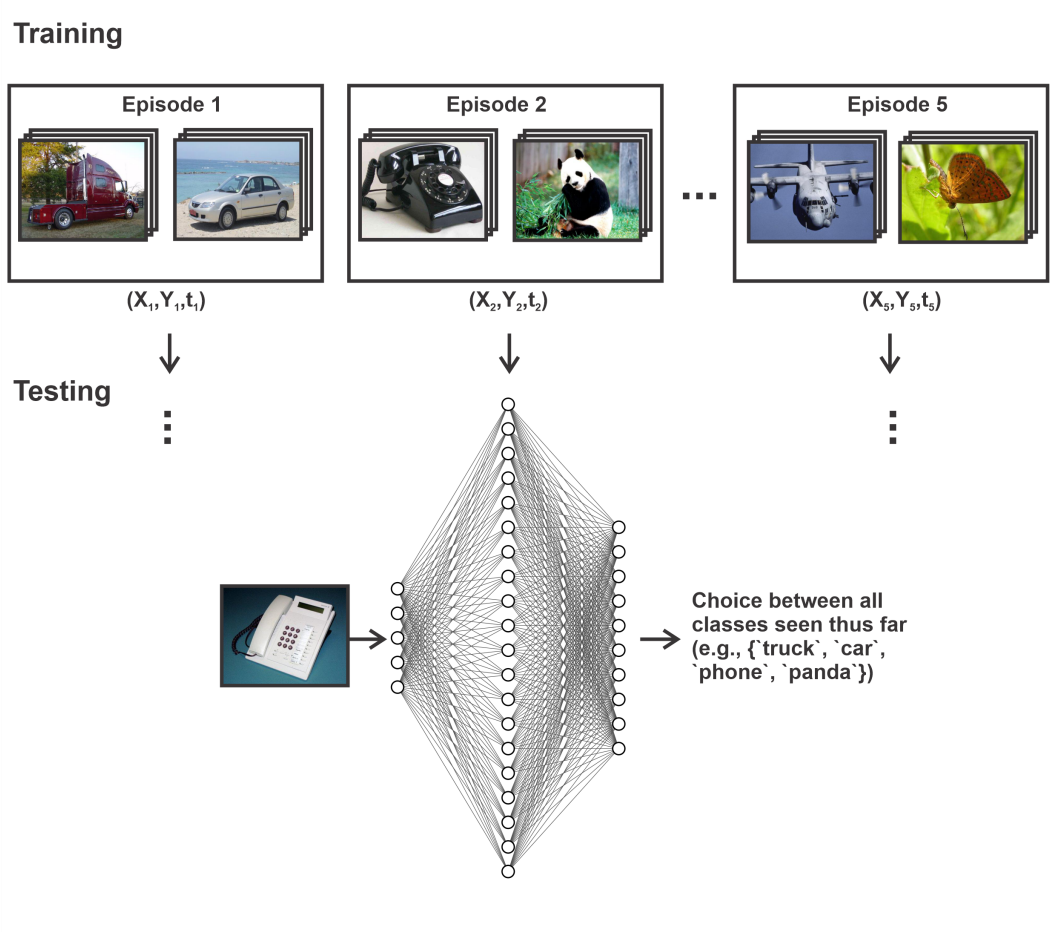

Figure 2: **Task Structure.** Top (*Training*): Each episode, t_i, consists of images from $k = 2$ classes, where X_i = images from $k = 2$ classes, y_i = labels from $k = 2$ classes, and t_i = each task. Both training and testing happen sequentially; that is, we perform $(train_1, test_1)$, $(train_2, test_2)$, ..., $(train_5, test_5)$. Bottom (*Testing*): During testing, the network is presented with images from all tasks $\leq i$.

4.2 Problem Setup

Every input, **x**, into the network is loaded into a sensory module, yielding a representation $\vec{x}$. We call this step the *sensory projection*. Since the goal of sensory processing to encode discriminative features and remove noise, we trained LeNet5 on KMNIST (Kuzushiji-MNIST) as our sensory module (Fig. 1) [LeC+89]. Thereby, for every MNIST image, $\mathbf{x}_1$, we input into the sensory module, we took the representation at the penultimate layer, which is $\vec{x} \in \mathbb{R}^{84}$, as input into each network. The representation $\vec{x}$ is analogous to the odor representation of the PNs.

4.3 Evaluation

How do we benchmark the performance of our proposed network and juxtapose its performance with the other continual learning models? We present two testing regimes to assess model performance in a continual learning framework.

The first measure is the **accuracy of the model on the classes trained thus far**. The goal of this measure is to assess how well our model "remembers" the classes learned thus far. Does learning a new task interfere with past, learned information? That is, after training on task i, we assess the

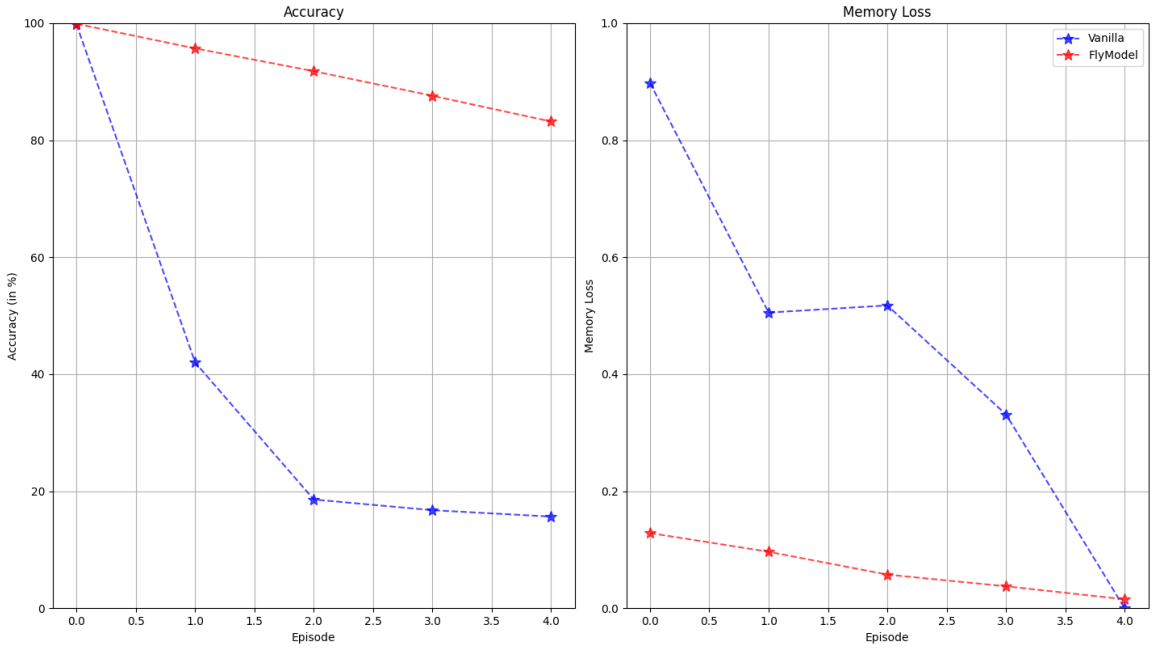

Figure 3: **Accuracy and Memory Loss.** Left (*Accuracy*): . Right (*Memory Loss*)

model accuracy by testing on all classes from tasks $\leq i$. For instance, suppose after training on task 3, we present the model with test data with the following classes, {0,1,2,3,4,5} (note, in this setup, we assume each task involves binary classification).

The second measure is **memory loss**. How well does the model remember each task individually? Memory loss is a metric that quantifies how much the model "forgets" each task individually. Formally, memory loss is computed as follows:

Memory Loss (on task i) = Accuracy on task i (*immediately after training on task* i) - Accuracy on task i (immediately after training on all tasks)

A larger memory loss corresponds to more "forgetting," while a smaller memory loss corresponds to better "remembering." Ideally, we want to minimize memory loss to preserve information learned in the past.

4.4 Network Architectures

How do we comparably assess the performance of the FlyModel with other network architectures? We standardize all neural networks to share the same network architecture; that is, all networks share the same number of hidden layers and hidden units. In particular, we compare the *FlyModel* to a standard (*Vanilla*) neural network (trained via backpropogation) (Table 1).

	Hidden Layers	Learning Rate	Batch Size
FlyModel	2	0.01	1
Vanilla	2	0.01	64

Table 1: **Network Parameters**

5 Results

In the continual learning framework, the FlyModel (trained via associative learning) significantly outperforms the standard 'Vanilla' network (trained via backpropogation) in both overall accuracy

and memory loss, thereby preserving past learned information and mitigating catastrophic forgetting (Fig. 3).

On each episode t_i, the *FlyModel* achieved a significantly higher accuracy, compared to the *Vanilla* network (besides episode t_1, where the network is not required to remember any past information), suggesting the *FlyModel* has a better recollection of information learned from the past and thereby forgets less information. While the accuracy on successive episodes drops approximately linearly for the *FlyModel*, the drop-off in the *Vanilla* model is considerably more acute. For instance, the *Vanilla* model observes a $\sim 40\%$ decline in accuracy between episode t_1 and t_2, while the *FlyModel* observes only a $\sim 4\%$ decline. As such, across all episodes t_i, the FlyModel achieved an average accuracy of $\sim 92\%$, while the *Vanilla* network achieved an average accuracy of $\sim 39\%$.

Furthermore, on each episode t_i, the *FlyModel* achieved a significantly lower memory loss, compared to the *Vanilla* network (besides episode t_5, where the task ends), suggesting the *FlyModel* 'forgets' less information learned from previous tasks. For instance, between episode t_4 and t_5, the *FlyModel* observed a Δ in memory loss of 0.02, while the *Vanilla* observed a Δ in memory loss of 0.33. On average, the *FlyModel* witnessed a memory loss of 0.067, while the *Vanilla* network witnessed an average memory loss of 0.45.

6 Discussion

Standard neural networks suffer from "catastrophic forgetting" when learning novel tasks in a sequential manner. However, the sequential learning regime is a natural context for human and insect learning. These biological agents have been endowed with computational capabilities in this learning setting.

Capitalizing on computational insights from the *Drosophila Melanogaster*, we endowed an artifical neural system with the capacity of life-long learning, mitigating "catastrophic forgetting."

References

[LeC+89] Yann LeCun et al. "Backpropagation applied to handwritten zip code recognition". In: *Neural computation* 1.4 (1989), pp. 541–551.

[Fre99] Robert M French. "Catastrophic forgetting in connectionist networks." In: *Trends in cognitive sciences* 3.4 (Apr. 1999), pp. 128–135.

[Zel+06] Tibor Zelles et al. "Branch-specific Ca2+ influx from Na+-dependent dendritic spikes in olfactory granule cells." In: *Journal of Neuroscience* 26.1 (Jan. 2006), pp. 30–40.

[Ven19] Gido M. van de Ven, et al. "Three scenarios for continual learning". In: *arXiv* (2019).

[She21] Yang Shen, et al. "Algorithmic insights on continual learning from fruit flies". In: *arXiv* (2021).

[Win22] Michael Winding, et al. "The connectome of an insect brain". In: *biorXiv* (2022).