frontiers | Frontiers in Behavioral Neuroscience

ORIGINAL RESEARCH
published: 06 April 2022
doi: 10.3389/fnbeh.2022.835753

Bottlenecks, Modularity, and the Neural Control of Behavior

Anjalika Nande[1,2†], *Veronika Dubinkina*[3,4†], *Riccardo Ravasio*[5,6†], *Grace H. Zhang*[1†] *and Gordon J. Berman*[7*]

[1] Department of Physics, Harvard University, Cambridge, MA, United States, [2] Institute for Computational Medicine, Johns Hopkins University, Baltimore, MD, United States, [3] Department of Bioengineering, University of Illinois at Urbana-Champaign, Urbana, IL, United States, [4] Carl R. Woese Institute for Genomic Biology, University of Illinois at Urbana-Champaign, Urbana, IL, United States, [5] Institute of Physics, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, [6] The James Franck Institute, University of Chicago, Chicago, IL, United States, [7] Departments of Biology and Physics, Emory University, Atlanta, GA, United States

OPEN ACCESS

Edited by:
Ilona C. Grunwald Kadow, Technical University of Munich, Germany

Reviewed by:
Moshe Parnas, Tel Aviv University, Israel
Carlotta Martelli, Johannes Gutenberg University Mainz, Germany

***Correspondence:**
Gordon J. Berman
gordon.berman@emory.edu

[†] *These authors have contributed equally to this work*

Specialty section:
This article was submitted to Individual and Social Behaviors, a section of the journal Frontiers in Behavioral Neuroscience

Received: *14 December 2021*
Accepted: *14 March 2022*
Published: *06 April 2022*

Citation:
Nande A, Dubinkina V, Ravasio R, Zhang GH and Berman GJ (2022) Bottlenecks, Modularity, and the Neural Control of Behavior. Front. Behav. Neurosci. 16:835753. doi: 10.3389/fnbeh.2022.835753

In almost all animals, the transfer of information from the brain to the motor circuitry is facilitated by a relatively small number of neurons, leading to a constraint on the amount of information that can be transmitted. Our knowledge of how animals encode information through this pathway, and the consequences of this encoding, however, is limited. In this study, we use a simple feed-forward neural network to investigate the consequences of having such a bottleneck and identify aspects of the network architecture that enable robust information transfer. We are able to explain some recently observed properties of descending neurons—that they exhibit a modular pattern of connectivity and that their excitation leads to consistent alterations in behavior that are often dependent upon the desired behavioral state of the animal. Our model predicts that in the presence of an information bottleneck, such a modular structure is needed to increase the efficiency of the network and to make it more robust to perturbations. However, it does so at the cost of an increase in state-dependent effects. Despite its simplicity, our model is able to provide intuition for the trade-offs faced by the nervous system in the presence of an information processing constraint and makes predictions for future experiments.

Keywords: neural control, modularity, bottlenecks, neural networks, robustness

1. INTRODUCTION

When presented with dynamical external stimuli, an animal selects a behavior to perform—or a lack thereof—according to its internal drives and its model of the world. Its survival depends on its ability to quickly and accurately select an appropriate action, as well as to transmit information from the brain to its motor circuitry in order to physically perform the behavior. In almost all animals, however, there exists a bottleneck between the number of neurons in the brain that make cognitive decisions and the motor units that are responsible for actuating movements, thus constraining the amount of information that can be transmitted from the brain to the body (Smarandache-Wellmann, 2016; Kandel et al., 2021).

In the fruit fly *Drosophila melanogaster*, descending commands from the brain to the ventral nerve cord (VNC) are transmitted through approximately 300 bilaterally symmetric pairs of neurons that have their cell bodies in the brain and have axons project into the VNC (Gronenberg and Strausfeld, 1990; Hsu and Bhandawat, 2016). Recent anatomical studies have shown that these neurons exhibit a modular pattern of connectivity, with the descending neurons clustering into groups that each innervate different parts of the motor system (Namiki et al., 2018; Phelps et al., 2021).

In addition to these anatomical properties, in the fruit fly, manipulating these descending neurons *via* optogenetics has shown that exciting individual neurons or subsets of neurons often result in dramatic and robust behavioral alterations—for example, exciting the DNg07 and DNg08 neurons reliably elicits head grooming, and exciting DNg25 elicits a fast running response (Cande et al., 2018). In many cases, however, it has been shown that exciting the same neuron in different contexts (e.g., walking and flying) often have state-dependent effects (Cande et al., 2018; Zacarias et al., 2018; Ache et al., 2019). In other words, the behavioral effect of stimulating the neuron often depends on the actions that the fly is attempting to perform.

In this study, we use a simplified model of behavioral control to explore how modularity may help increase the efficiency and robustness of behavioral control given an information bottleneck. Specifically, our model predicts that modularity of behavior increases the efficiency of the network and its robustness to perturbations, but also that this modularity increases the amount of state-dependent variability in how behavioral commands are transmitted through the bottleneck. While our feed-forward model is a vast oversimplification of the complicated recurrent circuitry that lives within a fly's ventral nerve cord, we show that it provides intuition into the trade-offs the nervous system is faced with, and makes qualitative predictions as to how the system might respond to inhibition or double-activation experiments.

2. RESULTS AND DISCUSSION

Inspired by the fly ventral nerve cord, we have developed an abstracted model that aims to generate insight into the general problem of behavior control through an information bottleneck. Specifically, we assume that there is a set of N behaviors that are in an animal's behavioral repertoire and that to perform one of these behaviors, the animal must excite a subset of M total binary "motor" neurons (e.g., task 14 requires units 1, 3, and 99 to turn-on, and all the rest to be turned off—see **Figures 1A,B**). However, to model the effect of having limited information transmission from the brain to the motor systems, any commands from the brain must travel through an hidden layer of $R < M, N$ descending neurons (Namiki et al., 2018).

We implemented this model using a feed-forward neural network, with the task being encoded in the top layer, the descending neurons being the hidden layer, and the motor units constituting the bottom layer (see **Figure 1A**). For simplicity, we assume that the brain's intended behavioral output is represented in a one-hot encoded manner, where only one "decision" neuron is turned on at once [i.e., behavior 2 is represented by a first layer of $(0, 1, 0, \cdots) \in \{0, 1\}^N$]. We start with the case where each behavior is randomly assigned a set of k motor neurons that must be activated. **Figure 1B** shows an example of this desired mapping, which we call our behavioral matrix. To perform a behavior, one of the decision neurons has to be activated and pass its signal through the network. The parameters of the network, weights $\{W^{(1)}_{\alpha,\beta}, W^{(2)}_{\alpha,\beta}\}$ and biases $\{B^{(1)}_{\beta}, B^{(2)}_{\beta}\}$, are trained to perform the mapping between the top and bottom layers as accurately as possible (see details in section 4).

Given this model, we would like to study how the network performs as a function of the bottleneck size and the sparsity of the behavioral matrix. The absolute maximum number of sequences that the network could encode is 2^R as each hidden neuron can either be activated or not. However, this simple neural network is incapable of reaching the ideal limit. In **Figure 1C**, the bottleneck size required for accurate encoding is $\sim 20 - 60$ for $N = M = 100$, depending on the sparsity of the behavioral matrix. These values are much larger than the minimal possible bottleneck size, $R = \log_2 100 \approx 7$. While we will explore the potential reasons for this discrepancy shortly, we empirically define the critical bottleneck size, R_c, as the minimal number of neurons in the hidden layer sufficient to reproduce 98% of the behaviors correctly, averaged across multiple random instantiations of the behavioral matrix. See **Supplementary Figure 1** for example learning and loss curves, and **Supplementary Figure 2** for example values of the hidden layer and the weights of the trained network. The values of the hidden layer get more binarized (**Supplementary Figures 2a,b**) as its size decreases, implying that the system is getting pushed out of its dynamic range.

2.1. Characterization of the Model

To explore how the statistics of the behavioral matrix affect the critical bottleneck size, we altered the sparsity of the outputs by manipulating the number of motor neurons activated per behavior (k) while keeping $M = N = 100$ (**Figure 1C**). Note that since our output size is 100 and its encoding is binary, a neural network with k and $100 - k$ activated motor neurons have the same statistical behavior. Thus, sparsity increases as k deviates from 50 in either direction. As evident from **Figure 1C** and the inset therein, as k decreases below 25, the network requires fewer neurons in the hidden layer (a lower R_c) to learn all of the behaviors perfectly, with the decrease starting around $k = 25$. Ultimately, for the sparsest output encoding we tested ($k = 5$), the network requires half the number of neurons compared to the densest ($k = 50$) case ($R_c \approx 24.4 \pm 0.8$ vs. $R_c \approx 57 \pm 2$), indicating that it is more difficult for our model to learn the more complicated patterns that are associated with a denser output. This effect can be more explicitly seen by plotting R_c as a function of the entropy of the behavioral matrix (**Figure 1D**, Equation 4). Furthermore, we note that the shape of the curve, as a function of hidden layer size, R, approaches that of a sigmoid function in the limit of dense output signal (as k approaches 50). Equivalently, sparsity can be varied by fixing k and varying the size of the output layer M (here, keeping $N = 100$ fixed) (**Supplementary Figure 3**). We again find that as the output signal becomes more sparse, that is, as M increases, it is easier to learn the mapping from behavior to motor commands. Moreover, we also notice that the learning curves split into two

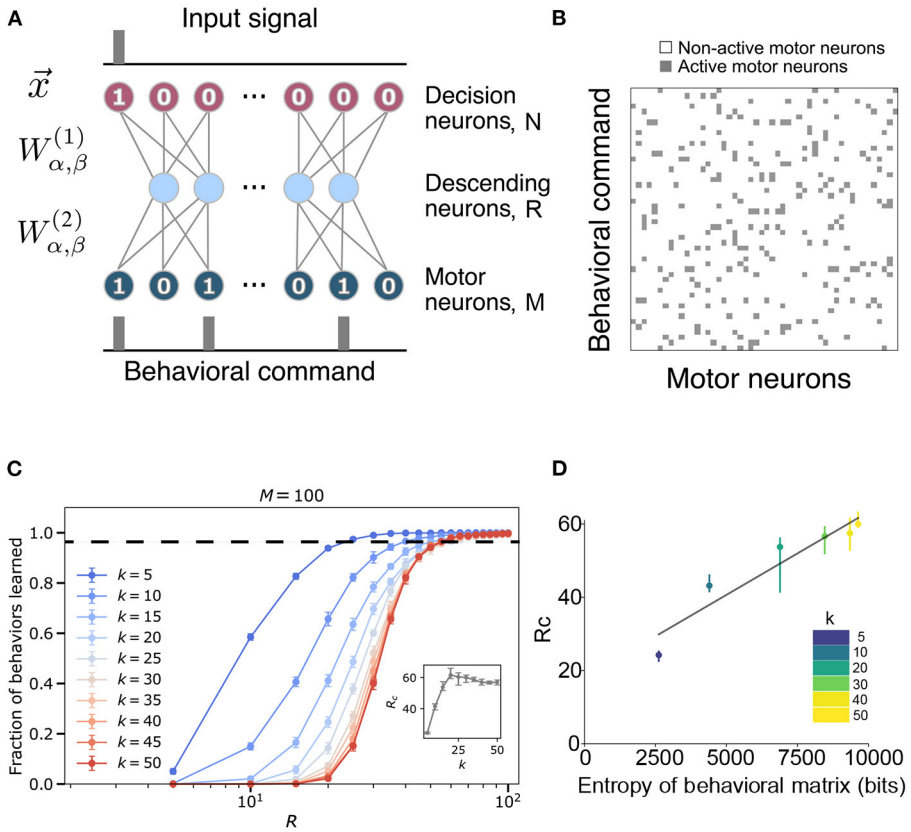

FIGURE 1 | Model construction and parameters. **(A)** The structure of the ventral nerve chord is modeled by a neural network that takes as input a task assignment represented by a binary sequence $\vec{x}$ of length N. The signal travels through a hidden layer (size R) to an output layer (size M), which corresponds to descending neurons and motor neurons, respectively. Each neuron in one layer communicates with all the neurons in the following layer through the weight matrices $W^{(\cdot)}_{\alpha,\beta}$, detailed in section 4. **(B)** An example of a behavioral matrix that indicates the motor units activated for each task. Row i corresponds to the i-th behavioral command (i.e., the i-th neuron activated in the input layer of the network). k is the number of motor neurons needed to execute a given behavior. Columns correspond to different motor neurons [i.e., the jth column indicates whether a particular motor neuron was active (gray) or not (white) in the behaviors]. **(C)** Fraction of behaviors learned as a function of hidden layer size R and fixed input layer size $N = 100$ for varying k and fixed output layer size $M = 100$. The inset shows the critical bottleneck size R_c as a function of k. Each point is averaged over 30 random input-output combinations. Dashed line indicates critical bottleneck threshold. **(D)** Values of the critical bottleneck size R_c for different values of sparsity ($k = 5, 10, 20, 30, 40, 50$) as a function of the behaviorial matrix entropy. Black line is the line of best fit and is provided for visual aid only.

regimes (**Supplementary Figures 3a,b**) corresponding to when M is smaller or larger than N. When $M > N$, the network finds it much easier to learn with the learning ability saturating when the bottleneck size is a certain fraction of the output layer.

2.2. Modularity of Behaviors

While the analyses presented in the previous section involved random mappings between behaviors and motor outputs, we now ask if imposing biologically inspired constraints on this mapping might affect the efficiency of the network. Specifically, we will assume that the behavioral matrix is modular, with similar behaviors (e.g., different locomotion gaits or different types of anterior grooming motions) more likely to require similar motor output patterns. This constraint is motivated from previous anatomical studies in *Drosophila* (Namiki et al., 2018).

To explore the effect of modular structure on our model, we performed a set of simulations with various degrees of behavioral matrix modularity. Specifically, we fixed $k = 10$ and split the behavioral matrix into 5 regions (see inset in **Figure 2A**). If there is no active motor neuron in common between the different clusters, then we have perfect modularity [$\mu = 0.8$, where μ is the fraction of the edges that fall within the modules minus the expected fraction within the modules for an equivalent random network (Newman, 2018), see section 4]. We then allowed for some overlap between regions to generate matrices with a spectrum of modularities (some examples given in **Figure 3C**) between the perfect modular limit and random mixing. We observed that the modular behavioral matrices can be learned more efficiently than random matrices, requiring far smaller critical bottleneck sizes to achieve the correct mapping of behavioral commands (**Figure 2A**). The perfectly modular output matrix (inset **Figure 2A**) was learned

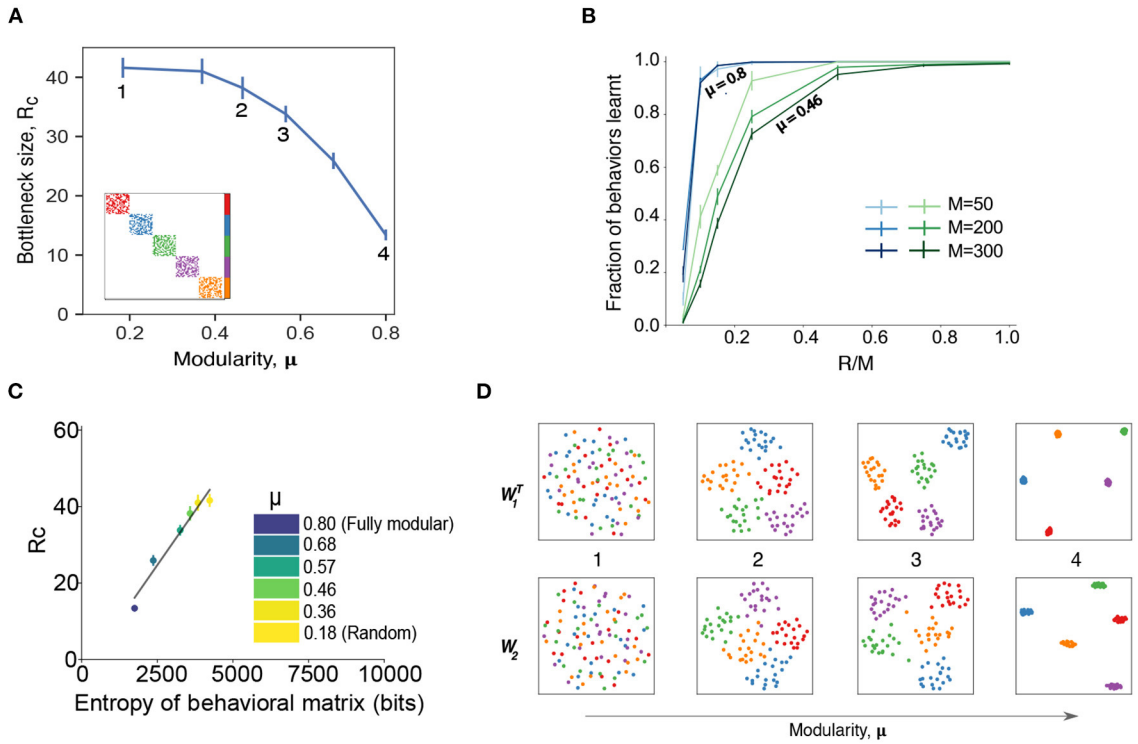

FIGURE 2 | Modularity in the behavioral commands reduces critical bottleneck size and affects other network properties. **(A)** Relationship between the size of the hidden layer R and the modularity of the behavior matrix. Each point corresponds to a set of numerical experiments with 10 different matrices around a given modularity value (see section 4 for details of data generation) for $k = 10$, $M = N = 100$. R_c is defined as the minimal hidden layer size that was able to achieve 98% accuracy in 10^5 epochs of training. Numbers indicate specific cases that are shown in panels **(B–D)** in more detail. Inset shows an example of behavioral command matrix for $\mu = 0.8$ case (point 4). **(B)** Fraction of behaviors learned as a function of the hidden layer size, R for different system sizes with $N = M$ for two levels of modularity ($\mu = 0.8$ and $\mu = 0.46$). Error bars correspond to the standard deviation. Results are averaged over 5 different runs with error bars corresponding to the standard deviation. **(C)** Values of the critical bottleneck size R_c for different values of modularity [$\mu = 0.8$ (fully modular), 0.68, 0.57, 0.46, 0.36, 0.18] as a function of the behavioral matrix entropy. Black line is the line of best fit and is provided for visual aid only. **(D)** Structure of the weight matrices W_1 and W_2 for different modularity values. The dimensionality reduction is performed *via* UMAP (McInnes et al., 2018), a non-linear method that preserves local structure in the data. The point colors correspond to the colors in **(A)** inset: 1 ($\mu = 0.18$, random matrix); 2 ($\mu = 0.46$); 3 ($\mu = 0.56$); 4 ($\mu = 0.8$, perfectly modular matrix with 5 clusters).

with only $R_c = 13$ neurons, which is less than half the number required for the random matrix ($R_c \simeq 35$) with the same amount of sparsity (**Supplementary Table 1**). Note that the dependence of the critical bottleneck layer size on matrix modularity is not linear, just 2 neurons overlapping between clusters makes learning much harder ($R_c = 30$, point #3 in **Figure 2A**).

In addition to making the mapping easier to learn, modularity in the behavioral matrix also helps learning scale with the system size. In **Figure 2B**, we plot the fraction of behaviors learned as a function of the relative size of the bottleneck layer R as compared to the output layer M, for different values of the system size (we assume $N = M$) and for different values of the modularity. Modularity values were chosen to highlight the differences between a perfectly modular matrix ($\mu = 0.8$) and a matrix that has a low amount of modularity ($\mu = 0.46$) while not being completely random. For highly modular behavioral matrices (blue curves in **Figure 2B**), we find that the size of the output doesn't affect the learning ability of the network, as the bottleneck occurs when the size of the hidden layer is a similar fraction of the output sizes. On the other hand, when the behavioral commands aren't very modular, smaller system sizes learn better for a relatively smaller bottleneck size (green curves in **Figure 2B**). This is again a reflection of our model finding it easier to learn the simpler patterns (less entropy) of a more modular behavioral matrix (**Figure 2C**). The similarities between **Figures 1D**, **2C** indicate that the entropy of the behavioral matrix is an important parameter that determines R_c, even while keeping other parameters constant.

Finally, we found that imposing a modular output structure also imposes a modular structure on the weights of the learned network (**Figure 2D**). The modularity in the weights becomes more pronounced as the modularity of the behavioral matrix increases, similar to results found in the study of more generalized artificial neural networks (Zavatone-Veth et al., 2021). Together, these results show that modularity in the behavioral matrix increases the efficiency and scaling properties of the network through creating a concomitantly modular representation within the model.

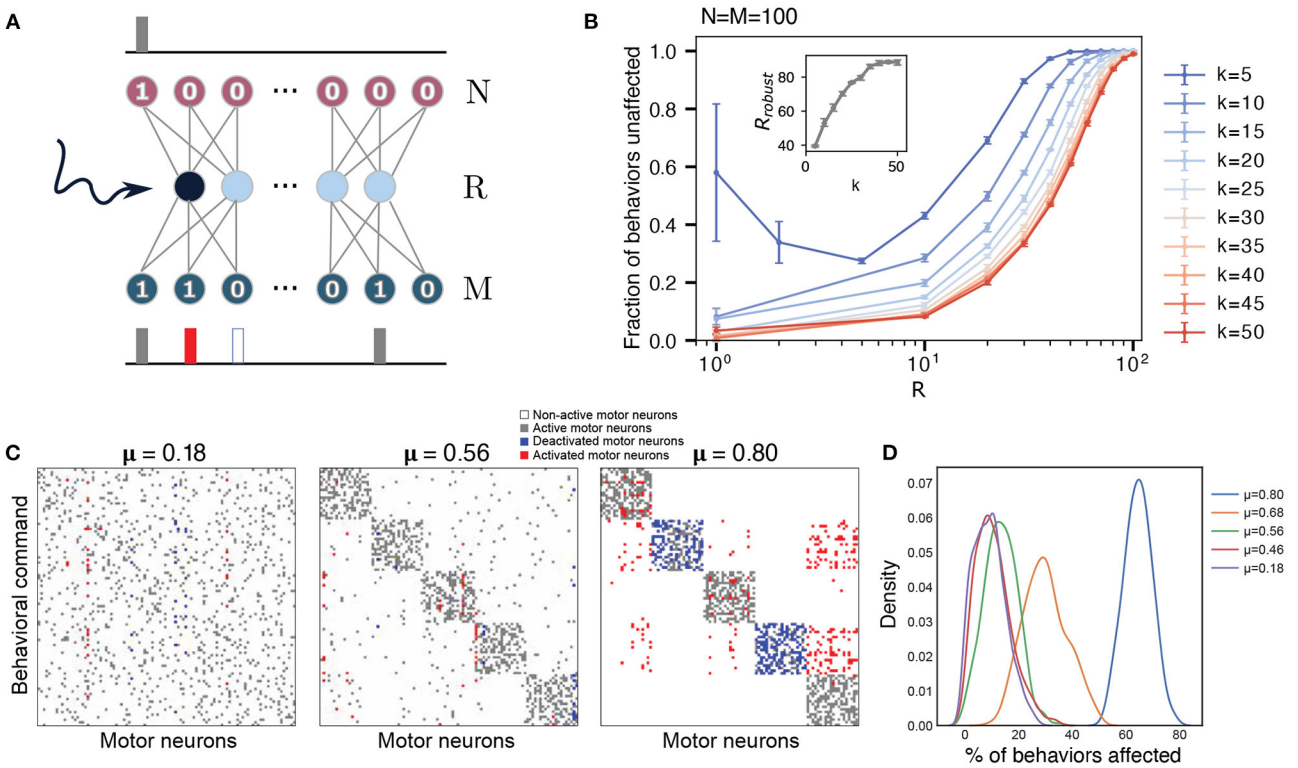

FIGURE 3 | Robustness of the network to perturbations increases with the size of the hidden layer and sparsity. **(A)** Schematic of the perturbation experiment. One of the hidden neurons of the trained network is artificially forced to be on, keeping all other network parameters unchanged. The network is re-run to generate new outputs for each behavioral command. **(B)** Robustness (fraction of outputs that are unaffected by the perturbation) averaged over the effects of activating each hidden neuron as function of the hidden layer size R, with $N = M = 100$, with k varied from $k = 5$ to $k = 50$. The error bars are obtained by considering 10 different behavioral matrices. The inset shows the size of the hidden layer for which such a perturbation leaves 98% of the behaviors unaffected, R_{robust}, as a function of changing sparsity (varying k). **(C)** Example of a hidden layer perturbation on the trained networks' behavior matrices with different modularities (all show with $R = R_c$). In each case, one of the hidden neurons is kept constantly activated, while the rest of the network operates according to the trained weights. White and gray colors correspond to unperturbed motor neurons, non-active and active correspondingly. Blue indicates motor units that have been turned off, and red shows motor units that have been activated. **(D)** Distribution of the number of behavioral commands affected by the hidden layer perturbation. Colors correspond to different degrees of modularity μ. Each distribution was calculated based on 10 different behavioral matrices, all with $R = R_c$.

2.3. Robustness to Perturbations of the Bottleneck Layer

Although the network is capable of reproducing behavioral commands nearly perfectly when it is near the critical bottleneck, it might be prone to errors due to minor perturbations, including noise in the firing of the descending layer. Inspired by previous studies in flies where descending neurons were artificially activated (Cande et al., 2018; Ache et al., 2019), we investigate the robustness of our trained neural networks by manually activating one hidden neuron at a time. We then observe the changes in the output (see **Figure 3A**) to see how these activations affect the mapping between command and behavior. An example of possible outcomes on a set of behaviors under these perturbation is shown in **Figure 3C** (for more examples, see **Supplementary Figure 4**). For each behavioral command, the motor neurons can either remain unaffected—their original "active" or "non-active" state is maintained (gray and white pixels in **Figure 3C**) or their state gets flipped—an "active" neuron gets inactivated or vice-versa (red and blue pixels in **Figure 3C**). The robustness of the network with respect to the activated neuron is calculated as the number of behaviors that are conserved, that is, behavioral commands where all activated motor neurons remain unaffected.

Figure 3B shows the robustness of the network to these perturbations as a function of the hidden layer size R and varying sparsity ($N = M = 100$ is fixed and k is varied), averaged over the effects of activating each hidden neuron and each behavioral command for a randomly generated behavioral matrix (no enforced modularity). For fixed sparsity, the fraction of behaviors that are unaffected increases as the size of the hidden layer increases. At the critical bottleneck size, for example, $R_c = 35$ for $k = 10$, 80% of behaviors were unaffected by the perturbation, indicating that the neural network has some margin of robustness. Robustness increases as we increase the hidden layer size R—the behavioral commands become less sensitive to changes in each individual hidden neuron. As long as the bottleneck layer size is less than the output layer ($R < M$), networks with output signals of high sparsity (lower k) are more

robust on average. The robustness is bounded below by the curve corresponding to maximum output signal density $k = 50 = M/2$. For sufficiently dense output signals $50 \geq k > 5$, the robustness decreases monotonically with decreasing hidden layer size for the entire range of $1 \leq R \leq M$. In contrast, the robustness of high sparsity outputs ($k = 5$) decreases initially with decreasing hidden layer size, but exhibits an increase in both its mean and variance at very small hidden layer sizes ($R < 5$). This behavior is likely caused by an all-or-nothing switching relationship between the hidden neurons and the output neurons.

When applying these perturbations to more modular behavioral matrices (**Figure 3C**), we find that the effects of the activations to the hidden neurons lead to more correlated changes in motor outputs. For these cases at the bottleneck size R_c (which varies depending upon the modularity, see **Figure 2A**), when some of the hidden neurons are activated, they not only affect a certain number of behaviors, but all of these commands tend to belong to the same cluster, which is what we would expect, given the modular structure of the weights in **Figure 2D**. Moreover, activation of a neuron can lead to the complete switch from one type of behavior to the another. An example of this effect is shown in **Figure 3C**. The first matrix in this panel corresponds to a random matrix of behavioral commands (also point #1 in **Figure 2A**). In this case, a particular hidden neuron may be attributed to at most some set of motor neurons as its activation leads to activation of two of them and deactivation of other three. However, in the perfectly modular case, there are some neurons that are responsible for the encoding of the whole cluster (rightmost panel in **Figure 3C**). When a hidden neuron is activated, it causes nearly an entire module of behaviors to be altered. This is in keeping with the previous studies showing that stimulating individual descending neurons in flies can result in dramatic behavioral effects (Bidaye et al., 2014; Cande et al., 2018; Ding et al., 2019; McKellar et al., 2019). Averaging over several behavioral matrices and perturbations (**Figure 3D**), we observe that this pattern holds true in general, with more modular behavioral matrices affected more by perturbations at R_c. This effect is likely due to the different sizes of the hidden layer where the critical bottleneck size R_c (the minimum number of hidden layer neurons needed to ably represent all behavioral commands) occurs, for varying levels of modularity. As the size of the hidden layer controls the susceptibility toward perturbations (**Figure 3B**), highly modular behavioral matrices that have a much smaller R_c (**Figure 2A**), are affected to a larger extent by the perturbations. For example, a fully modular behavioral matrix has $R_c = 13$, but at this size of the hidden layer, it is only approximately 40% robust to such perturbations (**Figure 4A**). This example highlights a trade-off between efficient information compression in the bottleneck layer and robustness in case of failure. In general however, if the constraint is that the size of the hidden layer is fixed, modularity *increases* robustness to perturbations (**Figure 4A**).

Thus, when constrained by a fixed size of the hidden layer, increasing the modularity and sparsity of the behavioral commands helps increase the robustness of the network to artificial perturbations. However, robustness suffers if the goal is to operate the network at the smallest possible critical bottleneck size for a given number of behavioral commands.

2.4. State-Dependency of Behaviors

Previous experimental studies in fruit flies observed that optogenetically activated behaviors in flies often depend on their behavioral state prior to activation (Cande et al., 2018; Ache et al., 2019). This effect can be quantified by calculating the mutual information between the distribution of a fly's behaviors before and after artificial neural activation. We refer to this effect as *state-dependency*. In essence, state-dependency implies that stimulating a neuron in the bottleneck layer will have varying—but predictable based on the input—behavioral results. In order to understand this experimentally observed effect within the framework of our model, we calculated the mutual information between the input and output distributions in the presence of an activated hidden neuron, while varying the size of the hidden layer and modularity (**Figure 4B** and **Supplementary Material 5**, see section 4 for details). This calculation provides a measure of how much information about the input distribution is contained in the output distribution in the presence of artificial activation.

With the input distribution corresponding to the fly's intended behavioral output (the one-hot encoded initial layer from **Figure 1A**) and the modified output corresponding to the set of behaviors that the artificial activation triggers, we see that increasing the bottleneck constraint (reducing R) lowers the overall mutual information—thus, it becomes harder to predict what the triggered behavior will be. On the other hand, a higher amount of modular structure in the output behavioral commands increases the mutual information for a fixed size of the hidden layer, with a maximum increase of around 0.8 corresponding to about a 30% increase between the two extreme values of modularity ($\mu = 0.18$ and $\mu = 0.8$) considered here. Thus, our model predicts that increased modular structure in the behavioral matrix not only increased robustness to perturbations (for a given N, M, and R), but also results in increased state-dependency. These results are consistent with the finding of state-dependency and modularity in the *Drosophila* VNC. In our model, this effect likely results from the fact that the model's weights are segregated at higher modularities (**Figure 2D**), meaning that the effect of stimulating a given bottleneck-layer neuron will be limited to a relatively small number of output behaviors.

It is worth mentioning that we find that the mutual information is proportional to the robustness (**Figure 4A** and **Supplementary Figure 5**) with a proportionality constant $\frac{1}{M}\log_2(N)$ (see section 4). This is a consequence of an absence of stereotypy in our simplified model, that is, multiple inputs don't give the same output on forced activation.

Given these results, we explored what predictions our model makes for two additional types of perturbation experiments that have not, to our knowledge, been systematically performed. First, we asked what the effects would be for deactivating, rather than activating, individual hidden layer neurons (**Figure 5A**). As one might expect for a binary encoded network, the effect of deactivating individual neurons on the robustness of the network is qualitatively similar to that for activation. The network is more robust to the perturbation as the size

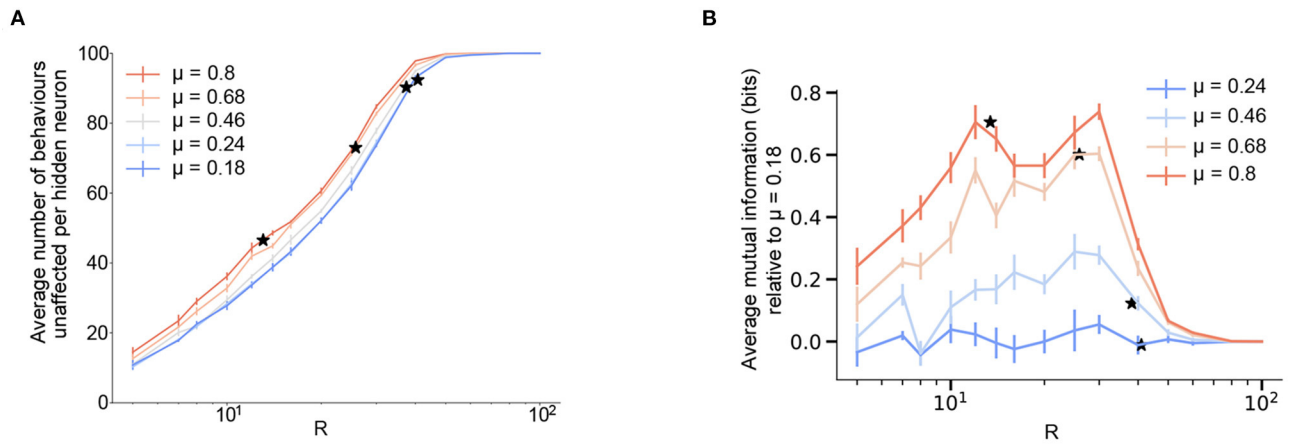

FIGURE 4 | Modularity improves robustness to perturbation and increases state-dependency for a fixed size of the hidden layer. **(A)** Robustness of the network averaged over the effects of activating each hidden neuron as a function of the hidden layer size, R and varying levels of modularity, μ. Here, robustness $\mathcal{A}(\mu)$ is defined as the numbers of behaviors that are not affected upon forcefully activating a neuron in the network. **(B)** Average mutual information (defined as in Equation 9) between the input and output distributions after forced activation of each hidden neuron as a function of the size of the hidden layer R, and varying levels of modularity, μ. To highlight the effects of increasing modularity, we show the results relative to the lowest modularity. The figure for the absolute values is reported in **Supplementary Figure 5**. The mutual information turns out to be $\mathcal{A}(\mu) \times \frac{1}{M}\log_2(N)$ due to the absence of stereotypy. **(A,B)** $N = M = 100$ and results are means over 5 iterations with the error bars corresponding to the standard deviation. Stars correspond to the R_c value for each value of modularity.

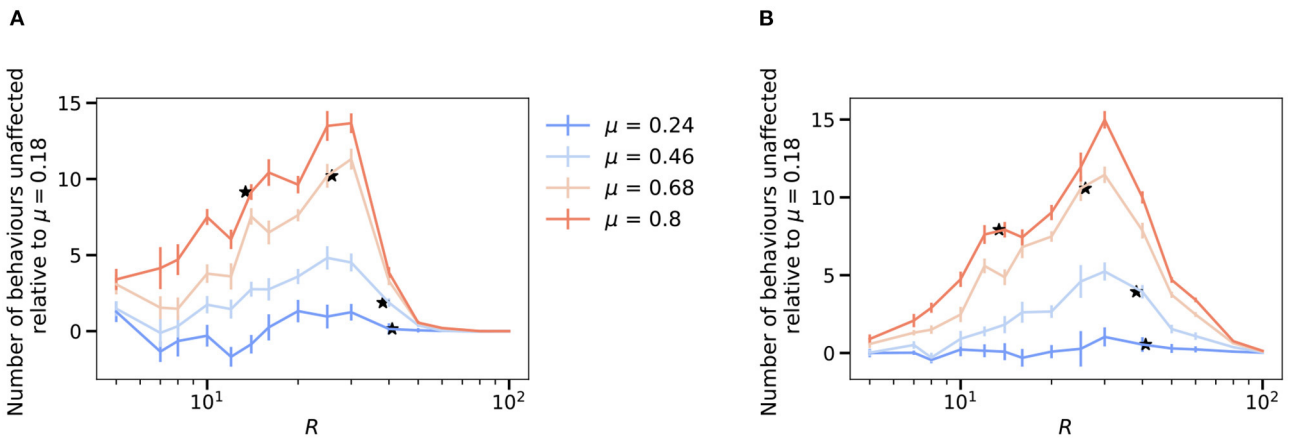

FIGURE 5 | Future excitation and inhibition experiments predict modularity is always associated with improved robustness. To highlight the effects of increasing modularity, we show the results relative to the lowest modularity, $\mu = 0.18$ as $\mathcal{A}(\mu) - \mathcal{A}(\mu = 0.18)$, where $\mathcal{A}$ is the robustness of the network upon de-activating each hidden neuron **(A)** and the robustness upon activating pairs of hidden neurons one at a time **(B)** defined as follows. The figure for the absolute values is reported in **Supplementary Figure 6**. The value of R_c for each modularity value is shown as stars. **(A)** Robustness of the network averaged over the effects of de-activating each hidden neuron as a function of the hidden layer size, R and varying levels of modularity, μ. **(B)** Robustness of the network averaged over the effects of activating a pair of hidden neurons as a function of the hidden layer size, R and varying levels of modularity, μ. **(A,B)** $N = M = 100$ and results are means over 5 iterations with the error bars corresponding to the standard deviation.

of the hidden layer increases. For any given size of the hidden layer, modularity increases the network's robustness to deactivating perturbations.

Similarly, we also explored whether activating pairs of hidden layer neurons (rather than individual neurons) leads to increased state-dependency with modularity as well (**Figure 5B**). We find similar results in this case (averaging over all possible pairs of hidden layer units across many networks).

3. CONCLUSION

Understanding how animals use their nervous system to control behavior is one of the key questions in neuroscience. A key component of most animal's nervous system is an information bottleneck between cognitive decision-making in the brain and the neurons that are responsible for the performance of behaviors. In this work, we use a simple feed-forward neural network, similar to an autoencoder architecture that is commonly

used in deep neural networks (Goodfellow et al., 2016), to understand the consequences of having such a bottleneck and identify different aspects of the network architecture that can still enable robust learning despite having such a constraint. For each set of network parameters, we identify the smallest size of the hidden layer (bottleneck size) that still allows near perfect learning. We find that increasing the sparsity of the output behavioral commands reduces this bottleneck size and increases the robustness of the network.

In addition to sparsity, we find that an increased modularity in the behavioral commands helps to reduce the bottleneck size and increases robustness. This observation could provide an explanation for why such a modular structure has evolved in the behavioral commands in animals, so far observed in flies. Our simple model is also able to predict the experimentally observed state-dependency between behavioral states before and after the forced activation of hidden neurons. We find that lowering the size of the hidden layer reduces state-dependent variability, but state-dependency increases with increasing modularity for a fixed hidden layer size. Overall, the modular nature of the output makes it easier for the network to learn in the presence of a bottleneck, increases its robustness but also leads to a higher amount of state-dependency.

This model described here is obviously simplistic in architecture and dynamics (in that it lacks them) and is highly unlikely to accurately describe the dynamical activity of ventral nerve cord function, where recurrent connections and temporal structure are important features of the system's functioning (Reyn et al., 2014; Phelps et al., 2021). Future work would incorporate the effects of temporal dynamics, as well as using more biophysically realistic neurons. In addition, our model only includes discrete inputs, and understanding how graded controls over more continuous variables (e.g., walking or flight speed) would be interesting for future study. In addition, our interpretation of the results implicitly assumes that the information bottleneck is the fundamental constraint that evolution has to contend with, rather than modularity itself being the constraint and an information bottleneck being the answer that maximizes efficiency. While the ubiquity of information bottlenecks in most nervous systems provides indirect evidence toward our interpretation, future comparative studies will be needed to assess which of the two hypotheses is more likely.

However, despite its simplicity, our model recapitulates several non-trivial features that are observed in experiment, and makes predictions as to the effects of artificially inhibiting neurons or of simultaneously stimulating multiple neurons, allowing for general principles of information-limited motor control to be elucidated, and new hypotheses to be tested.

4. MATERIALS AND METHODS

4.1. Network Architecture and Training

To mimic the structure of the neural chord, we built a feed-forward fully-connected neural network with one hidden layer (see **Figure 1**). The network is constructed with the Python framework PyTorch. The input layer represents decision neurons of number N: they send the signal from the brain down the network leading to a certain behavioral output. The hidden layer of size R represents descending neurons of the neural chord: it transmits the signal down to the motor neurons, which are the output layer of the network of size M. We used the sigmoid as our activation function, serving as an approximation of the transmission of the neural signal. The functioning of the neural network can be understood explicitly from its mathematical definition. The first layer applies a linear transformation on the input sequence $\vec{x}$ *via* the weight matrix, $W^{(1)}_{\alpha,\beta}$ connecting neuron α in the first layer with neuron β in the following equation,

$$a^{(1)}_{\beta} = \sum_{\alpha} W^{(1)}_{\alpha,\beta} x_{\alpha} - B^{(1)}_{\beta}\,, \tag{1}$$

while the second and last layer applies the activation function $\rho(\mathbf{a})$ on $\mathbf{a}^{(1)}$ as,

$$a^{(2)}_{\beta} = \sum_{\alpha} W^{(2)}_{\alpha,\beta} \rho(a^{(1)}_{\alpha}) - B^{(2)}_{\beta}\,, \tag{2}$$

with $\rho(\mathbf{a})$ given by the sigmoid $\rho(x) = 1/(1+e^{-x})$ and $\mathbf{B}^{(1)}$ ($\mathbf{B}^{(2)}$) is the bias, an additive constant. The output of the network is defined as $f(\mathbf{x}, \mathbf{W}) \equiv \mathbf{a}^{(2)}$, where $\mathbf{W}$ contains all the parameters, comprising the biases.

We fixed the size of the input layer ($N = 100$) throughout our experiments, while varying the sizes R, M of the hidden and output layers. We trained the network in the following fashion: we fixed the input and output matrices, i.e., decision and behavior matrices, respectively; we trained the network in a feed-forward manner using stochastic gradient descent with momentum and used the mean-squared error (MSE) loss function to assess learning performance; we stopped training after 10^5 epochs, which corresponds to when the loss curve flattens and the network is no longer learning. The output $\mathbf{y} = f(\mathbf{x}, \mathbf{W})$ of the trained network is then binarized by rounding each entry (using a Heavyside step function centered around 0.5) and the trained weights and biases defining the network are saved for further analysis. Along with these parameters, the number of behaviors learnt, obtained by comparing each entry of the output $\mathbf{y}$ with the imposed behavior, is also stored.

4.2. Modularity

We use the NetworkX 2.5 Python package to calculate modularity using the function 'networkx.algorithms.community.modularity' by treating the output matrix of behavioral commands as an adjacency matrix of a graph. Here modularity is defined as Newman (2018),

$$\mu = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j) \tag{3}$$

where m is the number of edges, A_{ij} is the adjacency matrix, k_i is the degree (number of connections of a node in a graph) of i and $\delta(c_i, c_j)$ is 1 if i and j are in the same community and 0 otherwise.

4.3. Entropy of the Behavioral Matrix

The entropy of the behavioral matrix depends upon the number of behaviors N, size of the output layer M, sparsity k, number

of modules m, and the noise σ associated with the modules (# of units active outside a module, for e.g., $\sigma = 0$ for perfect modularity). For a random behavioral matrix where for any output k random units are turned "on" the total entropy (in bits) is,

$$S = N\log_2\binom{M}{k} \tag{4}$$

For a modular behavioral matrix with equal sized square modules ($m_{\text{size}} \times m_{\text{size}}$, $m_{\text{size}} = M/m$) the entropy (in bits) is given by,

$$S = N\log_2\left[\binom{m_{\text{size}}}{k-\sigma} \times \binom{M - m_{\text{size}}}{\sigma}\right] \tag{5}$$

4.4. Data Generation

The input data for all of our numerical experiments is always a 100×100 identity matrix. Each row of this matrix corresponds to the signal of performing one behavior from the output matrix. We generated several sets of output behavior matrices. In **Figure 1**, we varied the sparsity of the output matrix by changing the number of randomly activated units in a given row, i.e., the number of 1s. In **Figure 3**, we generated modular behavior matrices by introducing dense and sparse clusters into the output matrix. We start with 5 perfect clusters, i.e., no activated units are in common between 2 different clusters. Then, we generate matrices with different degree of modularity by deactivating some of the units within the cluster and activating the same number of units outside of the cluster so that the sparsity is preserved. In each case we generated 10 different behavior matrices for statistical purposes.

4.5. Checking the Robustness of the Network

We checked the robustness of the network by forcefully activating one of the hidden layer neurons. This is achieved by setting its corresponding weight in the first weight matrix $\mathbf{W}^{(1)}$ to an arbitrarily high value. We propagate the input matrix through the resulting perturbed network to get an output behavior matrix to be compared to the original output. In this way we can monitor how many of the original output behaviors were changed by the forceful activation. These steps are repeated for each individual hidden neuron and the results are averaged over the number of hidden neurons.

4.6. Mutual Information Calculation

Mutual information (MI) between two distributions is the measure of the amount of information one distribution has about the other. For two discrete binary random variables X and Y embedded in $\mathbb{R}^N$ with joint distribution $P(X, Y)$ it is given by Cover and Thomas (2006),

$$I(X;\, Y) = \sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} P(x,y)\log_2\frac{P(x,y)}{P(x)P(y)} \tag{6}$$

where $P(X)$ and $P(Y)$ are the marginal distributions. In the absence of forced activation, the perfect learning case has a one to one mapping between the input and output distributions and hence the MI is $\log_2 N$. This perfect mapping gets perturbed on forced activation which can lead to one of the three different scenarios: (i) the input-output mapping is still unaffected, (ii) the input gets mapped to another output (stereotypy), and (iii) the input gets mapped to a completely different output that is not part of the original output distribution. This last case suggests that the input possess no information about the output.

Suppose we have N inputs x and M outputs y where we assume that they follow a uniform distribution, that is, $P(x) = 1/N$ and $P(y) = 1/M$. After forced activation, let n_i be the number of inputs associated with each output y_i where $n_i \geq 0$. This gives us $P(x|y_i) = \frac{1}{n_i}$ when $n_i > 0$ and $P(x|y_i) = 0$ when $n_i = 0$. The mutual information then reads

$$I(X, Y) = \sum_{y\in\mathcal{Y}} P(y) \sum_{x\in\mathcal{X}} P(x|y)\log_2\frac{P(x|y)}{P(x)} \tag{7}$$

$$= \sum_{y_i\in\mathcal{Y}'} P(y_i) \sum_{x\in\mathcal{X}'} \frac{1}{n_i}\log_2\left(\frac{N}{n_i}\right) \tag{8}$$

$$= \sum_{y_i\in\mathcal{Y}'} P(y_i)\log_2\left(\frac{N}{n_i}\right) \tag{9}$$

$$= \frac{1}{M}\sum_{y_i\in\mathcal{Y}'} \log_2\left(\frac{N}{n_i}\right) \tag{10}$$

where $\mathcal{X}'$ is the set of n_i inputs associated with each output y_i, $\mathcal{Y}'$ is the set of m outputs with $n_i > 0$. Note that in the absence of stereotypy that is, when n_i is either 1 or 0, the mutual information becomes

$$I(X, Y) = \frac{m}{M}\log_2(N), \tag{11}$$

where m is the number of original outputs that were unaffected by perturbation and hence, the mutual information becomes proportional to our definition of network robustness.

4.7. Statistical Analysis

Error bars in the figures are standard deviations that were calculated by averaging simulation results for 10 different output matrices unless specified otherwise. We used the UMAP (McInnes et al., 2018) method to visualize the structure in weight matrices.

4.8. Code Availability

The code for both our simulations and statistical analysis, can be downloaded from: https://github.com/drahcir7/bottleneck-behaviors.

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/**Supplementary Material**, further inquiries can be directed to the corresponding author/s.

AUTHOR CONTRIBUTIONS

AN, VD, RR, and GZ performed all the analyses. GB conceived the project and advised on all aspects of the modeling and analysis. All authors wrote the manuscript. All authors contributed to the article and approved the submitted version.

FUNDING

GB was supported by the Simons Foundation and a Cottrell Scholar Award, a program of the Research Corporation for Science Advancement (25999). AN was supported by a grant from the US National Institutes of Health (DP5OD019851). GZ acknowledges support from the Paul and Daisy Soros Fellowship and the National Science Foundation Graduate Research Fellowship under Grant No. DGE1745303. RR was supported by the Swiss National Science Foundation under grant No. 200021-165509/1.

ACKNOWLEDGMENTS

The authors thank the organizers of the 2019 Boulder Summer School for Condensed Matter and Materials Physics for the opportunity to meet and start a collaboration on this project.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fnbeh.2022.835753/full#supplementary-material

REFERENCES

Ache, J. M., Namiki, S., Lee, A., Branson, K., and Card, G. M. (2019). State-dependent decoupling of sensory and motor circuits underlies behavioral flexibility in Drosophila. *Nat. Neurosci.* 22, 1132–1139. doi: 10.1038/s41593-019-0413-4

Bidaye, S. S., Machacek, C., Wu, Y., and Dickson, B. J. (2014). Neuronal control of *Drosophila* walking direction. *Science* 344, 97–101. doi: 10.1126/science.1249964

Cande, J., Namiki, S., Qiu, J., Korff, W., Card, G. M., Shaevitz, J. W., et al. (2018). Optogenetic dissection of descending behavioral control in Drosophila. *eLife* 7, e34275. doi: 10.7554/eLife.34275

Cover, T. M., and Thomas, J. A. (2006). *Elements of Information Theory, 2nd Edn*. New York, NY: Wiley.

Ding, Y., Lillvis, J. L., Cande, J., Berman, G. J., Arthur, B. J., Long, X., et al. (2019). Neural evolution of context-dependent fly song. *Curr. Biol.* 29, 1089.e7–1099.e7. doi: 10.1016/j.cub.2019.02.019

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. Cambridge, MA: MIT Press.

Gronenberg, W., and Strausfeld, N. J. (1990). Descending neurons supplying the neck and flight motor of diptera: physiological and anatomical characteristics. *J. Comp. Neurol.* 302, 973–991. doi: 10.1002/cne.903020420

Hsu, C. T., and Bhandawat, V. (2016). Organization of descending neurons in *Drosophila melanogaster*. *Sci. Rep.* 6, 20259. doi: 10.1038/srep20259

Kandel, E. R., Koester, J. D., Mack, S. H., and Siegelbaum, S. A. (2021). *Principles of Neural Science, 6th Edn*. New York, NY: McGraw-Hill.

McInnes, L., Healy, J., Saul, N., and Groberger, L. (2018). UMAP: uniform manifold approximation and projection. *J. Open Sour. Softw.* 3, 861. doi: 10.21105/joss.00861

McKellar, C. E., Lillvis, J. L., Bath, D. E., Fitzgerald, J. E., Cannon, J. G., Simpson, J. H., et al. (2019). Threshold-based ordering of sequential actions during drosophila courtship. *Curr. Biol.* 29, 426.e6–434.e6. doi: 10.1016/j.cub.2018.12.019

Namiki, S., Dickinson, M. H., Wong, A. M., Korff, W., and Card, G. M. (2018). The functional organization of descending sensory-motor pathways in Drosophila. *eLife* 7, e34272. doi: 10.7554/eLife.34272

Newman, M. (2018). *Networks, 2 Edn*. Oxford, UK: Oxford University Press. doi: 10.1093/oso/9780198805090.001.0001

Phelps, J. S., Hildebrand, D. G. C., Graham, B. J., Kuan, A. T., Thomas, L. A., Nguyen, T. M., et al. (2021). Reconstruction of motor control circuits in adult Drosophila using automated transmission electron microscopy. *Cell* 184, 759.e18–774.e18. doi: 10.1016/j.cell.2020.12.013

Reyn, C. R., Breads, P., Peek, M. Y., Zheng, G. Z., Williamson, W. R., et al. (2014). A spike-timing mechanism for action selection. *Nat. Neurosci.* 17, 962–970. doi: 10.1038/nn.3741

Smarandache-Wellmann, C. R. (2016). Arthropod neurons and nervous system. *Curr. Biol.* 26, R960-R965. doi: 10.1016/j.cub.2016.07.063

Zacarias, R., Namiki, S., Card, G. M., Vasconcelos, M. L., and Moita, M. A. (2018). Speed dependent descending control of freezing behavior in *Drosophila melanogaster*. *Nat. Commun.* 9, 3697. doi: 10.1038/s41467-018-05875-1

Zavatone-Veth, J. A., Canatar, A., and Pehlevan, C. (2021). Asymptotics of representation learning in finite Bayesian neural networks. *arXiv[preprint].arXiv:2106.00651*. doi: 10.48550/arXiv.2106.00651

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.