Visual systems and the forces that shape them

Thesis by
Mason McGill

In partial fulfillment of the requirements for the degree of
Doctor of Philosophy in computation and neural systems

CALIFORNIA INSTITUTE OF TECHNOLOGY
Pasadena, California

2025
Defended March 25, 2025

Mason McGill
ORCID: 0000-0002-2782-3977

ABSTRACT

Vision neuroscience provides a unique opportunity to draw a correspondance between the physical world and its neural representation. But despite the amazing advances in neural recording technology that have occurred over the past two decades, we can't yet simultaneously record from more than a tiny fraction of the neurons in most of the visual systems currently being studied, which limits our ability to develop a holistic cause-and-effect understanding of how they operate. So it may make sense, as a complement to directly studying a visual system found in nature, to also study synthetic visual systems that in some way resemble it but are easier to inspect. This document describes four lines of work aimed at improving our ability to learn about biological visual systems using models optimized in ways that are analogous to the selective pressures that biological visual systems face, like the pressures to relay accurate information about the world, minimize energy consumption, and withstand perturbation. The first two of these lines of work—discussed in chapters 2 and 3—focus on expanding the space of selective forces that can be factored into optimization-guided models, and the other two—discussed in chapters 4 and 5—focus on modeling particular visual systems (in the macaque and the fruit fly, respectively). Taken together, optimization-guided modeling is shown to be a promising approach to advancing our understanding of visual processing across the animal kingdom, allowing us to leverage hypotheses about the high-level properties of visual systems to amplify the value of sparse neural data.

PUBLISHED CONTENT AND CONTRIBUTIONS

[1] Pinglei Bao et al. “A map of object space in primate inferotemporal cortex”. In: *Nature* 583.7814 (2020), pp. 103–108. URL: https://doi.org/10.1038/s41586-020-2350-5.

[2] Janne K Lappalainen et al. “Connectome-constrained networks predict neural activity across the fly visual system”. In: *Nature* 634.8036 (2024), pp. 1132–1140. URL: https://doi.org/10.1038/s41586-024-07939-3.

[3] Mason McGill and Pietro Perona. “Deciding how to decide: Dynamic routing in artificial neural networks”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 2363–2372.

Chapter 2 is adapted from [3]. My contributions to this line of work included conceptualization, designing and running computational experiments, visualizing the results, and writing.

The work described in chapter 3 has not yet been published, but my contributions to this line of work were similar.

Chapter 4 is adapted from [1]. My contributions to this line of work included (a) reconstructing images presented to macaque monkeys from neural firing patterns, (b) generating latent space embeddings for the stimulus images using both pre-trained and custom convolutional neural networks, (c) mapping deep network latent spaces to IT representation spaces, and (d) generating representation space visualizations that contributed to the characterization of “no man’s land” patches in IT—which contain neurons that preferentially respond to spiky or spindly objects—and the discovery of additional patches containing neurons that prefer stubby objects.

Chapter 5 is adapted from [2]. My contributions to this line of work included (a) writing software to simulate phototransduction in fruit flies, (b) writing software to efficiently train recurrent hexagonal lattice convolutional neural networks, (c) training a variety of connectome-constrained networks, (d) simulating probe stimuli, (e) analyzing the networks’ responses to the probes, (f) generating visualizations, (g) general methodology development, and (h) contributions to the manuscript.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

5.8 Fig. 5.8, continued: We characterize the motion detection mechanism by displaying the PD- and ND-responses of the T4 cell type, and the temporal and spatial contributions of its input cell types according to our connectome-measurement constrained model. Across all T4 cell types, our model predicts that the depolarization in response to PD motion is mainly driven by excitatory Mi1 current inputs (darkest red, solid) from roughly a two-column radius of Mi1 cells. The PD-motion response is increased through excitatory inputs from the neighboring T4 cells of the same type (third darkest red, solid) with the center of mass located towards the leading side of the receptive field (i.e. the motion stimulus towards the PD reaches those T4 cells first, enabling this mechanism). However, for ND-motion the neighboring T4 cells do not provide any excitatory currents (third darkest red, dashed). Tm3 cells provide additional excitatory currents that are, as for Mi1 cells, roughly agnostic to PD vs ND motion. For ND-motion, Mi4 cells decrease excitatory currents from Mi1 by providing roughly matching inhibitory currents from the trailing side of the receptive field (darkest blue, dashed). In contrast, for PD-motion, the inhibition from Mi4 cells is delayed (through the spatial layout and potentially neural time constants not characterized here; darkest blue, solid), which allows a strong depolarization of the T4 cell. CT1 shadows Mi4 in that it provides a similar but weaker inhibition from the same location of the receptive field (second darkest, blue). Noteworthy, our model suggests roles and an additional mechanism for Mi9 cells in **b:** and TmY15 cells in **c:**: both can contribute to the motion detection mechanism by different inhibitory mechanisms for PD-motion with respect to ND-motion. **b:** This figure should be compared to Gruntman et al. 2018, Fig. 4f. Predicted T4c responses to bars moving in the PD (left column) and in the ND (right column) at speeds of $56°/s$, $75°/s$, and $110°/s$ ('Measured', saturated red and blue, speeds varied from top to bottom row). Responses to moving bars are overlaid with the linear sum of responses to the individually flashed frames that constitute the moving bar video sequence ('Linear sum', faint red and blue). Faint grey traces in the background of the first panel show individual flash responses before linear summation. The duration that the flash stimulus was presented in each location precisely matched the duration that the flash remained at the location in the moving bar sequence. Bars were approximately 9° wide and 20.25° high and moved across 45° with respect to the receptive field in the center. **c:** The four T5 cell types detect OFF-edge motion towards the four cardinal directions

Chapter 1

INTRODUCTION

Imagine that you're sitting outside on a nice day and you see a squirrel pop her head out from behind a tree branch. This isn't an especially surprising occurrence, as you've already seen a few other squirrels climbing trees since you sat down. And the experience of seeing this squirrel doesn't have a strong positive or negative emotional valence, and won't cause you to form any long-term memories or perform any immediate action.

But still, the squirrel's decision to poke her head up over the branch has changed something within you. Electrical signals exist in your brain that would not exist if the squirrel wasn't there. And this isn't only true for the squirrel. If any object in your field of view was to suddenly disappear, the pattern of electrical signals being sent through your brain would change. In this sense, there are patterns of neural activity that can be thought of as reflections, shadows, or echoes of objects and surfaces in the world.

And analogous reflections exist in the brains of many other animals. Anatomical and genetic evidence suggests that visual systems may have evolved independently over 40 times in the history of life on planet Earth [5, 11]. Imagine there is a row of lawn chairs extending to your left and right, and on each chair there is a representative animal from one of these lineages. You take a moment to survey your company. In the chair directly to your left is a saltwater tank holding a cuttlefish. In the chair directly to your right is a monarch butterfly sipping from a tiny thimble of lemonade. Many of the remaining chairs hold jellyfish, mollusks, or worms. And a few hold animals that are too small to see.

By sheer good luck, the occupant of every chair seems to be looking directly at the squirrel that just popped her head out from behind the branch. What does the neural reflection of the squirrel's head look like in each of their visual systems? And what about the reflections of the tree, the ground, or the sky? Are there meaningful ways in which these independently evolved representations have something in common? And how can we best conceptualize the dimensions along which they differ?

At a fine-grained mechanistic level, we know quite a lot about the ingredients that contribute to neural representations. We have fairly detailed mathematical models of

how light reflects, diffracts, and scatters when interacting with matter [14, 4], how photoreceptors in our eyes translate photon streams into electrochemical signals [16], and how individual neurons respond to the signals they receive [9, 6].

But at a holistic level, we are much better at modeling what happens before light hits the eye than what happens afterward. An experienced digital artist can create a reasonable approximation of a scene within a few hours or a few days, depending on its complexity. And although the approximation will omit many details—like the contents of closed objects, or the molecular makeup of surfaces—we can use it to predict the image that would be captured for any viewpoint and lighting condition. An analogous approximation for a visual system—which could predict the neural signals that would be sent by any of its components when observing any scene, from any viewpoint, with any lighting condition—has not yet been constructed.

One of the reasons for this, of course, is that observing the behavior of neurons in a living brain is much more difficult than observing how an object responds to light, and the relative sparsity of neural measurements makes it more difficult to identify patterns and distinguish signal from noise. Even despite the amazing advances in neural recording technology that have occurred over the past two decades [10, 19, 3, 8], we can't yet simultaneously record from every neuron in most of the visual systems currently being studied, which limits our ability to develop a holistic cause-and-effect understanding of their operation. So it may make sense, as a complement to directly studying a visual system found in nature, to also study a range of fully inspectable synthetic visual systems that in some way resemble it.

But how can we tell if one visual system resembles another, especially if we don't fully understand how they both work? One approach is to think in terms of the characteristics relevant to natural selection, as it's the process responsible for any similarities between visual systems that evolved independently across the animal kingdom. Once evolution has stumbled upon a particular visual system (or more precisely, a genetic recipe that stochastically unfolds into a visual system through neural development and life experience), it will either exist briefly in a few individuals and then be lost, exist briefly as a stepping stone to further adaptation, or proliferate and remain stable over many generations. A variety of factors influence the likelihood of each of these outcomes, but I'll limit our discussion here to four: (1) the function the visual system performs, (2) the physical space it occupies, (3) the energy it consumes, and (4) its robustness in the face of modification. If we can better understand how optimization targeting these design space dimensions shapes synthetic visual systems,

we may be able to apply this understanding to visual systems found in nature.

1.1 Visual systems optimized to perform a particular function

Animals tend to evolve visual systems that are preferentially suited for tasks that are important to them. For example, we can interpret someone's facial expression much more quickly than we can count the bricks in a wall, even though counting bricks is arguably a simpler task from a computational perspective. Here are a few examples of ethologically relevant time-varying signals that a visual system might attempt to infer:

- **Object type:** This signal indicates whether an object extends through a given point in 3D space $\mathbf{x}$, and, if so, the kind of object that is located there.
- **Object identity:** This signal indicates, for each object in a dynamically updating collection of tracked objects, whether the object extends through a given point in 3D space $\mathbf{x}$.
- **Behavior type:** This signal indicates whether a given point in 3D space $\mathbf{x}$ is contained within an animal exhibiting a recognized behavior, and, if so, the kind of behavior it is exhibiting.
- **Surface trajectory:** This signal indicates whether a given point in 3D space $\mathbf{x}$ lies on a surface, and, if so, the rotational and translational velocity of the surface at $\mathbf{x}$.
- **Self trajectory:** This signal indicates the rotational and translational velocity of the observer.
- **Collision imminence:** This signal indicates whether the observer will collide with something within a given timespan Δt if it doesn't alter its trajectory beforehand.
- **Passageway size:** This signal indicates whether the observer would fit if it was moved so that its center of mass was located at a given point in 3D space $\mathbf{x}$.
- **Illumination level:** This signal indicates how much light is passing through a given point in 3D space $\mathbf{x}$. (Knowing this could be useful for an animal that wants to remain hidden.)

If we were to determine that a biological visual system and a synthetic model of it were both tracking one of these signals—that is, continually estimating its value over time—how similar should we expect them to be? Imagine that you're constructing a processing pipeline to track a signal given some vocabulary of primitive computational elements (*e.g.*, neurons or logic gates). And imagine that you first consider the space of pipelines with input-output latencies below some small duration ε. This will limit both the number of links away an output element can be from an input element—since each element adds some latency—and the total number of elements in the pipeline—since pipelines with more elements are physically larger, and physically larger pipelines require internal signals to be transmitted along longer paths. Imagine that next you consider the space of pipelines with input-output latencies between ε and 2ε, and after that the space of pipelines with latencies between 2ε and 3ε, and so on. As you increase the lower bound on the latency Δt_{min}, the task of estimating the signal of interest will become more difficult, since the estimate at any time t won't be able to take observations made after time $t - \Delta t_{\text{min}}$ into account. But in exchange you'll be able to use a more complex algorithm to perform this task.

In terms of the timescales at which the world can change in ways that would benefit or threaten an animal, the amount of time it can take for information to propagate through a single neuron can be significant [13]. So it might be the case that some vision tasks are so latency-sensitive that only a very restricted set of fairly simple biological visual systems will be suited for them. If a biological visual system and a model of it perform a task like this, then there is a reasonable chance that they'll be algorithmically similar so long as they are of similar complexity in terms of element count, pipeline depth, and the kind of processing individual elements perform.

Other vision tasks may be latency-tolerant enough that processing pipelines with many elements and deep topologies will be viable, in which case a wide variety of qualitatively distinct solutions may exist. If two visual systems were sampled randomly from the set of systems that could perform a task like this, they would probably be very different. However, neither biological evolution nor the processes that are most often used to discover synthetic visual systems—like stochastic gradient descent and synthetic evolution—select randomly from the full set of viable solutions. And they are similarly biased in at least one important way: because they generate solutions based on a finite amount of information—whether in the form of error measurements, reinforcement signals, or offspring counts—they tend to prefer simpler

solutions. (*i.e.*, solutions with shorter description lengths, for some solution-encoding scheme.) This means that, even if a biological visual system performs a task for which a wide variety of alternative solutions exist, synthetic visual systems optimized to perform the same task may cluster in a subspace of likely-to-be-discovered solutions that is much smaller than the full space of viable solutions, and solutions in this subspace may be mechanistically similar to the biological visual system even if the vast majority of solutions outside it are not.

1.2 Visual systems optimized for space efficiency

Animals with larger bodies require more energy to move and stay alive, and have more difficulty hiding from potential predators and prey. So any biological structure that increases an animal's size without granting it some kind of advantage—like the strength and speed conferred by muscle or the protection conferred by a thick shell—will tend to be selected away.

For a visual system, the effect of optimizing for space efficiency is similar to the effect of optimizing to minimize processing latency, albeit with a few important differences. When minimizing the processing latency of a visual system, modifications will be favored if they decrease the wiring length or the number of processing elements along the highest-latency input-to-output path. Such modifications will still be favored when optimizing for space efficiency, but modifications that remove elements or decrease the amount of wiring elsewhere in the system will be favored as well. At the algorithmic level, this means that optimizing for either processing latency or spatial footprint will tend to result in simpler solutions, but processes that weigh responsiveness more heavily than compactness will tend to decompose computations into parts that can be carried out in parallel, whereas processes that weigh compactness over responsiveness may decompose computations into many stages applied serially.

1.3 Visual systems optimized for energy efficiency

Attempting to minimize the amount of energy expended by a visual system will tend to make it simpler as well, since every additional processing element or micrometer of wiring can require energy to maintain, and expend additional energy during operation. However, unlike the spatial footprint of a visual system, the amount of energy that a visual system uses can change over time. And investing energy into any particular form of visual information processing may be worthwhile in some circumstances but not in others. This means that a visual system optimized for energy efficiency may perform conditional computation, activating only a subset

of its components at any given time, based on the sensory input and/or top-down control signals it has received.

It is worth noting though that a visual system that makes extensive use of conditional computation may be far from optimal from the perspective of space efficiency. If only a small, specialized portion of a visual system is active at a given time, and each of these subsystems must be sophisticated enough to perform useful information processing, then the system in its totality might need to be very large. Because of this tension, the extent to which a visual system in nature allocates energy dynamically may reflect the balance of the selective forces that shaped it.

1.4 Visual systems optimized for robustness

Even if evolution has discovered a locally optimal visual system for a particular animal, the central nervous system housing it may still be in flux. And, since a single gene can affect a wide swath of an animal's nervous system [22, 1], evolution may select for genes that contribute to adaptive non-vision-related capabilities, even if they have a somewhat disruptive effect on the visual system. After such a gene propagates through a population, it may take a long time for additional genes that "fix" the visual system to eventually arise. So, if neural innovation beyond the visual system is fairly frequent, the visual system may be operating under the burden of multiple genetic handicaps at a given time, above and beyond any handicaps accumulated via genetic drift. This means that, all else being equal, visual systems that degrade more gracefully in response to perturbation will tend to outcompete brittle visual systems.

Characterizing the extent to which a visual system is robust to modification is a complicated endeavor, since it requires analyzing the behavior of the original visual system, the ways in which it is likely to be perturbed, and the behavior of the systems resulting from the various potential perturbations. Empirically though, we can observe that when humans engineer systems to be robust, these systems tend to have three properties: (1) some degree of modularity, which limits the scope of a potential perturbation, (2) some degree of redundancy, and (3) some mechanism for identifying when a module is failing, and adapting to circumvent it [12, 7]. Modularity may arise naturally in a space-constrained visual system as a result of minimizing total wiring length. And the capacity for self-reconfiguration is compatible with the conditional computation that might arise in an energy-constrained system. But redundancy can only increase the amount of space and energy a system requires, so the extent to which a visual system in nature contains redundant components will be a function of

the extent to which the selective pressure for robustness is significant in relation to the selective pressure for efficiency.

1.5 Overview of the following chapters

The following chapters describe four lines of work aimed at improving our collective ability to learn about visual systems in nature using optimization-guided models. Broadly, the first two of these focus on expanding the space of selective forces that can be factored into optimization-guided models, and the second two focus on modeling particular visual systems.

chapter 2 presents strategies for optimizing artificial neural networks that can dynamically activate and inactivate subcomponents and attempt to balance task performance with energy efficiency. In evaluating these optimization strategies, we found that, in dynamically-routed networks trained to classify images, subnetworks became specialized to process distinct categories of images. And increasing the cost of energy reduced the average fraction of the network that was active at a given time, in line with the discussion of energy constraints in section 1.3. Additionally, given a fixed computational budget, dynamically-routed networks tended to perform better than comparable statically-routed networks.

chapter 3 presents an approach for simulating the evolution of neural networks encoding an innate behavior—threat detection—with very large populations, and biologically plausible (albeit simplified) genetic mechanisms. We were able to evolve populations of neural networks that integrated sensory information over time, were near-optimally adapted for their environment (>99% of the maximum possible fitness), and recapitulated predator-avoidance behavior observed in fruit flies. We found that these high-fitness networks were discovered more quickly in environments where subpopulations could remain isolated from each other for many generations, and we found that in these environments subpopulations competed for territory, with bands of low-fitness hybrids forming at territorial boundaries. This observation—that neural innovation occurred through parallel exploration followed by competition rather than gradual, cooperative refinement—suggests that, over long timescales, visual systems that can tolerate some amount of perturbation and continue to function will likely outcompete visual systems that can not, which is consistent with the discussion of the adaptive benefits of robustness in section 1.4.

chapter 4 discusses using an artificial neural network optimized to classify images to better understand how objects are represented in the macaque visual system.

The artificial network was not explicitly optimized for space efficiency or energy efficiency, but it did roughly match the complexity of the pathways being modeled in terms of the number of processing elements between inputs and outputs. (Signals may have traversed slightly more processing elements on average in the artificial network, but the artificial network also had simpler processing elements.) We constructed a low-dimensional embedding space for general object images using the object representations learned by the artificial network, and used this space to interpret the recordings from cells in macaque inferotemporal (IT) cortex. Anatomically, cells were clustered into four networks according to the first two principal components of their preferred axes in this object space. This four-quadrant map spanned three hierarchical stages of increasing view-invariance, and the cells within it collectively harbored sufficient coding capacity to approximately reconstruct object images. Notably, the observation that many images only elicited significant activity in one of four IT subnetworks is consistent with the observations about optimization for energy efficiency inducing specialization presented in chapter 2.

chapter 5 discusses combining task-performance optimization with strict anatomical constraints to obtain a fine-grained predictive model of the fruit fly visual system. We constructed a model neural network with the experimentally determined connectivity for 64 cell types in the motion pathways of the fruit fly optic lobe [15, 20, 21, 17, 18], but with unknown parameters for other neuron and synapse properties. We then optimized the values of those unknown parameters to allow the model network to represent visual motion [2]. Our mechanistic model made detailed, experimentally testable predictions for each neuron in the connectome, and we found that these predictions agreed with experimental measurements of neural activity across 26 studies. We were unable to achieve the same predictive power using either structural information alone or the task hypothesis alone, illustrating how optimization-guided modeling can combine synergistically with direct measurement.

References

[1] Robert RH Anholt. “Evolution of epistatic networks and the genetic basis of innate behaviors”. In: *Trends in Genetics* 36.1 (2020), pp. 24–29.

[2] Daniel J Butler et al. “A Naturalistic Open Source Movie for Optical Flow Evaluation (Sintel)”. In: *Eccv* (2012), pp. 611–625. DOI: 10.1007/978-3-642-33783-3_44.

[3] Tsai-Wen Chen et al. “Ultrasensitive fluorescent proteins for imaging neuronal activity”. In: *Nature* 499.7458 (2013), pp. 295–300.

[4] Bernardt Duvenhage, Kadi Bouatouch, and Derrick G Kourie. “Numerical verification of bidirectional reflectance distribution functions for physical plausibility”. In: *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*. 2013, pp. 200–208.

[5] Russell D Fernald. “Casting a genetic light on the evolution of eyes”. In: *Science* 313.5795 (2006), pp. 1914–1918.

[6] Carl Gold, Darrell A Henze, and Christof Koch. “Using extracellular action potential recordings to constrain compartmental models”. In: *Journal of computational neuroscience* 23 (2007), pp. 39–58.

[7] Ellis F Hitt and Dennis Mulcare. “Fault-tolerant avionics”. In: *Avionics: development and implementation* 2 (2001).

[8] Daniel R Hochbaum et al. “All-optical electrophysiology in mammalian neurons using engineered microbial rhodopsins”. In: *Nature methods* 11.8 (2014), pp. 825–833.

[9] Alan L Hodgkin and Andrew F Huxley. “A quantitative description of membrane current and its application to conduction and excitation in nerve”. In: *The Journal of physiology* 117.4 (1952), p. 500.

[10] James J Jun et al. “Fully integrated silicon probes for high-density recording of neural activity”. In: *Nature* 551.7679 (2017), pp. 232–236.

[11] Kristen M Koenig and Jeffrey M Gross. “Evolution and development of complex eyes: a celebration of diversity”. In: *Development* 147.19 (2020), dev182923.

[12] Israel Koren and C Mani Krishna. *Fault-tolerant systems*. Morgan Kaufmann, 2020.

[13] Matthew E Larkum, MARC G Rioult, and HANS-R Luscher. “Propagation of action potentials in the dendrites of neurons from rat spinal cord slice cultures”. In: *Journal of neurophysiology* 75.1 (1996), pp. 154–170.

[14] Fred E Nicodemus. “Directional reflectance and emissivity of an opaque surface”. In: *Applied optics* 4.7 (1965), pp. 767–775.

[15] Marta Rivera-Alba et al. “Wiring Economy and Volume Exclusion Determine Neuronal Placement in the Drosophila Brain”. In: *Current Biology* 21.23 (Dec. 2011), pp. 2000–2005.

[16] Julie L Schnapf and Denis A Baylor. “How photoreceptor cells respond to light”. In: *Scientific American* 256.4 (1987), pp. 40–47.

[17] Kazunori Shinomiya et al. “Comparisons between the ON- and OFF-edge motion pathways in the Drosophila brain.” In: *Elife* 8 (Jan. 2019), p. 2431.

[18] Kazunori Shinomiya et al. “Neuronal circuits integrating visual motion information in Drosophila melanogaster”. In: *Current Biology* 32.16 (2022), pp. 3529–3544.

[19] Nicholas A Steinmetz et al. “Neuropixels 2.0: A miniaturized high-density probe for stable, long-term brain recordings”. In: *Science* 372.6539 (2021), eabf4588.

[20] Shin-ya Takemura et al. “Synaptic circuits and their variations within different columns in the visual system of Drosophila ”. In: *Proceedings of the National Academy of Sciences* 112.44 (2015), pp. 13711–13716. ISSN: 0027-8424. DOI: 10.1073/pnas.1509820112.

[21] Shin-ya Takemura et al. “The comprehensive connectome of a neural substrate for ‘ON’ motion detection in Drosophila”. In: *eLife* 6 (2017), pp. 1–16. DOI: 10.7554/elife.24394.

[22] Akihiko Yamamoto et al. “Neurogenetic networks for startle-induced locomotion in Drosophila melanogaster”. In: *Proceedings of the national academy of sciences* 105.34 (2008), pp. 12393–12398.

Chapter 2

DECIDING HOW TO DECIDE: DYNAMIC ROUTING IN ARTIFICIAL NEURAL NETWORKS

Forward

This chapter discusses three strategies for training dynamically-routed artificial neural networks: graphs of learned transformations through which different input signals may take different paths. Though some approaches have advantages over others, the resulting networks are often qualitatively similar. We found that, in dynamically-routed networks trained to classify images, subnetworks become specialized to process distinct categories of images. Additionally, given a fixed computational budget, dynamically-routed networks tend to perform better than comparable statically-routed networks.

This chapter is adapted from a 2017 International Conference on Machine Learning (ICML) paper with the same name, coauthored with Pietro Perona [22]. Due to advances in computing hardware over the past eight years, the experiments described in this chapter may seem small in scale to someone reading in 2025. However, the central premise these experiments explore—that energy constraints can push a vision system to decompose what we might think of as a single task into subtasks, construct subsystems optimized for particular subtasks, and learn to activate each subsystem as it is needed—remains relevant today.

2.1 Introduction

Some decisions are easier to make than others—for example, large, unoccluded objects are often easier to recognize. Additionally, different difficult decisions may require different expertise—an avid birder may know very little about identifying cars. We hypothesize that complex decision-making tasks like visual classification can be meaningfully divided into specialized subtasks, and that a system designed to perform a complex task should first attempt to identify the subtask being presented to it, and then use that information to select the most suitable algorithm for its solution.

This approach—dynamically routing signals through an inference system, based

on their content—has already been incorporated into machine vision pipelines via methods such as boosting [32], coarse-to-fine cascades [33], and random decision forests [11]. Dynamic routing is also performed in the primate visual system: spatial information is processed somewhat separately from object identity information [9], and faces and other behaviorally-relevant stimuli ellicit responses in anatomically distinct, specialized regions [23, 17]. However, state-of-the-art artificial neural networks (ANNs) for visual inference are routed statically [27, 10, 6, 24]; every input triggers an identical sequence of operations.

With this in mind, we propose a mechanism for introducing cascaded evaluation to arbitrary feedforward ANNs, focusing on the task of object recognition as a proof of concept. Instead of classifying images only at the final stage of processing, multiple subnetworks may attempt to classify images in low-ambiguity regions of their input space, passing ambiguous images forward to subsequent subnetworks for further consideration (see Fig. 2.1 for an illustration). We propose three approaches to training these networks, test them on small image datasets synthesized from MNIST [19] and CIFAR-10 [18], and quantify the accuracy/efficiency trade-off that occurs when the network parameters are tuned to yield more aggressive early classification policies. Additionally, we propose and evaluate methods for appropriating regularization and optimization techniques developed for statically-routed networks.

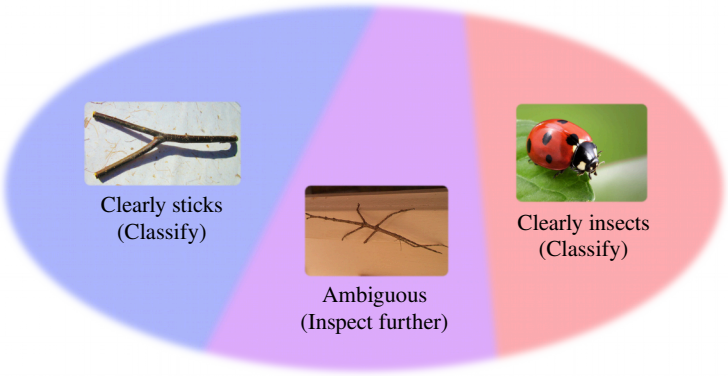

Figure 2.1: **Motivation for dynamic routing.** For a given data representation, some regions of the input space may be classified confidently, while other regions may be ambiguous.

2.2 Related work

Since the late 1980s, researchers have combined artificial neural networks with decision trees in various ways [31] [28]. More recently, approaches have been propsed to jointly optimize ANN and decision tree parameters [16], and randomized

multi-layer networks have been used to compute decision tree split functions [2]. To our knowledge, the family of inference systems we discuss was first described in [5]. Additionally, [1] explores dynamically skipping layers in neural networks, and [13] explores dynamic routing in networks with equal-length paths. Some recently-developed visual detection systems perform cascaded evaluation of convolutional neural network layers [20, 3, 7, 25]; though highly specialized for the task of visual detection, these modifications can radically improve efficiency.

While these approaches lend evidence that dynamic routing can be effective, they either ignore the cost of computation, or do not represent it explicitly, and instead use opaque heuristics to trade accuracy for efficiency. We build on this foundation by deriving training procedures from arbitrary application-provided costs of error and computation, comparing one actor-style and two critic-style strategies, and considering regularization and optimization in the context of dynamically-routed networks.

2.3 Setup

The computation performed by a statically-routed feedforward artificial neural network can be decomposed into a sequence of subcomputations, each of which generates a single output vector from a single input vector, which is either the output of the previous subcomputation or, in the case of the first subcomputation, an input from an external source. For a given such decomposition—because multiple will likely exist for a single network—we'll call a subnetwork that performs a subcomputation a *module*, and we'll call a subnetwork that directly uses a given module's output that module's *sink*.

We consider networks in which modules may have more than one sink. In such a network, for every n-way junction $\mathcal{J}$ a signal reaches, the network must make a decision, $d(\mathcal{J}) \in \{0..n\}$, such that the signal will propagate through the i^{th} sink if and only if $d(\mathcal{J}) = i$ (this is illustrated in Fig. 2.2). We compute $d(\mathcal{J})$ as the argmax of the score vector $\mathbf{s}(\mathcal{J})$, a learned function of the last feature vector computed before reaching $\mathcal{J}$. We'll refer to this rule for generating routing decisions from score vectors as the inference routing policy.

2.3.1 Multipath architectures for convolutional networks

Convolutional network modules compute collections of *local* descriptions of the input signal. It is unreasonable to expect that this kind of feature vector can explicitly encode the global information relevant to deciding how to route the entire signal

(*e.g.*, in the case of object recognition, whether the image was taken indoors, whether the image contains an animal, or the prevalence of occlusion in the scene).

To address this, instead of computing a 2-dimensional array of local features at each module, we compute a pyramid of features (resembling the pyramids described by [15]), with local descriptors at the bottom and global descriptors at the top. At every junction $\mathcal{J}$, the score vector $\mathbf{s}(\mathcal{J})$ is computed by a small routing network operating on the last-computed global descriptor. Our multipath architecture is illustrated in Fig. 2.3.

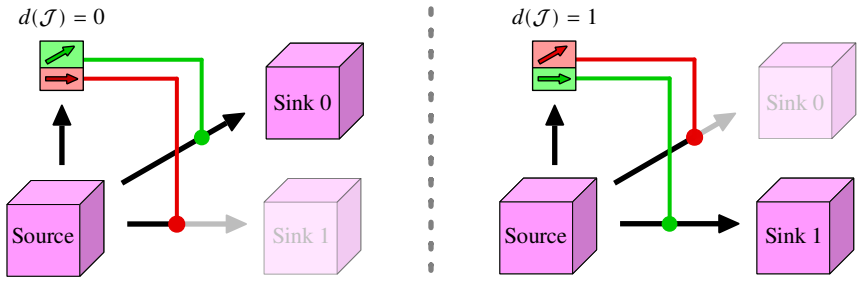

Figure 2.2: **A 2-way junction**, $\mathcal{J}$. $d(\mathcal{J})$ is an integer function of the source features. When $d(\mathcal{J}) = 0$, the signal is propagated through the top sink, and the bottom sink is inactive. When $d(\mathcal{J}) = 1$, the signal is propagated through the bottom sink, and the top sink is inactive.

2.3.2 Balancing accuracy and efficiency

For a given input, network $\mathcal{N}$, and sequence of routing decisions $\mathcal{D}$, we define the cost of performing inference:

$$c_{\text{inf}}(\mathcal{N}, \mathcal{D}) := c_{\text{err}}(\mathcal{N}, \mathcal{D}) + c_{\text{cpt}}(\mathcal{N}, \mathcal{D}), \tag{2.1}$$

where $c_{\text{err}}(\mathcal{N}, \mathcal{D})$ is the cost of the inference errors made by the network, and $c_{\text{cpt}}(\mathcal{N}, \mathcal{D})$ is the cost of computation. In our experiments, unless stated otherwise, c_{err} is the cross-entropy loss and

$$c_{\text{cpt}}(\mathcal{N}, \mathcal{D}) = \alpha_{\text{cpt}}\, n_{\text{ops}}(\mathcal{N}, \mathcal{D}), \tag{2.2}$$

where $n_{\text{ops}}(\mathcal{N}, \mathcal{D})$ is the number of multiply-accumulate operations performed and α_{cpt} is a scalar hyperparameter. This definition assumes a time- or energy-constrained system—every operation consumes roughly the same amount of time and energy, so every operation is equally expensive. c_{cpt} may be defined differently under other constraints (*e.g.* memory bandwidth).

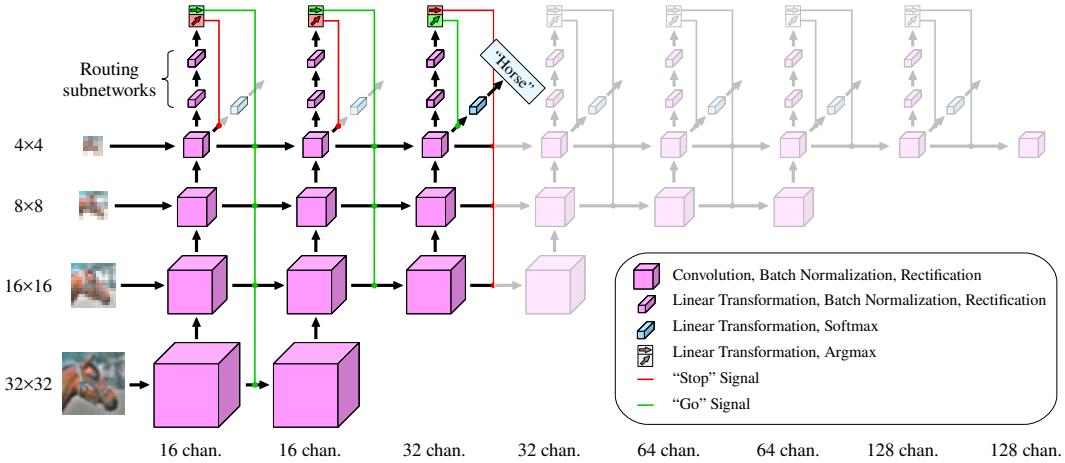

Figure 2.3: **Our multiscale convolutional architecture.** Once a column is evaluated, the network decides whether to classify the image or evaluate subsequent columns. Deeper columns operate at coarser scales, but compute higher-dimensional representations at each location. All convolutions use 3×3 kernels, downsampling is achieved via 2×2 max pooling, and all routing subnetwork layers have 16 channels.

2.4 Training

We propose three approaches to training dynamically-routed networks, along with complementary approaches to regularization and optimization, and a method for adapting to changes in the cost of computation.

2.4.1 Training strategy i: Actor learning

Since any given sequence of routing decisions $\mathcal{D}$ is discrete, $c_{\text{inf}}(\mathcal{N}, \mathcal{D})$ cannot be minimized via gradient-based methods. However, if $\mathcal{D}$ is replaced by a stochastic approximation, $\hat{\mathcal{D}}$, during training, we can engineer the gradient of the expectation $\mathrm{E}(c_{\text{inf}}(\mathcal{N}, \hat{\mathcal{D}}))$ to be nonzero. We can then learn the routing parameters and classification parameters simultaneously by minimizing the loss

$$\ell_{\text{ac}} := \mathrm{E}\Big(c_{\text{inf}}(\mathcal{N}, \hat{\mathcal{D}})\Big). \tag{2.3}$$

In our experiments, the training routing policy samples $\hat{\mathcal{D}}$ such that, for a given junction $\mathcal{J}$, the corresponding decision $\hat{d}(\mathcal{J})$ is sampled

$$\hat{d}(\mathcal{J}) \sim \text{Categorical}\Big(\text{softmax}(\mathbf{s}(\mathcal{J})/\tau)\Big), \tag{2.4}$$

where τ is the network "temperature": a scalar hyperparameter that decays over the course of training, converging the training routing policy towards the inference routing policy.

2.4.2 Training strategy ii: Pragmatic critic learning

Alternatively, we can attempt to learn to predict the expected utility of making every routing decision. In this case, we minimize the loss

$$\ell_{\text{cr}} := \mathrm{E}\left(c_{\text{inf}}(\mathcal{N}, \hat{\mathcal{D}}) + \sum_{\mathcal{J} \in \mathcal{P}(\hat{\mathcal{D}})} c_{\text{ure}}(\mathcal{J})\right), \tag{2.5}$$

where the activation path $\mathcal{P}(\hat{\mathcal{D}})$ is the set of junctions encountered when making the sequence of routing decisions $\hat{\mathcal{D}}$, and c_{ure} is the utility regression error cost. For an n-way junction $\mathcal{J}$, this cost is defined:

$$c_{\text{ure}}(\mathcal{J}) := \alpha_{\text{ure}} \left\| \mathbf{s}(\mathcal{J}) - \mathbf{u}(\mathcal{J}) \right\|^2, \tag{2.6}$$

where α_{ure} is a scalar hyperparameter and the utility vector $\mathbf{u}(\mathcal{J})$ is a vector in $\mathbb{R}^n$ whose i^{th} component is

$$u_i(\mathcal{J}) = \mathrm{E}\Big(-c_{\text{inf}}(\mathcal{N}, \hat{\mathcal{D}}) \,\Big|\, \mathcal{J} \in \mathcal{P}(\hat{\mathcal{D}}) \text{ and } \hat{d}(\mathcal{J}) = i\Big). \tag{2.7}$$

Since we want to learn the policy indirectly (via cost prediction), $\hat{\mathcal{D}}$ is treated as constant with respect to optimization.

2.4.3 Training Strategy iii: Optimistic critic learning

To improve the stability of the loss and potentially accelerate training, we can adjust the routing utility function $\mathbf{u}$ such that, for every junction $\mathcal{J}$, $\mathbf{u}(\mathcal{J})$ is independent of the routing parameters downstream of $\mathcal{J}$. Instead of predicting the cost of making routing decisions given the *current* downstream routing policy, we can predict the cost of making routing decisions given the *optimal* downstream routing policy. In this optimistic variant of the critic method,

$$u_i(\mathcal{J}) = \max\Big\{-c_{\text{inf}}(\mathcal{N}, \hat{\mathcal{D}}) \,\Big|\, \mathcal{J} \in \mathcal{P}(\hat{\mathcal{D}}) \text{ and } \hat{d}(\mathcal{J}) = i\Big\}. \tag{2.8}$$

2.4.4 Regularization

Many regularization techniques involve adding a model-complexity term, c_{mod}, to the loss function to influence learning, effectively imposing soft constraints upon the network parameters [12, 26, 30]. However, if such a term affects network modules in a way that is independent of the amount of signal routed through them, it will either underconstrain frequently-used modules or overconstrain infrequently-used modules. To support both frequently- and infrequently-used modules, we regularize modules as they are activated by $\hat{\mathcal{D}}$, instead of regularizing the entire network directly.

For example, to apply L2 regularization to critic networks, we define c_{mod}:

$$c_{\text{mod}} := \mathrm{E}\left(\alpha_{\text{L2}} \sum_{w \in \mathcal{W}(\hat{\mathcal{D}})} w^2\right), \tag{2.9}$$

where $\mathcal{W}(\hat{\mathcal{D}})$ is the set of weights associated with the modules activated by $\hat{\mathcal{D}}$, and α_{L2} is a scalar hyperparameter.

For actor networks, we apply an extra term to control the magnitude of s, and therefore the extent to which the net explores subpotimal paths:

$$c_{\text{mod}} := \mathrm{E}\left(\alpha_{\text{L2}} \sum_{w \in \mathcal{W}(\hat{\mathcal{D}})} w^2 + \alpha_{\text{dec}} \sum_{\mathcal{J} \in \mathcal{P}(\hat{\mathcal{D}})} \|\mathbf{s}(\mathcal{J})\|^2\right), \tag{2.10}$$

where α_{dec} is a scalar hyperparameter indicating the relative cost of decisiveness.

c_{mod} is added to the loss function in all of our experiments. Within c_{mod}, unless stated otherwise, $\hat{\mathcal{D}}$ is treated as constant with respect to optimization.

2.4.5 Adjusting learning rates to compensate for throughput variations

Both training techniques attempt to minimize the expected cost of performing inference with the network, over the training routing policy. With this setup, if we use a constant learning rate for every module in the network, then modules through which the policy routes examples more frequently will receive larger parameter updates, since they contribute more to the expected cost. To allow every module to learn as quickly as possible, we scale the learning rate of each module $\mathcal{M}$ dynamically, by a gain factor $\gamma(\mathcal{M})$, such that the elementwise variance of the loss gradient with respect to $\mathcal{M}$'s parameters is independent of the amount of probability density routed through it.

To derive $\gamma(\mathcal{M})$, we consider an alternative routing policy that routes all signals though $\mathcal{M}$, and then routes through subsequent modules following the training-time routing policy of the optimization method being used. Let $\boldsymbol{\delta}^*(\mathcal{M})$ be the change in $\mathcal{M}$'s parameter vector that would be induced by performing a single iteration of mini-batch stochastic gradient descent. This update vector can be computed

$$\boldsymbol{\delta}^*(\mathcal{M}) = -\lambda \sum_{i \in \{1..n_{\text{ex}}\}} \mathbf{g}(\mathcal{M}, i), \tag{2.11}$$

where λ is the global learning rate, n_{ex} is the number of examples in the mini-batch, and $\mathbf{g}(\mathcal{M}, i)$ is the gradient of the loss with respect to the parameters in $\mathcal{M}$, for

training example i, under the $\mathcal{M}$-biased routing policy. Analogously, the scaled parameter adjustment using the unbiased policy can be written

$$\delta(\mathcal{M}) = -\lambda\, \gamma(\mathcal{M}) \sum_{i \in \{1..n_{\text{ex}}\}} p(\mathcal{M}, i)\, \mathbf{g}(\mathcal{M}, i), \tag{2.12}$$

where $p(\mathcal{M}, i)$ is the probability with which the unbiased policy routes example i to $\mathcal{M}$.

We want to select $\gamma(\mathcal{M})$ such that

$$\text{Var}(\delta(\mathcal{M})) = \text{Var}(\delta^*(\mathcal{M})). \tag{2.13}$$

Substituting the definitions of $\delta(\mathcal{M})$ and $\delta^*(\mathcal{M})$,

$$\text{Var}\left(\gamma(\mathcal{M}) \sum_{i \in \{1..n_{\text{ex}}\}} p(\mathcal{M}, i)\, \mathbf{g}(\mathcal{M}, i)\right) = \text{Var}\left(\sum_{i \in \{1..n_{\text{ex}}\}} \mathbf{g}(\mathcal{M}, i)\right). \tag{2.14}$$

Since every $\mathbf{g}(\mathcal{M}, i)$ is sampled independently via the same mechanism, we can rewrite this equation:

$$n_{\text{ex}}\, v(\mathcal{M})\, \gamma(\mathcal{M})^2\, \|p(\mathcal{M})\|^2 = n_{\text{ex}}\, v(\mathcal{M}), \tag{2.15}$$

where $\mathbf{p}(\mathcal{M})$ the vector in $[0, 1]^{n_{\text{ex}}}$ whose i^{th} component is $p(\mathcal{M}, i)$, and $v(\mathcal{M})$ is the elementwise variance of $\mathbf{g}(\mathcal{M}, i)$, for any $i \in \{1..n_{\text{ex}}\}$. We can now show that

$$\gamma(\mathcal{M}) = \|\mathbf{p}(\mathcal{M})\|^{-1}. \tag{2.16}$$

So, for every module $\mathcal{M}$, we can scale the learning rate by $\|\mathbf{p}(\mathcal{M})\|^{-1}$, and the variance of the weight updates will be similar thoughout the network. We use this technique, unless otherwise specified, in all of our experiments.

2.4.6 Responding to changes in the cost of computation

We may want a single network to perform well in situations with various degrees of computational resource scarcity (*e.g.* computation may be more expensive when a device battery is low). To make the network's routing behavior responsive to a dynamic c_{cpt}, we can concatenate the operation cost α_{cpt} to the input of every routing subnetwork, to allow it to modulate the routing policy. To match the scale of the image features and facilitate optimization, we express α_{cpt} in units of cost per ten million operations.

2.4.7 Hyperparameters

In all of our experiments, we used a mini-batch size, n_{ex}, of 128, and trained for 80,000 iterations. We performed stochastic gradient descent using an initial learning rate of $0.1/n_{\text{ex}}$ and a momentum coefficient of 0.9. The learning rate decayed continuously with a half-life of 10,000 iterations. The weights of the final layers of routing networks were zero-initialized, and we initialized all other weights using the Xavier initialization method [8]. All biases were zero-initialized. We performed batch normalization [14] before every rectification operation, with an ϵ of 1×10^{-6}, and an exponential moving average decay constant of 0.9.

τ was initialized to 1.0 for actor networks and 0.1 for critic networks, and decayed with a half-life of 10,000 iterations. We set α_{dec} to 0.01, α_{ure} to 0.001, and α_{L2} to 1×10^{-4}. We selected these values (for τ, α_{dec}, α_{ure}, and α_{L2}) by exploring the hyperparameter space logarithmically, by powers of 10, training and evaluating on the hybrid MNIST/CIFAR-10 dataset we describe in section 2.5.1. At a coarse level, these values are locally optimal—multiplying or dividing any of them by 10 will not improve performance.

2.4.8 Data augmentation

We augmented our training data using an approach that is popular for use with CIFAR-10 [21, 29, 4]. We augmented each image by applying vertical and horizontal shifts sampled uniformly from the range $[-4\text{px}, 4\text{px}]$, and, if the image was from CIFAR-10, flipping it horizontally with probability 0.5. We filled blank pixels introduced by shifts with the mean color of the image (after gamma-decoding).

2.5 Experiments

We explored the design space described in the previous section by training 153 networks to classify small images, varying the policy-learning strategy, regularization strategy, optimization strategy, architecture, cost of computation, and details of the task. The results of these experiments are reported in Fig. 2.8–Fig. 2.10, and our source code is available on GitHub.

2.5.1 Comparing policy-learning strategies

To compare routing strategies in the context of a simple dataset with a high degree of difficulty variation, we trained networks to classify images from a small-image dataset synthesized from MNIST [19] and CIFAR-10 [18] (see Fig. 2.4). This dataset included the classes “0”, “1”, “2”, “3”, and “4” from MNIST and “airplane”,

"automobile", "deer", "horse", and "frog" from CIFAR-10 (see Fig. 2.4). The images from MNIST were resized to match the scale of images from CIFAR-10 (32×32), via linear interpolation, and were color-modulated to make them more difficult to trivially distinguish from CIFAR-10 images. (MNIST is a grayscale dataset.)

For a given computational budget, dynamically-routed networks achieved higher accuracy rates than architecture-matched statically-routed baselines (networks composed of the first n columns of the architecture illustrated in Fig. 2.3, for $n \in \{1..8\}$). Additionally, dynamically-routed networks tended to avoid routing data along deep paths at the beginning of training (see Fig. 2.7). This is possibly because the error surfaces of deeper networks are more complicated, or because deeper paths are less stable—changing the parameters in any constituent module to better classify images routed along other, overlapping paths may decrease performance. Whatever the mechanism, this tendency to initially find simpler solutions seemed to prevent some of the overfitting that occured with 7- and 8-module statically-routed networks.

Compared to other dynamically-routed networks, optimistic critic networks performed poorly, possibly because optimal routers were a poor approximation for our small, low-capacity router networks. Actor networks performed better than critic networks, possibly because the critic networks were forced to learn a potentially-intractable auxilliary task. (i.e., it's easier to decide who to call to fix your printer than it is to predict exactly how quickly and effectively everyone you know would fix it.) Actor networks also consistently achieved higher peak accuracy rates than comparable statically-routed networks, across experiments.

Although actor networks may be more performant, critic networks are more flexible. Since critic networks don't require $\mathrm{E}(c_{\mathrm{inf}}(\mathcal{N}, \hat{\mathcal{D}}))$ to be a differentiable function of

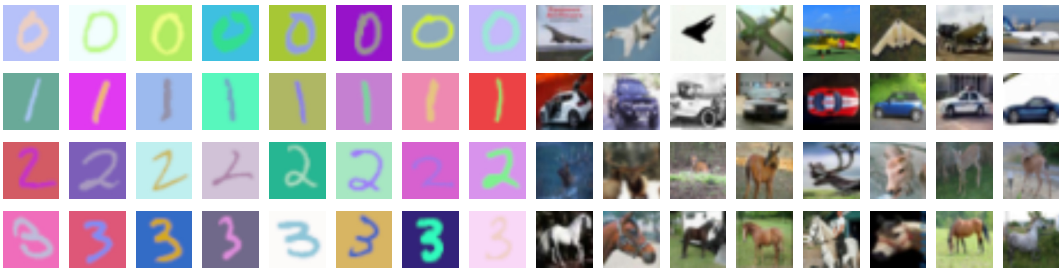

Figure 2.4: **Sample images from the hybrid MNIST/CIFAR-10 dataset.** We recolored images from MNIST via the following procedure: we selected two random colors at least 0.3 units away from each other in RGB space; we then mapped black pixels to the first color, mapped white pixels to the second color, and linearly interpolated in between.

$\hat{\mathcal{D}}$, they can be trained by sampling $\hat{\mathcal{D}}$, saving memory, and they support a wider selection of training routing policies (e.g. ϵ-greedy) and c_{inf} definitions. In addition to training the standard critic networks, we trained networks using a variant of the pragmatic critic training policy, in which we replaced the cross-entropy error in the definition of c_{ure} with the classification error. Although these networks did not perform as well as the original pragmatic critic networks, they still outperformed comparable statically-routed networks.

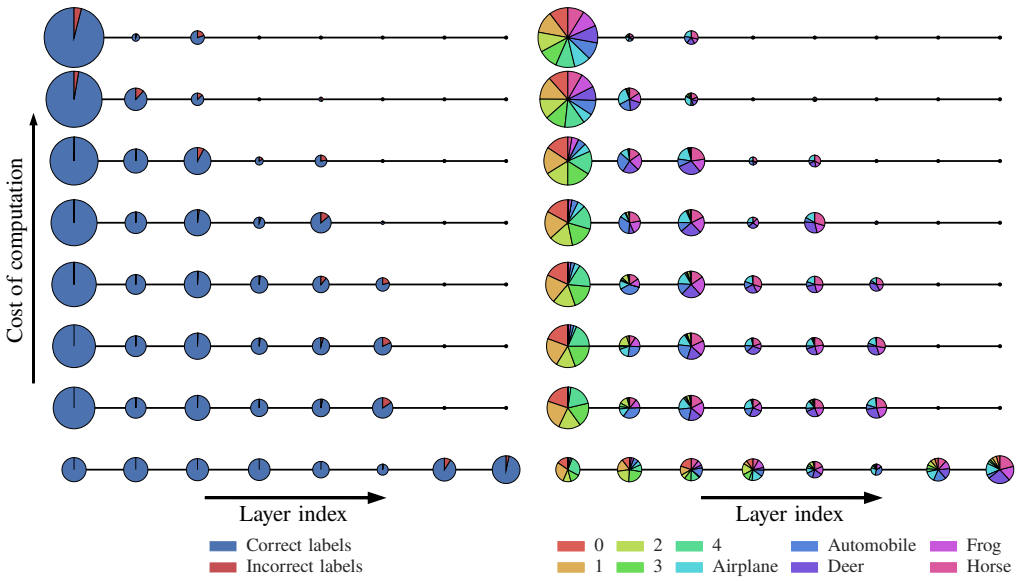

Figure 2.5: **Dataflow through actor networks** trained to classify images from the hybrid MNIST/CIFAR-10 dataset. Every row is a node-link diagram corresponding to a network, trained with a different α_{cpt}. Each circle indicates, by area, the fraction of examples that are classified at the corresponding module. The circles are colored to indicate the accuracy of each module (left) and the kinds of images classified at each module (right).

2.5.2 Comparing regularization strategies

In on our experiments, regularizing the training-time routing policy, as described in section 2.4.4, discouraged networks from routing data along deep paths, reducing peak accuracy. Additionally, some mechanism for encouraging exploration (in our case, a nonzero α_{dec}) appeared to be necessary to train effective actor networks.

2.5.3 Comparing optimization strategies

Throughput-adjusting the learning rates (TALR), as described in section 2.4.5, improved the hybrid dataset performance of both actor and critic networks in computational-resource-abundant, high-accuracy contexts.

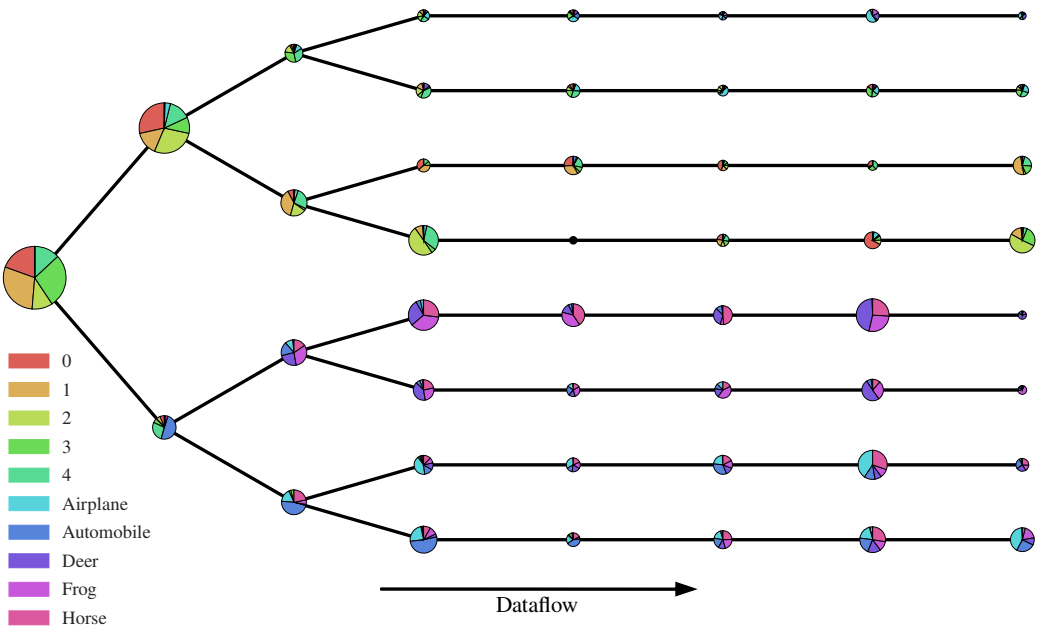

Figure 2.6: **Dataflow through a branching actor network** trained to classify images in the hybrid dataset, illustrated as in Fig. 2.5.

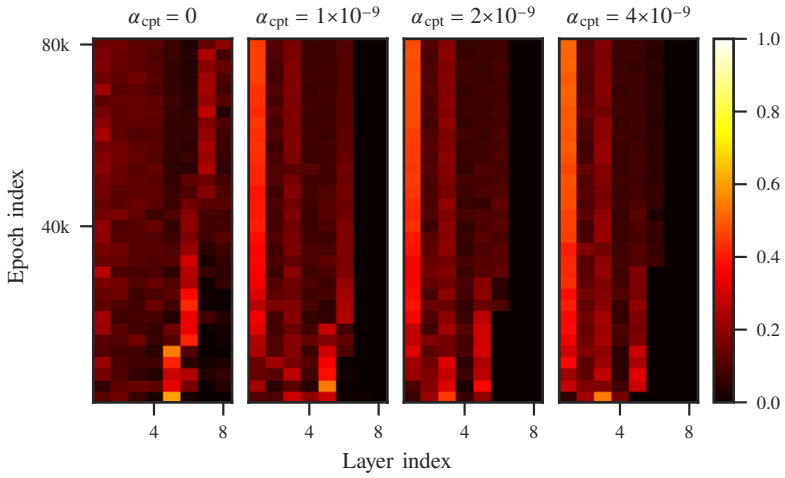

Figure 2.7: **Dataflow over the course of training.** The heatmaps illustrate the fraction of validation images classified at every terminal node in the bottom four networks in Fig. 2.5, over the course of training.

2.5.4 Comparing architectures

For a given computational budget, architectures with both 2- and 3-way junctions have a higher expressive capacity than subtrees with only 2-way junctions. On the hybrid dataset, under tight computational constraints, we found that trees with higher degrees of branching achieved higher accuracy rates. Unconstrained, however, they were prone to overfitting.

In dynamically-routed networks, early modules tended to have high accuracy rates, pushing difficult decisions downstream. Even without energy contraints, terminal modules specialized in detecting instances of certain classes of images. These classes were usually related (they either all came from MNIST or all came from CIFAR-10.) In networks with both 2- and 3-way junctions, branches specialized to an even greater extent. (See Fig. 2.5 and Fig. 2.6.)

2.5.5 Comparing specialized and adaptive networks

We trained a single actor network to classify images from the hybrid datset under various levels of computational constraints, using the approach described in section 2.4.6, sampling α_{cpt} randomly from the set mentioned in Fig. 2.8 for each training example. This network performed comparably to a collection of 8 actor nets trained with various static values of α_{cpt}, over a significant, central region of the accuracy/efficiency curve, with an approximately 8-fold reduction in memory consumption and training time.

2.5.6 Exploring the effects of the decision difficulty distribution

To probe the effect of the inference task's difficulty distribution on the performance of dynamically-routed networks, we trained networks to classify images from CIFAR-10, adjusting the classification task to vary the frequency of difficult decisions (see Fig. 2.9). We call these variants CIFAR-2—labelling images as "horse" or "other"—

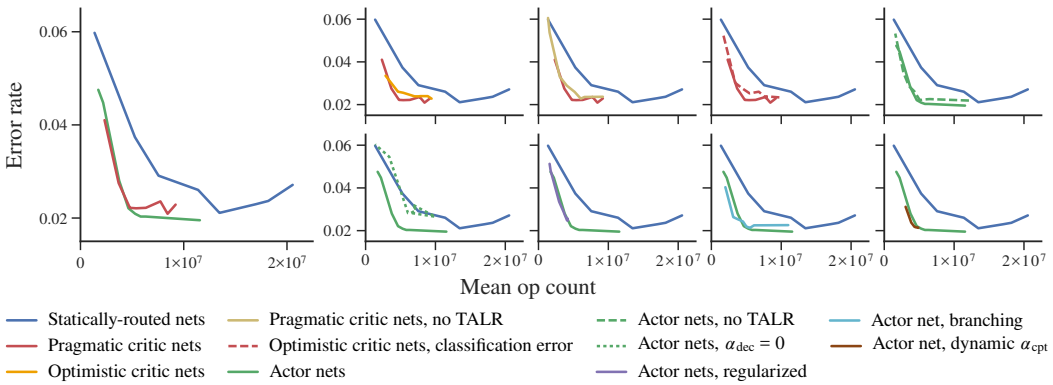

Figure 2.8: **Hybrid dataset performance.** Every point along the "statically-routed nets" curve corresponds to a network composed of the first n columns of the architecture illustrated in Fig. 2.3, for $1 \leq n \leq 8$. The points along the "actor net, dynamic α_{cpt}" curve correspond to a single network evaluated with various values of α_{cpt}, as described in section 2.4.6. The points along all other curves correspond to distinct networks, trained with different values of α_{cpt}. $\alpha_{\text{cpt}} \in \{0, 1 \times 10^{-9}, 2 \times 10^{-9}, 4 \times 10^{-9}, \ldots 6.4 \times 10^{-8}\}$.

and CIFAR-5—labelling images as “cat”, “dog”, “deer”, “horse”, or “other”. In this experiment, we compared actor networks (the best-performing networks from the first set of experiments) to architecture-matched statically-routed networks.

We found that dynamic routing was more beneficial when the task involved many low-difficulty decisions, allowing the network to route more data along shorter paths. While dynamic routing offered only a slight advantage on CIFAR-10, dynamically-routed networks achieved a higher peak accuracy rate on CIFAR-2 than statically-routed networks, at a third of the computational cost.

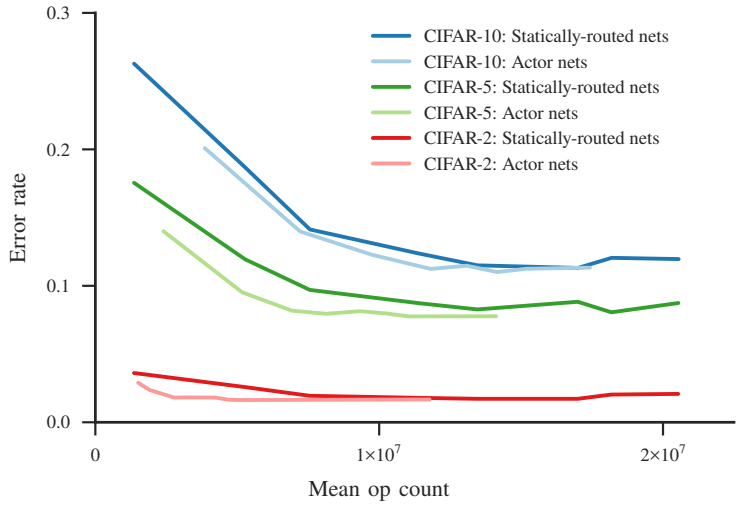

Figure 2.9: **Performance effects of the task difficulty distribution**, as described in section 2.5.6. The “statically-routed nets” and “actor nets” curves are drawn analogously to their counterparts in Fig. 2.8.

2.5.7 Exploring the effects of expressive capacity

To test whether dynamic routing is advantageous in higher-capacity settings, we trained actor networks and architecture-matched statically-routed networks to classify images from CIFAR-10, varying the width of the networks (see Fig. 2.10). Increasing the networks’ expressive capacity either increased or did not affect the relative advantage of dynamically-routed networks, suggesting that our approach may be applicable to more complicated tasks.

2.6 Discussion

Our results suggest that dynamically-routed networks trained under mild computational constraints can operate at least 2–3 times more efficiently than comparable statically-routed networks, without sacrificing performance. Additionally, despite their higher capacity, dynamically-routed networks may be less prone to overfitting.

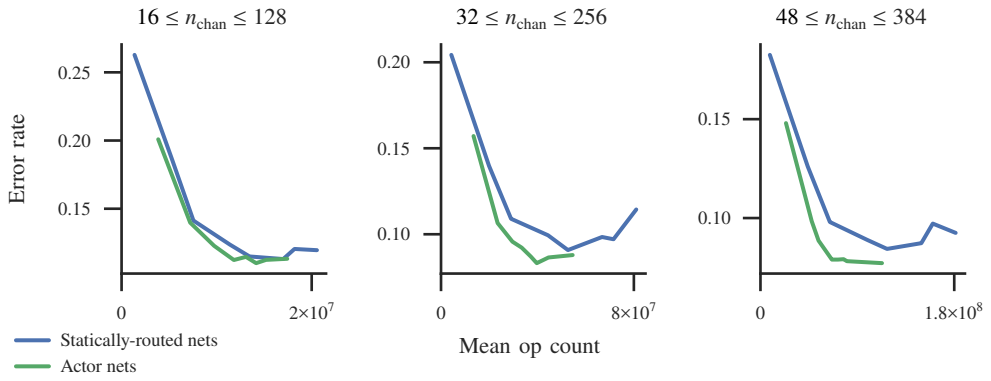

Figure 2.10: **Performance effects of network capacity**, training and testing on CIFAR-10. (Left) Networks with (subsets of) the architecture illustrated in Fig. 2.3. (Center) Networks otherwise identical to those presented in the left panel, with the number of output channels of every convolution operation multiplied by 2, and α_{cpt} divided by 4. (Right) Networks otherwise identical to those presented in the left panel, with the number of output channels of every convolution operation multiplied by 3, and α_{cpt} divided by 9.

When designing a multipath architecture, we suggest supporting early decision-making wherever possible, since cheap, simple routing networks seem to work well. In convolutional architectures, pyramidal modules appear to be reasonable sites for branching.

The actor strategy described in section 2.4.1 is generally an effective way to learn a routing policy. However, the pragmatic critic strategy described in section 2.4.2 may be better suited for very large networks (trained via decision sampling to conserve memory) or networks designed for applications with nonsmooth cost-of-inference functions—*e.g.* one in which α_{cpt} has units errors/operation. Adjusting learning rates to compensate for throughput variations, as described in section 2.4.5, may be useful when training particularly deep networks. And if the cost of computation is dynamic, a single network, trained with the procedure described in section 2.5.5, may still be sufficient.

While we tested our approach on tasks with some degree of difficulty variation, it is possible that dynamic routing is even more advantageous when performing more complex tasks. For example, video annotation may require specialized systems to recognize locations, objects, faces, human actions, and other entities of interest, but having every recognition system constantly operating may be extremely inefficient. A dynamic routing policy could fuse these systems, allowing them to share common components, and activate specialized components as necessary.

Another interesting topic for future research is growing and shrinking dynamically-routed networks during training. With such a network, it would not be necessary to specify an architecture. The network would instead take shape over the course of training, as computational contraints, memory contraints, and the data dictate.

References

[1] Emmanuel Bengio et al. “Conditional computation in neural networks for faster models”. In: *arXiv preprint arXiv:1511.06297* (2015).

[2] Samuel Bulo and Peter Kontschieder. “Neural decision forests for semantic image labelling”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 81–88.

[3] Zhaowei Cai, Mohammad Saberian, and Nuno Vasconcelos. “Learning complexity-aware cascades for deep pedestrian detection”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 3361–3369.

[4] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. “Fast and accurate deep network learning by exponential linear units (elus)”. In: *arXiv preprint arXiv:1511.07289* (2015).

[5] Ludovic Denoyer and Patrick Gallinari. “Deep sequential neural network”. In: *arXiv preprint arXiv:1410.0510* (2014).

[6] Alexey Dosovitskiy et al. “Flownet: Learning optical flow with convolutional networks”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2758–2766.

[7] Ross Girshick. “Fast r-cnn”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1440–1448.

[8] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks.” In: *Aistats*. Vol. 9. 2010, pp. 249–256.

[9] Melvyn A Goodale and A David Milner. “Separate visual pathways for perception and action”. In: *Trends in neurosciences* 15.1 (1992), pp. 20–25.

[10] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778.

[11] Tin Kam Ho. “Random decision forests”. In: *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*. Vol. 1. IEEE. 1995, pp. 278–282.

[12] Arthur E Hoerl and Robert W Kennard. “Ridge regression: Biased estimation for nonorthogonal problems”. In: *Technometrics* 12.1 (1970), pp. 55–67.

[13] Yani Ioannou et al. “Decision forests, convolutional networks and the models in-between”. In: *arXiv preprint arXiv:1603.01250* (2016).

[14] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *arXiv preprint arXiv:1502.03167* (2015).

[15] Tsung-Wei Ke, Michael Maire, and Stella X Yu. “Neural Multigrid”. In: *arXiv preprint arXiv:1611.07661* (2016).

[16] Peter Kontschieder et al. “Deep Neural Decision Forests”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1467–1475.

[17] Simon Kornblith et al. “A network for scene processing in the macaque temporal lobe”. In: *Neuron* 79.4 (2013), pp. 766–781.

[18] Alex Krizhevsky and Geoffrey Hinton. “Learning multiple layers of features from tiny images”. In: (2009).

[19] Yann LeCun, Corinna Cortes, and Christopher JC Burges. *The MNIST database of handwritten digits*. 1998.

[20] Haoxiang Li et al. “A convolutional neural network cascade for face detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 5325–5334.

[21] Min Lin, Qiang Chen, and Shuicheng Yan. “Network in network”. In: *arXiv preprint arXiv:1312.4400* (2013).

[22] Mason McGill and Pietro Perona. “Deciding how to decide: Dynamic routing in artificial neural networks”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 2363–2372.

[23] Sebastian Moeller, Winrich A Freiwald, and Doris Y Tsao. “Patches with links: a unified system for processing faces in the macaque temporal lobe”. In: *Science* 320.5881 (2008), pp. 1355–1359.

[24] Alejandro Newell, Kaiyu Yang, and Jia Deng. “Stacked hourglass networks for human pose estimation”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 483–499.

[25] Shaoqing Ren et al. “Faster R-CNN: Towards real-time object detection with region proposal networks”. In: *Advances in Neural Information Processing Systems*. 2015, pp. 91–99.

[26] Leonid I Rudin, Stanley Osher, and Emad Fatemi. “Nonlinear total variation based noise removal algorithms”. In: *Physica D: Nonlinear Phenomena* 60.1-4 (1992), pp. 259–268.

[27] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[28] JA Sirat and JP Nadal. “Neural trees: a new tool for classification”. In: *Network: Computation in Neural Systems* 1.4 (1990), pp. 423–438.

[29] Rupesh K Srivastava, Klaus Greff, and Jürgen Schmidhuber. “Training very deep networks”. In: *Advances in neural information processing systems*. 2015, pp. 2377–2385.

[30] Robert Tibshirani. “Regression shrinkage and selection via the lasso”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1996), pp. 267–288.

[31] Paul E Utgoff. “Perceptron trees: A case study in hybrid concept representations”. In: *Connection Science* 1.4 (1989), pp. 377–391.

[32] Paul Viola, Michael J Jones, and Daniel Snow. “Detecting pedestrians using patterns of motion and appearance”. In: *International Journal of Computer Vision* 63.2 (2005), pp. 153–161.

[33] Erjin Zhou et al. “Extensive facial landmark localization with coarse-to-fine convolutional network cascade”. In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2013, pp. 386–391.

Chapter 3

EVOLVING NEURAL NETWORKS FOR PREDATOR AVOIDANCE

Forward

This chapter discusses computational experiments in which we simulated the evolution of predator-avoidance strategies in large populations of agents performing a sequential decision task inspired by behavior observed in fruit flies. While performing this task, an agent must continually decide whether to flee or forage based on ambiguous evidence as to whether there are hidden predators in the foraging area. Our virtual agent populations successfully evolved genetically-encoded neural networks that could balance food collection with self-preservation, coming within 1% of the fitness score obtained using a strategy generated by a reinforcement-learning algorithm with direct access to the environment's statistics. The evolved agents also exhibited behavior observed in fruit flies, including (1) often ignoring individual predator cues, (2) fleeing in response to a sequence of many predator cues observed in quick succession, and (3) tolerating greater levels of risk if they were old or hungry. Additionally, we found that high-fitness networks were discovered more quickly in environments where subpopulations could remain isolated from each other for many generations, and populations that evolved in highly dynamic environments adapted more quickly to change. The work presented in this chapter was done in collaboration with Pietro Perona and has not yet been published.

3.1 Introduction

Predators generally don't want their presence to be known until it's too late for their prey to escape, so prey animals often find themselves in ambiguous situations, where they must assess their environment's safety based on limited information. In these situations, a prey animal can either continue whatever it was doing, and risk being eaten, or it can react—by freezing, hiding, or running away—and potentially forego foraging opportunities or waste valuable energy. And the better it is at navigating this ambiguity, the better it will be at striking a balance between safety and metabolic

efficiency.

However, while animals can tune many aspects of their behavior based on feedback from their environment, an individual animal can't learn from being eaten. Instead, strategies for responding to predator-associated sounds, odors, vibrations, and visual cues must be learned at the population level, through natural selection.

What does this evolution process look like as it's taking place? And what factors influence how long it takes to converge to something resembling an optimal strategy? To start to answer these questions, we constructed simulation environments inspired by the ecological niche occupied by the fruit fly—an animal whose predator-avoidance strategy has been studied quantitatively. In these environments, virtual foragers must make stay-or-flee decisions based on imperfect information, and a forager's fitness is defined as the total amount of time it spends foraging over the course of its lifetime. To establish a performance baseline, we constructed "ideal observer" foragers, which compute the probability that a predator is present using direct knowledge of the environment's dynamics, and decide whether to flee based on an action-lookup table generated using reinforcement learning and data from 500 billion forager lifetimes. We then simulated the evolution of foragers that make stay-or-flee decisions using

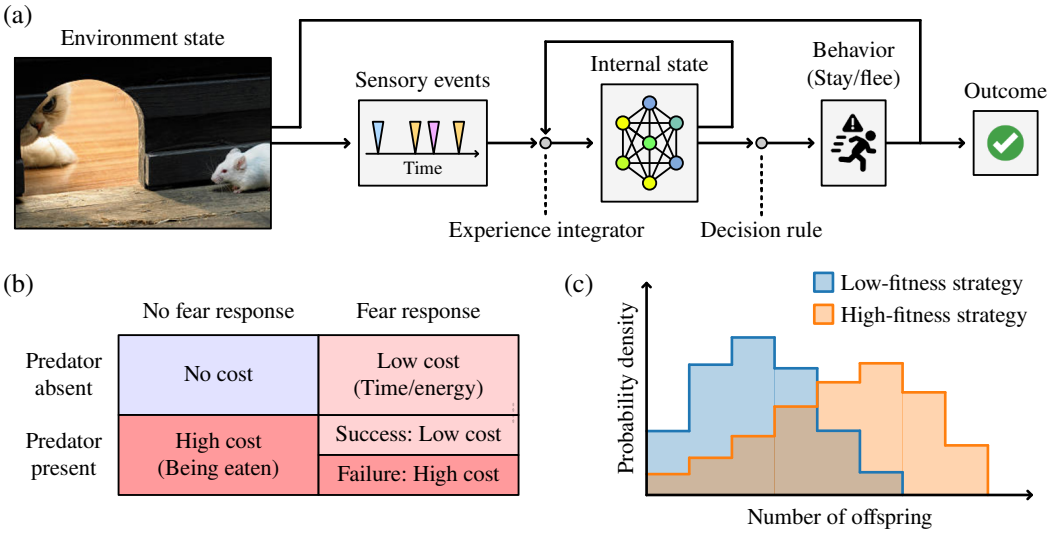

Figure 3.1: **Avoiding predation in the face of ambiguity**. (a) Prey animals integrate sensory cues to form an internal representation of their environment, which they use to decide whether to stay or flee. (b) Both staying and fleeing have potential costs. (c) For a given environment, some strategies will be more suitable than others. Luck can have a big influence on an individual animal's reproductive success, but a prey animal employing a higher-fitness predator-avoidance strategy will, on average, spend more time foraging over the course of its life and produce more offspring.

small neural networks, compared these to both the ideal observer foragers and real fruit flies, and investigated how changes to the simulation affect the evolution process.

We found that the evolved foragers could attain over 99% of the fitness of the ideal observer foragers, and that both recapitulated predator-avoidance behavior observed in fruit flies. Additionally, we found that high-fitness networks were discovered more quickly in environments where subpopulations move around slowly and can remain isolated from each other for many generations. In these environments, subpopulations with incompatible genotypes compete for territory, with bands of low-fitness hybrids forming at territorial boundaries. We also found that populations that evolved in highly dynamic environments adapted to change faster than populations that evolved in static or infrequently changing environments.

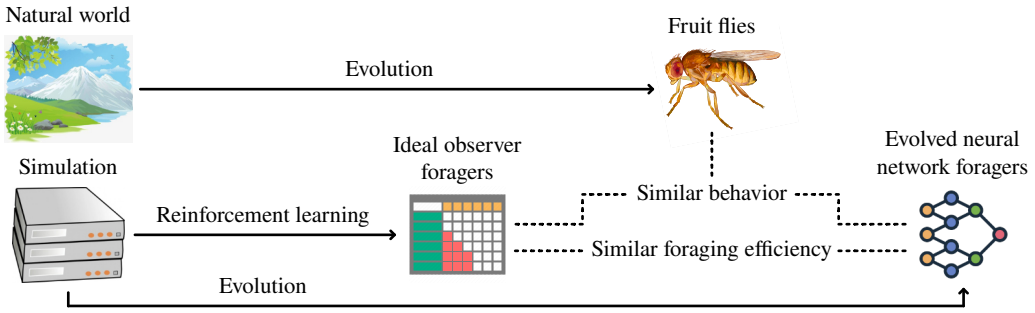

Figure 3.2: **A high-level overview of our modeling strategy**. Animals evolve sensorimotor mechanisms for avoiding predation. To better understand how these mechanisms come about, we defined a sequential decision-making task inspired by behavior observed in fruit flies (section 3.3 and section 3.4), and simulated the evolution of virtual foragers performing this task using genetically encoded neural networks (section 3.7–section 3.9). These networks recapitulated predator-avoidance behavior observed in fruit flies (section 3.12), and approximately matched the fitness of "ideal observer" foragers with direct access to the statistics of the environment (section 3.5 and section 3.6). After successfully evolving high-fitness neural network foragers, we investigated how changes to the simulation affect the evolution process (section 3.10 and section 3.11).

3.2 Related work

Predator-escape behavior has been studied in many nonhuman animals, including fruit flies [4, 11, 24, 1, 22], mice [30, 28], zebrafish [8], fiddler crabs [14, 13], and finches [23]. In terms of deriving optimal prey-animal behavior, [29] explored the cost/benefit calculus around when to flee in response to a detected-but-distant predator, and work in signal-detection theory [16, 21, 3] has addressed how to optimally integrate sparse perceptual information over time. Our sequential decision-making task can

be thought of as a variant of the task described in [29], with ambiguous predator-cue sequences substituted in place of distance, and our ideal-observer foragers combine optimal signal detection with a learned action policy.

We don't know of any other work specifically involving the evolution of neural networks for predator avoidance, but evolutionary algorithms have been used to optimize neural networks in many other contexts, dating back to at least 1989 [15, 25, 6, 17]. Of particular note is the AntFarm project [5], in which researchers simulated the evolution of cooperative foraging behavior using an early massively parallel computer. More recently, algorithms with varying degrees of resemblance to biological evolution—for example, algorithms incorporating selection and mutation, but not recombination—have been used to optimize relatively large neural networks for machine learning applications [9, 20]. (Table II in [10] provides a useful overview.) In addition to prior work on neural network optimization, our evolution simulations also took inspiration from the mathematical geneticist Seawall Wright, who argued that structured populations with many partially isolated subpopulations play an important role in animal evolution [26, 27, 7].

3.3 Predator avoidance in fruit flies

Fruit flies will freeze or flee in response to a series of expanding shadows cast from above [11]. But they often ignore expanding shadows they see in isolation. And a fly's proclivity to react with a fear response can change over time. For a given stimulus sequence, a satiated fly will be more likely to freeze or flee than a hungry fly [11], and a younger fly will be more likely to freeze or flee than an older fly [22].

While not all behaviors are necessarily adaptive, the way flies react to expanding shadows is consistent with a rational predator-avoidance strategy. An individual shadow may be caused by a falling leaf or a non-predating animal walking by, but a series of shadows is stronger evidence that a predator is actively hunting in the area, and may consequently be more worthy of a response. Considering the effect of hunger, abandoning the search for food in order to flee is a higher price to pay for a hungry fly than it is for a satiated fly, so it may make sense for hungry flies to tolerate a higher risk of predation. And considering the effect of age, the expected cost of predation—in terms of time spent alive and potentially reproducing—is lower for older flies than it is for younger flies, so it may make sense for them to accept a higher predation risk as well.

3.4 The life of a virtual forager

In our simulations, each forager's adult life unfolds as a continuous-time partially observable Markov decision process (POMDP) [2]. Each forager's state consists of its age $\in [0, \infty)$, a hunger level $\in [0, 1]$, and a discrete state class, which can either be "wandering", "safe foraging", "dangerous foraging", or "deceased". Foragers start out in the wandering state class, at age zero, with a hunger level of zero.

While in the wandering state class, a forager may find a foraging area free of predators, and enter the safe foraging state class. It may also find a foraging area with predators nearby, and enter the dangerous foraging state class. And it may encounter a predator, and enter the deceased state class. In our simulations, these state transitions occur at a rate of 0.008, 0.002, and 0.0001 transitions per second respectively. Also, a forager's hunger level will increase at a rate of 0.001 units per second spent wandering, and if it reaches 1, the forager will die of starvation.

In the safe foraging state class, predator cues occur at a rate of 0.03 per second, and transitions to the dangerous foraging state class—indicating that a predator has entered the area—occur at a rate of 0.002 per second. Forced transitions to the wandering state class—due to the forager getting tired, running out of food, or seeking shelter from the rain, for example—occur at a rate of 0.01 per second. And the forager's hunger level decreases at a rate of 0.006 units per second spent foraging, to a minimum of 0.

In the dangerous foraging state class, predator cues occur at a rate of 0.3 per second, transitions to the safe foraging state class occur at a rate of 0.008 per second, and forced transitions to the wandering state class occur at a rate of 0.01 per second. Additionally, transitions to the deceased state class, as a result of predation, occur at a rate of 0.01 per second. A forager's hunger level decreases in the dangerous foraging state class just as it would in the safe foraging state class.

Foragers can sense their age, their hunger level, and when they start foraging, but they can't directly determine whether a predator is present. If they suspect they are in danger while foraging, they can decide to flee back to the wandering state at any time. In addition to being eaten and starving, foragers can die of old age in any non-deceased state. Upon initialization, each forager's maximum lifespan is sampled from a Weibull distribution [12] with a median of 3 hours and shape parameter of 3 (meaning that the instantaneous natural death rate scales with the square of the forager's age).

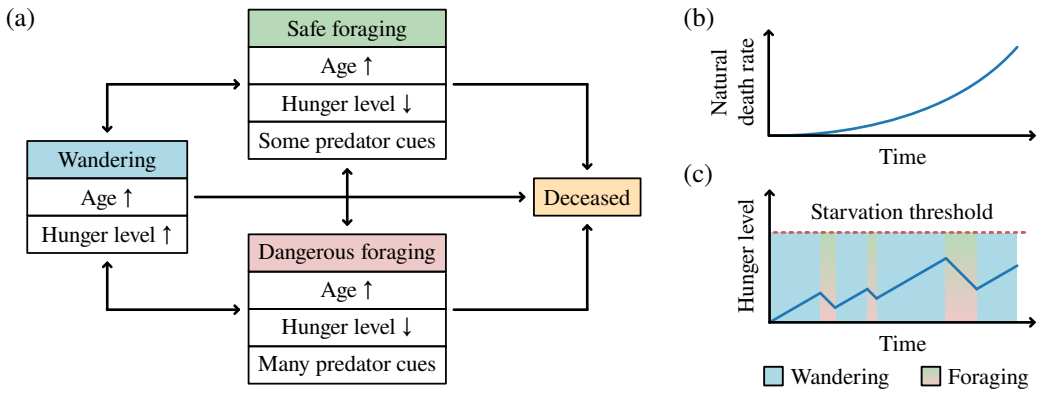

Figure 3.3: **Our forager-environment interaction model**. (a) Foragers wander through their environment until they come across a foraging area. Foraging reduces a forager's hunger level and increases the odds that they will come across a potential mate, but also exposes them to the risk of encountering a predator. Predator cues occur at a higher rate when a predator is present, and foragers can choose to flee from a foraging area if they suspect they are in danger. (b) Foragers can die of natural causes at any time, and their instantaneous natural-cause death rate is proportional to the square of their age. (c) A forager's hunger level increases when it is wandering and decreases when it is foraging. If it reaches the starvation threshold, the forager will die.

3.5 Tracking state class probabilities over time

Now that we've described the predator-avoidance task that we want to evolve neural networks to perform, we'll consider how an ideal observer could track the conditional probability distribution over the set of foraging state classes over time, based on the predator cues it observes. This computation—visualized in Fig. 3.4—turns out to be simple in the sense that the observer only needs to keep track of its instantaneous belief, as opposed to the full sequence of predator cues and inter-cue intervals that led to that belief. We'll start with a few definitions:

- Let $\mathcal{S}$ be the set of foraging state classes: {SafeForaging, DangerousForaging}.
- Let $\mathcal{O}$ be a set containing the observable events types: WanderingTransition, PredatorCue, and PredatorAttack. (A WanderingTransition event is an involuntary transition back to the wandering state class.)
- Let $\tau(s; \text{Wandering}) \in [0, \infty)$, for any state class $s \in \mathcal{S}$, be the rate at which foragers transition from the wandering state class to s.
- Let $\tau(s'; s) \in [0, \infty)$, for any state class pair $(s, s') \in \mathcal{S} \times \mathcal{S}$, be the rate at which foragers transition from s to s', if s and s' are distinct, or zero, if they

are identical.

- And let $\varepsilon(o; s) \in [0, \infty)$, for any observable event type $o \in \mathcal{O}$ and state class $s \in \mathcal{S}$, be the rate at which observable events of type o occur in state class s.

We'll also define a set of mathematical objects called "observation histories". An observation history is a sequence of event types and nonnegative real numbers representing inter-event durations, ordered chronologically. For example, the observation history (t_1, o, t_2) indicates that no events were observed for t_1 seconds, after which an event of type o was observed, and then no events were observed for the next t_2 seconds.

3.5.1 Computing initial state class probabilities

Let $p_{s|\mathcal{H}}(s; \mathcal{H}) \in [0, 1]$, for any state class $s \in \mathcal{S}$ and observation history $\mathcal{H}$, be the probability that a forager is in state class s after observing $\mathcal{H}$. For an empty history (*i.e.*, at the beginning of a foraging session),

$$p_{s|\mathcal{H}}(s; ()) = \frac{\tau(s; \text{Wandering})}{\sum_{s' \in \mathcal{S}} \tau(s'; \text{Wandering})}, \tag{3.1}$$

for all state classes $s \in \mathcal{S}$, since foraging sessions are always initiated via a transition from the wandering state class.

3.5.2 Updating state class probabilities between observable events

Between event observations, the conditional probability of being in any given state class may change over time for two reasons. First, observing an absence of events is informative, and will shift probability mass toward state classes that cause observable events to occur less frequently. And second, knowing that time has passed will shift probability mass toward state classes that are more likely to be transitioned into than away from, given the state class distribution at the beginning of the interval. Taking both of these factors into account, the between-observation time derivative of the probability that the forager is in a state class s after observing a history $\mathcal{H}$ is

$$\begin{aligned} \frac{d}{dt} p_{s|\mathcal{H}}(s; \mathcal{H}) = &\left(\hat{\varepsilon}_{\text{total}}(\mathcal{H}) - \varepsilon_{\text{total}}(s) - \tau_{\text{from}}(s) \right) p_{s|\mathcal{H}}(s; \mathcal{H}) \\ &+ \sum_{s' \in \mathcal{S}} p_{s|\mathcal{H}}(s'; \mathcal{H})\, \tau(s; s'), \end{aligned} \tag{3.2}$$

where $\hat{\varepsilon}_{\text{total}}(\mathcal{H})$ is the expected total observable event rate, given the observation history $\mathcal{H}$,

$$\hat{\varepsilon}_{\text{total}}(\mathcal{H}) := \sum_{s \in \mathcal{S}} \sum_{o \in \mathcal{O}} p_{s|\mathcal{H}}(s; \mathcal{H})\, \varepsilon(o; s), \tag{3.3}$$

$\varepsilon_{\text{total}}(s)$ is the total observable event rate in state class s,

$$\varepsilon_{\text{total}}(s) := \sum_{o \in \mathcal{O}} \varepsilon(o; s), \tag{3.4}$$

and $\tau_{\text{from}}(s)$ is the rate at which foragers make unobserved transitions away from state class s,

$$\tau_{\text{from}}(s) := \sum_{s' \in \mathcal{S}} \tau(s'; s). \tag{3.5}$$

(See section 3.A for a step-by-step derivation.)

3.5.3 Updating state class probabilities when events are observed

When an observable event occurs, probability mass shifts toward state classes in which events of that type occur more frequently. When an observation of type o occurs, following an observation history $\mathcal{H}$, the updated probability that the forager is in a state class s is

$$p_{s|\mathcal{H}}(s; \mathcal{H}^\frown o) = \frac{p_{s|\mathcal{H}}(s; \mathcal{H})\, \varepsilon(o; s)}{\sum_{s' \in \mathcal{S}} p_{s|\mathcal{H}}(s'; \mathcal{H})\, \varepsilon(o; s')}. \tag{3.6}$$

(The symbol "$\frown$" is used to denote sequence-element concatenation. See section 3.B for a step-by-step derivation.)

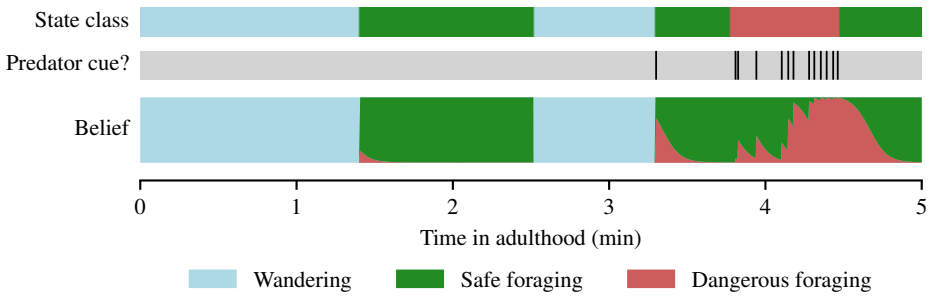

Figure 3.4: **Inferred state class probabilities over the course of an example life trajectory**. For each vertical slice of the "belief" trace, the height of each segment represents the probability that the forager is in the corresponding state class at the corresponding time, conditioned on the forager's observations.

3.6 Learning a stay-or-flee policy for ideal observer foragers

Our ideal observer foragers use the inference process described in the previous section to compute the probability that they are safe. But they also need an action policy to translate this probability into a stay-or-flee decision. In this section, we'll describe the stochastic iterative refinement process we used to constructed this policy. This process is visualized in Fig. 3.5, and the policy it ultimately yields is visualized in Fig. 3.6.

We started by defining a 64×64×64-bin grid in the space of (age, hunger level, safety probability) triples. The grid had opposing corners at $(0\text{min}, 0, 0)$ and $(240\text{min}, 1, 1)$, each age bin spanned a 4-minute timespan, and each hunger level and safety probability bin spanned $1/64$ units. We assigned each cell in the grid an initial flight inclination score (FIS) of 0, and constructed a stay-or-flee policy that applies the prescription

$$\text{action}(a, h, p) = \begin{cases} \text{Flee} & \text{if } \text{FIS}(\text{cell}(a, h, p)) > 0 \\ \text{Stay} & \text{otherwise,} \end{cases} \tag{3.7}$$

for all age, hunger level, and safety probability triples (a, h, p). Here $\text{cell}(a, h, p)$ is the grid cell containing the point (a, h, p), or the closest available cell, if (a, h, p) is outside the bounds of the grid.

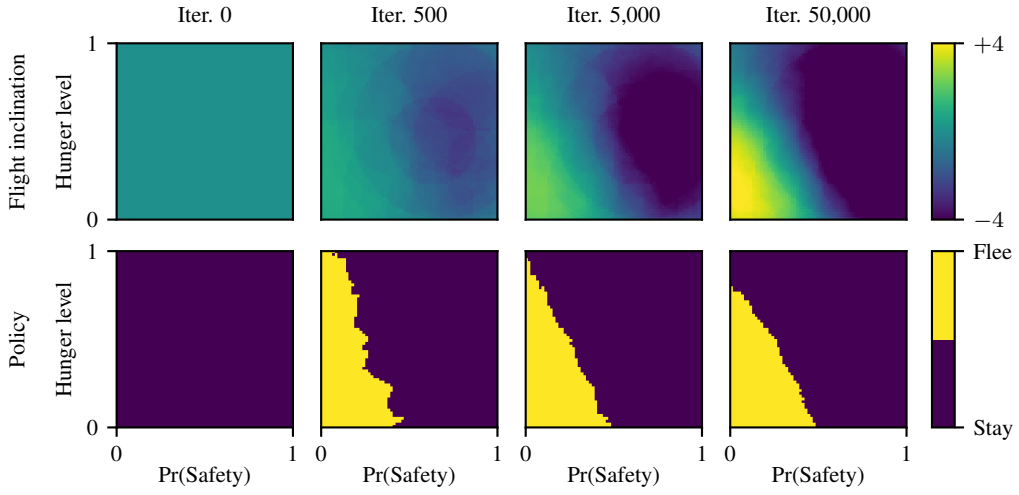

Figure 3.5: **Learning a lookup-table action policy**. This figure visualizes the policy-learning process using a 2-dimensional grid—omitting the age dimension—to make it easier to see how the FIS grid and action policy change over time. **Top:** Flight-inclination scores over course of the learning process. **Bottom:** Prescribed actions for each cell over the course of the learning process.

After constructing the initial FIS grid and initial policy, we refined them using a variant of simultaneous perturbation stochastic approximation (SPSA) optimization [18, 19]. We began every iteration of this optimization procedure by sampling an edit center from the set of cells in the grid, an edit radius from the set $\{1, 2, 4, 8, 16, 32\}$, and a perturbation from the range $[1/2, 3/2)$. Next, we generated two variants of the current FIS grid by modifying the flight-inclination scores assigned to the cells whose distance from the edit center was less than the edit radius. One variant was generated by adding the perturbation to their scores, and the other was generated by subtracting the perturbation from their scores. We then derived policies from the modified FIS grids using Equation 3.7, and computed the mean fitness score for each policy based on a sample of 1 million foragers.

These fitness scores were used to compute an estimate of the gradient of the fitness function with respect to the flight-inclination scores:

$$\hat{\mathbf{g}} := \frac{f_+ - f_-}{2\,\alpha}\,\mathbf{m}, \tag{3.8}$$

where f_+ and f_- are the mean fitness scores obtained using the FIS grid variants with raised and lowered flight-inclination scores, respectively, α is the perturbation, and the selection mask $\mathbf{m}$ is a vector whose i-th component is 1 if the i-th grid cell was modified and 0 otherwise. We generated flight-inclination scores for the next

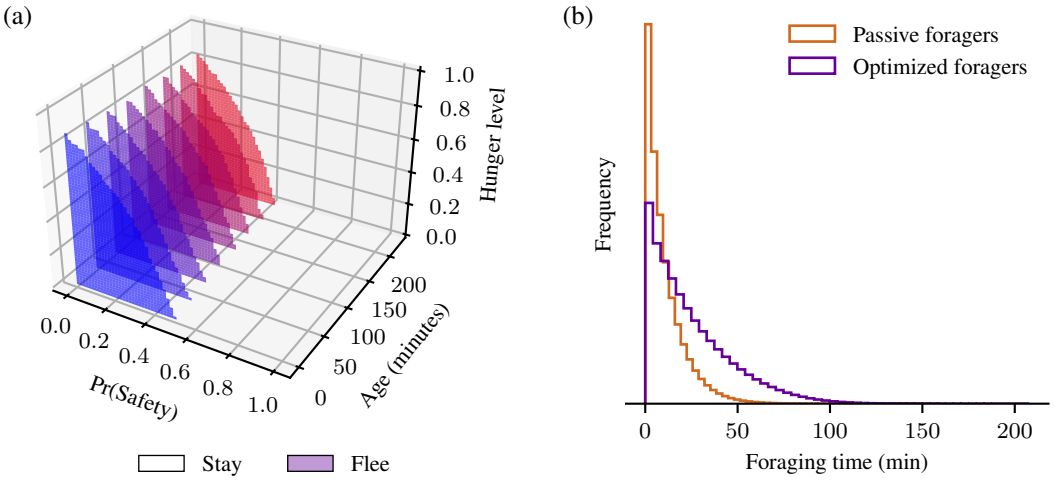

Figure 3.6: **The action policy learned after 250k refinement iterations**. (a) Stay/flee prescriptions for 8 of the 64 age slices. Shaded cells prescribe the "flee" action, and transparent cells prescribe the "stay" action. The shading color is varied between age slices to improve legibility. (b) Fitness histograms for foragers who never flee and foragers using the action policy visualized in (a).

iteration using the update rule

$$\mathbf{s}' = \mathbf{s} + \lambda\,\hat{\mathbf{g}}, \tag{3.9}$$

where $\mathbf{s}$ is a vector whose components are equal to the current scores, and λ is the current learning rate. The refinement process consisted of 250k iterations, and we used an initial learning rate of 10^{-5} and gradually decreased the learning rate to zero using cosine interpolation.

Foragers using the policy resulting from this optimization process have an expected lifetime foraging time of 1488 seconds—a 133% increase over the expected amount of time that foragers will forage if they ignore all predator cues and never flee (638 seconds). Foragers using this policy also behave in a way that qualitatively resembles the fruit fly predator-avoidance strategy discussed in section 3.3. For a given age and hunger level, they will flee if and only if their safety probability falls below a certain threshold. This means that they will sometimes ignore isolated predator cues, but rarely ignore a sequence of predator cues seen in rapid succession. And looking at the relationship between age, hunger level, and this escape threshold, both youth and satiation increase a forager's proclivity to flee.

3.7 Controlling foraging behavior with a network of neurons

In place of the Bayesian inference/lookup table controller described in the previous two sections, our evolved foragers make stay-or-flee decisions using a small neural network without any knowledge of the environment's statistics built in. The network consists of nodes—which can be interpreted as either representing individual neurons or homogenous populations of neurons—and directed connections between them. Nodes communicate to their targets via nonnegative continuous-valued firing rates that change over time.

Each forager's neural network has four input nodes: an age sensor, a hunger level sensor, a predator cue sensor, and a node that maintains a constant firing rate of 1, to use as a bias. The age sensor's firing rate is equal to the forager's age in hours. The hunger level sensor's firing rate is equal to the forager's hunger level. And the predator cue sensor has a firing rate of 1 if a predator cue was observed in the past 0.25 seconds, and 0 otherwise.

These inputs send signals to seven hidden nodes and a single output node. The hidden nodes are present to help the forager integrate information over time, and the output node controls the forager's behavior; when its firing rate is nonzero, the forager flees. Each of these downstream nodes has an excitement level $x(t)$, which changes over

time, and 25 genetically-encoded parameters: an initial excitement level x_{init}, a decay rate r, a firing threshold θ, 11 excitatory connection weights, $w_{\text{ex}}(1)..w_{\text{ex}}(11)$, and 11 inhibitory connection weights, $w_{\text{in}}(1)..w_{\text{in}}(11)$.

A hidden or output node's excitement level begins each foraging session at x_{init} and changes over time in response to its inputs. After initialization, the excitement level follows the dynamics

$$\frac{d}{dt}x(t) = -r\,x(t) + \begin{cases} \max(\mathbf{w}_{\text{net}} \cdot \mathbf{f}(t), 0) & \text{if } x(t) = 0 \\ \mathbf{w}_{\text{net}} \cdot \mathbf{f}(t) & \text{otherwise,} \end{cases} \tag{3.10}$$

where

$$\mathbf{w}_{\text{net}} := \begin{bmatrix} w_{\text{ex}}(1) \\ \vdots \\ w_{\text{ex}}(11) \end{bmatrix} - \begin{bmatrix} w_{\text{in}}(1) \\ \vdots \\ w_{\text{in}}(11) \end{bmatrix}, \tag{3.11}$$

and $\mathbf{f}(t)$ is a vector $\in \mathbb{R}^{11}$ whose components are equal to the firing rates of the node's inputs (the 4 input nodes and the 7 hidden nodes). The node's firing rate $f(t)$ is computed by applying a rectified sigmoid function to its excitement level:

$$f(t) = \frac{2}{1 + \exp(-\max(x(t) - \theta, 0))} - 1. \tag{3.12}$$

3.8 Encoding neural network parameters genetically

Each forager's genome consists of 256 gene pairs, and each gene can either encode a trait, which alters the forager's neural network parameters, or encode the null effect, which leaves its parameters unchanged. If either gene at a locus encodes a trait, it will be expressed, but if both genes encode the same trait, it will only be expressed once. This means that, given a genome $\mathcal{G}$, the value of the forager's i-th parameter will be

$$\theta(i) = \sum_{(g_1,g_2)\in\mathcal{G}} \begin{cases} \text{effect}(g_1, i) + \text{effect}(g_2, i) & \text{if } g_1 \neq g_2 \\ \text{effect}(g_1, i) & \text{otherwise,} \end{cases} \tag{3.13}$$

where $\text{effect}(g, i)$ is the effect that gene g would have on $\theta(i)$ if no other genes were present.

When a mutation event creates a gene encoding a trait, the trait's effect is determined via a three-step process: sampling a target parameter group, sampling a target mask indicating which parameters in the group the trait will impact, and sampling an effect size. The target parameter group is selected randomly from the five network parameter groups—initial excitement levels, decay rates, firing thresholds, excitatory

weights, and inhibitory weights. Each entry in the target mask is sampled from a Bernoulli distribution with mean $1/4$, if the trait targets a small parameter group (initial excitement levels, decay rates, or firing thresholds), or $1/32$, if the trait targets a large parameter group (excitatory weights or inhibitory weights). And the effect size is drawn from an exponential distribution with mean $1/4$. When a trait is expressed, it increases the value of each targeted parameter by its effect size.

3.9 Evolving foragers

In our evolution simulations, foragers live, die, and reproduce in a 2-dimensional world divided into discrete grid cells. For a given generation, every cell in the grid has two properties: a population density and a set of foragers representing the genotype distribution within the cell. Cells are initialized with a population density of 1 and a set of foragers with null genes (meaning their neural network parameters will all be 0).

At each generation, every forager's life is simulated to obtain a fitness score, equal to the total amount of time the forager spent foraging, in seconds. These scores are then used to determine the population density map for the next generation, and sample parents for new representative foragers. The first step in this process is computing the relative rate at which each forager sires offspring in each cell in the grid. For a grid cell c and a forager f, this siring rate is defined

$$s(c, f) := \begin{cases} \left(1 - \gamma \left|\mathcal{N}(c)\right|\right) d(\text{cell}(f))\, \phi(f) & \text{if } \text{cell}(f) = c \\ \gamma\, d(\text{cell}(f))\, \phi(f) & \text{if } \text{cell}(f) \in \mathcal{N}(c) \\ 0 & \text{otherwise,} \end{cases} \tag{3.14}$$

where $\text{cell}(f)$ is the grid cell that the forager f belongs to, $d(\text{cell}(f))$ is the population density at that cell, $\phi(f)$ is the forager's fitness score, the neighbor set $\mathcal{N}(c)$ is the set of cells sharing a border with cell c, and the drift coefficient γ is a parameter $\in [0, 0.2]$ specific to the simulation. Next, we compute the total siring rate for each cell,

$$s_{\text{total}}(c) := \sum_{f \in \mathcal{F}} s(c, f), \tag{3.15}$$

where $\mathcal{F}$ is the set of foragers in the current generation. The population densities for the next generation are computed

$$d_{\text{next}}(c) = \frac{s_{\text{total}}(c)}{\text{mean}\left\{s_{\text{total}}(c') : c' \in \mathcal{C}\right\}}, \tag{3.16}$$

where $\mathcal{C}$ is the set of cells in the grid. And the parent foragers for each cell c are sampled independently, from the distribution

$$\text{Parent}(c) = \text{Categorical}\left\{ f \mapsto \frac{s(c,f)}{s_{\text{total}}(c)} \;:\; f \in \mathcal{F} \right\}. \tag{3.17}$$

After sampling two parents from the spawning cell or a neighboring cell, a new forager's genotype is constructed by inheriting one randomly selected gene per locus from each parent, and then possibly mutating some of the inherited genes. Each inherited gene has a 0.001% chance of mutating into a functional gene encoding a new trait, and a 0.001% chance of breaking and being replaced by a gene encoding the null effect. Except where otherwise specified, our simulations used a 32×32 grid with 512 representative foragers per cell per generation. The evolution process described in this section is illustrated in Fig. 3.7.

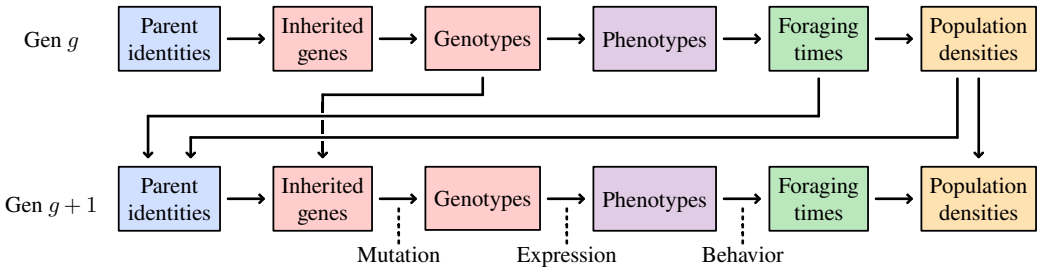

Figure 3.7: **The neural-network-controlled forager lifecycle**. Each forager's parents are sampled from the previous generation, and one (potentially mutated) gene at each locus is inherited from each parent. A forager's genome encodes its neural network controller, which influences how successfully it will forage while avoiding predators. And foragers that spend more time foraging will on average produce more offspring. Each generation, population density moves from grid cells with low mean foraging times to neighboring cells with higher mean foraging times. And, within a cell, foragers that spent more time foraging are more likely to be selected as parents.

3.10 Assessing the effect of subpopulation isolation

To assess the effect of allowing subpopulation to evolve independently and compete, we compared forager populations evolving on 1×1 grids to populations evolving on 32×32 grids with various drift coefficients. We ran five 1×1 simulations, and five 32×32 simulations for each drift coefficient $\in \{10^{-1}, 10^{-2}, ..., 10^{-6}\}$. All of these simulations ran for 25,000 generations and spawned a total of 2^{19} foragers per generation. (The simulations with 32×32 grids spawned 512 foragers per cell.)

We periodically computed cell fitness scores by simulating 100 foragers per genotype and averaging the individual foraging times we obtained within each cell. These

cell fitness scores were then used to compute population-level fitness scores via population-density-weighted averaging. Comparing the population fitness trajectories across conditions, it seems like allowing a variety of predator-avoidance strategies to evolve in parallel can be beneficial. After 1000 generations, foragers evolving on a 32×32 grid with a drift coefficient of 10^{-4} were on average 5.4 percentage points away from the ideal observer foragers' fitness level, while foragers evolving on a 1×1 grid were on average 57.1 points away. And this difference persisted throughout the simulations; after 25,000 generations, the average performance gaps were 0.9 and 2.5 percentage points, respectively. Fig. 3.8 visualizes subpopulations competing on

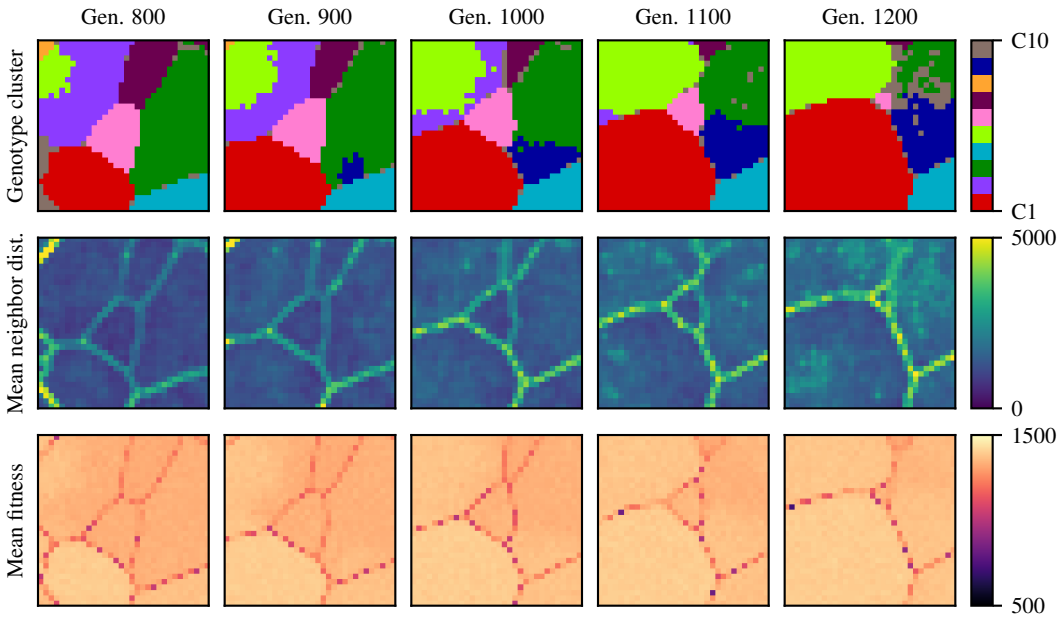

Figure 3.8: **Visualizations illustrating competition between subpopulations**. In each panel, each pixel represents a spatial location containing a semi-isolated subpopulation. **Top row:** Genetic clustering results over time, for an evolution simulation using a population drift coefficient of 0.01. We periodically stored occurrence counts for the 100 most common non-null genes in each grid cell. These counts were then used to create sparse description vectors for each cell across the five generations shown. For a given cell, the i-th component of its descriptor is equal to the occurrence count for gene i, if a count was stored for gene i, or 0, if it was not. The clusters shown were discovered by applying k-means clustering to these cell descriptors, using the clusters discovered at generation 800 to initialize the algorithm at generation 900, and initializing subsequent runs analgously. **Middle row:** Mean L1 distances between cells' descriptors and their direct neighbors' descriptors. (The descriptors of cells that share an edge.) **Bottom row:** Mean fitness scores for each cell, in foraging seconds, based on 100 lifetime simulations per genotype. Populations with higher fitness tend to expand.

a 32×32-cell grid, Fig. 3.9 visualizes the evolution trajectory of individual grid cells in simulations with different drift coefficients, and Fig. 3.10 shows how the grid size and drift coefficient affect the quality of the strategies that evolve.

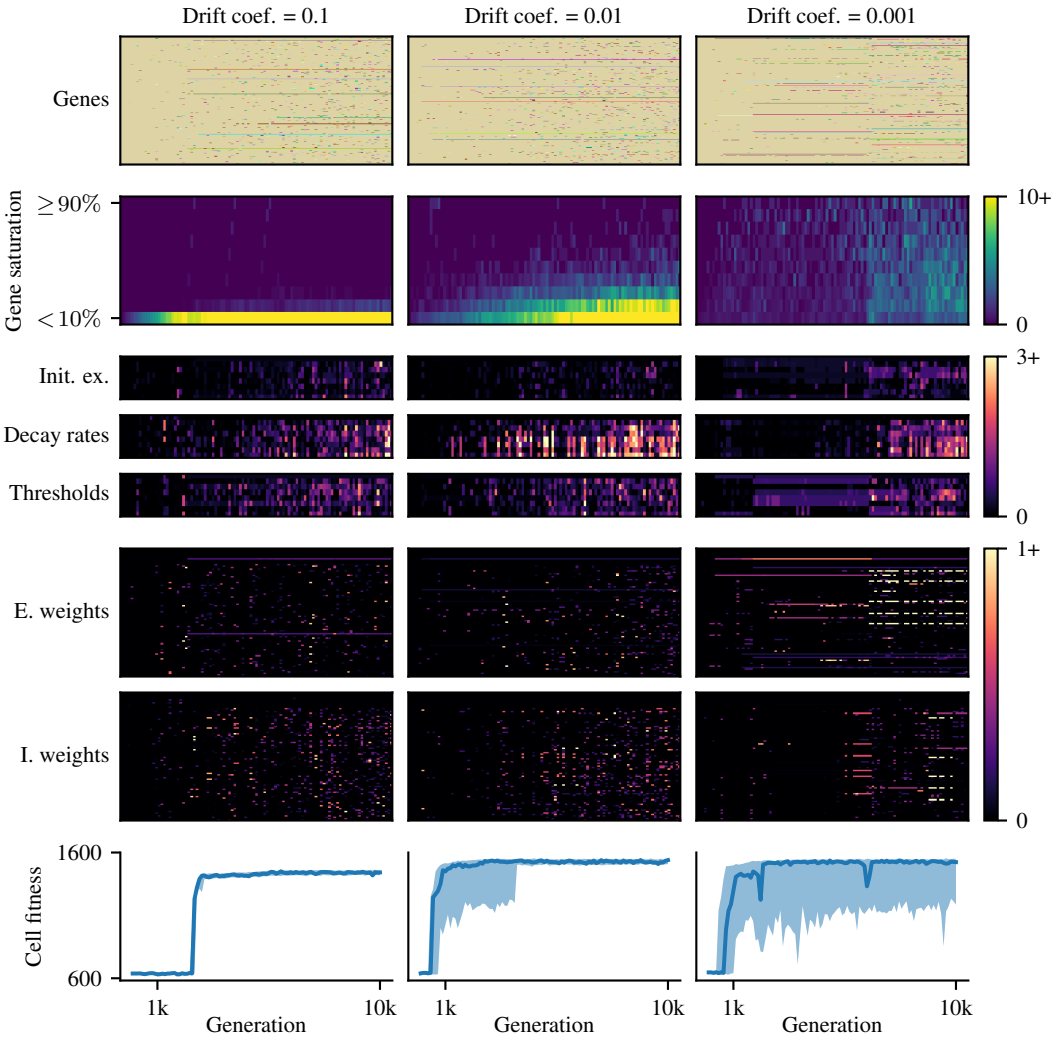

Figure 3.9: **Genotypes, phenotypes, and fitness levels over time** for a single cell in three simulations. **Row 1:** Genes from one strand of a random forager's genotype every 100 generations. Null genes are shown in beige and genes encoding traits are assigned random colors. **Row 2:** Gene saturation distributions for every 100 generations. The brightness of each frequency bin indicates the number of genes present per forager with that level of rarity/ubiquity. For example, if the bottom frequency bin (0–10% saturation) encodes the value 5.2, then foragers in the cell carry on average 5.2 genes with a cell-wide prevalence below 10%. **Rows 3–7:** Network parameters from a random forager, sampled once every 100 generations. **Row 8:** Fitness levels over time, in foraging seconds, based on 100 lifetime simulations per genotype. Within each chart, the shaded region indicates the range of fitness levels across the grid, and the curve indicates the fitness level for the cell.

3.11 Assessing the effect of environmental changes

After building an understanding of how populations of virtual foragers evolve in a static environment, we ran additional simulations with dynamic environments to see how the foragers would respond. We ran 125 simulations in which foragers were exposed to a set of "training" conditions presented in sequence over 800 generations, and then evaluated as they adapted to a new condition presented over the next 800 generations. We ran 25 simulations for each training set size $\in \{1, 2, 4, 8, 16\}$, and split the 800 training generations equally between the conditions in the training set. The conditions varied in their POMDP transition and emission rates, hunger increase and decrease rates, and median natural death time. Each of these parameters was sampled by multiplying its canonical value from section 3.4 by a scaling factor drawn (per-parameter) from a lognormal distribution with a log standard deviation of 0.4.

We found that populations exposed to more training conditions adapted to the evaluation environment more quickly, even though the number of training generations was held constant across populations. Repeating the experiment using a log standard deviation of 0.6, the qualitative result was the same, and the performance difference between populations exposed to large and small training sets was slightly more

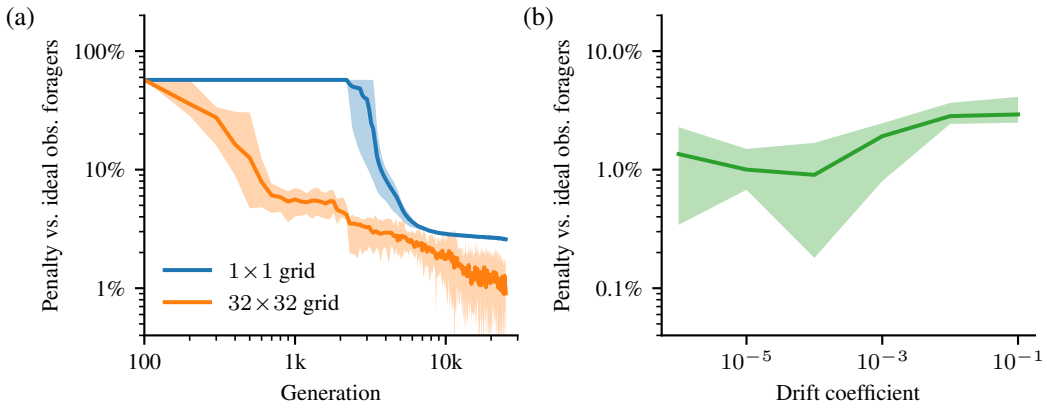

Figure 3.10: **Fitness trajectories for environments with different degrees of subpopulation isolation**. (a) Fitness trajectories for foragers evolving in a single-cell grid and foragers evolving in a 32×32 grid with a drift coefficient of 10^{-4}. We ran five 25,000-generation simulations in each condition, each with 2^{19} foragers spawned per generation. Cell fitness scores were computed every 100 generations—based on 100 lifetime simulations per genotype—and then used to compute population fitness scores via population-density-weighted averaging. The shaded regions indicate the range of these population fitness scores across simulations, and the curves indicate their averages. (b) Fitness levels at generation 25,000 for evolution simulations with a 32×32 grid and differing diffusion coefficients. As in (a), the shaded regions indicate population fitness ranges across five simulations, and the curves indicate averages.

pronounced. Fig. 3.11 summarizes our observations from all 250 of these simulations.

3.12 Comparing evolved foragers to fruit flies

To get a first-order understanding of the predator-avoidance strategies our evolved foragers were using, we examined the situations in which they decided to flee. We collected flight events across 1000 lifetimes for 100 forager genotypes. Each of these genotypes was sampled from the 25.000th generation of a simulation that used a 32×32 grid, used a drift coefficient of 10^{-4}, and spawned 512 representative foragers per cell per generation. Based on a qualitative assessment, it was clear that, like fruit flies, our evolved foragers did not flee in response to every predator cue, but usually fled in response to a series of predator cues in short succession.

Understanding the more subtle effects of age and hunger level, however, required quantitative analysis. With this in mind, we computed the correlation coefficients between age, hunger level, and predation risk (the probability the ideal observer described in 3.5 would assign to the dangerous foraging state class) immediately preceeding flight events. Overall, the results were consistent with the aspects of fruit fly behavior exhibited by the ideal observers foragers. The age/predation risk coefficients were all positive, indicating that older evolved foragers were more

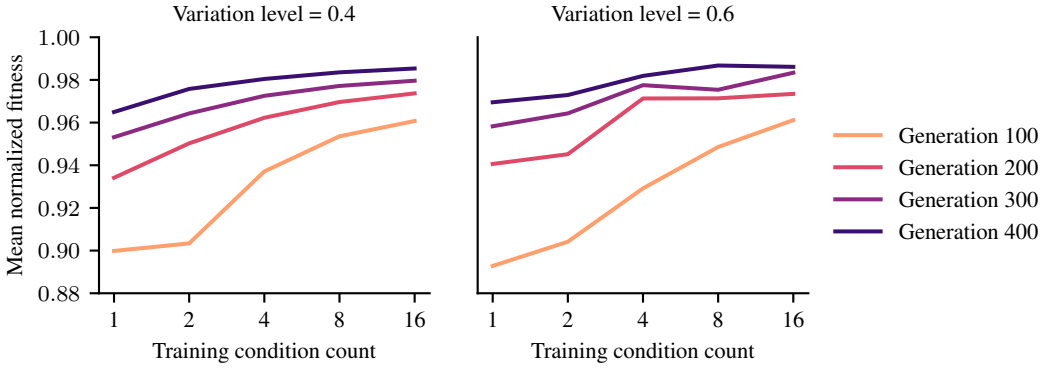

Figure 3.11: **Fitness trajectories for populations exposed to different levels of environmental change**. We exposed populations of foragers to a series of environment configurations over 800 "training" generations, and then assessed how they adapted to a new configuration over an additional 800 "test" generations. Five populations were simulated per test configuration, one for each training set size $\in \{1, 2, 4, 8, 16\}$. We periodically computed population fitness scores during each run, and then computed the ratio between each of these scores and the best final fitness score obtained on the test condition across the competing populations. We ran 25 simulations per (variation level, training set size) pair, and each curve vertex is located at the geometric mean of 25 of these fitness-score ratios. (See section 3.11 for details.)

risk-tolerant. And the hunger level/predation risk coefficients were all positive as well, indicating that the same was true for hungry evolved foragers. Fig. 3.12 contains these coefficient distributions rendered as histograms, along with input, output, and internal state traces from an individual evolved forager.

3.13 Discussion

We defined an ethologically inspired sequential decision-making task which requires virtual prey animals to balance foraging and predator avoidance in ambiguous situations. And we showed that two very different approaches to learning to perform this task—evolving neural networks, and using reinforcement learning to generate an action policy to pair with probabilistic inference—yielded solutions that aligned with the behavior of the species that inspired the task: fruit flies. We then used this task to better understand what the evolution of an adaptive behavior looks like as it unfolds over time, and the factors that influence how quickly it converges. We found that, in many cases, evolution was more successful in environments where subpopulations experience some degree of isolation and competing strategies can develop in parallel. And we found that frequently changing environments prepared populations better for subsequent changes than static environments.

Simulating populations of hundreds of thousands of neural networks evolving over tens of thousands of generations has only recently become feasible using commodity hardware, and there are still significant open questions about what to expect at this scale. For example, we probably have many things to learn about the relative advantages and disadvantages of different ways of mapping genotypes to network parameters. Our foragers' genes influenced parameters independently, but they could instead have been organized hierarchically, with some genes enabling or disabling the effects of others. Another subject for investigation is whether and/or when the recombination step in the evolution process introduces a bias toward networks with modular structure, allowing subpopulations to adopt genes disrupting part of the network while other parts continue to function. And there are also open questions around what kind of knowledge and behavior can be encoded genetically, and what must be learned. We hope our work can inform future investigations into these topics and many others.

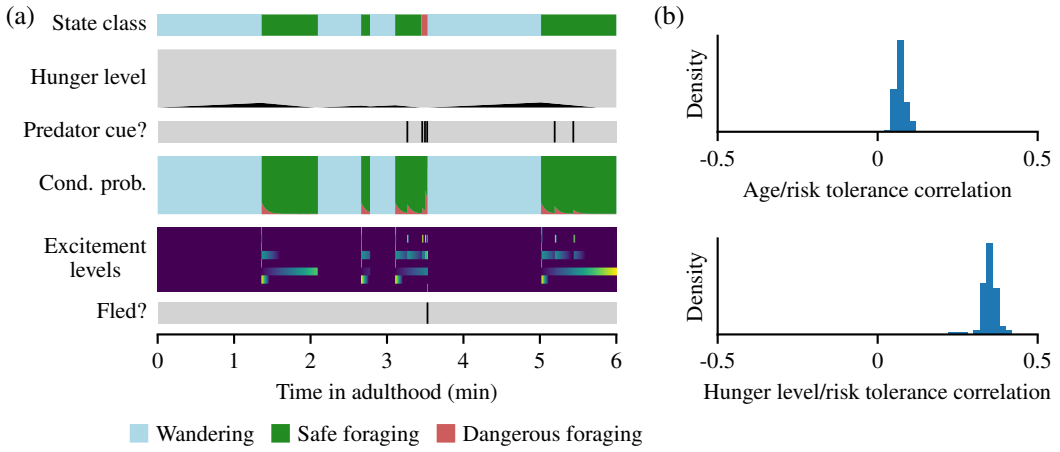

Figure 3.12: **Analyzing the behavior of evolved foragers**. (a) Forager and environment state traces from the first 6 minutes of a neural-network-controlled forager's life. **Row 1:** The hidden true state class. **Row 2:** The forager's hunger level, on a scale from 0 (completely gray) to 1 (completely black). **Row 3:** Lines indicating when predator cues were observed. **Row 4:** The conditional state class probabilities an ideal observer forager would compute, given the same observation history. **Row 5:** The forager's hidden node and output node excitement levels; brighter regions correspond to higher excitement levels. **Row 6:** Lines indicating when the forager fled. (b) Distributions of correlation coefficients relating a neural-network-controlled foragers' age and hunger level to its proclivity to flee. **Top:** Correlations between forager age and dangerous-foraging probability (the red-filled curve in row four of (a)) immediately preceeding flight events, computed across all flight events occurring in 1000 simulated lifetimes per genotype, for 100 genotypes sampled from a population. **Bottom:** The analogous correlation histogram, substituting hunger level for age. (See section 3.12 for details.)

3.A Derivation for Equation 3.2

Between event observations, the conditional probability of being in any given state class may change over time for two reasons. First, observing an absence of events is informative, and may shift probability mass toward state classes that cause observable events to occur less frequently. And second, knowing that time has passed may shift probability mass toward state classes that are more likely to be transitioned into than away from, given the state class distribution at the beginning of the interval.

Translating this into mathematical language, for any state class $s \in \mathcal{S}$, and observation history $\mathcal{H}$ whose final element is the inter-event duration t,

$$\frac{d}{dt} p_{s|\mathcal{H}}(s; \mathcal{H}) = c_{\text{ob}}(s; \mathcal{H}) + c_{\text{ex}}(s; \mathcal{H}), \tag{3.18}$$

where the observation component $c_{\text{ob}}(s; \mathcal{H})$ is what the derivative of $p_{s|\mathcal{H}}(s; \mathcal{H})$ with respect to time would be if the environment's hidden behavior was disabled,

$$c_{\text{ob}}(s; \mathcal{H}) := \lim_{\Delta t \to 0} \frac{p_{s|\mathcal{H}}(s; \mathcal{H}^\frown \text{DisableHB}^\frown \Delta t) - p_{s|\mathcal{H}}(s; \mathcal{H})}{\Delta t}, \tag{3.19}$$

and the extrapolation component $c_{\text{ex}}(s; \mathcal{H})$ is what the derivative of $p_{s|\mathcal{H}}(s; \mathcal{H})$ with respect to time would be if the environment's observable behavior was disabled,

$$c_{\text{ex}}(s; \mathcal{H}) := \lim_{\Delta t \to 0} \frac{p_{s|\mathcal{H}}(s; \mathcal{H}^\frown \text{DisableOB}^\frown \Delta t) - p_{s|\mathcal{H}}(s; \mathcal{H})}{\Delta t}. \tag{3.20}$$

Looking at the definition of $c_{\text{ob}}(s; \mathcal{H})$, we can expand $p_{s|\mathcal{H}}(s; \mathcal{H}^\frown \text{DisableHB}^\frown \Delta t)$ using Bayes rule:

$$c_{\text{ob}}(s; \mathcal{H}) = \lim_{\Delta t \to 0} \left(\frac{p_{s|\mathcal{H}}(s; \mathcal{H})\ \text{pois}(0; \Delta t\, \varepsilon_{\text{total}}(s))}{\Delta t \sum_{s' \in \mathcal{S}} p_{s|\mathcal{H}}(s'; \mathcal{H})\ \text{pois}(0; \Delta t\, \varepsilon_{\text{total}}(s'))} - \frac{p_{s|\mathcal{H}}(s; \mathcal{H})}{\Delta t} \right), \tag{3.21}$$

where $\varepsilon_{\text{total}}(s)$ is the total event occurrence rate when the system is in state s. Expanding the Poisson probability mass function,

$$c_{\text{ob}}(s; \mathcal{H}) = \lim_{\Delta t \to 0} \left(\frac{p_{s|\mathcal{H}}(s; \mathcal{H}) \exp(-\Delta t\, \varepsilon_{\text{total}}(s))}{\Delta t \sum_{s' \in \mathcal{S}} p_{s|\mathcal{H}}(s'; \mathcal{H}) \exp(-\Delta t\, \varepsilon_{\text{total}}(s'))} - \frac{p_{s|\mathcal{H}}(s; \mathcal{H})}{\Delta t} \right). \tag{3.22}$$

Taking the limit by substituting an infinitessimal value dt in place of Δt,

$$c_{\text{ob}}(s; \mathcal{H}) = \frac{p_{s|\mathcal{H}}(s; \mathcal{H}) \exp(-dt\, \varepsilon_{\text{total}}(s))}{dt \sum_{s' \in \mathcal{S}} p_{s|\mathcal{H}}(s'; \mathcal{H}) \exp(-dt\, \varepsilon_{\text{total}}(s'))} - \frac{p_{s|\mathcal{H}}(s; \mathcal{H})}{dt}. \tag{3.23}$$

Expanding the exponentials using the Taylor series definition and then dropping the terms with second-order-or-higher infinitessimals,

$$c_{\text{ob}}(s;\mathcal{H}) = \frac{p_{s|\mathcal{H}}(s;\mathcal{H})\,(1 - dt\,\varepsilon_{\text{total}}(s))}{dt\sum_{s'\in\mathcal{S}} p_{s|\mathcal{H}}(s';\mathcal{H})\,(1 - dt\,\varepsilon_{\text{total}}(s'))} - \frac{p_{s|\mathcal{H}}(s;\mathcal{H})}{dt}. \tag{3.24}$$

Multiplying,

$$c_{\text{ob}}(s;\mathcal{H}) = \frac{p_{s|\mathcal{H}}(s;\mathcal{H}) - dt\,p_{s|\mathcal{H}}(s;\mathcal{H})\,\varepsilon_{\text{total}}(s)}{dt\sum_{s'\in\mathcal{S}}\left(p_{s|\mathcal{H}}(s';\mathcal{H})\right) - dt^2\sum_{s'\in\mathcal{S}}\left(p_{s|\mathcal{H}}(s';\mathcal{H})\,\varepsilon_{\text{total}}(s')\right)} - \frac{p_{s|\mathcal{H}}(s;\mathcal{H})}{dt}. \tag{3.25}$$

Because the sum of the conditional state probabilities, taken over all possible state, is 1,

$$c_{\text{ob}}(s;\mathcal{H}) = \frac{p_{s|\mathcal{H}}(s;\mathcal{H}) - dt\,p_{s|\mathcal{H}}(s;\mathcal{H})\,\varepsilon_{\text{total}}(s)}{dt - dt^2\sum_{s'\in\mathcal{S}} p_{s|\mathcal{H}}(s';\mathcal{H})\,\varepsilon_{\text{total}}(s')} - \frac{p_{s|\mathcal{H}}(s;\mathcal{H})}{dt}. \tag{3.26}$$

Defining $\hat{\varepsilon}_{\text{total}}(\mathcal{H})$ as the expected event observation rate, given the observation history $\mathcal{H}$,

$$\hat{\varepsilon}_{\text{total}}(\mathcal{H}) := \sum_{s\in\mathcal{S}} p_{s|\mathcal{H}}(s;\mathcal{H})\,\varepsilon_{\text{total}}(s), \tag{3.27}$$

and substituting,

$$c_{\text{ob}}(s;\mathcal{H}) = \frac{p_{s|\mathcal{H}}(s;\mathcal{H}) - dt\,p_{s|\mathcal{H}}(s;\mathcal{H})\,\varepsilon_{\text{total}}(s)}{dt - dt^2\hat{\varepsilon}_{\text{total}}(\mathcal{H})} - \frac{p_{s|\mathcal{H}}(s;\mathcal{H})}{dt}. \tag{3.28}$$

Converting to a common denominator,

$$c_{\text{ob}}(s;\mathcal{H}) = \frac{p_{s|\mathcal{H}}(s;\mathcal{H}) - dt\,p_{s|\mathcal{H}}(s;\mathcal{H})\,\varepsilon_{\text{total}}(s)}{dt - dt^2\hat{\varepsilon}_{\text{total}}(\mathcal{H})} - \frac{p_{s|\mathcal{H}}(s;\mathcal{H}) - dt\,p_{s|\mathcal{H}}(s;\mathcal{H})\,\hat{\varepsilon}_{\text{total}}(\mathcal{H})}{dt - dt^2\hat{\varepsilon}_{\text{total}}(\mathcal{H})}. \tag{3.29}$$

Subtracting,

$$c_{\text{ob}}(s;\mathcal{H}) = \frac{dt\,p_{s|\mathcal{H}}(s;\mathcal{H})\,\hat{\varepsilon}_{\text{total}}(\mathcal{H}) - dt\,p_{s|\mathcal{H}}(s;\mathcal{H})\,\varepsilon_{\text{total}}(s)}{dt - dt^2\hat{\varepsilon}_{\text{total}}(\mathcal{H})}. \tag{3.30}$$

Dividing the numerator and denominator by dt,

$$c_{\text{ob}}(s;\mathcal{H}) = \frac{p_{s|\mathcal{H}}(s;\mathcal{H})\,\hat{\varepsilon}_{\text{total}}(\mathcal{H}) - p_{s|\mathcal{H}}(s;\mathcal{H})\,\varepsilon_{\text{total}}(s)}{1 - dt\hat{\varepsilon}_{\text{total}}(\mathcal{H})}. \tag{3.31}$$

And then replacing the only infinitessimal term left with zero,

$$c_{\text{ob}}(s;\mathcal{H}) = p_{s|\mathcal{H}}(s;\mathcal{H})\,(\hat{\varepsilon}_{\text{total}}(\mathcal{H}) - \varepsilon_{\text{total}}(s;\mathcal{H})). \tag{3.32}$$

$c_{\text{ex}}(s;\mathcal{H})$ can be computed by subtracting the rate at which the system transitions away from state s from the rate at which the system transitions into state s:

$$c_{\text{ex}}(s;\mathcal{H}) = \sum_{s'\in\mathcal{S}} \Big(p_{s|\mathcal{H}}(s';\mathcal{H})\, \tau(s;s') \Big) - p_{s|\mathcal{H}}(s;\mathcal{H})\, \tau_{\text{from}}(s), \tag{3.33}$$

where $\tau_{\text{from}}(s)$ is the rate at which the system transitions away from state s,

$$\tau_{\text{from}}(s) := p_{s|\mathcal{H}}(s;\mathcal{H}) \sum_{s'\in\mathcal{S}} \tau(s';s). \tag{3.34}$$

Expanding both c_{ob} and c_{ex} in Equation 3.18, we can now arrive at Equation 3.2:

$$\boxed{\begin{aligned} \frac{d}{dt} p_{s|\mathcal{H}}(s;\mathcal{H}) = p_{s|\mathcal{H}}(s;\mathcal{H}) \Big(\hat{\varepsilon}_{\text{total}}(\mathcal{H}) - \varepsilon_{\text{total}}(s) - \tau_{\text{from}}(s) \Big) \\ + \sum_{s'\in\mathcal{S}} p_{s|\mathcal{H}}(s';\mathcal{H})\, \tau(s;s'). \end{aligned}} \tag{3.2}$$

3.B Derivation for Equation 3.6

When an observation of type o occurs, by Bayes rule,

$$p_{s|\mathcal{H}}(s; \mathcal{H}^\frown o) = \lim_{\Delta t \to 0} \frac{p_{s|\mathcal{H}}(s; \mathcal{H})\, p_{o|s}(o; s, \Delta t)}{\sum_{s' \in \mathcal{S}} p_{s|\mathcal{H}}(s'; \mathcal{H})\, p_{o|s}(o; s', \Delta t)}, \tag{3.35}$$

where "$^\frown$" denotes sequence-element concatenation and $p_{o|s}(o; s, \Delta t)$ is the probability that exactly one event, of type o, will occur in a Δt-second period in which the forager is in state class s. The number of events of any type o that will occur in such a timespan follows a Poisson distribution with a mean $\Delta t\, \varepsilon(o; s)$, so

$$p_{o|s}(o; s, \Delta t) = \text{pois}(1; \Delta t\, \varepsilon(o; s))\, \text{pois}(0; \Delta t\, \varepsilon(\neg o; s)), \tag{3.36}$$

where $\text{pois}(n; \mu)$ is the probability mass of the Poisson distribution with mean μ at n, and $\varepsilon(\neg o; s)$ is the rate at which events other than those of type o occur when the forager is in state class s.

Expanding the Poisson probability mass function,

$$p_{o|s}(o; s, \Delta t) = \Delta t\, \varepsilon(o; s) \exp(-\Delta t\, \varepsilon(o; s)) \exp(-\Delta t\, \varepsilon(\neg o; s)). \tag{3.37}$$

Substituting this definition into Equation 3.35,

$$p_{s|\mathcal{H}}(s; \mathcal{H}^\frown o) = \lim_{\Delta t \to 0} \frac{p_{s|\mathcal{H}}(s; \mathcal{H})\, \Delta t\, \varepsilon(o; s) \exp(-\Delta t\, \varepsilon(o; s)) \exp(-\Delta t\, \varepsilon(\neg o; s))}{\sum_{s' \in \mathcal{S}} p_{s|\mathcal{H}}(s'; \mathcal{H})\, \Delta t\, \varepsilon(o; s') \exp(-\Delta t\, \varepsilon(o; s')) \exp(-\Delta t\, \varepsilon(\neg o; s'))}. \tag{3.38}$$

Letting the factors of Δt cancel each other out,

$$p_{s|\mathcal{H}}(s; \mathcal{H}^\frown o) = \lim_{\Delta t \to 0} \frac{p_{s|\mathcal{H}}(s; \mathcal{H})\, \varepsilon(o; s) \exp(-\Delta t\, \varepsilon(o; s)) \exp(-\Delta t\, \varepsilon(\neg o; s))}{\sum_{s' \in \mathcal{S}} p_{s|\mathcal{H}}(s'; \mathcal{H})\, \varepsilon(o; s') \exp(-\Delta t\, \varepsilon(o; s')) \exp(-\Delta t\, \varepsilon(\neg o; s'))}. \tag{3.39}$$

And taking the limit,

$$\boxed{p_{s|\mathcal{H}}(s; \mathcal{H}^\frown o) = \frac{p_{s|\mathcal{H}}(s; \mathcal{H})\, \varepsilon(o; s)}{\sum_{s' \in \mathcal{S}} p_{s|\mathcal{H}}(s'; \mathcal{H})\, \varepsilon(o; s')}.} \tag{3.6}$$

References

[1] Jan M. Ache et al. “Neural Basis for Looming Size and Velocity Encoding in the Drosophila Giant Fiber Escape Pathway”. en. In: *Current Biology* 29.6 (Mar. 2019), 1073–1081.e4. ISSN: 09609822. DOI: 10.1016/j.cub.2019.01.079. URL: https://linkinghub.elsevier.com/retrieve/pii/S0960982219301381 (visited on 10/09/2024).

[2] Karl Johan Åström. “Optimal control of Markov processes with incomplete state information I”. In: *Journal of mathematical analysis and applications* 10 (1965), pp. 174–205.

[3] Sebastian Bitzer et al. “Perceptual decision making: drift-diffusion model is equivalent to a Bayesian model”. en. In: *Frontiers in Human Neuroscience* 8 (2014). ISSN: 1662-5161. DOI: 10.3389/fnhum.2014.00102. URL: http://journal.frontiersin.org/article/10.3389/fnhum.2014.00102/abstract (visited on 10/09/2024).

[4] Gwyneth Card and Michael H. Dickinson. “Visually Mediated Motor Planning in the Escape Response of Drosophila”. en. In: *Current Biology* 18.17 (Sept. 2008), pp. 1300–1307. ISSN: 09609822. DOI: 10.1016/j.cub.2008.07.094. URL: https://linkinghub.elsevier.com/retrieve/pii/S0960982208010488 (visited on 10/09/2024).

[5] Robert J Collins and David Jefferson. *Antfarm: Towards simulated evolution*. Computer Science Department, University of California Los Angeles, CA, 1990.

[6] Robert J. Collins and David R. Jefferson. “An artificial neural network representation for artificial organisms”. en. In: *Parallel Problem Solving from Nature*. Ed. by Hans-Paul Schwefel and Reinhard Männer. Vol. 496. Series Title: Lecture Notes in Computer Science. Berlin/Heidelberg: Springer-Verlag, 1991, pp. 259–263. ISBN: 978-3-540-54148-6. DOI: 10.1007/BFb0029761. URL: http://link.springer.com/10.1007/BFb0029761 (visited on 10/09/2024).

[7] James F Crow. “Wright and Fisher on inbreeding and random drift”. In: *Genetics* 184.3 (2010), pp. 609–611.

[8] Lawrence M. Dill. “The escape response of the zebra danio (Brachydanio rerio) I. The stimulus for escape”. en. In: *Animal Behaviour* 22.3 (Aug. 1974), pp. 711–722. ISSN: 00033472. DOI: 10.1016/S0003-3472(74)80022-9. URL: https://linkinghub.elsevier.com/retrieve/pii/S0003347274800229 (visited on 10/09/2024).

[9] Emmanuel Dufourq and Bruce A Bassett. “Eden: Evolutionary deep networks for efficient machine learning”. In: *2017 Pattern Recognition Association of South Africa and Robotics and Mechatronics (PRASA-RobMech)*. IEEE. 2017, pp. 110–115.

[10] Edgar Galvan and Peter Mooney. "Neuroevolution in Deep Neural Networks: Current Trends and Future Challenges". en. In: *IEEE Transactions on Artificial Intelligence* 2.6 (Dec. 2021), pp. 476–493. ISSN: 2691-4581. DOI: 10.1109/TAI.2021.3067574. URL: https://ieeexplore.ieee.org/document/9383028/ (visited on 10/09/2024).

[11] William T Gibson et al. "Behavioral responses to a repetitive visual threat stimulus express a persistent state of defensive arousal in Drosophila". In: *Current Biology* 25.11 (2015), pp. 1401–1415.

[12] Arthur J Hallinan Jr. "A review of the Weibull distribution". In: *Journal of quality technology* 25.2 (1993), pp. 85–93.

[13] Jan M Hemmi and Daniel Tomsic. "The neuroethology of escape in crabs: from sensory ecology to neurons and back". en. In: *Current Opinion in Neurobiology* 22.2 (Apr. 2012), pp. 194–200. ISSN: 09594388. DOI: 10.1016/j.conb.2011.11.012. URL: https://linkinghub.elsevier.com/retrieve/pii/S0959438811002157 (visited on 10/09/2024).

[14] Jan M. Hemmi. "Predator avoidance in fiddler crabs: 1. Escape decisions in relation to the risk of predation". en. In: *Animal Behaviour* 69.3 (Mar. 2005), pp. 603–614. ISSN: 00033472. DOI: 10.1016/j.anbehav.2004.06.018. URL: https://linkinghub.elsevier.com/retrieve/pii/S0003347204004191 (visited on 10/09/2024).

[15] David J Montana, Lawrence Davis, et al. "Training feedforward neural networks using genetic algorithms." In: *IJCAI*. Vol. 89. 1989. 1989, pp. 762–767.

[16] WWTG Peterson, T Birdsall, and We Fox. "The theory of signal detectability". In: *Transactions of the IRE professional group on information theory* 4.4 (1954), pp. 171–212.

[17] João Carlos Figueira Pujol and Riccardo Poli. "Efficient evolution of asymmetric recurrent neural networks using a PDGP-inspired two-dimensional representation". en. In: *Genetic Programming*. Ed. by Gerhard Goos et al. Vol. 1391. Series Title: Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 130–141. ISBN: 978-3-540-64360-9 978-3-540-69758-9. DOI: 10.1007/BFb0055933. URL: http://link.springer.com/10.1007/BFb0055933 (visited on 10/09/2024).

[18] James C Spall. "A stochastic approximation technique for generating maximum likelihood parameter estimates". In: *1987 American control conference*. IEEE. 1987, pp. 1161–1167.

[19] James C Spall. "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation". In: *IEEE transactions on automatic control* 37.3 (1992), pp. 332–341.

[20] Felipe Petroski Such et al. *Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning*. en. arXiv:1712.06567 [cs]. Apr. 2018. URL: http://arxiv.org/abs/1712.06567 (visited on 10/09/2024).

[21] Wilson P Tanner Jr and John A Swets. "A decision-making theory of visual detection." In: *Psychological review* 61.6 (1954), p. 401.

[22] Jessica Thiem et al. "Biological aging of two innate behaviors of Drosophila melanogaster: Escape climbing versus courtship learning and memory". In: *Plos one* 19.4 (2024), e0293252.

[23] Ineke T. Van Der Veen and Karin M. Lindström. "Escape flights of yellowhammers and greenfinches: more than just physics". en. In: *Animal Behaviour* 59.3 (Mar. 2000), pp. 593–601. ISSN: 00033472. DOI: 10.1006/anbe.1999.1331. URL: https://linkinghub.elsevier.com/retrieve/pii/S0003347299913313 (visited on 10/09/2024).

[24] Catherine R. Von Reyn et al. "Feature Integration Drives Probabilistic Behavior in the Drosophila Escape Response". en. In: *Neuron* 94.6 (June 2017), 1190–1204.e6. ISSN: 08966273. DOI: 10.1016/j.neuron.2017.05.036. URL: https://linkinghub.elsevier.com/retrieve/pii/S0896627317304749 (visited on 10/09/2024).

[25] D Whitley, T Starkweather, and C Bogart. "Genetic algorithms and neural networks: optimizing connections and connectivity". en. In: *Parallel Computing* 14.3 (Aug. 1990), pp. 347–361. ISSN: 01678191. DOI: 10.1016/0167-8191(90)90086-O. URL: https://linkinghub.elsevier.com/retrieve/pii/0167819190900860 (visited on 10/09/2024).

[26] Sewall Wright. "Evolution in Mendelian populations". In: *Genetics* 16.2 (1931), p. 97.

[27] Sewall Wright. "The roles of mutation, inbreeding, crossbreeding, and selection in evolution". In: *Proceedings of the Sixth International Congress of Genetics*. Vol. 1. 1932, pp. 356–366.

[28] Xing Yang et al. "A simple threat-detection strategy in mice". en. In: *BMC Biology* 18.1 (Dec. 2020), p. 93. ISSN: 1741-7007. DOI: 10.1186/s12915-020-00825-0. URL: https://bmcbiol.biomedcentral.com/articles/10.1186/s12915-020-00825-0 (visited on 10/09/2024).

[29] Ron C Ydenberg and Lawrence M Dill. "The economics of fleeing from predators". In: *Advances in the Study of Behavior*. Vol. 16. Elsevier, 1986, pp. 229–249.

[30] Melis Yilmaz and Markus Meister. "Rapid Innate Defensive Responses of Mice to Looming Visual Stimuli". en. In: *Current Biology* 23.20 (Oct. 2013), pp. 2011–2015. ISSN: 09609822. DOI: 10.1016/j.cub.2013.08.015. URL: https://linkinghub.elsevier.com/retrieve/pii/S0960982213009913 (visited on 10/09/2024).

Chapter 4

A MAP OF OBJECT SPACE IN PRIMATE INFEROTEMPORAL CORTEX

Forward

This chapter discusses an investigation into the organization of primate inferotemporal (IT) cortex. IT cortex is responsible for object recognition, but it is unclear how the representation of visual objects is organized in this part of the brain. Areas that are selective for categories such as faces, bodies, and scenes have been found [16, 37, 10, 30, 19], but large parts of IT cortex lack any known specialization, raising the question of what general principle governs IT organization.

We used functional MRI, microstimulation, electrophysiology, and deep networks to investigate the organization of macaque IT cortex. We built a low-dimensional object space to describe general objects using a feedforward deep neural network trained on object classification [20]. Responses of IT cells to a large set of objects revealed that single IT cells project incoming objects onto specific axes of this space. Anatomically, cells were clustered into four networks according to the first two components of their preferred axes, forming a map of object space. This map was repeated across three hierarchical stages of increasing view-invariance, and cells that comprised these maps collectively harbored sufficient coding capacity to approximately reconstruct objects. These results provide a unified picture of IT organization in which category-selective regions are part of a coarse map of object space whose dimensions can be extracted from a deep network.

This chapter is adapted from a 2020 *Nature* paper with the same name, coauthored with Pinglei Bao, Liang She, and Doris Tsao [5]. My contributions to this study included (a) reconstructing images presented to macaque monkeys from neural firing patterns, (b) generating latent space embeddings for the stimulus images using both pre-trained and custom convolutional neural networks, (c) mapping deep network latent

> spaces to IT representation spaces, and (d) generating representation space visualizations that contributed to the characterization of "no man's land" patches in IT—which contain neurons that preferentially respond to spiky or spindly objects—and the discovery of additional patches containing neurons that prefer stubby objects.

Object recognition—the process by which distinct visual forms are assigned distinct identity labels—lies at the heart of our ability to make sense of the visual world. It underlies many neural processes that operate on objects, including attention, visual memory, decision making, and language. Befitting the central importance and computational complexity of object recognition, a large volume of the brain—inferotemporal (IT) cortex—is dedicated to solving this challenge [13].

One of the most striking features of IT is the existence of several distinct anatomical networks that are specialized for processing specific object categories [37, 30, 19] or stimulus dimensions [22, 39, 15, 41]. However, these networks comprise only part of IT, and much of IT is not differentially activated by any known stimulus comparison. Here we investigate whether this "unexplained" IT shows any functional specialization. Furthermore, beyond simply parcelling IT, we investigate whether there is an overarching general principle governing the anatomical layout of IT cortex.

Many previous studies have tried to address this latter question, but the answers obtained remain piecemeal. Early studies using electrophysiology in monkeys suggested a columnar architecture for visual shape [12], but the small field-of-view of electrophysiology precluded understanding the larger-scale organization of these columns. Later studies, using functional MRI (fMRI) in humans, proposed various schemes to explain large-scale IT organization, including retinotopy [23] and real-world size [18], but these proposals did not provide a complete account of IT organization and lacked ground-truth validation at the level of single units. To investigate the organization of macaque IT at multiple scales, we combined fMRI, electrical microstimulation, and electrophysiology in the same animals. And we found that a large portion of macaque IT cortex is topographically organized into a map of object space that is repeated three times.

4.1 Identifying a new IT network

To discover the functional specialization of still-unexplained parts of IT cortex, one strategy would be to guess. However, lacking any good guesses, we decided to approach the problem from an anatomical perspective. We ran a large set of

stimulus comparisons to localize face, body, scene, color, and disparity patches in a specific monkey (M1) and thereby defined the "no man's land" of IT cortex in this monkey: regions that were not identified by any known localizer (Fig. 4.1a, b). We then electrically microstimulated a random site within this no man's land in central IT cortex [26]. This experiment revealed that the stimulated region (NML2) was connected to two other, discrete regions in IT (NML1, NML3) (Fig. 4.1b, Fig. 4.6), forming a previously unknown anatomical network within the no man's land.

To understand the function of this new network, we first recorded the neural responses of cells in the three patches to 1,224 images, each containing one of 51 objects—spanning 6 object categories—presented at one of 24 viewing angles (Fig. 4.7a, b). Responses were remarkably consistent (Fig. 4.2a, and Fig. 4.8a–d). Cells in all three patches responded minimally to faces. And their preferred stimuli, while consistent across patches, were not confined to any one semantic category (Fig. 4.2a).

To investigate whether this network exists in every animal, we identified the five most- and least-preferred objects of the network based on mean responses of cells recorded from monkey M1 (Fig. 4.2a). We presented these stimuli to monkey M1 in an fMRI experiment and confirmed that the resulting map overlapped with the map revealed by microstimulation (Fig. 4.2e). We then presented these stimuli to three other monkeys (M2–M4) and found similar networks in all three animals (Fig. 4.2e). Single-unit recordings targeted to this network in monkey M2 revealed a response

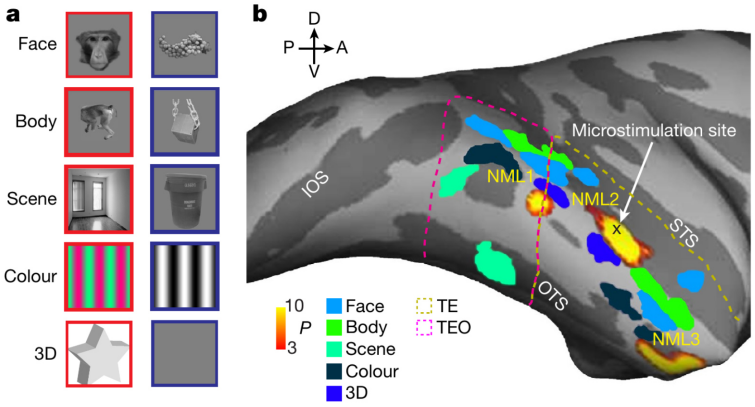

Figure 4.1: **a:** Stimulus contrasts used to identify known networks in IT (see Methods). **b:** Inflated brain (right hemisphere) for monkey M1 showing known IT networks mapped in this animal. Regions activated by microstimulation of NML2 are shown in yellow. All activation maps shown at a threshold of $p < 10^{-3}$, not corrected for multiple comparisons. Yellow and magenta outlines indicate the boundaries of TE and TEO, respectively [34].

pattern that was highly consistent with that in monkey M1 (Fig. 4.2a). (Pearson correlation of the mean responses to each object between monkeys M1 and M2, $r = 0.\ 89$, $p < 10^{-16}$.) This justifies referring to an "NML network" across animals.

In the face patch network, neurons in posterior patches are view-specific, whereas those in the most anterior patch are more view-invariant [11]. We found a similar difference between the three NML patches in terms of their view-invariance. Significantly more cells in NML3 were view-invariant than in NML1 (two-tailed t-test; $t(137) = 5.10$, $p < 10^{-5}$; Fig. 4.8e). Population similarity matrices relating responses to objects across views also showed an increase in view-invariance going anteriorly, with emergence of parallel diagonal stripes in the NML3 similarity matrix (Fig. 4.3a, top, and Fig. 4.8f). Notably, many cells showed view-invariance to objects that the monkey had not experienced, such as an airplane (Fig. 4.3b, top).

Next, we investigated what is being encoded by cells in this network. Scrutinizing the most- and least-preferred objects (Fig. 4.2a, bottom), we noticed that all of the preferred objects contained thin protrusions, whereas the non-preferred objects were round. This suggested that one feature NML neurons might be selective for is high aspect ratio. We confirmed this using both responses to the original object image set (Fig. 4.8g, see Methods) as well as a simplified stimulus set consisting of a line segment independently varied in aspect ratio, curvature, and orientation (Fig. 4.2f, Fig. 4.7c).

4.2 NML cells encode axes of object space

We next attempted to identify the relevant shape dimensions for the NML network in a systematic way that did not depend on subjective visual inspection. Until recently this was difficult, due to the lack of a computational scheme for parametrizing the shape of arbitrary objects. But modern deep networks trained to classify objects provide a powerful solution to this problem [40]. They can be used to describe arbitrary objects as a sequence of a few thousand numbers: the unit activations in a deep layer. And to make the parametrization even more compact, one can perform principal components analysis (PCA) on these unit activations.

We built an object space by passing the stimulus set we presented to the monkey (Fig. 4.7a, b) through AlexNet—a deep network trained on object classification [20]—and then performing PCA on the responses of units in layer fc6 of this network (Fig. 4.9a). The first principal component (PC) roughly divides things with protrusions (spiky) from those without (stubby) (Fig. 4.9b). And the second

PC roughly divides animate object from inanimate objects. (Note that we use "animate" and "inanimate" as shape descriptors, without any semantic connotation.) We determined that 50 object dimensions could explain 85% of the variance in the AlexNet fc6 response (Fig. 4.9c) and so used 50 dimensions in the remaining analyses. We then analysed the responses of cells in the NML network by computing a "preferred axis" for each cell through linear regression. That is, we computed coefficients $\mathbf{c}$ in the equation $R = \mathbf{c} \cdot \mathbf{f} + c_0$, where R is the response of the cell, $\mathbf{f}$ is the 50D object feature vector, and c_0 is a constant offset (see Methods).

Cells showed significant tuning to many of the 50 object dimensions (Pearson correlation $p < 10^{-3}$ between feature values and neural responses). On average, each cell was significantly tuned to seven dimensions. Notably, the preferred axis of each cell was stable to the precise image set (Fig. 4.10a). The 50D linear object space model could explain 44.7% of the variance, or 53.3% of the explainable variance, of NML neurons on average (Fig. 4.10b). This is significantly higher than a Gaussian model, and similar in efficacy to a quadratic model (Fig. 4.10c, d).

Consistent with the high explained variance by the linear model, cell tuning along the preferred axis in the 50D object space was ramp-shaped (Fig. 4.3c, top). Similar ramp-shaped tuning has previously been reported for face-selective cells [7]. NML neurons also showed approximately flat tuning along orthogonal axes (Fig. 4.10e)—another property that has been previously observed in face-selective cells [7]. Together, ramp-shaped tuning along the preferred axis and flat tuning along orthogonal axes implies that cells in the NML network are linearly projecting incoming objects, formatted as vectors in object space, onto specific preferred axes.

Overall, the organization and code of the NML network are strikingly similar to those of the face patch network. The NML network consists of connected patches, cells within the network show a consistent pattern of selectivity, there is increasing view-invariance along the network, and single cells in the network represent object identity through an axis code. So there seems to be a clear structural parallel between the face network and the NML network. We therefore investigated whether additional networks in IT cortex follow the same scheme.

4.3 The body network follows the same scheme

We next recorded from the macaque body network—a set of regions adjacent to face patches that respond more to animate compared to inanimate objects [30] (Fig. 4.2b)—as well as the face network (Fig. 4.2c). Population similarity matrices

showed increased view-invariance in the most anterior body patch (Fig. 4.3a, b, middle, and Fig. 4.8e, f), consistent with a previous study [21]. Cells in the body network also showed ramp-shaped tuning along their preferred axes (Fig. 4.3c, middle, and Fig. 4.10a) and flat tuning along orthogonal axes (Fig. 4.10e). So the body network follows the same general anatomical organization and encoding scheme as the NML and face networks.

4.4 A general rule governing IT organization

The discovery of three networks (NML, body, and face) that all follow the same organization and encoding scheme suggests that there might be a general principle that governs the organization of IT cortex. Recall that the first two axes of our object space were roughly stubby versus spiky, and animate versus inanimate (Fig. 4.9b). We noticed a remarkable relationship between these two axes and the selectivity of the NML, body, and face networks. Face patches prefer stubby animate objects; body patches prefer spiky, animate objects; and NML patches prefer spiky objects regardless of animacy (Fig. 4.2a). These observations made us wonder whether all of IT might be topographically organized according to the first two dimensions of our object space (Fig. 4.4a), in the same way that retinotopic cortex is organized according to polar angle and eccentricity.

As a first step to test this hypothesis, we projected all the stimuli that we showed to the monkey onto the first two dimensions of our object space, and marked the top 100 images for the NML, body, and face networks (Fig. 4.4b; orange, green, and blue dots). They approximately spanned three quadrants of the space. If IT cortex is indeed laid out according to the first two dimensions of this object space, we predicted there should be a fourth network representing objects that project strongly onto the remaining unrepresented quadrant: stubby, inanimate objects without protrusions, like a ball or a USB stick.

To test this prediction, we first ran an fMRI experiment with four blocks, corresponding to the four quadrants of the object space (Fig. 4.4a). And comparing measurements during stubby blocks to measurements during the other blocks revealed a network that contained multiple patches selective for stubby objects (Fig. 4.4c). Electrophysiology targeted to two of these patches revealed cells that were strongly selective for stubby objects (Fig. 4.2d), whose preferred axes occupied the previously unrepresented quadrant (Fig. 4.4b, magenta dots). The general properties of the stubby network were very similar to those of the NML, face, and body networks. Population

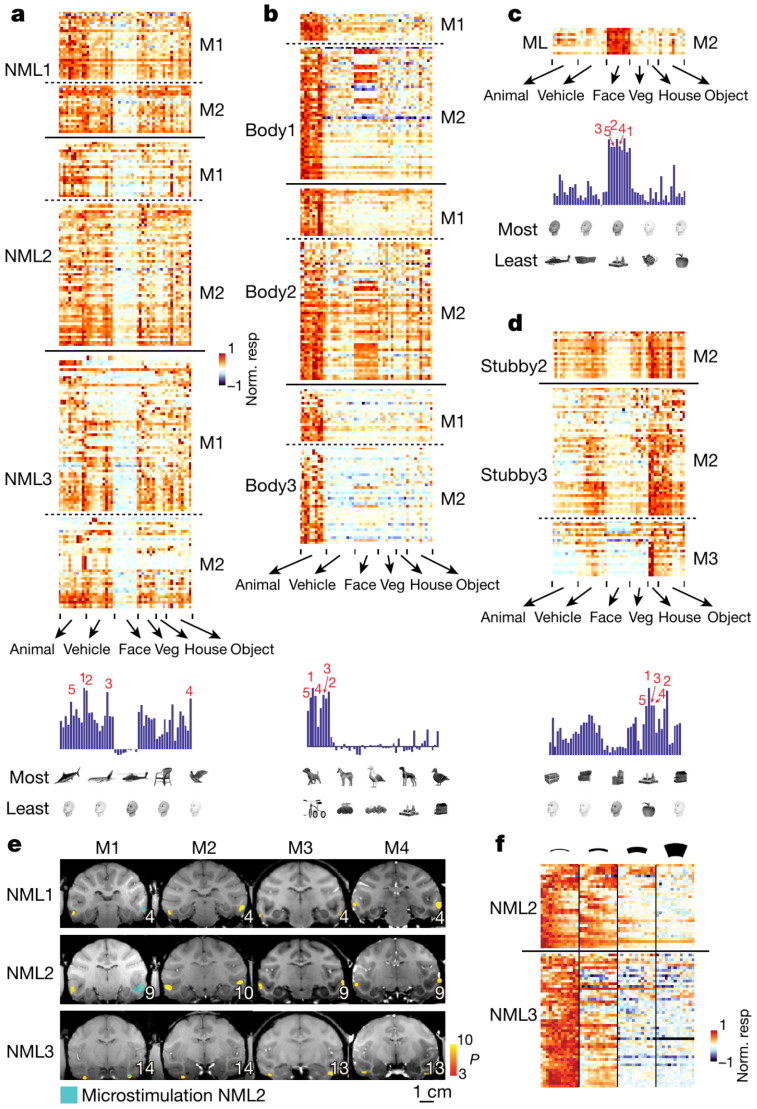

Figure 4.2: **a–d, top:** Responses of cells to 51 objects from six different categories. Responses to each object were averaged across 24 views. Cells were recorded in three patches (NML1, NML2 and NML3) from the NML network (a), in three patches of the body network (b), in patch ML of the face network (c), and in two patches of the stubby network (d). **a–d, middle:** Blue charts show average responses to each object in each network. Numbers indicate the five most-preferred objects. **a–d, bottom:** The five most-preferred (top row) and least-preferred (bottom row) objects for each network, based on averaged responses. Images 1 to 5 are shown from left to right. **e:** Coronal slices containing NML1, NML2, and NML3 from monkeys M1, M2, M3, and M4 showing difference in activation in response to the five most-preferred versus five least-preferred objects determined from electrophysiology in the NML network of monkey M1. In M1, the microsimulation result is also shown as a cyan overlay with threshold $p < 10^{-3}$, uncorrected. Inset numbers indicate AP coordinate relative to interaural 0 [34]. (Continued on the next page)

Fig. 4.2, continued: **f:** Responses of cells from patches NML2 and NML3 of the NML network to a line segment that varied in aspect ratio, curvature, and orientation. Responses are averaged across orientation, and curvature runs from low to high from left to right for each aspect ratio. Aspect ratio accounts for 22.8% of the response variance on average across cells, curvature for 5.6% of the variance, and orientation for 3.5% of the variance.

similarity matrices showed increased view-invariance in the most anterior stubby patch (Fig. 4.3a, b, bottom), Fig. 4.8f). Cells in the stubby network also showed ramp-shaped tuning along their preferred axes (Fig. 4.3c, bottom, and, Fig. 4.10a) and flat tuning along orthogonal axes (Fig. 4.10e). Thus, the hypothesis that IT is organized according to the first two dimensions of our object space revealed a second new shape network.

One potential concern is that the collection of 51 objects at 24 views that we used to assess the selectivity of cells in each network was too sparse to allow us to identify the true selectivity of cells. With this in mind, we presented 1,593 completely different objects to a subset of cells in the NML, body, and stubby networks, and found responses consistent with the responses to our original stimulus set (Fig. 4.11a, b). In particular, preferred axes measured using the new stimuli segregated into three distinct regions of object PC1-PC2 space (Fig. 4.11a), and the preferred stimuli of each network were qualitatively similar to those identified using the original stimuli (Fig. 4.11b).

It might seem suspiciously serendipitous for IT to be organized according to the first two dimensions of an object space computed using a specific image set with a specific deep convolutional network. But it turns out that these first two axes do not depend strongly on the particular image set (Fig. 4.9d–f) or network (Fig. 4.9g–j) used to compute them.

4.5 A map of object space

What is the anatomical layout of the face, body, NML, and stubby networks? An overlay of the four networks onto coronal slices and a cortical flat map revealed a remarkably ordered progression (Fig. 4.4c, d; see Fig. 4.12 for response time courses from each patch). There is a clear sequence from body to face to stubby to NML in both hemispheres that is repeated in the same order in posterior, middle, and anterior IT. This pattern was consistent across animals (Fig. 4.4c, d) and confirmed by quantitative analysis of the linear fit between patch-ordered label and cortical location

of patch peak ($p < 10^{-18}$ for posterior, middle, and anterior IT'; Fig. 4.4e–g). This strikingly regular progression suggests the existence of a coarse map of object space that is repeated at least three times, with increasing view-invariance at each stage.

These four networks, together with the disparity, scene, and color networks, occupy about 53% of IT cortex, so additional networks may exist. Not all of the networks consisted of exactly three patches; for example, the stubby and NML networks each contained four patches (Fig. 4.4d), and previous work has suggested that there are six face patches in each hemisphere, with some individual variability [35]. Thus, IT cortex may contain additional repetitions of the object space map. Furthermore, we emphasize that our study addresses IT organization at a coarse spatial scale and does not exclude the possibility of additional organization at finer spatial scales

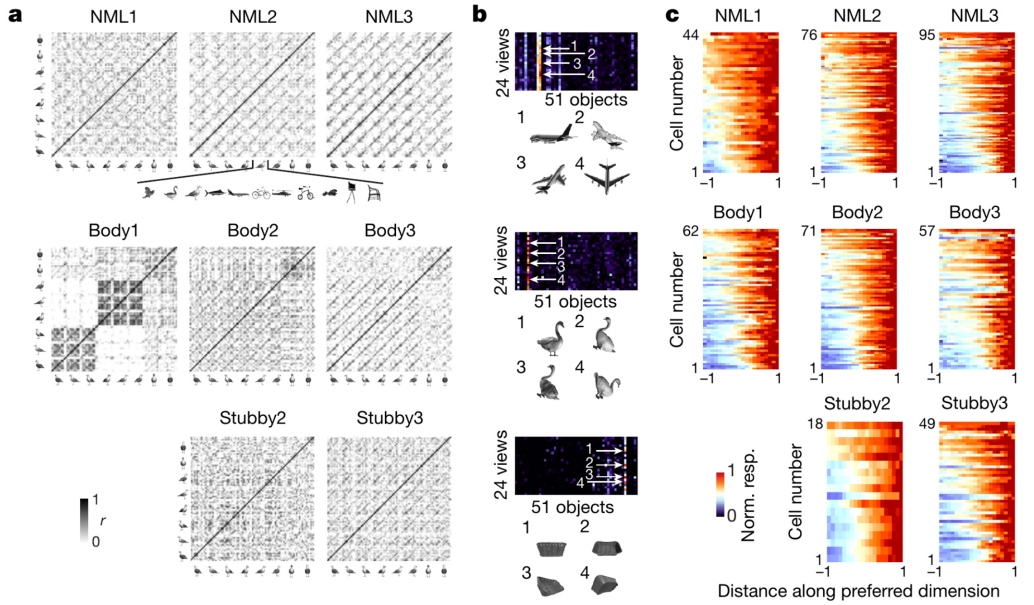

Figure 4.3: **a:** Population similarity matrices in the three patches of the NML network (top), three patches of the body network (middle) and two patches of the stubby network (bottom) pooled across monkeys M1 and M2. An 88×88 matrix of correlation coefficients was computed from responses of cells in each patch to 88 stimuli (8 views × top-11 preferred objects). **b:** Responses from three example cells recorded in NML3 (top), the body network (middle) and the stubby network (bottom) to 51 objects at 24 views. Four views of the most preferred object are shown below each response matrix. **c:** Responses of neurons recorded from patches in the NML network (top), the body network (middle) and the stubby network (bottom) as a function of distance along the preferred axis. The abscissa is rescaled so that the range $[-1, 1]$ covers 95% of the stimuli. Half of the stimulus trials were used to compute the preferred axis for each cell, and held-out data was used to plot the responses shown.

(Fig. 4.13). Recordings from multiple grid holes suggest that each patch spans 3–4 mm (Fig. 4.13a–d). Although we failed to find clustering at finer scales within a patch (Fig. 4.13e, f) or clustering for any dimensions beyond the first two (Fig. 4.13g, h), it is possible that mapping techniques with higher spatial resolution may reveal additional substructure within patches.

If the first two dimensions of object space derived from a deep network are indeed meaningful in terms of brain representation, then we should be able to design novel stimuli to identify the four networks. With this in mind, we generated three new image sets—silhouettes, fake objects, and deep dream images—with very different properties from those of the original image set shown in Fig. 4.4a. In each case, fMRI revealed four networks similar to those in Fig. 4.4 (Fig. 4.11c–e).

4.6 Explaining previous accounts of IT

The principle that IT cortex is organized according to the first two axes of object space provides a unified explanation for many previous observations concerning the functional organization of IT, including not only the existence of face [16] and body areas [10], but also gradients for representing animate versus inanimate and small versus large objects [18] (Fig. 4.14a, b), a gradient for representing open versus closed topologies [38], (Fig. 4.14c), the curvature network [41] (Fig. 4.14d), and the visual word form area [25] (Fig. 4.14e). Furthermore, within category-selective regions, the object space model explains activity better than the semantic category hypothesis [3] (Fig. 4.15). Overall, these results demonstrate the large explanatory power of the object space model.

4.7 Reconstructing general objects

We next investigated the richness of the feature space represented by cells in the four networks that comprise the map of object space. To quantify the object information available in the map of object space formed by the four networks, we attempted to decode object identity using the responses of cells from these networks. We used leave-one-object-out cross-validation to learn a linear transform that maps responses to features (Fig. 4.16a, b). The explained variance for each dimension showed that many dimensions are represented in each network beyond the first two (Fig. 4.16c), allowing a target object to be identified among distractors (Fig. 4.16d–f).

To directly visualize the information about object features that is carried by neurons in these four networks, we attempted to reconstruct general objects using neural activity. We inferred object images from decoded feature vectors using a generative

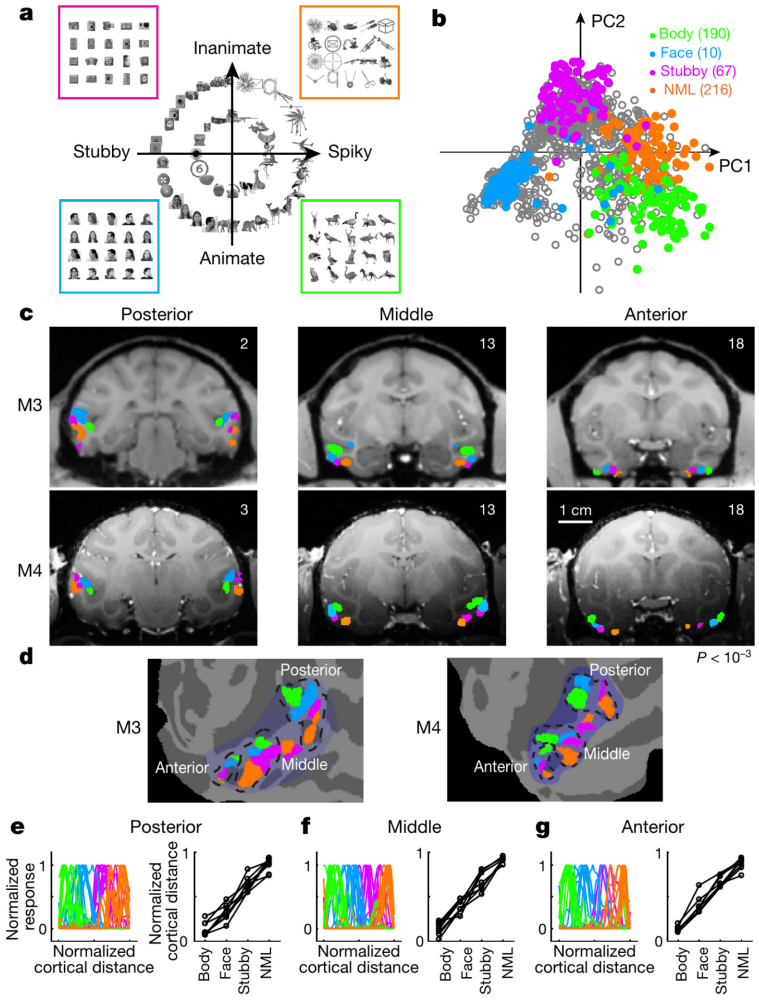

Figure 4.4: **a:** A schematic plot showing the map of objects generated by the first two PCs of our object space. The stimuli in the rectangular boxes were used for mapping the four networks shown in (c) and (d) using fMRI. **b:** All the stimuli used in the electrophysiology experiments (Fig. 4.7a, b), projected onto the first two dimensions of the object space (grey circles). For each network, the top 100 preferred images are marked (body network: green, face network: blue, stubby network: magenta, NML network: orange). Numbers in parentheses indicate the number of neurons recorded from each network. **c:** Coronal slices from posterior, middle, and anterior IT of monkeys M3 and M4 showing the spatial arrangement of the four networks (maps thresholded at $p < 10^{-3}$, uncorrected). Here, the networks were computed using responses to the stimuli in (a). **d:** As in (c), showing the four networks in monkeys M3 and M4 overlaid on a flat map of the left hemisphere. **e, left:** Spatial profiles of the four patches along the cortical surface within posterior IT for data from two hemispheres of four animals. The y-axis shows the normalized significance level for each comparison of each voxel, and the x-axis shows the position of the voxel on the cortex (see Methods). **e, right:** Anatomical locations of the peak responses plotted against the sequence of quadrants in object space. **f, g:** As in (e), for voxels from middle IT (f) and anterior IT (g).

adversarial network that was trained to invert the computation performed when passing an image into AlexNet [9] and reading out the representation at layer fc6. Reconstructions captured details of the original images with an impressive level of accuracy (Fig. 4.5a). Figure 5b shows the distribution of normalized reconstruction distances between the actual and best possible reconstructions (see Methods). As a second method to recover objects from neural activity, we searched a large auxiliary object database for the object with a feature vector closest to that decoded from neural activity. This method also yielded recovered images that picked up many fine structural details (Fig. 4.16g). Overall, these results suggest that the four networks of the IT object space map are sufficient to encode a reasonably complete representation of general objects, which implies that the number of networks used to solve general object recognition does not need to be astronomically high.

4.8 Discussion

We have shown that IT contains a coarse map of object space that is repeated three times, with increasing invariance at each stage. This map consists of at least four regions that tile object space. The map parsimoniously accounts for the previously reported face and body networks, as well as two new networks: the NML network and the stubby network. Single cells in each of the four networks use an encoding principle similar to that previously identified for the face network—projection of incoming objects, formatted as points in an object space, onto a preferred axis. The

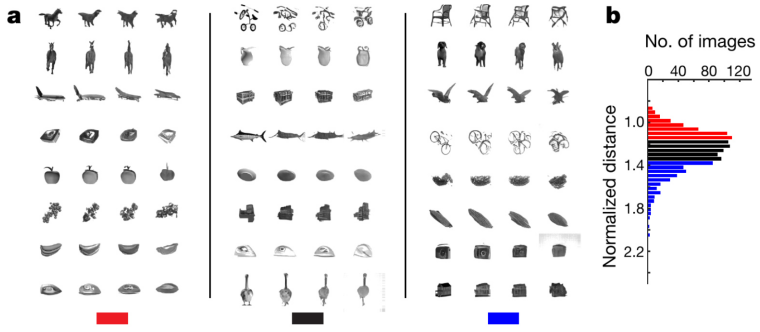

Figure 4.5: **a:** Reconstructions using 482 cells from the NML, body, stubby, and face networks. Example reconstructed images from the three groups defined in (b) are shown. Each row of four images shows from left to right: (1) the original image, (2) the reconstruction using the fc6 response to the original image, (3) the reconstruction using the fc6 response projected onto the 50D object space, and (4) the reconstruction based on neuronal data. **b:** The distribution of normalized distances between reconstructed feature vectors and best-possible reconstructed feature vectors (see Methods).

four networks that comprise the IT object-topic map, together with the scene, color, and disparity networks, cover about 53% of IT. Pooling responses across the four networks enabled reasonable reconstruction of general objects, suggesting that these four networks provide a basis that spans a general object space. By showing that the modular organization previously thought to be unique to a few categories may actually extend across a much larger swath of IT, we provide a powerful new map for experiments that require spatially specific interrogation of object representations.

It remains unknown whether borders between the patches are continuous or discrete [1], as fMRI-guided single-unit recording is not ideal for mapping sub-millimetre-scale structure. If the borders turn out to be continuous, this would imply that the entire notion of IT modularity may be an artifact of limited field of view. On the other hand, if the borders turn out to be discrete, this would suggest that additional factors (for example, extensive experience with specific categories [2]) may support the formation of uniquely specialized modules in cortex. The coarse map of object space identified here provides a foundation for future fine-scale mapping studies to tackle this question.

The finding that neurons in IT are clustered according to axis similarity resonates with recent approaches to unsupervised learning of object representations that seek optimal clustering of data in low-dimensional embeddings [42]. It will be important to understand why IT physically clusters neurons with similar axes—something not currently implemented in deep networks. One possible reason is that physical clustering may help to refine object representations through lateral inhibition and aid object identification in clutter [4].

Our results cast the face patch system in a new light. Previously, it was thought that the face system, with its striking clustering of face-selective cells, was a unique evolutionary consequence of the importance of face recognition to primate social behavior. Here we show that the face system arises naturally from the statistical structure of object space. One prediction is that face-deprived animals should still show a network specialized for round objects (for example, clocks, apples), even if it is not specialized for faces per se. Selectivity for additional features may develop with face experience [2].

Our hypothesis that IT cortex is organized according to the first two dimensions of an object space makes multiple new predictions. We have already confirmed several of these, including the existence of the stubby network (see Appendix ii). Other predictions to consider include (1) that lesions in any part of IT should lead

to agnosias in specific sectors of object space [31], and (2) that other brain regions containing face patches [14] may also harbor object space maps. Additionally, it will be important to discover whether remaining unaccounted-for regions of IT can be explained by relating the behavior of the neurons within them to a map of object space.

4.A Methods

Five male rhesus macaques (*Macaca mulatta*) between 5 and 8 years old were used in this study. All procedures conformed to local and US National Institutes of Health guidelines, including the US National Institutes of Health Guide for Care and Use of Laboratory Animals. All experiments were performed with the approval of the Caltech Institutional Animal Care and Use Committee.

No statistical methods were used to predetermine sample size. The experiments were not randomized and investigators were not blinded to allocation during experiments and outcome assessment.

4.A.1 Visual stimuli

Stimuli for electrophysiology experiments Three stimulus sets were used. (1) A set of 51 objects from 6 object categories, each presented from 24 viewpoints (Fig. 4.7a, b). Non-face 3D models were downloaded from `https://www.3d66.com`. And face models were generated with Facegen (Singular Inversions) using random parameters. Images were rendered using 3dMax (Autodesk). Object images were presented for 250 ms windows interleaved with 150 ms windows in which solid gray was displayed. Each object image was presented 4–8 times. (2) A set of line segments that varied along three dimensions: curvature, aspect ratio, and orientation (Fig. 4.7c). Segment images were presented for 150 ms windows interleaved with 150 ms windows in which solid gray was displayed. Each image was presented 6–8 times. (3) A set of 1,593 object images consisting of 1,392 images downloaded from `www.freepngs.com` and 201 face images from the FEI database (`https://fei.edu.br/~cet/facedatabase.html`) (Fig. 4.7d). Object images were presented for 150 ms windows interleaved with 150 ms windows in which solid gray was displayed. Each image was presented 4–8 times.

Localizer for the NML network Preferred and non-preferred objects were identified from electrophysiological responses recorded in the NML network of monkey M1 (Fig. 4.2a, top) by computing average, baseline-subtracted responses in the window $[60, 220]$ ms after stimulus onset, averaging across all 24 views. (The baseline was computed from the window $[-25, 25]$ ms.) The localizer contained blocks of three types. Type 1 blocks contained images of the five most-preferred objects, each at eight views (0°rotation in the y-z space; first row in Fig. 4.7b). Type 2 blocks contained images of the five least-preferred objects, each at eight views. Type 3 blocks contained images of five objects that belonged to the animal

category, each at eight views. A block containing phase-scrambled noise patterns preceded each stimulus block (using the images shown in blocks 1–3). To construct phase-scrambled images, we performed discrete Fourier transforms (DFT) on the object images, added a random phase to each frequency component, and then performed an inverse DFT. During the fMRI experiment, stimuli were presented in 24-s blocks, with an interstimulus interval of 500 ms. In each scan, the order of the stimulus blocks was fixed as follows: preferred objects, non-preferred objects, animals, non-preferred objects, animals, preferred objects, animals, preferred objects, non-preferred objects. In addition, a block containing phase-scrambled noise was added at the end of each scan. Each scan lasted 456 s. Four monkeys were tested with this localizer, and 6–9 scans were performed for each monkey.

Localizer for the body network This localizer contained blocks of eight types, each consisting of 16 images taken from the following 8 categories: monkey bodies, animals, faces, fruits, hands, man-made objects, houses, and scenes. Stimuli were presented in 24-s blocks, with an interstimulus interval of 500 ms. In each run, the eight blocks were each presented once, interleaved with phase-scrambled noise patterns (computed using images from the eight object blocks). A block containing phase-scrambled noise was added at the end of each scan. Each scan lasted 408 s. Four monkeys were tested with this localizer, and 6–9 scans were performed for each monkey.

Localizer for the stubby network This localizer contained blocks of four types, each consisting of 20 images taken from the four quadrants of object PC1-PC2 space (Fig. 4.4a). The images were selected from a collection of 19,300 background-free object images (`http://www.freepngs.com`). The images were passed through AlexNet, and projected to the object PC1-PC2 space constructed using the original 1,224 images. (See "Building an object space using a deep network".) Then 20 images were selected from each of the four quadrants of object PC1-PC2 space, each with a polar angle roughly centered on the respective quadrant. The images were presented in 24-s blocks with an interstimulus interval of 500 ms. In each run, the four blocks were each presented twice, interleaved with phase-scrambled noise patterns (computed using images from the four object blocks). A block containing phase-scrambled noise was added at the end of each scan. Each scan lasted 408 s. Four monkeys were tested with this localizer, and 6–18 scans were performed for each monkey.

Localizer for the face network This localizer contained blocks of five types, containing images of faces, hands, technological objects, vegetables/fruits, and bodies. Face blocks were presented in alternation with non-face blocks. Stimuli were presented in 24-s blocks with an interstimulus interval of 500 ms. In each run, the face block was repeated four times and each of the non-face blocks was shown once. Blocks of grid-scrambled noise patterns preceded each stimulus block. A block containing grid-scrambled noise was added at the end of each scan. Each scan lasted 408 s. Additional properties of this localizer were as described in [36]. Four monkeys were tested with this localizer, and 5–12 scans were performed for each monkey.

Localizer for the scene network This localizer contained blocks of ten types: five scene block types and five non-scene block types. Stimuli were presented in 24-s blocks with an interstimulus interval of 500 ms. In each run, the ten blocks were each presented once, interleaved with blocks of grid-scrambled noise. Additional properties of this localizer were as described in [19]. Two monkeys were tested with this localizer, and 8–12 scans were performed for each monkey.

Localizer for the color network This localizer contained blocks of two types: color and grayscale. During color blocks, an equiluminant red/green color grating was presented (2.9 cycles/degree, drifting at 0.75 cycles/s). And during grayscale blocks, an otherwise-identical grayscale grating was presented. Stimuli were presented in 24-s blocks, 16 blocks to a run. Each scan lasted 432 s. Additional properties of this localizer were as described in [22, 6]. Four monkeys were tested with this localizer, and 8–14 scans were performed for each monkey.

Localizer for the 3D network This localizer contained blocks of two types. Blocks of the first type contained 3D shapes presented as random dot stereograms, including curved shapes, like ripples and saddles, and simple, flat shapes, like stars and squares. And blocks of the second type contained random dots, presented at zero disparity. Blocks of these two types were interleaved, and each block lasted 24 s. The images were presented with an interstimulus interval of 500 ms. Each scan lasted 600 s. Monkeys viewed the stimuli through red-green glasses. Four monkeys were tested with this localizer, and 5–12 scans were performed for each monkey.

Silhouette experiment This localizer contained blocks of four types, each consisting of 20 images taken from the four quadrants of the AlexNet fc6 PC1-PC2 space (Fig. 4.11c). The images were selected from an image set containing 19,300 background-free object images (images from `http://www.freepngs.com`). The images were first binarized by making any pixel that belonged to the object to black and any pixel that did not belong to the object white. Images were then passed through AlexNet and projected onto the PC1-PC2 space built using the original 1,224 images (see "Building an object space using a deep network"). Then, 20 images were selected from each of the four quadrants of PC1-PC2 space. The images were presented in 24-s blocks with an interstimulus interval of 500 ms. In each run, blocks of each type were presented twice, interleaved with blocks only showing a background and a fixation point. A block containing a background and a fixation point was added at the end of each scan. Each scan lasted 408 s. Three monkeys were tested with this localizer, and 12–24 scans were performed for each monkey.

Fake object experiment This experiment was largely identical to the silhouette experiment, but with different stimuli. We used a deep generative adversarial network (GAN) [9] to generate "fake object" images (Fig. 4.11d). The GAN was trained to generate images using AlexNet layer fc6 responses. To generate fake objects, we first passed an image set consisting of 19,300 real object images through AlexNet; for each object image, a 4,096-unit response pattern from layer fc6 was generated. We randomly selected pairs of patterns, and recombined these pairs into new patterns [29]. Each new pattern was passed into the GAN to generate a fake object image. Twenty thousand new "fake objects" were generated, and four groups of stimuli (twenty images per group) were selected from this set on the basis of the projections onto PC1-PC2 space. Three monkeys were tested with this localizer, and 10–32 scans were performed for each monkey.

Deep dream experiment This experiment was largely identical to the silhouette experiment, but with different stimuli. We used deep dream techniques (Matlab 2017b, Deep Learning Toolbox, `deepdreamImage` function) to generate images projecting strongly onto the four quadrants of object space. Instead of performing gradient ascent on activity of a single fc6 unit, four groups of images were generated by ascending the gradient of four fictive units corresponding to linear weighted sums of fc6 units (PC1 + PC2, PC1 − PC2, −PC1 + PC2, −PC1 − PC1) (Fig. 4.11e). For each fictive unit, 20 images were generated after 100 iterations of gradient ascent,

starting from Gaussian noise. We further confirmed that the images projected to extreme coordinates in PC1-PC2 space by passing the images through AlexNet and projecting the resulting fc6 response pattern onto PC1-PC2 space. Three monkeys were tested with this localizer, and 12–22 scans were performed for each monkey.

4.A.2 fMRI scanning and analysis

Five male rhesus macaques were trained to maintain fixation on a small spot for a juice reward. Eye position was monitored using an infrared camera (ISCAN) sampled at 120 Hz. Monkeys were scanned in a 3T TIM (Siemens, Munich, Germany) magnet equipped with an AC88 gradient insert while passively viewing images on a screen. Feraheme contrast agent was injected to improve the signal/noise ratio for functional scans. A single-loop coil was used for structural scans at isotropic 0.5 mm resolution. A custom eight-channel coil was used for functional scans at isotropic 1 mm resolution. Additional details of the scanning protocol are described in [28].

Surface reconstruction based on anatomical volumes was performed using FreeSurfer [8] after skull stripping using FSL's Brain Extraction Tool (University of Oxford). After applying these tools, segmentations were further refined manually.

Analysis of functional volumes was performed using the FreeSurfer Functional Analysis Stream [32]. Volumes were corrected for motion and undistorted based on acquired field map. The resulting data was analysed using a standard general linear model. For the scene contrast, the average of all scene blocks was compared to the average of all non-scene blocks. For the face contrast, the average of all face blocks was compared to the average of all non-face blocks. For the color contrast, the color block was compared to the non-color blocks. For the body contrast, monkey body and animal blocks were compared to all other blocks. For the stubby contrast, the stubby, inanimate object block was compared to the three other blocks. For the 3D contrast, the 3D shape blocks were compared to the zero disparity blocks. For the microstimulation contrast, blocks with concomitant electrical stimulation were compared to blocks without stimulation. All of the contrasts were performed using a non-paired two-sided t-test. p values were not adjusted for multiple comparisons.

To determine the area of TE and TEO in each subject, we first co-registered the MRI volume for each subject to a monkey atlas [33]. Then, each subject's TE and TEO were defined using the atlas.

To quantify the reproducibility of patch progression on the cortical surface, we plotted significance values for the four stimulus comparisons defining the four networks in

Fig. 4.4c along three paths in posterior, middle, and anterior IT tracing the centre of the gray matter, spanning the following ranges: (1) the lower bank of STS and inferotemporal gyrus at AP position 3; (2) the lower bank of STS and inferotemporal gyrus at AP position 13; (3) antero-dorsal (TEad) and antero-ventral (TEav) parts of area TE at AP position 18. Non-significant responses ($p > 10^{-3}$) were set to 0.

4.A.3 Microstimulation

The stimulation protocol followed a block design. We interleaved nine blocks of fixation with eight blocks of fixation paired with electrical microstimulation. We started and ended with a block without microstimulation, and each block lasted 32 s. During microstimulation blocks, we applied one pulse train per second, lasting 200 ms, with a pulse frequency of 300 Hz. Bipolar current pulses were charge-balanced, with a phase duration of 300 μs and an inter-phase distance of 150 μs. We used a current amplitude of 300 μA. Stimulation pulses were delivered using a computer-triggered pulse generator (S88X; Grass Technologies) connected to a stimulus isolator (A365, World Precision Instruments). All stimulus-generation equipment was stored in the scanner control room, and the coaxial cable was passed through a wave guide into the scanner room. We obtained 30 scans for monkey M1.

4.A.4 Single-unit recording

Tungsten electrodes (1–20 MΩ at 1 kHz, FHC) were back-loaded into plastic guide tubes. The guide tube length was set to reach approximately 3–5 mm below the dura surface. The electrode was advanced slowly using a manual advancer (Narishige Scientific Instrument, Tokyo, Japan). Neural signals were amplified and extracellular action potentials were isolated using the box method in an online spike sorting system (Plexon, Dallas, TX, USA). Spikes were sampled at 40 kHz. All spike data was re-sorted using offline spike sorting algorithms (Plexon). We recorded data from every neuron encountered. Only well-isolated units were considered for further analysis. Electrodes were lowered through custom angled grids that allowed us to reach the desired targets; custom software was used to design the grids and plan the electrode trajectories [27].

4.A.5 Behavioral task

Monkeys were head fixed and passively viewed the screen in a dark Wisconsin box. Stimuli for electrophysiology were presented on a CRT monitor (DELL P1130). The screen size covered 27.7×36.9 visual degrees and the stimuli spanned 5.7°. The

fixation spot size was 0.2°in diameter. Images presentation order was randomized using custom software. Eye position was monitored using an infrared eye tracking system (ISCAN). Juice reward was delivered every 2–4 s if fixation was properly maintained.

4.A.6 Data analysis

Computing view-identity similarity matrices For each network, we first identified the 11 most preferred objects by computing average, baseline-subtracted responses in the window $[60, 220]$ ms after stimulus onset, averaging across all 24 views. (The baseline was computed from the window $[-25, 25]$ ms.) We then used responses to these 11 most preferred objects for the analysis (using all 24 views for each object, coming to a total of 264 images). For each patch, we computed a 264×264 similarity matrix containing Pearson's correlation coefficients for each population response vector pair. To compute view-invariant identity selectivity as a function of time (Fig. 4.8f), at each time point t between 0 and 400 ms following stimulus onset, in increments of 50 ms, we computed a similarity matrix for mean responses between $t - 25$ and $t + 25$ ms. We then calculated a "same object correlation value" as the average correlation between responses to two distinct images of a single object (solid traces in Fig. 4.8f), and a "different object correlation value" as the average correlation between responses to images of two different objects (dashed traces in Fig. 4.8f).

Building an object space using a deep network The stimulus set consisting of 51 objects at 24 views (1,224 images) was fed into the pre-trained network AlexNet [20]. The responses of 4,096 nodes in layer fc6 were then extracted to form a 1,224×4,096 matrix. We performed PCA on this matrix and retained the first 50 PCs, which captured 85% of the response variance across AlexNet fc6 units. The first two dimensions accounted for 27% of the response variance across AlexNet fc6 units. To test the robustness of object PC1-PC2 space to the particular set of 1,224 images used to build it (Fig. 4.9d, e), we randomly selected 1,224 images from a new database (`http://www.freepng.com`) containing 19,300 background-free object images. These images were fed into Alexnet, and we followed the same procedure to build a new object space, which we call PC1′-PC2′ space. The original 1,224 images were passed through Alexnet, and the vector of fc6 unit activations was projected onto both PC1-PC2 space and PC1′-PC2′ space. We then determined the affine transform of PC1′-PC2′ space that minimized the distance between the coordinates

of the images in the two spaces, using linear regression.

$$\begin{pmatrix} x_{1,1} & x_{1,1} \\ & \vdots & \\ x_{1224,1} & x_{1224,1} \end{pmatrix} = \begin{pmatrix} x'_{1,1} & x'_{1,1} \\ & \vdots & \\ x'_{1224,1} & x'_{1224,1} \end{pmatrix} \begin{pmatrix} u & w \\ u' & w' \end{pmatrix}, \tag{4.1}$$

where $(x_{i,1}, x_{i,2})$ is the location of image i in PC1-PC2 space, and $(x'_{i,1}, x'_{i,2})$ is the location of image i in PC1′-PC2′ space. After aligning the two spaces, we calculated the Pearson's correlation r between PC1 and PC1′, and PC2 and PC2′. We used a similar procedure to test the robustness of PC1-PC2 space to the particular network used to compute it (Fig. 4.9i).

Quantifying the aspect ratio of objects The aspect ratio of an object (Fig. 4.8g) was defined as a function of its perimeter P and area A:

$$\text{Aspect ratio} := \frac{P^2}{4\pi A}. \tag{4.2}$$

The perimeter was defined as the number of pixels lying on the object image's boundary, which was computed using Matlab's `bwboundaries` function. The area was defined as the number of pixels that fell within the object's boundary, including those at the boundary, which was computed using Matlab's `regionsprops` function.

Computing the preferred axis of an IT cell We counted the number of spikes that occurred in the time window 60–220 ms after stimulus onset for each stimulus. To estimate a cell's the preferred axis, we used linear regression to compute the coefficients $\mathbf{c}$ in the equation $\mathbf{R} = \mathbf{c} \cdot \mathbf{F} + c_0$, where $\mathbf{R}$ is the response vector of the cell to the set of images, $\mathbf{F}$ is the matrix of 50D object feature vectors for the set of images, and c_0 is a constant offset. Using this definition of preferred axis, an increased value of the projection onto the preferred axis corresponds to an increased firing rate. To generate Fig. 4.3c, we randomly picked half of the stimulus trials to compute the preferred axis for each cell, and then used the held-out data to plot the responses shown.

Computing tuning along dimensions orthogonal to the preferred axis To compute tuning along orthogonal dimensions (Fig. 4.10e, black traces) for each neuron, we first computed the preferred axis. There are 49 dimensions spanning the subspace orthogonal to this preferred axis. To find the longest orthogonal axis in this 49D subspace, we first represented each of the 1,224 images in our stimulus set as a

50D vector in the object space, and subtracted the preferred axis projection from each of these image feature vectors, to obtain a set of feature vectors lying in the 49D orthogonal subspace. We performed PCA on this set of 1,224 vectors, and picked the top PC. This PC represents the axis orthogonal to the preferred axis of the cell along which the images vary the most. For each cell, we computed the tuning curve along this axis.

Quantifying consistency of a cell's preferred axis The consistency of the preferred axis of each cell (Fig. 4.10a) was measured as follows: in each iteration, we randomly split the image set (1,224 images) into two subsets of 612 images, and calculated a preferred axis using the responses to each subset. Then, we computed the Pearson correlation (r) between the two. We repeated this 100 times, and the consistency of the preferred axis for the cell was defined as the average r value across the 100 iterations.

Quantifying explained variance along an object dimension In Fig. 4.16b, c, the explained variance R^2 was computed

$$R^2 = \frac{\sum_{i\in\{1..1224\}}(y_i - y_i')^2}{\sum_{i\in\{1..1224\}}(y_i - \bar{y})^2}, \tag{4.3}$$

where y_i is the true feature value for image i, y_i' is the reconstructed feature value for image i, and $\bar{y}$ is the mean true feature value across the 1224 images.

Quantifying explained variance in single-neuron firing rates and model comparison In Fig. 4.10b–d, to compute explained variance, we first fit the axis model to responses to 1,593 objects (Fig. 4.7d), and then tested it model on responses to 100 other objects. To obtain high signal quality, the test objects were repeated 15–30 times. In Fig. 4.10c, d, we compared three models: (1) the axis model, which assumed the 50D features were combined linearly; (2) a Gaussian model, $R = a\exp(-(\mathbf{x}-\mathbf{x}_0)^2/\sigma^2)$, and (3) a quadratic model, $R = a(\mathbf{x}-\mathbf{x}_0)^2 + b(\mathbf{x}-\mathbf{x}_0) + c$. The fraction of explainable variance in responses to the test objects explained by each model was used to quantify the quality of the fit. In Fig. 4.10b, for each cell, the explained variance R^2 was computed

$$R^2 = \frac{\sum_{i\in\{1..100\}}(r_i - r_i')^2}{\sum_{i\in\{1..100\}}(r_i - \bar{r})^2}, \tag{4.4}$$

where r_i is the true response to image i, r_i' is the predicted response to image i, and $\bar{r}$ is the mean true response across the 100 test images. When calculating the upper

bound of the explained variance (y-axis values in Fig. 4.10b), the responses from half of the presentations of each image (selected randomly) were used to fit the model, and the remaining data was used to test it. We calculated the Pearson correlation (r) across splits and corrected it using the Spearman-Brown correction,

$$r' = \frac{2r}{r+1}. \tag{4.5}$$

The square of r' was considered the upper bound for the explained variance.

k-means cluster analysis To determine whether neurons were grouped based on their preferred axes, we applied k-means clustering on the entire population of neurons recorded from the four networks (Fig. 4.13g, h). The distance between each pair of neurons was calculated as the Pearson's correlation between preferred axes of the neurons in the 50D object space. To determine the optimal number of clusters to use, we calculated the Calinski-Harabasz value CH for multiple cluster counts. For a given cluster count k, CH is defined

$$\mathrm{CH}(k) := \frac{B(k)\,(n-k)}{w(k)\,(k-1)}, \tag{4.6}$$

where $B(k)$ is the between-cluster variation, $w(k)$ is the within-cluster variation, and n is the number of neurons being clustered. We used the cluster count k that maximized CH. To check whether clusters existed beyond the first two PCs, we computed each neuron's preferred axis in the space spanned by PCs 3–50, and then performed k-means clustering using the correlations between these new preferred axes.

Decoding analysis We found that cells in each IT network could be well modelled using the equation $\hat{\mathbf{r}} = \mathbf{C}\,\mathbf{f} + \mathbf{c}_0$, where $\hat{\mathbf{r}}$ is the vector of estimated neural responses, $\mathbf{C}$ is a matrix of weight coefficients, $\mathbf{f}$ is the feature vector in the object space, and $\mathbf{c}_0$ is an offset vector (Fig. 4.3c, Fig. 4.10a, e). This suggests that by inverting this equation, we should be able to decode a feature vector in the object space from an IT response vector, $\hat{\mathbf{f}} = \mathbf{C}'\,\mathbf{r} + \mathbf{c}'_0$. We used responses to all but one object (1,224 - 24 = 1,200 images) to fit $\mathbf{C}'$ and $\mathbf{c}'_0$. Then we applied the linear model to responses to the remaining object for each of the 24 views to compute the predicted feature vector (Fig. 4.5, Fig. 4.16). To quantify the decoding accuracy (Fig. 4.16d–f), we randomly selected images from the set of 1,224 object images and compared their true feature vectors to the reconstructed feature vector for a target image. If the nearest neighbor of the target's reconstruction (using the Euclidean distance) corresponded to the

same object as the target, then the decoding was considered correct. We repeated this procedure 100 times for each of the 1,224 object images to estimate the overall decoding accuracy.

Object reconstruction We used a pre-trained GAN [9] to reconstruct objects from neural activity vectors (Fig. 4.5). For each image, we estimated a 50D object feature vector from neural activity elicited by the image, and then transformed the 50D object feature vector into a layer fc6 response pattern using the Moore-Penrose pseudoinverse. We passed this fc6 response pattern into the generative network to generate image-space reconstructions. Since the generative network cannot perfectly reconstruct images from AlexNet fc6 layer responses, for comparison we also generated reconstructions for each image using (1) its original fc6 response pattern and (2) its original fc6 response pattern projected onto the 50D object space; with the latter serving as an upper bound on the reconstruction quality that can be attained using our approach. We computed a "normalized distance" to quantify the reconstruction accuracy for each object:

$$\text{Normalized distance} := \frac{|\mathbf{fc6}_{\text{recon}} - \mathbf{fc6}_{\text{original}}|}{|\mathbf{fc6}_{\text{bestPossibleRecon}} - \mathbf{fc6}_{\text{original}}|}, \tag{4.7}$$

where $\mathbf{fc6}_{\text{recon}}$ is the fc6 response to the reconstruction obtained from the neural data, $\mathbf{fc6}_{\text{original}}$ is the fc6 response to the image originally shown to the monkey, and $\mathbf{fc6}_{\text{bestPossibleRecon}}$ is the fc6 response to the best possible reconstruction. As an alternative to reconstructing images using a GAN, we recovered images using an auxiliary database (Fig. 4.16g, h). We passed an image set containing 18,700 background-free object images (`http://www.freepngs.com`) and 600 face images (FEI database), none of which had been shown to the monkey, through AlexNet, and projected the fc6 responses onto the object space computed using our original stimulus set of 1,224 images. For each image, the object feature vector reconstructed from neural activity was compared with the object feature vectors for images from the new image set. The image in the new image set with the smallest Euclidean distance to the reconstructed object feature vector was considered to be the reconstruction of this object feature vector. To take into account the fact that the object images used for reconstruction did not include any of the object images shown to the monkey, limiting how good the reconstruction can be, we computed a normalized distance to quantify the reconstruction accuracy for each object:

$$\text{Normalized distance} := \frac{|\mathbf{v}_{\text{recon}} - \mathbf{v}_{\text{original}}|}{|\mathbf{v}_{\text{bestPossibleRecon}} - \mathbf{v}_{\text{original}}|} \tag{4.8}$$

where $\mathbf{v}_{\text{recon}}$ is the feature vector reconstructed from neuronal responses, $\mathbf{v}_{\text{original}}$ is the feature vector corresponding to the image originally presented to the monkey, and $\mathbf{v}_{\text{bestPossibleRecon}}$ is the feature vector corresponding to the best possible reconstruction. A normalized distance of 1 means that the best possible reconstruction has been found.

Object specialization index computation To quantify whether particular objects were better-represented by particular networks (Fig. 4.16i), we computed a specialization index for each of the 1,224 objects and each of the three networks (body, NML, stubby). For object i and network j, the specialization index SI_{ij} is defined

$$\text{SI}_{ij} := \frac{\text{DA}(i, j, n) - \text{DA}(i, \sim j, n)}{\text{DA}(i, j, n) + \text{DA}(i, \sim j, n)}, \tag{4.9}$$

where n is a sample size, $\text{DA}(i, j, n)$ is the decoding accuracy for object i computed using n neurons randomly selected from network j, and $\text{DA}(i, \sim j, n)$ is the decoding accuracy for object i computed using n neurons randomly selected from networks other than network j. SI_{ij} quantifies the extent to which network j is specialized for representing object i.

4.B Additional figures

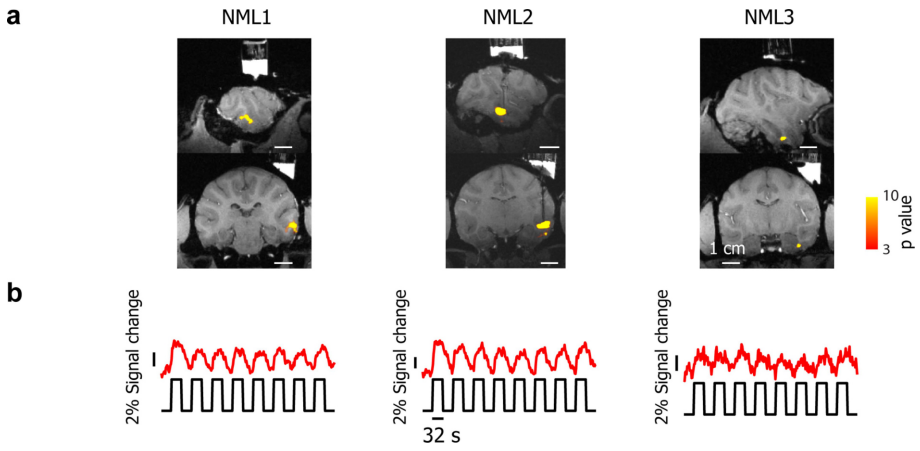

Figure 4.6: **Time courses from NML1–3 during microstimulation of NML2. a:** Sagittal (top) and coronal (bottom) slices showing activation in response to microstimulation of NML2. The dark track shows the electrode targeting NML2. **b:** Time course of microstimulation (black) and the fMRI response (red) from each of the three patches in the NML network.

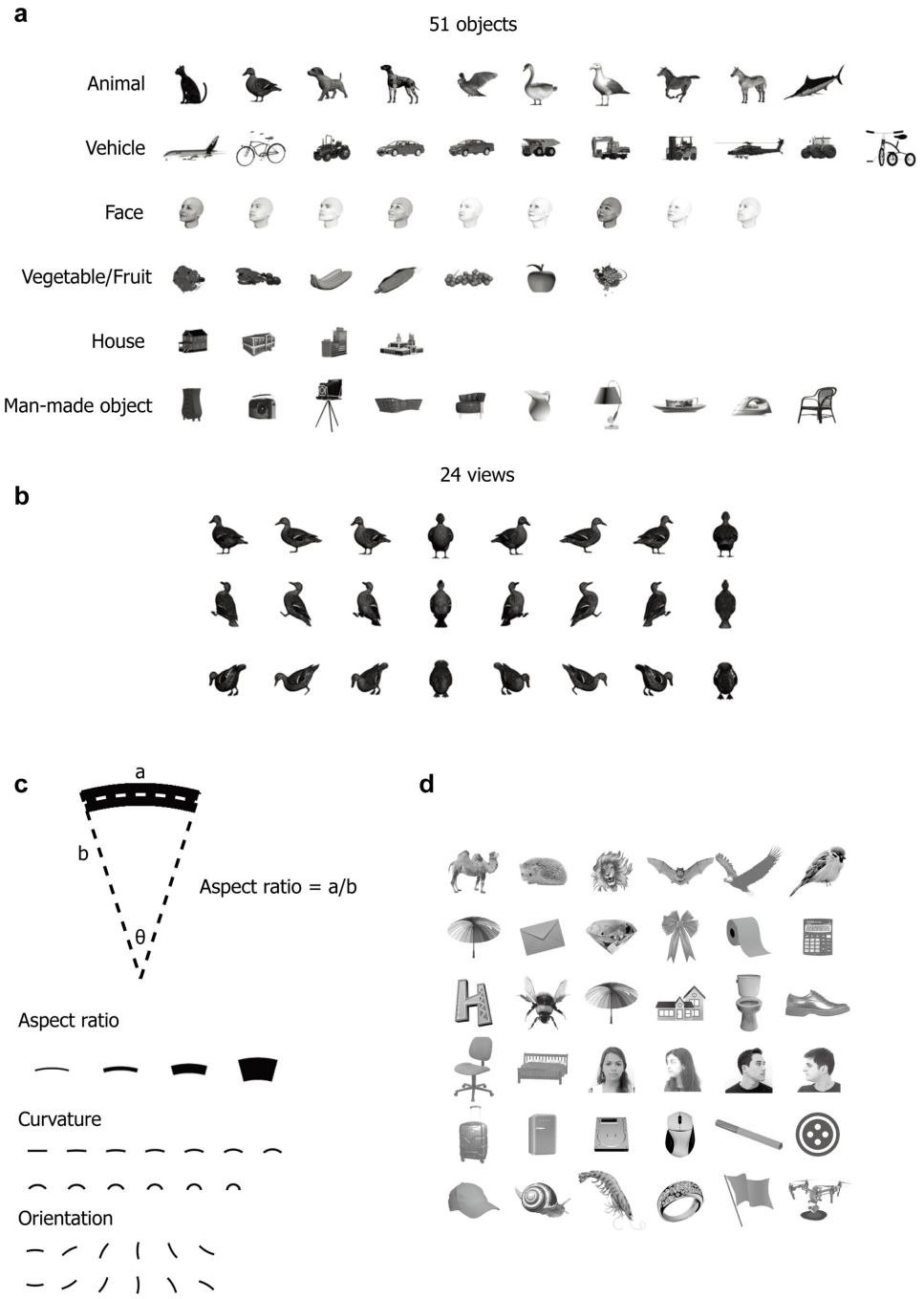

Figure 4.7: **Stimuli used in electrophysiological recordings. a:** 51 objects from 6 categories were shown to monkeys. **b:** 24 views for one example object, resulting from rotations in the *x*-*z* plane (abscissa) combined with rotations in the *y*-*z* plane (ordinate). **c:** A line segment that was parametrically varied along 3 dimensions was used to test the hypothesis that cells in the NML network are selective for aspect ratio (4 aspect ratio levels × 13 curvature levels × 12 orientation levels). **d:** 36 example object images from our 1,593-image stimulus set.

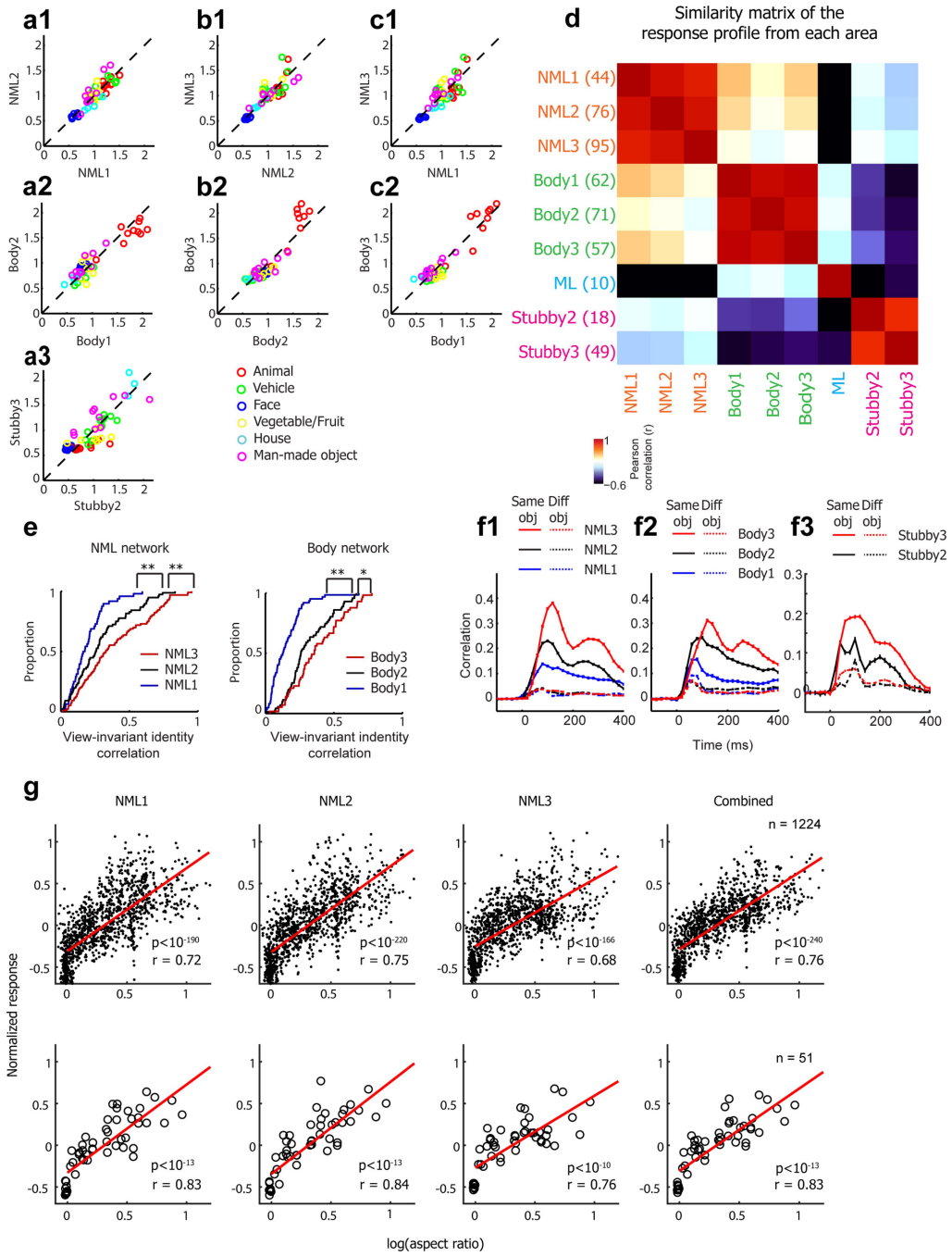

Figure 4.8: **Additional neuronal response properties across the patches. a1:** Average responses to 51 objects across all cells from patch NML2 plotted against those from patch NML1. The response to each object was defined as the average response across 24 views and across all cells recorded from a given patch. **b1:** As in (a1), for NML3 against NML2. **c1:** As in (a1,) for NML3 against NML1. **a2, b2, c2:** As in (a1), (b1), and (c1), for three patches in the body network. **a3:** As in (a1), for Stubby3 against Stubby2. **d:** A similarity matrix showing the Pearson correlation values (r) between the average responses to 51 objects from 9 patches across 4 networks. (Continued on the next page)

Fig. 4.8, continued: **e, left:** Cumulative distributions of view-invariant identity correlations for cells in the three patches of the NML network. **e, right:** As on the left, for cells in the three patches of the body network. For each cell, the view-invariant identity correlation was computed as the average correlation between response vectors across all view pairs. The distribution of view-invariant identity correlations was significantly different between NML1 and NML2 (two-tailed t-test, $p < 0.005$, $t(118) = 2.96$), NML2 and NML3 (two-tailed t-test, $p < 0.005$, $t(169) = 2.9$), Body1 and Body2 (two-tailed t-test, $p < 0.0001$, $t(131) = 6.4$), and Body2 and Body3 (two-tailed t-test, $p < 0.05$, $t(126) = 2.04$). $*p < 0.05$; $**p < 0.01$. **f1:** The time course of view-invariant object identity selectivity for the three patches in the NML network, computed using responses to 11 objects at 24 views and a 50-ms sliding response window (solid lines). As a control, time courses of correlations between responses to different objects across different views were also computed (dashed lines) (see Methods). **f2:** As in (f1), for the body network. **f3:** As in (f1), for the stubby network. **g, top:** Average responses to each image across all cells recorded from each patch plotted against the logarithm of the aspect ratio of the object in each image (see Methods). Pearson correlation values are indicated in each plot (all $p < 10^{-10}$). The rightmost column shows results with cells from all three patches grouped together. **g, bottom:** As on top, with responses to each object averaged across 24 views, and the corresponding aspect ratios also averaged. The rightmost column shows results with cells from all three patches grouped together.

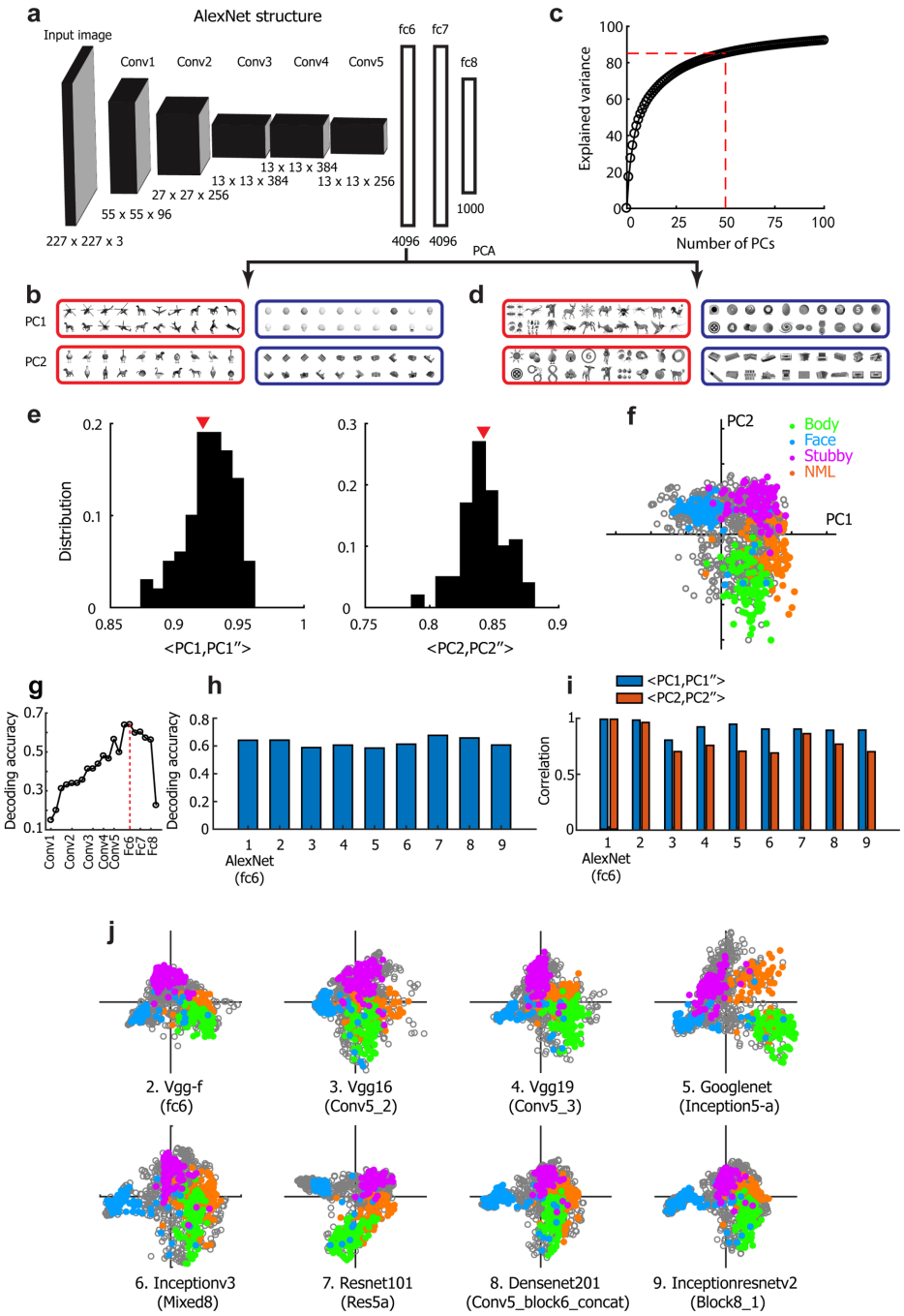

Figure 4.9: **Building an object space using a deep network. a:** A diagram illustrating the structure of AlexNet6. Five convolution layers are followed by three fully connected layers. The number of units in each layer is indicated below it. **b:** Images with extreme values (highest: red, lowest: blue) of PC1 and PC2. **c:** The cumulative explained variance of responses of units in fc6 by 100 PCs; 50 dimensions explain 85% of variance. (Continued on the next page)

Fig. 4.9, continued: **d:** Images in the 1,593-image set with extreme values (highest: red, lowest: blue) of PC1 and PC2 (see Methods). Preferred features are generally consistent with those computed using the original image set shown in (b). However, PC2 no longer clearly corresponds to an animate-inanimate axis; instead, it corresponds to curved versus rectilinear shapes. **e:** Distributions showing the canonical correlation value between the first two PCs obtained by the 1,224-image set and the first two PCs constructed using other image sets (1,224 randomly selected non-background object images; left: PC1, right: PC2; see Methods for details). The red triangles indicate the arithmetic mean of the distributions. **f:** We passed 19,300 object images through AlexNet and constructed the PC1-PC2 space using PCA. Then we projected 1,224 images onto this space. The top 100 images for each network are indicated by colored dots (compare Fig. 4.4b). **g:** Decoding accuracy for 40 images using object spaces constructed using responses of different layers of AlexNet (computed as in Fig. 4.16d). There are multiple points for each layer because we performed PCA at multiple points in the pooling, activation, and normalization progression within individual layers. Layer fc6 yielded the highest decoding accuracy, motivating our use of the object space generated by this layer throughout the paper. **h:** To compare IT clustering using AlexNet with clustering using other deep network architectures, we first identified the layer of each network that yielded the best decoding accuracy, as in (g). The bar plot shows the decoding accuracy for 40 images in 9 deep networks using the best-performing layer for each network. **i:** Canonical correlation values between the first two PCs obtained by Alexnet and first two PCs built using 8 other deep networks (labelled 2-9). The layer of each network that yielded the highest decoding accuracy for a sample of 40 images was used for this analysis. The name of each network and layer can be found in (j). **j:** As in Fig. 4.4b, using principal components computed using 8 other networks.

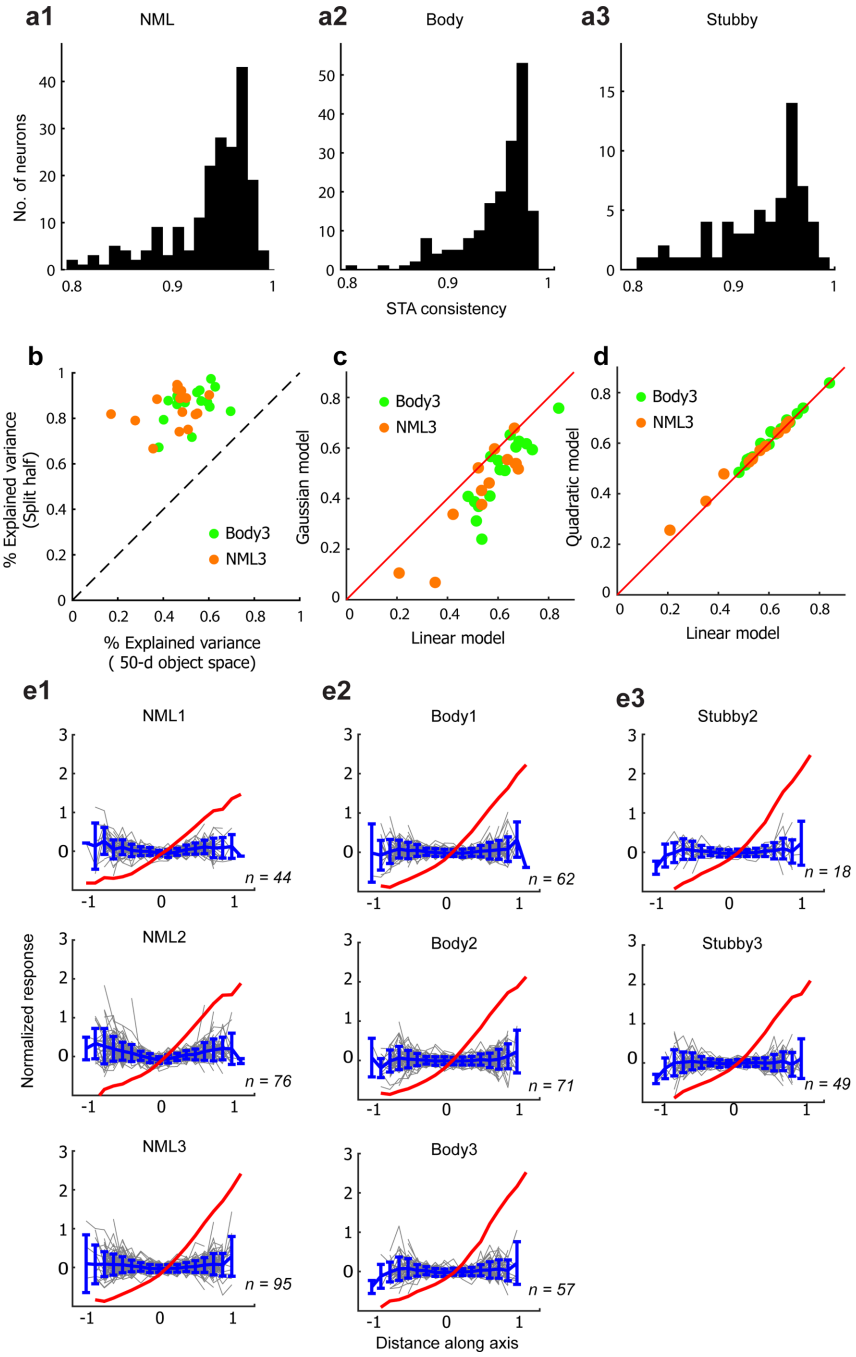

Figure 4.10: **Axis coding in neurons across IT. a1:** The distribution of preferred-axis consistency for cells in the NML network (see Methods). **a2:** As in (a1), for the body network. **a3:** As in (a1), for the stubby network. **b:** The set of responses recorded for each image was split in half, and the average response in one half of the trials was used to predict the average response in the other. Percentage of variance explained, after Spearman-Brown correction (mean 87.8%), is plotted against the percentage of variance explained by the axis model (mean 49.1%). The mean explainable variance across the 29 cells was 55.9%. (Continued on the next page)

Fig. 4.10, continued: **c:** Percentage of variance explained by a Gaussian model, plotted against the percentage of variance explained by the axis model. **d:** Percentage variances explained by a quadratic model, plotted against the percentage of variance explained by the axis model. Inspecting the quadratic model coefficients revealed a negligible quadratic term. (The mean ratio of second-order coefficients to first-order coefficient was 0.028.) **e1, top:** The red line shows the average modulation along the preferred axis across the population of NML1 cells. The grey lines show, for each cell in NML1, the modulation along the single axis orthogonal to the preferred axis in the 50D object space that accounts for the most variability. The blue line and error bars represent the mean and standard deviation, respectively. **e1, middle:** An analogous plots for NML2. **e1, bottom:** An analogous plots for NML3. **e2:** As in (e1), for the three body patches. **e3:** As in (e1), for the two stubby patches.

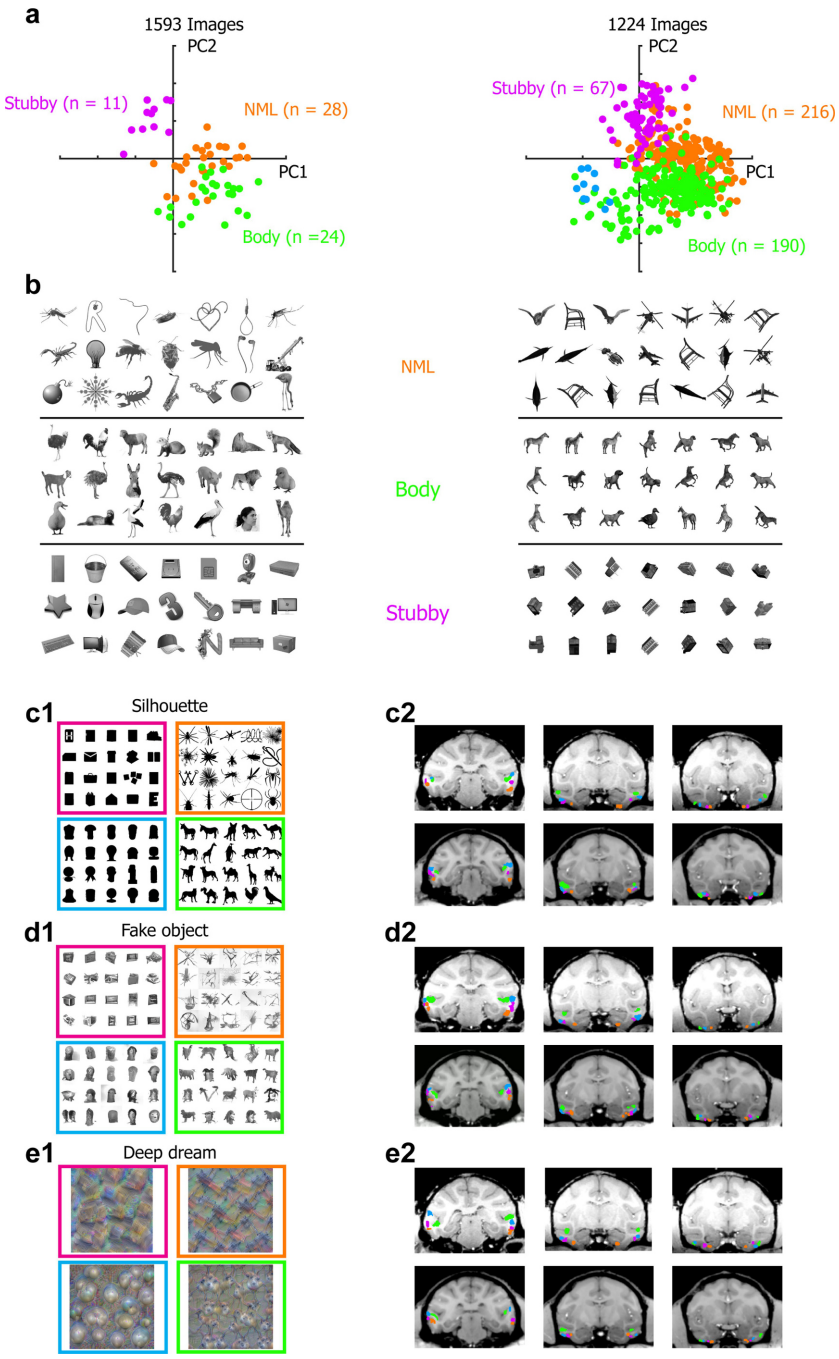

Figure 4.11: **Similar functional organization observed using a different stimulus set. a:** Projection of preferred axes onto PC1 and PC2 for all neurons recorded using two stimulus sets (left: 1,593 images from `freepngs.com`; right: the original 1,224 images of 51 objects × 24 views). The PC1-PC2 space for both plots was computed using the 1,224-image set. Different colors encode neurons from different networks. **b:** The top-21 preferred stimuli based on average responses from the neurons recorded in the three networks. (Continued on the next page)

Fig. 4.11, continued: **c1:** Silhouette images that project strongly onto the four quadrants of the object space. **c2:** Coronal slices from posterior, middle, and anterior IT of monkeys M2 and M3 showing the spatial arrangement of the four networks revealed using the silhouette images in (c1), in an experiment analogous to that illustrated in Fig. 4.4a. **d1:** "Fake object" images that project strongly onto the four quadrants of the object space. Note that fake objects that project onto the face quadrant do not resemble real faces. **d2:** As in (c2), with fake object images from (d1). **e1:** Stimuli generated using deep dream techniques that project strongly onto the four quadrants of object space. **e2:** As in (c2), with deep dream images from (e1). The results shown in (c)–(e) support the idea that IT is organized according to the first two axes of an object space, rather than low-level features or semantics.

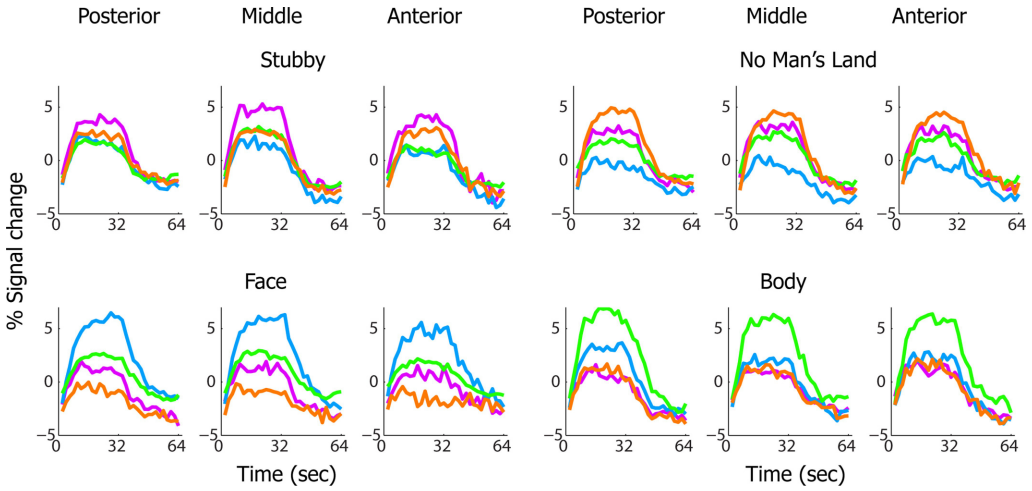

Figure 4.12: **Response time courses from the four IT networks spanning object space.** Time courses were averaged across two monkeys. To avoid selection bias, odd runs were used to identity regions of interest, and even runs were used to compute average time courses from these regions.

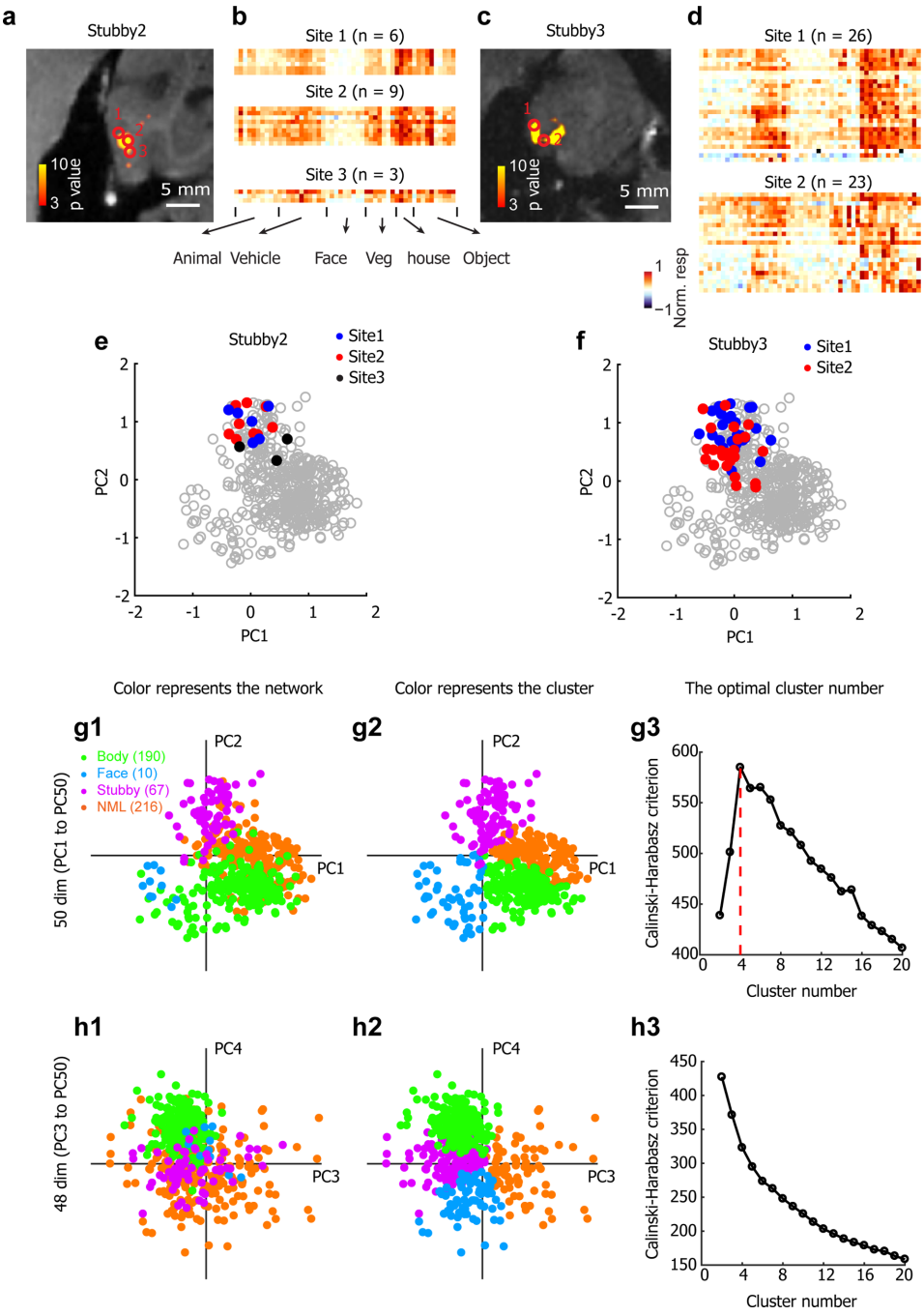

Figure 4.13: **Searching for substructure within patches. a:** Axial view of the Stubby2 patch, together with projections of three recording sites. **b:** Mean responses to 51 objects from neurons recorded at the sites shown in (a), grouped by recording site (same format as Fig. 4.2a, top). **c:** Axial view of the Stubby3 patch, together with projections of two recording sites. **d:** Mean responses to 51 objects from neurons recorded at the sites shown in (c), grouped by recording site. The grey dots represent the other neurons recorded across the four networks. (Continued on the next page)

Fig. 4.13, continued: **e:** Projections of the preferred axes of Stubby2 patch neurons onto PC1-PC2 space. There is no clear separation between neurons from the three sites in PC1-PC2 space. **f:** As in (e), for cells recorded from two sites in the Stubby3 patch. **g1:** PPC1-PC2 projections of the preferred axes of all recorded neurons. Different colors encode neurons from different networks. **g2:** As in (g1), but the color represents the cluster to which the neurons belong. Clusters were constructed using k-means clustering, with the cluster count set to four, and the distance between neurons defined as the correlation between preferred axes in the 50D object space (see Methods). Comparing (g1) and (g2) reveals a high degree of similarity between the anatomical and functional clustering of IT networks. **g3:** Calinski-Harabasz criterion values were plotted against the number of clusters for k-means clustering performed with different cluster counts (see Methods). The optimal cluster count is four. **h1:** As in (g1), for projections of preferred axes onto PC3 and PC4. **h2:** As in (h1), but the color represents the cluster to which the neurons belong. Clusters were constructed using k-means clustering, with the cluster count set to four, and the distance between neurons defined by the correlation between preferred axes in the 48D object space obtained by removing the first two dimensions. The difference between (h1) and (h2) suggests that there is no anatomical clustering for dimensions beyond the first two PCs. **h3:** As in (g3), with k-means clustering in the 48D object space. By the Calinski-Harabasz criterion, there is no functional clustering for dimensions beyond the first two.

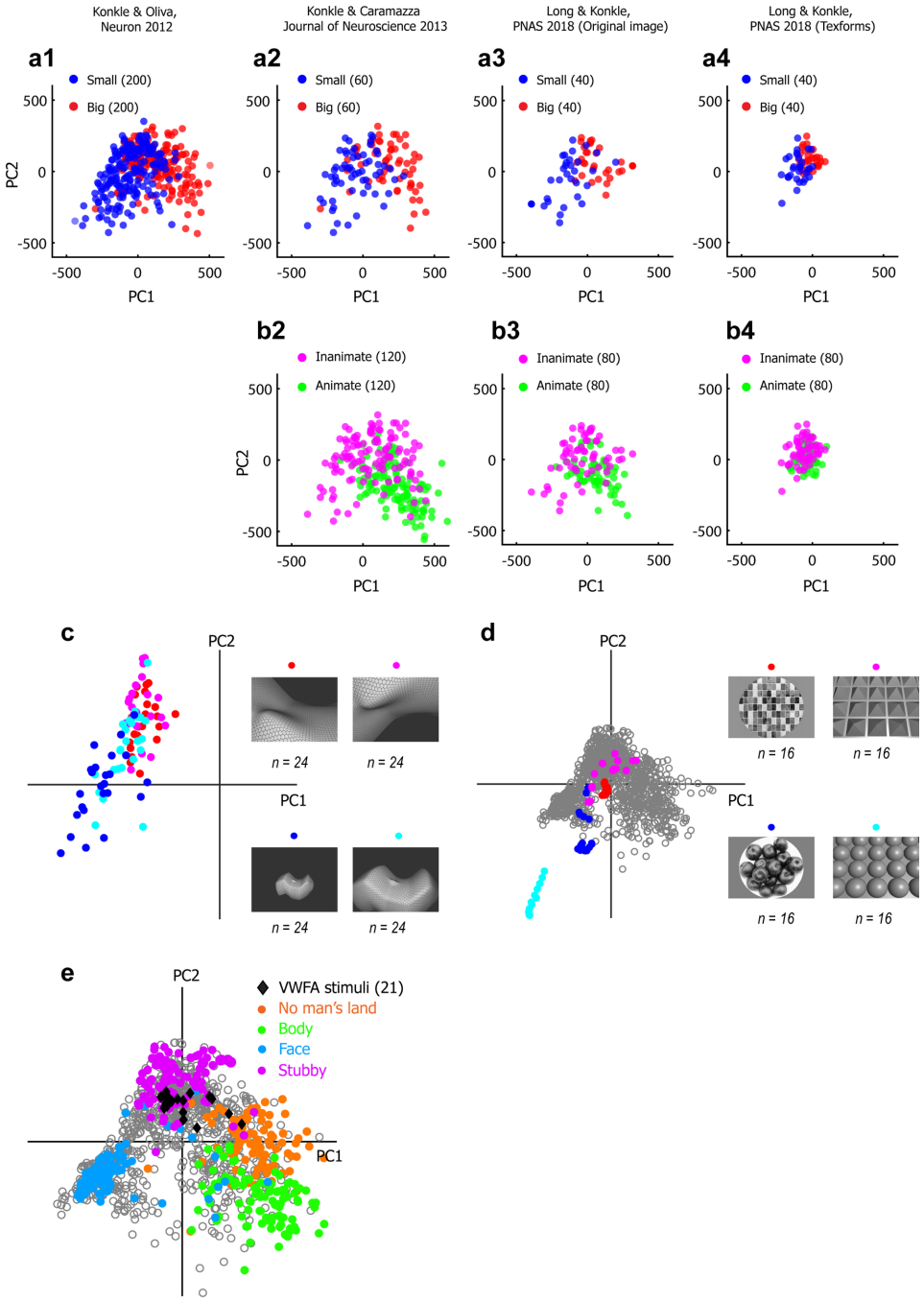

Figure 4.14: **Relating the object space model to previous accounts of IT organization. a1:** The object images used in [18] are projected onto PC1-PC2 space (computed as in Fig. 4.4b, by first passing each image through AlexNet). A clear gradient from large (red) to small (blue) objects is seen. **a2:** As in (a1), for the inanimate objects (large and small) used in [17]. **a3:** As in (a1), for the original object images used in [24]. **a4:** As in (a1), for the texform images used in [24]. **b2–b4:** Projection of animate and inanimate images from original object images (b2, b3) and texforms (b4). (Continued on the next page)

Fig. 4.14, continued: **c, left:** Colored dots depict the projection of stimuli from the four conditions used in [38]. **c, right:** Example stimuli (blue: small object-like; cyan: large object-like; red: landscape-like; magenta: cave-like). **d, left:** Grey dots depict 1,224 stimuli projected onto object PC1-PC2 space; colored dots depict the projection of stimuli from the four blocks of the curvature localizer used in [41]. **d, right:** Example stimuli from the four blocks of the curvature localizer (blue: real-world round shapes; cyan: computer-generated 3D sphere arrays; red: real-world rectilinear shapes; magenta: computer-generated 3D pyramid arrays). **e:** Images of English and Chinese words projected onto object PC1-PC2 space (black diamonds), superimposed on the plot from Fig. 4.4b. The projections are grouped within a small region, consistent with the hypothesis that the visual word form area is specialized to represent stimuli in a particular region in the object space.

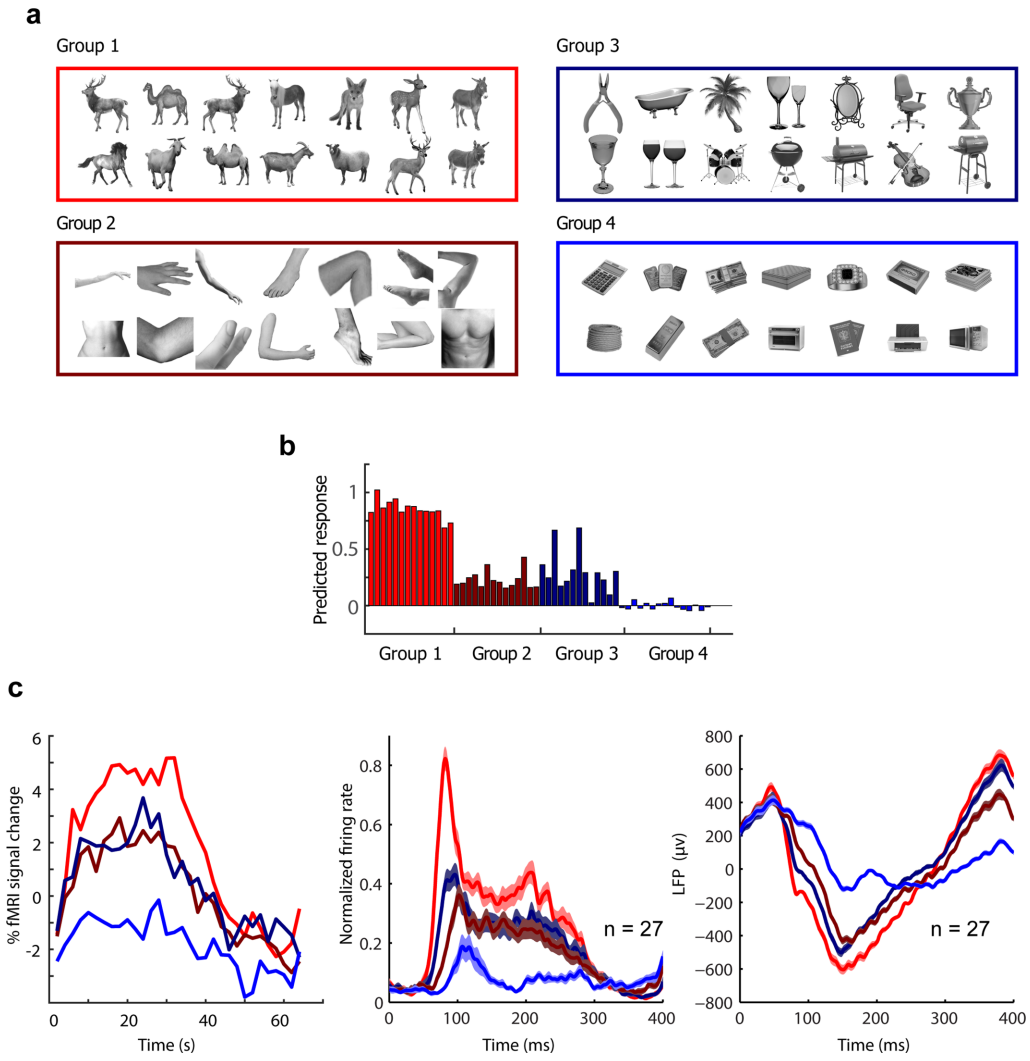

Figure 4.15: **Comparing object space dimensions to category labels as descriptor of response selectivity in the body patch. a:** Four classes of stimuli: (1) body stimuli that project strongly onto the body quadrant of object space (bright red), (2) body stimuli that project weakly onto the body quadrant of object space (dark red), (3, non-body stimuli that project as strongly as the weak body stimuli onto the body quadrant of object space (dark blue), and (4) non-body stimuli that project negatively onto the body quadrant of object space (bright blue). **b:** The predicted response of the body patch to each image in the four stimulus conditions in (a), computed by projecting the object space representation of each image onto the preferred axis of the body patch (determined from the average response of body patch neurons to images in the 1,224-image stimulus set). **c, left:** fMRI response time courses from the body patches in the four stimulus conditions in (a). **c, middle:** Mean normalized single-unit responses from neurons in the Body1 patch to the four stimulus conditions. **c, right:** Mean local field potential from the Body1 patch to the four stimulus conditions. Shading represents the standard error.

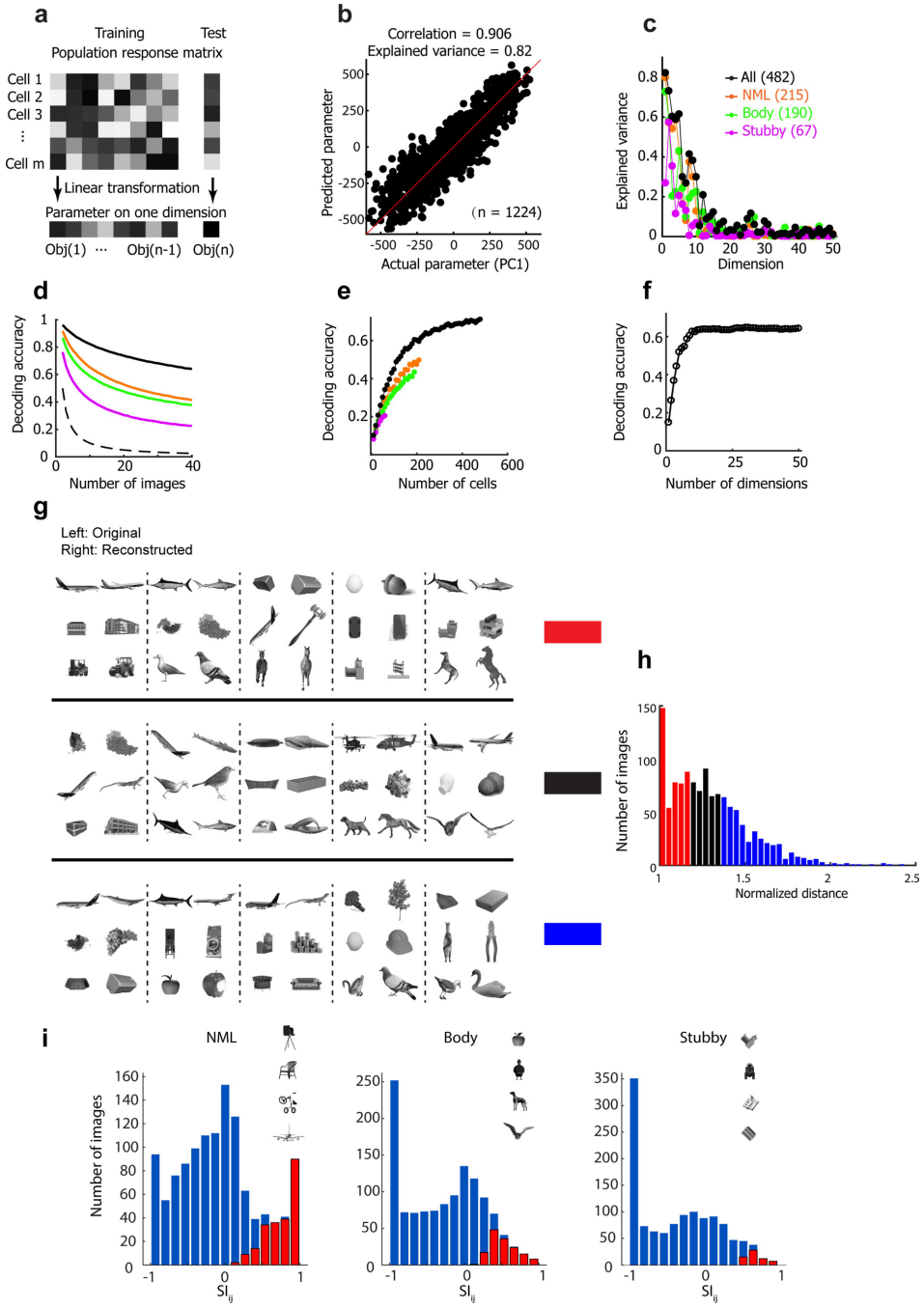

Figure 4.16: **Object and image decoding using a large object database. a:** A schematic illustrating the decoding model. To construct and test the model, we used *m* recorded cells' responses to *n* images. Population responses to images from all but one object were used to determine the transformation from responses to feature values via linear regression, and then the feature values of the remaining object were predicted (for each of 24 views). **b:** Model predictions plotted against true feature values for the first PC of the object space. (Continued on the next page)

Fig. 4.16, continued: **c:** Percentage of explained variance for all 50 dimensions using linear regression, based on the responses of four neural populations (yellow: 215 NML cells; green: 190 body cells; magenta: 67 stubby cells; black: 482 combined cells). **d:** Decoding accuracy as a function of the number of object images randomly drawn from the stimulus set for the four neural populations used in (c). The dashed line indicates chance performance. **e:** Decoding accuracy for 40 images, plotted against cell count, with cells drawn randomly from same four populations used in (c). **f:** Decoding accuracy for 40 images, plotted as a function of the numbers of PCs used to parametrize object images. **g:** Example reconstructed images from the three groups defined in (h). In each pair, the original image is shown on the left, and the image reconstructed using neural data is shown on the right. **h:** The distribution of the normalized distance between predicted and reconstructed feature vectors. The normalized distance takes into account the fact that the object images used for reconstruction did not include any of the object images shown to the monkey, setting a limit on the reconstruction quality (see Methods). A normalized distance of 1 means that the best possible solution has been found. Images were sorted into three groups based on these normalized distances. **i:** The distribution of specialization indices SI_{ij} across objects for the NML (left), body (middle) and stubby (right) networks (see Methods). Example objects for each network with $SI_{ij} \approx 1$ are shown. Red bars indicate objects with specialization indices significantly greater than 0 (two-tailed t-test, $p < 0.01$).

References

[1] Paul L Aparicio, Elias B Issa, and James J DiCarlo. "Neurophysiological organization of the middle face patch in macaque inferior temporal cortex". In: *Journal of Neuroscience* 36.50 (2016), pp. 12729–12745.

[2] Michael J Arcaro et al. "Seeing faces is necessary for face-domain formation". In: *Nature neuroscience* 20.10 (2017), pp. 1404–1412.

[3] Carlo Baldassi et al. "Shape similarity, better than semantic membership, accounts for the structure of visual object representations in a population of monkey inferotemporal neurons". In: *PLoS computational biology* 9.8 (2013), e1003167.

[4] Pinglei Bao and Doris Y Tsao. "Representation of multiple objects in macaque category-selective areas". In: *Nature communications* 9.1 (2018), p. 1774.

[5] Pinglei Bao et al. "A map of object space in primate inferotemporal cortex". In: *Nature* 583.7814 (2020), pp. 103–108. URL: https://doi.org/10.1038/s41586-020-2350-5.

[6] Le Chang, Pinglei Bao, and Doris Y Tsao. "The representation of colored objects in macaque color patches". In: *Nature communications* 8.1 (2017), p. 2064.

[7] Le Chang and Doris Y Tsao. "The code for facial identity in the primate brain". In: *Cell* 169.6 (2017), pp. 1013–1028.

[8] Anders M Dale, Bruce Fischl, and Martin I Sereno. "Cortical surface-based analysis: I. Segmentation and surface reconstruction". In: *Neuroimage* 9.2 (1999), pp. 179–194.

[9] Alexey Dosovitskiy and Thomas Brox. "Generating images with perceptual similarity metrics based on deep networks". In: *Advances in neural information processing systems* 29 (2016).

[10] Paul E Downing et al. "A cortical area selective for visual processing of the human body". In: *Science* 293.5539 (2001), pp. 2470–2473.

[11] Winrich A Freiwald and Doris Y Tsao. "Functional compartmentalization and viewpoint generalization within the macaque face-processing system". In: *Science* 330.6005 (2010), pp. 845–851.

[12] Ichiro Fujita et al. "Columns for visual features of objects in monkey inferotemporal cortex". In: *Nature* 360.6402 (1992), pp. 343–346.

[13] Charles G Gross, CE de Rocha-Miranda, and DB Bender. "Visual properties of neurons in inferotemporal cortex of the Macaque." In: *Journal of neurophysiology* 35.1 (1972), pp. 96–111.

[14] Theodros M Haile et al. "Visual stimulus-driven functional organization of macaque prefrontal cortex". In: *Neuroimage* 188 (2019), pp. 427–444.

[15] Peter Janssen, Rufin Vogels, and Guy A Orban. “Selectivity for 3D shape that reveals distinct areas within macaque inferior temporal cortex”. In: *Science* 288.5473 (2000), pp. 2054–2056.

[16] Nancy Kanwisher, Josh McDermott, and Marvin M Chun. “The fusiform face area: a module in human extrastriate cortex specialized for face perception”. In: (2002).

[17] Talia Konkle and Alfonso Caramazza. “Tripartite organization of the ventral stream by animacy and object size”. In: *Journal of Neuroscience* 33.25 (2013), pp. 10235–10242.

[18] Talia Konkle and Aude Oliva. “A real-world size organization of object responses in occipitotemporal cortex”. In: *Neuron* 74.6 (2012), pp. 1114–1124.

[19] Simon Kornblith et al. “A network for scene processing in the macaque temporal lobe”. In: *Neuron* 79.4 (2013), pp. 766–781.

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems* 25 (2012).

[21] Satwant Kumar, Ivo D Popivanov, and Rufin Vogels. “Transformation of visual representations across ventral stream body-selective patches”. In: *Cerebral Cortex* 29.1 (2019), pp. 215–229.

[22] Rosa Lafer-Sousa and Bevil R Conway. “Parallel, multi-stage processing of colors, faces and shapes in macaque inferior temporal cortex”. In: *Nature neuroscience* 16.12 (2013), pp. 1870–1878.

[23] Ifat Levy et al. “Center–periphery organization of human object areas”. In: *Nature neuroscience* 4.5 (2001), pp. 533–539.

[24] Bria Long, Chen-Ping Yu, and Talia Konkle. “Mid-level visual features underlie the high-level categorical organization of the ventral stream”. In: *Proceedings of the National Academy of Sciences* 115.38 (2018), E9015–E9024.

[25] Bruce D McCandliss, Laurent Cohen, and Stanislas Dehaene. “The visual word form area: expertise for reading in the fusiform gyrus”. In: *Trends in cognitive sciences* 7.7 (2003), pp. 293–299.

[26] Sebastian Moeller, Winrich A Freiwald, and Doris Y Tsao. “Patches with links: a unified system for processing faces in the macaque temporal lobe”. In: *Science* 320.5881 (2008), pp. 1355–1359.

[27] Shay Ohayon and Doris Y Tsao. “MR-guided stereotactic navigation”. In: *Journal of neuroscience methods* 204.2 (2012), pp. 389–397.

[28] Shay Ohayon et al. “Saccade modulation by optical and electrical stimulation in the macaque frontal eye field”. In: *Journal of Neuroscience* 33.42 (2013), pp. 16684–16697.

[29] Carlos R Ponce et al. “Evolving images for visual neurons using a deep generative network reveals coding principles and neuronal preferences”. In: *Cell* 177.4 (2019), pp. 999–1009.

[30] Ivo D Popivanov et al. “Heterogeneous single-unit selectivity in an fMRI-defined body-selective patch”. In: *Journal of Neuroscience* 34.1 (2014), pp. 95–111.

[31] Rishi Rajalingham and James J DiCarlo. “Reversible inactivation of different millimeter-scale regions of primate IT results in different patterns of core object recognition deficits”. In: *Neuron* 102.2 (2019), pp. 493–505.

[32] Martin Reuter and Bruce Fischl. “Avoiding asymmetry-induced bias in longitudinal image processing”. In: *Neuroimage* 57.1 (2011), pp. 19–21.

[33] Colin Reveley et al. “Three-dimensional digital template atlas of the macaque brain”. In: *Cerebral cortex* 27.9 (2017), pp. 4463–4477.

[34] Kadharbatcha S Saleem and Nikos K Logothetis. *A combined MRI and histology atlas of the rhesus monkey brain in stereotaxic coordinates*. Academic Press, 2012.

[35] Doris Y Tsao, Sebastian Moeller, and Winrich A Freiwald. “Comparing face patch systems in macaques and humans”. In: *Proceedings of the National Academy of Sciences* 105.49 (2008), pp. 19514–19519.

[36] Doris Y Tsao et al. “A cortical region consisting entirely of face-selective cells”. In: *Science* 311.5761 (2006), pp. 670–674.

[37] Doris Y Tsao et al. “Faces and objects in macaque cerebral cortex”. In: *Nature neuroscience* 6.9 (2003), pp. 989–995.

[38] Siavash Vaziri et al. “A channel for 3D environmental shape in anterior inferotemporal cortex”. In: *Neuron* 84.1 (2014), pp. 55–62.

[39] Bram-Ernst Verhoef, Kaitlin S Bohon, and Bevil R Conway. “Functional architecture for disparity in macaque inferior temporal cortex and its relationship to the architecture for faces, color, scenes, and visual field”. In: *Journal of Neuroscience* 35.17 (2015), pp. 6952–6968.

[40] Daniel LK Yamins et al. “Performance-optimized hierarchical models predict neural responses in higher visual cortex”. In: *Proceedings of the national academy of sciences* 111.23 (2014), pp. 8619–8624.

[41] Xiaomin Yue et al. “Curvature-processing network in macaque visual cortex”. In: *Proceedings of the National Academy of Sciences* 111.33 (2014), E3467–E3475.

[42] Chengxu Zhuang, Alex Lin Zhai, and Daniel Yamins. “Local aggregation for unsupervised learning of visual embeddings”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 6002–6012.

Chapter 5

PREDICTING NEURAL ACTIVITY ACROSS THE FLY VISUAL SYSTEM WITH CONNECTOME-CONSTRAINED NETWORKS

Forward

We can now measure the connectivity of every neuron in a neural circuit [58, 69, 70, 63, 64, 44, 19, 61, 50], but we are still blind to other biological details, including the dynamical characteristics of each neuron. And the degree to which measurements of connectivity alone can inform understanding of neural computation is an open question [5]. This chapter describes how, by measuring only the connectivity of a neural circuit, it is possible to predict the neural activity underlying a computation.

We constructed a model neural network with the experimentally determined connectivity for 64 cell types in the motion pathways of the fruit fly optic lobe [58, 69, 70, 63, 64] but with unknown parameters for other neuron and synapse properties. We then optimized the values of those unknown parameters using techniques from deep learning [26], to allow the model network to represent visual motion [14].

Our mechanistic model makes detailed, experimentally testable predictions for each neuron in the connectome. We found that model predictions agreed with experimental measurements of neural activity across 26 studies. Our work demonstrates a strategy for generating detailed hypotheses about the mechanisms of neural circuit function from connectivity measurements. Additionally, this strategy is more likely to be successful when neurons are sparsely connected—a universally observed feature of biological neural networks across species and brain regions.

This chapter is adapted from a 2024 *Nature* paper coauthored with Janne Lappalainen, Fabian Tschopp, Sridhama Prakhya, Aljoscha Nern, Kazunori Shinomiya, Shin-ya Takemura, Eyal Gruntman, Jakob Macke, and Srinivas Turaga [35]. My contributions to this study included (a) writing software to simulate phototransduction in fruit flies, (b) writing

software to efficiently train recurrent hexagonal lattice convolutional neural networks, (c) training a variety of connectome-constrained networks, (d) simulating probe stimuli, (e) analyzing the networks' responses to the probes, (f) generating visualizations, (g) general methodology development, and (h) contributions to the manuscript.

5.1 Introduction

Volume electron microscopy can now be used to comprehensively measure the connectivity of each neuron in a neural circuit, and even entire nervous systems [58, 69, 70, 63, 64, 44, 50, 19, 61]. However, we do not yet have the means to also comprehensively measure all other biological details in the same circuits, including the electrochemical properties of neurons and synapses, and the effects of neuromodulation and glia [60]. Since the propagation of neural activity is shaped by both synaptic connectivity, which we can measure exhaustively, and other factors, which we cannot, there has been considerable debate about the utility of connectome measurements for understanding brain function [32]. Is it possible to use measurements of connectivity to generate accurate predictions about how a neural circuit functions, without directly measuring neural activity from a living brain?

There is considerable evidence from computer science and neuroscience that there is not necessarily a strong link between the connectivity of a neural network and its computational function. Universal function approximation theorems for artificial neural networks [31] imply that a single computational task can be performed by many networks with very different connectivity patterns. And empirically, networks with a wide variety of general-purpose architectures have been trained to be roughly functionally equivalent [26, 53]. In other words, the mapping from architectures to tasks they are suited for is many-to-many. Similarly, in neuroscience, competing implementation hypotheses often exist simultaneously for a single computation; for example, [55] and [6] in the case of visual motion processing. And even circuits with the same connectivity pattern can function differently [42]. So neither the connectivity of a circuit alone nor its computational task alone can uniquely determine the mechanism of circuit function [9].

However, we found that by combining information about a neural circuit's connectivity with a hypothesis about the computational task it performs we could make accurate predictions about the roles played by individual neurons. We constructed a differentiable [51] neural network model whose connectivity was given by the

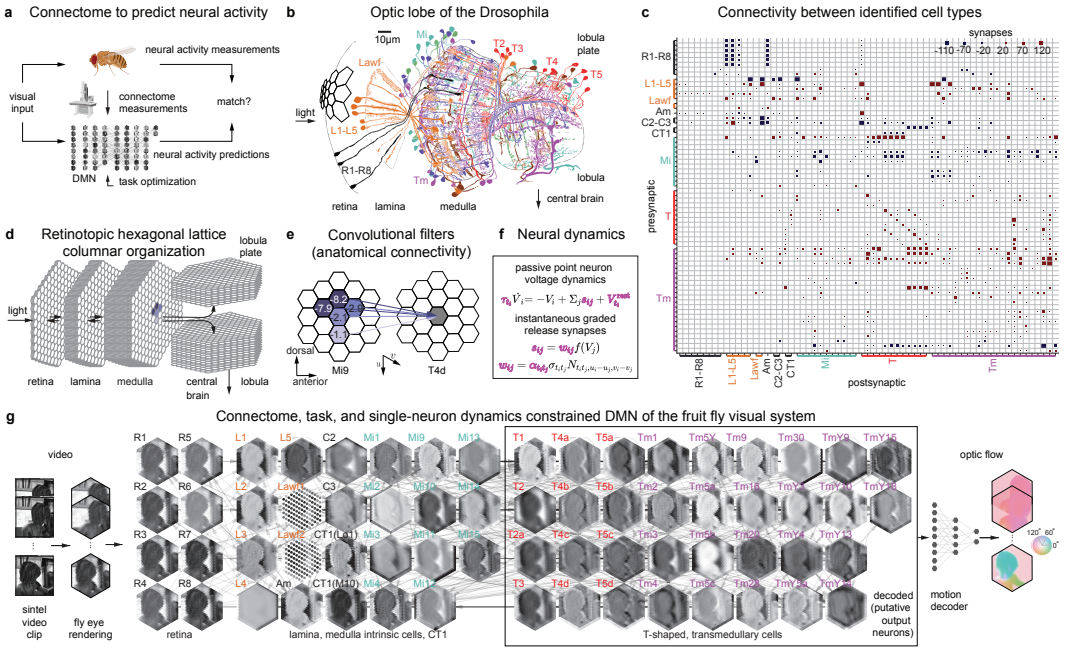

Figure 5.1: **Connectome-constrained and task-optimized models of the fly visual system.** **a:** Deep mechanistic network models (DMNs) aim to satisfy three constraints: The architecture is based on connectome measurements (b–e), cellular and synaptic dynamics are given by simple mechanistic models (f), and free parameters are optimized by training the model to perform optic flow estimation (g). **b:** A schematic of the optic lobe of *D.* melanogaster with several processing stages (neuropils) and cell types (adapted from [21]). **c:** Identified connectivity between 64 cell types, represented in terms of the total number of synapses from all neurons for each (presynaptic cell type, postsynaptic cell type) pair. Blue indicates putative hyperpolarizing inputs, red indicates putative depolarizing inputs, and the size of the squares corresponds to the number of input synapses. **d:** The retinotopic hexagonal lattice columnar organization of our visual system model. Each lattice represents a cell type, and each hexagon represents an individual cell. Photoreceptor columns are aligned with downstream columns. The model contains synapses from all neuropils. **e:** An example convolutional filter, representing Mi9 inputs onto T4d cells. The numbers in the cells are average synapse counts. **f:** Single-neuron and synaptic dynamics are described by simple mechanistic models. Free parameters (magenta) are optimized by training the recurrent network model to perform optic flow estimation. (Continued on the next page)

Fig. 5.1, continued: **g:** An illustration of a DMN performing optic flow estimation. Each hexagonal lattice shows a snapshot of simulated voltage levels of all cells of each type in response to stimuli presented to the photoreceptors (R1–R8). Edges illustrate connectivity between cell types. A decoder receives the simulated neural activity of all output neurons to estimate optic flow. The parameters of the DMN and the decoder are tuned using gradient-based optimization.

connectome measurements of a real neural circuit. We then optimized the unknown neuron and synapse parameters of the model using techniques from deep learning [26], to enable the model to accomplish a computational task [79]. The optimized model was used to make predictions about how individual neurons would behave in response to a variety of sensory stimuli. We call such models connectome-constrained and task-optimized deep mechanistic networks (DMNs; Fig. 5.1a).

We applied this approach to model the motion pathways in the optic lobe of the *Drosophila* visual system. We constructed a DMN with experimentally measured connectivity [58, 69, 70, 63, 64], and unknown single-neuron parameters and synapse strengths. We optimized the model to perform the computer vision task of optic flow regression—that is, estimating the image-plane projection of surface trajectories—using an annotated video dataset [14]. Visual motion computation in the fly and its mechanistic underpinnings have been extensively studied [10], so we were able to compare the detailed predictions of our model with experimental measurements of neural activity in response to visual stimuli, on a neuron-by-neuron basis. We found that our connectome-constrained and task-optimized DMN accurately predicted the separation of the visual system into light-increment (ON) and light-decrement (OFF) channels, as well as direction selectivity in the well-known T4 and T5 motion detector neurons [17]. We've released our model as a resource for the community.[1]

5.2 Our deep mechanistic network model

The optic lobe of the fruit fly is analogous to the mammalian retina. It is composed of several layered neuropils whose columnar arrangement has a one-to-one correspondence with the ommatidia—the fly's photoreceptor clusters—with both possessing a remarkably crystalline organization in a hexagonal lattice. Visual input from the photoreceptors is received by the lamina and medulla, which send projections to the lobula and lobula plate (Fig. 5.1b[21]). Many components of the optic lobe are highly regular, with columnar cell types appearing once per column, and multi-columnar

[1] `https://github.com/TuragaLab/flyvis`

neurons appearing with only small deviations from a well-defined periodicity in columnar space [21, 49]. Several studies have reported on the local connectivity within the optic lobe and its motion pathways [58, 69, 70, 63, 64]. We assembled these separate local reconstructions into a coherent local connectome spanning the retina, lamina, medulla, lobula, and lobula plate (Fig. 5.1c).

We approximated the circuitry across the entire visual field as perfectly periodic [69, 49], and tiled this local connectivity architecture in a hexagonal lattice across retinotopic space to construct a consensus connectome for 64 cell types across the central visual field of the right eye (Fig. 5.1d; Methods). Because of this periodic tiling, the synapse count between each pair of neurons was the same across all pairs of neurons with the same pre- and postsynaptic cell type and relative location in retinotopic space. We'll refer to this partial connectome of the motion pathways as "the connectome" for simplicity.

We constructed a recurrent neural network modeling these first stages of visual processing in the optic lobe based on the connectome for the right eye. Each neuron in this DMN corresponds to a real neuron in the fly visual system, belonging to an identified cell type, and is connected to other neurons only if they are connected by synapses in the connectome (Fig. 5.1e). While our model has detailed connectivity, we used simple relatively simple neuron and synapse dynamics (Fig. 5.1f). We used passive leaky linear non-spiking voltage dynamics to model the time-varying activity of single neurons, since many neurons in the early visual system are non-spiking. And we modeled most neurons with a single electrical compartment, as this has been previously shown to be a good approximation, given the small size of many neurons in the optic lobe [30]. The CT1 neuron—which is amongst the largest in the brain, spanning the entire optic lobe—was the exception to this, and we modeled it with one compartment per column in the medulla and lobula, since it is highly electrotonically compartmentalized [46]. We modeled the graded release chemical synapses between non-spiking neurons with a threshold-linear function to approximate the nonlinear voltage-gated release of neurotransmitter. The resulting network model follows well-known threshold-linear dynamics and is piece-wise differentiable. Such dynamics are often used to approximate the firing rates of a network of spiking neurons with the nonlinearity arising from spike generation, whereas in our network, the nonlinearity represents the voltage-gated neurotransmitter release.

We used the cell type structure of the connectome to reduce the number of free parameters in the model (Fig. 5.1f). We assumed that neurons of the same cell

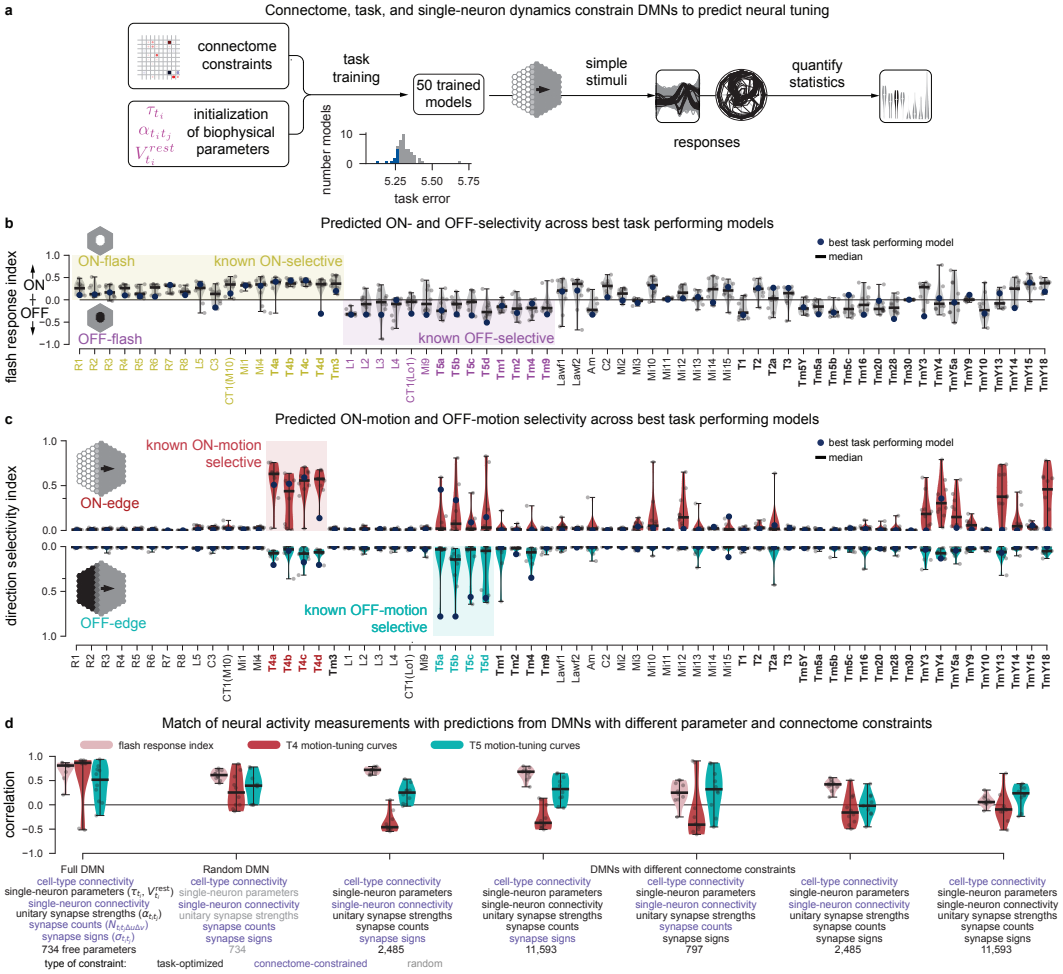

Figure 5.2: **Ensembles of DMNs predict tuning properties. a:** We optimized 50 connectome-constrained DMNs, yielding a variety of solutions, and compared the tuning properties of their cells to experimental measurements. Inset: The task error distribution. Blue: The 10 best models, also shown in (b–d). **b:** ON- and OFF-contrast selectivity indices for each cell type for the 10 models with best task performance. (See Fig. 5.13 for the 10 worst models.) Yellow: Cell types known to be ON-selective. Violet: Cell types known to be OFF-selective. Black: Selectivity not yet established experimentally. Bold: Inputs to the optic flow decoder. **c:** Direction selectivity indices (DSI) computed from neural responses to moving edges, using the same 10 models as above. **d:** Correlations between measurements and neural activity predictions for seven types of DMNs with different connectome constraints. Dashes indicate the median correlation across models. The first DMN type on the left corresponds to the main DMNs analyzed in panels (b) and (c), and and all subsequent figures. The remaining six DMNs incorporate fewer constraints.

type shared the same neuron time constant and resting membrane potential. We modeled synaptic weights as proportional to the discrete number of synapses as reported in the connectome between a connected neuron pair [38], with a scale factor representing the strength of a unitary synapse. The unitary synapse scale factor and the sign of each synapse was the same for all pairs of neurons with the same pre- and postsynaptic cell type. For example, an Mi1-neuron-to-T4-neuron connection with five synapses is assumed to be exactly half as strong as an Mi1-neuron-to-T4-neuron connection with ten synapses, but could be stronger or weaker than a five-synapse connection where the presynaptic cell type, the postsynaptic cell type, or both were different. The sign of each connection type was determined via neurotransmitter and receptor expression profiling [18] (Methods).

In total, the connectome-constrained model comprises 45,669 neurons and 1,513,231 connections, across 64 cell types arranged in a hexagonal lattice consisting of 721 columns, modeling the central visual field of the roughly 700–900 ommatidia typically found in the fruit fly retina [27]. Connectome constraints and our assumption of spatial homogeneity (*i.e.*, the hexagonally convolutional structure of the network) result in a dramatic reduction to just 734 free parameters for this large network model. The only free parameters in our model are the single neuron time constants and resting membrane potentials (two parameters per cell type), and the unitary synapse strengths (one parameter per type-to-type connection). In the absence of connectome measurements, we would have needed to estimate well over a million parameters corresponding to the weights of all possible connections (Methods).

We used task optimization [79] to further constrain the parameters of the model; *i.e.*, we trained the model to perform a computational task that we hypothesized was an approximation of the computation carried out by the circuit in living flies. The task we selected was to perform dense optic flow regression during the passive viewing of naturalistic stimuli [14], meaning that, at regular time intervals, for every point on a 2D grid across the visual field, the model had to estimate the image-plane velocity of the surface currently visible at that point. Motion processing requires a neural circuit to compare visual stimuli across space and time, and we hypothesized that training our model to perform the optic flow regression could help attract the parameters toward regions of the parameter space in which meaningful temporal integration occurred.

To decode optic flow from the DMN, we used a decoding network to map the representation of motion used in the fly nervous system to the representation of

optic flow specified by the computer vision task. This two-layer convolutional decoding network was given only the instantaneous neural activity of the medulla and downstream areas as input. Importantly, the decoding network cannot by itself detect motion, which requires the comparison of current and past visual stimuli, but must instead rely on the temporal dynamics of the DMN to compute motion-selective visual features. The resulting combination of our recurrent connectome-constrained DMN model and the feedforward decoding network was trained end-to-end: We rendered video sequences from the Sintel database [14] as input to the photoreceptors of the connectome-constrained model, and used gradient-based optimization (backpropagation through time [26]) to minimize the optic flow regression error (Fig. 5.1g, Methods).

5.3 Our DMN ensemble predicts known activity

Since we only used structural and task information to construct our DMN, we can validate it by comparing neural activity predictions for each of the 64 identified cell types to experimental measurements. As the connectome and the task together did not uniquely constrain the model parameters to a single point in the parameter space [43], we generated an ensemble of 50 models, all constrained with the same connectome, and optimized to perform the same task. Each model in the ensemble corresponds to a local optimum of task performance. The models achieved similar (but not identical) task performance, so the ensemble reflects the diversity of possible models consistent with these constraints. The model ensemble found a variety of parameter configurations (Fig. 5.11), and the models that were discovered performed better than both the decoder network alone and models with random parameters (Fig. 5.12a). We focused our analysis on the 10 models that achieved the best task performance (Fig. 5.2a). We simulated neural responses to multiple experimentally characterized visual stimuli, and comprehensively compared model responses for each cell type to experimentally reported responses from 26 previously reported studies.

As has been observed in other species [11, 25], neural responses in the fly visual system are known to segregate into ON- and OFF-channels [33], defined by whether a neuron depolarizes more strongly to an increase or a decrease in stimulus intensity, respectively. We probed the contrast preference of each cell type using flash stimuli [68] and found that the ensemble predicts the segregation into ON- and OFF-pathways with a high level of accuracy. The median flash response index (FRI) across the ensemble predicts the correct ON- and OFF-preferred contrast selectivity for all 32

of the 32 cell types for which contrast selectivity has been experimentally established. This is also the case for the model with the best task performance (the "task-optimal" model), which correctly predicts the preferred contrast of 30 of 32 cells (Fig. 5.2b). Additionally, the ensemble provides predictions for the remaining 33 cell types, and consistency across the ensemble provides a measure of confidence in the predictions (Fig. 5.2b).

Another major result in fly visual neuroscience is the identification of the T4 (ON) and T5 (OFF) neurons as direction-selective neurons with four subtypes (T4a, T4b, T4c, T4d, and T5a, T5b, T5c, T5d), each responding to motion in one of the four cardinal directions [39]. We characterized the motion selectivity of all 64 cell types by their responses to ON- and OFF-edges moving in 12 directions. We found that the ensemble of models correctly predicted that T4 neurons are ON-motion selective, and T5 neurons are OFF-motion selective (Fig. 5.2c). The ensemble also correctly predicted the lack of motion tuning in the inputs of the T4 and T5 neurons (Mi1, Tm3, Mi4, Mi9, Tm1, Tm2, Tm4, Tm9, CT1; see Methods).

Our models also suggest that the transmedullary cell types—TmY3, TmY4, TmY5a, TmY13, and TmY18—might be tuned to ON-motion. Of these cell types, TmY3 neurons do not receive inputs from other known motion-selective neurons, which suggests that these neurons might be involved in a motion-processing pathway operating in parallel to the pathway that contains the T4 and T5 neurons (Fig. 5.9 and Fig. 5.10). We investigated whether our model predicted motion selectivity for all cell types with asymmetric, multicolumnar inputs, as this is a necessary connectivity motif for direction selectivity. Based on their local spatial connectivity profiles, we estimated that 19 cell types receive asymmetric, multi-columnar inputs (Fig. 5.7b, Methods), but found that the ensemble only predicted 12 to be motion-selective (Methods). The spatial offset of excitatory and/or inhibitory inputs did not correlate strongly with direction selectivity (Fig. 5.7c, d). This suggests that the manner in which our simulated neurons responded to motion was influenced by the task and/or the connectivity pattern across much of the network, as opposed to only local connectivity.

5.4 The connectome and the task are both necessary

We investigated the importance of connectome constraints and task optimization in enabling accurate neural activity predictions. We found that both task optimization and connectome constraints at the single-neuron resolution were necessary in order

to predict the preferred contrast of the 32 characterized cell types and the preferred direction of motion for the T4 and T5 subtypes (Fig. 5.2d, Fig. 5.12).

We conducted ablation studies comparing the DMN ensemble studied in this paper (Full DMN) with a range of alternative models. First, we verified the importance of task optimization by constructing an ensemble (Random DMN) with random single-cell and synapse parameters, and full connectome constraints. This ensemble yielded accurate predictions of preferred contrast, but poor predictions of direction selectivity and preferred direction.

We then studied which aspects of the connectome must be measured accurately to lead to accurate predictions. We found that task-optimized models that only had access to cell-type-level connectivity predicted neural activity poorly. We then considered several scenarios in which the models had access to additional measurements, including (1) access to synapse signs and per-neuron connectivity, but not synapse counts, (2) access to synapse signs, but not per-neuron connectivity or synapse counts, (3) access to per-neuron connectivity and synapse counts, but not synapse signs, and (4) access to per-neuron connectivity, but not synapse counts or synapse signs. Across these modeling assumptions, we found that accurately predicting contrast preference (FRI) was possible as long the connection signs were known. We also found that accurately predicting direction selectivity—but not preferred directions—if the cell-to-cell connectivity pattern was known, even if the synapse counts were unknown (Fig. 5.12c). This demonstrates that both the detailed connectome measurements and the task optimization contributed to achieving accurate predictions of neural activity.

Across the ensemble constrained by all connectome measurements and task training, we found that models with lower task error scores (Methods) also had more realistic tuning. Models with higher task performance predict the direction selectivity index (DSI) of T4 and T5 cells and their inputs more accurately ($r = 0.60$, $p = 2.6 \times 10^{-6}$; Fig. 5.12h and Fig. 5.13b). This suggests that it may be possible to use task error scores to rank models in terms of their likelihood to accurately predict neural activity.

Our model relies on an accurate classification of neurons into cell types in order to share single-neuron and synapse parameters across all neurons of the same cell type. We investigated the degree to which we could coarse-grain the cell type categorization, leading to fewer cell types and fewer parameters (Fig. 5.12e–g). We found that grouping the four T4 subtypes into a single T4 cell type, and grouping the four T5 subtypes into a single T5 cell type had no negative impact on the quality

of the ensemble predictions. However, grouping all 37 excitatory cell types into a single "E" type, 22 inhibitory cell types into a single "I" type, and 4 mixed cell types into a single "mixed" type lead to poor performance on par with the random DMN.

5.5 Predictions cluster across the DMN ensemble

How similar are the predictions of different DMNs in an ensemble in which each network was constructed using the same connectome and trained to perform the same task? To address this question, we simulated neural activity in response to naturalistic video sequences from the Sintel dataset. We then used UMAP [7] to perform nonlinear dimensionality reduction on activity vectors across the model ensemble, and clustered the models in the resulting 2D projection (Fig. 5.3a, Methods). For many cell types, we found that models predict strongly clustered neural responses. For T4c neurons, for example, we found three clusters corresponding to qualitatively distinct responses to naturalistic stimuli. Two clusters contain models with direction-selective T4c cells (Fig. 5.3a, b) with up- and down-selective cardinal tuning, respectively, and neurons in the third cluster are not direction tuned. The direction selective cluster with the (correct) upward preference has the lowest average task error (circular marker, average task error 5.297), followed by the cluster with the opposite preference (triangular marker, average task error 5.316). The non-selective cluster has the worst performance (square marker, average task error 5.357), suggesting that models with accurate tuning correlate with lower task error (see also Fig. 5.12h).

What differences in circuit mechanisms underly the different predictions for direction selectivity in the three clusters (Fig. 5.3c)? We found that direction selectivity in the two tuned clusters is associated with opposing preferred-contrast tuning of Mi4 and Mi9 neurons, which provide direct flanking inhibitory input to T4 neurons (Fig. 5.3d). Models with the correct direction selectivity for T4 neurons also predict the correct contrast selectivity for Mi4 and M9 neurons, and vice versa (Fig. 5.3e).

So, the ensemble can be used to provide hypotheses about the circuit mechanisms that might underlie the response properties of individual cells. Additionally, it shows that the experimentally measured tuning of one neuron can be used to apply constraints on other neurons in the circuit. Here, filtering models in the ensemble with the experimentally measured direction selectivity for the T4c neurons (by only selecting models from the correct cluster) is sufficient to correctly recover the tuning of both Mi4 and Mi9 neurons.

5.6 Predicted mechanism of T4 & T5 tuning

Our DMN modelling approach enables a variety of analyses that can illuminate the mechanistic basis of computation in a circuit, and suggest novel stimuli for experimental characterization. We illustrate these analyses using averages from the model cluster with the best task performance (the task-optimal cluster), focusing on the well-studied T4 and T5 neurons (Fig. 5.4). In the task-optimal cluster, the four T4 subtypes respond strongly to bright (ON) edges, and the four T5 subtypes respond

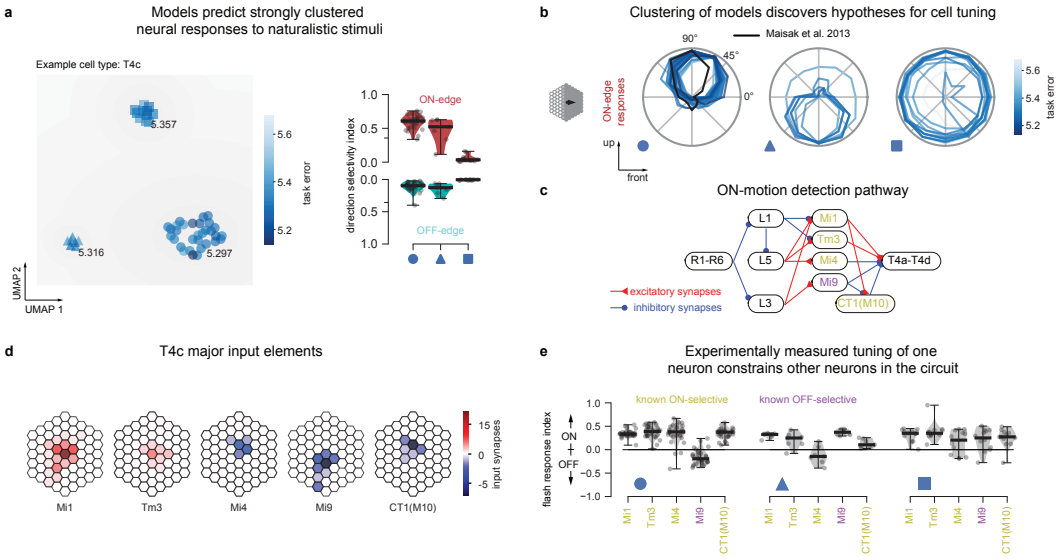

Figure 5.3: **Cluster analysis of DMN ensembles enables hypothesis generation and suggests experimental tests.** We clustered 50 DMNs after embedding them in a two-dimensional space based on their responses to naturalistic scenes, and aimed to identify whether the clusters corresponded to qualitatively different tuning mechanisms. **a:** T4c cell responses exhibited three clusters: two with ON-motion direction selectivity (the circular and triangular markers), and one without (the square marker). **b:** T4c tuning in the three clusters. Circular marker: Upward tuning (the cluster with lowest average task error: 5.297; the known tuning of T4c is shown in black). Triangular marker: Downward tuning (5.316 error). Square marker: No motion tuning (5.357 error). **c:** A schematic of the corresponding ON-motion-detection pathway. **d:** Connectivity of major inputs to T4c. Blue and red: Putative hyper- and depolarizing inputs. Saturation: The average number of input synapses for each spatial offset. **e:** Tuning properties within each cluster reveal dependencies between T4 tuning and the tuning of Mi4 and Mi9 cells in the ensemble. Switching Mi4 (known ON-contrast selective) and Mi9 (known OFF-contrast selective) contrast preferences results in directionally opposite motion tuning in T4. DMNs in first cluster (the circular marker) exhibit ON selectivity for Mi1, Tm3, Mi4, and CT1(M10), and OFF selectivity for Mi9. In response to ON motion stimuli, in these DMNs T4c receives central depolarizing input from Mi1 and Tm3 and dorsal hyperpolarizing input from Mi4 and CT1(M10).

strongly to dark (OFF) edges, moving in the four cardinal directions, in agreement with experimental findings[39, 22, 30, 29] (Fig. 5.4a). We probed the mechanism of this direction selectivity in T4 and T5 neurons (Fig. 5.4b, Fig. 5.8). Examining the input currents to a single T4 neuron (Fig. 5.8a), we found that fast excitation and offset, delayed inhibition enables T4 neurons in the model to detect motion, in agreement with experimental findings [30]. The differential response of T4 neurons to motion in the preferred vs. the null direction is primarily produced by the differential timing of inhibition from Mi4. Additionally, excitatory T4-to-T4 currents between neurons with the same preferred direction lead to an increased response to coherent motion across the visual field. While research into T4 motion selectivity has largely focused on the role of feedforward inputs, our modeling predicts an important role for the lateral connectivity between T4 neurons. The mechanisms for T5 motion selectivity in our model are similar (Fig. 5.8c), with differential timing of inhibition from CT1 as well as excitation from Tm9 contributing to motion-selective responses.

To relate the mechanism of direction selectivity to the well-studied mechanisms of preferred-direction enhancement and null-direction suppression, we compared the responses of T4 and T5 neurons to moving bars and static bars as in [30]. Consistent with voltage measurements [30, 29], voltage response predictions based on our model show null-direction suppression but no preferred-direction enhancement (Fig. 5.8b, d).

We computed and compared the spatial and temporal receptive fields of the major columnar inputs to T4 and T5 neurons. These input neurons have been the focus of multiple experimental studies of the motion detection pathways [8, 80, 4, 67, 46, 54] (Fig. 5.4d). In agreement with experimental findings [4, 46], the DMNs predicted that Tm3 and Tm4 have broad spatial receptive fields (two column radius, 11.6°), while Mi1, Mi4, Mi9, Tm1, Tm2, Tm9, and CT1 compartments in both the medulla and lobula have narrow spatial receptive fields (single column radius, 5.8°).

We characterized the temporal response properties of cells in the motion pathways, including the lamina monopolar cells (L1-L5) and direct inputs to the T4 and T5 neurons. We simulated neural responses to single-ommatidium flashes of varying contrast and duration and compared them to the temporal responses observed in real flies (Fig. 5.4e). The model accurately predicts the preferred contrast of each cell type—*i.e.*, whether they depolarize more strongly to ON or OFF single-ommatidium flashes (5ms–300ms duration, Methods) [33]. These cells either depolarize (in which case we call them ON-selective) or hyperpolarize (in which case we call

them OFF-selective) in response to light increment flashes. The temporal response properties were correctly predicted for all major T4 inputs except Tm4 cells in this model; Mi1, Tm3, and Mi4 cells respond with transient depolarization to ON flashes. In contrast, CT1(M10) responds with a longer sustained depolarization. Mi9 hyperpolarizes. For major T5 inputs, Tm1, Tm2, Tm9, and CT1(Lo1) respond with transient hyperpolarization, and Tm4 is incorrectly predicted to depolarize. For lamina cell types, the DMNs predict biphasic hyperpolarization in L1, L2 and monophasic hyperpolarization in L3, and L4, as well as depolarization in L5.

For motion-selective neurons like T4 and T5, the spatiotemporal receptive fields are not separable in space and time. We characterized the full spatiotemporal receptive field for T4c and T5c neurons (Fig. 5.4f) using single-ommatidium ON and OFF flashes (20ms, Methods). ON flashes on the leading side of the receptive field of the ON-contrast, upwards-direction-selective T4c cell lead to fast depolarization, whereas ON flashes on the trailing side lead to delayed hyperpolarization, again matching experimental findings [30]. Because T5c is OFF-selective, its OFF-impulse responses are inverted, resembling the T4c spatiotemporal receptive field (Fig. 5.14a). This reflects the fact that in our models T5c uses a similar mechanism to respond to OFF edges as T4c uses to respond to ON edges, in agreement with experimental findings [29].

Finally, we show that the model can be used to design optimized stimuli. We used the task-optimal model to find the video sequences in the Sintel dataset that elicited the largest responses in the motion-selective neurons (Fig. 5.4g; Methods). One might expect that pure ON or OFF stimuli would elicit the largest responses in T4 and T5, respectively. However, we found both ON and OFF elements in optimized stimuli, suggesting an interplay between the ON and OFF pathways. The stimulus that elicited the strongest response in the T4c cell was a central OFF disc followed by an ON edge moving upwards, matching the preferred direction of the cell. Similarly, for the T5c cell, the stimulus that elicits the strongest response was a central ON disc followed by an OFF edge moving upwards in the preferred direction of the cell. (SeeFig. 5.14b for corresponding full-field naturalistic stimuli, numerically optimized stimuli, and preferred moving edge stimuli.) Taken together, our DMN ensemble predicts many functional properties of T4 cells, T5 cells, and their inputs.

5.7 Sparsity enables accurate predictions

What conditions affect whether a connectome-constrained and task-optimized DMN can accurately predict neural responses at a single-neuron resolution? Sparse connectivity is a hallmark of biological neural circuits, so we investigated whether sparse connectivity could enable DMNs to make accurate neural activity predictions. For sparsely connected circuits—assuming the connectome is known—there are

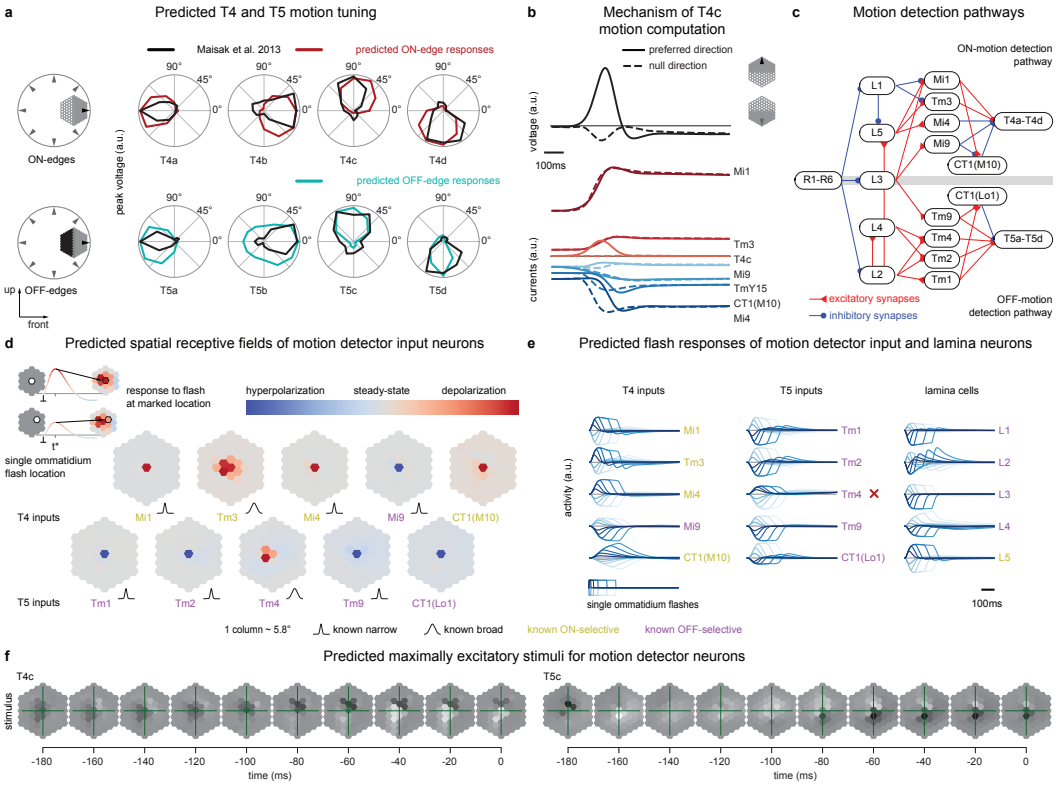

Figure 5.4: **Task-optimal DMNs largely recapitulate known mechanisms of motion computation. a:** Responses to moving edges for T4 and T5 subtypes from task-optimal model clusters, and comparison with experimental measurements [39, 22] **b:** The voltage of a T4c neuron (top) and contributions from major input cells (bottom) while an ON edge moves across the visual field in preferred (solid) and null (dashed) directions. **c:** Major cell types and connectivity in the ON- (T4) and OFF- (T5) motion detection pathways (simplified). **d:** Spatial receptive fields of major motion detector input neurons revealed by single-ommatidium flashes and comparison with experimental measurements [4, 46]. **e:** Single-ommatidium flash responses agree with experimental measurements [8, 4], with the exception of Tm4 (red cross). **f:** The stimulus sequences predicted to elicit the strongest responses in T4c and T5c cells. A central OFF disc followed by an ON edge moving upwards elicits the strongest response in a T4c cell, and an ON disc followed by an OFF edge elicits the strongest response in a T5c cell.

fewer synapse parameters left to estimate using task optimization. We hypothesized that such networks might support fewer possible mechanisms by which to perform a given task, increasing the likelihood that a task-optimized DMN would find the true mechanism and accurately predict single-neuron activity.

We tested this hypothesis in simulation (Fig. 5.5) by constructing feedforward artificial neural networks to solve the MNIST handwritten digit classification task. These networks had varying degreees of connection sparsity, and neurons were randomly assigned as excitatory and inhibitory respecting Dale's law (25 ground truth networks for each sparsity level, Methods). We then simulated the process of acquiring connectome measurements from these ground truth networks, and constructed task-optimized DMNs constrained by these synthetic connectomes (Fig. 5.5a).

Since the degree to which connection strength can be inferred from noisy connectome measurements is still unknown, we simulated two settings. In the first setting, we assumed that connectome measurements reveal connectivity but not connection strength. In this setting, DMNs were optimized to infer both the resting membrane potential of each neuron and the strength of the connection between each pair of connected neurons. In the second setting, we assumed that connectome measurements additionally reveal noisy estimates of connection strengths, and so the synapse counts were used as soft constraints during optimization.

Consistent with our hypothesis, we found that sparsity in the connectome greatly improves the accuracy of neural activity predictions with measurements of connectivity alone (Fig. 5.5b, median Pearson correlation of 0.85 for 10% connectivity vs 0.38 for 80% connectivity, 100 randomly selected neurons from 25 randomly generated groundtruth networks). However, with the additional availability of connection strength estimates, we found that DMN simulations accurately predicted neural activity even in the absence of sparse connectivity (median Pearson correlation >0.9 across all connectivities).

Our model of the fly visual system lies in an intermediate regime with regards to our knowledge of connection strength. We assumed that connectome measurements could be used to infer relative connection strengths but not absolute connection strengths, since we assumed that the unitary synaptic strength was unknown but the same for connections with the same cell type pair. Consequently we attribute the success of our visual system model—in terms of predicting neural activity—to both the sparsity of the circuit and the connection strength information we derived from synapse counts.

5.8 Discussion

We constructed a neural network with connectivity measured at the microscopic scale. We also required that at the macroscopic scale, the collective neural activity dynamics across the entire network resulted in an ethologically relevant computation. This combination of microscopic and macroscopic constraints enabled us to construct an ensemble of large-scale computational models spanning many tens of cell types and tens of thousands of neurons. We showed that such large-scale mechanistic models could accurately make detailed predictions of the neural responses of individual neurons to dynamic visual stimuli, revealing the mechanisms by which computations are performed. Knowledge of the connectome played a critical role in this success, in part by leading to a massive reduction in the number of free model parameters.

We have opted for a somewhat minimalistic modeling approach, using simple models of individual neurons and synapses, to focus on the role played by the connectivity of a neural network. We found that for the motion pathways of the fruit fly visual system, our model correctly predicts many aspects of visual selectivity. However,

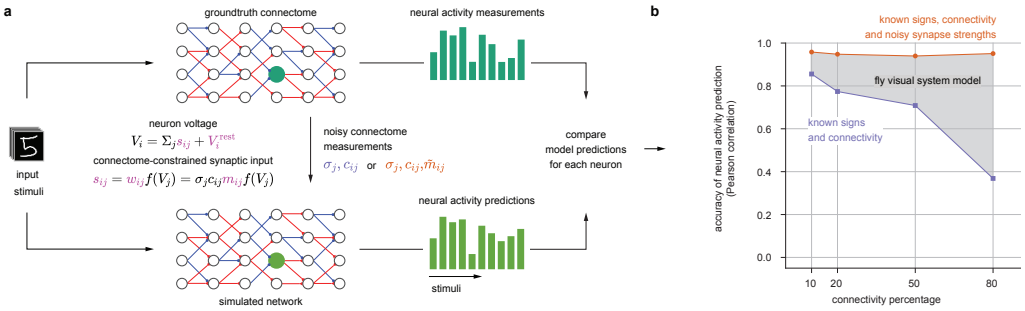

Figure 5.5: **Connectome measurements constrain neural networks in circuits with sparse connectivity. a:** We constructed synthetic "ground truth connectome" networks with varying degrees of sparse connectivity for classifying hand-written digits. For each ground truth connectome network, we simulated connectome measurements and constructed a connectome-constrained and task-optimized "simulated network" (Methods). We measured the correlation of the neural response vector, across all stimuli, between ground truth (dark green) and simulated networks (light green). **b:** Median neural response correlation coefficients from 100 randomly-sampled neuron pairs from each layer and across 25 network pairs. Two conditions were considered, including a condition in which connectome measurements revealed only binary connectivity (blue), and a condition in which connectome measurements also contained information about connection strengths (orange). The fly visual system model presented here likely falls in the region between the two curves, since measured synapse counts inform relative connection strengths between pairs of neurons for the same pair of cell types, but not absolute connection strengths.

our model cannot, for example, account for the roles played by electrical synapses [3], nonlinear chemical synapses [28], and neuromodulation [66]. Richer models of neurons, synapses, plasticity, and extra-synaptic modulation can enable accurate modeling of these and other effects in the fly visual system and beyond. Additionally, we only considered the role this circuit plays in processing motion, which is just one of many computations performed by the visual system [17], and it would be interesting to see our approach applied using a broader range of ethologically relevant tasks.

Task-optimized artificial neural network models, for instance of mammalian visual pathways [79], have previously demonstrated only a coarse correspondence in terms of population neural activity between model regions and brain regions. In contrast, every neuron and synapse in our connectome-constrained model [71, 40, 47, 65, 41] has a direct correspondence to a neuron or synapse in the fruit fly brain. This correspondence enables highly detailed, experimentally testable predictions at the single-neuron resolution.

Our modeling approach can be used as a discovery tool, leveraging connectome measurements to generate detailed, experimentally testable hypotheses about the computational roles of individual neurons. Measurements of neural activity are necessarily sparse and involve difficult trade-offs. Activity can frequently only be measured in a limited number of contexts, and for either a limited number of neurons or for a larger number of neurons with poorer temporal resolution. Connectome-constrained DMN models generate meaningful predictions even in the complete absence of neural activity measurements, and can be further constrained by sparse measurements of neural activity (Fig. 5.3). They can also be directly fit to measured neural activity [47] or behavior [16].

Whole-brain connectome projects have just been completed for the larval and adult fruit fly [78, 81, 61, 19], including two new connectomes of the entire fruit fly optic lobe [44, 50], and whole mouse brain connectome projects are now being discussed [1]. Large-scale whole nervous system models [47, 76, 65] will be of critical importance for integrating connectomic, transcriptomic, neural activity, and animal behavior measurements across labs, scales, and the nervous system [60]. And with the recent development of detailed biomechanical body models for the fruit fly [74, 57], we can now contemplate constructing whole animal models spanning brain and body.

5.A Methods

5.A.1 Construction of spatially invariant connectome from local reconstructions

We built a computational model of the fly visual system which is consistent with available connectome data [63, 70, 69, 73, 72, 58], which has biophysically plausible neural dynamics, and which can be computationally trained to solve an ethiologically relevant behavioural task, namely estimation of optic flow. To achieve this, we developed algorithms to blend annotations from two separate data-sets by transforming, sanitizing, combining and pruning the raw data sets into a coherent connectome spanning all neuropils of the optic lobe.

The original data stems from focused ion beam scanning EM datasets (FIBSEM) from the FlyEM project at Janelia Research Campus. The FIB-25 dataset volume comprises seven medulla columns and the FIB-19 dataset volume comprises the entire optic lobe and, in particular, detailed connectivity information for inputs to both the T4 and T5 pathways [69, 70, 63]. The data available to us consisted of 1801 neurons, 702 neurons from FIB-25 and 1099 neurons from FIB-19. For about 830 neurons the visual column was known from hand annotation. These served as reference positions. Of the 830 reference positions, 722 belong to neuron types selected for simulation. None of the T5 cells, whose directional selectivity we aimed to elucidate, were annotated. We therefore built an automated, probabilistic expectation maximization algorithm that takes synaptic connection statistics, projected synapse center-of-mass clusters and existing column annotations into account. Only the neurons consistently annotated with both 100% and 90% of reference positions used were counted to estimate the number of synapses between cell types and columns, in order to prune neuron offsets with low confidences.

Synaptic signs for most cell types were predicted based on known expression of neurotransmitter markers (primarily the cell type specific transcriptomics data from Davis et al 2020). For a minority of cell types included in the model, no experimental data on transmitter phenotypes were available. For these neurons, we used guesses of plausible transmitter phenotypes. To derive predicted synaptic signs from transmitter phenotypes, we assigned the output of histaminergic, GABAergic and glutamatergic neurons as hyperpolarizing and the output of cholinergic neurons as depolarizing. In a few cases, we further modified these predictions based on distinct known patterns of neurotransmitter receptor expression (see [18] for details). For example, output from R8 photoreceptor neurons, predicted to release both acetylcholine and histamine,

was treated as hyperpolarizing or depolarizing respectively, depending on whether a target cell type is known to express the histamine receptor ort (a histamine-gated chloride channel).

5.A.2 Representing the model as a hexagonal convolutional neural network

Our end-to-end differentiable [2] DMN model of the fly visual system can be interpreted as a continuous-time neural ordinary differential equation (neural ODE) [15] with a deep convolutional recurrent neural network (convRNN) [62] architecture that is trained to perform a computer vision task using backpropagation through time (BPTT) [59, 77]. Our goal was to optimize a simulation of the fly visual system to perform a complex visual information processing task using optimization methods from deep learning. One hallmark of visual systems that has been widely exploited in such tasks are their convolutional nature [24, 37, 56, 34], i.e. the fact that the same computations are applied to each pixel of the visual input. To model the hexagonal arrangement of photoreceptors in the fly retina, we developed a hexagonal convolutational neural network in the widely used deep learning framework Pytorch [51] (ignoring neuronal superposition [12]), which we used for simulation and optimization of the model. We model columnar cell types including retinal cells, lamina monopolar and wide-field cells, medulla intrinsic cells, transmedullary cells, and T-shaped cells, as well as amacrine cells. The model comprises synapses from all neuropils and downstream and upstream projecting connections from the retina, lamina, and medulla.

5.A.3 Neuronal Dynamics

In detail, we simulated point neurons with voltages V_i of a postsynaptic neuron i, belonging to cell type t_i using threshold-linear dynamics, mathematically equivalent to commonly used formulations of firing-rate models [48]

$$\tau_{t_i}\dot{V}_i = -V_i + \sum_j s_{ij} + V_{t_i}^{\text{rest}} + e_i \tag{5.1}$$

Neurons of the same cell type share time constants, τ_{t_i}, and resting potentials, $V_{t_i}^{\text{rest}}$. Dynamic visual stimuli were delivered as external input currents e_i to the photoreceptor (R1-R8), for all other cell types, $e_i = 0$. In our model, instantaneous graded synaptic release from presynaptic neuron j to postsynaptic neuron i is described by

$$s_{ij} = w_{ij} f(V_j) = \alpha_{t_i t_j} \sigma_{t_i t_j} N_{t_i t_j \Delta u \Delta v} f(V_j), \tag{5.2}$$

comprising the anatomical filters in terms of the synapse count from EM-reconstruction, $N_{t_i t_j \Delta u \Delta v}$, at the offset location $\Delta u = u_i - u_j$ and $\Delta v = v_i - v_j$ in the hexagonal lattice between two types of cells, t_i and t_j, and further characterised by a sign, $\sigma_{t_i t_j} \in \{-1, +1\}$, and a non-negative scaling factor, $\alpha_{t_i t_j}$.

The synapse model in Equation 5.2 entails a trainable non-negative scaling factor per filter that is initialized as

$$\alpha_{t_i,t_j} = \frac{0.01}{\langle N_{t_i,t_j} \rangle_{\Delta u, \Delta v}},$$

with the denominator describing the average synapse count of the filter. Synapse counts, $N_{t_i t_j \Delta u \Delta v}$, and signs, $\sigma_{t_i t_j}$, from reconstruction and neurotransmitter and receptor profiling were kept fixed. The scaling factor was clamped during training to remain non-negative.

Moreover, at initialization, the resting potentials were sampled from a Gaussian distribution

$$V_{t_i}^{\text{rest}} \sim \mathcal{N}(\mu_{V^{\text{rest}}}, \sigma^2_{V^{\text{rest}}})$$

with mean $\mu_{V^{\text{rest}}} = 0.5$ (a.u.) and variance $\sigma^2_{V^{\text{rest}}} = 0.05$ (a.u.). The time constants were initialized at $\tau_{t_i} = 50$ms. The 50 task-optimized DMNs were initialized with the same parameter values. During training, in Euler integration of the dynamics, we clamped the time constants as $\tau_i = \max(\tau_i, \Delta t)$, so that they remain above the integration time step Δt at all times.

In total, the model comprises 45669 neurons and 1513231 synapses, across two-dimensional hexagonal arrays 31 columns across. Independently of the extent of the two-dimensional hexagonal arrays are the numbers of free parameters: 65 resting potentials, 65 membrane time constants, 604 scaling factors; and connectome determined parameters: 604 signs, and 2355 synapse counts. Thus, the number of free parameters in the visual system model is 734.

In the absence of connectome measurements, the number of parameters to be estimated is much larger. With $T = 65$ cell types (counting CT1 twice for the compartments in the medulla and lobula) and $C = 721$ cells per type for simplicity, the number of cells in our model would be $TC = 46,865$. Assuming an RNN with completely unconstrained connectivity and simple dynamics $\tau_i V_i = -V_i + \sum_j w_{ij} f(V_j) + V_i^{\text{rest}}$ we would have to find $(TC)^2 + 2(TC) = 2,196,421,955$ free parameters. Assuming a convolutional RNN with shared filters between cells of the same postsynaptic

type, shared time constants, and resting potential, the amount of parameters reduces drastically to $T^2C + 2T = 3,046,355$. Further assuming the same convolutional RNN but additionally convolutional filters are constrained to $F = 5$ visual columns, i.e. the number of presynaptic input columns in hexagonal lattice is $P = 3F(F + 1) + 1$, the amount of parameters reduces to $T^2P + 2T = 384,605$. Assuming as in our connectome only $Q = 604$ connections between cell types exist, this reduces the number of parameters further to $QP + 2T = 55,185$. Instead of parametrizing each individual synapse strength, we assume that synapse strength is proportional to synapse count from the connectome times a scalar for each filter, reducing the number of parameters to $Q + 2T = 734$ while providing enough capacity for the DMNs to yield realistic tuning to solve the task.

Convolutions using scatter and gather operations For training the network, we compiled the convolutional architecture specified by the connectome and the sign constraints to a graph representation containing (1) a collection of parameter buffers shared across neurons and/or connections, (2) a collection of corresponding index buffers indicating where the parameters relevant to a given neuron or connection can be found in the parameter buffers, and (3) a list of pairs (presynaptic neuron index, postsynaptic neuron index) denoting connectivity. This allowed us to efficiently simulate the network dynamics via Euler integration using a small number of element-wise, scatter, and gather operations at each time step. We found that this is more efficient than using a single convolution operation, or performing a separate convolution for each cell type, since each cell type has its own receptive field - some much larger than others - and the number of cells per type is relatively small.

5.A.4 Optic flow task

Model training An optic flow field for a video sequence consists of a 2D vector field for each frame. The 2D vector at each pixel represents the magnitude and direction of the apparent local movement of the brightness pattern in an image.

We frame the training objective as a regression task

$$\hat{\mathbf{Y}}[n] = \text{Decoder}(\text{DMN}(\mathbf{X}[0], ..., \mathbf{X}[n])),$$

with $\hat{\mathbf{Y}}$ the optic flow prediction, and $\mathbf{X}$ the visual stimulus sequence from the Sintel dataset, both sampled to a regular hexagonal lattice of 721 columns. With the

objective to minimize the square error loss between predicted optic flow and target optic flow fields, we jointly optimized the parameters of both the decoder and the visual system network model described above.

In detail, for training the network, we added randomly augmented grey-scaled video sequences from the Sintel dataset sampled to a regular hexagonal lattice of 721 columns to the voltage of the eight photoreceptor cell types (Fig. 5.1f, Equation 5.1). We denote a sample from a minibatch of video sequences as $\mathbf{X} \in \mathbb{R}^{N,C}$ with N the number of time steps, and C the number of photoreceptor columns. The dynamic range of the input lies between 0 and 1. Input sequences during training entailed 19 consecutive frames drawn randomly from the dataset and resampled to match the integration rate. The original framerate of 24 Hz and 19 frames lead to a simulation of 792ms. We did not find that an integration time step smaller than 20 ms, i.e. a framerate of 50 Hz after resampling, yielded qualitatively superior task performance nor more realistic tuning predictions. We interpolated the target optic flow in time to 50 Hz temporal resolution, instead of resampling it. To increase the amount of training data for better generalization, we augmented input and target sequences as described further below. At the start of each epoch, we computed an initial state of the network's voltages after 500ms of grey stimuli presentation to initialize the network at a steady state for each minibatch during that epoch. The network integration given input $\mathbf{X}$ results in simulated sequences of voltages $\mathbf{V} \in \mathbb{R}^{N,T_C}$ with T_C the total number of cells. The subset of voltages, $\mathbf{V}_{\text{out}} \in \mathbb{R}^{N,D,C}$, of the D cell types in the black box in Fig. 5.1g was passed to a decoding network. For decoding, the voltage was rectified to avoid that the network finds biologically implausible solutions by encoding in negative dynamic ranges. Further, it was mapped to cartesian coordinates to apply Pytorch's standard spatial convolution layers for decoding and on each timestep independently. In the decoding network, one layer implementing spatial convolution, batch normalization, softplus activation, and dropout, followed by one layer of spatial convolution, transformed the D feature maps into the two-dimensional representation of the estimated optic flow, $\hat{\mathbf{Y}} \in \mathbb{R}^{N,2,C}$.

Using stochastic gradient descent with adaptive moment estimation ($\beta_1 = 0.9$, $\beta_2 = 0.999$, learning rate decreased from $5{\times}10^{-5}$ to $5{\times}10^{-6}$ in ten steps over iterations, batch size of four) and the automatic gradient calculation of the fully differentiable pipeline, we optimized the biophysical parameters through backpropagation through time such that they minimize the L2-norm between the predicted optic flow, $\hat{\mathbf{Y}}$, and the groundtruth optic flow, $\mathbf{Y}$:

$$L(\mathbf{Y}, \hat{\mathbf{Y}}) = ||\mathbf{Y} - \hat{\mathbf{Y}}||$$

We additionally regularized the shared resting potentials for 150,000 iterations, using stochastic gradient descent without momentum, based on time-averaged responses to naturalistic stimuli of the central column cell of each cell type, t_{central}, to encourage configurations of resting potentials that lead to nonzero and nonexploding activity in all neurons in the network. We weighted these terms independently with $\gamma = 1$, encouraging activity greater than a, and $\delta = 0.01$, encouraging activity less than a. We chose $\lambda_V = 0.1$ and $a = 5$ in arbitrary units. With B being the batch size and T the number of all cell types, the regularizer is

$$R(V) = \frac{\lambda_V}{BT} \sum_b \sum_{t_{\text{central}}} \begin{cases} \gamma(\bar{V} - a)^2, & \text{if } \bar{V} = \frac{1}{N} \sum_n V_{bt_{\text{central}}}[n] \leq a \\ \delta(\bar{V} - a)^2, & \text{if } \bar{V} > a. \end{cases}$$

We regularly checkpointed the above error measure $L(\mathbf{Y}, \hat{\mathbf{Y}})$ averaged across a held out validation set of Sintel video clips. Models generalized on optic flow computation after round about 250,000 iterations, yielding functional candidates for our fruit fly visual system models that we analyzed with respect to their tuning properties.

Task-optimization dataset We optimized the network on 23 sequences from the publicly available computer-animated movie Sintel [13]. The sequences have 20-50 frames, at a frame rate of 24 frames per second and a pixel resolution of 1024x436. The dataset provides optical flow in pixel space for each frame after the first of each sequence. Since the integration time steps we use are faster than the actual sampling rate of the sequences, we resample input frames accordingly over time and interpolate the optic flow.

Fly-eye rendering We first transformed the RGB pixel values of the visual stimulus to normalized greyscale between 0 and 1. We translated cartesian frames into receptor activations by placing simulated photoreceptors in a two-dimensional hexagonal array in pixel space, 31 columns across resulting in 721 columns in total, spaced 13 pixels apart. The transduced luminance at each photoreceptor is the greyscale mean value in the 13×13-pixel region surrounding it.

Augmentation We used (1) random flips of input and target across one of the three principal axes of the hexagonal lattice, (2) random rotation of input and target around its six-fold rotation axis, (3) adding element-wise Gaussian noise with mean zero and variance $\sigma_n = 0.08$ to the input X (then clamped at 0) (4) random adjustments of contrasts, $\log c \sim \mathcal{N}(0, \sigma_c^2 = 0.04)$, and brightness, $b \sim \mathcal{N}(0, \sigma_b^2 = 0.01)$, of the input with $X' = c(X - 0.5) + 0.5 + cb$.

In addition, we strided fly-eye rendering across the rectangular raw frames in width, subsampling multiple scenes from one. We ensured that such subsamples from the same scene do not distribute across training and validation sets. Input sequences in chunks of 19 consecutive frames were drawn randomly in time from the full sequences.

Black-box decoding network The decoding network is feedforward, convolutional and has no temporal structure. Aspects of the architecture are explained in the paragraph Model training. The spatial convolutions have a filter size of 5×5. The first layer transforms the $D = 34$ feature maps to an eight-channel intermediate representation, that is further translated by an additional convolutional layer to a three-channel intermediate representation of optic flow. The third channel is used as shared normalization of each coordinate of the remaining two-dimensional flow prediction. The decoder uses Pytorch-native implementations for two-dimensional convolutions, batch normalization, softplus activation, and dropout. We initialized its filter weights homogeneously at 0.001.

5.A.5 Model characterization

Task error To rank models based on their task performance, we computed the standard optic flow metric of average end-to-end point error (EPE) [20] which calculates the average over all timesteps and pixels (i.e. here columns) of the error

$$\mathrm{EPE}(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{1}{NC} \sum_n \sum_c \sqrt{(y_{1c}[n] - \hat{y}_{1c}[n])^2 + (y_{2c}[n] - \hat{y}_{2c}[n])^2}.$$

between predicted optic flow and groundtruth optic flow, and averaged across the held out validation set of Sintel sequences.

Importance of task optimization and connectome constraints We generated DMNs with different constraints in order to assess their relative importance for predicting tuning properties. First, we studied the importance of task-optimization of DMN parameters. We created 50 DMNs with random Gaussian distributed

parameters, and task-optimized only their decoding network, in order to obtain baseline values for both the task error and the accuracy of predicting tuning curves without task-optimization of the DMN.

In the Full DMN, we constrained single synapses by connectome cell-type connectivity, cell connectivity, synapse counts, and synapse signs (Equation 5.2) and task-optimized the non-negative type-to-type unitary synapse scaling factor $\boldsymbol{\alpha}_{t_i,t_j}$. Next, we trained five additional task-optimized DMNs with different connectome constraints (Fig. 5.2d and Fig. 5.12a–d).

In these five additional types of DMN, we additionally task-optimized the terms in bold, rather than using connectome measurements, related to synaptic currents from a presynaptic cell j to a postsynaptic cell i: (1) known single cell connectivity, unknown synapse count: $w_{ij} = \sigma_{t_i,t_j}\boldsymbol{m}_{t_i,t_j,\Delta u,\Delta v}$, (where $\boldsymbol{m}_{t_i,t_j,\Delta u,\Delta v}$ is non-negative) (2) known cell-type connectivity, unknown single cell connectivity and synapse counts: $w_{i,j} = \sigma_{t_i,t_j}\boldsymbol{m}_{t_i,t_j,-3<\Delta u,\Delta v,\Delta u+\Delta v<3}$ (i.e., for all connected cell-types, a connection weight was learned for all cells up to a distance of 3 columns in hexagonal coordinates, with known signs), (3) known single cell connectivity and synapse counts, but unknown synapse signs $w_{i,j} = \boldsymbol{\alpha}_{t_i,t_j}\boldsymbol{\sigma}_{t_j}N_{t_i,t_j,\Delta u,\Delta v}$ (i.e., connection weights were fixed by measurements, but signs optimized), (4) known single cell connectivity, but unknown synapse signs and synapse counts $w_{i,j} = \boldsymbol{w}_{t_i,t_j,\Delta u,\Delta v}$, (i.e., all non-zero connection weights were optimized, including their signs) or (5) known cell type connectivity, unknown single cell connectivity, synapse counts, and synapse signs $w_{i,j} = \boldsymbol{w}_{t_i,t_j,-3<\Delta u,\Delta v,\Delta u+\Delta v<3}$ (i.e., for all connected cell-types, a connection weight and sign was learned for all cells up to distance of 3 columns). We trained 50 models per DMN-type. The task-optimized parameters in each case are highlighted using bold symbols. We randomly initialized the models with: $\boldsymbol{m}_{t_i,t_j}, \boldsymbol{w}_{t_i,t_j} \sim \mathcal{N}(0, \frac{2}{n_{\text{in}}})$ where n_{in} is the number of cell connections and $\boldsymbol{m}$ non-negative, and $\boldsymbol{\sigma}_{t_j} \in \{-1, 1\}$ with equal probability.

Unconstrained CNN We trained unconstrained, fully convolutional neural networks on the same dataset and task to provide an estimate of a lower bound for the task error of the DMN. Optic flow was predicted by the CNN from two consecutive frames

$$\hat{Y}[n] = \text{CNN}(X[n], X[n-1]).$$

with the original frame rate of the Sintel movie. We chose 5 layers for the CNN with 32, 92, 136, 8, 2 channels respectively and kernel size 5 for all but the first layer which kernel size is 1. Each layer performs a convolution, batch normalization, and ELU activation except the last layer which only performs a convolution. We optimized an ensemble of 5 unconstrained CNNs with 414,666 free parameters each using the same loss function, $L(Y, \hat{Y})$, as for the DMN. We used the same dataset, i.e. hexagonal sequences and augmentations from Sintel, for training and validating the CNN as for training and validating the DMN, allowed by two custom modules mapping from hexagonal lattice to cartesian map and back.

Circular flash stimuli To evaluate the contrast selectivity of cell types in task-constraint model candidates, we simulated responses of each DMN to circular flashes. The networks were initialized at an approximate steady state after 1s of grey-screen stimulation. Afterwards the flashes were presented for 1s. The flashes with a radius of 6 columns were ON (intensity $I = 1$) or OFF ($I = 0$) on grey ($I = 0.5$) background. We integrated the network dynamics with an integration time step of 5 ms. We recorded the responses of the modeled cells in the central columns to compute the flash response index.

Flash response index To derive the contrast selectivity of a cell type, t_i, we computed the flash response index as

$$\mathrm{FRI}_{\mathrm{t_i}} = \frac{r^{\mathrm{peak}}_{\mathrm{t_{central}}}(I=1) - r^{\mathrm{peak}}_{\mathrm{t_{central}}}(I=0)}{r^{\mathrm{peak}}_{\mathrm{t_{central}}}(I=1) + r^{\mathrm{peak}}_{\mathrm{t_{central}}}(I=0)} \tag{5.3}$$

from the non-negative activity

$$r^{\mathrm{peak}}_{\mathrm{t_{central}}}(I) = \max_n V_{t_{\mathrm{central}}}[n](I) + |\min_{n,I} V_{t_{\mathrm{central}}}[n](I)|,$$

from voltage responses $V_{t_{\mathrm{central}}}[n](I)$ to circular flash stimuli of intensities $I \in \{0, 1\}$ lasting for 1s after 1s of grey stimulus. We note that our index quantifies whether the cell depolarizes to ON- or to OFF-stimuli. However, cells like R1-R8, L1, and L2 can be unrectified, i.e., sensitive to both light increments and light decrements, which is not captured by our index.

For the p-values reported in the results, we performed a binomial test with probability of correct prediction 0.5 (H0) or greater (H1) to both test whether the median FRI from the DMN-ensemble and the task-optimal model can predict the contrast preferences. Additionally, we found for each individual cell type across 50 DMS that predictions for 29 out of 31 cell types are significant ($P < 0.05$, binomial).

Moving edge stimuli To predict the motion sensitivity of each cell type in task-constrained DMNs, we simulated the response of each network, initialized at an approximate steady state after 1s of grey-screen stimulation, to custom generated edges moving to 12 different directions, $\theta \in [0°, 30°, 60°, 90°, 120°, 150°, 180°, 210°, 240°, 270°, 300°, 330°]$. We integrated the network dynamics with an integration time step of 5ms. ON-edges ($I = 1$) or OFF-edges ($I = 0$) moved on grey ($I = 0.5$) background. Their movement ranged from -13.5° to 13.5° visual angle and we moved them at six different speeds, ranging from $13.92°/s$ to $145°/s$ ($S \in [13.92°/s, 27.84°/s, 56.26°/s, 75.4°/s, 110.2°/s, 145.0°/s]$). In Fig. 5.2d, we report the correlation between predicted motion tuning curves to the single experimentally measured tuning curve. We take the maximum correlation across six investigated speeds in order to make the correlation measure robust to potential variations in preferred speeds.

Direction selectivity index We computed a direction selectivity index[45] of a particular type t_i as

$$\mathrm{DSI}_{\mathrm{t_i}}(I) = \frac{1}{|\mathbb{S}|} \sum_{S \in \mathbb{S}} \frac{|\sum_{\theta \in \Theta} r^{\mathrm{peak}}_{\mathrm{t_{central}}}(\mathrm{I}, \mathrm{S}, \theta) \exp(i\theta)|}{\max_{I \in \mathbb{I}} |\sum_{\theta} r^{\mathrm{peak}}_{\mathrm{t_{central}}}(\mathrm{I}, \mathrm{S}, \theta)|} \tag{5.4}$$

from rectified peak voltages

$$r^{\mathrm{peak}}_{\mathrm{t_{central}}}(\mathrm{I}, \mathrm{S}, \theta) = \max_{n} V^{+}_{t_{\mathrm{central}}}[n](\mathrm{I}, \mathrm{S}, \theta), \tag{5.5}$$

elicited from moving edge stimuli. We rectify to quantify the tuning on the effective output of the cell, and to avo that the denominator becomes zero. We parameterized movement angle $\theta \in \Theta$, intensities $I \in \mathbb{I}$, and speeds $S \in \mathbb{S}$ of the moving edges. To take the response magnitudes into account for comparing DSI for ON- and for

OFF-edges, we normalized by the maximum over both intensities in the denominator. To take different speeds into account, we averaged over $\mathbb{S}$.

Normalization of model neural activity for averaging across models in a cluster Threshold linear networks have arbitrary units for the voltages and currents. Therefore, we normalized the neural activity before averaging the neural activity predictions from different models. For a single cell or cell type t, we first divided responses (voltages or rectified voltages) by the root mean square across the cell's responses to the naturalistic stimuli:

$$r_t^{||\cdot||=1}[n] = \frac{r_t[n]}{||\frac{1}{SN}\mathbf{R}_t^{\text{nat.}}||_2}, \tag{5.6}$$

where $\mathbf{R}_t^{\text{nat.}} \in \mathbb{R}^{S,N}$ is the cell's response vector to S sequences from the Sintel dataset with N timesteps and $r_t[n]$ is the cell's response to any stimuli. This normalization makes averages (Fig. 5.4a-b, d-f; Fig. 5.8–Fig. 5.9) independent to variation in the scale of neural activity from model to model. We normalized input currents equivalently (Fig. 5.4b and Fig. 5.8–Fig. 5.9) by the same normalization factor. We exclude solutions for which the denominator becomes zero.

Determining whether a cell type with asymmetric inputs counts as direction selective We counted a cell type as direction selective if the DSIs from its synthetic measurements were larger than 99% of DSIs from non-motion selective cell types (i.e. those with symmetric filters). We note, however, that estimates of the spatial asymmetry of connectivity from existing connectome reconstructions can be noisy.

For deriving the 99%-threshold, we first defined a distribution $p(d^*|\mathrm{t}_{\text{sym}})$ over the DSI for non-direction selective cells, from peak responses to moving edges of cell types with symmetric inputs, t_{sym}. We computed that distribution numerically by sampling

$$d^* = \frac{|\sum_{\theta^*} r^{\text{peak}}_{\mathrm{t}_{\text{central}}}(\mathrm{I}, \mathrm{S}, \theta^*) \exp i\theta|}{|\sum_{\theta} r^{\text{peak}}_{\mathrm{t}_{\text{central}}}(\mathrm{I}, \mathrm{S}, \theta)|}$$

for 100 independent permutations of the angle θ^*. We independently computed d^* for all stimulus conditions, models, and cell types with symmetric inputs. From $p(d^*|\mathrm{t}_{\text{sym}})$, we derived the threshold $d_{\text{thresh}} = 0.357$ as the 99% quantile of the random variable d^*, meaning that the probability that a realization of $d^* > d_{\text{thresh}}$ is less than 1% for cell types with symmetric inputs. To determine whether an

asymmetric cell type counts as direction selective, we tested whether synthetically measuring direction selectivity larger than d_{thresh} in that cell type is binomial with probability 0.1 (H0) or greater (H1). We identified 12 cell types with asymmetric inputs (T4a, T4b, T4c, T4d, T5a, T5b, T5c, T5d, TmY3, TmY4, TmY5a, TmY18) as direction selective ($P < 0.05$) from our models, and seven cell types with asymmetric inputs to not count as direction selective (T2, T2a, T3, Tm3, Tm4, TmY14, TmY15; see Fig. 5.7 as reference for cell types with symmetric and asymmetric inputs).

UMAP and clustering We first simulated central column responses to naturalistic scenes (24Hz Sintel video clips from the full augmented dataset) with an integration time step of 10 ms. We clustered models in feature space of concatenated central column responses and sample dimension. Next, we computed a nonlinear dimensionality reduction to 2d (UMAP), and finally fitted Gaussian mixtures of 2 to 5 components to the embedding to label the clusters based on the Guassian mixture model with the number of components that minimize the Bayesian information criterion, using the python libraries umap-learn and scikit-learn [7, 52].

Single-ommatidium flashes To derive spatio-temporal receptive fields of cells, we simulated the response of each network to single ommatidium flashes. Flashes were ON ($I = 1$) or OFF ($I = 0$) on grey ($I = 0.5$) background and presented for [5, 20, 50, 100, 200, 300] ms after 2 s of grey-screen stimulation and followed by 5 s of grey-screen stimulation.

Spatio-temporal, spatial and temporal receptive fields We derived the spatio-temporal receptive field (STRF) of a cell type t_i as the baseline subtracted responses of the central column cell to single ommatidium flashes $J(u, v)$ at ommatidium locations (u, v):

$$\text{STRF}_{t_{\text{central}}}[n](u, v) = V_{t_{\text{central}}}[n](J(u, v)) - V_{t_{\text{central}}}[n = 0](J(u, v)).$$

We derived spatial receptive fields, SRFs, from the responses to flashes (20 ms in Fig. 5.4d) $J(u, v)$ at the point in time at which the response to the central ommatidium impulse is at its extremum:

$$\text{SRF}(u, v) = \text{STRF}(n = \arg\max_n |\text{STRF}[n](0, 0)|, u, v).$$

We derive temporal receptive fields, TRFs, from the response to a flash $J(0,0)$ at the central ommatidium: $\mathrm{TRF}[n] = \mathrm{STRF}[n](0,0)$. For averaging receptive fields across multiple models, we first normalize the voltages as described above.

Maximally excitatory naturalistic and artificial stimuli First, we found the naturalistic maximally excitatory stimulus, $\mathbf{X}^*$, by identifying the Sintel video clip, $\mathbf{X}$, from the full dataset with geometric augmentations that elicited the highest possible response in the central column cell of a particular cell type in our models.

$$\mathbf{X}^* = \underset{\mathbf{X}\in\mathrm{Sintel}}{\arg\max} V_{t_{\mathrm{central}}}(\mathbf{X}).$$

Next, we regularized the naturalistic maximally excitatory stimulus, to yield $\mathbf{X}'$, capturing only the stimulus information within the receptive field of the cell, with the objective to minimize

$$L(\mathbf{X}') = \sum_n \|V_{t_{\mathrm{central}}}(\mathbf{X}^*)[n] - V_{t_{\mathrm{central}}}(\mathbf{X}')[n]\|^2 + \frac{1}{C}\sum_c \|\mathbf{X}'[n,c] - 0.5\|^2.$$

The first summand maintains the exact central response to $\mathbf{X}^*$, while the second sets the redundant stimulus information outside of the receptive field to grey ($I = 0.5$).

In addition, we computed artificial maximally excitatory stimuli[75].

Model selection To describe the most data-consistent motion tuning mechanisms predicted by the ensemble at the level of single-cell currents, for Fig. 5.8, Fig. 5.8, and Fig. 5.9, we automatically selected those models from the ensemble with tuning matching to empirical data. Specifically, we selected models with correct contrast tuning in the respective target cells and their inputs (Fig. 5.4c and Fig. 5.9d), with the direction selectivity index larger than the threshold d^* derived above, and with a correctly predicted preferred direction (45° acceptance angle, assuming 225° for TmY3).

5.A.6 Training synthetic connectomes

Training feedforward synthetic groundtruth-connectome networks Sparsified feedforward neural networks with 6 hidden layers (linear transformations sandwiched between rectifications) with equal number of neurons in each hidden layer functioned

as groundtruth-connectome networks. The main results describe networks with 128 neurons per hidden layer. We interpret the individual units as neurons with voltage

$$V_i = \sum_j s_{ij} + V_i^{\text{rest}} = \sum_j \sigma_j c_{ij} m_{ij} f(V_j) + V_i^{\text{rest}},$$

with presynaptic inputs s_{ij} and resting potentials V_i^{rest}. The connectome-constrained synapse strength, w_{ij}, is characterized by the adjacency matrix c_{ij}, the signs, σ_j, and the non-negative weight magnitudes m_{ij}. $c_{ij} = 1$ if the connection exists, else $c_{ij} = 0$. To respect Dale's law, the signs were tied to the presynaptic identity, j.

We identified the parameters σ_j, m_{ij}, and V_i^{rest} by task-optimization on handwritten digit classification (MNIST)[36]. We determined adjacency matrices, c_{ij}, for a given connectivity percentage using an iterative local pruning technique, the Lottery Ticket Hypothesis algorithm[23]. The algorithm decreases the connectivity percentage of the groundtruth-connectome networks while maintaining high task accuracy.

We optimized the groundtruth-connectome networks and all simulated networks described below in Pytorch with stochastic gradient descent with adaptive moment estimation (ADAM with AMSGrad), learning rate 0.001, batch size 500, and an exponentially decaying learning rate decay factor of 0.5 per epoch. To constrain the weight magnitudes to stay non-negative, we clamped the values at zero after each optimization step (projected gradient descent). The parameters after convergence minimize the cross-entropy loss between the predicted and the groundtruth classes of the handwritten digits.

Simulated networks with known connectivity and unknown strength Simulated networks inherited connectivity, c_{ij}, and synapse signs, σ_j, from their respective groundtruth-connectome networks. In simulated networks, signs and connectivity were held fixed. Weight magnitudes, m_{ij}, and resting potentials, V_i^{rest}, were initialized randomly and task-optimized. Just like groundtruth-connectome networks, simulated networks were trained on the MNIST handwritten digit classification task until convergence.

Simulated networks with known connectivity and known strength Alternatively, we imitate measurements of synaptic counts from the groundtruth weight magnitudes:

$$\tilde{m}_{ij} = m_{ij}\epsilon_{ij} \text{ with } \epsilon_{ij} \sim \mathcal{U}(1-\sigma, 1+\sigma),$$

with multiplicative noise to imitate spurious measurements. We used $\sigma = 0.5$ for the main results. Weight magnitudes were initialized at the measurement,

$\tilde{m}_{ij}$, and task-optimized on MNIST with the additional objective to minimize the squared distance between optimized and measured weight magnitudes, $\tilde{m}_{ij}$ (L2 constraint, Gaussian weight magnitude prior centered around the simulated network's initialization). We weighted the L2 constraint ten times higher than the cross-entropy objective to keep weight magnitudes of the simulated networks close to the noisy connectome measurements. Resting potentials, V_i^{rest}, were again initialized randomly and task-optimized.

Measuring groundtruth-simulated network similarity Groundtruth-simulated network similarity was measured by calculating the median Pearson's correlation of tuning responses (rectified voltages) of corresponding neurons in the groundtruth-simulated network pair. In each of the 6 hidden layers, $N = 100$ randomly-sampled neurons were used for comparison. Response tuning was measured over input stimuli from the MNIST test-set ($N = 10,000$ images). Results are medians over all hidden layers and over 25 groundtruth-simulation network pairs.

5.B Additional figures

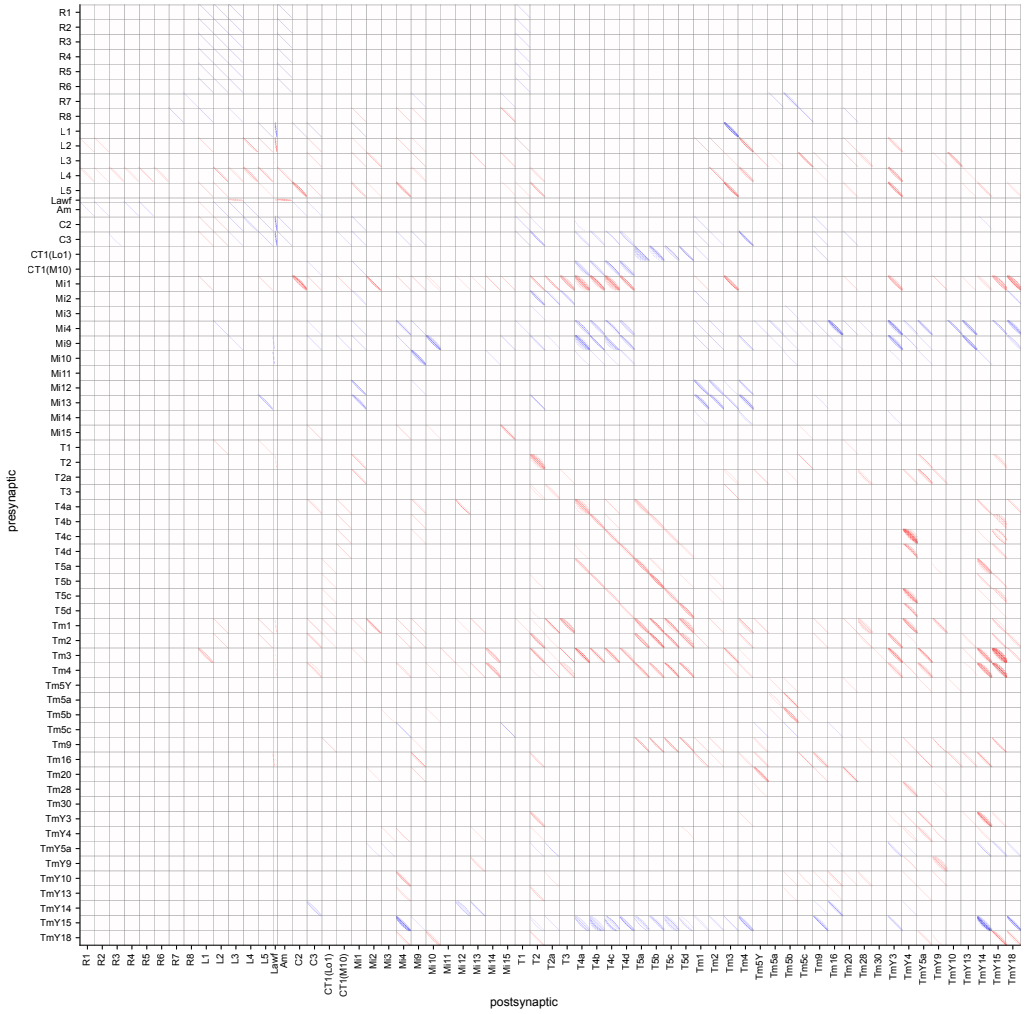

Figure 5.6: **Cell connectivity.** The matrix shows how cells of the 64 cell types within the inner 91 columns (of 721) of the recurrent convolutional DMN connect, either by excitatory connections (red) or inhibitory connections (blue).

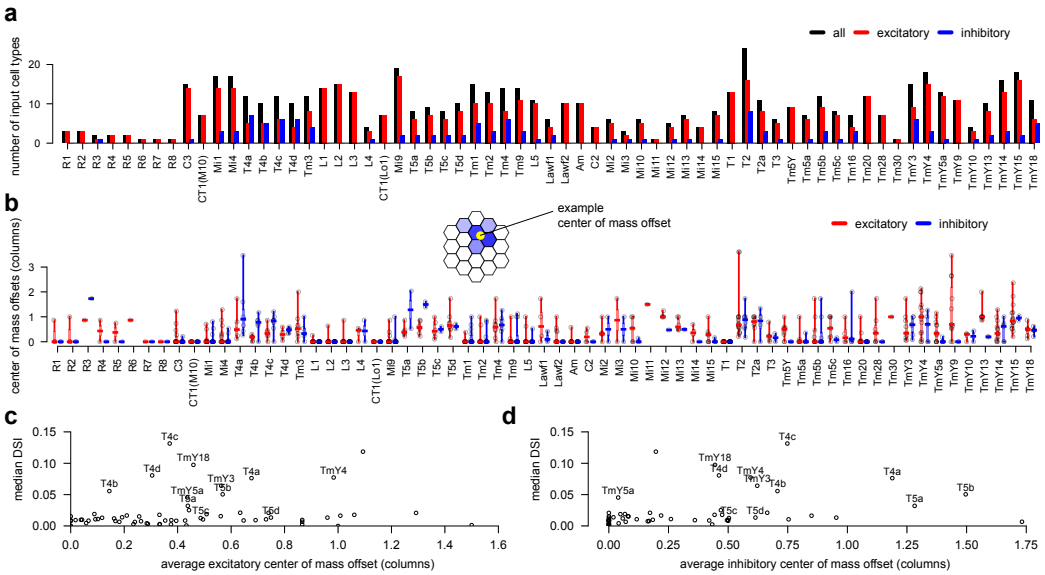

Figure 5.7: **Statistics of inhibitory and excitatory synapse inputs. a:** Number of input cell types per cell type. **b:** Center of mass offsets of synaptic input. **c:** Average excitatory and **d:** inhibitory center of mass offset of synaptic inputs against median predicted direction selectivity index for all cell types. Datapoints for cell types that were predicted as significantly motion selection are labeled.

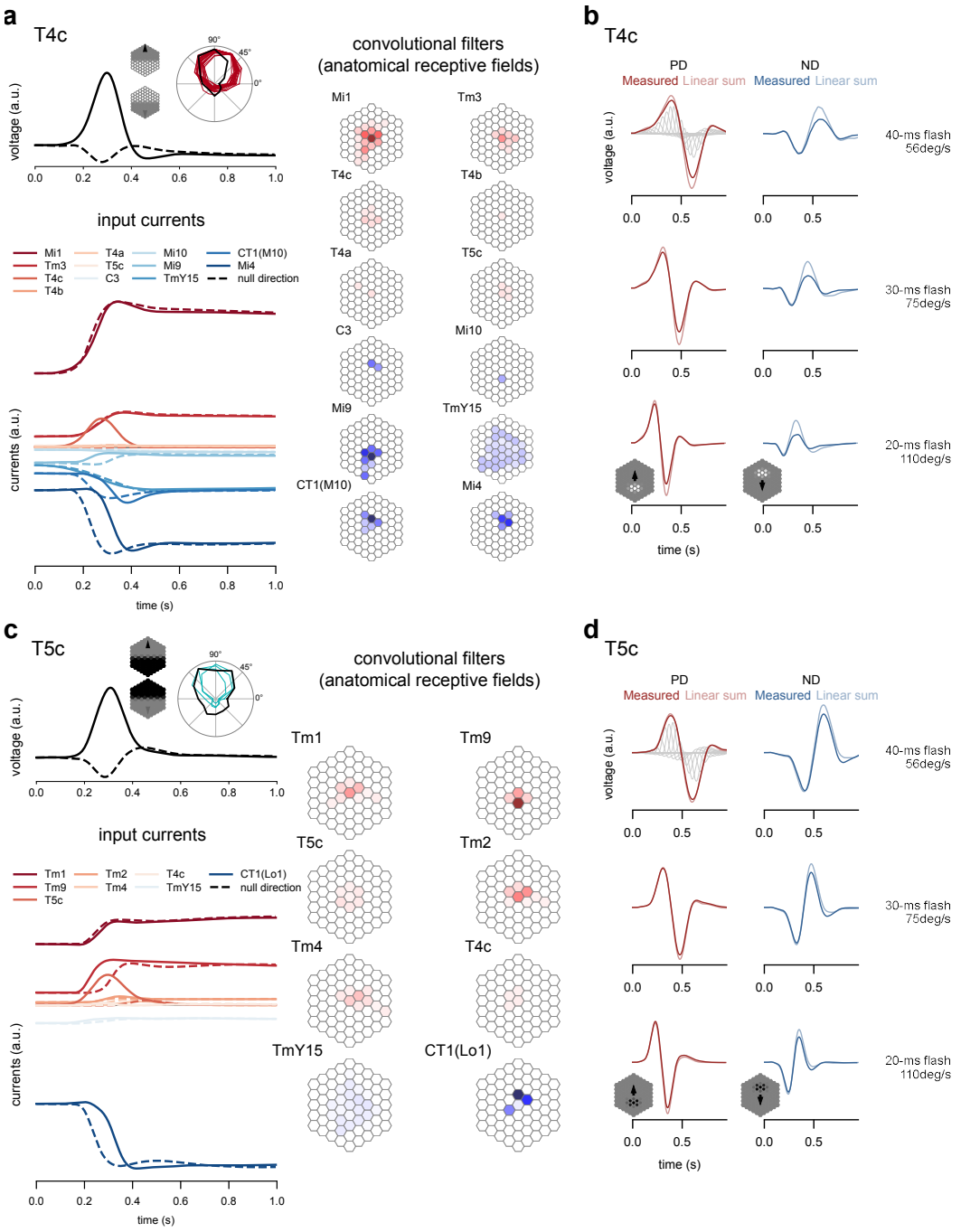

Figure 5.8: **T4 and T5 motion detection mechanisms hypothesized by the model. a:** The four T4 cell types detect ON-edge motion towards the four cardinal directions (here T4c). An ON-edge moving towards the preferred direction (PD) of the cell elicits a high depolarization in the central T4 cell (black, solid). In contrast, an edge moving towards the null direction (ND) of the cell elicits a wiggle from weak hyperpolarization to weak depolarization (black, dashed). (Continued on the next page)

Fig. 5.8, continued: We characterize the motion detection mechanism by displaying the PD- and ND-responses of the T4 cell type, and the temporal and spatial contributions of its input cell types according to our connectome-measurement constrained model. Across all T4 cell types, our model predicts that the depolarization in response to PD motion is mainly driven by excitatory Mi1 current inputs (darkest red, solid) from roughly a two-column radius of Mi1 cells. The PD-motion response is increased through excitatory inputs from the neighboring T4 cells of the same type (third darkest red, solid) with the center of mass located towards the leading side of the receptive field (i.e. the motion stimulus towards the PD reaches those T4 cells first, enabling this mechanism). However, for ND-motion the neighboring T4 cells do not provide any excitatory currents (third darkest red, dashed). Tm3 cells provide additional excitatory currents that are, as for Mi1 cells, roughly agnostic to PD vs ND motion. For ND-motion, Mi4 cells decrease excitatory currents from Mi1 by providing roughly matching inhibitory currents from the trailing side of the receptive field (darkest blue, dashed). In contrast, for PD-motion, the inhibition from Mi4 cells is delayed (through the spatial layout and potentially neural time constants not characterized here; darkest blue, solid), which allows a strong depolarization of the T4 cell. CT1 shadows Mi4 in that it provides a similar but weaker inhibition from the same location of the receptive field (second darkest, blue). Noteworthy, our model suggests roles and an additional mechanism for Mi9 cells in **b:** and TmY15 cells in **c:**: both can contribute to the motion detection mechanism by different inhibitory mechanisms for PD-motion with respect to ND-motion. **b:** This figure should be compared to Gruntman et al. 2018, Fig. 4f. Predicted T4c responses to bars moving in the PD (left column) and in the ND (right column) at speeds of $56°/s$, $75°/s$, and $110°/s$ ('Measured', saturated red and blue, speeds varied from top to bottom row). Responses to moving bars are overlaid with the linear sum of responses to the individually flashed frames that constitute the moving bar video sequence ('Linear sum', faint red and blue). Faint grey traces in the background of the first panel show individual flash responses before linear summation. The duration that the flash stimulus was presented in each location precisely matched the duration that the flash remained at the location in the moving bar sequence. Bars were approximately 9° wide and 20.25° high and moved across 45° with respect to the receptive field in the center. **c:** The four T5 cell types detect OFF-edge motion towards the four cardinal directions (here T5c). An OFF-edge moving towards the preferred direction (PD) of the cell elicits a high depolarization in the central T5 cell (black, solid). In contrast, an edge moving towards the null direction (ND) of the cell elicits a wiggle from weak hyperpolarization to weak depolarization (black, dashed). We characterize the motion detection mechanism by displaying the PD- and ND-responses of the T5 cell type, and the temporal and spatial contributions of its input cell types according to our connectome-measurement constrained model. (Continued on the next page)

Fig. 5.8, continued: **d:** Same as (b) for T5c. Across all T5 cell types, our model predicts that Tm1 and Tm9 cell types contribute to the T5 cell depolarization with excitatory input currents in response to moving edges. Tm1 inputs come from roughly a centered, two-column radius of Tm1 cells and Tm9 inputs from one column offset towards the leading side of the receptive field. We observe delayed excitation from Tm9 cells in all cases for ND-motion vs PD-motion. As for T4 cells, the PD-motion response is increased through excitatory inputs from the neighboring T5 cells of the same type. For ND-motion, the neighboring T5 cells do not provide any excitatory currents. For ND-motion, CT1(Lo1) cells decrease excitatory currents by providing strong inhibitory currents from the trailing side of the receptive field. In contrast, for PD-motion, the decrease from CT1(Lo1) cells is delayed, which allows a strong depolarization of the T5 cell to discriminate motion.

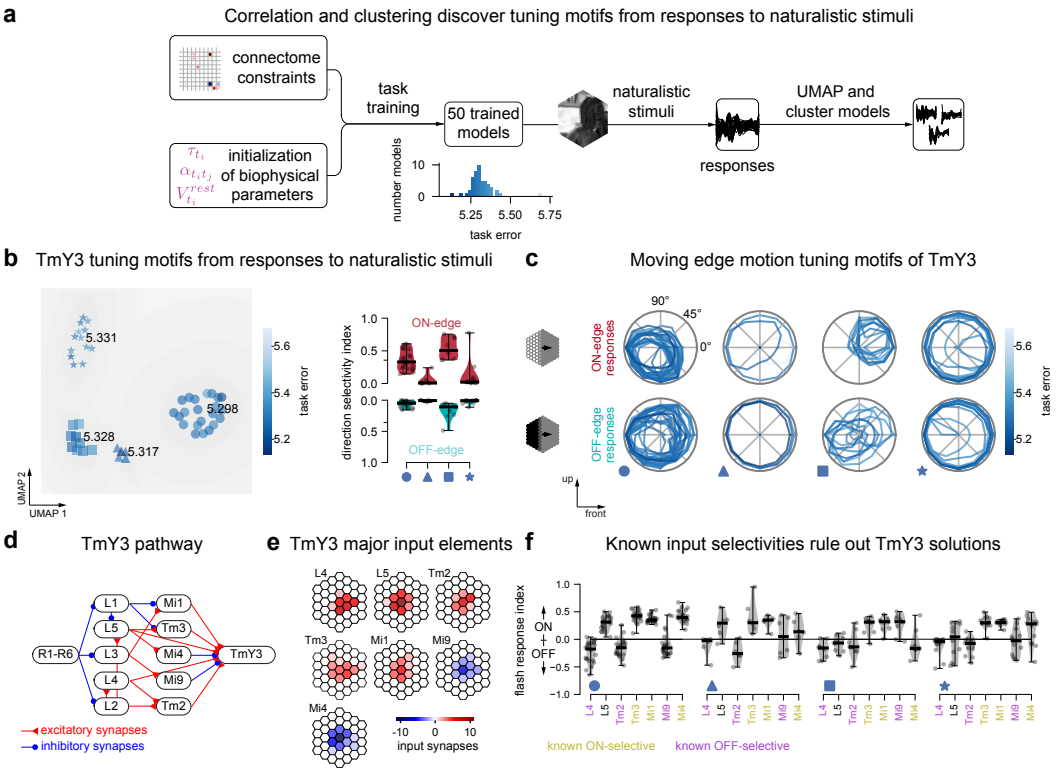

Figure 5.9: **DMNs suggest that TmY3 neurons compute motion independently of T4 and T5 neurons.** **a:** We clustered 50 DMNs after performing nonlinear dimensionality reduction of their responses to naturalistic scenes for each cell type, and aimed to identify whether clusters correspond to qualitatively different tuning mechanisms. **b:** Dimensionality reduction on TmY3 responses to naturalistic stimuli reveals 4 clusters of DMNs with average task errors 5.298 (circle), 5.317 (triangle), 5.328 (square) and 5.331 (star). Across clusters, TmY3 shows different strengths of direction selectivity (evaluated with moving edge stimuli). ON-edge direction selectivity is strong in the first and the third cluster. **c:** Normalized peak responses of TmY3 to moving edge stimuli in the DMNs of each cluster. **d:** Major cell types and synaptic connections in the pathway that projects onto TmY3 (simplified). **e:** The input elements of TmY3 with the highest amount of synapses are L4, L5, Tm2, Tm3, Mi1, Mi9, and Mi4. The asymmetries of their projective fields could allow TmY3 to become motion selective. **f:** Dependencies between TmY3 tuning and the contrast preference of its input cells. For clusters in which TmY3 is motion selective, cluster 1 (TmY3 tuning to downwards/front-to-back motion, circular marker) indicates ON-selectivity for Tm3, Mi1, and Mi4 cells, and OFF-selectivity for L4, Tm2, and Mi9 cells, in agreement with known selectivities. In contrast, cluster 3 (TmY3 tuning to upwards/back-to-front motion, square marker) indicates ON-selectivity for Mi9 in contradiction to the known selectivities and hence ruling out the third TmY3 tuning solution.

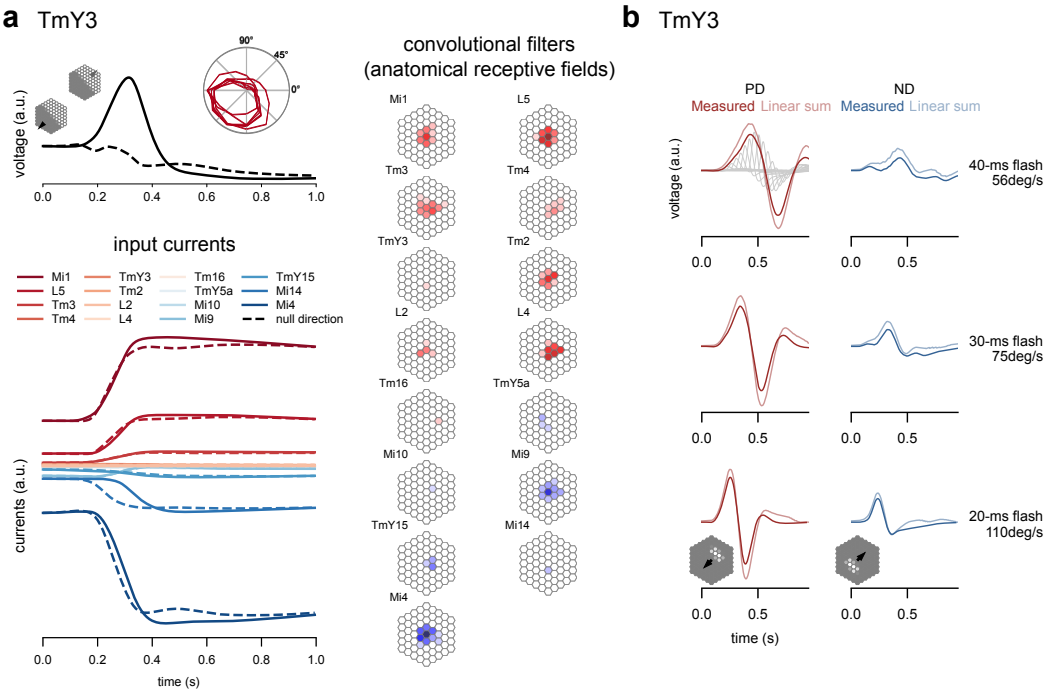

Figure 5.10: **TmY3 motion detection mechanisms hypothesized by the model. a:** Responses to PD and ND ON-edge motion and contributions from input elements as in Fig. 5.8. **b:** PD enhancement and ND suppression in the model. Same as Fig. 5.8b for T5c.

Figure 5.11: **Statistics of learned parameters of best 20% models vs. worst 20% models. a:** Task-optimized resting potentials. **b:** Task-optimized time constants. **c:** Task-optimized filter scaling factors.

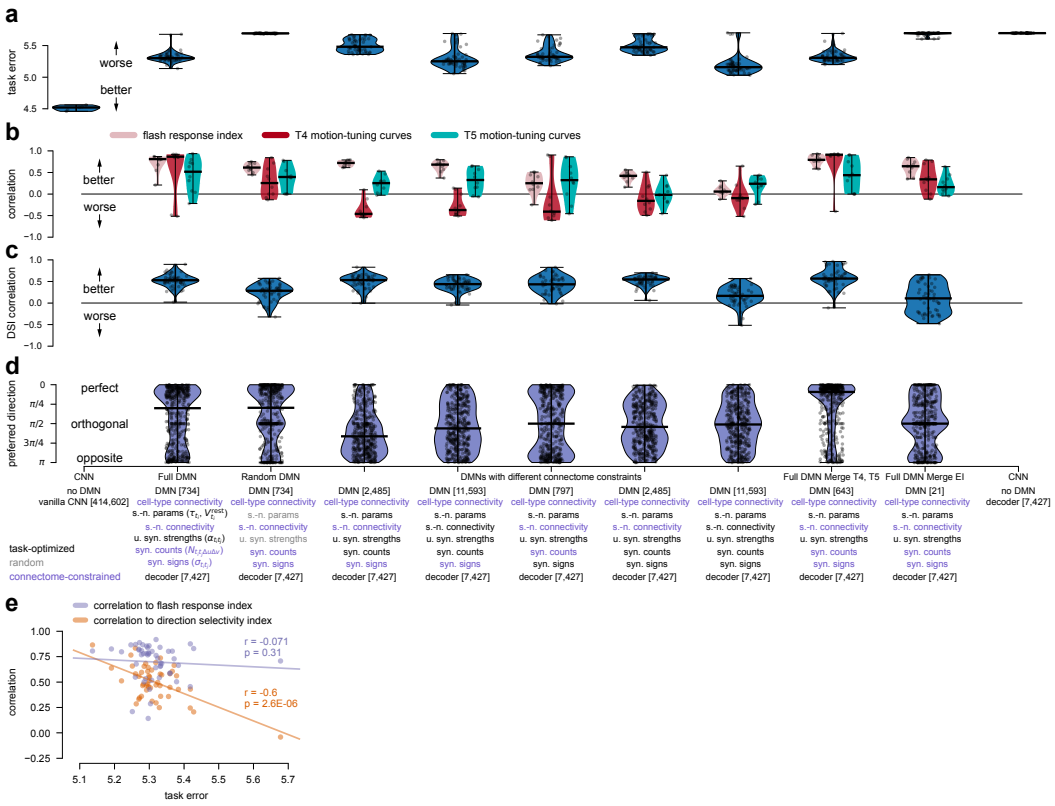

Figure 5.12: **DMN benchmark of connectomic constraints. a-d: How would incomplete knowledge of connectome affect the tuning predictions?** We artificially varied DMNs with random parameters, connectome-constrained or task-optimized parameters. Five experiments: Four 'Synapse-optimized models', one 'Fully optimized'. Details in Methods. **How would incomplete knowledge of cell types affect the tuning predictions?** We artificially assumed some cell types to be indistinguishable, with shared physiological parameters (resting potentials, time constants, and unitary synapse strengths). Two experiments: (1) 'Full DMN Merge T4, T5' assumes that 'T4' and 'T5' subtypes were indistinguishable, reducing the number of cell types to 58. (2) 'Full DMN Merge E/I' assumes that we had three cell types, 'excitatory' (37 cell types), 'inhibitory' (22 cell types) or 'both' (4 cell types), based on our knowledge of synapse signs. Tuning predictions are shown in comparison to the Full DMN and the DMN with random parameters. **a:** Task error. **b:** Predicted correlations to flash response indices, T4-, and T5 motion-tuning curves (10 best models). **c:** Predicted correlations to known direction selectivity indices. **d:** Distances between known preferred directions and predicted preferred directions for T4 and T5 neurons. (Continued on the next page)

Fig. 5.12, continued: **e: Better task performing models predict motion tuning neurons better.** We correlate predicted tuning metrics from each model to the known tuning properties to understand when better performing models give us better tuning predictions. **(orange)** When correlating the direction selectivity index of each model to the binary known properties for T4 and T5 and their input cell types, we find that this correlation is higher for better performing models (Pearson correlation, $r = -0.60$, p = 2.6×10^{-6}, $t = r\sqrt{\frac{\mathrm{df}}{1-r^2}}$, 95% CI = [-1, -0.42], df = 48). **(magenta)** While the models predicted the known contrast preferences generally well, the correlation of flash response index to the binary known contrast preferences of 31 cell types did not significantly increase with better performing models.

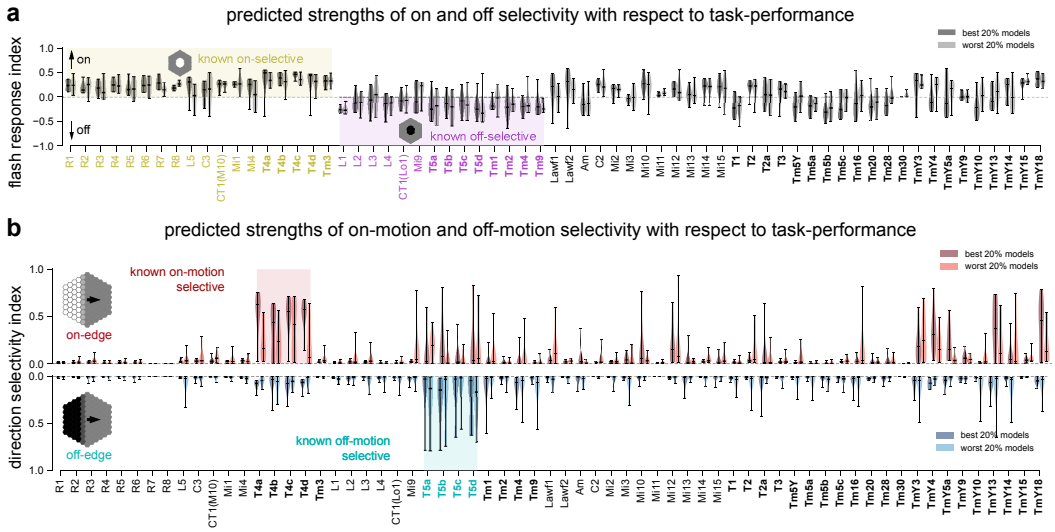

Figure 5.13: **Predicted tuning with respect to task-performance. a:** Flash response index computed as the max-abs-scaled peak response to an off flash subtracted from the max-abs-scaled peak response to an on flash – both of approximately 35° radius and presented for 1s after 2 seconds of grey input. Values above 0 indicate on-polarity, values below zero indicate off-polarity. Known on-polar and off-polar cell types are colored in yellow and magenta. **b:** Single cell type direction selectivity of best 20% task-performing models versus worst 20% task-performing models of an ensemble of 50 models as a result of peak voltage responses in central columns to on-edges and off-edges moving towards all possible directions on grey background (Equation 9). The bolded cell types are those which optic flow is decoded from.

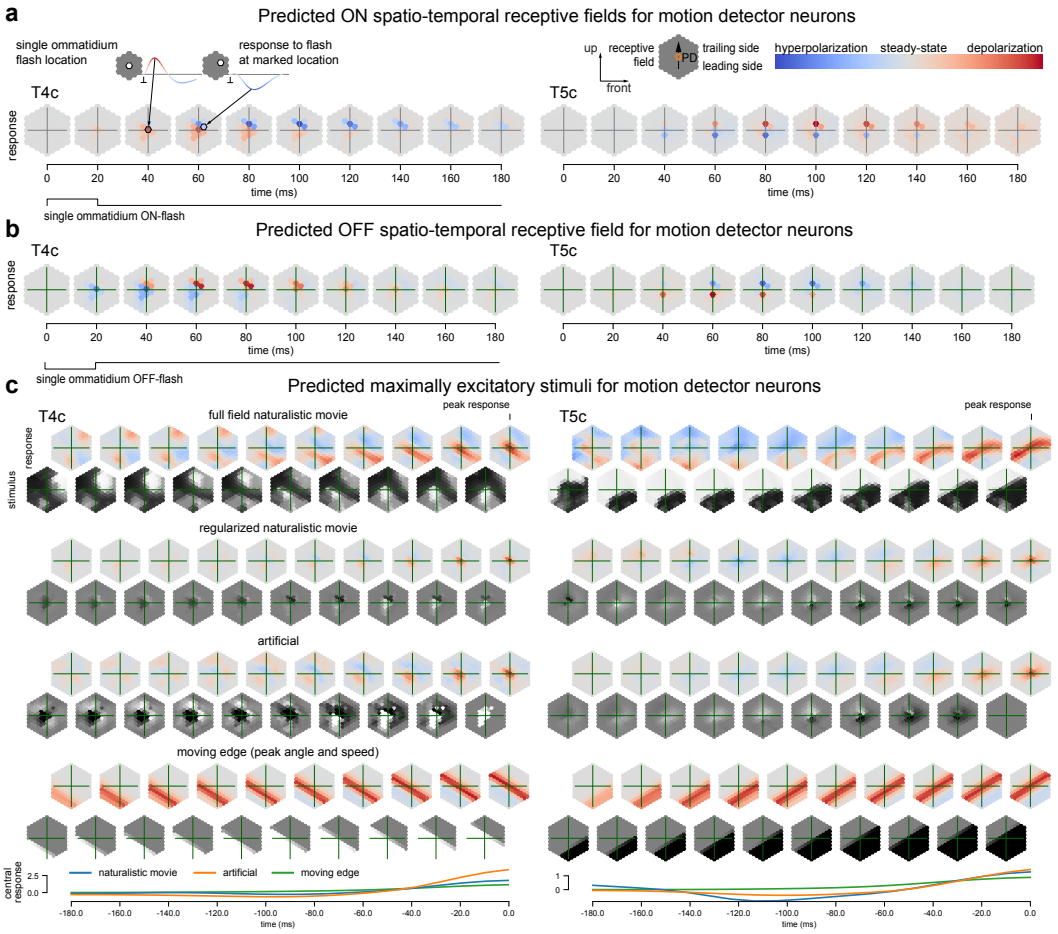

Figure 5.14: **Spatio-temporal receptive fields mapped with ON- and OFF-impulses and maximally excitatory stimuli. a:** Spatiotemporal receptive fields for motion detector neurons agree with experimental measurements (Gruntman et al. 2018). **b:** Spatio-temporal receptive field mapping with single ommatidium OFF-impulses. **c:** Maximally excitatory stimuli and baseline-subtracted responses. Including full-field naturalistic, regularized naturalistic, artificial, and moving edge stimuli and responses. Moving edge angle and speed maximize the central cell peak response. Artificial stimuli are optimized from initial noise to maximize the central cell activity using gradient ascent plus full-field regularization towards grey. The last row shows the baseline-subtracted central cell responses. Peak central cell responses at time point zero.

References

[1] Larry F Abbott et al. “The mind of a mouse”. In: *Cell* 182.6 (2020), pp. 1372–1376.

[2] Mohammed AlQuraishi and Peter K Sorger. “Differentiable biology: using deep learning for biophysics-based and data-driven modeling of molecular mechanisms”. In: *Nature methods* 18.10 (2021), pp. 1169–1180.

[3] Georg Ammer et al. “Functional specialization of neural input elements to the Drosophila ON motion detector”. In: *Current Biology* 25.17 (2015), pp. 2247–2253.

[4] Alexander Arenz et al. “The Temporal Tuning of the Drosophila Motion Detectors Is Determined by the Dynamics of Their Input Elements”. In: *Current Biology* 27.7 (2017), pp. 929–944. ISSN: 09609822. DOI: 10.1016/j.cub.2017.01.051.

[5] Cornelia I Bargmann and Eve Marder. “From the connectome to brain function”. In: *Nature methods* 10.6 (2013), pp. 483–490.

[6] H B Barlow and W R Levick. “The mechanism of directionally selective units in rabbit’s retina.” In: *The Journal of Physiology* 178.3 (June 1965), pp. 477–504.

[7] Etienne Becht et al. “Dimensionality reduction for visualizing single-cell data using UMAP”. In: *Nature biotechnology* 37.1 (2019), pp. 38–44.

[8] Rudy Behnia et al. “Processing properties of ON and OFF pathways for Drosophila motion detection”. In: *Nature* 512.7515 (July 2014), pp. 427–430.

[9] Tirthabir Biswas and James E Fitzgerald. “Geometric framework to predict structure from function in neural networks”. In: *Physical Review Research* 4.2 (2022), p. 023255.

[10] Alexander Borst, Jürgen Haag, and Alex S Mauss. “How fly neurons compute the direction of visual motion”. In: *Journal of Comparative Physiology A* 206.2 (2020), pp. 109–124.

[11] Alexander Borst and Moritz Helmstaedter. “Common circuit design in fly and mammalian motion vision”. In: *Nature Neuroscience* 18.8 (June 2015), pp. 1067–1076. DOI: 10.1038/nn.4050.

[12] V. Braitenberg. “Patterns of projection in the visual system of the fly. I. Retina-lamina projections”. In: *Experimental Brain Research* 3.3 (1967), pp. 271–298.

[13] Daniel J Butler et al. “A naturalistic open source movie for optical flow evaluation”. In: *European conference on computer vision*. Springer. 2012, pp. 611–625.

[14] Daniel J Butler et al. "A Naturalistic Open Source Movie for Optical Flow Evaluation (Sintel)". In: *Eccv* (2012), pp. 611–625. DOI: 10.1007/978-3-642-33783-3_44.

[15] Ricky T. Q. Chen et al. "Neural Ordinary Differential Equations". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018.

[16] Benjamin R Cowley et al. "One-to-one mapping between deep network units and real neurons uncovers a visual population code for social behavior". In: *bioRxiv* (2022).

[17] Timothy A Currier, Michelle M Pang, and Thomas R Clandinin. "Visual processing in the fly, from photoreceptors to behavior". en. In: *Genetics* 224.2 (May 2023).

[18] Fred P Davis et al. "A genetic, genomic, and computational resource for exploring neural circuit function". In: *bioRxiv* (Aug. 2018), p. 385476.

[19] Sven Dorkenwald et al. "Neuronal wiring diagram of an adult brain". en. In: *bioRxiv* (July 2023).

[20] Alexey Dosovitskiy et al. "Flownet: Learning optical flow with convolutional networks". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2758–2766.

[21] K. F. Fischbach and A. P.M. Dittrich. "The optic lobe of Drosophila melanogaster. I. A Golgi analysis of wild-type structure". In: *Cell and Tissue Research* 258.3 (1989), pp. 441–475. ISSN: 14320878. DOI: 10.1007/BF00218858.

[22] Yvette E. Fisher, Marion Silies, and Thomas R. Clandinin. "Orientation Selectivity Sharpens Motion Detection in Drosophila". In: *Neuron* 88.2 (2015), pp. 390–402. ISSN: 10974199. DOI: 10.1016/j.neuron.2015.09.033.

[23] Jonathan Frankle and Michael Carbin. "The lottery ticket hypothesis: Finding sparse, trainable neural networks". In: *arXiv preprint arXiv:1803.03635* (2018).

[24] Kunihiko Fukushima and Sei Miyake. "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition". In: *Competition and cooperation in neural nets*. Springer, 1982, pp. 267–285.

[25] Julijana Gjorgjieva, Haim Sompolinsky, and Markus Meister. "Benefits of pathway splitting in sensory coding". In: *Journal of Neuroscience* 34.36 (2014), pp. 12127–12144.

[26] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[27] Karl Geokg Götz. "Optomotorische untersuchung des visuellen systems einiger augenmutanten der fruchtfliege Drosophila". In: *Kybernetik* 2.2 (1964), pp. 77–92.

[28] Lukas N Groschner et al. “A biophysical account of multiplication by a single neuron”. In: *Nature* 603.7899 (2022), pp. 119–123.

[29] Eyal Gruntman, Sandro Romani, and Michael B Reiser. “The computation of directional selectivity in the Drosophila OFF motion pathway”. In: *Elife* 8 (2019), e50706.

[30] Eyal Gruntman, Sandro Romani, and Michael B. Reiser. “Simple integration of fast excitation and offset, delayed inhibition computes directional selectivity in Drosophila”. In: *Nature Neuroscience* 21.2 (2018), pp. 250–257. ISSN: 15461726. DOI: 10.1038/s41593-017-0046-4.

[31] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. “Multilayer feedforward networks are universal approximators”. In: *Neural Netw.* 2.5 (Jan. 1989), pp. 359–366.

[32] Ferris Jabr. “The Connectome Debate: Is Mapping the Mind of a Worm Worth It?” In: *Scientific American* (Oct. 2012).

[33] Maximilian Joesch et al. “ON and OFF pathways in Drosophila motion vision”. In: *Nature* 468.7321 (Nov. 2010), pp. 300–304. DOI: 10.1038/nature09545.

[34] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[35] Janne K Lappalainen et al. “Connectome-constrained networks predict neural activity across the fly visual system”. In: *Nature* 634.8036 (2024), pp. 1132–1140. URL: https://doi.org/10.1038/s41586-024-07939-3.

[36] Yann LeCun, Corinna Cortes, and CJ Burges. “MNIST handwritten digit database”. In: *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2 (2010).

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

[38] Tony X. Liu et al. “Connectomic features underlying diverse synaptic connection strengths and subcellular computation”. In: *Current Biology* 32.3 (2022), 559–569.e5. ISSN: 0960-9822. DOI: https://doi.org/10.1016/j.cub.2021.11.056.

[39] Matthew S. Maisak et al. “A directional tuning map of Drosophila elementary motion detectors”. In: *Nature* 500.7461 (2013), pp. 212–216. ISSN: 00280836. DOI: 10.1038/nature12320.

[40] Omer Mano et al. “Predicting individual neuron responses with anatomically constrained task optimization”. In: *Current Biology* 31.18 (2021), pp. 4062–4075.

[41] Manuel Beiran and Ashok Litwin-Kumar. “Prediction of neural activity in connectome-constrained recurrent networks”. In: *bioRxiv* (Jan. 2024), p. 2024.02.22.581667.

[42] Eve Marder. “Neuromodulation of neuronal circuits: back to the future”. In: *Neuron* 76.1 (2012), pp. 1–11.

[43] Eve Marder and Adam L Taylor. “Multiple models to capture the variability in biological neurons and networks”. In: *Nature Neuroscience* 14.2 (2011), pp. 133–138.

[44] Arie Matsliah et al. “Neuronal “parts list” and wiring diagram for a visual system”. en. In: *bioRxiv* (Dec. 2023).

[45] Mark Mazurek, Marisa Kager, and Stephen D. Van Hooser. “Robust quantification of orientation selectivity and direction selectivity”. In: *Frontiers in Neural Circuits* 8 (Aug. 2014).

[46] Matthias Meier and Alexander Borst. “Extreme Compartmentalization in a Drosophila Amacrine Cell”. In: *Current Biology* 29.9 (2019), 1545–1550.e2. ISSN: 09609822. DOI: 10.1016/j.cub.2019.03.070.

[47] Lu Mi et al. “Connectome-constrained Latent Variable Model of Whole-Brain Neural Activity”. In: *International Conference on Learning Representations*. 2022.

[48] Kenneth D Miller and Francesco Fumarola. “Mathematical equivalence of two common forms of firing rate models of neural networks”. In: *Neural computation* 24.1 (2012), pp. 25–31.

[49] Aljoscha Nern, Barret D Pfeiffer, and Gerald M Rubin. “Optimized tools for multicolor stochastic labeling reveal diverse stereotyped cell arrangements in the fly visual system.” In: *Proceedings of the National Academy of Sciences of the United States of America* 112.22 (May 2015), E2967–76. ISSN: 0027-8424. DOI: 10.1073/pnas.1506763112.

[50] Aljoscha Nern et al. “Connectome-driven neural inventory of a complete visual system”. en. In: *bioRxiv* (Apr. 2024), p. 2024.04.16.589741.

[51] Adam Paszke et al. “Pytorch: An imperative style, high-performance deep learning library”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[52] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[53] Maithra Raghu et al. “Do vision transformers see like convolutional neural networks?” In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 12116–12128.

[54] Giordano Ramos-Traslosheros and Marion Silies. “The physiological basis for contrast opponency in motion computation in Drosophila”. In: *Nature communications* 12.1 (2021), pp. 1–16.

[55] W Reichardt. “Autocorrelation, a principle for evaluation of sensory information by the central nervous system”. In: *Principles of sensory communications* (1961).

[56] Maximilian Riesenhuber and Tomaso Poggio. “Hierarchical models of object recognition in cortex”. In: *Nature neuroscience* 2.11 (1999), pp. 1019–1025.

[57] Victor Lobato Ríos et al. “NeuroMechFly, a neuromechanical model of adult Drosophila melanogaster”. In: *bioRxiv* (2021).

[58] Marta Rivera-Alba et al. “Wiring Economy and Volume Exclusion Determine Neuronal Placement in the Drosophila Brain”. In: *Current Biology* 21.23 (Dec. 2011), pp. 2000–2005.

[59] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. “Learning representations by back-propagating errors”. In: *Nature* 323.6088 (1986), pp. 533–536.

[60] Louis K Scheffer and Ian A Meinertzhagen. “A connectome is not enough–what is still needed to understand the brain of Drosophila?” In: *Journal of Experimental Biology* 224.21 (2021), jeb242740.

[61] Philipp Schlegel et al. “A consensus cell type atlas from multiple connectomes reveals principles of circuit stereotypy and variation”. en. In: *bioRxiv* (June 2023), p. 2023.06.27.546055.

[62] Xingjian Shi et al. “Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting”. In: *Advances in Neural Information Processing Systems*. Ed. by C. Cortes et al. Vol. 28. Curran Associates, Inc., 2015.

[63] Kazunori Shinomiya et al. “Comparisons between the ON- and OFF-edge motion pathways in the Drosophila brain.” In: *Elife* 8 (Jan. 2019), p. 2431.

[64] Kazunori Shinomiya et al. “Neuronal circuits integrating visual motion information in Drosophila melanogaster”. In: *Current Biology* 32.16 (2022), pp. 3529–3544.

[65] Philip K Shiu et al. “A leaky integrate-and-fire computational model based on the connectome of the entire adult Drosophila brain reveals insights into sensorimotor processing”. en. In: *bioRxiv* (May 2023).

[66] James A Strother et al. “Behavioral state modulates the ON visual motion pathway of Drosophila”. In: *Proceedings of the National Academy of Sciences* 115.1 (2018), E102–E111.

[67] James A Strother et al. “The emergence of directional selectivity in the visual motion pathway of Drosophila”. In: *Neuron* 94.1 (2017), pp. 168–182.

[68] James A. Strother, Aljoscha Nern, and Michael B. Reiser. "Direct observation of on and off pathways in the drosophila visual system". In: *Current Biology* 24.9 (2014), pp. 976–983. ISSN: 09609822. DOI: 10.1016/j.cub.2014.03.017.

[69] Shin-ya Takemura et al. "Synaptic circuits and their variations within different columns in the visual system of Drosophila ". In: *Proceedings of the National Academy of Sciences* 112.44 (2015), pp. 13711–13716. ISSN: 0027-8424. DOI: 10.1073/pnas.1509820112.

[70] Shin-ya Takemura et al. "The comprehensive connectome of a neural substrate for 'ON' motion detection in Drosophila". In: *eLife* 6 (2017), pp. 1–16. DOI: 10.7554/elife.24394.

[71] Fabian David Tschopp, Michael B. Reiser, and Srinivas C. Turaga. "A Connectome Based Hexagonal Lattice Convolutional Network Model of the Drosophila Visual System". In: *arXiv preprint arXiv:1806.04793* (2018). arXiv: 1806.04793.

[72] John C Tuthill et al. "Contributions of the 12 neuron classes in the fly lamina to motion vision". In: *Neuron* 79.1 (2013), pp. 128–140.

[73] John C Tuthill et al. "Wide-field feedback neurons dynamically tune early visual processing". In: *Neuron* 82.4 (2014), pp. 887–895.

[74] Roman Vaxenburg et al. "Whole-body simulation of realistic fruit fly locomotion with deep reinforcement learning". In: *bioRxiv* (Mar. 2024).

[75] Edgar Y Walker et al. "Inception loops discover what excites neurons most using deep predictive models". In: *Nature Neuroscience* 22.12 (2019), pp. 2060–2065.

[76] Andrew Warrington, Arthur Spencer, and Frank Wood. "The Virtual Patch Clamp: Imputing C. elegans Membrane Potentials from Calcium Imaging". In: *arXiv [q-bio.NC]* (July 2019). arXiv: 1907.11075 [q-bio.NC].

[77] Paul J Werbos. "Backpropagation through time: what it does and how to do it". In: *Proceedings of the IEEE* 78.10 (1990), pp. 1550–1560.

[78] Michael Winding et al. "The connectome of an insect brain". In: *Science* 379.6636 (2023), eadd9330. DOI: 10.1126/science.add9330.

[79] D. L. K. Yamins et al. "Performance-optimized hierarchical models predict neural responses in higher visual cortex". In: *Proceedings of the National Academy of Sciences* 111.23 (May 2014), pp. 8619–8624.

[80] Helen H Yang et al. "Subcellular imaging of voltage and calcium signals reveals neural processing in vivo". In: *Cell* 166.1 (2016), pp. 245–257.

[81] Zhihao Zheng et al. "A complete electron microscopy volume of the brain of adult Drosophila melanogaster". In: *Cell* 174.3 (2018), pp. 730–743.