Inductive biases of neural networks for generalization in spatial navigation

Ruiyi Zhang[1], Xaq Pitkow[3,4,5,6], and Dora E Angelaki[1,2,6]

[1]Department of Mechanical and Aerospace Engineering, New York University
[2]Center for Neural Science, New York University
[3]Department of Neuroscience, Baylor College of Medicine
[4]Department of Electrical and Computer Engineering, Rice University
[5]Center for Neuroscience and Artificial Intelligence, Baylor College of Medicine
[6]Contributed equally as co-senior authors

Abstract

Artificial reinforcement learning agents that perform well in training tasks typically perform worse than animals in novel tasks. We propose one reason: generalization requires modular architectures like the brain. We trained deep reinforcement learning agents using neural architectures with various degrees of modularity in a partially observable navigation task. We found that highly modular architectures that largely separate computations of internal belief of state from action and value allow better generalization performance than agents with less modular architectures. Furthermore, the modular agent's internal belief is formed by combining prediction and observation, weighted by their relative uncertainty, suggesting that networks learn a Kalman filter-like belief update rule. Therefore, smaller uncertainties in observation than in prediction lead to better generalization to tasks with novel observable dynamics. These results exemplify the rationale of the brain's inductive biases and show how insights from neuroscience can inspire the development of artificial systems with better generalization.

Introduction

Generalization out-of-distribution requires one to act upon the correct prior knowledge of the world distilled from experience. However, David Hume's famous "problem of induction" indicates that one's prior knowledge can be objectively wrong [1]. Animals evolved to master their daily tasks, thus can correctly learn the underlying structures of these tasks as prior knowledge, and then use it to generalize to other out-of-distribution tasks with a similar structure [2]. In contrast, artificial intelligence (AI) models with inappropriate designs generalize poorly, possibly due to the failure to correctly learn the task structure as prior knowledge, thus, failing to leverage this knowledge for generalization [3, 4].

Natural tasks have infinitely many solutions that are equally good at the training task. One solution might master the task by capturing its structure, while another might solve it by memorizing all encountered input-output pairs. Any learning system for the training task has a bias that prioritizes one solution over other possible solutions, known as the inductive bias [3, 4, 5]. For a neural network, its architecture is one important aspect of this bias. To prioritize a network solution that generalizes well, the inductive bias must specialize in matching the specific task structure of interest: a universal inductive bias appropriate for all tasks does not exist ("no free lunch theorem" [6]). To generalize well in many daily tasks with a single large network, the brain may have evolved a *modular* architecture to flexibly enable distinct circuits with different functionally specialized modules, depending on the current task demands [7, 8, 9]. Inspired by this,

we hypothesize that task-appropriate modular architectures empower neural networks to capture the task structure during training, so that networks can use this prior knowledge to generalize to novel tasks with a similar structure. Less modular neural architectures lead to inferior generalization abilities.

We tested this hypothesis using deep reinforcement learning (RL) [10] agents in a naturalistic virtual navigation task. Agents must steer to the location of a transiently visible target (a "firefly") using optic flow cues. We previously designed this task to explore neural computations underlying macaques' flexible naturalistic behaviors [11, 12]. This task benefits from the simultaneous mental computation of multiple variables: the subject's internal state representation of the outside world (belief) given partial and noisy sensory cues; the motor commands (action) controlling a joystick to navigate toward targets; and the value of actions in each state [13]. Neural networks for this task can use a spectrum of architectures varying in functional modularity, which we define as the degree to which separate neural units compute distinct task variables. At one extreme, they can have a holistic architecture (lowest modularity), where all neurons are interconnected and must compute variables jointly; at the other extreme, they can have a modular architecture (highest modularity), where neurons are segregated in distinct modules, each dedicated to distinct variables. We trained RL agents using neural networks endowed with various degrees of modularity in this navigation task. After training, we tested them in two novel tasks sharing a similar task structure to the training task: one which manipulates the sensorimotor mapping from joystick movements to subjects' movements in the environment, and the other which randomly applies passive perturbations to subjects' movements.

We found that highly modular neural architectures are more appropriate for the navigation task: agents with these architectures can learn internal state representations that are still accurate in novel tasks to support generalization. Such network ability resembles macaque behaviors: after training, macaques showed flexibility in performing novel tasks with a similar structure without additional training [14]. In contrast, agents with less modular architectures demonstrated inferior generalization abilities.

We further found that a belief update rule akin to recursive Bayesian estimation [15] emerges in the modular agent's network after training: a posterior belief is a weighted average of a prior prediction using a motor efference copy and a likelihood over states given visual observation, where a higher weight is assigned to the more reliable source, similar to how the brain performs probabilistic inference [16]. Training agents with greater uncertainty in observation than prediction biases agents toward ignoring observations in belief formation, which then cripples their generalization abilities in novel tasks that require the use of novel observations to construct the belief, instead of the outdated internal model. Therefore, to generalize, agents must develop prior knowledge of relying more on observation, similar to macaques and humans in this task [17].

Together, these findings shed light on how to bridge the gap between AI and neuroscience [3, 4, 18, 19]: from AI to neuroscience, our work validates the rationale of the brain's inductive biases in spatial navigation [20, 17]. From neuroscience to AI, our work exemplifies how insights from the brain can inspire the development of AI with better generalization abilities.

Results

RL agents trained to navigate using partial and noisy sensory cues

To study naturalistic, continuous-time computations involved in foraging behaviors, we previously designed a virtual reality navigation task where macaques navigate to targets ("fireflies") using sparse and transient visual cues [11]. At the beginning of each trial, the subject is situated at the origin facing forward; a target is presented at a random location within the field of view on the ground plane and disappears after 300 ms. The subject can freely control its linear and angular velocities with a joystick to move in the virtual environment (Fig. 1**a**). The objective is to navigate toward the memorized target location, then stop inside the reward zone—a circular region centered at the target location with a radius of 65 cm. A reward is given only if the subject stops inside the reward zone. The subject's self-location is not directly observable because there are no stable landmarks; instead, the subject needs to use optic flow cues on the ground plane to

perceive self-motion and perform path integration. Note that optic flow cues cannot serve as stable landmarks because ground plane textural elements—small triangles—appear at random locations and orientations, and disappear after only a short lifetime (~ 250 ms). A new trial starts after the subject stops moving or the trial exceeds the maximum trial duration. Details of this task are described in [11]. Macaques can be trained to master this task, and all macaque data presented in this paper were adapted from previously published works [11, 12, 14, 17, 20, 21].

RL is a reasonable framework for modeling behavior in this task because, like animals, RL agents can learn this task through sparse reward signals. We formulate this task as a Partially Observable Markov Decision Process (POMDP) [22] in discrete time, with continuous state and action spaces (Fig. 1**b**). At each time step t, the environment is in the state $\boldsymbol{s}_t$ (including the agent's position and velocity, and the target's position). The agent takes an action $\boldsymbol{a}_t$ (controlling its linear and angular velocities) to update $\boldsymbol{s}_t$ to the next state $\boldsymbol{s}_{t+1}$ following the environmental dynamics given by the transition probability $T(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$, and receives a reward r_t from the environment following the reward function $R(\boldsymbol{s}_t, \boldsymbol{a}_t)$ (a positive scalar if the agent stops inside the reward zone). The state $\boldsymbol{s}_t$ is not fully observable, so to support decision-making, the agent needs to maintain an internal state representation that synthesizes the history of evidence. For example, a recursive Bayesian estimation (Methods) can form a posterior density $p(\boldsymbol{s}_t|\boldsymbol{o}_{0:t}, \boldsymbol{a}_{0:t-1})$ over the current world state $\boldsymbol{s}_t$ given the history of two sources of evidence. One source is a prediction based on its internal model of the dynamics, its previous posterior, and the last self-action $\boldsymbol{a}_{t-1}$ (e.g., a motor efference copy). The other source is a partial and noisy observation $\boldsymbol{o}_t$ of the state $\boldsymbol{s}_t$ drawn from the observation probability $O(\boldsymbol{o}_t|\boldsymbol{s}_t)$. For our task, this observation includes the target location when it is visible, and information about the agent's movement estimated via optic flow. We call the resultant posterior the agent's "belief" b_t. We will show later that agents using neural networks trained in this task approximately encode this belief b_t in their neural activity $\boldsymbol{h}_t$, and its dynamics are consistent with approximate recursive Bayesian estimation.

Here we use an *actor-critic* approach to learning [10] (Fig. 1**b**), where actor and critic are implemented using distinct neural networks, and each network individually encodes b_t in its neural activity (Methods). At each t, the actor computes the belief b_t of the state $\boldsymbol{s}_t$ using two sources of evidence $\boldsymbol{o}_t$ and $\boldsymbol{a}_{t-1}$, and generates an action $\boldsymbol{a}_t$. It is trained to generate the action that maximizes the value Q_t—the expected discounted cumulative rewards from t until the trial's last step, given taking $\boldsymbol{a}_t$ in $\boldsymbol{s}_t$. Since the ground truth value is unknown, the critic computes the belief b_t of $\boldsymbol{s}_t$ and estimates the value Q_t, learned through the reward prediction error after receiving the reward r_t (Fig. 1**b**). The actor is updated more slowly than the critic so that the actor can always learn something new from the critic (Methods, [23]). Over time, as the state evolves in the outside world following the environmental dynamics, the belief evolves in parallel following the internal belief update rule (learned through training). Sequential observations are drawn given the outside $\boldsymbol{s}_t$; sequential actions are taken based on the internal b_t (Fig. 1**c**). Full details of this formulation are shown in Methods.

Actor and critic networks can have various architectures using two types of building blocks: a recurrent neural network (RNN) that possesses memory, and a memoryless feedforward network (here, a multi-layer perceptron [MLP]). Each architecture defines the network's functional modularity for this task, i.e., the degree of separation in computing task variables. At one extreme, all neurons in a holistic actor/critic network are interconnected, and must compute the belief and the action/value jointly; at the other extreme, neurons in a modular actor/critic network are segregated into distinct modules to compute the belief and the action/value sequentially (Fig. 1**d**–**e**). We can manipulate which variables can be computed in each module (thought bubbles in Fig. 1**d**–**e**) by using two types of segregation. First, to synthesize the belief from the sequence of evidence, networks must have some memory. RNNs satisfy this requirement with their hidden state $\boldsymbol{h}_t$ evolving over time. In contrast, the computations of value and action do not need a memory when the belief is given, so feedforward MLPs can perform these. Therefore, using an architecture consisting of an RNN followed by an MLP, we segregate the temporal processing: the computation of b_t over time is constrained only to the RNN, as the MLP cannot integrate signals over time. Second, using a critic architecture comprising two connected modules and only providing the input of the current action $\boldsymbol{a}_t$ to the second module, we segregate inputs in a manner that limits the computation of value Q_t to only the second module: Q_t is a function of $\boldsymbol{a}_t$, but the first module does not have access to $\boldsymbol{a}_t$ and thus cannot compute Q_t (Fig. 1**e**, right).

Agents with four combinations of actors and critics in Fig. 1**d**–**e** were successfully trained in the navigation

task (Fig. 1**f**). We refer to the agent whose actor and critic are both holistic or both modular as the holistic or modular agent. Fully trained agents' behavior was compared with that of two monkeys (Fig. 1**g**) for a representative set of targets uniformly sampled on the ground plane (distance: 100–400 cm, angle: $-35°$ to $+35°$; modular agent: Fig. 1**h**, holistic agent: Fig. S1**a**). In the next section we will contrast the properties of the holistic and modular agents' trajectories, but first, we focus on their stop locations (linear: $\tilde{r}$, angular: $\tilde{\theta}$) versus the target location (linear: r, angular: θ; Fig. 1**g**, inset). The tight correspondence between stop and target locations indicates that, similar to monkeys, all agents had mastered the training task (Fig. 1**i**; Pearson's r: Fig. S1**b**). When stop locations were regressed against target locations (without intercept), we found that, similar to monkeys, agents also systematically undershot targets (Fig. S1**c**: the regression slope is smaller than 1). This finding can be predicted based on RL framework: although the immediate reward for stopping at any location within the reward zone is the same, those considering long-term values discounted over time should prefer closer reward locations to save time.

We used a Receiver Operating Characteristic (ROC) analysis [11, 12] to systematically quantify and compare behavioral performance. A psychometric curve for stopping accuracy is constructed from a large representative dataset by counting the fraction of rewarded trials as a function of a hypothetical reward boundary size (radius 65 cm is the true size; infinitely small/large reward boundary leads to no/all rewarded trials). A shuffled curve is constructed similarly after shuffling targets across trials (Fig. 1**j**). Then, an ROC curve is obtained by plotting the psychometric curve against the shuffled curve (Fig. 1**k**). An ROC curve with a slope of 1 denotes a chance level (true=shuffled) with the area under the curve (AUC) equal to 0.5. High AUC values indicate that all agents reached good performance after training (Fig. 1**k**, inset). This performance can be explained by accurate task variables encoded in their neural networks (actor: Fig. 1**l**, critic: Fig. S1**d**; see Methods), as previously also shown in the macaque brain [12, 20, 21]. In summary, like macaques, all agents with different neural architectures can be trained to master our navigation task (Fig. 1**k**).

To test the generalization abilities of these agents after training, all parameters in their neural networks are frozen to prevent further learning, and we challenge them in the following novel tasks.

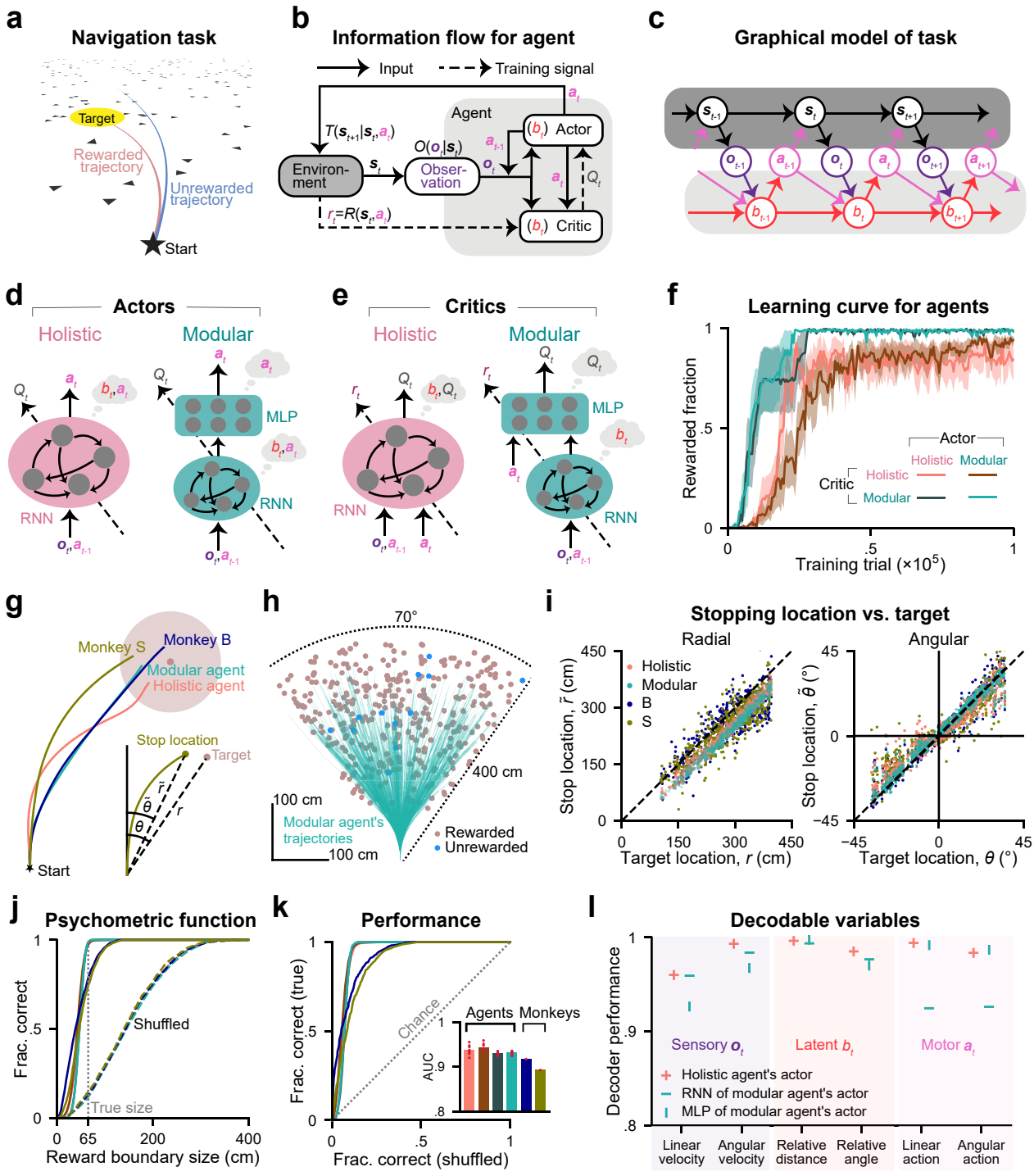

Figure 1: RL agents with different neural architectures were trained in a partially observable navigation task. **a**. Schematic of the navigation task from the subject's perspective. Optic flow cues are generated by the motion of many randomly positioned and oriented triangle elements on the ground. These cues cannot serve as landmarks since each one exists for only a short time. A trial is rewarded only if the subject stops in the reward zone, and is otherwise unrewarded. **b**. Block diagram showing the interaction between an RL agent and the task environment. At each time step t, a partial and noisy observation $\boldsymbol{o}_t$ of the state $\boldsymbol{s}_t$ drawn from the distribution O, and the last action $\boldsymbol{a}_{t-1}$ are provided to the actor and critic to form the belief b_t that is

encoded in their network states $\boldsymbol{h}_t$. The actor outputs an action $\boldsymbol{a}_t$ based on b_t, and this action updates $\boldsymbol{s}_t$ to $\boldsymbol{s}_{t+1}$; the critic estimates the value Q_t of this action based on b_t. The actor is trained to generate the action that maximizes Q_t; the critic updates its synaptic weights using the reward prediction error after receiving reward r_t from the environment. **c**. Graphical model of the task. In the environment (dark gray), $\boldsymbol{s}_t$ changes over time following the transition probability. Internally (light gray), b_t evolves following the belief update rule using observations $\boldsymbol{o}_t$ and motor efference copies $\boldsymbol{a}_{t-1}$. Observations $\boldsymbol{o}_t$ are drawn given $\boldsymbol{s}_t$. Actions $\boldsymbol{a}_t$ are taken given b_t. **d**. Schematic of actors with a holistic (left) or modular (right) architecture. A holistic actor uses an RNN to compute b_t and $\boldsymbol{a}_t$ jointly; a modular actor consisting of an RNN and an MLP constrains the computation of b_t to the RNN, and leaves the MLP only to compute $\boldsymbol{a}_t$. Thought bubbles denote the variables computed in each module. **e**. Similar to **d**, but for critic networks computing b_t and Q_t. **f**. Fraction of rewarded trials during the course of training. Performance is measured in a validation set (300 trials) for agents with various architectures. Shaded regions denote ± 1 SEM across training runs with $n = 8$ random seeds. **g**. An example trial showing monkeys and agents navigating toward the same target from the start location. Shaded circle: reward zone. Inset compares the target location (radial: r, angular: θ) vs. the stop location of monkey S (radial: $\tilde{r}$, angular: $\tilde{\theta}$). **h**. Overhead view of the spatial distribution of 500 representative targets and an example modular agent's trajectories while navigating toward these targets. **i**. Comparison of agents/monkeys' stop locations for the target locations from **h**. Black dashed lines have a slope of 1. **j**. Fraction of correct trials in a test set (2000 trials) as a function of hypothetical reward boundary size. Solid lines denote true data; dashed lines denote shuffled data. The gray dotted line denotes the true reward boundary size. **k**. True data versus shuffled data in **j** (ROC curve). Inset shows the area under the ROC curve (AUC). **j**–**k**. Agents' data are averaged across $n = 8$ training runs. **l**. Performance of linear decoders trained to decode task variables from example neural modules. Performance is quantified by computing the Pearson correlation coefficient between true and decoded values of variables.

Gain task: generalization to novel sensorimotor mappings

An important sensorimotor mapping is the joystick gain. This task variable linearly maps linear and angular motor actions on the joystick (dimensionless, bounded in $[-1, 1]$) to linear and angular velocities in the environment. During training, the joystick gain is fixed (linear: 200 cm/s, angular: 90°/s). We refer to this condition as $1\times$ gain. By increasing the gain to values that were not previously experienced, we create a "gain task" manipulation. Monkeys and agents (see Methods, agent selection) were tested with novel gains in the range $[1\times, 2\times]$ and $[1\times, 4\times]$, respectively (Fig. 2**a**).

Since novel gains used in the gain task are larger than the training gain, agents would overshoot if they blindly used the same sequence of actions as that used in the training task (no-generalization hypothesis: Fig. 2**b**, dashed lines; see Methods). Instead, agents exhibited adaptive behaviors and could generalize to various extents (Fig. 2**b**, solid lines). To systematically quantify the behavioral accuracy and also consider the effect of over-/under-shooting, we defined a radial error, whose absolute value is the Euclidean distance between one's stop location and the target location in each trial, and whose positive/negative sign denotes over-/under-shooting (using idealized trajectories, see Methods). Across gain task trials, agents, like macaques (Fig. 2**c**, Fig. S2**a**–**b**), produced radial errors that are much smaller than those of no-generalization trajectories (Fig. S2**a**), indicating that they generalized in the novel gain task. ROC analyses also confirmed this result (Fig. 2**e**). Across agents, the modular agent generalized much better than other agents (Fig. 2**d**–**f**).

When we trained agents (in the training task), we periodically assessed their performance on gain task trials (see Methods). We found that even though agents could master the training task fairly quickly (Fig. 2**g**, vertical bars on the x-axis), their generalization abilities emerged only when they experienced many more training trials than were necessary to achieve solid performance in the training task (Fig. 2**g**). The modular agent reached the highest generalization performance at the end of training.

Finally, we quantified the similarities in curvature and length between agents' and macaques' trajectories (see Methods) under different gain conditions to assess the similarities between their control policies in the gain task. At $1\times$ gain (trained) and $2\times$ gain (generalized), trajectories of the agents using a modular critic are more similar in curvature (Fig. S2**c**) and length (Fig. S2**e**) to monkeys' trajectories, than trajectories of the agents with a holistic critic. Trajectories of the modular agent exhibited greater consistency in curvature and

length across different gains than trajectories of other agents (Fig. S2**d**,**f**), reflecting the robustness of the modular agent's control policy.

Together, these results demonstrate that the modular agent, whose actor and critic are both modular, generalized the best in the gain task.

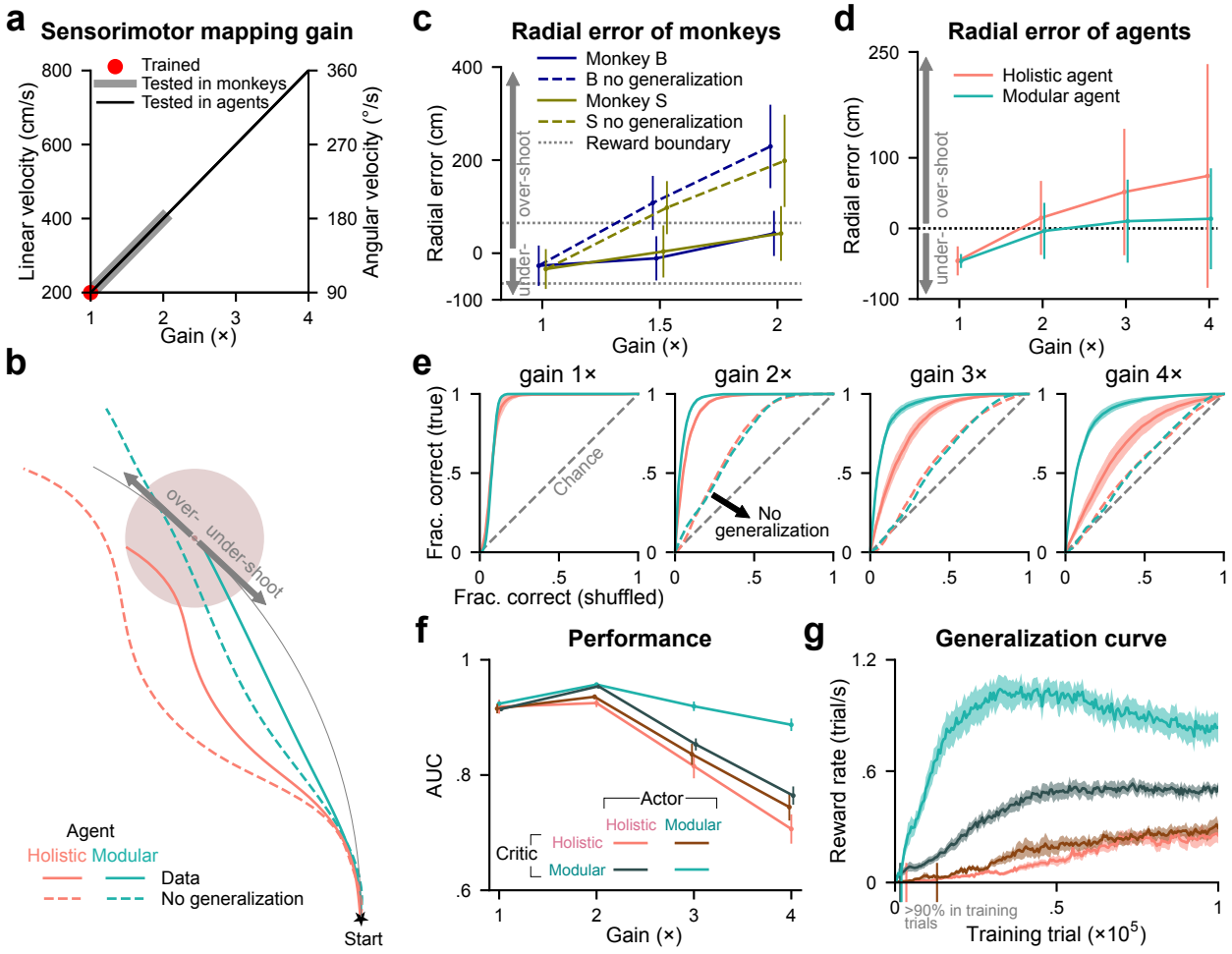

Figure 2: The modular agent exhibits the best generalization performance in the gain task. **a**. Gain is the variable that linearly maps the joystick actions (bounded in $[-1, 1]$) onto velocities in the environment. $1\times$ gain used in training has linear and angular components of 200 cm/s and 90 °/s, respectively. After training, animals and agents were tested with gain values within the ranges $[1\times, 2\times]$ and $[1\times, 4\times]$, respectively. **b**. Example trajectories of agents navigating toward a target with a novel $1.5\times$ gain. Dashed lines denote hypothetical no-generalization trajectories (see Methods). Arrows indicate regions of over- or under-shooting relative to the distance along an idealized circular trajectory connecting the start location to the target (grey line). **c**. Radial error of monkeys' stop locations in a subset of trials (1000 trials for each gain for each subject). The absolute value of a trial's radial error is defined as the distance between the subject's stop and target locations. The positive/negative sign of the radial error denotes over-/under-shooting (see Methods). Error bars denote ± 1 SD across trials. **d**. Similar to **c**, but for agents (8000 trials for each gain for each agent, concatenated from $n = 8$ random seeds with 1000 trials each). The dotted line denotes the unbiased stop locations against target locations. **e**. ROC curves quantifying the accuracy of agents' stop locations in various gain conditions. Gray dashed lines denote the chance level; dashed lines in other colors denote agents' hypothetical no-generalization hypotheses. **f**. Area under the ROC curves in **e** (AUC). **g**. Agents' reward

rate (number of rewarded trials per second) averaged over three validation sets (using the same 300 targets, gain= $2\times, 3\times, 4\times$) as a function of the number of training trials (gain= $1\times$) experienced after training phase I (see definition in Methods). Vertical bars overlaid on the x-axis denote the first time agents reach 90% accuracy in a training gain validation set (300 targets, gain= $1\times$). **e**–**g**: Lines denote means across $n = 8$ random seeds for each agent; shaded regions and error bars denote ± 1 SEM.

Accuracy of agents' internal beliefs explains generalization in the gain task

Although we have confirmed that agents with different neural architectures exhibit different generalization abilities in the gain task, the underlying reason is still unknown. Here, we open the "black box" to elucidate neural mechanisms that account for the behaviors we observed in the last section.

Agents that generalize well must have accurate internal beliefs in their actor networks to support action. When we tested agents in the gain task, we recorded their RNN activities in actors, and found that RNN neurons are sensitive to agents' locations in the environment (spatial tuning; holistic agent: Fig. 3**a**, modular agent: Fig. 3**b**; see Methods). We are particularly interested in RNN activities because the belief is computed in the recurrent module (Fig. 1**d**), and the spatial tuning implies that these neurons encode agents' belief of where they are.

To systematically quantify the accuracy of this belief, we used linear regression (with ℓ_2 regularization) to decode agents' locations from RNN activities in their actors (Fig. 3**c**; see Methods), and used the decoding error to indicate the belief accuracy, defined as the Euclidean distance between the true and the decoded self-location. Decoding errors in training gain ($1\times$) trials are small for all agents, indicating that agents' locations are linearly decodable from RNN units (Fig. 3**d**). We found that agents that cannot generalize well when faced with increased gains (Fig. 2**f**) become less accurate about where they are (Fig. 3**d**, Fig. S3**a**). In fact, agents' behavioral performance correlates with their belief accuracy (Fig. 3**e**, Fig. S3**b**; gain values are sampled from the range $[3\times, 4\times]$). This analysis suggests that inductive biases, as introduced through the choice of neural architectures, affect generalization performance by determining the accuracy of agents' internal state representation in the novel gain task after being trained in the training task. The modular agent generalizes the best because it is supported by its most accurate internal belief.

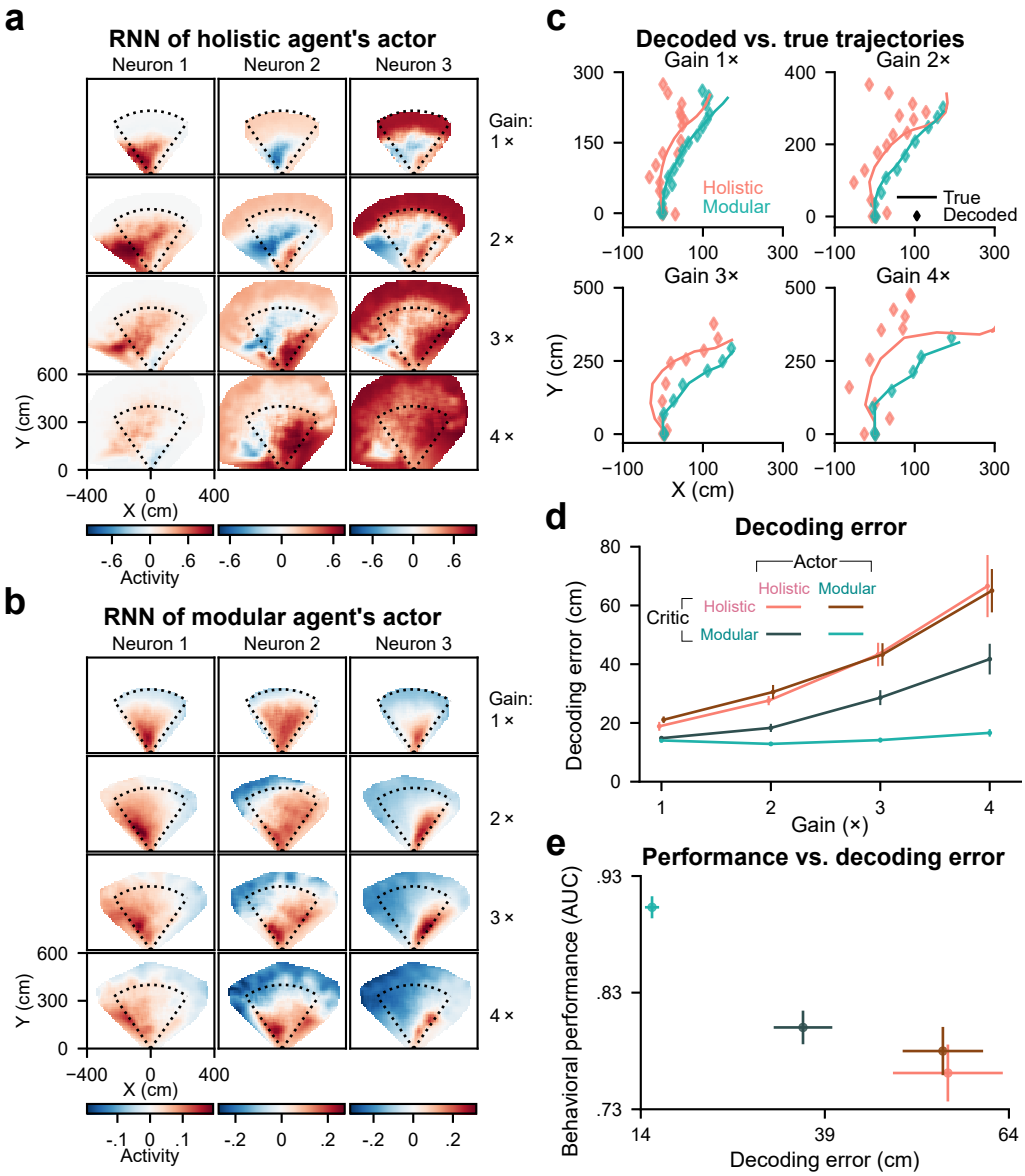

Figure 3: Decoding error of agents' internal beliefs correlates with their behavioral performance in the gain task. **a**. Spatial tuning of example RNN neurons in a holistic agent's actor. Each column denotes a neuron; each row denotes a gain value used to generate a test set (2000 trials). Dotted lines denote the boundary of a region containing all target locations. **b**. Similar to **a**, but for example RNN neurons in a modular agent's actor. **c**. Decoded belief trajectories versus agents' true trajectories during navigation to an example target under various gain conditions. Agents' belief trajectories were estimated by linear decoders trained to decode agents' locations from RNNs' neural activities (see Methods). **d**. Decoding error as a function of gain. The decoding error is defined as the distance between true and decoded locations at each time step, and is averaged across time steps and trials in the test set (2000 trials for each gain for each target). **e**. AUC versus decoding error. The test set for this panel contains 1500 trials with gains randomly sampled from the range $[3\times, 4\times]$. **d**–**e**: Error bars denote ± 1 SEM across $n = 8$ random seeds.

Perturbation task: generalization to passive motions, explained by accurate beliefs

To challenge one's generalization abilities when the latent state (its position in the virtual world) is manipulated, we use a "perturbation task" that randomly applies passive velocities (a Gaussian temporal profile for 1 second, see Methods) to monkeys' or agents' velocities to dislodge their intended trajectory. For each trial, the perturbation peak time and the peak of linear and angular passive velocities are uniformly sampled from the ranges shown in Fig. 4**a**. The inset of Fig. 4**b** shows example perturbations.

If agents simply take the same sequence of actions as in training while ignoring perturbations, their stop locations will be biased (Fig. 4**b**). Instead, like macaques (Fig. 4**c**, Fig. S4**e**), we found that agents (see Methods, agent selection) generalized to perturbations and exhibited more accurate responses (quantified using the absolute radial error between the stop and target locations) than if there was no adjustment for the perturbation (see Methods, no-generalization hypotheses; Fig. 4**d**). Agents also developed macaque-like actions that compensate for perturbations (linear: Fig. S4**a**, angular: Fig. S4**b**).

By summarizing the behavioral data across perturbation trials (AUC: Fig. 4**e**, absolute radial error: Fig. S4**c**), we conclude that agents with a modular critic generalize better than those with a holistic critic; given the same critic architecture, a modular actor is better than a holistic actor. The modular agent with modular actor and modular critic generalizes the best not only under novel gains (Fig. 2**f**), but also under passive perturbations (Fig. 4**e**). Similar to what we have observed in the gain task (Fig. 2**g**), such abilities emerged through experiencing an increased number of perturbation-free training trials (Fig. S4**d**).

To understand why agents with different architectures have different generalization abilities under perturbations, we again examined their internal beliefs encoded in actors that support action. We recorded agents' locations in the environment and their RNN activities in actors as we did in the gain task, but this time under none, small, or large perturbation conditions (Fig. 5**a**–**b**). Perturbations for the latter two conditions were drawn from the ranges shown in Fig. 4**a** (small/large: ranges for monkeys/agents). We then linearly decoded agents' locations from RNN activities (Fig. 5**c**; see Methods), and measured the decoding error to probe if the belief accuracy of where they are is affected by passive perturbations. We found that agents that generalized poorly to larger perturbations (Fig. 4**d**) exhibited less accurate internal beliefs (Fig. 5**d**, Fig. S5**a**). Similar to the result in the gain task (Fig. 3**e**, Fig. S3**b**), agents' behavioral performance also correlates with their belief accuracy in the perturbation task (Fig. 5**e**, Fig. S5**b**; showing data under the large perturbation condition). These analyses again exemplify how architectural inductive biases affect generalization: they determine the agents' abilities to maintain accurate internal beliefs in the perturbation task when trained without perturbations. The modular agent generalizes the best because it has the most accurate internal belief.

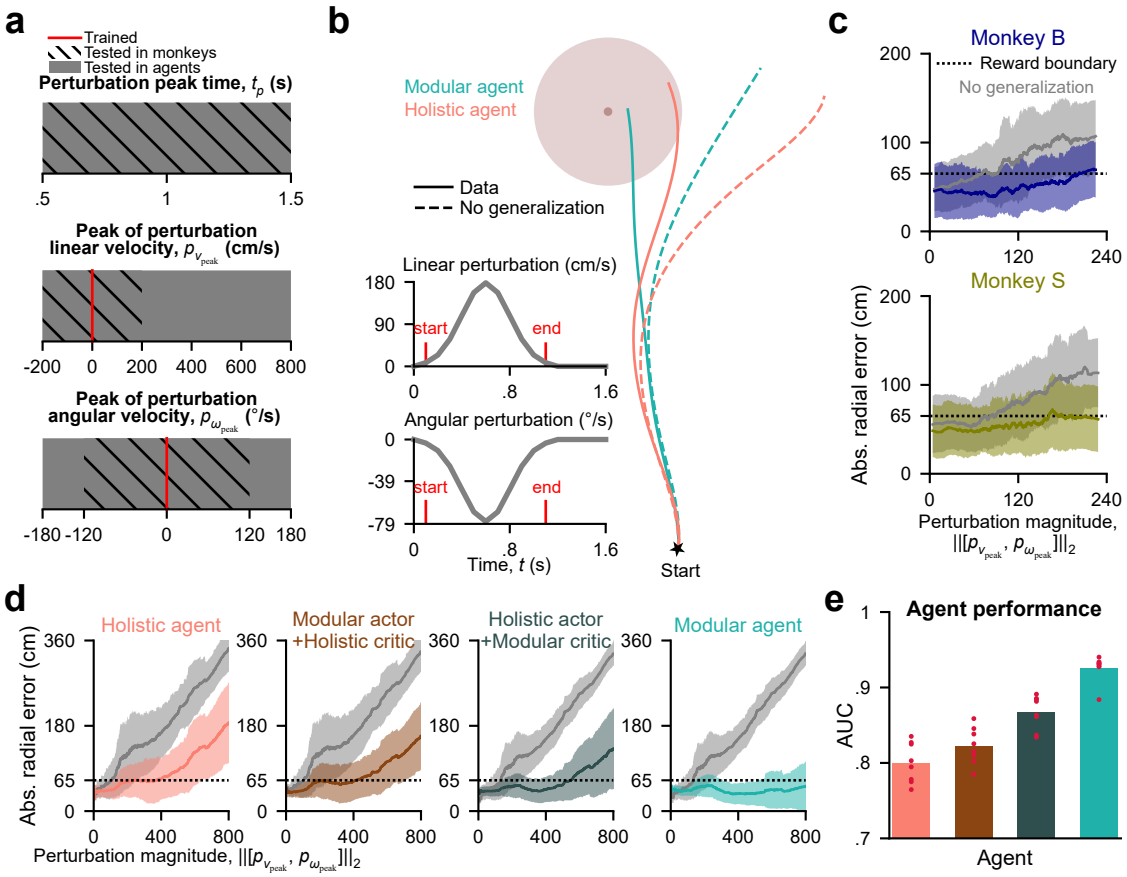

Figure 4: The modular agent exhibits the best generalization performance in the perturbation task. **a**. A perturbation trial adds Gaussian-shaped linear and angular velocities to a monkey/agent's controlled velocities at a random time within the trial for one second. For each trial, the perturbation peak time relative to the trial start (top) and the peak amplitude of linear (middle) and angular (bottom) perturbations are uniformly sampled from the corresponding ranges. No perturbations were used during training. **b**. Agents navigate in an example perturbation trial. Dashed lines denote hypothetical no-generalization trajectories (see Methods). Inset shows linear (top) and angular (bottom) perturbations in this trial. **c**. Monkeys' absolute radial error as a function of perturbation magnitude (defined as the Euclidean norm of linear and angular perturbations) in 1000 trials. Errors for hypothetical no-generalization trajectories are in gray. Solid lines and shaded regions denote means and ±1 SD obtained using a moving window (size=100 trials). The dotted black line denotes the reward boundary. **d**. Similar to **c**, but for artificial agents (the same block of 1000 targets tested with 8 random seeds for each agent, 8000 trials in total; moving window size=800 trials). **e**. AUC for agents' data in **d**. Bars denote means across $n = 8$ random seeds for each agent; red dots denote data for individual seeds.

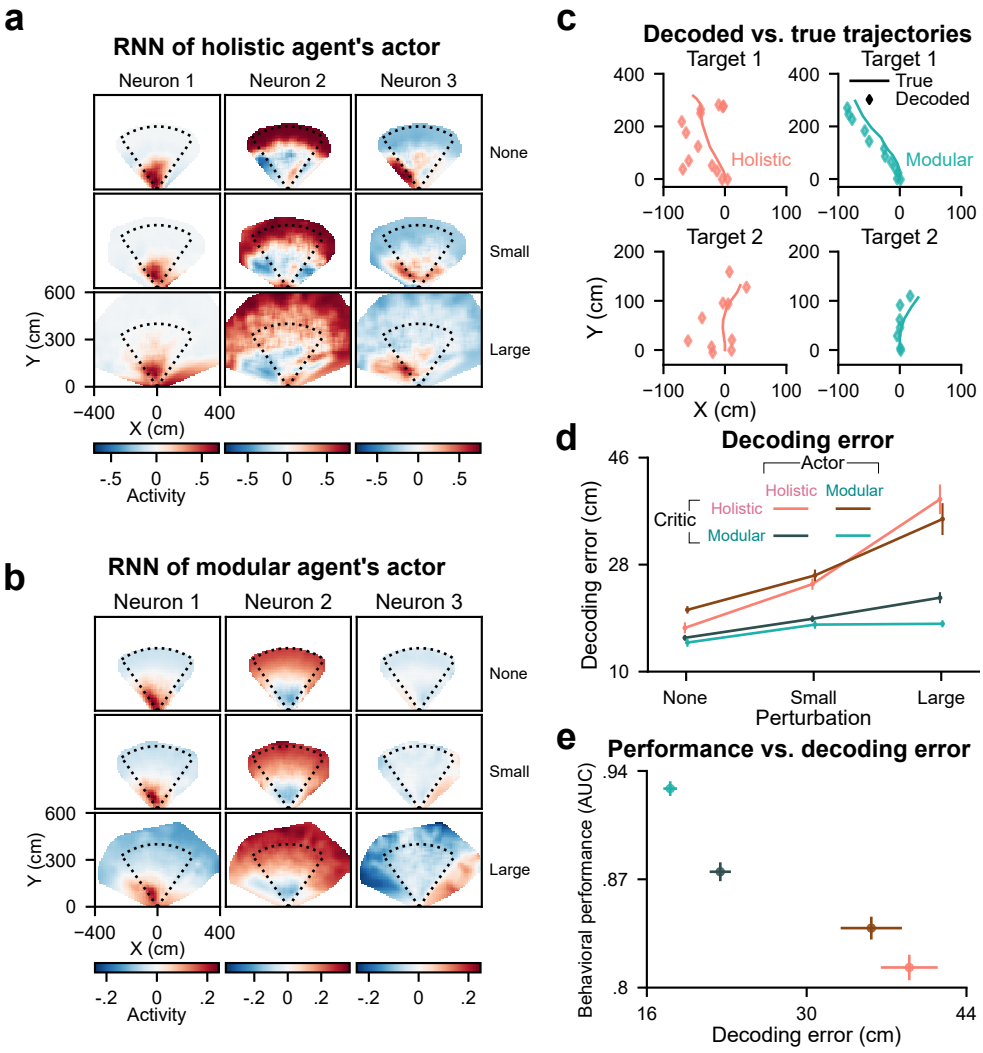

Figure 5: Decoding error of agents' internal beliefs correlates with their behavioral performance in the perturbation task. **a**–**b**. Similar to Fig. 3**a**–**b**, but for the perturbation task. Each row denotes the neural activity of example neurons under a perturbation condition for 5000 trials. Small and large perturbations are sampled from the ranges used to test monkeys/agents in Fig. 4**a**. **c**. Similar to Fig. 3**c**, but showing trajectories navigating to two example targets; each row denotes a target, and each column denotes an example agent. **d**–**e**. Similar to Fig. 3**d**–**e**, but for the perturbation task. **e** uses data under the large perturbation condition. 5000 trials were used for each agent for each perturbation condition.

Generalization of RL agents with a spectrum of neural architectures

Holistic and modular architectures (Fig. 1**d**–**e**) analyzed in previous sections can be deemed two extremes on a modularity spectrum (critic: Fig. 6**a**, actor: Fig. 6**b**). The holistic critic/actor fully mixes the computations of variables in an RNN and has the lowest modularity. The modular critic/actor separates these computations to the greatest degree and has the highest modularity. Here we consider the inductive biases of other network architectures with intermediate modularity between these two extremes with the following rationale. A critic/actor with two sequential RNNs (critic 2/actor 2) instead of one (holistic) can potentially distribute the computation of variables to two modules but without constraints. By substituting the second RNN of critic 2 with an MLP (critic 3), the belief computation requiring integration over time is constrained to the first RNN module (segregation of temporal processing). By retaining two RNNs in the critic and only providing

the action input to the second RNN (critic 4), we constrain the value computation to the second module (segregation of inputs). The modular critic (critic 5) has the greatest modularity because it has both types of segregation. The total number of trainable parameters across architectures is designed to be similar (Fig. S6**a**, see Methods).

Agents with all combinations of actors and critics in Fig. 6**a**–**b** can be fully trained to master the training task (gain$= 1\times$, no perturbations; Fig. S6**b**). We then tested their generalization abilities in the gain and perturbation tasks (gains are sampled from the range $[3\times, 4\times]$; perturbation variables are sampled from the ranges used to test agents in Fig. 4**a**; see Methods, agent selection). We found that agents need critics with higher modularity to achieve good generalization, such as critic 5, the modular critic (Fig. 6**c**–**d**). When using the modular critic, the agent's generalization could benefit from more modularity in the actor. However, with a less modular critic, the actor's higher modularity does not improve the agent's generalization much. This result has a logical interpretation: the actor is trained by the critic; without a good critic providing adequate training signals, an actor cannot learn the task structure well even with appropriate architecture. Behavioral and neural data for all agents (Fig. 6**c**–**d**) further consolidate previous conclusions that agents' behavioral performance in novel tasks is explained by their belief accuracy (Fig. 6**e**–**f**). Together, we conclude that the modularity of agents' neural architectures determines the accuracy of their internal beliefs that support generalization behavior: the more modular, the better (Fig. 6**c**–**d**).

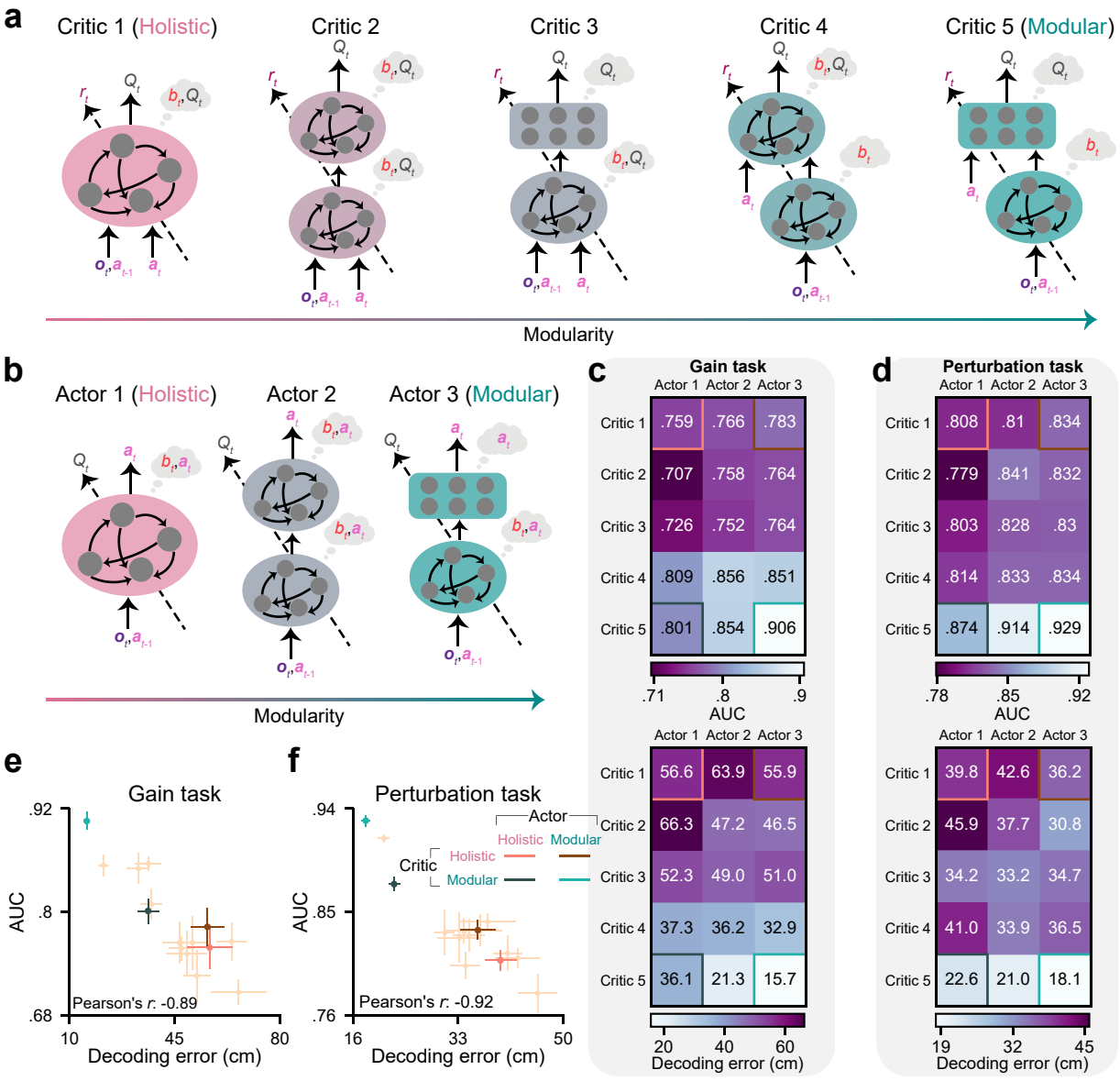

Figure 6: Increasing the modularity of neural architectures results in a more accurate internal belief in novel tasks for generalization. **a**–**b**. Architecture diagrams as in Fig. 1**e** and 1**d**, but showing a spectrum from holistic to modular for the critic (**a**) and the actor (**b**). **c**–**d**. AUC and decoding error (averaged across time steps and trials) of agents with all combinations of actors and critics in the gain (**c**) and perturbation (**d**) tasks (2000 trials for each agent for each task), averaged across $n = 8$ random seeds for each agent. Beliefs were decoded from the RNN for actors 1 and 3, or the first RNN for actor 2. Note that the four corners represent the four agents that were used for analyses in previous figures. **e**–**f**. Generalization performance measured by AUC compared to the decoding error of beliefs, as in Fig. 3**e** and 5**e**, but here including agents with all architectures.

RNNs learn a Kalman filter-like belief update rule

We have thus far demonstrated that the modular agent has the most appropriate neural architecture to construct an accurate internal belief for our navigation task. Here, we explore this agent's belief update rule using two information sources with uncertainties: observation (optic flow) and prediction (using a motor

efference copy).

The Kalman filter [15] is a practical method to implement recursive Bayesian estimation when all variables are Gaussian, and the state transitions and observations are linear (Fig. 7**a**). It constructs an internal belief of states (posterior) using motor predictions (prior) and visual cues (likelihood). A prior predicts the state using the last self-action $\boldsymbol{a}_{t-1}$ with uncertainty $\boldsymbol{\sigma}_a$, the last belief b_{t-1} with uncertainty P_{t-1}, and the state transition T. The posterior (belief b_t) is a weighted average of the prior and the likelihood over states, given the observation $\boldsymbol{o}_t$ with uncertainty $\boldsymbol{\sigma}_o$. The weight is known as the Kalman gain, which weighs more on the source with a smaller uncertainty. In combining these sources of evidence, the posterior has an uncertainty P_t smaller than only relying on a single source. Note that for our task, the Kalman gain is only affected by $\boldsymbol{\sigma}_a$ and $\boldsymbol{\sigma}_o$, and is independent of the prior uncertainty P_{t-1} in prediction (see Methods).

The RNN in an ideal modular agent may learn a belief update rule similar to the Kalman filter (more precisely, an extended Kalman filter [EKF] [24] allowing nonlinear state transitions; see Methods). We therefore compare modular agents with an EKF agent whose architecture is designed by replacing the modular actor's and modular critic's RNNs with an EKF. We trained and tested these agents in an "uncertainty task" (see Methods and below) to probe RNNs' belief update rule. Uncertainties are represented in units of the joystick gain $\boldsymbol{G}$, and an uncertainty of $\mathbf{0}$ denotes the noise-free case.

We trained three types of modular agents under three uncertainty conditions. The first type is referred to as the prior model, trained with $\boldsymbol{\sigma}_a = \mathbf{0} \ll \boldsymbol{\sigma}_o$, which learns to only rely on prediction rather than the uninformative observation; the second type is the likelihood model, trained with $\boldsymbol{\sigma}_o = \mathbf{0} \ll \boldsymbol{\sigma}_a$, which learns to only rely on its perfectly accurate observation; the last one, the posterior model, is trained with a sampled value of $\boldsymbol{\sigma}_o$ on each trial which can be larger or smaller than $\boldsymbol{\sigma}_a$, and must learn to rely on both sources. Testing these models with $\boldsymbol{\sigma}_a = 0.3\boldsymbol{G}$ and $\boldsymbol{\sigma}_o$ within the range of $[\mathbf{0}, 0.6\boldsymbol{G}]$, we found that the prior model relying on prediction exhibits errors that are independent of $\boldsymbol{\sigma}_o$; the likelihood model relying on observation exhibits errors that increase with $\boldsymbol{\sigma}_o$; and the posterior model exhibits smaller errors than either the prior or the likelihood models, similar to the EKF agent (Fig. 7**b**; see Methods, agent selection). This suggests that the modular agent's internal belief is formed by combining prediction and observation, weighted by their relative uncertainty, akin to the EKF. However, unlike the EKF that is provided with the ground truth values of uncertainties and the state transition, RNNs in the modular agent must learn to infer these from inputs $(\boldsymbol{o}_t, \boldsymbol{a}_{t-1})$ in training.

Relative reliability of information sources in training shapes generalization

Since an EKF-like belief update rule can emerge in modular agents' RNNs after training, we explored how the bias in this rule favoring one information source over the other influences generalization. In all previous sections (except the last one), agents were trained with fixed $\boldsymbol{\sigma}_a = 0.2\boldsymbol{G}, \boldsymbol{\sigma}_o = 0.1\boldsymbol{G}$ in the training task, learning to rely more on observation. Here, we trained 16 modular agents, each with a combination of $\boldsymbol{\sigma}_a$ and $\boldsymbol{\sigma}_o$ within $\{\mathbf{0}, 0.1\boldsymbol{G}, 0.2\boldsymbol{G}, 0.3\boldsymbol{G}\}$ in the training task. We then tested their generalization performance (AUC) and corresponding belief accuracy (decoding error) in the novel gain and perturbation tasks (gains are sampled from the range $[3\times, 4\times]$; perturbation variables are sampled from the ranges used to test agents in Fig. 4**a**; see Methods, agent selection). For both novel tasks, we found that agents trained with smaller observation uncertainties ($\boldsymbol{\sigma}_o < \boldsymbol{\sigma}_a$) generalized better than agents with equal uncertainties ($\boldsymbol{\sigma}_o = \boldsymbol{\sigma}_a$), and worst of all were agents trained with larger observation uncertainties ($\boldsymbol{\sigma}_o > \boldsymbol{\sigma}_a$; Fig. 7**c**–**d**). Decoding errors were larger for agents exhibiting poorer performance (Fig. S7**a**–**b**), evidence that poor generalization is associated with inaccurate internal beliefs (Fig. S7**c**–**d**).

This result is expected, given the structure of the novel tasks: agents must be aware of novel gains or perturbations via optic flow, since their internal model for prediction is outdated in novel tasks with novel state transitions. Agents trained with larger observation uncertainties tend to trust their prediction more and ignore changes that must be observed via visual cues in novel tasks (a mismatch between prior knowledge and the task structure). Therefore, like humans and macaques [17], agents must rely more on observation to generalize to these tasks.

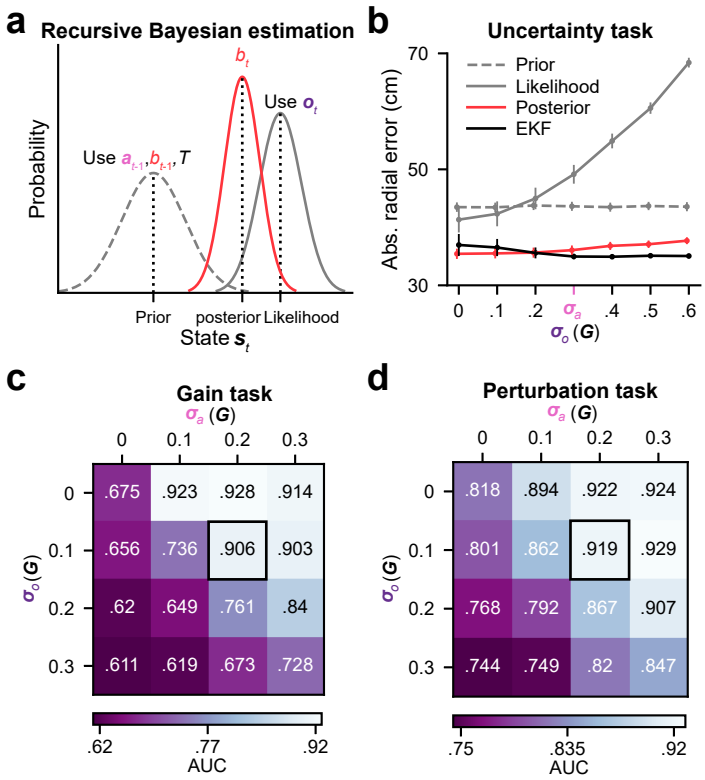

Figure 7: Modular agents with a Kalman filter-like belief update rule generalize worse in the gain and perturbation tasks when relying less on observation. **a**. Schematic of recursive Bayesian estimation (as implemented by the Kalman filter, Methods) in the 1D case, given zero-mean Gaussian noises. The prior is a prediction of the state $\boldsymbol{s}_t$ using the last action $\boldsymbol{a}_{t-1}$, the last belief b_{t-1}, and the state transition T. The state's likelihood depends on visual observation $\boldsymbol{o}_t$. The posterior b_t combines these two sources and provides a state estimation with an uncertainty smaller than only relying on a single source. **b**. Absolute radial error of agents ($n = 8$ random seeds each) tested in the uncertainty task as a function of the observation uncertainty $\boldsymbol{\sigma}_o$, given a fixed uncertainty in prediction $\boldsymbol{\sigma}_a$. The prior model was trained with $\boldsymbol{\sigma}_a = \mathbf{0} \ll \boldsymbol{\sigma}_o$. The likelihood model was trained with $\boldsymbol{\sigma}_o = \mathbf{0} \ll \boldsymbol{\sigma}_a$. The posterior model was trained with $\boldsymbol{\sigma}_o$ values that can be larger or smaller than $\boldsymbol{\sigma}_a$ on each trial (see Methods and text). Error bars denote ± 1 SEM across random seeds. **c**–**d**. AUC of agents trained with combinations of $\boldsymbol{\sigma}_a$ and $\boldsymbol{\sigma}_o$ tested in the gain (**c**) and perturbation (**d**) tasks, averaged across $n = 8$ random seeds for each agent. Black boxes denote the combination used in the previous analyses as the default setting. **b**–**d**. 2000 trials were used for each agent for each uncertainty condition.

Discussion

The brain has evolved advantageous modular architectures for mastering daily tasks. Here, we examined the impact of architectural inductive biases on generalization using deep RL agents. We posited that choosing a task-appropriate modular neural architecture would allow agents to capture the structure of the desired task during training, and then use this knowledge to support generalization in novel tasks with different parameters but a similar structure. To test this hypothesis, we trained agents with neural architectures varying in modularity on a partially observable navigation task, and tested these agents with novel sensorimotor mappings or passive perturbations. Although all agents mastered the training task, agents with more modular architectures separating the computation of internal beliefs from other task variables were better able to form accurate state representations of the environment (task structure) in novel tasks to support better generalization than agents with less modular architectures. This result helps rationalize that macaques use

multiple brain regions in this task [20] and achieve generalization [14].

We also found that modular agents learn a Kalman filter-like belief update rule, weighing the relative reliability of information sources for belief formation, similar to how the brain performs probabilistic inference [16]. Therefore, having higher uncertainty in observation than in prediction during training can bias agents toward dead-reckoning strategies, as they learn not to trust their observation. The agents are then less sensitive to novel patterns in their observation, and thus form wrong internal beliefs when tested in novel tasks, impeding their generalization abilities.

Modular actor-critic RL models bridge neuroscience and AI

The orbitofrontal cortex (OFC) has been suggested to compute state representations by integrating sensory inputs in partially observable environments [25], then project these beliefs to the basal ganglia (BG) [26, 27]. One possible role for the BG is to construct the value of a task using reward-prediction error of the dopamine (DA) system [28], and then select the best action at each state proposed by the cortex that leads to the highest value [29, 30, 31, 32].

Inspired by these brain mechanisms, we similarly modeled our modular actor-critic agent. The modular critic is updated with the reward-prediction error, using an architecture separating the computation of belief states from the computation of value in two different modules, an analog of the OFC-BG circuit. The actor's synaptic weights are optimized for generating the best action maximizing the BG's value. Our result that the modular critic enables agents to achieve significantly greater generalization than agents without such modularization (Fig. 6**c**–**d**) provides one possible rationale for the brain's modular OFC-BG circuit. Furthermore, the advantage of high modularity in actors (Fig. 6**c**–**d**) justifies that the macaque brain in our navigation task uses multiple cortical areas (e.g., dorsomedial superior temporal area, parietal area 7a, dorsolateral prefrontal cortex [PFC]) to construct a control policy [20].

Interestingly, the property that our learning algorithm is only stable when the actor is updated more slowly than the critic (Methods; [23]) can potentially explain the finding that the BG learns from experience faster than the PFC [33]: the actor/cortex cannot learn anything new from the critic/BG if the latter's value estimation has not improved since the last training step.

Achieving zero-shot generalization in RL

Different classes of RL models have different generalization abilities. The classic model-free RL algorithm, reminiscent of the brain's DA system [28], selects actions that maximize the stored value function associated with the environmental model (state transition probabilities and reward functions) used in training. It does not adapt to any changes in the environmental model that should lead to a new value function. The successor representation algorithm [34, 35] decomposes the value into a representation of transition probabilities learned from experience and a reward function model, realizing immediate generalization on new rewards by reconstructing new values with learned transitions and new reward function models. However, upon encountering changes in transition probabilities, such as posed by our gain and perturbation tasks, the successor representation requires a similarly lengthy relearning as a model-free algorithm. On the other hand, the meta-RL algorithm [36] bypasses the tedious relearning of new values for generalization. Inspired by the standalone fast learning system of the PFC that is shaped by (but distinct from) the slow DA-based RL [37], meta-RL uses model-free values to train a standalone RNN policy network that maps inputs to actions over multiple tasks, where the policy network is provided with not only current observations but also previous actions and rewards as inputs. This structure allows the policy network to learn not a single policy, but a learning algorithm that can learn the transition and reward structures of the task at hand from its sequential inputs ("learning to learn"). Therefore, the policy network itself can generalize by learning structures of novel tasks with its own learning algorithm and inputs, without referencing the outdated value for the training tasks [37].

Macaques in our navigation task were trained in a single task instead of multiple tasks, and exhibited instantaneous generalization in the gain and perturbation tasks without further learning (Fig. S2**b**, Fig. S4**e**).

Therefore, rather than "learning to learn" like meta-RL, our modeling objective is to achieve zero-shot generalization like animals. We similarly used model-free values (critic) to train a standalone policy (actor), but we did not provide the reward as input to the actor, so it cannot use the reward feedback to assess and improve the quality of its actions in novel tasks. We only trained the model in a single task to mimic macaque training, so the actor does not experience a wide range of transition probabilities from which to infer high-level rules. With task-appropriate inductive biases, however, it can acquire an accurate task structure through a single training task, and use it to map novel observations from novel tasks to suitable actions without referencing the outdated critic's value. Our modeling successfully overcomes changing transition probabilities to achieve zero-shot generalization like animals [14].

Future directions

A fundamental question bridging AI and neuroscience is how to reconcile the many distinct RL models for which analogs have been observed in the brain and behavior. For example, the midbrain DA neurons are believed to implement model-free RL from trial and error [28]; the successor representation has the capacity to recapitulate place and grid cell properties [34]; the learning-to-learn system in PFC is explicated by meta-RL [36]; the capacity of humans and animals to flexibly plan is suggestive of model-based RL [38]; the ability of macaques to achieve zero-shot generalization is explained by our model. The brain's reward-based learning system could be a family of many overlapping algorithms implemented by different brain regions, such that the aforementioned models each capture a single aspect of our cognitive capacities. Animal behavior might inherit properties from multiple algorithms [39], or flexibly choose the most appropriate algorithm depending on the current task demands [40]. Future studies may aspire to harmonize the diverse findings in RL literature to develop a more comprehensive understanding of the brain's learning mechanisms.

It is also important to consider how the brain exhibits two levels of modularity for generalization. On the architectural level, the modularity enforced by heterogeneity among functionally specialized brain regions is an inductive bias that has evolved for animals' daily tasks. On the representational level, even in the same region, specialized neural clusters as a form of modularity can emerge through learning [41]. Our finding that agents with architecturally modular networks generalize better than those with holistic networks suggests that for the current task, representational modularity in a homogeneous recurrently connected module might not emerge, or be insufficient for generalization. Because in theory, a holistic critic could have internally formed two neural clusters to compute beliefs and values separately, as enforced by architectural modularity. One possibility is that specialized neural clusters shall emerge when agents are trained over many tasks with some shared task variables instead of a single task [42], and then these clusters could be flexibly combined in the face of novel tasks reusing these variables, leading to the potential for holistic agents on par with architecturally modular agents in generalization. It is also suggested that imposing more realistic constraints in optimizing artificial neural networks, e.g., embedding neurons in physical and topological spaces where longer connections are more expensive, can lead to modular clusters [43]. Together, these prompt future investigations into the reconciliation of these two levels of modularity, e.g., studying the generalization of modular neural architectures after being exposed to a diverse task set and more biological constraints. We hope these directions will deepen our understanding of the brain's generalization mechanisms, and inspire the development of more generalizable AI.

Methods

Task structure

The navigation task and its manipulations were originally designed for macaques [11, 12, 14, 17, 20, 21]. All macaque data used in this paper were from previous works [11, 12], where details of the animal experiment setup can be found. We modeled this task as a POMDP [22] for RL agents, containing a state space S, an action space A, a transition probability T, a reward function R, an observation space Ω, an observation probability O, and a temporal discount factor $\gamma = 0.97$ over steps within a trial.

Each state $\boldsymbol{s}_t \in S$ is a vector $[s_{x_t}, s_{y_t}, s_{\theta_t}, s_{v_t}, s_{\omega_t}, g_{x_t}, g_{y_t}]^\top$ containing the agent's x and y positions (cm), head direction (°), linear and angular velocities (cm/s, °/s), and the target's x and y positions (cm). The initial state of each trial was defined as $\boldsymbol{s}_0 = [0, 0, 90, 0, 0, g_{x_0}, g_{y_0}]^\top$, since the agent always starts from the origin facing forward (90°). The target location was uniformly sampled on the ground plane before the agent, with the radius $g_r \in [100\text{ cm}, 400\text{ cm}]$ and the angle $g_\theta \in [90° - 35°, 90° + 35°]$ relative to the agent's initial location. Specifically, angles were drawn uniformly within the field of view, $g_\theta \sim \mathcal{U}(55°, 125°)$, and we sampled radial distances as $g_r \sim \sqrt{\mathcal{U}(100^2, 400^2)}$ to ensure a spatially uniform distribution in 2D. Target positions in Cartesian coordinates are then $g_{x_0} = g_r \cos(g_\theta)$, $g_{y_0} = g_r \sin(g_\theta)$.

Each action $\boldsymbol{a}_t \in A$ is a vector $[a_{v_t}, a_{\omega_t}]^\top$ containing the agent's linear and angular joystick actions, bounded in $[-1, 1]$ for each component.

State transitions $\boldsymbol{s}_{t+1} \sim T(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, \boldsymbol{a}_t)$ were defined as $\boldsymbol{s}_{t+1} = \boldsymbol{f}_{\text{env}}(\boldsymbol{s}_t, \boldsymbol{a}_t) + \boldsymbol{\eta}_t$, where

$$\boldsymbol{f}_{\text{env}}(\boldsymbol{s}_t, \boldsymbol{a}_t) = \begin{bmatrix} s_{x_t} + \Delta t\, s_{v_t} \cos s_{\theta_t} \\ s_{y_t} + \Delta t\, s_{v_t} \sin s_{\theta_t} \\ s_{\theta_t} + \Delta t\, s_{\omega_t} \\ n\, G_v a_{v_t} + p_{v_t} \\ n\, G_\omega a_{\omega_t} + p_{\omega_t} \\ g_{x_t} \\ g_{y_t} \end{bmatrix} \tag{1}$$

and zero-mean independent Gaussian process noise added to the velocities

$$\boldsymbol{\eta}_t = [0, 0, 0, \eta_{v_t}, \eta_{\omega_t}, 0, 0]^\top, \qquad [\eta_{v_t}, \eta_{\omega_t}]^\top \sim \mathcal{N}(\mathbf{0}, \text{diag}(\boldsymbol{\sigma}_{\text{a}}^2))$$

with standard deviation $\boldsymbol{\sigma}_a = [\sigma_{a_v}, \sigma_{a_\omega}]^\top$. The operator diag$(\cdot)$ constructs a diagonal matrix with its vector argument on the diagonal. The timestep is $\Delta t = 0.1$ s. Joystick gain $\boldsymbol{G} = [G_v, G_\omega]^\top = [200\text{ cm/s}, 90°/\text{s}]^\top$ maps dimensionless linear and angular joystick actions to units of velocities. Gain multiplier n scales $\boldsymbol{G}$. Linear and angular perturbation velocities are p_{v_t} and p_{ω_t}.

The reward function $R(\boldsymbol{s}_t, \boldsymbol{a}_t)$ maps a state-action pair to a scalar r_t. We firstly defined an action threshold $a^* = 0.1$ to distinguish between when the agent had not yet begun moving, and when they moved and stopped: the agent must increase the magnitude of at least one action component above a^* in the beginning (start criterion), then the agent must reduce the magnitude of both action components below a^* to indicate a stop (stop criterion). Non-zero rewards were only offered in the last step of each trial and if the agent satisfied both criteria. For the non-zero rewards, we defined $\boldsymbol{d}_t = [s_{x_t}, s_{y_t}]^\top - [g_{x_t}, g_{y_t}]^\top$ as the displacement between the agent's and the target's locations, and a reward $r_t = 10$ would be given if the Euclidean distance $||\boldsymbol{d}_t||_2$ was smaller than the radius of the reward zone $d^* = 65$ cm. To facilitate training in the early stages, we allowed a small reward $r_t = 10 \exp(-\frac{1}{2}\boldsymbol{d}_t^\top \Sigma_r^{-1} \boldsymbol{d}_t)$ if the agent stopped outside the reward zone, where $\Sigma_r = (\frac{d^*}{1.5})^2 I_2$ is a constant matrix, and I_2 denotes the identity matrix of size 2.

A trial ended when the agent stopped, or if t exceeded the maximum trial duration 3.4 s. For later convenience, let D_t denote a trial completion flag that equals 1 if the trial is done at t, otherwise 0. A new trial thereafter started with a new sampled initial state $\boldsymbol{s}_0$.

Observation $\boldsymbol{o}_t \in \Omega$ is a vector $[o_{v_t}, o_{\omega_t}, o_{g_{x,t}}, o_{g_{y,t}}]^\top$ containing observations of the agent's linear and angular velocities through optic flow, and the target's x and y positions when visible in the first 0.3 s of each trial. $\boldsymbol{o}_t \sim O(\boldsymbol{o}_t|\boldsymbol{s}_t)$ was defined as

$$\boldsymbol{o}_t = H_t \boldsymbol{s}_t + \boldsymbol{\zeta}_t \tag{2}$$

where $\boldsymbol{\zeta}_t$ is a zero-mean Gaussian observation noise, and the observation model H_t is a 4×7 matrix filled mostly with zeros, except for a few observable elements depending on the time within a trial: When $t \leq 0.3$ s, the target is visible, so $H^{1,4}, H^{2,5}, H^{3,6}, H^{4,7}$ are equal to 1, where superscripts denote row and column; after $t = 0.3$ s the target disappears and only the optic flow is observable, so only $H^{1,4}, H^{2,5}$ are 1. For the observation noise, $\boldsymbol{\zeta}_0 = \mathbf{0}$, $\boldsymbol{\zeta}_{t>0} = [\zeta_{v_t}, \zeta_{\omega_t}, 0, 0]^\top$, where ζ_{v_t} and ζ_{ω_t} denote linear and angular observation noises, and $[\zeta_{v_t}, \zeta_{\omega_t}]^\top \sim \mathcal{N}(\mathbf{0}, \text{diag}(\boldsymbol{\sigma}_{\text{o}}^2))$ with standard deviation $\boldsymbol{\sigma}_o = [\sigma_{o_v}, \sigma_{o_\omega}]^\top$.

Task variables

Training task. The gain multiplier is given by $n = 1$. There were no perturbations, so for any t, $p_{v_t} = p_{\omega_t} = 0$. Process and observation noise standard deviations were in units of $\boldsymbol{G}$, i.e., $\boldsymbol{\sigma}_a = \alpha_a \boldsymbol{G}$, $\boldsymbol{\sigma}_o = \alpha_o \boldsymbol{G}$, where $\alpha_a = 0.2$, $\alpha_o = 0.1$ were used to train agents before Fig. 7; $\alpha_a, \alpha_o \in [0, 0.3]$ were used in Fig. 7 except Fig. 7**b**.

Gain task. The gain multiplier varied within $n \in [1, 4]$ for testing agents. Noise standard deviations were also multiplied by the same gains, i.e., $\boldsymbol{\sigma}_a = \alpha_a n \boldsymbol{G}$, $\boldsymbol{\sigma}_o = \alpha_o n \boldsymbol{G}$. There were no perturbations.

Perturbation task. Variables $n, \boldsymbol{\sigma}_a, \boldsymbol{\sigma}_o$ were the same as those in the training task. There were three perturbation variables uniformly sampled for each trial: perturbation peak time relative to the trial start $t_p \sim \mathcal{U}(0.5 \text{ s}, 1.5 \text{ s})$, perturbation linear and angular peaks $p_{v_{\text{peak}}} \sim \mathcal{U}(-200 \text{ cm/s}, 800 \text{ cm/s})$ or $\mathcal{U}(-200 \text{ cm/s}, 200 \text{ cm/s})$, $p_{\omega_{\text{peak}}} \sim \mathcal{U}(-180°/\text{s}, 180°/\text{s})$ or $\mathcal{U}(-120°/\text{s}, 120°/\text{s})$. These sampled variables determined Gaussian-shaped linear and angular perturbations, defined as

$$[p_{v_t}, p_{\omega_t}]^\top = \begin{cases} [p_{v_{\text{peak}}}, p_{\omega_{\text{peak}}}]^\top \cdot \exp\left[-\frac{1}{2}\left(\frac{t-t_p}{0.2}\right)^2\right], & \text{if } t_p - 0.5 \le t \le t_p + 0.5 \\ [0, 0]^\top, & \text{otherwise} \end{cases}$$

Uncertainty task. The gain multiplier was $n = 1$, and there were no perturbations. The prior model was trained with $\boldsymbol{\sigma}_a = \mathbf{0}$, $\boldsymbol{\sigma}_o = 0.8\boldsymbol{G}$; the likelihood model was trained with $\boldsymbol{\sigma}_a = 0.8\boldsymbol{G}$, $\boldsymbol{\sigma}_o = \mathbf{0}$; the posterior and the EKF models were trained with $\boldsymbol{\sigma}_a = 0.3\boldsymbol{G}$ and a randomly varying $\boldsymbol{\sigma}_o = \alpha_o \boldsymbol{G}$ where $\alpha_o \sim \mathcal{U}(0, 1)$ was drawn independently for each trial. A standard deviation of $\mathbf{0}$ denotes the noise-free case.

Belief modeling

The state $\boldsymbol{s}_t$ is partially observable in our task, therefore, an agent cannot decide on $\boldsymbol{a}_t$ only based on the current sensory inputs. It can internally maintain a belief state representation b_t, which is a posterior of $\boldsymbol{s}_t$, for decision-making. We considered both a model-based inference method and a gradient-based optimization method to model the belief.

Recursive Bayesian estimation. When the transition probability T and the observation probability O are known, the belief is a posterior of $\boldsymbol{s}_t$ given all available observations and actions, i.e., $b_t = p(\boldsymbol{s}_t | \boldsymbol{o}_{0:t}, \boldsymbol{a}_{0:t-1})$, and can be inferred recursively as

$$b_t = \frac{1}{C} O(\boldsymbol{o}_t | \boldsymbol{s}_t) \int_{\boldsymbol{s}_{t-1}} T(\boldsymbol{s}_t | \boldsymbol{s}_{t-1}, \boldsymbol{a}_{t-1})\, b_{t-1}\, d\boldsymbol{s}_{t-1} \tag{3}$$

where $C = p(\boldsymbol{o}_t | \boldsymbol{o}_{0:t-1}, \boldsymbol{a}_{0:t-1})$ is a normalization constant, and $b_{t-1} = p(\boldsymbol{s}_{t-1} | \boldsymbol{o}_{0:t-1}, \boldsymbol{a}_{0:t-2})$.

EKF belief. When all variables are Gaussian in the Recursive Bayesian estimation and T is nonlinear, the EKF [24] is a tractable method that uses a local linearization to approximate eq. (3). The belief here is a Gaussian density $b_t = \mathcal{N}(\hat{\boldsymbol{s}}_t, P_t)$. To simplify the computation here, we express position in relative coordinates by letting the initial belief mean be $\hat{\boldsymbol{s}}_0 = [\hat{s}_{x_0}, \hat{s}_{y_0}, \hat{s}_{\theta_0}, \hat{s}_{v_0}, \hat{s}_{\omega_0}] = [-g_{x_0}, -g_{y_0}, 90, 0, 0]$, and let the state transition $\boldsymbol{f}_{\text{env}}$ contain the first five equations in eq. (1) to reduce the dimensionality of the state by two. Let ϵ denote a small number 10^{-8}, we defined the initial belief covariance $P_0 = \epsilon I_5$. Let a 5×5 matrix N_a denote the Gaussian process noise covariance filled with 0 except $N_a^{4,4} = \sigma_{a_v}^2$, $N_a^{5,5} = \sigma_{a_\omega}^2$. The observation's dimensionality was reduced by two by omitting the target location, yielding $\boldsymbol{o}_t = [o_{v_t}, o_{\omega_t}]^\top$. The observation model H in eq. (2) then becomes a 2×5 matrix filled with 0, except $H^{1,4} = H^{2,5} = 1$. Let $N_o = \text{diag}(\boldsymbol{\sigma}_o^2)$ denote the Gaussian observation noise covariance. Any 0 variance components in $\boldsymbol{\sigma}_a^2, \boldsymbol{\sigma}_o^2$ were replaced with a minimal variance of ϵ for N_a, N_o.

$\boldsymbol{f}_{\text{env}}$ at $b_{t-1} = \mathcal{N}(\hat{\boldsymbol{s}}_{t-1}, P_{t-1})$ was locally linearized as

$$A_{t-1} = \frac{\partial \boldsymbol{f}_{\text{env}}}{\partial \hat{\boldsymbol{s}}_{t-1}} = \begin{bmatrix} 1 & 0 & -\hat{s}_{v_{t-1}} \Delta t \sin \hat{s}_{\theta_{t-1}} & \Delta t \cos \hat{s}_{\theta_{t-1}} & 0 \\ 0 & 1 & \hat{s}_{v_{t-1}} \Delta t \cos \hat{s}_{\theta_{t-1}} & \Delta t \sin \hat{s}_{\theta_{t-1}} & 0 \\ 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The EKF's prediction step (eq. (4)) uses $\boldsymbol{a}_{t-1}$ to get a predicted belief $b_{t|t-1}$. Note that given our A_{t-1}, velocity variance elements in the prediction $P^{4,4}_{t|t-1}, P^{5,5}_{t|t-1}$ only depend on N_a.

$$\begin{aligned} \hat{\boldsymbol{s}}_{t|t-1} &= \boldsymbol{f}_{\text{env}}(\hat{\boldsymbol{s}}_{t-1}, \boldsymbol{a}_{t-1}) \\ P_{t|t-1} &= A_{t-1} P_{t-1} A_{t-1}^{\top} + N_a \end{aligned} \tag{4}$$

The EKF's update step (eq. (5)) uses $b_{t|t-1}$ and $\boldsymbol{o}_t$ to get the final belief $b_t = \mathcal{N}(\hat{\boldsymbol{s}}_t, P_t)$. K_t is known as the Kalman gain which specifies the relative weights of prediction and observation. Mathematically, it is only affected by N_a and N_o and is independent of P_{t-1} in our task. As intuitively, past velocities do not predict current velocities except via the agent's actions, which it knows.

$$\begin{aligned} K_t &= P_{t|t-1} H^{\top} (H P_{t|t-1} H^{\top} + N_o)^{-1} \\ \hat{\boldsymbol{s}}_t &= \hat{\boldsymbol{s}}_{t|t-1} + K_t (\boldsymbol{o}_t - H \hat{\boldsymbol{s}}_{t|t-1}) \\ P_t &= (I_5 - K_t H) P_{t|t-1} \end{aligned} \tag{5}$$

RNN belief. When the transition and the observation probabilities T, O are unknown to the agent, to support decision-making, an internal belief could be approximated via gradient-based optimization. We used RNNs to integrate partial observations $\boldsymbol{o}_t$ and motor efference copies $\boldsymbol{a}_{t-1}$ over time, trained end-to-end using the RL objective in our task (see below). It was shown in the result that an EKF-like belief update rule could emerge in RNNs, and therefore, the belief b_t resides in the RNN's hidden state $\boldsymbol{h}_t$. Each RNN maintains a hidden state $\boldsymbol{h}_t = \boldsymbol{f}_{\text{RNN}}(\boldsymbol{o}_t, \boldsymbol{a}_{t-1}, \boldsymbol{h}_{t-1})$ or $\boldsymbol{h}_t = \boldsymbol{f}_{\text{RNN}}(\boldsymbol{o}_t, \boldsymbol{a}_{t-1}, \boldsymbol{h}_{t-1}, \boldsymbol{a}_t)$ depending on its inputs (Fig. 6**a**–**b**). b_t encoded implicitly in $\boldsymbol{h}_t$ is used by other neurons to compute $\boldsymbol{a}_t$ or Q_t in the actor or critic.

RL with EKF belief

Our RL algorithm for training the EKF agent with an EKF belief is based on an actor-critic approach called the Twin Delayed Deep Deterministic Policy Gradient (TD3) [23], referred to as EKF-TD3. We first computed beliefs using EKF as described above, and then trained neural networks to use those beliefs as inputs to guide actions.

Networks. Each agent has two critics with identical architectures but different initial weights to address the maximization bias in value estimation (see the critic update section below and [44]), although in the main text we only showed one of the critics used to train the actor to generate actions. Let $\boldsymbol{i}_t$ denote the state-related inputs. All neural networks in an EKF agent were feed-forward, provided with the mean and covariance of b_t computed by the EKF, i.e., $\boldsymbol{i}_t = \{\hat{\boldsymbol{s}}_t, P_t\}$. The actor and two critics are $\boldsymbol{a}_t = \boldsymbol{\pi}_\mu(\boldsymbol{i}_t)$, $Q_{t_1} = \mathcal{Q}_{\nu_1}(\boldsymbol{i}_t, \boldsymbol{a}_t)$, $Q_{t_2} = \mathcal{Q}_{\nu_2}(\boldsymbol{i}_t, \boldsymbol{a}_t)$, where μ, ν_1, ν_2 denote neural parameters.

Exploration. Since our actor is a deterministic function, to realize exploration in training, we combined the actor's output with a zero-mean Gaussian exploration noise $\boldsymbol{\beta}_t$, and clipped the sum to the box $[-1, 1]$:

$$\boldsymbol{a}_t = \text{clip}(\boldsymbol{\pi}_\mu(\boldsymbol{i}_t) + \boldsymbol{\beta}_t, -1, 1), \qquad \boldsymbol{\beta}_t \sim \mathcal{N}(\boldsymbol{0}, \sigma^2_{\text{exp}} \text{I}_2) \tag{6}$$

To ensure the agent can properly stop without noise variability, we let $\boldsymbol{\beta}_t = \boldsymbol{0}$ if the actor's output $\boldsymbol{\pi}_\mu(\boldsymbol{i}_t)$ is below the action threshold.

Experience replay. Instead of learning on the current trial, we used off-policy RL by storing experience in a replay buffer $\mathcal{B}$ and frequently sampling data from $\mathcal{B}$ to train the agent. At each state $\boldsymbol{s}_t$, the EKF computed $\boldsymbol{i}_t$ for the actor to generate $\boldsymbol{a}_t$ following eq. (6). The agent observed the reward r_t, next input $\boldsymbol{i}_{t+1}$, and trial completion flag D_t, and stored the one-step transition tuple $(\boldsymbol{i}_t, \boldsymbol{a}_t, r_t, \boldsymbol{i}_{t+1}, D_t)$ in $\mathcal{B}$. The buffer $\mathcal{B}$ had a capacity of 1.6×10^6 transitions, storing data on a first-in, first-out (FIFO) basis. Furthermore, we augmented the experience by also storing the mirror transition $(\hat{\boldsymbol{i}}_t, \hat{\boldsymbol{a}}_t, r_t, \hat{\boldsymbol{i}}_{t+1}, D_t)$ generated by reflecting the original data across the y-axis.

Target networks. The learning of value in TD3 is akin to deep Q-learning [45]. Using the Bellman equation, ideally, the agent can learn to estimate the value $\mathcal{Q}_{\nu_j}(\boldsymbol{i}_t, \boldsymbol{a}_t)$ by regressing the learning target $y_t = r_t +$

$\gamma \mathcal{Q}_{\nu_j}(\boldsymbol{i}_{t+1}, \boldsymbol{\pi}_\mu(\boldsymbol{i}_{t+1}))$, i.e., the one-step bootstrapping of the value after receiving the reward r_t, observing the next input $\boldsymbol{i}_{t+1}$, and estimating the next action $\boldsymbol{\pi}_\mu(\boldsymbol{i}_{t+1})$. One stability issue here is that the neural parameters for optimization are also used to construct the learning target y_t, which changes at each learning step. To obtain a more stable y_t, we thus maintained a copy of actor and critic networks with more slowly changing parameters μ' and ν'_j used in y_t, referred to as target actor and critic networks. These parameters were initialized to be the same as μ, ν_j and passed through an exponential moving average,

$$\begin{aligned} \mu' &\leftarrow \tau\mu + (1-\tau)\mu' \\ \nu'_j &\leftarrow \tau\nu_j + (1-\tau)\nu'_j \end{aligned} \tag{7}$$

We used $\tau = 0.005$.

Critic update. We sampled a batch of $M = 256$ transitions from the buffer each time,

$$(\boldsymbol{i}^{(k)}, \boldsymbol{a}^{(k)}, r^{(k)}, \boldsymbol{i}'^{(k)}, D^{(k)})_{k=1,2,\cdots,M} \sim \mathcal{B}$$

where the temporal subscript is omitted, and $\boldsymbol{i}'^{(k)}$ denotes the next input after $\boldsymbol{i}^{(k)}$. The next action given $\boldsymbol{i}'^{(k)}$ was estimated by the target actor network as

$$\boldsymbol{a}'^{(k)} = \mathrm{clip}(\boldsymbol{\pi}_{\mu'}(\boldsymbol{i}'^{(\mathrm{k})}) + \boldsymbol{\beta}'^{(\mathrm{k})}, -1, 1), \qquad \boldsymbol{\beta}'^{(\mathrm{k})} \sim \mathrm{clip}(\mathcal{N}(\boldsymbol{0}, 0.05^2 \mathrm{I}_2), -0.1, 0.1) \tag{8}$$

where $\boldsymbol{\beta}'^{(k)}$ is small zero-mean Gaussian noise clipped to $[-0.1, 0.1]$ to smooth the action estimation.

The learning target $y^{(k)}$ used the smaller value estimation between two target critics to reduce the maximization bias [44], and was truncated at the end of each trial ($D^{(k)} = 1$). The learning objective of the two critics, $J(\nu_j), j = 1, 2$, was to regress the learning target $y^{(k)}$, defined as

$$\begin{aligned} y^{(k)} &= r^{(k)} + (1 - D^{(k)})\,\gamma \min_{j=1,2} \mathcal{Q}_{\nu'_j}(\boldsymbol{i}'^{(k)}, \boldsymbol{a}'^{(k)}) \\ J(\nu_j) &= \frac{1}{M}\sum_{k=1}^{M}(y^{(k)} - \mathcal{Q}_{\nu_j}(\boldsymbol{i}^{(k)}, \boldsymbol{a}^{(k)}))^2 \end{aligned} \tag{9}$$

The gradient $\nabla_{\nu_j} J(\nu_j)$ was computed by backpropagation (BP). Critic parameters ν_j were updated (see the agent training section below for optimizers) using $\nabla_{\nu_j} J(\nu_j)$ to minimize $J(\nu_j)$.

Actor update. The actor's parameter μ was updated once for every two critic updates. The actor's learning objective $J(\mu)$ was to maximize the value of the first critic, defined as

$$J(\mu) = \frac{1}{M}\sum_{k=1}^{M} \mathcal{Q}_{\nu_1}(\boldsymbol{i}^{(k)}, \boldsymbol{\pi}_\mu(\boldsymbol{i}^{(k)})) \tag{10}$$

The gradient $\nabla_\mu J(\mu)$ was computed by BP. The actor parameter μ was updated using $\nabla_\mu J(\mu)$ to maximize $J(\mu)$. Note that the critic parameter ν_1 was not updated here.

RL with RNN belief

We developed a memory-based TD3 model leveraging RNNs to construct a form of internal beliefs to tackle POMDPs, referred to as RNN-TD3. All agents except the EKF agent were trained by this algorithm.

Networks. Let $\boldsymbol{i}_t = \{\boldsymbol{o}_t, \boldsymbol{a}_{t-1}\}$ and $\boldsymbol{h}_t$ denote the state-related inputs and the RNN's hidden state. The actor and two critics are $\{\boldsymbol{a}_t, \boldsymbol{h}^\mu_t\} = \boldsymbol{\pi}_\mu(\boldsymbol{i}_t, \boldsymbol{h}^\mu_{t-1})$, $\{Q_{t_j}, \boldsymbol{h}^{\nu_j}_t\} = \mathcal{Q}_{\nu_j}(\boldsymbol{i}_t, \boldsymbol{a}_t, \boldsymbol{h}^{\nu_j}_{t-1}), j = 1, 2$, where we interpret that the belief b_t is implicitly encoded in all $\boldsymbol{h}_t$ evolving over time. At the beginning of each trial, $\boldsymbol{h}_{t-1}$ and $\boldsymbol{a}_{t-1}$ were initialized to zeros. For simplicity, we drop $\boldsymbol{h}_t$ in our notations for all networks' outputs.

Exploration. Similar to that of EKF-TD3 (eq. (6)), we added zero-mean Gaussian exploration noise to the output of the actor during training if the output is above the action threshold,

$$\boldsymbol{a}_t = \text{clip}(\boldsymbol{\pi}_\mu(\boldsymbol{i}_t, \boldsymbol{h}_{t-1}^\mu) + \boldsymbol{\beta}_t, -1, 1), \qquad \boldsymbol{\beta}_t \sim \mathcal{N}(\mathbf{0}, \sigma_{\text{exp}}^2 I_2) \tag{11}$$

Experience replay. Similar to that of EKF-TD3, but rather than storing one-step transition tuples, the replay buffer $\mathcal{B}$ stored the whole trajectory for each trial of N time steps,

$$(\boldsymbol{i}_0, \boldsymbol{a}_0, r_0, D_0, \cdots, \boldsymbol{i}_{N-1}, \boldsymbol{a}_{N-1}, r_{N-1}, D_{N-1})$$

and its mirror image, because RNNs have hidden states $\boldsymbol{h}_t$ generally depend on the entire history of inputs, not just the most recent ones. Each action was obtained using eq. (11). The FIFO buffer had a capacity of 10^5 trajectories.

Target networks. Same as that of EKF-TD3.

Critic update. Similar to that of EKF-TD3, but critics here also needed to learn the temporal structure. Since the trial duration N varies across trials, we first sampled a trial duration $\tilde{N}$ from the buffer $\mathcal{B}$, then sampled a batch of $M = 16$ trajectories with the same duration $\tilde{N}$,

$$\left(\boldsymbol{i}_t^{(k)}, \boldsymbol{a}_t^{(k)}, r_t^{(k)}, \boldsymbol{i'}_t^{(k)}, D_t^{(k)}\right)_{t=0,\cdots,\tilde{N}-1}^{k=1,\cdots,M} \sim \mathcal{B}$$

where $\boldsymbol{i'}_t^{(k)} = \boldsymbol{i}_{t+1}^{(k)}$. The next action $\boldsymbol{a'}_t^{(k)}$, the learning target $y_t^{(k)}$, and the learning objective of the two critics $J(\nu_j)$ were

$$\boldsymbol{a'}_t^{(k)} = \text{clip}(\boldsymbol{\pi}_{\mu'}(\boldsymbol{i'}_{\text{t}}^{(\text{k})}, \boldsymbol{h}_{\text{t}}^{\mu'(\text{k})}) + \boldsymbol{\beta'}_{\text{t}}^{(\text{k})}, -1, 1), \qquad \boldsymbol{\beta'}_{\text{t}}^{(\text{k})} \sim \text{clip}(\mathcal{N}(\mathbf{0}, 0.05^2 \text{I}_2), -0.1, 0.1) \tag{12}$$

$$\begin{aligned} y_t^{(k)} &= r_t^{(k)} + (1 - D_t^{(k)})\, \gamma \min_{j=1,2} \mathcal{Q}_{\nu'_j}(\boldsymbol{i'}_t^{(k)}, \boldsymbol{a'}_t^{(k)}, \boldsymbol{h}_t^{\nu'_j(k)}) \\ J(\nu_j) &= \frac{1}{M\tilde{N}} \sum_{k=1}^{M} \sum_{t=0}^{\tilde{N}-1} \left(y_t^{(k)} - \mathcal{Q}_{\nu_j}(\boldsymbol{i}_t^{(k)}, \boldsymbol{a}_t^{(k)}, \boldsymbol{h}_{t-1}^{\nu_j(k)})\right)^2 \end{aligned} \tag{13}$$

The gradient $\nabla_{\nu_j} J(\nu_j)$ was computed by BP through time (BPTT). Critic parameters ν_j were updated using $\nabla_{\nu_j} J(\nu_j)$ to minimize $J(\nu_j)$.

Actor update. Similar to that of EKF-TD3, but the actor here needed to learn the temporal structure. The actor's learning objective $J(\mu)$ was

$$J(\mu) = \frac{1}{M\tilde{N}} \sum_{k=1}^{M} \sum_{t=0}^{\tilde{N}-1} \mathcal{Q}_{\nu_1}\left(\boldsymbol{i}_t^{(k)}, \boldsymbol{\pi}_\mu(\boldsymbol{i}_t^{(k)}, \boldsymbol{h}_{t-1}^{\mu(k)}), \boldsymbol{h}_{t-1}^{\nu_1(k)}\right) \tag{14}$$

The gradient $\nabla_\mu J(\mu)$ was computed by BPTT. The actor parameters μ were updated using $\nabla_\mu J(\mu)$ to maximize $J(\mu)$.

Agent training

All network parameters μ, ν_1, ν_2 were updated by the RAdam optimizers [46]. Optimizer parameters were set as follows: learning rates annealed from 3×10^{-4} to 5×10^{-5}, exponential decay rates for the first and second moment estimates $= 0.9, 0.999$, a constant added in denominators for numerical stability $= 1.5 \times 10^{-4}$, and weight decay $= 0$. The critics were updated once for every $c = 4$ interactions with the environment. The actor was updated once for every two critic updates.

During training, we periodically validated the agent's performance with 300 validation trials, and used the moments when the agent achieved 20% and 90% accuracy to split the whole training course into three phases. The learning rates for the actor and critics, the exploration noise σ_{exp} (eq. (6),(11)), and the observation noise $\boldsymbol{\sigma}_o$ (eq. (2)) in each phase were set as follows: In phase I, learning rates were 3×10^{-4}, $\sigma_{\text{exp}} = 0.8$, $\boldsymbol{\sigma}_o = \mathbf{0}$. In phase II, learning rates were 3×10^{-4}, $\sigma_{\text{exp}} = 0.5$, $\boldsymbol{\sigma}_o = \alpha_o \boldsymbol{G}$, where α_o is defined in the training or uncertainty tasks. In phase III, learning rates were 5×10^{-5}, $\sigma_{\text{exp}} = 0.4$, $\boldsymbol{\sigma}_o = \alpha_o \boldsymbol{G}$.

We summarize the EKF/RNN-TD3 algorithms for training all our agents in Algorithm 1.

Algorithm 1 EKF/RNN-TD3

Initialize network parameters μ, ν_1, ν_2, let target network parameters $\mu', \nu_1', \nu_2' \leftarrow \mu, \nu_1, \nu_2$
Initialize replay buffer $\mathcal{B}$ and optimizers for each network, let update timer $\tilde{t} = 0$
for trial $= 1$ **to** max number of trials **do**
  Choose phase-specific $\boldsymbol{\sigma}_o, \sigma_{\text{exp}}$, sample initial state $\boldsymbol{s}_0$
  for t $= 0$ **to** max trial duration **do**
    Receive $\boldsymbol{o}_t$, construct $\boldsymbol{i}_t$ with $\boldsymbol{o}_t, \boldsymbol{a}_{t-1}$
    Select $\boldsymbol{a}_t$ using eq. (6) (EKF) / eq. (11) (RNN)
    Receive $r_t, \boldsymbol{o}_{t+1}, D_t$, construct $\boldsymbol{i}_{t+1}$ with $\boldsymbol{o}_{t+1}, \boldsymbol{a}_t$
    Store transition $(\boldsymbol{i}_t, \boldsymbol{a}_t, r_t, \boldsymbol{i}_{t+1}, D_t)$ and its mirror image in $\mathcal{B}$ (EKF only)
    if $\tilde{t}$ mod $c = 0$ **then** ▷ Update critic for every c steps
      Sample transitions (EKF) / trajectories (RNN) from $\mathcal{B}$
      Update ν_1, ν_2 by critic optimizers following eqs. (8) to (9) (EKF) / eqs. (12) to (13) (RNN)
    end if
    if $\tilde{t}$ mod $2c = 0$ **then** ▷ Update actor for every $2c$ steps
      Update μ by actor optimizer following eq. (10) (EKF) / eq. (14) (RNN)
      Update μ', ν_1', ν_2' using eq. (7)
    end if
    $\tilde{t} \leftarrow \tilde{t} + 1$
    if $D_t = 1$ **then** ▷ Trial is done
      break
    end if
  end for
  Store trajectory $(\boldsymbol{i}_t, \boldsymbol{a}_t, r_t, D_t)_{t=0,\cdots,N-1}$ and its mirror image in $\mathcal{B}$ (RNN only)
end for

Agent selection

During phases II and III training, every 500 trials we saved neural parameters of each network, and ended training after the agent had experienced 10^5 trials. To fairly compare agents' performance in each task (training, gain, perturbation, uncertainty), we tested all of these 200 sets of stored parameters for each task with one or multiple test sets with 300 trials each. Training task: one test set with the training task's variables; gain task: four test sets with the gain$= 1\times, 2\times, 3\times, 4\times$; perturbation task: three test sets with the same perturbation variables as those in the none, small, and large conditions in Fig. 5**a**; uncertainty task: three test sets with $\boldsymbol{\sigma}_a = 0.3\boldsymbol{G}$ and $\boldsymbol{\sigma}_o = \{\mathbf{0}, 0.4\boldsymbol{G}, 0.8\boldsymbol{G}\}$. For analyses in each task, we endowed each agent with the neural parameters that allowed it to achieve the highest reward rate (number of rewarded trials per second averaged across test sets) in that task.

Agent architectures

Although all agents had two architecturally identical critics, we only showed one in the main text (Fig. 1**e**, Fig. 6**b**). All RNNs were implemented as Long Short-Term Memory (LSTM) networks [47]. All MLP layers linearly transformed inputs and then applied ReLU nonlinearities. The output of critics Q_t was produced by

a linear unit without any nonlinearity; the linear and angular control outputs of the actors $\boldsymbol{a}_t$ were bounded to $[-1, 1]$ by hyperbolic tangent nonlinearities. In the holistic critic/actor (Fig. 6**a**–**b**), there were 220 LSTM units. In all other architectures in Fig. 6**a**–**b**, each RNN module had 128 LSTM units, and each MLP module contained two layers with 300 ReLU units in each. All architectures, as a result, had a similar number of parameters (Fig. S6**a**). The EKF agent's actor and two critics used the same architecture consisting of an MLP module with two layers, each with 300 ReLU units.

No-generalization hypothesis

For each gain trial with a novel gain $n\boldsymbol{G}$ for $n > 1$, or for each perturbation trial with novel non-zero perturbation velocities p_{v_t}, p_{w_t}, the hypothetical no-generalization trajectory was obtained as follows. We first recorded the agent/monkey's sequential actions $(\boldsymbol{a}_0, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_{N-1})$ in the training task ($1\times$ gain, no perturbations) navigating to the same target (for agents) or the closest target in the data set (for monkeys). We then regenerated a new trajectory using $(\boldsymbol{a}_0, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_{N-1})$ following the environmental transition (eq. (1), process noise $\boldsymbol{\eta}_t = 0$), but with the novel gain multiplier n for the gain task or the novel perturbation velocities p_{v_t}, p_{w_t} for the perturbation task.

Under-/over-shooting definition using idealized circular trajectories

To determine when an agent or a monkey under- or over-shot the target in the gain task, we asked whether its stop location exceeded the target location in the distance along their corresponding idealized circular trajectories.

Specifically, given an arbitrary endpoint $[\tilde{x}, \tilde{y}]^\top$, the circular trajectory connecting it from a forward heading (90°, initial head direction) at the origin (start location) has a radius as a function of this point $\tilde{R}(\tilde{x}, \tilde{y})$. The arc length of this trajectory is a function:

$$L(\tilde{x}, \tilde{y}) = 2\tilde{r} \arcsin\left(\frac{\sqrt{\tilde{x}^2 + \tilde{y}^2}}{2\tilde{r}}\right), \qquad \tilde{r} = \tilde{R}(\tilde{x}, \tilde{y}) = \frac{\tilde{x}^2 + \tilde{y}^2}{2\tilde{x}}$$

We deemed the agent's stop location $[s_{x_{N-1}}, s_{y_{N-1}}]^\top$ to have overshot the target $[g_x, g_y]^\top$ if $L(s_{x_{N-1}}, s_{y_{N-1}}) > L(g_x, g_y)$, otherwise it undershot.

Trajectory length and curvature

We approximated the length $\tilde{l}$ and the curvature $\tilde{k}_t$ of a trajectory $(s_{x_t}, s_{y_t})_{t=0,\cdots,N-1}$ as follows:

$$\tilde{l} = \sum_{t=0}^{N-2} \sqrt{(s_{x_{t+1}} - s_{x_t})^2 + (s_{y_{t+1}} - s_{y_t})^2}$$
$$\tilde{k}_t = \frac{|s'_{x_t} s''_{y_t} - s'_{y_t} s''_{x_t}|}{(s'^2_{x_t} + s'^2_{y_t})^{\frac{3}{2}}}$$

where first derivatives s'_{x_t}, s'_{y_t} and second derivatives s''_{x_t}, s''_{y_t} were estimated using first-order one-sided differences for the first and last points and second-order central differences for interior points. Note that the monkeys' trajectories here were downsampled to have the same 0.1 s time step as the agents' trajectories.

Spatial tuning

We obtained the approximate spatial tuning of each neuron by linearly interpolating its activity and the agent's x and y location using data from each step across trials (2000 trials for the gain, and 5000 for the perturbation task), followed by a convolution over the 2D space using a boxcar filter with height and width of 40 cm.

Neural decoding

While agents were being tested, we recorded their sensory, latent, and motor task variables for the analyses in Fig. 1**l** and Fig. S1**d** and their positions s_{x_t}, s_{y_t} for all other decoding analyses. We also recorded their neural activities in each module for both their actors and critics. Let S denote a partitioned matrix where rows are time steps, and columns are decoding target variables, e.g., $[\boldsymbol{s}_x, \boldsymbol{s}_y]$ for agent's positions. Recorded neural activities X were concatenated over time, where rows are time steps and columns are units. A linear decoder regressed S on X, whose partitioned parameters for all decoding variables W were obtained by the ridge estimator following

$$W = (X^\top X + \lambda I)^{-1} X^\top S$$

where λ is a penalty term chosen from $\{0.1, 1, 10\}$ by cross-validation. Importantly, we always used 70% trials in the dataset to train the decoder, and used the remaining 30% trials to test the decoder's predictions.

The decoding error of the belief in each trial was defined as

$$\frac{1}{N}\sum_{t=0}^{N-1} ||\hat{s}_{x_t} - s_{x_t}, \hat{s}_{y_t} - s_{y_t}||_2$$

where $\hat{s}_{x_t}, \hat{s}_{y_t}$ are predicted x and y positions.

Statistics

All agents were trained with 8 different random seeds, which determined the initialized neural network parameters and random variables in training (e.g., process and observation noises, agent's initial state, exploration noise, and sampling from the buffer). All analyses for agents did not exclude any data, and included data from training runs with all random seeds unless otherwise noted. We reported mean, SD, or SEM throughout the paper. All correlations were quantified by Pearson's r.

In all violin plots, we determined upper and lower whiskers following $q_1 - \text{whis} \cdot (q_3 - q_1)$ and $q_1 + \text{whis} \cdot (q_3 - q_1)$, where q_1, q_3 are the first and third quartiles, and whis = 1.5 [48]. We did not plot outliers beyond the whisker range for better visualization, but we did not exclude them in quantification.

Author contributions

Conceptualization and methodology: R.Z., X.P., and D.E.A.; formal modeling and analysis: R.Z.; original draft: R.Z.; review and editing: R.Z., X.P., and D.E.A.; funding acquisition: X.P. and D.E.A.; supervision: X.P. and D.E.A.

Acknowledgements

We express our greatest appreciation to Kaushik Lakshminarasimhan for useful discussions. This work was supported by National Institutes of Health grants U19 NS118246 and R01 NS120407.

Code availability

All training and analysis codes are available on GitHub `https://github.com/ryzhang1/Inductive_bias`.

Data availability

All data used in this work are available from the corresponding author upon request.

References

[1] David Hume. *An enquiry concerning human understanding*. Routledge, 2016.

[2] Timothy EJ Behrens, Timothy H Muller, James CR Whittington, Shirley Mark, Alon B Baram, Kimberly L Stachenfeld, and Zeb Kurth-Nelson. What is a cognitive map? organizing knowledge for flexible behavior. *Neuron*, 100(2):490–509, 2018.

[3] Fabian H Sinz, Xaq Pitkow, Jacob Reimer, Matthias Bethge, and Andreas S Tolias. Engineering a less artificial intelligence. *Neuron*, 103(6):967–979, 2019.

[4] Anirudh Goyal and Yoshua Bengio. Inductive biases for deep learning of higher-level cognition. *Proceedings of the Royal Society A*, 2022.

[5] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[6] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[7] Maxwell A Bertolero, BT Thomas Yeo, and Mark D'Esposito. The modular and integrative functional architecture of the human brain. *Proceedings of the National Academy of Sciences*, 112(49):E6798–E6807, 2015.

[8] Sarah Genon, Andrew Reid, Robert Langner, Katrin Amunts, and Simon B Eickhoff. How to characterize the function of a brain region. *Trends in cognitive sciences*, 22(4):350–364, 2018.

[9] David Meunier, Renaud Lambiotte, Alex Fornito, Karen Ersche, and Edward T Bullmore. Hierarchical modularity in human brain functional networks. *Frontiers in neuroinformatics*, 3:37, 2009.

[10] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[11] Kaushik J Lakshminarasimhan, Eric Avila, Erin Neyhart, Gregory C DeAngelis, Xaq Pitkow, and Dora E Angelaki. Tracking the mind's eye: Primate gaze behavior during virtual visuomotor navigation reflects belief dynamics. *Neuron*, 106(4):662–674, 2020.

[12] Kaushik J Lakshminarasimhan, Eric Avila, Xaq Pitkow, and Dora E Angelaki. Dynamical latent state computation in the posterior parietal cortex. *bioRxiv*, 2022.

[13] Mitsuko Watabe-Uchida, Neir Eshel, and Naoshige Uchida. Neural circuitry of reward prediction error. *Annual review of neuroscience*, 40:373, 2017.

[14] Jean-Paul Noel, Baptiste Caziot, Stefania Bruni, Nora E Fitzgerald, Eric Avila, and Dora E Angelaki. Supporting generalization in non-human primate behavior by tapping into structural knowledge: Examples from sensorimotor mappings, inference, and decision-making. *Progress in Neurobiology*, 201:101996, 2021.

[15] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

[16] Alexandre Pouget, Jeffrey M Beck, Wei Ji Ma, and Peter E Latham. Probabilistic brains: knowns and unknowns. *Nature neuroscience*, 16(9):1170–1178, 2013.

[17] Panos Alefantis, Kaushik J Lakshminarasimhan, Eric Avila, Jean-Paul Noel, Xaq Pitkow, and Dora E Angelaki. Sensory evidence accumulation using optic flow in a naturalistic navigation task. *Journal of Neuroscience*, 2022.

[18] Demis Hassabis, Dharshan Kumaran, Christopher Summerfield, and Matthew Botvinick. Neuroscience-inspired artificial intelligence. *Neuron*, 95(2):245–258, 2017.

[19] Matthew Botvinick, Jane X Wang, Will Dabney, Kevin J Miller, and Zeb Kurth-Nelson. Deep reinforcement learning and its neuroscientific implications. *Neuron*, 107(4):603–616, 2020.

[20] Jean-Paul Noel, Edoardo Balzani, Eric Avila, Kaushik Janakiraman Lakshminarasimhan, Stefania Bruni, Panos Alefantis, Cristina Savin, and Dora E Angelaki. Coding of latent variables in sensory, parietal, and frontal cortices during closed-loop virtual navigation. *Elife*, 11:e80280, 2022.

[21] Edoardo Balzani, Kaushik Lakshminarasimhan, Dora Angelaki, and Cristina Savin. Efficient estimation of neural tuning during naturalistic behavior. *Advances in Neural Information Processing Systems*, 33:12604–12614, 2020.

[22] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[23] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[24] Simon J Julier and Jeffrey K Uhlmann. New extension of the kalman filter to nonlinear systems. In *Signal processing, sensor fusion, and target recognition VI*, volume 3068, pages 182–193. Spie, 1997.

[25] Yael Niv. Learning task-state representations. *Nature neuroscience*, 22(10):1544–1553, 2019.

[26] Robert C Wilson, Yuji K Takahashi, Geoffrey Schoenbaum, and Yael Niv. Orbitofrontal cortex as a cognitive map of task space. *Neuron*, 81(2):267–279, 2014.

[27] Nicolas W Schuck, Robert Wilson, and Yael Niv. A state representation for reinforcement learning and decision-making in the orbitofrontal cortex. In *Goal-directed decision making*, pages 259–278. Elsevier, 2018.

[28] Paul W Glimcher. Understanding dopamine and reinforcement learning: the dopamine reward prediction error hypothesis. *Proceedings of the National Academy of Sciences*, 108(Supplement 3):15647–15654, 2011.

[29] Tiago V Maia and Michael J Frank. From reinforcement learning models to psychiatric and neurological disorders. *Nature neuroscience*, 14(2):154–162, 2011.

[30] Jonathan W Mink. The basal ganglia: focused selection and inhibition of competing motor programs. *Progress in neurobiology*, 50(4):381–425, 1996.

[31] Michael J Frank. Dynamic dopamine modulation in the basal ganglia: a neurocomputational account of cognitive deficits in medicated and nonmedicated parkinsonism. *Journal of cognitive neuroscience*, 17(1):51–72, 2005.

[32] Kazuyuki Samejima, Yasumasa Ueda, Kenji Doya, and Minoru Kimura. Representation of action-specific reward values in the striatum. *Science*, 310(5752):1337–1340, 2005.

[33] Anitha Pasupathy and Earl K Miller. Different time courses of learning-related activity in the prefrontal cortex and striatum. *Nature*, 433(7028):873–876, 2005.

[34] Kimberly L Stachenfeld, Matthew M Botvinick, and Samuel J Gershman. The hippocampus as a predictive map. *Nature neuroscience*, 20(11):1643–1653, 2017.

[35] Andre Barreto, Diana Borsa, John Quan, Tom Schaul, David Silver, Matteo Hessel, Daniel Mankowitz, Augustin Zidek, and Remi Munos. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *International Conference on Machine Learning*, pages 501–510. PMLR, 2018.

[36] Jane X Wang, Zeb Kurth-Nelson, Dharshan Kumaran, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Demis Hassabis, and Matthew Botvinick. Prefrontal cortex as a meta-reinforcement learning system. *Nature neuroscience*, 21(6):860–868, 2018.

[37] Matthew Botvinick, Sam Ritter, Jane X Wang, Zeb Kurth-Nelson, Charles Blundell, and Demis Hassabis. Reinforcement learning, fast and slow. *Trends in cognitive sciences*, 23(5):408–422, 2019.

[38] Bradley B Doll, Dylan A Simon, and Nathaniel D Daw. The ubiquity of model-based reinforcement learning. *Current opinion in neurobiology*, 22(6):1075–1081, 2012.

[39] Nathaniel D Daw, Yael Niv, and Peter Dayan. Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature neuroscience*, 8(12):1704–1711, 2005.

[40] Peter Dayan and Kent C Berridge. Model-based and model-free pavlovian reward learning: revaluation, revision, and revelation. *Cognitive, Affective, & Behavioral Neuroscience*, 14(2):473–492, 2014.

[41] Nancy Kanwisher. Functional specificity in the human brain: a window into the functional architecture of the mind. *Proceedings of the National Academy of Sciences*, 107(25):11163–11170, 2010.

[42] Guangyu Robert Yang, Madhura R Joglekar, H Francis Song, William T Newsome, and Xiao-Jing Wang. Task representations in neural networks trained to perform many cognitive tasks. *Nature neuroscience*, 22(2):297–306, 2019.

[43] Jascha Achterberg, Danyal Akarca, DJ Strouse, John Duncan, and Duncan E Astle. Spatially-embedded recurrent neural networks reveal widespread links between structural and functional neuroscience findings. *bioRxiv*, 2022.

[44] Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, volume 6, 1993.

[45] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[46] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019.

[47] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[48] John W Tukey et al. *Exploratory data analysis*, volume 2. Reading, MA, 1977.

Supplementary figures

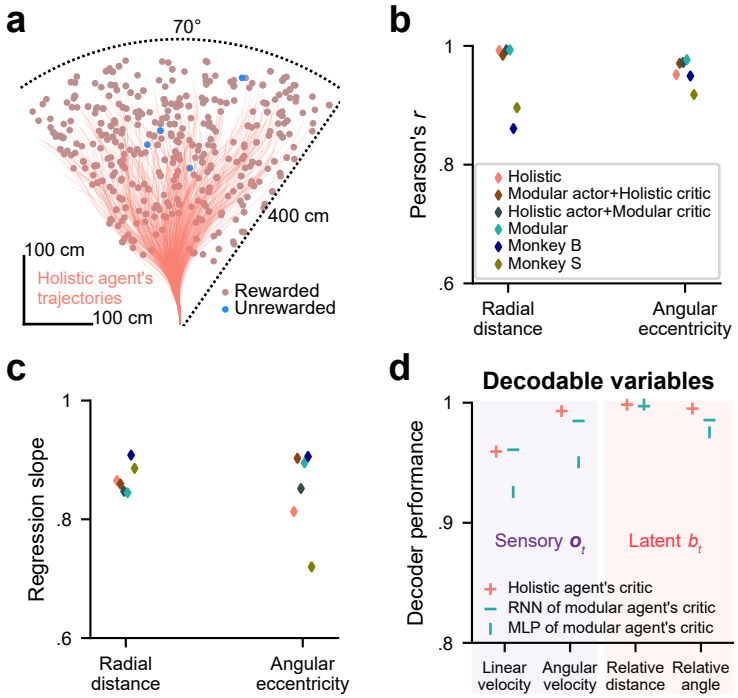

Figure S1: [Related to Fig. 1]. **a**. Similar to Fig. 1**h**, but showing an example holistic agent's trajectories. **b**. Pearson correlation coefficient for agents' and monkeys' stop locations versus target locations after training, for data shown in Fig. 1**i**. **c**. Similar to **b**, but showing regression slopes ($> 1/< 1$: over-/under-shooting). **d**. Similar to Fig. 1**l**, but for modules in critics.

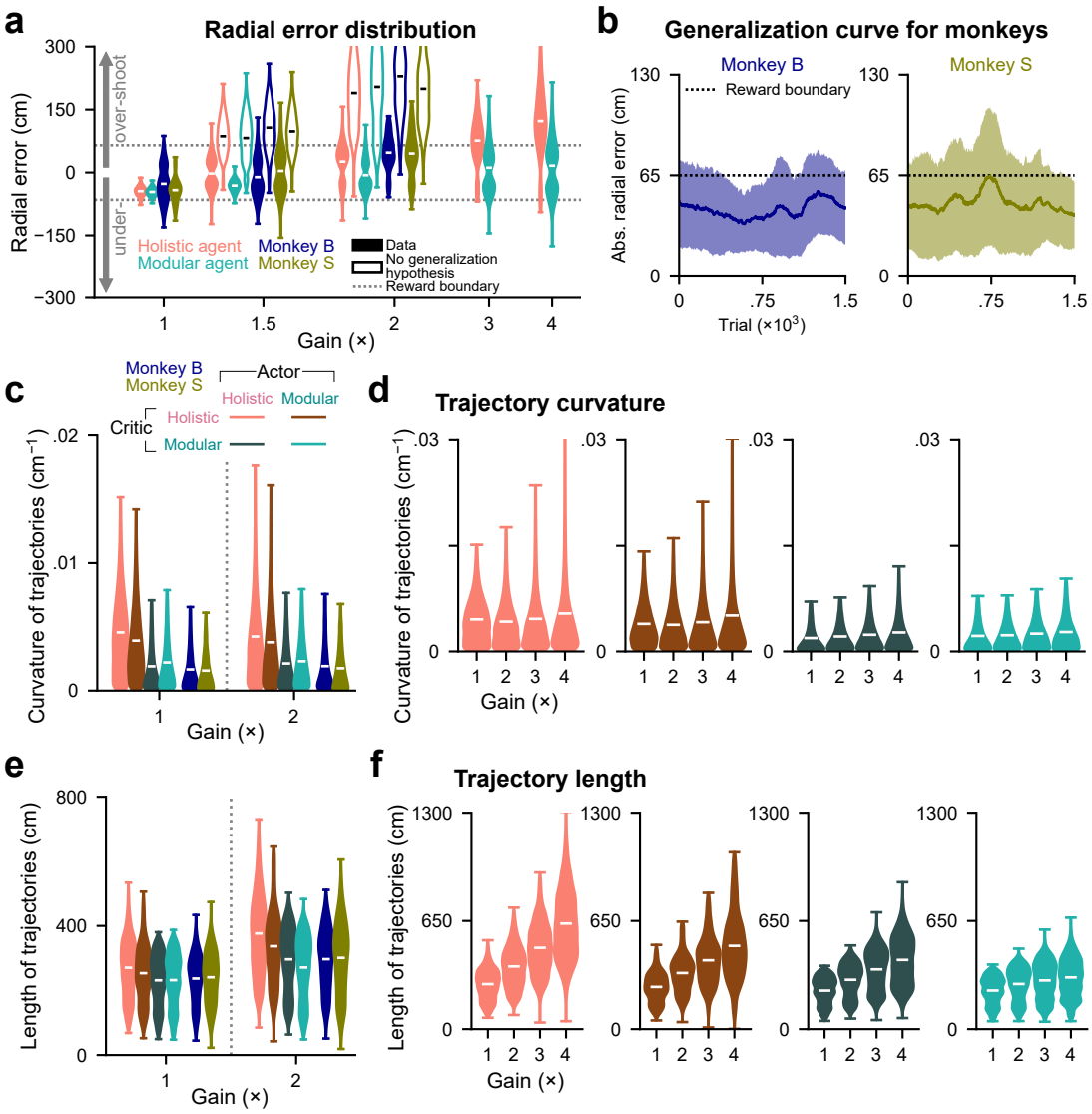

Figure S2: [Related to Fig. 2]. **a**. Similar to Fig. 2**c**–**d**, but showing detailed distributions of agents' and monkeys' radial errors. **b**. Monkeys' absolute radial error as a function of the number of 1.5× gain trials experienced. Solid lines and shaded regions denote means and ±1 SD obtained using a moving window (size= 150 trials). **c**. Curvature of agents' and monkeys' trajectories with 1× and 2× gains. **d**. Curvature of agents' trajectories as a function of gain. **e**–**f**. Similar to **c**–**d**, but for the length of trajectories. **a**,**c**–**f**: Containing data from $n = 8$ random seeds for each agent. White bars denote means across trials.

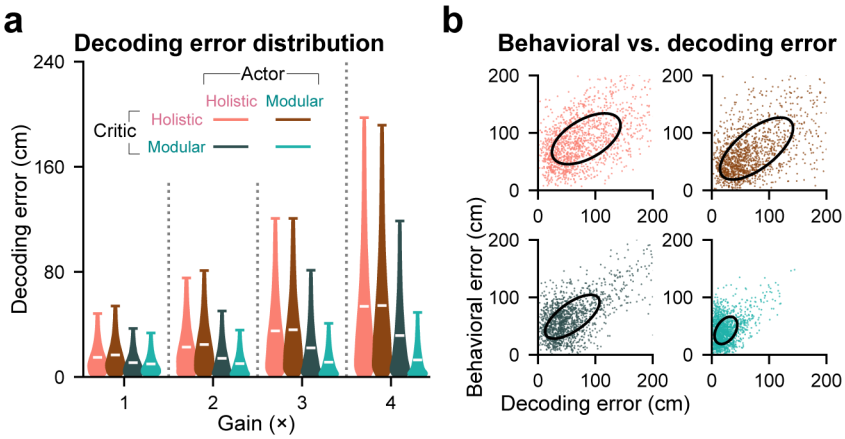

Figure S3: [Related to Fig. 3]. **a**. Similar to Fig. 3**d**, but showing detailed distributions of decoding errors. White bars denote means across trials. **b**. Behavioral error (absolute radial error between the target and the agent's stop location) versus decoding error of stop locations (distance between decoded and true stop location) for example agents in the test set used in Fig. 3**e**. Confidence ellipses capture 1 SD. Pearson's r, Holistic: 0.53, Modular actor+Holistic critic: 0.66, Holistic actor+Modular critic: 0.68, Modular: 0.40.

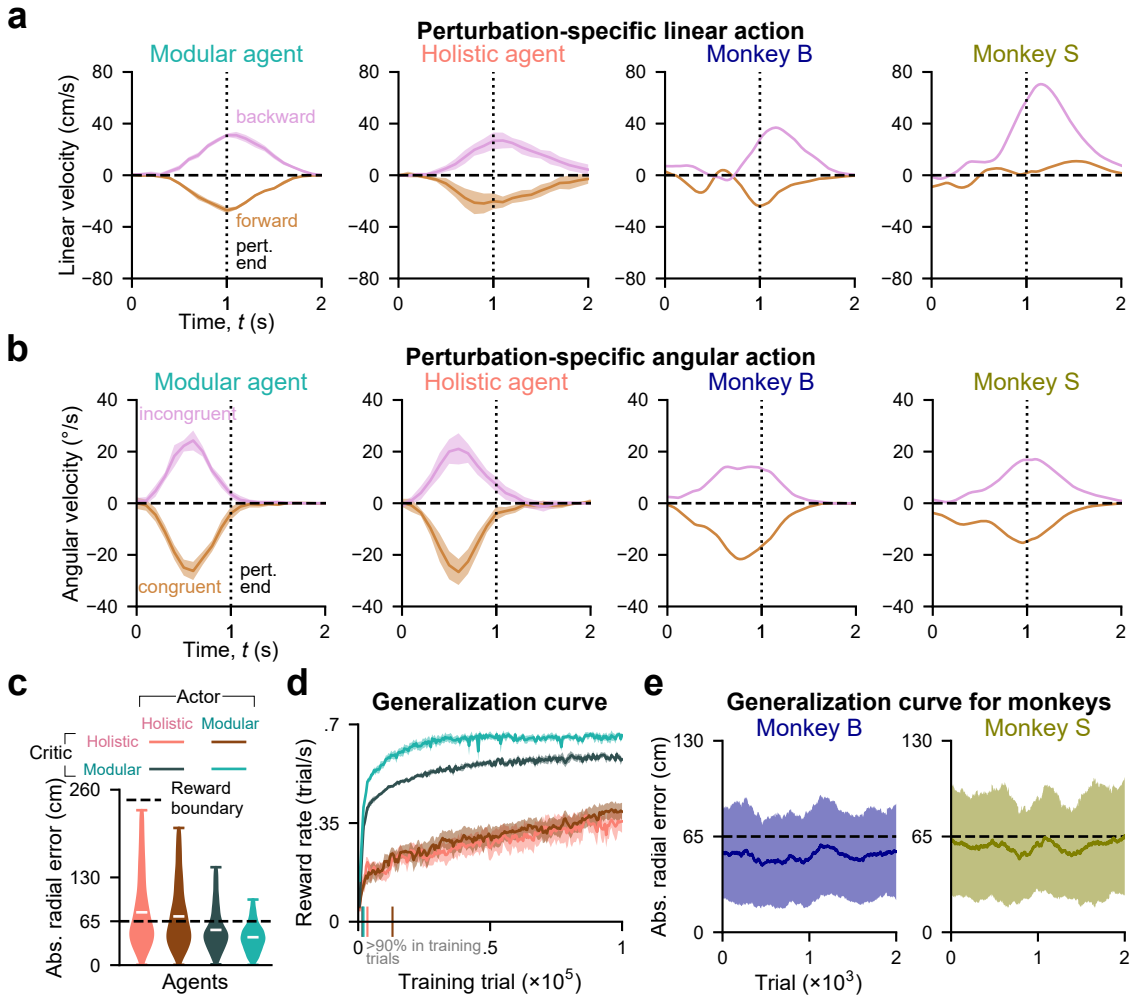

Figure S4: [Related to Fig. 4]. **a**–**b**. Agents' and monkeys' perturbation-specific linear (**a**) and angular (**b**) actions (in units of velocities) in a 2 s time window, averaged across a test set comprising 1000 trials. Task variables (target location, perturbation peak time, and peak of perturbation linear and angular velocities) used in agents' trials are the same as those in monkey B's trials. Trials for monkey S are the most similar trials (measured in Euclidean distance of task variables that were all normalized in $[0, 1]$) in the whole data set to monkey B's trials. Trials are aligned such that perturbations start at $t = 0$ s. The perturbation-specific linear/angular actions are obtained by subtracting linear/angular actions in target-matched unperturbed trials from those in corresponding perturbation trials. Perturbations causing one to get closer to/further from targets are grouped as forward (+)/backward (−) for linear perturbations (**a**) and congruent (+)/incongruent (−) for angular perturbations (**b**). Vertical dotted lines denote the perturbation end time ($t = 1$ s). Horizontal dashed lines denote the null perturbation-specific action. Shaded regions for agents denote ±1 SD across $n = 8$ random seeds. **c**. Distributions of agents' absolute radial error in Fig. 4**d**. White bars denote means across trials. **d**. Agents' reward rate (number of rewarded trials per second) averaged over 2 validation sets as a function of the number of training trials (no perturbations) experienced after training phase I (see definition in Methods). Two validation sets share the same 300 targets, and their perturbation task variables are sampled from the ranges for monkeys and agents in Fig. 4**a**, respectively. Shaded regions denote ±1 SEM across $n = 8$ random seeds. Vertical bars overlaid on the x-axis denote the first time agents reach 90% accuracy in the no perturbation validation set (the same 300 targets). **e**. Monkeys' absolute radial error as a function of the number of perturbation trials experienced. Solid lines and shaded regions denote means and ±1 SD obtained using a moving window (size= 200 trials).

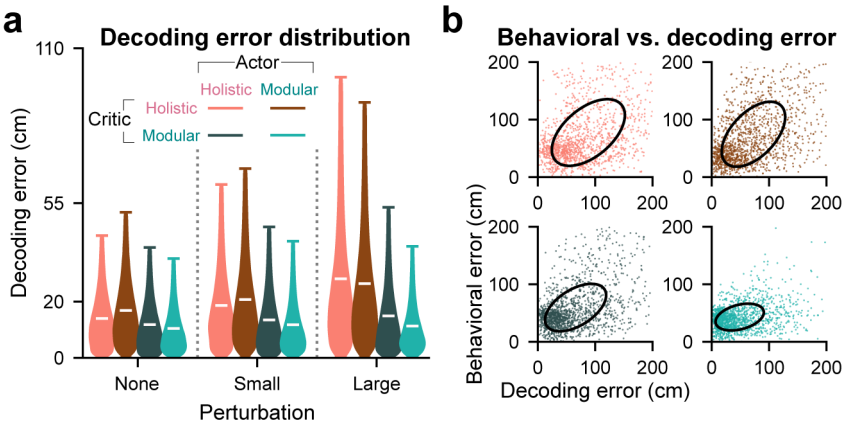

Figure S5: [Related to Fig. 5]. **a**–**b**. Similar to Fig. S3**a**–**b**, but for the perturbation task. **b**. Pearson's r, Holistic: 0.52, Modular actor+Holistic critic: 0.53, Holistic actor+Modular critic: 0.52, Modular: 0.35.

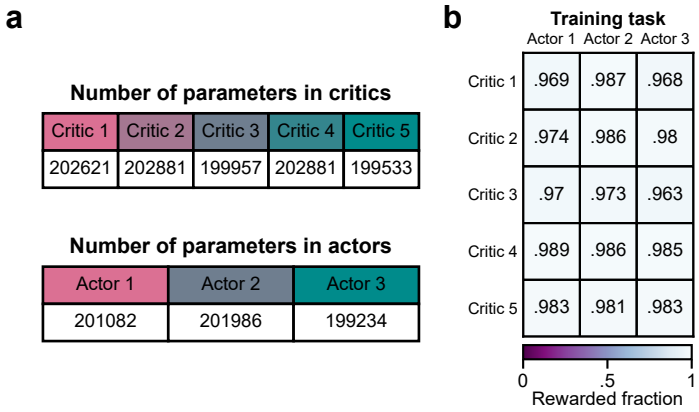

Figure S6: [Related to Fig. 6]. **a**. Total number of neural parameters in critics (Fig. 6**a**) and actors (Fig. 6**b**). **b**. Fraction of rewarded trials in the training task (2000 trials) for agents with all combinations of actors and critics after training, averaged across $n = 8$ random seeds for each agent.

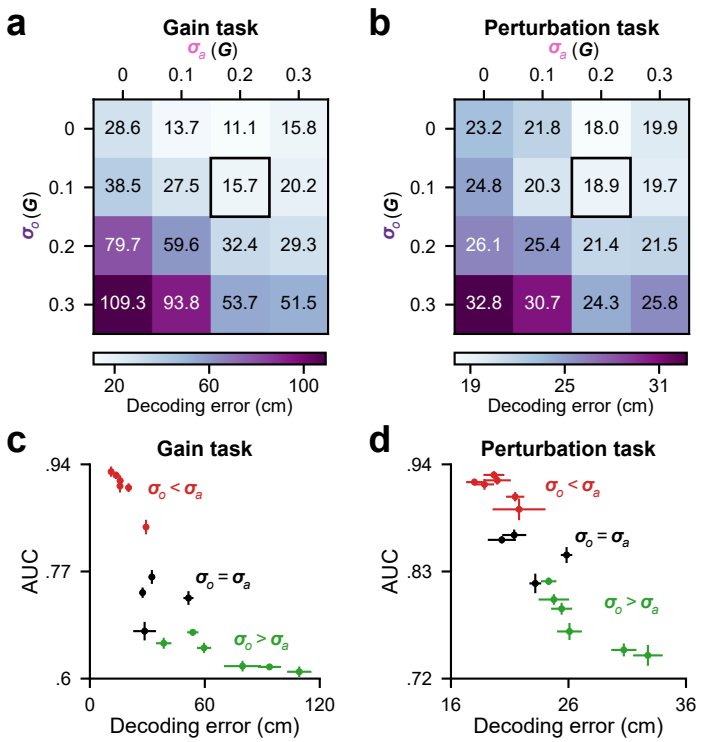

Figure S7: [Related to Fig. 7]. **a**–**b**. Similar to Fig. 7**c**–**d**, but showing the decoding error, averaged across time steps and trials. **c**–**d**. AUC (Fig. 7**c**–**d**) versus decoding error (**a**–**b**) for agents trained with all combinations of $\boldsymbol{\sigma}_a$ and $\boldsymbol{\sigma}_o$, tested in the gain (**c**) and the perturbation (**d**) tasks. Error bars denote ± 1 SEM across $n = 8$ random seeds.