Automatic Discovery of Cognitive Strategies with Tiny Recurrent Neural Networks

Li Ji-An[1], Marcus K. Benna[2], and Marcelo G. Mattar[3, 4, @]

[1]Neurosciences Graduate Program, University of California San Diego, La Jolla, CA 92093
[2]Department of Neurobiology, School of Biological Sciences, University of California San Diego, La Jolla, CA 92093
[3]Department of Cognitive Science, University of California San Diego, La Jolla, CA 92093
[4]Department of Psychology, New York University, New York, NY 10012
[@]corresponding author: marcelo.mattar@nyu.edu

Summary

Normative frameworks such as Bayesian inference and reward-based learning are useful tools for explaining the fundamental principles of adaptive behavior. However, their ability to describe realistic animal behavior is limited by the often small number of parameters that are fit to data, leading to a cycle of handcrafted adjustments and model comparison procedures that are often subject to research subjectivity. Here, we present a novel modeling approach leveraging Recurrent Neural Networks (RNNs) to automatically discover the cognitive algorithms governing biological decision-making. We demonstrate that RNNs with only one or two units can predict individual animals' choices more accurately than classical normative models, and as accurately as larger neural networks, in three well-studied reward learning tasks. Using tools from discrete dynamical systems theory, such as state-space and fixed point attractors, we show that trained networks uncovered numerous cognitive strategies overlooked in classical cognitive models and task-optimized neural networks. Our approach also allows for a unified comparison of different models and provides insights into the dimensionality of behavior and the emergence of meta-learning algorithms. Overall, we offer a systematic approach for the automatic discovery of interpretable cognitive strategies underlying decision-making, shedding light onto neural mechanisms and providing novel insights into healthy and dysfunctional cognition.

Introduction

A fundamental goal in neural and cognitive sciences is to understand how neural activity gives rise to adaptive behavior. Historically, researchers have sought to capture the intricacies of learning and decision-making using computational models, ranging from early symbolic models[1] to connectionist approaches[2]. More recently, normative frameworks such as Bayesian inference[3,4] and reward-based learning[5–8] have gained prominence due to their ability to elucidate the fundamental principles governing adaptive behavior. These cognitive models formalize how agents accumulate and apply knowledge from environmental interactions for decision-making, a process carried out by neural circuits in the prefrontal cortex and striatum[9–13]. A key advantage of normative models is that they typically have few parameters and can be easily augmented by additional assumptions such as forgetting, choice biases, choice perseveration, exploration, and capacity limitations[12,14]. However, this simplicity and extensibility also make these models prone to bias and researcher subjectivity, potentially leading to incorrect or incomplete characterizations of behavior[15].

An alternative modeling paradigm employs artificial neural networks (ANNs), which impose fewer structural assumptions, require less handcrafting, and provide a more flexible and biologically plausible framework for modeling behavior and neural activity[16,17]. Network parameters are typically adjusted to produce optimal behavior in a given task, in line with the connectionist tradition. This approach has been successful in describing the emergence of hierarchical representations in the visual cortex[18,19], predictive representations in the hippocampus and entorhinal cortex[20–22], and neural activity underlying learning and decision-making in the prefrontal cortex[23–26]. Neural networks can alternatively be fitted to human and animal behavior[27–31]. However, the flexibility enabled by a large number of parameters in ANNs poses challenges, such as requiring extensive data for training and limited interpretability, hindering the discovery of underlying cognitive strategies and neural mechanisms.

In this study, we present a novel approach combining the flexibility of ANNs with the simplicity and interpretability of classical normative models. Our method consists of fitting tiny recurrent neural networks (RNNs), composed of only one or two units, to the behavior of individual subjects in reward learning tasks. The fitted models succinctly describe how agents accumulate knowledge from environmental interactions and apply it during decision-making. Across three datasets involving three animal species, we demonstrate that these models provide a superior fit to animal behavior than classical

cognitive models of equal dimensionality (i.e., number of dynamical variables) and perform no worse than larger networks. Utilizing discrete dynamical systems theory concepts, such as state-space and fixed point attractors, we offer a unified comparison of different models, systematically investigating the emerging cognitive strategies and neural mechanisms required to account for the observed decision-making behavior. This approach discovers novel behavioral patterns and cognitive strategies overlooked by classical cognitive models in all three reward learning tasks, including variable learning rates, a novel confirmation effect, and previously unrecognized forms of value forgetting, choice perseveration, and choice biases. We propose that behavioral modeling based on tiny RNNs addresses both the interpretability challenges of large ANNs and the subjectivity of classical cognitive models. These results hold significant implications for the fields of computational neuroscience, cognitive science, and computational psychiatry, with the potential to transform our understanding of adaptive decision-making mechanisms and to guide future research directions.

Results

Reward learning tasks

We studied animal behavior in three reward learning tasks featuring increasing levels of complexity: a reversal learning task[11], a two-stage task[13], and a novel two-stage task[12]. These tasks have been widely utilized in neuroscience and psychology studies to examine how subjects adapt their behavior to changing state and reward contingencies (see Fig. 1a). The three tasks share a common structure: in each trial, the subject chooses between action A_1 and A_2, resulting in second-stage states S_1 or S_2, each associated with a distinct probability of receiving a unit reward. In neutral trials, both second-stage states offer the same reward probability (e.g., 0.5); in non-neutral trials, one state (e.g., S_1) offers a higher reward probability (e.g., 0.8) than the other state (e.g., 0.2 for S_2). Reward probabilities remain constant for several trials before randomly switching, without any cues, their assignments to second-stage states (a moment termed "reversal"). In all tasks, the subject's objective is to choose the action currently offering the highest likelihood of yielding a reward.

The three studied tasks also have unique characteristics. In the reversal learning task, each action leads deterministically to one state (e.g., A_1 to S_1 and A_2 to S_2). In the two-stage task, each action leads to one state with high probability (e.g., A_1 leads to S_1 with probability 0.8, a "common" transition) and to another state with low probability (e.g., A_1 leads to S_2 with probability 0.2, a "rare" transition). In the novel two-stage task, actions similarly lead to one state with high probability and to another with low probability, but the state associated with high/low probability also switches at other random times (e.g., A_1 leads to S_1/S_2 with high/low probability for a few trials, and after a switch, A_1 leads to S_1/S_2 with low/high probability).

We analyzed data from two monkeys (Bartolo dataset,[11]) and ten mice (Akam dataset,[12]) performing the reversal learning task; from four rats (Miller dataset,[13]) and ten mice (Akam dataset,[12]) performing the two-stage task; and from seventeen mice (Akam dataset,[12]) performing the novel two-stage task.

Computational models of decision-making

We used two types of RNN models to analyze animals' choices. As a benchmark, we also implemented several classical cognitive models commonly used to describe behavior in reward learning tasks, including Bayesian learning models, RL models, and variants. All cognitive models and RNNs share the same input-output structure. The inputs consist of the previous action a_{t-1}, the previous second-stage state s_{t-1}, and the previous reward r_{t-1} (note that the current state is not specified as an input because it is always the same state, called "choice state"). These inputs update the agent's internal state, which in turn determines the agent's output — a probability distribution over actions, or "policy" (Fig. 1). The agent's internal state is described by a set of d internal dynamical variables that summarize the agent's prior experience (e.g., action values, belief states, etc.). Although all models share the same input-output structure, they differ from one another with respect to the rule specifying how internal dynamical variables are updated given inputs, which in turn leads to different policies.

The classical cognitive models considered here can be classified into three model families: model-free, model-based, and latent-state. The model-free family uses reward input to directly update state or action values — the model's dynamical variables. The model-based family uses a state-transition model (either learned or assumed) to update state or action values. In both model-free and model-based families, these dynamical variables are updated via RL algorithms. The latent-state model family, in contrast, uses Bayesian inference to update a dynamical variable, which is a belief probability specifying which second-stage state has a higher reward probability. Other dynamical variables sometimes used in these cognitive models include multi-trial choice perseveration, learned state-transition probabilities, among others (see Methods). As mentioned previously, each cognitive model that we considered instantiates a different rule for updating its dynamical variables (e.g., learning rules in RL, or inference rules in Bayesian models), which is typically described by a few parameters.

The RNN models we proposed use gated recurrent units (GRUs) in their hidden layers. GRUs have gating mechanisms that allow input information to be selectively integrated into the network's state[32]. By adjusting the network parameters,

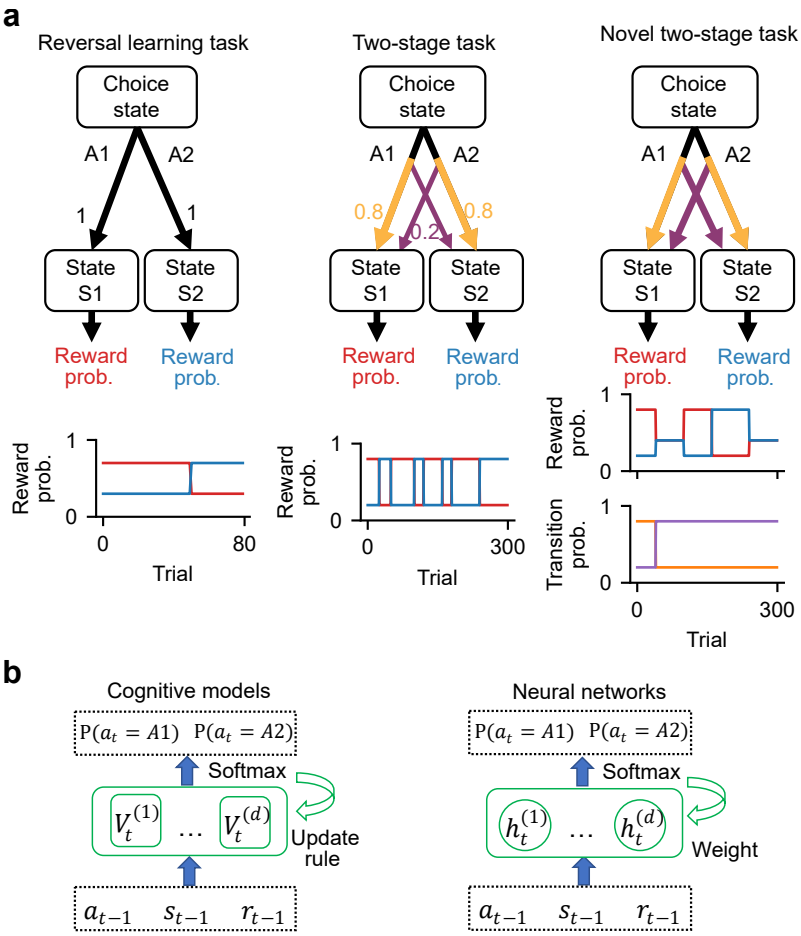

Fig. 1. Task structures and model architectures. (a) In each trial of the three tasks, subjects choose action A_1 or A_2 at the choice state, transitioning into one of two second-stage states, S_1 or S_2, which then probabilistically yield a reward. In the reversal learning task, action A_i is always followed by the corresponding state S_i, with one state having a higher reward probability than the other, switching at random moments in the experiment. In the two-stage task, the reward probabilities and reversals are similar to the reversal learning task, but action A_1 (A_2) stochastically transitions to state S_1 (S_2) with high probability (common transition) and to state S_2 (S_1) with low probability (rare transition). In the novel two-stage task, the reward structure mirrors the other tasks, but the state transitions also switch between two rules: common transitions from A_1 to S_1 and from A_2 to S_2, or common transitions from A_1 to S_2 and from A_2 to S_1. (b) Cognitive models and neural networks have similar architectures. Inputs consist of the previous action a_{t-1}, the previous second-stage state s_{t-1}, and the previous reward r_{t-1}; the model's internal dynamical variables ($V_t^{(1)}, \cdots, V_t^{(d)}$ or $h_t^{(1)}, \cdots, h_t^{(d)}$) are updated based on the inputs and on their previous values; the model's outputs are action probabilities computed from the dynamical variables via the softmax function. The dynamical variables in the cognitive models can represent state values, action values, state-transition probability, latent-state probability, choice perseveration, among others. Each model specifies a different rule for updating the dynamical variables. The dynamical variables in the neural networks are the activities of each network unit.

these models can learn a wide range of functions mapping inputs (observations) into outputs (policies). The number of hidden layer units d determines the dimensionality of the network's state-space. Crucially, the hidden layer activations fully specify the network's internal state and policy, rendering the system Markovian; this is in contrast to alternative RNN architectures such as the long short-term memory[33], where the use of a cell state renders the system non-Markovian. In addition to standard GRUs, we propose a novel RNN model that uses a GRU variant called "switching GRU" (SGRU). In these RNNs, recurrent weights and biases are input-dependent, similar to switching linear dynamical systems[34]. Because the inputs in the three tasks we studied are discrete, SGRUs are a generalization of GRUs (i.e., a GRU can be viewed as an SGRU whose recurrent weights do not vary with the input).

Tiny RNNs outperform classical cognitive models in predicting choices

To analyze the experimental data, we optimized the parameters in each cognitive model and RNN to predict subject choices with maximum likelihood (equivalent to minimizing cross-entropy). Given that the RNN models have substantially more free parameters than the classical cognitive models (see Fig. S1), we used a nested cross-validation procedure where models were optimized on a training dataset, selected on a validation dataset, and evaluated on an unseen test dataset; this procedure ensures a fair comparison between models with different numbers of parameters (see Methods). We then averaged the predictive performance of each model across subjects, yielding a predictive performance for each model in each dataset: monkeys (Bartolo dataset, Fig. 2a) and mice (Akam dataset, Fig. 2b) performing the reversal learning task; rats (Miller dataset, Fig. 2c) and mice (Akam dataset, Fig. 2d) performing the two-stage task; and mice (Akam dataset, Fig. 2e) performing the novel two-stage task. Our primary focus was on comparing models with an equal number of dynamical variables d, as these models use an equivalent number d of functions of the past to predict future behavior.

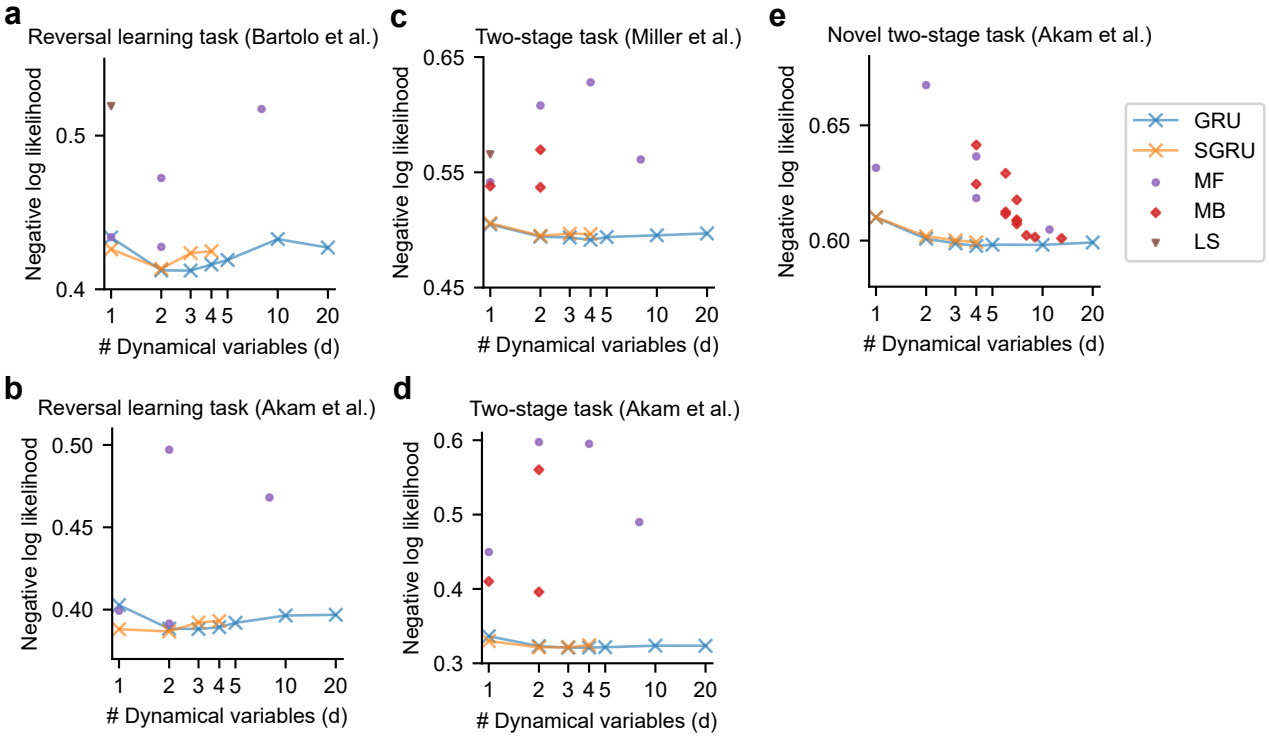

Fig. 2. Tiny RNNs outperform classical cognitive models in predicting animals' choices. The panels display each model's predictive performance (negative log-likelihood) and the corresponding number of dynamical variables d used in that model. (a) In the reversal learning task with monkeys (Bartolo et al.), the GRU model with $d = 2$ outperforms all other models. The SGRU model with $d = 1$ outperforms other models with $d = 1$. (b) In the reversal learning task with mice (Akam et al.), the SGRU model with $d = 2$ outperforms all other models. The SGRU model with $d = 1$ outperforms other models with $d = 1$. (c) In the two-stage task with rats (Miller et al.), the GRU/SGRU model with $d = 2$ outperforms all other models. The GRU/SGRU models with $d = 1$ outperform models with $d = 1$. (d) In the two-stage task with mice (Akam et al.), the GRU/SGRU model with $d = 2$ outperforms all other models. The GRU/SGRU models with $d = 1$ outperform other models with $d = 1$. (e) In the novel two-stage task with mice, the GRU model with $d = 4$ outperforms all other models. The GRU/SGRU models with $d = 1$ or $d = 2$ outperform other models with the same d. Abbreviations: MF, model-free family; MB, model-based family; LS, latent-state family. Each model family may include multiple models, leading to two or more identical markers for a given d.

We found that very small RNNs were generally better at predicting animals' choices than classical cognitive models such as those from model-free, model-based, and Bayesian latent-state families (see Methods). At the group level (i.e., model's predictive performance averaged across subjects), RNNs with 2 units were best at predicting choices in the reversal learning and two-stage tasks, and RNNs with 4 units were best at predicting choices in the novel two-stage task. These networks also achieved comparable predictive performance to larger RNNs (e.g., RNNs with 10 or 20 units, as used in[28]), possibly due to the simplicity of the behavior expressed by animals in these tasks. Furthermore, the SGRU model with 1 unit consistently outperformed all other single-variable models, and RNNs with d units similarly outperformed all classical cognitive models with d variables. At the individual level, the behavior of most animals was best explained by RNNs with 1-2 units in the reversal learning task, 2 units in the two-stage task, and 2-4 units in the novel two-stage task (see a detailed description of each dataset in Supplementary Results and Fig. S2). These findings indicate that tiny RNNs serve as versatile models capable of elucidating behavioral patterns not typically captured by classical cognitive models or their variants (e.g., models with forgetting, perseveration, etc.).

A noteworthy observation is that incorporating more dynamical variables did not always yield superior predictive performance, and at some times performance even declined. The number of dynamical variables for which predictive performance is maximized provides an estimation of the dimensionality of behavior d_*, defined as the minimal number of functions of the past required to accurately predict future behavior as well as possible[35,36]. In RNN models, if the number of dynamical variables is lower than d_*, including more variables in the model should increase predictive performance up to d_* dynamical variables. Using more variables than d_* should not improve performance significantly and may even hurt it due to the lack of training data (see "Nested cross-validation procedure" in Methods). Reinterpreting the previous results in terms of dimensionality of behavior, we find that the behavior of most animals in the reversal learning and two-stage tasks has dimensionality d_* of 1 or 2, while the behavior of most animals in the novel two-stage task has dimensionality d_* ranging between 2 and 4. We note that low-dimensional tasks — such as the three tasks studied here, as well as the vast majority of laboratory tasks used in neuroscience and psychology studies — are particularly well-suited for RNN models with few dynamical variables.

Tiny RNNs achieve better performance with larger training datasets

The superior performance of tiny RNNs compared to classical cognitive models can be attributed to the increased flexibility provided by a greater number of free parameters. This flexibility allows a single RNN to be molded into a wider range of behaviors than possible with classical cognitive models. However, this flexibility also raises important concerns such as the risk of overfitting and the requirement for large amounts of training data. While the use of nested cross-validation and early stopping effectively eliminated the risk of overfitting (see next section for empirical evidence), we wished to determine how much data was needed for training the RNNs in comparison to the classical cognitive models.

We found that RNN models fitted to individual subject's only outperformed cognitive models of equal dimensionality if a sufficient number of training trials were available: over approximately 2,000 trials for the reversal learning task (Fig. 3a), 500 trials for the two-stage task (Fig. 3b), and 1,500 trials for the novel two-stage task (Fig. 3c). Importantly, these dataset sizes fall within the range of most neuroscience datasets. Interestingly, RNNs' predictive performance continued to improve with increasing training data, in contrast to classical cognitive models. For instance, in the reversal learning task, the performance of the 2-unit RNNs improved when the number of training trials increased from 2,000 to 4,000, and the absence of an asymptotic behavior suggests further improvement would be possible with even larger datasets. Similar trends were observed for the 2-unit RNNs in the two-stage task and for the 4-unit RNNs in the novel two-stage task, indicating that RNN models can capitalize on additional data due to their inherent flexibility.

Tiny RNNs identify ground-truth strategies in simulated behavior

The results thus far suggest that animals exhibit behavioral patterns that are better captured by flexible RNN models than by cognitive models designed from normative considerations. What if animals exhibited a different behavioral pattern other than observed in these datasets, such as those exhibited by an RL or Bayesian inference agent? One possibility is that RNN models can only approximate, but not fully account for, the behavior of such agents. In this scenario, a cognitive model well-matched to the animal's behavior would outperform the RNN models. Alternatively, the RNN models might be flexible enough to fully account for the behavior of an RL or Bayesian inference agent. In this scenario, the RNN models may be viewed as a superset of the cognitive models, and their predictive performance would be no worse than that of any cognitive model.

To determine how well can an RNN model predict the behavior of RL and Bayesian inference agents, we simulated the behavior of such agents over 10,000 choices in the two-stage task, with parameters fitted to one example rat from the Miller dataset. By design, thus, these choices embodied the behavior of a normative RL or Bayesian inference agent with one or two dynamical variables. We then fitted all cognitive models and RNN models to optimize predictive performance for this simulated behavior. We found that the best RNN models (either GRU or SGRU) achieved identical predictive performance as

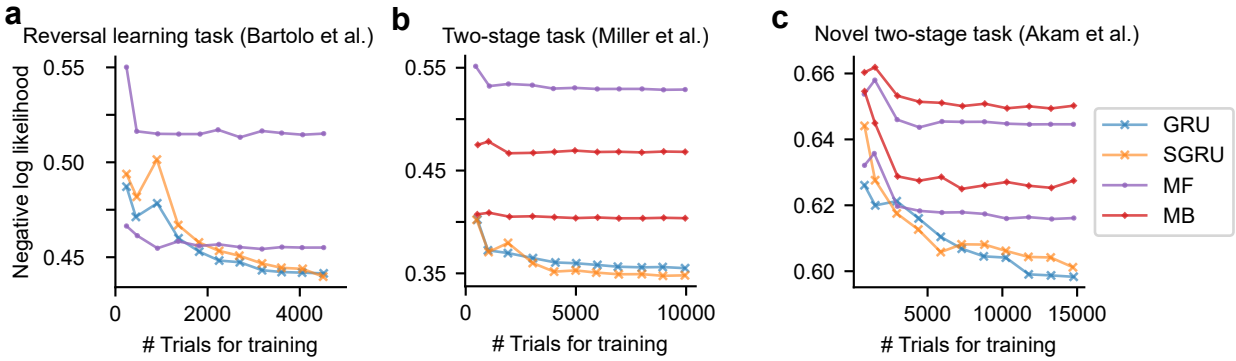

Fig. 3. The predictive performance of tiny RNNs improves as the training dataset size increases. The panels display each model's predictive performance (negative log-likelihood) for different training dataset sizes (number of training trials). (a) Two-dimensional models applied to reversal learning task data from a representative monkey (Bartolo et al.). (b) Two-dimensional models applied to two-stage task data from a representative rat (Miller et al.). (c) Four-dimensional models applied to novel two-stage task data from a representative mouse (Akam et al.). Abbreviations: MF, model-free family; MB, model-based family; LS, latent-state family. Each model family may include multiple models, leading to two or more identical markers.

the ground-truth model of equal dimensionality that generated the behavior (Fig. 4). Furthermore, these RNNs accurately recovered the ground-truth model's dynamics (see next section and Fig.S3). This demonstrates the remarkable flexibility of RNN models, which can describe diverse behavioral strategies and serve as a superset of classical cognitive models, despite using a consistent architecture that requires virtually no manual engineering.

Incidentally, these results also demonstrate that the nested cross-validation procedure employed in all of our results successfully prevented overfitting, which would have been diagnosed if the more flexible RNNs had achieved higher performance on the training dataset (and lower on the test dataset) than the data-generating model.

Leveraging dynamical systems theory for model interpretation and comparison

A significant barrier to the widespread use of ANNs in cognitive modeling is their limited interpretability. The usually small size of our RNNs, however, enables the interpretation of the network dynamics in terms of cognitive processes. Our interpretative framework is grounded in the theory of discrete dynamical systems[37], which uses difference equations to describe how state variables change over time as a function of the inputs. We adopt this framework to analyze and compare the behavior of cognitive models and RNNs.

We first provide a dynamical systems interpretation for one-dimensional cognitive models — a model-free RL and a latent-state model. To describe the state and the evolution of the system, we defined the logit (log-odds) at trial t as $L(t) = \log\left(\Pr_t(A_1)/\Pr_t(A_2)\right)$ and the logit-change as the difference between logits in consecutive trials $\delta L(t) = L(t+1) - L(t)$. In the cognitive models, the logit represents the agent's preference for one action over the other. Viewed as a dynamical system, logit and logit-change can be respectively understood as the "position" and "velocity" of the system. After fitting both models to the choices of one monkey in the reversal learning task, we plotted logit-change δL against logit L over trials, grouped according to the four input possibilities I (combinations of actions and rewards) — understood as the "force" acting on the system (Fig. 5a). These plots, called phase portraits, completely describe the dynamical system and reveal important characteristics (see mathematical details in "Phase portraits" in Methods), such as fixed points (the x-intercept L_*, representing states where logit-change δL is zero) and learning rates (related to the slopes at each position L, or $d\delta L/dL$).

The phase portraits of model-free RL and latent-state models reveal similarities and differences in how these models process input and handle uncertainty. In both models, repeated selection of a given action followed by reward shifts the system towards a fixed point corresponding to a strong preference for that action (dark blue and dark grey lines in Fig. 5a). The straight vs. curved shape of the logit-change plots highlight the distinct ways in which the models process input that disagrees with current beliefs: if an agent selects a strongly disfavored action and receives a reward, a model-free RL agent will experience a large state change (because of a large prediction error), while a latent-state agent will experience only a small state change (because of a strong prior at extreme logit values). The models also behave differently when receiving no reward: while a model-free agent moves towards an indifference state (note fixed point at $L_* = 0$), the latent-state model shifts towards a preference for the unchosen action, as if a reward had been received there.

Crucially, the dynamical systems approach can also be applied to RNNs to provide a detailed comparison with cognitive models. Analyzing a one-unit SGRU fitted to data from the reversal learning task, we found that unrewarded trials moved the system towards an indifference state, and that receiving a reward for a strongly disfavored action resulted

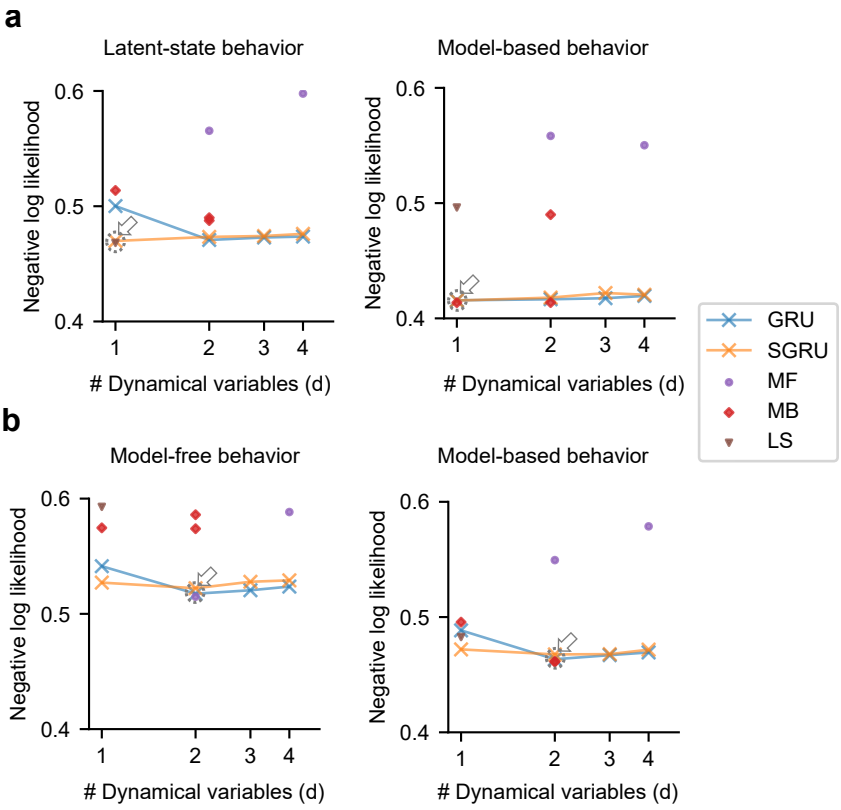

Fig. 4. Tiny RNNs can predict choices as well as the ground-truth model that generated those choices. The panels display each model's performance (negative log-likelihood) in predicting the choices generated by a cognitive model in the two-stage task, with randomness in the choices sampled by the agent's policy. (a) Simulated behavior generated by a latent-state agent with $d = 1$ and a model-based RL agent with $d = 1$. (b) Simulated behavior generated by a model-free (Q(1)) agent with $d = 2$ and a model-based RL agent with $d = 2$ (without value forgetting). Abbreviations: MF, model-free family; MB, model-based family; LS, latent-state family. Each model family may include multiple models, leading to two or more identical markers for a given d. Arrows indicate the ground-truth cognitive model that generated the fitted data.

in a large change in preferences. Additionally, we found no evidence that the animals interpreted an unrewarded action as indicating a reward for the unchosen action. Since the trained SGRUs could have produced either model-free RL and latent-state behavior, the findings above suggest that the monkey's behavior aligns more closely with that of a model-free RL agent.

Identifying novel behavioral patterns with one-unit RNNs

Our proposed approach not only facilitates model interpretation and comparison, but can also uncover novel insights into the animal's cognitive processes driving decision-making. In particular, we can discern crucial differences between the phase portraits of the SGRU and the model-free RL agent (Fig. 5a). For instance, the curved shapes in the SGRU diagram suggest that learning rates depend on animals' internal states (notice light blue and light gray lines). The decoupling of the two curves for $R = 0$ is a sign of choice perseveration, but since they decouple only for extreme values of L, it suggests a peculiar pattern of preference-dependent choice perseveration. Crucially, insights gathered from the phase portraits can then be confirmed by targeted hypothesis testing (see Supplementary Results and Fig. S4).

Another intriguing aspect of the SGRU diagram is that the fixed point for $A_1, R = 1$ has a larger magnitude than the fixed point for $A_2, R = 1$. This suggests that the monkey never exhibits a preference for A_2 as high as it does for A_1, akin to a choice bias. To better understand this phenomenon, we defined the utility $u(I) = L_*(I)/\max_I L_*(I)$ as the perceived effect of a reward on the normalized action values (i.e., a "utility prediction error" $[u(I) - L(t)/\max_I L_*(I)]$), and compared $u(I)$ for different inputs across models (Fig. 5b). These results suggest an alternative interpretation where the utility of a reward was perceived as lower when following A_2 than when following A_1.

The utility $u(I)$ provides an alternative way of describing the effect of inputs on the state of the system. To illustrate this, we computed $u(I)$ for various models fitted to one rat's behavior on the two-stage task (note that we again found no evidence for a latent-state Bayesian strategy, see Fig. S5). In this task, each cognitive model assumes a specific pattern for $u(I)$ as a function of the eight possible inputs (combinations of two actions, two second-stage states, and two rewards; Fig. 5c). Interpreting the utilities in the SGRU model, however, uncovers an intriguing pattern overlooked by all cognitive models: when the animal experiences a rare transition followed by a reward (dark blue marker for A_1, S_2 and dark gray marker for A_2, S_1), the animal gravitates towards indifference ($|u(I)| \approx 0$). Thus, while rewards after common transitions lead to certainty, rewards after rare transitions result in indifference (possibly due to uncertainty). We found this effect in several rats (Miller et al.) and mice (Akam et al.), with some other animals showing slightly different patterns in rare transitions (see Fig. S5).

Lastly, we analyzed the behavior of mice on the novel two-stage task. We found that the pattern of utilities for the SGRU resembled that of a model-free RL agent where the presence of reward leads to more certainty than no reward ($|u(R = 1)| > 0.5 > |u(R = 0)|$). However, substantial individual differences in utility patterns exist (see Fig. S6). Comparing an individual's tendency to perceive rewards as less certain (calculated as $1 - |u(I)|$ averaged over four situations when the reward is present) with the individual's performance (the average number of trials in a block before reaching a performance threshold), we found that the two quantities are strongly correlated ($\rho = 0.62$, $p = 0.008$). This suggests that mice who perceive rewards as less certain tend to exhibit poorer task performance. More broadly, these analyses demonstrate how the dynamics of tiny RNNs can reveal numerous insights that would be difficult with traditional neuroscience methods.

RNNs with two units reveal additional behavioral patterns

Our findings thus far demonstrate the potential of one-unit RNNs to reveal novel insights into cognitive processes underlying decision-making. However, we previously showed that RNNs with 2-4 units are even better than one-unit RNNs at predicting animals' choices, often outperforming more complex models. Although these models are no longer fully describable by our logit-change analysis, it remains a useful diagnostic tool for identifying policy differences between models and revealing behavioral patterns (Fig. S7). In this section, we demonstrate a method that provides even deeper insights from two-dimensional models, analyzing a monkey's behavior on the reversal learning task as an example.

We analyzed the dynamics of the fitted two-dimensional model-free RL model and the fitted two-unit GRU model by studying their vector fields, with axes corresponding to state variables and arrows indicating state changes during each trial (Fig.6). For the two-dimensional model-free RL model, we used action values $Q(A_1)$ and $Q(A_2)$ to define the state-space (Fig.6a). The vector fields show that experiencing a reward after choosing A_i increases $Q(A_i)$, while experiencing no reward decreases $Q(A_i)$, with no effect on the other Q-value. These dynamics correspond to a line attractor for each input condition. For a two-dimensional model-free RL model with value forgetting, the arrow trajectories instead follow point attractor dynamics (Fig.S8a). As in any dynamical system, the dynamics around the fixed points can be fully specified by the singular value decomposition.

For the fitted two-unit RNN model, we used the two dynamical variables (hidden layer activations) to define the state-space (Fig.6b). We found that these variables learned a quantity analogous to an action value even when not constrained to do so (see Fig.S8b for the constrained case). The network dynamics correspond to an approximate line attractor in each of

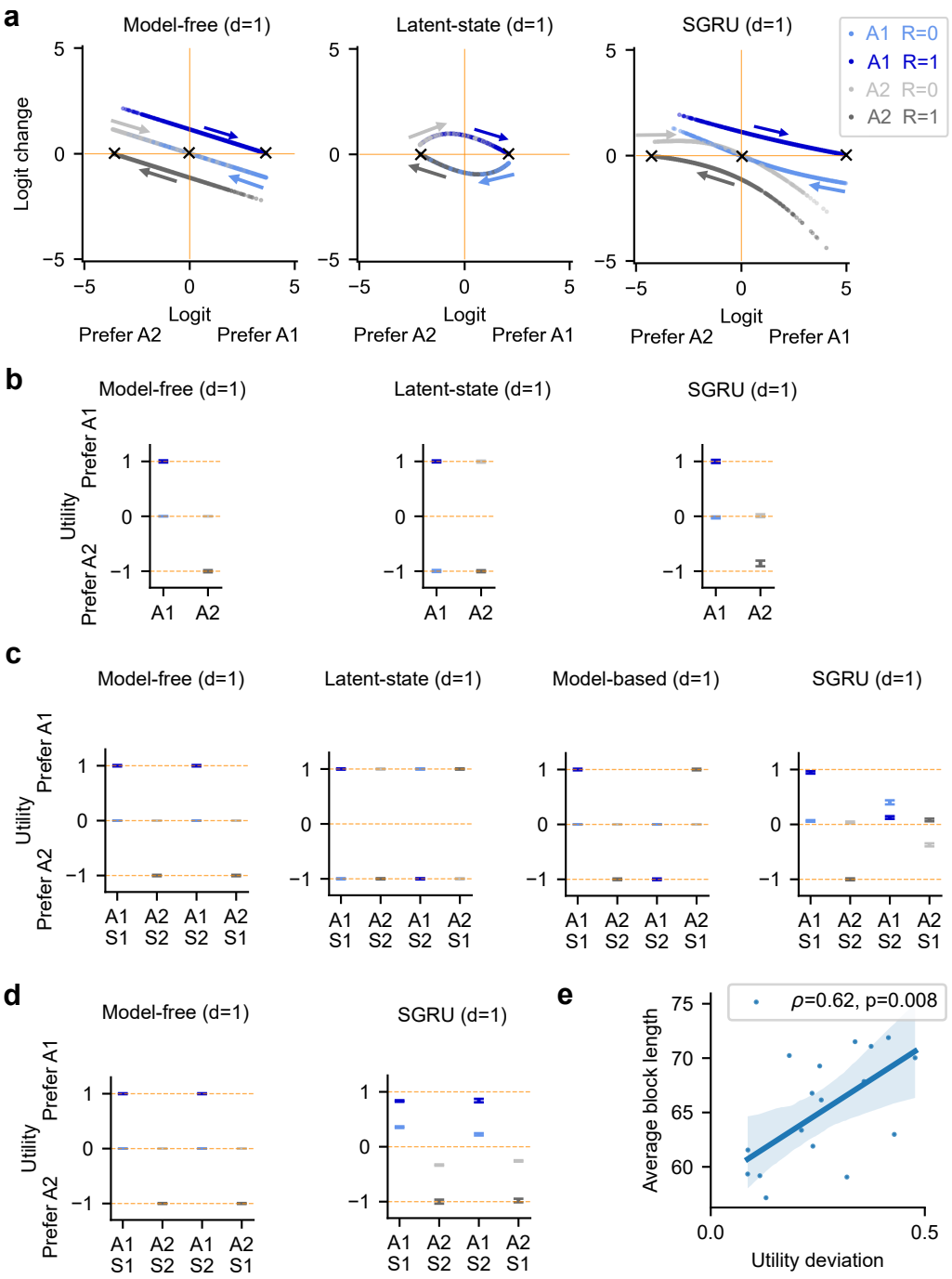

Fig. 5. Leveraging dynamical systems theory for model interpretation and comparison, one-unit RNNs identify novel behavioral patterns. The panels display the logit analysis, for models fitted to animals in three tasks. (a) Logit-change as a function of input (action and reward) and the logit at each trial, in three one-dimensional models fitted to one monkey in the reversal learning task. Each point, colored according to the corresponding input, represents a trial. The arrows indicate the flow direction of the model state (logit) after receiving the input, with rightward arrows for a positive logit-change and leftward arrows for a negative logit-change. The crosses are the stable fixed points (attractors) of the dynamics where the logit-change vanishes. (b) Attractors $L_*(I)$ for the input I in each model dynamics are extracted and normalized by the largest magnitude, termed utility $u(I) = L_*(I)/\max_I L_*(I)$. The error bars are the standard deviations over different model instantiations (in nested cross-validation). (c) Utility pattern of each model dynamics fitted to one rat in the two-stage task. The best-performing SGRU model discovered a utility pattern different from other models. (d) Utility pattern of each model dynamics fitted to one mouse in the novel two-stage task. (e) Utility deviation, an individual's tendency to perceive rewards as less certain (calculated as $1 - |u(I)|$ averaged over four situations when the reward is present), strongly correlates with the length of adaptive blocks, an indicator of task performance. The shaded region is the 95% confidence interval.

the four input conditions. The vector fields show that, in rewarded trials, the value of the chosen action gravitates towards a maximum (represented as crosses in the diagram), and the value of the unchosen action decays slightly. In unrewarded trials, interestingly, the value of the chosen action gravitates towards the value of the unchosen action, representing a rather peculiar decay-to-the-other forgetting rule.

We validated this surprising prediction through a targeted analysis comparing variants of a model-free RL agent, including (i) no forgetting; (ii) conventional decay-to-zero forgetting rule; and (iii) decay-to-the-other forgetting rule. The model-free RL model with the decay-to-the-other rule outperformed the other two variants and achieved a predictive performance close to the two-unit GRU model (Fig.S9). Finally, by analyzing phase portrait diagrams of two-unit RNNs (including a new variant called "switching linear neural network") in the two-stage and novel two-stage tasks, we found that these networks were also able to discover strategies strikingly different from typical two-action-value formalizations (Supplementary Results, Fig.S10, S11, S12).

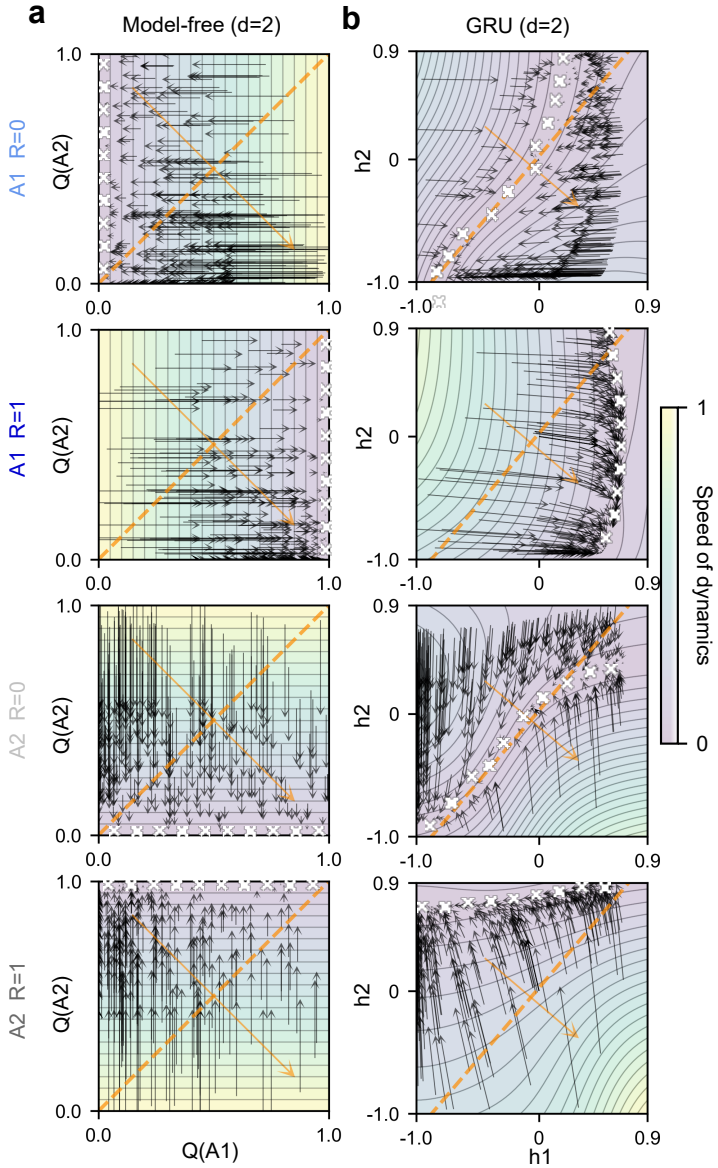

Fig. 6. RNNs with two units reveal additional behavioral patterns. Panels display the vector field analysis for the two-dimensional models fitted to a monkey's behavior on the reversal learning task. (a) The model-free RL model. (b) The GRU model. Black arrows (flow lines) are the actual state trajectories experienced by the models. Orange dashed lines are the decision boundaries, and the orange arrows orthogonal to decision boundaries are the readout vectors. The crosses mark the approximate line attractors. The two-dimensional state space is colored by the speed of the dynamics, defined by the norm of the velocity vector at each position (normalized by the maximal norm). See details in the main text.

Analyzing behavior in task-optimized neural networks

In this study, our primary focus has been on analyzing neural networks whose parameters were fitted to observable behavior. We note that this approach is distinct from another common use of neural networks in neuroscience, where parameters are adjusted to achieve optimal performance on a specific task. In such scenarios, researchers compare the behavior and neural representations of task-optimized networks to those of biological organisms, hoping to gain valuable insights into the latter. In this final section, we wish to assess whether our approach can also shed light on computations in task-optimized networks, enabling a more comprehensive comparison between biological and artificial systems.

To investigate the potential of our approach for analyzing task-optimized networks, we trained an RNN agent within a meta-reinforcement learning (meta-RL) framework to maximize total rewards in a two-stage task (Fig.7a; Fig.S13). Previous work demonstrated that task-optimized meta-RL agents are capable of learning a standalone learning algorithm, akin to how interactions between the prefrontal cortex and striatum might learn network dynamics implementing different algorithms according to task demands[24]. In particular, prior work suggested that task-optimized meta-RL agents use a model-based RL strategy in the two-stage task, based on analyses of "stay probability" and "prediction errors"[24]. To test this hypothesis more directly, a common approach is to use model comparison, asking which of several models is most consistent with the data (e.g., Fig. S14a). However, such approaches cannot shed light on the mechanisms generating the data.

To gain mechanistic insights into the strategy learned by the meta-RL agent, we generated the phase portrait for the agent and a one-dimensional latent-state agent fitted to the agent's behavior. Surprisingly, we discovered that the well-trained meta-RL agent learned a latent-state-*like* solution (Fig. 7b,c). Our logit analysis further revealed that, during training, the intermediate representations before convergence were substantially different from any cognitive models (see Fig. S14b). Subtle differences nonetheless existed between the well-trained meta-RL agent and the latent-state agent, with the logit-change lying on several adjacent curves of the logit for a given input condition, deviating slightly from the latent-state's prediction of an exact function of logit. We determined that these adjacent curves directly reflect information about historical input sequences (see Fig. S14c), suggesting that such a "history effect" slightly disrupts the acquisition of a perfect Bayesian inference algorithm. These analyses demonstrate that meta-RL agents learned a strategy akin to Bayesian inference in the two-stage task, markedly different from the strategies employed by animals. Overall, it illustrates how our approach contributes to our understanding of computational processes in both biological and artificial systems.

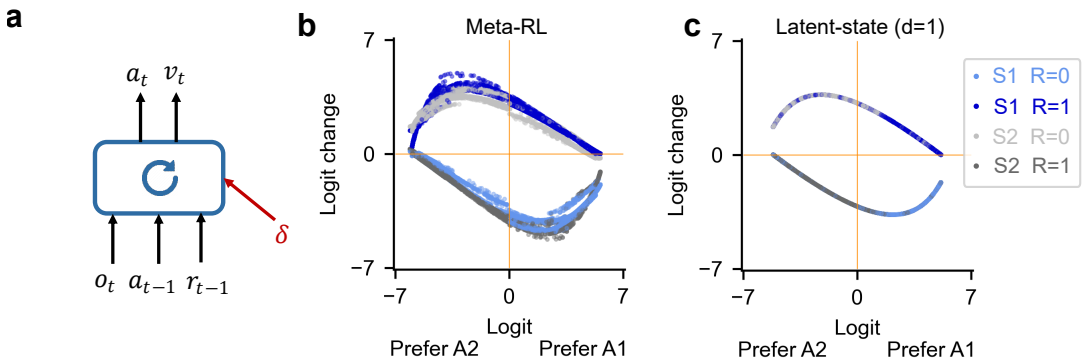

Fig. 7. A meta-reinforcement learning agent learns a latent-state-like strategy different from animals. (a) Architecture of the meta-RL agent. The agent receives the observation, the past action, and the past reward as input, and generates a new action and an estimate of the current state value. It adjusts the synaptic weights to optimize the performance in the two-stage task using gradient descent based on the reward prediction error signals δ. (b-c) The logit-change pattern computed from the output of the meta-RL agent is similar to the logit-change pattern of a latent-state model fitted to the agent' behavior.

Discussion

In this study, we presented a novel modeling approach using tiny RNNs to describe computations underlying animal behavior in decision-making tasks. Across three well-studied reward learning tasks and three animal species, we found that RNNs with only one or two units can predict the behavior of individual animals better than any of the tested classical cognitive models of equal dimensionality. Leveraging discrete dynamical systems theory, we obtained numerous insights into the cognitive strategies employed by animals during decision-making. For example, we found that the animals generally employed cognitive strategies that resembled RL, but not Bayesian inference, though this behavior was additionally colored by various patterns, including previously unrecognized forms of value forgetting, choice perseveration, and choice biases.

We argue that these insights would have been difficult to obtain with conventional methods. We also demonstrated how our approach provides an estimate of the dimensionality of behavior and mechanistic insights into the emergent algorithms in task-optimized RL agents. Altogether, the proposed approach has the potential to shed light onto new cognitive strategies underlying animal decision-making and to advance our understanding of the neural mechanisms underlying behavior.

Modeling decision-making with tiny RNNs

Our finding that ANNs outperform classical cognitive models in capturing variance in behavior is consistent with various previous studies demonstrating the superior performance of both feedforward[27] and recurrent neural networks[28,29] over simpler models. The superior performance of ANNs can be attributed to the increased flexibility enabled by the much larger number of parameters typically used by these models. The uniqueness of our work lies in our use of RNNs at the smallest scale, with only a handful of units, in strong contrast to the typical RNNs with tens to hundreds of neurons in cognitive tasks[24,26,28,29] (but see tiny RNNs for knowledge distillation[38]). The smaller scale of such networks enabled a more interpretable model. While larger networks typically achieve higher performance, this was not the case in the three tasks studied, possibly due to the simplicity of the behavior expressed by animals on those tasks. We speculate that behavior will be similarly low-dimensional in many other experimental paradigms used in psychology and cognitive science, and that tiny RNNs will be a useful tool for understanding such behaviors.

An expected concern is that networks with only one or two units are too simple to fully capture the behavior of animals in laboratory tasks. We found that this was not the case, as it outperformed all cognitive models of equal dimensionality. As with other ANN models, the superior performance of our tiny RNNs can be attributed to them having a larger number of parameters, which enables them to learn a richer class of functions for integrating past information into the state variables. In particular, tiny RNNs could learn to express behaviors consistent with normative models, including RL and Bayesian inference agents. This crucial observation suggests that tiny RNNs can be viewed as a superset of cognitive models, capable of accounting for a wide range of cognitive strategies and for normative behavior in particular.

Another expected concern is that neural networks are too complex to model the simple behavior expressed by animals in laboratory tasks. For instance, highly flexible models typically require larger datasets are needed for training. We have not found this to be a serious concern, as datasets of standard size in neuroscience appear to be sufficient to take advantage of the network's superior flexibility. Another possible concern is that the complexity of neural networks renders them less interpretable. We have found that this was also not the case. Using tools from dynamical systems, we could comprehensively analyze the properties of the trained RNNs and compare them to other cognitive models, leading to critical insights into the behavior of both models. This is a distinct advantage over analyzing larger models that require simulations with non-representative input sequences[28] or dimensionality reduction of dynamics that may have behaviorally irrelevant dimensions[39].

Interpretation and insights from discrete dynamical systems approach

The application of discrete dynamical systems theory to cognitive models and RNNs enabled a rich understanding of their inner workings, akin to how continuous dynamical systems theory has been used to explain the coordinated activity of interacting neural populations[40] implementing functions such as motor control[41], working memory[42], perceptual decision-making[23] and value prediction[43]. We have analyzed how the one- or two-dimensional models' internal states (formally, the logits and the phase portraits of the policy) change over time as a function of reward, observations, and past actions, which can be directly generalized to models with more dynamical variables. By describing all models using the same framework, we could draw direct comparisons between the strategies learned by the RNNs and those assumed in the cognitive models. Our results showed that, in all tasks, animals employed RL-like heuristics in updating their internal variables, in contrast with the nonlinear dynamics predicted by the latent-state model using Bayesian inference and by meta-RL agents optimized to perform the same tasks (see Supplementary Discussion). This discrepancy suggests that either animals might not fully grasp the task reward rule or that animals indeed resort to RL as a general approach for solving reward learning tasks.

Our results also suggest that animals treated the presence and absence of reward differently. This contradicts the ideal-observer behavior assumed in latent-state models where an animal should leverage its knowledge of the task structure to infer that the absence of reward for one action indicates the presence of reward for the other action. One possible reason for the asymmetrical treatment of reward is that animals were never shown the reward for the unchosen action, and thus were unable to learn the symmetric nature of the task. Another possibility is that animals exhibited a "confirmation effect" whereby uncertainty is reduced by confirmatory evidence (e.g., a reward when a reward is expected) but increased otherwise (e.g., a lack of reward when a reward is expected). This explanation is indeed consistent with the suboptimal behavior commonly observed in perceptual decision tasks where animals process past success and failure differently[44], reflecting qualitatively different neural mechanisms triggered by outcomes with valence[45]. Our analyses of the two-stage task data provide further support for a confirmation effect. Here, when a rare transition leads to a reward, animals do not increase their preference for the chosen (model-free) or the unchosen (model-based and latent-state) actions, but rather become

increasingly indifferent. Overall, our results suggest that, in the three examined tasks but possibly also many others, the modulatory effect of reward on behavior may depend on past events, including past actions and observations.

Additionally, we observed choice perseveration in the reversal learning task, where monkeys tended to repeat the same action regardless of the reward outcome (see Supplementary Results). However, such perseveration was only found when animals had a high preference for either action. Note also that we did not need to introduce an additional dynamical variable for perseveration as commonly done in other models[12]; rather, this nuanced behavior can be inferred by visually inspecting the logit dynamics.

Finally, our approach for identifying the interpretable cognitive strategy most consistent with the observed behavior could have important implications for computational psychiatry[46,47], which takes a theory-based approach to provide a formal account of mental illness. Our framework can improve the understanding of the cognitive dynamics underlying behavior in both healthy and clinical populations, enabling improved characterization of behavioral disorders.

Dimensionality of cognitive dynamics

The dimensionality of an agent's behavior can be defined as the number of functions of the agent's past one needs to measure in order to accurately predict its future as well as possible[35,36]. Our analysis of the predictive performance of RNNs with varying sizes revealed that merely 1-4 dynamical variables were adequate for an optimal explanation of behavior in reward learning tasks, signifying the low-dimensional nature of these tasks. This observation aligns with other studies that reported low-dimensional dynamics in cortical regions[23] and the surprising effectiveness of low-rank RNNs in accounting for intricate neural dynamics[48].

It is reasonable to assume that more complex, naturalistic tasks would necessitate higher-dimensional behavior. Nevertheless, we propose that even in such scenarios, employing tiny RNNs for model selection and interpretation can offer valuable insights. In fact, other low-dimensional models, such as Bayesian inference and RL, have been successfully applied to tasks with a diverse range of complexities. More broadly, the identification of dynamical variables in experimental data is an ongoing research topic in the field of neuroscience, complex systems, and physics, including efforts to extract a small set of variables from neural data[39,48], from video recordings of physical dynamical systems[49] and to learn effective dynamics for multiscale complex systems[50].

GRU, SGRU, and extensions

Our networks utilized Gated Recurrent Units (GRUs), a recurrent architecture whose hidden state is updated by learnable functions of its inputs. This architecture features Markovian hidden variables, in contrast to the more widespread Long Short-Term Memory (LSTM) networks. We also introduced the Switching GRU (SGRU) architecture, a versatile extension of GRUs that accommodates distinct sets of recurrent weights for each discrete input. Generalizations of SGRUs from discrete to continuous inputs are closely related to multiplicative integration GRUs[51]. Even without additional engineering, vanilla GRUs can be applied as usual in contexts with continuous inputs and a larger number of actions.

We cannot rule out the possibility that a more complex set of variable-updating equations can explain more variance in behavior with fewer dynamical variables, as the family of low-dimensional dynamics that can be captured by SGRUs is also limited (see Fig. S15 for the case of one-dimensional dynamics). Low-rank RNNs could serve as a candidate for this purpose[48,52–54]. In these networks, the dimensionality of the hidden dynamics is constrained by the rank of recurrent connection matrices. Under appropriate conditions, such d-rank RNNs can approximate any d-dimensional autonomous dynamical systems, provided they have a sufficient number of neurons[53]. For non-constant inputs, the dynamics of low-rank RNNs exist in a $(d + N_{\text{in}})$-dimensional space, where N_{in} denotes the number of input dimensions. This contrasts with our RNN models, in which the dimensionality of hidden dynamics is independent of the number of input dimensions. Further investigation is needed to determine the advantages and disadvantages of these two approaches.

Individual and grouped choice data

In contrast to prior studies that utilized grouped choice data for training RNNs in reward-based learning tasks[28,29,55], our approach focused on training separate RNNs for each individual. Future investigations could benefit from comparing our individually trained RNNs to those trained on group data. Although training a single RNN on all subjects may have more trials and potentially enhance performance, the resulting impact on learned strategies remains uncertain.

Conclusion

In conclusion, we have developed a framework to discover the cognitive algorithms underlying decision-making using low-dimensional RNNs. Our approach not only facilitates model interpretation and selection but also enables the discovery of new aspects of behavior inaccessible by previous models. This work paves the way for more sophisticated analyses of

decision-making and other cognitive processes, ultimately contributing to a deeper understanding of the neural mechanisms underlying both adaptive and pathological behavior.

Acknowledgments

We thank the support from Swarma Club and Causal Emergence Reading Group supported by the Save 2050 Programme jointly sponsored by Swarma Club and X-Order. We thank K. T. Jensen, H. Xiong, Z. Jin, and X. Li for their inspirational discussions. This work was supported by the Kavli Institute for Brain and Mind (KIBM) Innovative Research Grant #2022-2209 and in part by National Science Foundation (NSF) awards CNS-1730158, ACI-1540112, ACI-1541349, OAC-1826967, OAC-2112167, CNS-2100237, CNS-2120019, the University of California Office of the President, and the University of California San Diego's California Institute for Telecommunications and Information Technology/Qualcomm Institute. Thanks to CENIC for the 100Gbps networks.

Author Contributions

All authors conceived the study, participated in the discussions, wrote the paper, and acquired funding.

Declaration of Interests

All authors declare no competing interests.

Methods

Tasks and datasets

Reversal learning task

The reversal learning task is a paradigm designed to assess subjects' ability to adapt their behavior in response to changing reward contingencies. In each trial, subjects are presented with two actions, A_1 and A_2, yielding a unit reward with probability p_1^{reward} and p_2^{reward}, respectively. These reward probabilities remain constant for several trials before switching unpredictably and abruptly, without explicit cues. When this occurs, the action associated with the higher reward probability becomes linked to the lower reward probability, and vice versa. The task necessitates continuous exploration of which action currently has a higher reward probability in order to maximize total rewards. For consistency with the other tasks, we assume that actions (A_1 and A_2) are made at the choice state, and A_i deterministically leads to state S_i, where the reward is delivered.

In the Bartolo dataset[11], two monkeys completed a total of 15,500 trials of the reversal learning task with two state-reward types: (1) $p_1^{\text{reward}} = 0.7$ and $p_2^{\text{reward}} = 0.3$; (2) $p_1^{\text{reward}} = 0.3$ and $p_2^{\text{reward}} = 0.7$. Blocks were 80 trials long, and the switch happened at a "reversal trial" between 30 and 50. We predicted the behavior from trials 10 to 70, similar to the original preprocessing procedure[11] because the monkeys were inferring the current block type ("what" block: choosing from two objects; "where" block: choosing from two locations) in the first few trials.

In the Akam dataset[12], ten mice completed a total of 67,009 trials of the reversal learning task with three state-reward types: (1) $p_1^{\text{reward}} = 0.75$ and $p_2^{\text{reward}} = 0.25$; (2) $p_1^{\text{reward}} = 0.25$ and $p_2^{\text{reward}} = 0.75$; (3) $p_1^{\text{reward}} = 0.5$ and $p_2^{\text{reward}} = 0.5$ (neutral trials). Block transitions from non-neutral blocks were triggered 10 trials after an exponential moving average (tau = 8 trials) crossed a 75% correct threshold. Block transitions from neutral blocks occurred with a probability 10% on each trial after the 15th of the block to give an average neutral block length of 25 trials.

Two-stage task

The two-stage task is a paradigm commonly used to distinguish between the influences of model-free and model-based reinforcement learning on animal behavior[56,57]. In each trial, subjects are presented with two actions, A_1 and A_2, while at the choice state. Action A_1 leads with a high probability to state S_1 and a low probability to state S_2, while action A_2 leads with a high probability to state S_2 and a low probability to state S_1. From second-stage states S_1 and S_2, the animal can execute an action for a chance of receiving a unit reward. Second-stage states are distinguishable by visual cues and have different probabilities of yielding a unit reward: p_1^{reward} for S_1 and p_2^{reward} for S_2. These reward probabilities remain constant for several trials before switching unpredictably and abruptly, without explicit cues.

In the Miller dataset[13], four rats completed a total of 33,957 trials of the two-stage task with two state-reward types: (1) $p_1^{\text{reward}} = 0.8$ and $p_2^{\text{reward}} = 0.2$; (2) $p_1^{\text{reward}} = 0.2$ and $p_2^{\text{reward}} = 0.8$. Block switches occurred with a 2% probability on each trial after a minimum block length of 10 trials.

In the Akam dataset[12]), ten mice completed a total of 133,974 trials of the two-stage task with three state-reward types: (1) $p_1^{\text{reward}} = 0.8$ and $p_2^{\text{reward}} = 0.2$; (2) $p_1^{\text{reward}} = 0.2$ and $p_2^{\text{reward}} = 0.8$; (3) $p_1^{\text{reward}} = 0.4$ and $p_2^{\text{reward}} = 0.4$ (neutral trials). Block transitions occur 20 trials after an exponential moving average (tau = 8 trials) of subject's choices crossed a 75% correct threshold. In neutral blocks, block transitions occurred with 10% probability on each trial after the 40th. Transitions from non-neutral blocks occurred with equal probability either to another non-neutral block or to the neutral block. Transitions from neutral blocks occurred with equal probability to one of the non-neutral blocks.

Novel two-stage task

The novel two-stage task is a modified version of the original two-stage task, with the introduction of occasional reversals in action-state transition probabilities[12]. This modification was proposed to facilitate the dissociation of state prediction and reward prediction in neural activity and to prevent habit-like strategies that may produce model-based control-like behavior without forward planning. In each trial, subjects are presented with two actions, A_1 and A_2, at the choice state. One action commonly leads to state S_1 and rarely to state S_2, while the other action commonly leads to state S_2 and rarely to state S_1. These action-state transition probabilities remain constant for several trials before switching unpredictably and abruptly, without explicit cues. In second-stage states S_1 and S_2, subjects execute an action for a chance of receiving a unit reward. The second-stage states are visually distinguishable and have different reward probabilities that also switch unpredictably and abruptly, without explicit cues, similar to the other two tasks.

In the Akam dataset[12]), seventeen mice completed a total of 230,237 trials of the novel two-stage task with two action-state types: (1) $\Pr(S_1|A_1) = \Pr(S_2|A_2) = 0.8$ and $\Pr(S_2|A_1) = \Pr(S_1|A_2) = 0.2$; (2)$\Pr(S_1|A_1) = \Pr(S_2|A_2) = 0.2$ and $\Pr(S_2|A_1) = \Pr(S_1|A_2) = 0.8$. There were also three state-reward types: (1) $p_1^{\text{reward}} = 0.8$ and $p_2^{\text{reward}} = 0.2$; (2)

$p_1^{\text{reward}} = 0.2$ and $p_2^{\text{reward}} = 0.8$; (3) $p_1^{\text{reward}} = 0.4$ and $p_2^{\text{reward}} = 0.4$ (neutral trials). Block transitions occur 20 trials after an exponential moving average (tau = 8 trials) of subject's choices crossed a 75% correct threshold. In neutral blocks, block transitions occurred with 10% probability on each trial after the 40th. Transitions from non-neutral blocks occurred with equal probability (25%) either to another non-neutral block via reversal in the reward or transition probabilities, or to one of the two neutral blocks. Transitions from neutral blocks occurred via a change in the reward probabilities only to one of the non-neutral blocks with the same transition probabilities.

Recurrent neural networks

Network architecture

Recurrent layer The neural network models in this paper all used gated recurrent units (GRU) in their hidden layers[32]. The hidden state h_t at the beginning of trial t consists of d elements (dynamical variables). The initial hidden state h_1 is set to 0 and h_t ($t > 1$) is updated as follows:

$$\begin{aligned} r_t &= \sigma(W_{ir}x_{t-1} + b_{ir} + W_{hr}h_{t-1} + b_{hr}) \\ z_t &= \sigma(W_{iz}x_{t-1} + b_{iz} + W_{hz}h_{t-1} + b_{hz}) \\ n_t &= \tanh(W_{in}x_{t-1} + b_{in} + r_t \odot (W_{hn}h_{t-1} + b_{hn})) \\ h_t &= (1 - z_t) \odot n_t + z_t \odot h_{t-1} \end{aligned} \tag{1}$$

where σ is the sigmoid function, $\odot$ is the Hadamard (element-wise) product, x_{t-1} and h_{t-1} are the input and hidden state from the last trial $t-1$, and r_t, z_t, n_t are the reset, update, new gates (intermediate variables) at trial t, respectively. The weight matrices $W_{..}$ and biases $b_{..}$ are trainable parameters. The hidden state of the network, h_t, represents a summary of past inputs and is the only information used to generate outputs.

In this work, we also introduce a modified architecture called switching GRU (SGRU). In this architecture, the hidden state h_t ($t > 1$) is updated as follows:

$$\begin{aligned} r_t &= \sigma(b_{ir}^{(x_{t-1})} + W_{hr}^{(x_{t-1})}h_{t-1} + b_{hr}^{(x_{t-1})}) \\ z_t &= \sigma(b_{iz}^{(x_{t-1})} + W_{hz}^{(x_{t-1})}h_{t-1} + b_{hz}^{(x_{t-1})}) \\ n_t &= \tanh(b_{in}^{(x_{t-1})} + r_t \odot (W_{hn}^{(x_{t-1})}h_{t-1} + b_{hn}^{(x_{t-1})})) \\ h_t &= (1 - z_t) \odot n_t + z_t \odot h_{t-1} \end{aligned} \tag{2}$$

where $W_{h\cdot}^{(x_{t-1})}$ and $b_{\cdot\cdot}^{(x_{t-1})}$ are the weight matrices and biases selected by the input x_{t-1} (i.e., each input x_{t-1} induces an independent set of weights $W_{h\cdot}$ and biases $b_{\cdot\cdot}$).

In switching linear neural networks (SLIN), the hidden state h_t ($t > 1$) is updated as follows:

$$h_t = W^{(x_{t-1})}h_{t-1} + b^{(x_{t-1})} \tag{3}$$

where $W^{(x_{t-1})}$ and $b^{(x_{t-1})}$ are the weight matrices and biases selected by the input x_{t-1}. In some variants, we constrained $W^{(x_{t-1})}$ to be symmetric.

Input layer The network's input x_t consists of the previous action a_{t-1}, the previous second-stage state s_{t-1}, and the previous reward r_{t-1} (but $a_t = s_t$ in the reversal learning task). In the GRU networks, the input x_t is three-dimensional and projects with linear weights to the recurrent layer. In the SGRU networks, the input x_t is used as a selector variable where the network's recurrent weights and biases depend on the network's inputs. Thus, SGRUs trained on the reversal learning task have four sets of recurrent weights and biases corresponding to all combinations of a_{t-1} and r_{t-1}, and SGRUs trained on the two-stage and novel two-stage tasks have eight sets of recurrent weights and biases corresponding to all combinations of a_{t-1}, s_{t-1}, and r_{t-1}.

Output layer The network's output consists of two units whose activities are linear functions of the hidden state h_t. A softmax function (a generalization of the logistic function) is used to convert these activities into a probability distribution (a policy). In the first trial, the network's output is read out from the initial hidden state h_1, which has not yet been updated on the basis of any input. For two-unit networks, the network's output was computed either as a linear function of the hidden state (i.e., $o_t^{(i)} = \beta_{1,i} \cdot h_t^{(1)} + \beta_{2,i} \cdot h_t^{(2)}$) or as a direct readout of the hidden state (i.e., $o_t^{(i)} = \beta \cdot h_t^{(i)}$).

Network training

Networks were trained using the AdamW optimizer[58] (learning rate of 0.005, no weight decay) on batched training data with cross-entropy loss, recurrent weight L1-regularization loss (coefficient drawn between 1e-5 and 1e-1, depending on experiments), and early-stop (if the validation loss does not improve for 200 iteration steps). All networks were implemented with PyTorch.

Classical cognitive models

Models for the reversal learning task

In this task, we implemented one model from the latent-state family and five models from the model-free family (adopted from[57] and[13], or constructed from GRU phase portraits).

Latent-state strategy (d=1). This model assumes the existence of the latent state h, with $h = i$ representing a higher reward probability following action A_i (state S_i). The probability $\mathrm{Pr}_t(h = 1)$, as the dynamical variable, is first updated via Bayesian inference:

$$\hat{\mathrm{Pr}}_t(h = 1) = \frac{\mathrm{Pr}(r_{t-1}|h = 1, s_{t-1})\,\mathrm{Pr}_{t-1}(h = 1)}{\mathrm{Pr}(r_{t-1}|h = 1, s_{t-1})\,\mathrm{Pr}_{t-1}(h = 1) + \mathrm{Pr}(r_{t-1}|h = 2, s_{t-1})\,\mathrm{Pr}_{t-1}(h = 2)}, \tag{4}$$

where the left-hand side is the posterior probability (we omit the conditions for simplicity). The agent also incorporates the knowledge that, in each trial, the latent-state h can switch (e.g., from $h = 1$ to $h = 2$) with a small probability p_r. Thus the probability $\mathrm{Pr}_t(h)$ reads,

$$\mathrm{Pr}_t(h = 1) = (1 - p_r)\hat{\mathrm{Pr}}_t(h = 1) + p_r(1 - \hat{\mathrm{Pr}}_t(h = 1)). \tag{5}$$

The action probability is then derived from softmax$(\beta\mathrm{Pr}_t(h = 1), \beta\mathrm{Pr}_t(h = 2))$ with inverse temperature β ($\beta \geq 0$).

Model-free strategy (d=1). This model hypothesizes that the two action values $Q_t(A_i)$ are fully anti-correlated ($Q_t(A_1) = -Q_t(A_2)$) as follows:

$$\begin{aligned} Q_t(a_{t-1}) &= Q_{t-1}(a_{t-1}) + \alpha(r_{t-1} - Q_{t-1}(a_{t-1})) \\ Q_t(\bar{a}_{t-1}) &= Q_{t-1}(\bar{a}_{t-1}) - \alpha(r_{t-1} + Q_{t-1}(\bar{a}_{t-1})), \end{aligned} \tag{6}$$

where $\bar{a}_{t-1}$ is the unchosen action, and α is the learning rate ($0 \leq \alpha \leq 1$). We specify the $Q_t(A_1)$ as the dynamical variable.

Model-free strategy (d=2). This model hypothesizes that the two action values $Q_t(A_i)$, as two dynamical variables, are updated independently:

$$Q_t(a_{t-1}) = Q_{t-1}(a_{t-1}) + \alpha(r_{t-1} - Q_{t-1}(a_{t-1})). \tag{7}$$

The unchosen action value $Q_t(\bar{a}_{t-1})$ is unaffected.

Model-free strategy with the decay-to-zero rule (d=2). The chosen action value is updated as in the previous model. The unchosen action value $Q_t(\bar{a}_{t-1})$, instead, is gradually forgotten:

$$Q_t(\bar{a}_{t-1}) = DQ_{t-1}(\bar{a}_{t-1}), \tag{8}$$

where D is the value forgetting rate ($0 \leq D \leq 1$).

Model-free strategy with the decay-to-other rule (d=2). This strategy is constructed from the phase diagram of the two-unit GRU. When there is a reward, the chosen action value is updated as follows,

$$Q_t(a_{t-1}) = D_1 Q_{t-1}(a_{t-1}) + 1, \tag{9}$$

where D_1 is the value decaying rate ($0 \leq D_1 \leq 1$). The unchosen action value is slightly decreased:

$$Q_t(\bar{a}_{t-1}) = Q_{t-1}(\bar{a}_{t-1}) - b, \tag{10}$$

where b is the decaying bias ($0 \leq b \leq 1$, usually small). When there is no reward, the unchosen action value is unchanged and the chosen action value decays to the other:

$$Q_t(a_{t-1}) = Q_{t-1}(a_{t-1}) + \alpha_0(Q_{t-1}(\bar{a}_{t-1}) - Q_{t-1}(a_{t-1})), \tag{11}$$

where α_0 is the learning rate ($0 \leq \alpha_0 \leq 1$).

For all model-free RL models with $d = 2$, the action probability is determined by softmax$(\beta Q_t(A_1), \beta Q_t(A_2))$.

Model-free reward-as-cue strategy (d=8). This model assumes that the animal considers the combination of the second-stage state s_{t-1} and the reward r_{t-1} from the trial $t-1$ as the augmented state $\mathcal{S}_t$ for trial t. The eight dynamical variables are the values for the two actions at the four augmented states. The action values are updated as follows:

$$Q_t(\mathcal{S}_{t-1}, a_{t-1}) = Q_{t-1}(\mathcal{S}_{t-1}, a_{t-1}) + \alpha(r_{t-1} - Q_{t-1}(\mathcal{S}_{t-1}, a_{t-1})). \tag{12}$$

The action probability at trial t is determined by softmax$(\beta Q_t(\mathcal{S}_t, A_1), \beta Q_t(\mathcal{S}_t, A_2))$.

Models for the two-stage task

We implemented one model from the latent-state family, four models from the model-free family, and three from the model-based family (adopted from[57] and[13]).

Latent-state strategy (d=1). Same as latent-state strategy (d=1) in the reversal learning task, except that $h = i$ represents a higher reward probability following state S_i (not action A_i).

Model-free strategy (d=1). Same as the model-free strategy (d=1) in the reversal learning task.

Model-free Q(1) strategy (d=2). Same as the model-free strategy (d=2) in the reversal learning task.

Model-free Q(0) strategy (d=4). This model first updates the first-stage action values $Q_t(a_{t-1})$ with the second-stage state values $V_{t-1}(s_{t-1})$:

$$Q_t(a_{t-1}) = Q_{t-1}(a_{t-1}) + \alpha(V_{t-1}(s_{t-1}) - Q_{t-1}(a_{t-1})), \tag{13}$$

while the unchosen action value $Q_t(\bar{a}_{t-1})$ is unaffected. Then the second-stage state value $V_t(s_{t-1})$ is updated by the observed reward:

$$V_t(s_{t-1}) = V_{t-1}(s_{t-1}) + \alpha(r_{t-1} - V_{t-1}(s_{t-1})). \tag{14}$$

The four dynamical variables are the two action values and two state values.

Model-free reward-as-cue strategy (d=8). Same as model-free reward-as-cue strategy (d=8) in the reversal learning task.

Model-based strategy (d=1). In this model, the two state values $V_t(S_i)$ are fully anti-correlated ($V_t(S_1) = -V_t(S_2)$):

$$\begin{aligned} V_t(s_{t-1}) &= V_{t-1}(s_{t-1}) + \alpha(r_{t-1} - V_{t-1}(s_{t-1})) \\ V_t(\bar{s}_{t-1}) &= V_{t-1}(\bar{s}_{t-1}) - \alpha(r_{t-1} + V_{t-1}(\bar{s}_{t-1})), \end{aligned} \tag{15}$$

where $\bar{s}_{t-1}$ is the unvisited state. The dynamical variable is the state value $V_t(S_1)$.

Model-based strategy (d=2). The visited state value is updated:

$$V_t(s_{t-1}) = V_{t-1}(s_{t-1}) + \alpha(r_{t-1} - V_{t-1}(s_{t-1})). \tag{16}$$

The unvisited state value is unchanged. The two dynamical variables are the two state values.

Model-based strategy with value forgetting (d=2). The visited state value is updated as in the previous model. The unvisited state value is gradually forgotten:

$$V_t(\bar{s}_{t-1}) = DV_{t-1}(\bar{s}_{t-1}), \tag{17}$$

where D is the value forgetting rate ($0 \leq D \leq 1$).

For all model-based RL models, the action values at the first stage are directly computed using the state transition model:

$$Q_t(A_i) = \sum_j \Pr(S_j|A_i)V_t(S_j), \tag{18}$$

where $\Pr(S_j|A_i)$ is known. The action probability is determined by softmax$(\beta Q_t(A_1), \beta Q_t(A_2))$.

Models for the novel two-stage task

For this task, we further include cognitive models proposed in[12]. We first describe different model components (ingredients) and corresponding numbers of dynamical variables, and then specify the components employed in each model.

Second-stage state value component. The visited state value is updated:

$$V_t(s_{t-1}) = V_{t-1}(s_{t-1}) + \alpha_Q(r_{t-1} - V_{t-1}(s_{t-1})). \tag{19}$$

The unvisited state value $V_t(\bar{s}_{t-1})$ is either unchanged or gradually forgotten with f_Q as the value forgetting rate. This component requires two dynamical variables.

Model-free action value component. The first-stage action values $Q_t^{mf}(a_{t-1})$ are updated by the second-stage state values $V_{t-1}(s_{t-1})$ and the observed reward:

$$Q_t^{mf}(a_{t-1}) = Q_{t-1}^{mf}(a_{t-1}) + \alpha(\lambda r_{t-1} + (1-\lambda)V_{t-1}(s_{t-1}) - Q_{t-1}^{mf}(a_{t-1})), \tag{20}$$

where λ is the eligibility trace. The unchosen action value $Q_t^{mf}(\bar{a}_{t-1})$ is unaffected or gradually forgotten with f_Q as the value forgetting rate. This component requires two dynamical variables.

Model-based component. The action-state transition probabilities are updated as:

$$\begin{aligned} P_t(s_{t-1}|a_{t-1}) &= P_{t-1}(s_{t-1}|a_{t-1}) + \alpha_T(1 - P_{t-1}(s_{t-1}|a_{t-1})) \\ P_t(\bar{s}_{t-1}|a_{t-1}) &= P_{t-1}(\bar{s}_{t-1}|a_{t-1}) + \alpha_T(0 - P_{t-1}(\bar{s}_{t-1}|a_{t-1})), \end{aligned} \tag{21}$$

where α_T is the transition probability learning rate. For the unchosen action, the action-state transition probabilities are either unchanged or forgotten:

$$\begin{aligned} P_t(s_{t-1}|\bar{a}_{t-1}) &= P_{t-1}(s_{t-1}|\bar{a}_{t-1}) + f_T(0.5 - P_{t-1}(s_{t-1}|\bar{a}_{t-1})) \\ P_t(\bar{s}_{t-1}|\bar{a}_{t-1}) &= P_{t-1}(\bar{s}_{t-1}|\bar{a}_{t-1}) + f_T(0.5 - P_{t-1}(\bar{s}_{t-1}|\bar{a}_{t-1})), \end{aligned} \tag{22}$$

where f_T is the transition probability forgetting rate.

The model-based action values at the first stage are directly computed using the learned state transition model:

$$Q_t^{mb}(A_i) = \sum_j P_t(S_j|A_i)V_t(S_j). \tag{23}$$

This component requires two dynamical variables ($P_t(S_1|A_1)$ and $P_t(S_1|A_2)$), since other variables can be directly inferred.

Motor-level model-free action component. Due to the apparatus design in this task[12], it is proposed that the mice consider the motor-level actions a_{t-1}^{mo}, defined as the combination of the last-trial action a_{t-1} and the second-stage state s_{t-2} before it. The motor-level action values $Q_t^{mo}(a_{t-1}^{mo})$ are updated as:

$$Q_t^{mo}(a_{t-1}^{mo}) = Q_{t-1}^{mo}(a_{t-1}^{mo}) + \alpha(\lambda r_{t-1} + (1-\lambda)V_{t-1}(s_{t-1}) - Q_{t-1}^{mo}(a_{t-1}^{mo})), \tag{24}$$

where λ is the eligibility trace. The unchosen motor-level action value Q_t^{mo} is unaffected or gradually forgotten with f_Q as the value forgetting rate. This component requires four dynamical variables (four motor-level actions).

Choice perseveration component. The single-trial perseveration $\tilde{X}^{cp}_{t-1}$ is set to -0.5 for $a_{t-1} = A_1$ and 0.5 for $a_{t-1} = A_2$. The multi-trial perseveration Q^{cp}_{t-1} (exponential moving average of choices) is updated as:

$$X^{cp}_t = X^{cp}_{t-1} + \alpha_c(\tilde{X}^{cp}_{t-1} - X^{cp}_{t-1}), \tag{25}$$

where α_c is the choice perseveration learning rate. In some models, the α_c is less than 1, so one dynamical variable is required; while in some other models, the α_c is fixed to 1, suggesting that it is reduced to the single-trial perseveration and no dynamical variable is required.

Motor-level choice perseveration component. The multi-trial motor-level perseveration $X^{mocp}_{t-1}(s_{t-2})$ is updated as:

$$X^{mocp}_t(s_{t-2}) = X^{mocp}_{t-1}(s_{t-2}) + \alpha_m(\tilde{X}^{cp}_{t-1} - X^{mocp}_{t-1}(s_{t-2})), \tag{26}$$

where α_m is the motor-level choice perseveration learning rate. This component requires two dynamical variables.

Action selection component. The net action values are computed as follows:

$$Q^{net}_t(A_i) = G^{mf}Q^{mf}_t(A_i) + G^{mo}Q^{mo}_t(A_i, s_{t-1}) + G^{mb}Q^{mb}_t(A_i) + X_t(A_i), \tag{27}$$

where G^{mf}, G^{mo}, G^{mb} are model-free, motor-level model-free, model-based inverse temperatures, and $X_t(A_i)$ is:

$$\begin{aligned} &X_t(A_1) = 0 \\ &X_t(A_2) = B_c + B_r\tilde{X}^s_{t-1} + P_cX^{cp}_t + P_mX^{mocp}_t(s_{t-1}), \end{aligned} \tag{28}$$

where B_c (bias), B_r (rotation bias), P_c, P_m are weights controlling each component, and $\tilde{X}^s_{t-1}$ is -0.5 for $s_{t-1} = S_1$ and 0.5 for $s_{t-1} = S_2$.

The action probabilities are generated via softmax$(Q^{net}_t(A_1), Q^{net}_t(A_2))$.

Model-free strategies. We include five model-free RL models:

- the model-free strategy (d=1) same as the two-stage task;
- the model-free Q(1) strategy (d=2) same as the two-stage task;
- state value [2] + model-free action value [2] + bias [0] + rotation bias [0] + single-trial choice perseveration [0];
- state value [2] + model-free action value with decay [2] + bias [0] + rotation bias [0] + single-trial choice perseveration [0];
- state value [2] + model-free action value with decay [2] + motor-level model-free action value with decay [4] + bias [0] + rotation bias [0] + multi-trial choice perseveration [1] + multi-trial motor-level choice perseveration [2].

Here, we use the format of "model component [required number of dynamical variables]" (more details in [12]).

Model-based strategies. We include twelve model-based RL models:

- state value [2] + model-based [2] + bias [0] + rotation bias [0] + single-trial choice perseveration [0];
- state value [2] + model-free action value [2] + model-based [2] + bias [0] + rotation bias [0] + single-trial choice perseveration [0];
- state value [2] + model-based with decay [2] + bias [0] + rotation bias [0] + single-trial choice perseveration [0];
- state value [2] + model-free action value with decay [2] + model-based with decay [2] + bias [0] + rotation bias [0] + single-trial choice perseveration [0];
- state value [2] + model-free action value with decay [2] + model-based [2] + bias [0] + rotation bias [0] + single-trial choice perseveration [0];
- state value [2] + model-free action value [2] + model-based [2] + bias [0] + rotation bias [0] + multi-trial choice perseveration [1];

- state value [2] + model-free action value with decay [2] + model-based with decay [2] + bias [0] + rotation bias [0] + multi-trial choice perseveration [1];
- state value [2] + model-free action value with decay [2] + model-based [2] + bias [0] + rotation bias [0] + multi-trial choice perseveration [1];
- state value [2] + model-free action value with decay [2] + model-based with decay [2] + bias [0] + rotation bias [0] + multi-trial motor-level choice perseveration [2];
- state value [2] + model-based with decay [2] + bias [0] + rotation bias [0] + multi-trial choice perseveration [1] + multi-trial motor-level choice perseveration [2];
- state value [2] + model-free action value with decay [2] + model-based with decay [2] + bias [0] + rotation bias [0] + multi-trial choice perseveration [1] + multi-trial motor-level choice perseveration [2];
- state value [2] + model-free action value with decay [2] + model-based with decay [2] + motor-level model-free action value with decay [4] + bias [0] + rotation bias [0] + multi-trial choice perseveration [1] + multi-trial motor-level choice perseveration [2].

Here, we use the format of model component [required number of dynamical variables] (more details in[12]).

Model fitting

Maximum likelihood estimation

The parameters in all models were optimized on the training dataset to maximize the log-likelihood (i.e., minimize the negative log-likelihood, or cross-entropy) for the next-action prediction, defined as follows:

$$\begin{aligned} LL &= \log \Pr[\text{action sequences from one subject given one model}] \\ &= \sum_{n=1}^{N_{\text{session}}} \sum_{t=1}^{T_n} \log \Pr[\text{observing } a_t \text{ given past observations and the model}], \end{aligned} \tag{29}$$

where N_{session} is the number of sessions and T_n is the number of trials in session n.

Nested cross-validation

To avoid overfitting and ensure a fair comparison between models with varying numbers of parameters, we implemented nested cross-validation. For each animal, we first divided sessions into non-overlapping shorter blocks (approximately 150 trials per block) and allocated these blocks into ten folds. In the outer loop, nine folds were designated for training and validation, while the remaining fold was reserved for testing. In the inner loop, eight of the nine folds were assigned for training (optimizing a model's parameters for a given set of hyperparameters), and the remaining fold of the nine was allocated for validation (selecting the best-performing model across all hyperparameter sets). Notice that this procedure allows different hyperparameters for each test set.

RNNs' hyperparameters encompassed the L1-regularization coefficient on recurrent weights (drawn from 1e-5, 1e-4, 1e-3, 1e-2, 1e-1, depending on the experiments), the number of training epochs (i.e., early stopping), and the random seed (three seeds). For cognitive models, the only hyperparameter was the random seed (used for parameter initialization). The inner loop produced nine models, with the best-performing model, based on average performance in the training and validation datasets, being selected and evaluated on the unseen testing fold. The final testing performance was computed as the average across all ten testing folds, weighted by the number of trials per session. This approach ensures that test data is exclusively used for evaluation and is never encountered during training or selection.

During RNN training, we employed early-stopping if the validation performance failed to improve after 200 training epochs. This method effectively prevents RNN overfitting on the training data. According to this criterion, a more flexible model may demonstrate worse performance than a less flexible one, as the training for the former could be halted early due to insufficient training data. However, it is expected that the more flexible model would continue to improve with additional training data (e.g., see Fig. 3).

Phase portraits

Models with $d = 1$

Logit. In each trial t, a model predicts the action probabilities $\Pr(a_t = A_1)$ and $\Pr(a_t = A_2)$. We define the logit $L(t)$ (log-odds) at trial t as $L(t) = \log\left(\Pr(a_t = A_1)/\Pr(a_t = A_2)\right)$. When applied to probabilities computed via softmax, the

logit yields $L(t) = \log\left(e^{\beta o_t^{(1)}}/e^{\beta o_t^{(2)}}\right) = \beta(o_t^{(1)} - o_t^{(2)})$, where $o_t^{(i)}$ is the model's output for action $a_t = A_i$ before softmax. Thus, the logit can be viewed as reflecting the preference for action A_1 over A_2: in RNNs, the logit corresponds to the score difference $o_t^{(1)} - o_t^{(2)}$; in model-free and model-based RL models, the logit is proportional to the difference in first-stage action values $Q_t(A_1) - Q_t(A_2)$; in latent-state models, the logit is proportional to the difference in latent-state probabilities $\Pr_t(h = 1) - \Pr_t(h = 2)$.

Logit change. We define the logit change, $\delta L(t)$, in trial t as the difference between $L(t+1)$ and $L(t)$. In one-dimensional models, $\delta L(t)$ is a function of the input and $L(t)$, forming a vector field.

Stability of fixed points. Here we derive the stability of a fixed point in one-dimensional discrete dynamical systems. The system's dynamics update according to:

$$L_{\text{next}} = f_I(L), \tag{30}$$

where L is the current-trial logit, L_{next} is the next-trial logit, and f_I is a function determined by input I (omitted for simplicity). At a fixed point, denoted by $L = L_*$, we have

$$L_* = f(L_*). \tag{31}$$

Next, we consider a small perturbation ΔL around the fixed point:

$$\begin{aligned} L_{\text{next}} &= f(L_* + \Delta L) \\ &\approx f(L_*) + f'(L_*)\Delta L \\ &= L_* + f'(L_*)\Delta L. \end{aligned} \tag{32}$$

The fixed point is stable only when $-1 < f'(L_*) < 1$. Because the logit change δL is defined as $g(L) = f(L) - L$, we have the stability condition $-2 < g'(L_*) < 0$.

Effective learning rate and slope. We define the effective learning rate $\alpha(L)$ at L relative to a stable fixed point L_* as:

$$\alpha(L) = -\frac{g(L) - g(L_*)}{L - L_*} = -\frac{g(L)}{L - L_*}. \tag{33}$$

At L_*, $\alpha(L_*)$ is the negative slope $-g'(L_*)$ of the tangent at L_*. However, for general $L \neq L_*$, $\alpha(L)$ is the negative slope of the secant connecting $(L, g(L))$ and $(L_*, 0)$, which is different from $-g'(L)$.

We have

$$\begin{aligned} \alpha'(L)\Delta L &\approx \alpha(L + \Delta L) - \alpha(L) \\ &= \frac{g(L)}{L - L_*} - \frac{g(L + \Delta L)}{L + \Delta L - L_*} \\ &\approx \frac{-\alpha(L) - g'(L)}{L - L_*}\Delta L. \end{aligned} \tag{34}$$

Letting ΔL go to zero, we have:

$$\alpha(L) = -g'(L) - \alpha'(L)(L - L_*), \tag{35}$$

which provides the relationship between the effective learning rate $\alpha(L)$ and the slope of the tangent $g'(L)$.

Models with $d > 1$

In models with more dynamical variables, $\delta L(t)$ is no longer solely a function of the input and $L(t)$ due to added degrees of freedom. In these models, the state-space is spanned by a set of dynamical variables, collected by the vector $F(t)$. For example, the action value vector is the $F(t) = (Q_t(A_1), Q_t(A_2))^T$ in the two-dimensional RL models. The vector field $\delta F(t)$ can be defined as $\delta F(t) = F(t+1) - F(t) = (Q_{t+1}(A_1) - Q_t(A_1), Q_{t+1}(A_2) - Q_t(A_2))^T$, a function of $F(t)$ and the input in trial t.

Meta-reinforcement learning models

We trained meta-RL agents on the two-stage task (common transition: $\Pr(S_1|A_1) = \Pr(S_2|A_2) = 0.8$, rare transition: $\Pr(S_2|A_1) = \Pr(S_1|A_2) = 0.2$; see Fig. S13) implemented in NeuroGym[59]. Each second-stage state leads to a different probability of a unit reward, with the most valuable state switching stochastically ($\Pr(r = 1|S_1) = 1 - \Pr(r = 1|S_2) = 0.8$ or 0.2 with probability of 0.025 on each trial). There are three periods (discrete time steps) on one trial: Delay 1, Go, and Delay 2. During Delay 1, the agent receives the observation (choice state $S0$ and a fixation signal), and the reward (1 or 0) from second-stage states on the last trial. During Go, the agent receives the observation of the choice state and a go signal. During Delay 2, the agent receives the observation of state S1/S2 and a fixation signal. If the agent does not select action A1 or A2 during Go or select action F (Fixate) during Delay periods, a small negative reward (-0.1) is given. The contributions of second-stage states, rewards, and actions on networks are thus separated in time.

The agent architecture is a fully connected, gated recurrent neural network (LSTM, long short-term memory,[33]) with 48 units[24]. The input to the LSTM network consists of the current observation (state $S_0/S_1/S_2$ and a scalar fixation/go signal), a scalar reward signal of the previous time step, and a one-hot action vector of the previous time step. The LSTM network outputs a scalar baseline (value function for the current state) serving as the critic and a real-valued action vector (passed through a softmax layer to sample one action from A1/A2/F) serving as the actor. The agents are trained using the Advantage Actor-Critic RL algorithm[60] with the policy gradient loss, value estimate loss, and entropy regularization. We trained and analyzed agents for five seeds. Our agents obtained 0.64 rewards on average on each trial (0.5 rewards for chance level), close to optimal performance (0.68 rewards obtained by an oracle agent knowing the correct action).

References

[1] Allen Newell and Herbert A Simon. Computer science as empirical inquiry: Symbols and search. In *ACM Turing award lectures*, page 1975. 2007.

[2] James L McClelland, David E Rumelhart, PDP Research Group, et al. *Parallel Distributed Processing, Volume 2: Explorations in the Microstructure of Cognition: Psychological and Biological Models*, volume 2. MIT press, 1987.

[3] Joshua B Tenenbaum, Charles Kemp, Thomas L Griffiths, and Noah D Goodman. How to grow a mind: Statistics, structure, and abstraction. *science*, 331(6022):1279–1285, 2011.

[4] Brenden M Lake, Tomer D Ullman, Joshua B Tenenbaum, and Samuel J Gershman. Building machines that learn and think like people. *Behavioral and brain sciences*, 40:e253, 2017.

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

[6] Yael Niv. Reinforcement learning in the brain. *Journal of Mathematical Psychology*, 53(3):139–154, 2009.

[7] 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.

[8] Anne Gabrielle Eva Collins. Reinforcement learning: Bringing together computation and cognition. *Current Opinion in Behavioral Sciences*, 29:63–68, 2019.

[9] Paul Cisek and John F Kalaska. Neural mechanisms for interacting with a world full of action choices. *Annual review of neuroscience*, 33:269–298, 2010.

[10] Marcelo G Mattar and Máté Lengyel. Planning in the brain. *Neuron*, 2022.

[11] Ramon Bartolo and Bruno B Averbeck. Prefrontal cortex predicts state switches during reversal learning. *Neuron*, 106 (6):1044–1054, 2020.

[12] Thomas Akam, Ines Rodrigues-Vaz, Ivo Marcelo, Xiangyu Zhang, Michael Pereira, Rodrigo Freire Oliveira, Peter Dayan, and Rui M Costa. The anterior cingulate cortex predicts future states to mediate model-based action selection. *Neuron*, 109(1):149–163, 2021.

[13] Kevin J Miller, Matthew M Botvinick, and Carlos D Brody. Value representations in the rodent orbitofrontal cortex drive learning, not choice. *Elife*, 11:e64575, 2022.

[14] Robert C Wilson and Anne GE Collins. Ten simple rules for the computational modeling of behavioral data. *Elife*, 8: e49547, 2019.

[15] Carolina Feher da Silva and Todd A Hare. Humans primarily use model-based inference in the two-stage task. *Nature Human Behaviour*, 4(10):1053–1066, 2020.

[16] Blake A Richards, Timothy P Lillicrap, Philippe Beaudoin, Yoshua Bengio, Rafal Bogacz, Amelia Christensen, Claudia Clopath, Rui Ponte Costa, Archy de Berker, Surya Ganguli, et al. A deep learning framework for neuroscience. *Nature neuroscience*, 22(11):1761–1770, 2019.

[17] Guangyu Robert Yang and Xiao-Jing Wang. Artificial neural networks for neuroscientists: a primer. *Neuron*, 107(6): 1048–1070, 2020.

[18] Daniel LK Yamins, Ha Hong, Charles F Cadieu, Ethan A Solomon, Darren Seibert, and James J DiCarlo. Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the national academy of sciences*, 111(23):8619–8624, 2014.

[19] Daniel LK Yamins and James J DiCarlo. Using goal-driven deep learning models to understand sensory cortex. *Nature neuroscience*, 19(3):356–365, 2016.

[20] Christopher J Cueva and Xue-Xin Wei. Emergence of grid-like representations by training recurrent neural networks to perform spatial localization. *arXiv preprint arXiv:1803.07770*, 2018.

[21] Andrea Banino, Caswell Barry, Benigno Uria, Charles Blundell, Timothy Lillicrap, Piotr Mirowski, Alexander Pritzel, Martin J Chadwick, Thomas Degris, Joseph Modayil, et al. Vector-based navigation using grid-like representations in artificial agents. *Nature*, 557(7705):429–433, 2018.

[22] James CR Whittington, Timothy H Muller, Shirley Mark, Guifen Chen, Caswell Barry, Neil Burgess, and Timothy EJ Behrens. The tolman-eichenbaum machine: unifying space and relational memory through generalization in the hippocampal formation. *Cell*, 183(5):1249–1263, 2020.

[23] Valerio Mante, David Sussillo, Krishna V Shenoy, and William T Newsome. Context-dependent computation by recurrent dynamics in prefrontal cortex. *nature*, 503(7474):78–84, 2013.

[24] 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.

[25] Valeria Fascianelli, Fabio Stefanini, Satoshi Tsujimoto, Aldo Genovesio, and Stefano Fusi. Neural representational geometry correlates with behavioral differences between monkeys. *bioRxiv*, pages 2022–10, 2022.

[26] 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.

[27] Joshua C Peterson, David D Bourgin, Mayank Agrawal, Daniel Reichman, and Thomas L Griffiths. Using large-scale experiments and machine learning to discover theories of human decision-making. *Science*, 372(6547):1209–1214, 2021.

[28] Amir Dezfouli, Kristi Griffiths, Fabio Ramos, Peter Dayan, and Bernard W Balleine. Models that learn how humans learn: The case of decision-making and its disorders. *PLoS computational biology*, 15(6):e1006903, 2019.

[29] Mingyu Song, Yael Niv, and Mingbo Cai. Using recurrent neural networks to understand human reward learning. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 43, 2021.

[30] Paul I Jaffe, Russell A Poldrack, Robert J Schafer, and Patrick G Bissett. Modelling human behaviour in cognitive tasks with latent dynamical systems. *Nature Human Behaviour*, pages 1–15, 2023.

[31] Ryoma Hattori and Takaki Komiyama. Context-dependent persistency as a coding mechanism for robust and widely distributed value coding. *Neuron*, 110(3):502–515, 2022.

[32] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

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

[34] Scott W Linderman, Andrew C Miller, Ryan P Adams, David M Blei, Liam Paninski, and Matthew J Johnson. Recurrent switching linear dynamical systems. *arXiv preprint arXiv:1610.08466*, 2016.

[35] William Bialek. On the dimensionality of behavior. *Proceedings of the National Academy of Sciences*, 119(18):e2021860119, 2022.

[36] Surya Ganguli. Measuring the dimensionality of behavior. *Proceedings of the National Academy of Sciences*, 119(43): e2205791119, 2022.

[37] Oded Galor. *Discrete dynamical systems*. Springer Science & Business Media, 2007.

[38] Rylan Schaeffer, Mikail Khona, Leenoy Meshulam, International Brain Laboratory, and Ila Rani Fiete. Reverse-engineering recurrent neural network solutions to a hierarchical inference task for mice. *bioRxiv*, pages 2020–06, 2020.

[39] Christopher Langdon and Tatiana A Engel. Latent circuit inference from heterogeneous neural responses during cognitive tasks. *bioRxiv*, pages 2022–01, 2022.

[40] Saurabh Vyas, Matthew D Golub, David Sussillo, and Krishna V Shenoy. Computation through neural population dynamics. *Annual review of neuroscience*, 43:249–275, 2020.

[41] Mark M Churchland, John P Cunningham, Matthew T Kaufman, Justin D Foster, Paul Nuyujukian, Stephen I Ryu, and Krishna V Shenoy. Neural population dynamics during reaching. *Nature*, 487(7405):51–56, 2012.

[42] Yang Xie, Peiyao Hu, Junru Li, Jingwen Chen, Weibin Song, Xiao-Jing Wang, Tianming Yang, Stanislas Dehaene, Shiming Tang, Bin Min, et al. Geometry of sequence working memory in macaque prefrontal cortex. *Science*, 375(6581): 632–639, 2022.

[43] Jay Hennig, Sandra A. Romero Pinto, Takahiro Yamaguchi, Scott W. Linderman, Naoshige Uchida, and Samuel J. Gershman. Emergence of belief-like representations through reinforcement learning. *bioRxiv*, 2023. doi: 10.1101/2023.04.04.535512. URL `https://www.biorxiv.org/content/early/2023/04/07/2023.04.04.535512`.

[44] Manuel Molano-Mazón, Yuxiu Shao, Daniel Duque, Guangyu Robert Yang, Srdjan Ostojic, and Jaime de la Rocha. Recurrent networks endowed with structural priors explain suboptimal animal behavior. *Current Biology*, 2023.

[45] Pamela Lyon and Franz Kuchling. Valuing what happens: a biogenic approach to valence and (potentially) affect. *Philosophical Transactions of the Royal Society B*, 376(1820):20190752, 2021.

[46] P Read Montague, Raymond J Dolan, Karl J Friston, and Peter Dayan. Computational psychiatry. *Trends in cognitive sciences*, 16(1):72–80, 2012.

[47] Quentin JM Huys, Michael Browning, Martin P Paulus, and Michael J Frank. Advances in the computational understanding of mental illness. *Neuropsychopharmacology*, 46(1):3–19, 2021.

[48] Adrian Valente, Jonathan W Pillow, and Srdjan Ostojic. Extracting computational mechanisms from neural data using low-rank rnns. In *Advances in Neural Information Processing Systems*.

[49] Boyuan Chen, Kuang Huang, Sunand Raghupathi, Ishaan Chandratreya, Qiang Du, and Hod Lipson. Automated discovery of fundamental variables hidden in experimental data. *Nature Computational Science*, 2(7):433–442, 2022.

[50] Pantelis R Vlachas, Georgios Arampatzis, Caroline Uhler, and Petros Koumoutsakos. Multiscale simulations of complex systems by learning their effective dynamics. *Nature Machine Intelligence*, 4(4):359–366, 2022.

[51] Yuhuai Wu, Saizheng Zhang, Ying Zhang, Yoshua Bengio, and Russ R Salakhutdinov. On multiplicative integration with recurrent neural networks. *Advances in neural information processing systems*, 29, 2016.

[52] Francesca Mastrogiuseppe and Srdjan Ostojic. Linking connectivity, dynamics, and computations in low-rank recurrent neural networks. *Neuron*, 99(3):609–623, 2018.

[53] Manuel Beiran, Alexis Dubreuil, Adrian Valente, Francesca Mastrogiuseppe, and Srdjan Ostojic. Shaping dynamics with multiple populations in low-rank recurrent networks. *Neural Computation*, 33(6):1572–1615, 2021.

[54] Alexis Dubreuil, Adrian Valente, Manuel Beiran, Francesca Mastrogiuseppe, and Srdjan Ostojic. The role of population structure in computations through neural dynamics. *Nature neuroscience*, 25(6):783–794, 2022.

[55] Matan Fintz, Margarita Osadchy, and Uri Hertz. Using deep learning to predict human decisions and using cognitive models to explain deep learning models. *Scientific reports*, 12(1):4736, 2022.

[56] Nathaniel D Daw, Samuel J Gershman, Ben Seymour, Peter Dayan, and Raymond J Dolan. Model-based influences on humans' choices and striatal prediction errors. *Neuron*, 69(6):1204–1215, 2011.

[57] Thomas Akam, Rui Costa, and Peter Dayan. Simple plans or sophisticated habits? state, transition and learning interactions in the two-step task. *PLoS computational biology*, 11(12):e1004648, 2015.

[58] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[59] Manuel Molano-Mazon, Joao Barbosa, Jordi Pastor-Ciurana, Marta Fradera, Ru-Yuan Zhang, Jeremy Forest, Jorge del Pozo Lerida, Li Ji-An, Christopher J Cueva, Jaime de la Rocha, et al. Neurogym: An open resource for developing and sharing neuroscience tasks. 2022.

[60] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.