Curriculum learning inspired by behavioral shaping trains neural networks to adopt animal-like decision making strategies

David Hocker[1*], Christine M. Constantinople[1†], and Cristina Savin[1,2†]

[1]Center for Neural Science, New York University, New York, NY 10003, USA
[2]Center for Data Science, New York University, New York, NY 10011, USA
[*]Corresponding author
[†]Co-senior authors

January 13, 2024

1 Abstract

Recurrent neural networks (RNN) are ubiquitously used in neuroscience to capture both neural dynamics and behaviors of living systems. However, when it comes to complex cognitive tasks, traditional methods for training RNNs can fall short in capturing crucial aspects of animal behavior. To address this challenge, we leverage a commonly used (though rarely appreciated) approach from the experimental neuroscientist's toolkit: behavioral shaping. Taking as target a temporal wagering task previously studied in rats, we designed a pretraining curriculum of simpler cognitive tasks that are prerequisites for performing it well. These pretraining tasks are not simplified versions of the temporal wagering task, but rather reflect relevant sub-computations. We show that this approach is required for RNNs to adopt similar strategies as rats, including long-timescale inference of latent states, which conventional pretraining approaches fail to capture. Mechanistically, our pretraining supports the development of key dynamical systems features needed for implementing both inference and value-based decision making. Overall, our approach addresses a gap in neural network model training by incorporating inductive biases of animals, which is important when modeling complex behaviors that rely on computational abilities acquired from past experiences.

2 Introduction

Recurrent neural networks (RNN) are frequently used in neuroscience to study the role of coordinated network activity in supporting trained behaviors, with the hope of generating hypotheses about neural mechanisms that can be tested in animal models [1]. However, as cognitive tasks increase in complexity, training RNNs becomes harder. In extreme cases, such as in reinforcement learning tasks with salient suboptimal solutions, sparse rewards, or long temporal dependencies, RNNs may struggle to learn optimal task solutions [2]. Importantly, there are no guarantees that the solutions found by RNNs in complex tasks will reflect the same computational strategies that are employed by animals, limiting their utility as models for neuroscience. Many factors, including training history and innate biases due to evolution [3, 4] can contribute to such mismatch. Animals learn many cognitive tasks throughout their lifetime, and previously-learned tasks can impact their behavior in new ones [5–7]. Moreover, early experiences can shape cognitive abilities [8–11]. Therefore, appropriately training RNNs to reflect relevant prior experiences (due to explicit experimental shaping or otherwise) may increase their ability to model complex animal behaviors and their underlying neural substrates.

Machine learning has conventionally addressed both difficult training and mismatches in prior experience through *pretraining*, a process in which neural networks are first trained on variants of the task, to help improve final target task performance. Two of the most commonly used approaches are curriculum learning, and meta-learning. Curriculum learning (Fig. 1a) first trains on simpler versions of the target task (e.g., smaller input dimensionality, search spaces, actions spaces, etc.), and then increases complexity until the RNN is ready to learn the target task [12, 13]. Meta-learning (Fig. 1b) instead trains networks on a family of tasks that have a common underlying structure [14–16] (e.g. a collection of binary classifications of images). After training on these related tasks, performing a novel task from the same family requires considerably less training to reach desired performance. The working hypothesis behind meta-learning is that in order to perform well on many related tasks, neural networks must learn to represent and leverage their shared structure, which primes them for faster learning on a novel task with similar structure. Importantly, both curriculum learning and meta-learning always train within the context of the desired target task.

Training animals to perform cognitive tasks presents similar challenges: complex behaviors require long training times and animals may often adopt suboptimal strategies. To address them, behavioral scientists have developed their own pretraining procedures, in the form of *behavioral shaping* [17]. Shaping is the process by which an animal is gradually exposed to (and thus learns) the core components of a behavioral task, such as whether or not to interact with effectors like levers or ports, or associate sensory cues with rewards or abstract rules, based on reinforcement. Shaping is ubiquitous in behavioral neuroscience, and is more than just acclimating the animal to a training environment (i.e., a behavioral chamber): it first trains animals on fundamental cognitive elements that are required to engage in the final task. Importantly, and in contrast to the machine learning approaches, shaping often involves training outside of the context of the target task. For example, many decision-making tasks require animals to make judgments about a stimulus after evaluating it for a period of time defined by the experimenter. The shaping procedure might require animals to gradually increase how long they fixate during stimulus presentation. This training is not expressly about decision-making, but implicitly encourages stimulus integration and/or working memory. These cognitive operations, however, are generally useful components of performing a decision-making task well.

From this perspective, complex behavioral tasks can viewed as the composition of simpler, core tasks that each require specific computations. One implicit aspect of behavioral shaping is that animals learn to perform these core computations before training on the final task. Inspired by this perspective, we borrowed ideas from behavioral shaping to create a novel pretraining paradigm which we call *kindergarten curriculum learning* (Fig. 1c). The name reflects the building of expertise on a set of basic "building block tasks" (kindergarten) that are out-of-context, but teach core computations that facilitate target task performance. It is similar to CL in that the numbers of core cognitive tasks to be performed increases over time, and to meta-learning in that the out-of-context tasks are assumed to share underlying computation with the target. It is unique in that it emphasizes a compositional view of complex tasks as a combination of simpler computational elements, which can be pre-trained with simpler means.

We demonstrate how kindergarten CL can successfully train deep recurrent neural networks (RNNs) on a complex cognitive task that was previously performed in rats, resulting in animal-like behavioral strategies. By formalizing the task as a Markov decision process (MDP), we also show that the kindergarten trained behaviors reflect an optimal strategy for this task, whereas classical forms of CL learn different, suboptimal behavioral strategies. To determine how different pretraining strategies influence RNNs' task solutions, we examine the dynamics of RNN activity over the course of training. We find that there is a diversity in the types of dynamical systems features present in well-trained networks (*i.e.*, point attractors, saddles, and line attractors),though typically well-performing networks possess attractor dynamics for latent states with strong beliefs, and saddles for uncertain states with semi-stable beliefs. Moreover, kindergarten CL produces networks with richer dynamics than those trained on classical CL which can be exploited for learning the target task. Finally, we demonstrate that kindergarten CL works by virtue of training on subtasks with related computational requirements, not by simply by training on a wide variety of tasks: Networks trained on a curriculum of tasks with different computational elements than the target task do not assist the networks in learning optimal strategies.

3 Results

3.1 Behaving animals use inference in a temporal wagering task

We investigated curriculum learning procedures in the context of a temporal wagering task performed by rats [18]. In this task (Fig. 2a), rats initiate trials by poking into a center port, then hear an auditory tone, the frequency of which denotes the volume of a water reward. A side port is randomly chosen as the rewarded side on each trial, indicated by an LED. On each trial, a probabilistic reward (p=0.8) is delivered with a time delay drawn from an exponential distribution (mean λ=2.5s). The side light turns off after this delay, signalling to the rat that reward is available. If the rat chooses the other port before reward delivery, the trial is terminated (an "opt-out" trial), and the center light immediately turns on again to signal that a new trial can begin. Most trials are rewarded, but we intentionally withheld reward on a subset of trials (p=0.2), termed "catch trials," to see how long rats were willing to wait before exercising the opt-out choice. There are 5 reward volume offers in the task (5, 10, 20, 40 and 80 μL), as well as a hidden block structure that is not cued to the rats (Fig. 2b). After 40 completed trials, there is a transition from a mixed block, which contains all 5 offer values, into a high block (red) or low block (blue) that contain a subset of offers. Importantly 20 μL offers are present in all blocks. Each session starts with a mixed block, randomly choosing either a high or low block, and then deterministically alternating between mixed and high/low blocks over the session. Block transitions are pseudo-probabilistic, since on some trials rats fail to maintain fixation in the center port during the auditory stimulus delivery ("violation trials") and 40 successful trials must be completed in each block. The animal must repeat the trial following a violation.

On catch trials, rats are forced to exercise the opt-out option. Their wait times on these trials are sensitive to both the size of the reward offer and the latent block structure of the task (Fig. 2c). Rats wait longer for larger reward offers, and also modulate their wait times based on the reward statistics of the current block. Similar sensitivity to local reward statistics has also been observed in humans in "willingness-to-pay" paradigms [19]. Additionally,

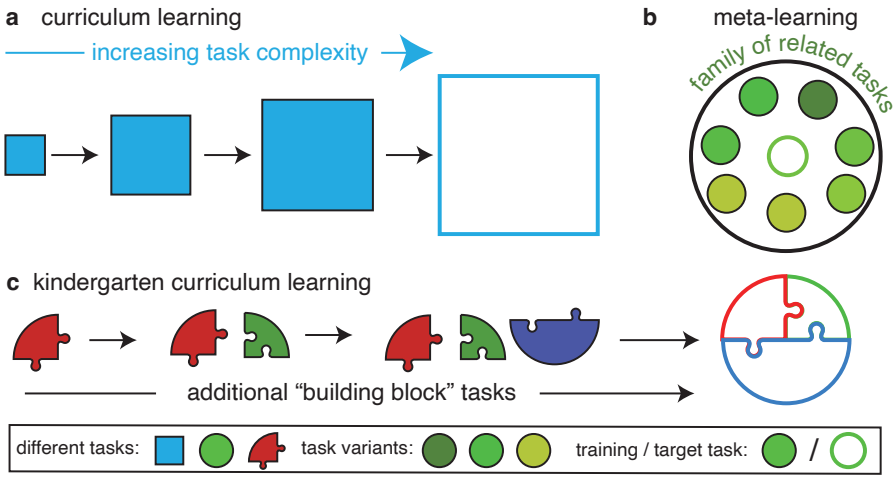

Figure 1: Shaping procedures in neural networks and behaving animals. **a)** Curriculum learning. Neural networks begin with simple versions of some target task (*e.g.*, small action space or small stimulus input space), then sequentially increase complexity of the task until the reaching the complexity of the target task. **b)** Meta-learning trains on a family of related tasks that all share some underlying structure, such that solving many related tasks effectively will utilize the shared structure. This learned structure can be leveraged on never-seen tasks that contain this shared structure. **c)** Kindergarten CL, a proposed neural network training procedure inspired by behavioral shaping. Target tasks are decomposed into nominal, "building block" computations that are simpler in nature than the target task, but necessary for efficient task performance. Kindergarten tasks are out of context of the final task, but contain the same core computation. Solid shapes mark tasks used in training, open shapes target tasks. Different shapes mark different computations, while same shape in varying colors marks variants of that task.

there is often an asymmetry in the magnitude of change to wait times for 20 μL offers when comparing low vs high blocks: the bias in wait times for low blocks relative to mixed blocks is stronger than the bias of high block offers. To quantify behavioral sensitivity to the blocks, we calculated a ratio of wait times for 20μL offers in high vs. low blocks. Wait time ratios less than 1 indicate that rats wait less time for 20 μL in a high block compared to a low block. Across a large population (N=291 animals), we found that rats modulate their wait times based on the block context (Fig. 2d). Computational modeling and detailed analysis of behavior indicated that this state-dependent behavior relies on an inferential strategy, in which rats infer the most likely state based on the current offer value and their prior beliefs about the current reward block [18]. This inference-based strategy accounts for aspects of the rats' behavior, including the behavioral dynamics at state transitions (Fig. 2e). Indeed, regressing wait times against previous reward and block over training shows that the block regressor increases with the same time-course as the behavioral sensitivity to blocks, suggesting that contextual sensitivity reflects learned knowledge of block structure (Fig. 2f). We previously modeled the inferential strategy using Bayes' rule to estimate the block probabilities based on previous rewards and knowledge of the task structure to predict the probability of being in each block on a trial-by-trial basis (Fig. 2g).

3.2 Optimal task behavior

Many rats exhibit linear sensitivity to log reward offers and a sensitivity to the reward blocks, so we next investigated if these behaviors reflected an optimal decision making strategy for this task. To do so, we formulated the task as a Markov Decision Process (MDP), in which agents must either wait for a reward, or opt out of the trial (Fig. 2h). We used reinforcement learning to determine optimal actions in this MDP, similar to the approach in [20]. In this formulation, agents decide between "wait" and "opt out" actions. Waiting can either transition to the same state, or end in a terminal "hit" state where reward is delivered. Opting out deterministically ends the trial, with a reward penalty. Every time a wait action is performed, there is a time penalty k. The probability of being rewarded at each step is pulled from the exponential delay distribution $p(t)$, and the probability of being rewarded on each trials is p_r. For an exponential distribution $p(t) = \lambda^{-1}\exp(-t/\lambda)$, the optimal wait time t^* is log-linear in reward offer R as

$$t^* = \lambda\left(\log[Rp_r] - \log[k\lambda]\right) \tag{1}$$

In foraging theory, wait time penalties are often interpreted to be related to the opportunity cost, or the average reward rate of the environment, meaning that wait time decisions should incorporate the value of potential rewards being missed out on by continuing to wait [21, 22]. In the MDP formulation here, this implies $k = \rho\Delta = \mathbb{E}[\tilde{R}^{(B)}]t$, where the expectation averages reward of an environment, B. In context of the wait time task, the low, mixed,

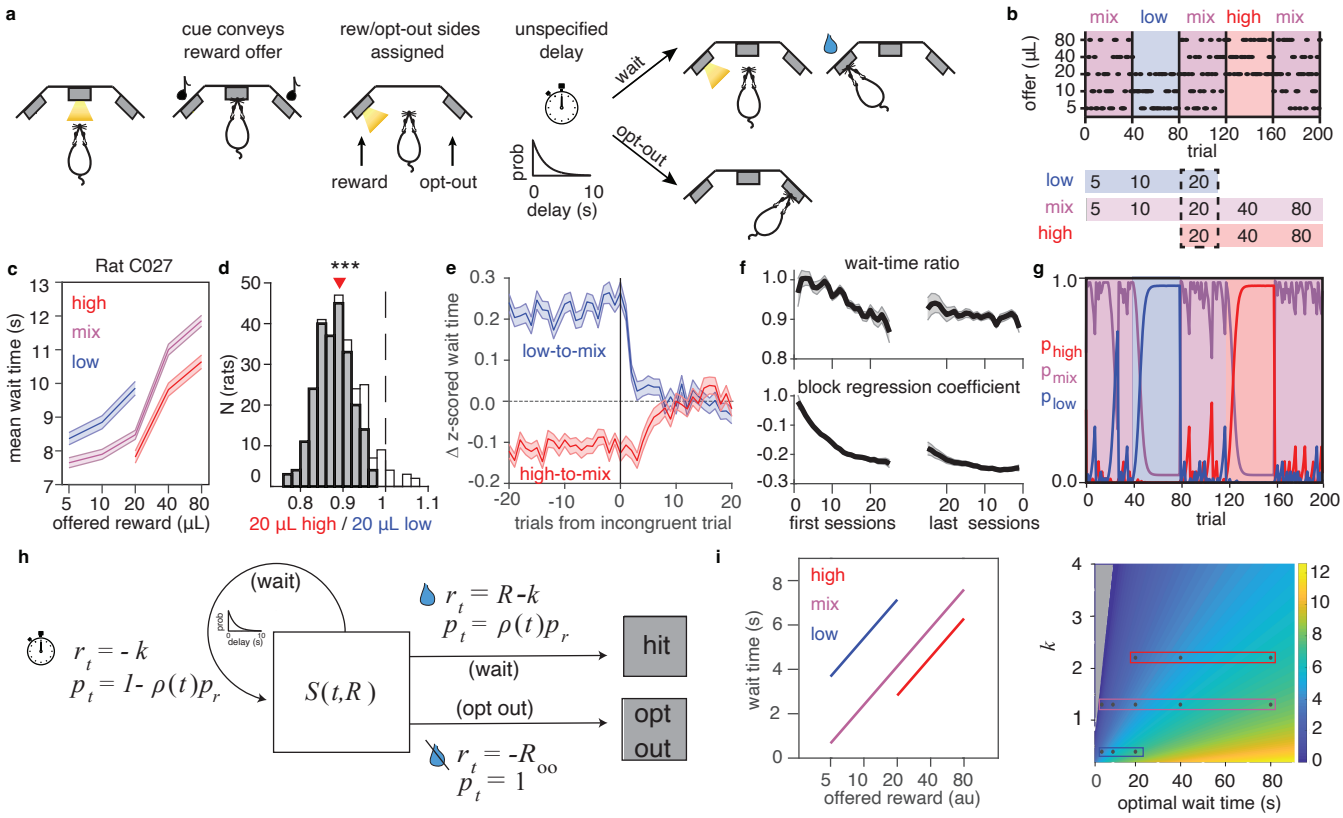

Figure 2: Rodent behavior in temporal wagering task. **a)** Temporal wagering task. Rats wait for known reward offers conveyed by an auditory tone, but with an uncertain reward delay. Rats may choose to wait for reward at lit-up side port until water arrives, or poke into the other port to terminate the trial and begin a new one. **b)** Latent block structure of task that is uncued to rats. After every 40 successful trials a transition from a mixed block (purple) containing all potential reward offers moves into a block with a subset of the highest (red) or lowest (low) offers. **c)** Example wait time behavior on catch trials where reward is withheld. **d)** Distribution of wait time ratios high vs. low for 20μL offers on catch trials. *** : $p << 0.001$, (Wilcoxon signed rank test). Histogram includes rats with significantly different wait-time ratios (shaded, p ¡ 0.05, rank sum test) and insignificant wait-time ratios (unshaded, p ¿ 0.05, rank sum test) **e)** Trial-by-trial changes in wait times across a block transition, averaged over rats; error bars mark s.e.m., colors reflect transitions from either a high (red) or low (block) block into a mix block, aligned to first incongruent trial that violates beliefs about the previous block. **f)** Time-course of the mean wait time ratio (top) and mean regression coefficient for block in a linear regression of wait time (bottom) for catch trials. Left: the first 20 session after block structure is introduced in the task; right: last 20 sessions of training. **g)** Predicted block probabilities of a Bayes-optimal model of the task. Background colors denote true block. **h)** MDP model of task, showing actions, state-transfer probabilities p, and rewards r for each state-action pair. **i)** Optimal wait times for three different wait time penalties (left). Wait times for all wait time penalties (right). Colored boxes denote k for optimal wait times in left plot. Gray region denotes negative wait time based on Eq. 1. In panels D,E and F, the mean is over rats (N=291), and error bars mark s.e.m.

and high blocks are distinct reward environments, and Eq. 1 then describes qualitative optimal behaviors in this task: rewards should be linear in log-offer, with a bias to wait longer amounts of time in reward-sparse environments (Fig. 2i). Moreover, an asymmetry in the bias of wait times for high vs. low blocks is also predicted (derivations and further details are provided in the Suppl. Material).

In sum, based on the MDP formulation of our temporal wagering task, modulating wait time with linear sensitivity to the log of a reward offer is an optimal behavior. If agents further try to optimize long-term rewards in which the time penalty is the opportunity cost of the environment, then waiting longest in low blocks, followed by mixed and the high blocks is also an optimal behavior. Importantly, rats display *all* of these hallmarks (Figs. 2C-D), which suggests that they are using a qualitatively optimal strategy to perform the temporal wagering task.

3.3 Deep meta-reinforcement learning agents mimic rodent behavior

In order to study the impact of training on behavioral performance, we modeled rats' behavior using deep recurrent neural networks (RNNs) trained with meta-reinforcement learning (meta RL) methods that can capture aspects of both animal behavior and electrophysiological recordings during complex tasks (Fig. 3a) [23]. The RNN architecture is a two-layer LSTM; the first layer models OFC activity that has been implicated in state representation and inference [24–26], while the second layer models striatum activity, an area associated with action selection and representing decision variables in actor-critic frameworks [27]. Meta-reinforcement learning is an Actor-Critic approach that trains networks to generate optimal decision policies π_t and value function estimates V_t, but does so in a way that allows trial-by-trial learning to occur through persistent dynamical activity, as opposed to continual parameter updating of the connection weights between units. The key architectural ingredient of teaching this network to generate an internal RL procedure via its dynamics (the "meta" component) is to provide the network with explicit feedback about the past action taken, and last reward received [23]. In our two-layer architecture, this feedback is provided only to the OFC layer.

We trained RNN agents on a simplified variant of the wait time task (Fig. 3 b). The task still requires an agent to wait for known rewards with an uncertain delay drawn from an exponential delay distribution, but i) abstracts away left- and right-sided actions, ii) omits the requirement to persevere in a center port while stimulus information is provided, and iii) modifies the inter-trial interval to be variable and stochastic in length, though out of the control of the agent. At every time point ($\Delta t = 50$ms), the agent will make a decision based upon its stochastic decision policy. By omitting the requirement to fixate during stimulus presentation, noise-free inputs indicating the start of a trial and the reward offer (on a log scale) were provided to the RNN at a single time point. The RNN agents were trained with a modified deep meta-RL loss adapted from [23], where we included auxiliary costs that were associated with simpler cognitive tasks that are also utilized in the target wait time task (see Methods). These include i) supervised learning of a memory task that must persistently output a stimulus input until another is provided, ii) a counting task that counts time until a new trial starts, iii) an integrator of stimulus input that can be used to keep track of average reward, and iv) a classification task to infer latent states based solely on stimulus information (see Fig.3c, bottom). We first trained the RNN on these "kindergarten" tasks (full description of curriculum provided in Suppl. Fig. S1), then introduced progressively harder variants of the target wait time task while continuing to regularize the loss with auxiliary costs related to the kindergarten tasks (Fig. 3c, top).

RNNs trained with our kindergarten curriculum successfully learned to perform the temporal wagering task, exhibiting behavioral wait times on catch trials that resembled rat behavior: Agents waited longer amounts of time for larger offers, and had wait times that were sensitive to the hidden block structure (Fig. 3d), and additionally had accurate block estimates that reflected the true block (compare Fig. 3h to Fig. 2g). RNNs trained with kindergarten CL had strong block sensitivity across the population, as quantified by the wait time ratio, whereas RNNs without this training generally failed to become sensitive to the blocks in a manner consistent with rat behavior (Fig. 3e). Finally, when comparing the time-course of the wait time ratio and block sensitivity over training, we found that RNNs displayed similar learning dynamics to rats (Fig. 3g). Initially, RNNS do not have different wait times across blocks, nor do regressions of wait times have strong contributions from block regressors. Over training, as wait times become different across blocks for $20\mu L$ offers, the block regressor becomes stronger, suggesting that knowledge of block structure is driving the difference in wait times as in rats (compare to Fig. 2f). In summary, deep meta-RL agents trained with kindergarten CL display similar qualitative behavior on the wait time task as rats, have similar inferential strategies, and learn to become sensitive to latent structure that varies on long timescales in the same manner as animals trained on the same task.

3.4 Kindergarten curriculum learning supports near optimal task performance

We next tested the extent to which this structured curriculum was necessary to mimic rat behavior, and whether or not the behavior displayed by RNN agents and rats reflected an optimal decision-making strategy. We specifically compared the performance of networks trained with the out-of-context form of kindergarten curriculum learning to classical forms of curriculum learning that only build up task complexity within the context of the target task, as

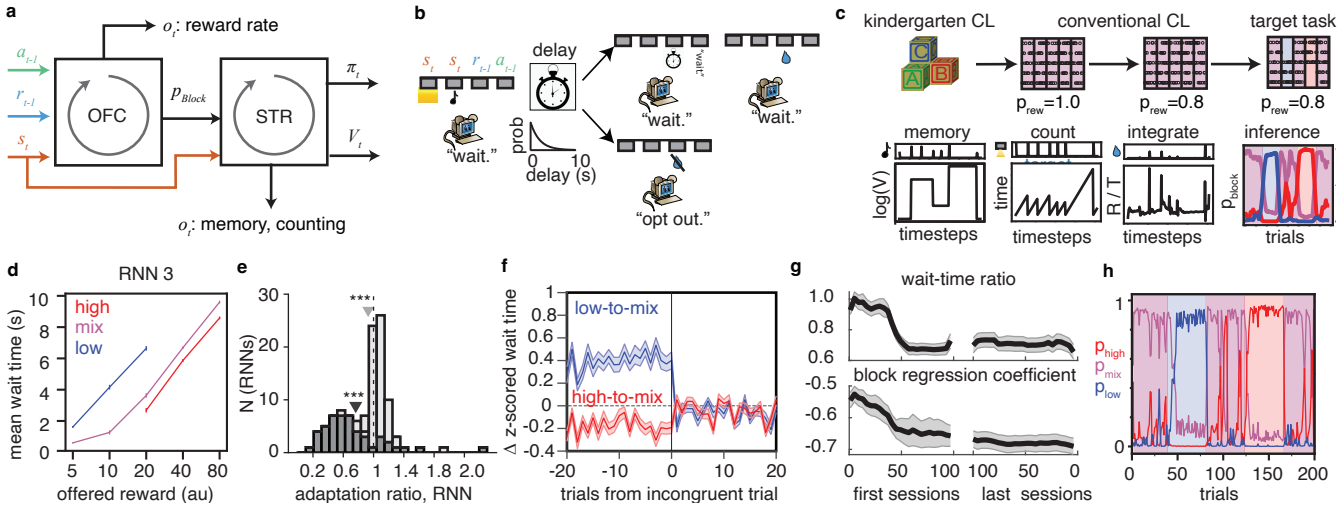

Figure 3: Deep meta RL agents can learn the temporal wait time task. **a)** Schematic of deep meta-RL architecture. An LSTM-based RNN contains two layers, modeling orbitofrontal cortex (OFC) and striatal (STR) activity. Inputs include trial stimulus information, and previous reward and action history. For the wait time task, the STR layer outputs the decision policy and value function. Outputs o are used for auxiliary costs. In particular p_{block} outputs from OFC are trained to infer the current block, and are sent as inputs to STR units. **b)** Simplified temporal wagering tasks for RNN agents. **c)** Kindergarten curriculum learning. (top) Supervised learning tasks outside the context of the wait time task are performed first, followed by progressively harder variants of the target wait-time task. (bottom) Kindergarten tasks include tasks to encourage working memory, maintaining an internal estimate of time, integrating input stimuli, and inferring latent state **d)**. Example wait time performance of RNN agent. **e)** Wait-time ratio of population of $N = 113$ RNNs. Dark bars denote $N = 46$ RNNs trained with kindergarten CL, and light bars denote networks with simpler CL strategies (including no CL). Dark gray arrow is mean of kindergarten CL population, and light gray arrow is mean of entire population. ***: $p << 0.001$, Wilcoxon signed rank test. **f)** RNN wait-time dynamics, as in Fig 2e. **g)** Wait-time ratio (top) and regression coefficient for block in a regression of wait time (bottom) over training (see Methods). **h)** p_{block} during the wait time task for network in **e**. Same color convention as Fig. 2g.

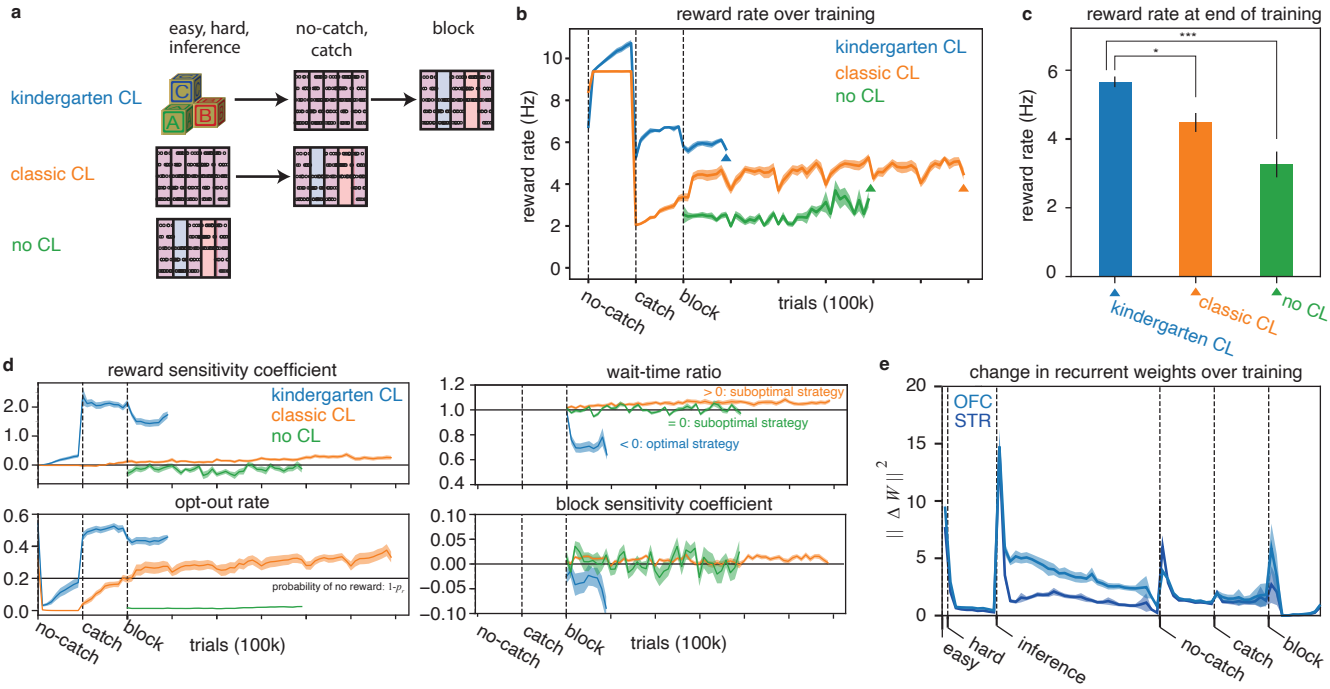

Figure 4: Performance of pretraining methods on the temporal wagering task. **a)** Schematic of pretraining methods. Kindergarten CL first trains on out-of-context supervised learning tasks, followed by a classical curriculum learning approach of training on deterministic rewards in mixed blocks only ("no-catch" training), then mixed block trials with stochastic rewards ("catch" training). Finally, blocks are introduced ("block" training). See also Suppl. Fig. S1. Classic CL omits kindergarten training, and no CL training only trains on the target task with block structure. **b)** Average reward rate over training for each method. **c)** Reward at end of training stage, corresponding to colored triangles in **b**. *: $p = 0.019$, ***: $p = 2 \times 10^{-6}$, rank-sum test. **d)** Evolution of different behavioral features over training. Reward sensitivity coefficient (left, top) is the slope of a linear fit to the wait time vs. the log of reward offer. Opt-out rates (left, bottom) reflect the proportion of trials that an agent chose the opt-out option, with the optimal rate, given by the catch probability, in black line. The wait-time ratio (right, top) compares average wait times for a 20 μL offer in high vs. low blocks , optimally ¡ 1. Block sensitivity coefficients (right, bottom) are the coefficients in a linear regression of wait time on catch trials (see Methods). **e)** Change in the LSTM recurrent hidden-layer weights over training for kindergarten CL (see Methods). In all plots, error bars denote sem over RNNs.

well as directly training on the task without any prior training (Fig. 4a). We compared the reward rate during each phase of the task, and found that even when controlling for total amount of training steps, networks trained with kindergarten CL outperformed both classical CL and networks trained directly on the target task (Fig. 4b). To trace the root of these differences, we investigated key features of the corresponding behavior (Fig. 4c). We found that in absence of CL training, networks adopt a simple, suboptimal strategy, where agents simply wait until a timeout penalty occurs (Suppl. Fig. 3), demonstrated by a low sensitivity to reward offer and block, and with opt-out rates below the true catch probability of the task. Classical CL was able to display some mild sensitivity to the reward offer, though not nearly as strongly as kindergarten CL. Additionally, these agents did not learn to opt out as frequently. Importantly, we found that the wait-time ratio in these networks was reversed when compared to rats and kindergarten CL, meaning that these networks waited longer for $20\mu L$ in high blocks. A reversed modulation of wait time with block is suboptimal in the MDP framework. Taken together, we find that RNNs and rats display the qualitative patterns of behavior that are consistent with the optimal strategy in this task, and that kindergarten curriculum learning is sufficient to replicate this behavior in RNN agents.

3.5 Kindergarten CL produces distinct dynamical task solutions

To determine the neural substrate for learning benefits of kindergarten CL, we investigated how it changes networks over training (Fig. 4d). We calculated the average change in recurrent connection weights over learning (see Methods), and found that large changes in network structure occurred at the beginning of each phase of training for any form of training, as networks reorganize from a random initial state to create low-dimensional manifolds supporting structured behavior [28]. Unexpectedly, we found that kindergarten CL had a dramatic reorganization of the network during the kindergarten phases of training, in particular when learning to perform inference of latent states.

It is interesting that the largest change in network structure occurred *after* the initial phase of training, which

may reflect the development of computationally relevant dynamical systems features. In the context of our task, those features should support computations that update and maintain information at multiple timescales: i) long timescales for maintaining and modulating beliefs about latent states, and ii) shorter timescales for deciding when to opt out of a trial. A natural form of dynamics that can support maintenance and integration of information at long timescales are "slow points," which allow network population activity to persist and influence behavior [29]. Moreover, "fixed points" —the extreme version of slow points— are known to support decision-making on shorter timescales by allowing for selection between a discrete set of behavioral actions [30–32]. Thus, we identified and analyzed the slow dynamical systems features of RNN dynamics, determined as local optima of the network's kinetic energy (see Methods) and characterized the qualitative form of each feature (e.g., point attractors, saddles, line attractors) by local linearization of the dynamics (see Methods) to assess their potential computational relevance.

First, we investigated maintenance of long-timescale latent beliefs in the OFC layer of many well-trained networks trained with kindergarten CL. While we found variability in the qualitative nature of the network dynamics across agents, the OFC dynamics generally displayed a common motif involving three slow features (Fig. 5a): two stabilizing features such as a fixed point and line attractor, as well as a semi-stable feature (saddle) separating them. Of the networks that learned the task in a rat-like way (29 of 45 RNNS), a large proportion (70% , or 20 of 29 RNNs) displayed this common motif. Conversely, the networks that did not learn a rat-like strategy (16 of 45 RNNs) did not display the same motif (88%, or 14 of 16 RNNs). Using the OFC layer's output that reports block confidence, we identified regions of state space in which the network was highly confident of being in a given block. By this delineation, the network seems to learn a representation that partitions state space in three contiguous sub-regions based on block confidence. Importantly, the stabilizing features lay in areas of low-block and high-block confidence, respectively, whereas the semi-stable feature is located in the mixed-block region. While we noted equal probability of locating a line or point attractor in high blocks, we always a found point attractor in the low block int his motif.

We next analyzed how latent beliefs are updated using transient offer and reward history information. In single trials with high probability of being in a given block, the network maintained beliefs by revolving only within their respective regions in state space. New congruent evidence introduces within attractor basin perturbations, so no change in inferred block identity. In contrast, when beliefs change due to incongruent evidence signaling block transitions (Fig. 5b, left), this triggers shifts in the state space across regions (Fig. 5b, right). Specifically, when we examined the transition from a mixed block into a low block we found that these transient inputs "kicked" the activity of the network away from all of the dynamical systems features, but with a bias towards the regions of low-block confidence. We saw similar behavior for the other block transitions (Suppl. Fig. S4a-d). Overall, this suggests a dual mechanism of inference in OFC (Fig. 5d): i) beliefs of highly confident states such as low and high blocks are maintained with strong attractor dynamics, whereas states with higher overall uncertainty (mixed blocks) are supported by saddles. ii) Beliefs are updated using both offer and reward information to push activity away from attractors, but with biases that reflect the frequency of those offers in each block. This yields a concrete and testable prediction about signatures of context inference in the population dynamics in OFC of behaving animals.

We also analyzed the fast-timescale dynamics in the STR layer that support the decision to opt of trials. STR dynamics were variable across RNNs, but to a lesser degree than OFC. We often found only 1-2 dynamical systems features (mean and median or 2 features over 45 RNNs), one of which was almost always a point attractor (found in 41 RNNS out of 45 analyzed). We reasoned that since activity of the network would be drawn towards the attractor, the wait-time probability at these points may be correlated with behavior, specifically the probability of waiting given by RNN output π_t. Indeed, when visualizing the location of the fixed point attractor along the first PC of STR activity (Fig. 5d), we found that the location of the attractors for single trial offers from each block type had varying wait-time probabilities, and that the relative location of these attractors in PC space supported asymmetric wait time behavior over blocks (e.g., Fig. 3d). This makes another testable prediction about neural correlates of behavior in rats, which is that the locations of attractors within the manifold of population activity should correlate with the behavioral wait-time ratios between block types.

Finally, we analyzed how the identified dynamical systems features were learned over training (Fig. 5e). We found that networks trained with kindergarten CL built a small set of dynamical features in the early stages of kindergarten. OFC then increased its number of features during the inference kindergarten task. Once the wait time task was introduced, features were trimmed from the network's dynamic repertoire stabilizing to the final computational motifs supporting the target task. This expansion and then pruning of dynamical features over training has been previously reported in other tasks [28], and appears to be a common learning trait for attractor-based dynamics. Interestingly, while the STR layer had similar numbers of dynamical features across learning procedures, the total number of dynamical features in the OFC layer at the end of kindergarten CL training conserved a higher number of features than sub-optimal networks trained with classical CL (Fig. 5f), which may account for their differences in inferential strategy.

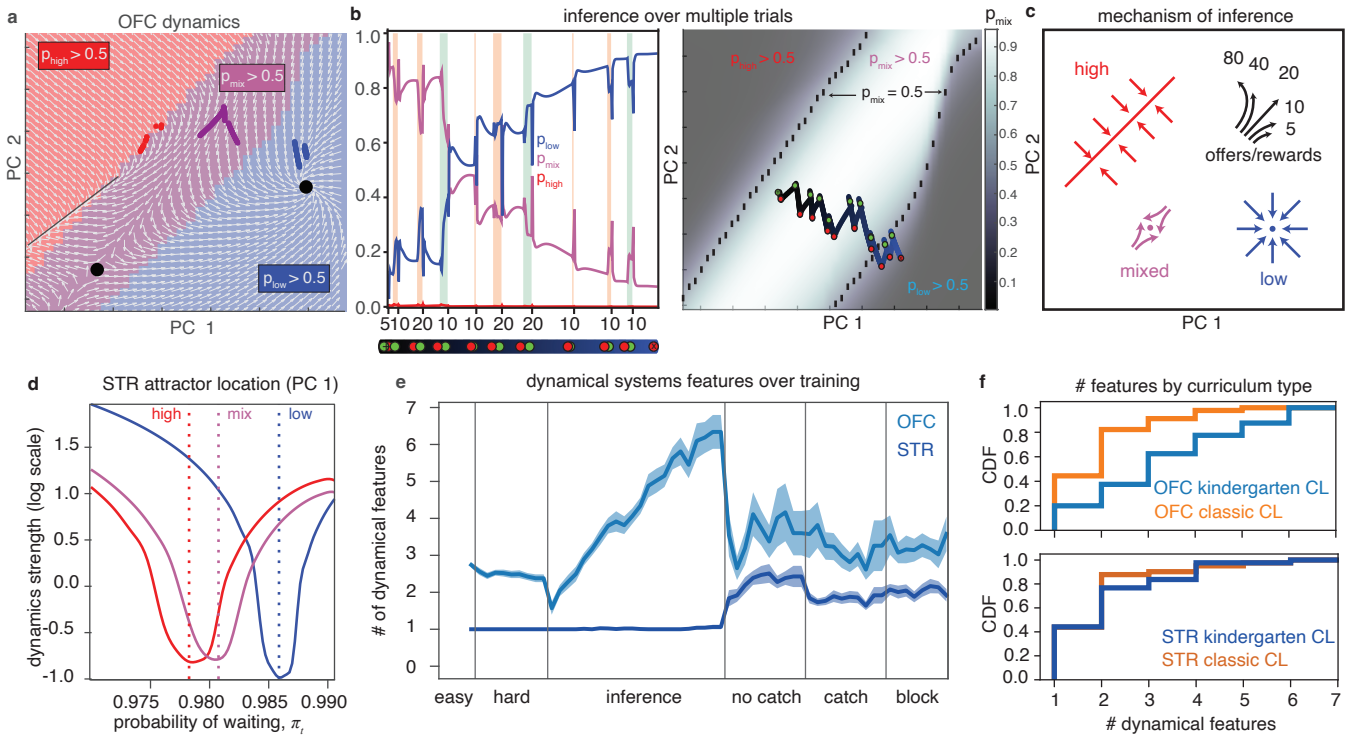

Figure 5: Dynamical systems features of kindergarten CL networks. **a)** Dynamical gradient flow fields for the OFC layer of a sample network. Background color denotes region in which p_{block} was $> 50\%$ for that respective block (same color convention as in Fig. 2-3). Colored curves denote evolution of single-trial 20μL offers from different blocks in which the agent ultimately opted out. Dynamical features are denoted in black (line attractor in high block, saddle in mix block, point attractor in low block). **b)** Updating beliefs from mix to low block. left: p_{block} over the block transition. Trial offers are on the x axis. Colored regions denote the ITI between trials when the agent either opted out (orange) or received reward (green). White regions are the waiting epoch. Progress across multiple trials is also shown as a colored line below, with start denoted as a green dot, and the ITI as a red dot. right: analogous evolution of state activity over the trials. The background color denotes p_{mix}, and the $p_{\text{mix}} = 0.5$ boundary is denoted with a dotted line. Trial progress uses the same convention from left plot. **c)** Proposed mechanism of inference in OFC. **d)** Dynamics of the STR layer along PC1, but isolated to the timepoint just before opting out for single trials from each block. PC activity was instead labeled with π_t on the x axis, and dynamics kinetic energy is plotted on log scale on y axis. **e)** Average number of dynamical systems features of training for kindergarten CL networks. Averages are over networks, and error bars are sem over networks. Error bars are sem over networks **f)** Number of dynamical systems features at the end of training for kindergarten CL and classical CL. (OFC: $p = 6 \times 10^{-4}$. STR: $p = 0.94$. KS test).

3.6 Only computationally relevant shaping tasks aid learning optimal strategies

We reasoned that kindergarten CL functions by learning dynamical systems features that support useful sub-computations and composes them for learning the target task. A consequence of this hypothesis is that irrelevant pretraining tasks would not generate the required dynamical systems structures, and thus they would not display rat-like behavior. Alternatively, pretraining irrelevant tasks might lead to optimal decision-making strategies simply by virtue of introducing overall task variability to training (Fig.6a). To test this, we introduced a delay-to-nonmatch task [33, 34] during pretraining, replacing explicit inference, and quantified the resulting networks' performance and decision-making strategy. This task requires an agent to report if two presented stimuli that are lagged in time are the same (a match) or different (a non-match). We chose this "sham" task because it feasibly replaces the classification and working memory components learned from memory and inference kindergarten. However, its classification component is distinctly different from that required in the original task set, in that the ability to do inference is not needed for ideal performance. Hence, we hypothesized that using delay-to-nonmatch as pretraining should not lead to rat-like behavior in RNNs. We studied two versions of such pretraining using delay-to-nonmatch alone or additionally including the counting and stimulus integration tasks. Both forms of pretraining underperformed compared to the kindergarten curriculum designed with task compositionality in mind (Fig. 6b). When characterizing their final wait time behavior we observed that networks adopted the same sub-optimal, reversed block sensitivity strategy of classical CL, meaning longer wait times in high compared to low blocks. From this we conclude that kindergarten tasks must involve the set of computational elements that are needed to the final target task in order to benefit training.

4 Discussion

It has been long recognised that *"Key to shaping is identifying the essential sub-components in a task."* [2]. Here we argue that the same insight can be used to shape the training of RNNs towards solutions that are both computationally more effective and a closer match to animal behavior. Our kindergarten curriculum learning decomposes a target cognitive task into its prerequisite computational elements, and first trains on simpler tasks requiring these individual computations. We demonstrated the utility of this approach using a temporal wagering task, in which kindergarten training lead to optimal solutions that mimicked the strategies adopted by rats [18]. These properties could not be replicated by traditional forms of curriculum learning or sham kindergarten tasks. Mechanistically, the benefits of kindergarten training could be traced back to its ability of building richer dynamical systems features, compared to other pretraining methods. For our target task, this higher complexity in dynamics was localized in the subcircuit subserving inference. Inference was also the task that most strongly reorganized network dynamics throughout training, consistent with its critical role for supporting latent context dependent behavior.

Rather than a complete replica of the shaping procedure used to train the animals, our specific kindergarten curriculum also included key computational elements like inference that experimentalists do not have to explicitly train on animals, but which are a reflection of general cognitive abilities that the animals have developed outside the experiment. Thus, the model captures prior inductive biases that are already present in behaving animals. In general, we aimed to operationalize an intuitive and holistic idea that if we wish to model animal behavior in a complex task, then it is necessary to posit the pre-existing computational abilities that an animal likely possesses, either due to heritable mechanisms or prior experiences. While in some cases it may be important to mimic the shaping procedure precisely, often this information may not be available in detail, so devising the curriculum with computational elements in mind seems to be more practical in general.

Our approach traces its roots to behavioral shaping but also machine learning approaches, in particular classic curriculum learning and meta-learning. We are not the first to consider the utility of behavioral shaping for RNNs. In fact, the introduction of curriculum learning in ML leaned heavily on work that brought ideas about behavioral shaping into training neural networks [35]. Nonetheless, previous work has focused on using incremental training to expand the trainable temporal horizon of RNNs [2], or as an assay to uncover learning principles on tasks where behavior is difficult to analyze, but that could be optimally performed even without curriculum learning [36]. In contrast, the complexity of our task demands some form of CL and a precise design of the curriculum. This relates to forms of meta-learning which have been used to account for structure learning [37], where invariant relationships in a task are stored and reused for more efficient learning. It is distinct from such meta-learning in its emphasis on task compositionality, while sharing its emphasis on common computational structure that generalizes across tasks.

Pretraining on sub-computations of the target task was a requirement for learning behaviorally-relevant strategies seen in rats. This likely involves deep-RL agents learning to compose dynamics elements used for the easy tasks and reuse them in the complex task. A direct composition of two behaviors into a complex one was previously demonstrated in [38], revealed by analyzing the Q values of deep RL agents on two separate components of a navigation task. There, they found that the Q value for the target task was a sum of the constituent subtask Q values, which was also observed in mice performing the same task. In our case, the segregation of functionality into distinct subcircuits was partially enforced via the choice of architecture. This choice reflects known functional

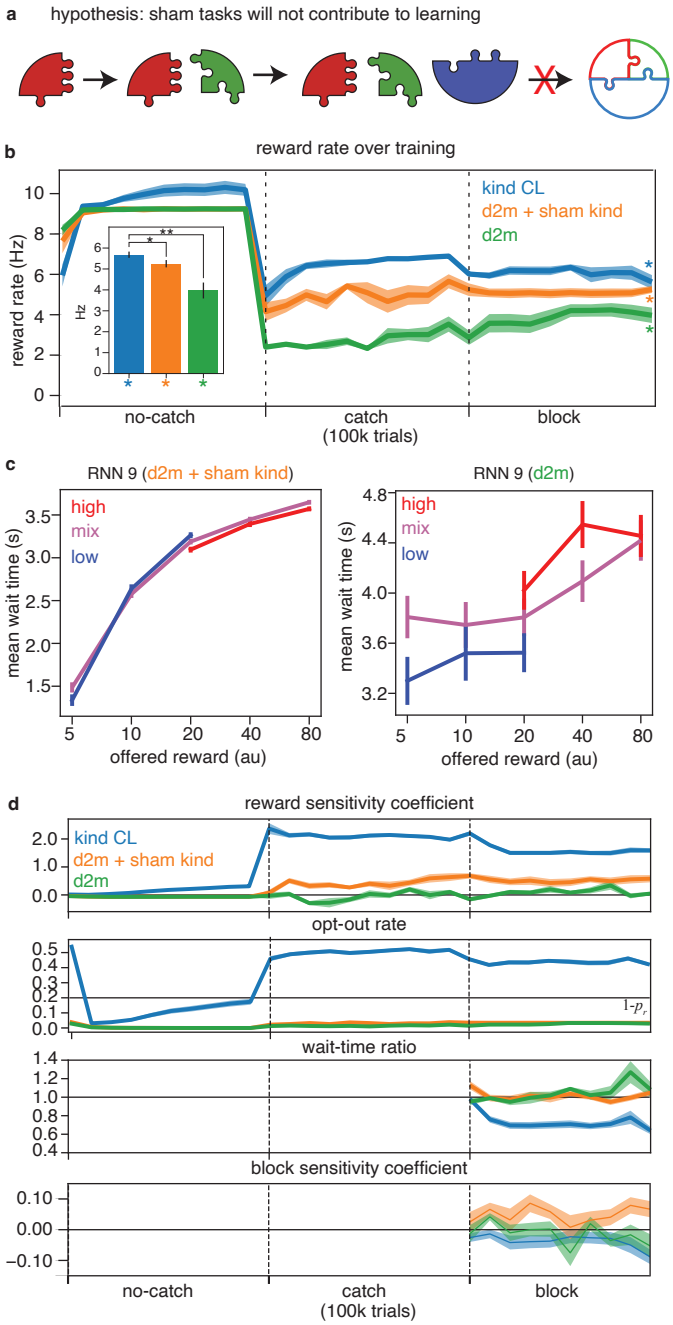

Figure 6: Curriculum learning with unhelpful kindergarten tasks. A delay-to-match/non-match task was employed as a kindergarten task (2dm), in addition to a kindergarten curriculum with the counting and stimulus integration tasks (d2m+sham kind). **a)** The hypothesis behind kindergarten curriculum learning predicts that kindergarten tasks that are unrelated to the target task, at a computational level, will not help target task performance. **b)** Average reward received over training, relative to maximum possible reward, as in Fig. 4b (*:p = 0.04, **:p=0.008, rank-sum test). **c)** Sample wait time behavior from a network trained on the unhelpful delay-to-match task during kindergarten CL. **d)** Behavioral analyses over training, as in Fig. 4c. Error bars in all plots denote sem over networks

distinctions between OFC and striatum, but complicates the analysis of dynamics compositionality. For this, it would be useful to trace the evolution of individual dynamics features over learning, to more directly characterize how features developed at one stage of learning support later computations, in the spirit of [28]. Alternatively, it may be useful to consider forms of regularization that encourage functional specialization into subcircuits [39].

One conserved aspect of learning was that deep RL agents had a point at which they introduced a large number of dynamical systems features in the the OFC layer, and then pruned them later in training, something which appears to be a common learning strategy in attractor-based networks [28]. Detailed analysis of these features showed that these dynamic features were exclusively slow points that acted like line attractors. This is consistent with previous accounts of slow points as key dynamics features driving behavior in other tasks [29], as well as the utility of line attractors for storing and updating beliefs about context.

Beyond these qualitative commonalities, the exact geometry of the solution was variable across networks, unlike the "universal" dynamical systems solutions of simpler tasks, where all networks that do well solve the task in the same way and thus are dynamically isomorphic [29, 28]. This task complexity may open interesting avenues for exploring how across-animal behavioral variability can be traced back to variability in their OFC dynamics. This is important in the context of our task, as there is a high degree of variability in wait-time behavior among rats [18].

As problems become more complex it becomes computationally advantageous to compose primitive actions into subroutines or skills, an idea formalized within the framework of hierarchical RL [40]. In this case, the decision policy is augmented to include abstract options (sequences of primitive actions) as possible actions. This also adds complexity in terms of option discovery, as an agent must first recognize that an option is even a feasible action to take in the current state. In our task, the use of inference (learned during kindergarten) does not become a relevant option until block structure is introduced. While we did not provide an action space that would explicitly allow agents to compose an inference action, followed by a decision action, the deep RL agent learned to implicitly use both in its decision policy. Locating dynamical signatures of this aspect of the learning process, in which inference is introduced into the decision policy, could be another useful neural signature to test in neurophysiological recordings.

Acknowledgements

We thank Paul Glimcher, Jane Wang, Owen Marschall and Vishwa Goudar for helpful discussions and comments on the manuscript. This work was supported by NIMH grant 1K01MH132043-01A1 (DH) and NIMH grant 1R01MH125571-01 (CMC and CS).

References

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

[2] Kai A Krueger and Peter Dayan. Flexible shaping: How learning in small steps helps. *Cognition*, 110(3):380–394, 2009.

[3] Anthony M Zador. A critique of pure learning and what artificial neural networks can learn from animal brains. *Nature communications*, 10(1):3770, 2019.

[4] Jane X Wang. Meta-learning in natural and artificial intelligence. *Current Opinion in Behavioral Sciences*, 38:90–95, 2021.

[5] Rachele McAndrew and Stephen I Helms Tillery. Laboratory primates: their lives in and after research, 2016.

[6] Syed A Chowdhury and Gregory C DeAngelis. Fine discrimination training alters the causal contribution of macaque area mt to depth perception. *Neuron*, 60(2):367–377, 2008.

[7] Charlotte Arlt, Roberto Barroso-Luque, Shinichiro Kira, Carissa A Bruno, Ningjing Xia, Selmaan N Chettih, Sofia Soares, Noah L Pettit, and Christopher D Harvey. Cognitive experience alters cortical involvement in goal-directed navigation. *Elife*, 11:e76051, 2022.

[8] Louk JMJ Vanderschuren, EJ Marijke Achterberg, and Viviana Trezza. The neurobiology of social play and its rewarding value in rats. *Neuroscience & Biobehavioral Reviews*, 70:86–105, 2016.

[9] Louk JMJ Vanderschuren and Viviana Trezza. What the laboratory rat has taught us about social play behavior: role in behavioral development and neural mechanisms. *The neurobiology of childhood*, pages 189–212, 2014.

[10] Petra JJ Baarendse, Jules HW Limpens, and Louk JMJ Vanderschuren. Disrupted social development enhances the motivation for cocaine in rats. *Psychopharmacology*, 231:1695–1704, 2014.

[11] Dorothy F Einon and MJ Morgan. A critical period for social isolation in the rat. *Developmental Psychobiology: The Journal of the International Society for Developmental Psychobiology*, 10(2):123–132, 1977.

[12] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. In *Journal of Machine Learning Research 21*, pages 87–94, 2020.

[13] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

[14] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017.

[15] Sebastian Thrun and Lorien Pratt. Learning to learn: Introduction and overview. In *Learning to learn*, pages 3–17. Springer, 1998.

[16] Harry F Harlow. The formation of learning sets. *Psychological review*, 56(1):51, 1949.

[17] Burrhus Frederic Skinner. How to teach animals. *Scientific American*, 185(6):26–29, 1951.

[18] Andrew Mah, Shannon S Schiereck, Veronica Bossio, and Christine M Constantinople. Distinct value computations support rapid sequential decisions. *Nature Communications*, 14(1):7573, 2023.

[19] Mel W Khaw, Paul W Glimcher, and Kenway Louie. Normalized value coding explains dynamic adaptation in the human valuation process. *Proceedings of the National Academy of Sciences*, 114(48):12696–12701, 2017.

[20] Sara M Constantino and Nathaniel D Daw. Learning the opportunity cost of time in a patch-foraging task. *Cognitive, Affective, & Behavioral Neuroscience*, 15:837–853, 2015.

[21] Eric L Charnov. Optimal foraging, the marginal value theorem. *Theoretical population biology*, 9(2):129–136, 1976.

[22] John M McNamara and Alasdair I Houston. Optimal foraging and learning. *Journal of Theoretical biology*, 117(2):231–249, 1985.

[23] 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.

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

[25] Nicolas W Schuck, Ming Bo Cai, Robert C Wilson, and Yael Niv. Human orbitofrontal cortex represents a cognitive map of state space. *Neuron*, 91(6):1402–1412, 2016.

[26] Ramon Bartolo and Bruno B Averbeck. Inference as a fundamental process in behavior. *Current opinion in behavioral sciences*, 38:8–13, 2021.

[27] Matthijs AA van Der Meer and A David Redish. Ventral striatum: a critical look at models of learning and evaluation. *Current opinion in neurobiology*, 21(3):387–392, 2011.

[28] Owen Marschall and Cristina Savin. Probing learning through the lens of changes in circuit dynamics. *bioRxiv*, pages 2023–09, 2023.

[29] David Sussillo and Omri Barak. Opening the black box: low-dimensional dynamics in high-dimensional recurrent neural networks. *Neural computation*, 25(3):626–649, 2013.

[30] Kong-Fatt Wong and Xiao-Jing Wang. A recurrent network mechanism of time integration in perceptual decisions. *Journal of Neuroscience*, 26(4):1314–1328, 2006.

[31] Xiao-Jing Wang. Neural dynamics and circuit mechanisms of decision-making. *Current opinion in neurobiology*, 22(6):1039–1046, 2012.

[32] Arseny Finkelstein, Lorenzo Fontolan, Michael N Economo, Nuo Li, Sandro Romani, and Karel Svoboda. Attractor dynamics gate cortical information flow during decision-making. *Nature neuroscience*, 24(6):843–850, 2021.

[33] Yasushi Miyashita and Han Soo Chang. Neuronal correlate of pictorial short-term memory in the primate temporal cortexyasushi miyashita. *Nature*, 331(6151):68–70, 1988.

[34] 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.

[35] Jeffrey L Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99, 1993.

[36] D Kepple, Rainer Engelken, and Kanaka Rajan. Curriculum learning as a tool to uncover learning principles in the brain. In *International Conference on Learning Representations*, 2022.

[37] Daniel A Braun, Carsten Mehring, and Daniel M Wolpert. Structure learning in action. *Behavioural brain research*, 206(2):157–165, 2010.

[38] Hiroshi Makino. Arithmetic value representation for hierarchical behavior composition. *Nature Neuroscience*, 26(1):140–149, 2023.

[39] Laura Driscoll, Krishna Shenoy, and David Sussillo. Flexible multitask computation in recurrent networks utilizes shared dynamical motifs. *bioRxiv*, pages 2022–08, 2022.

[40] Matthew M Botvinick, Yael Niv, and Andew G Barto. Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective. *cognition*, 113(3):262–280, 2009.

[41] Mark S Goldman. Memory without feedback in a neural network. *Neuron*, 61(4):621–634, 2009.

[42] Tankut Can, Kamesh Krishnamurthy, and David J Schwab. Gating creates slow modes and controls phase-space complexity in grus and lstms. In *Mathematical and Scientific Machine Learning*, pages 476–511. PMLR, 2020.

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

5 Methods

5.1 Animal subjects and behavior

Behavioral procedures have been published in detail elsewhere ([18])). Briefly, a total of 291 Long-evans rats (184 male, 107 female) between the ages of 6 and 24 months were used for this study (Rattus norvegicus). The Long-evans cohort also included ADORA2A-Cre (N =10), ChAT-Cre (N =2), DRD1-Cre (N=3), and TH-Cre (N =12). Animal use procedures were approved by the New York University Animal Welfare Committee (UAWC #2021-1120)and carried out in accordance with National Institutes of Health standards. Animals were water restricted to motivate them to perform behavioral trials. From Monday to Friday, they obtained water during behavioral training sessions, which were typically 90 minutes per day, and a subsequent ad libitum period of 20 minutes. Following training on Friday until mid-day Sunday, they received ad libitum water. Rats were weighed daily. Rats were trained in a high-throughput behavioral facility in the Constantinople lab using a computerized training protocol. Rats were trained in operant boxes with three nose poke ports. Left and right ports contained speakers to generate audio tones, and contained lick tubes to delivery water reward. All ports contained an internal LED light inside the port.

LED illumination from the center port indicated that the animal could initiate a trial by poking its nose in that port - upon trial initiation the center LED turned off. While in the center port, rats needed to maintain center fixation for a duration drawn uniformly from [0.8, 1.2] seconds. During the fixation period, a tone played from both speakers, the frequency of which indicated the volume of the offered water reward for that trial [1, 2, 4, 8, 16kHz, indicating 5, 10, 20, 40, 80 μL rewards]. Following the fixation period, one of the two side LEDs was illuminated, indicating that the reward might be delivered at that port; the side was randomly chosen on each trial. This event (side LED ON) also initiated a variable and unpredictable delay period, which was randomly drawn from an exponential distribution with mean = 2.5 seconds. The reward port LED remained illuminated for the duration of the delay period, and rats were not required to maintain fixation during this period, although they tended to fixate in the reward port. When reward was available, the reward port LED turned off, and rats could collect the offered reward by nose poking in that port. The rat could also choose to terminate the trial (opt-out) at any time by nose poking in the opposite, un-illuminated side port, after which a new trial would immediately begin. On a proportion of trials (15-25%), the delay period would only end if the rat opted out (catch trials). If rats did not opt-out within 100s on catch trials, the trial would terminate.

The trials were self-paced: after receiving their reward or opting out, rats were free to initiate another trial immediately. However, if rats terminated center fixation prematurely, they were penalized with a white noise sound

and a time out penalty (typically 2 seconds, although adjusted to individual animals). Following premature fixation breaks, the rats received the same offered reward, in order to disincentivize premature terminations for small volume offers. We introduced semi-observable, hidden-states in the task by including uncued blocks of trials with varying reward statistics17: high and low blocks, which offered the highest three or lowest three rewards, respectively, and were interspersed with mixed blocks, which offered all volumes. There was a hierarchical structure to the blocks, such that high and low blocks alternated after mixed blocks (e.g., mixed-high-mixed-low, or mixed-low-mixed-high). The first block of each session was a mixed block. Blocks transitioned after 40 successfully completed trials. Because rats prematurely broke fixation on a subset of trials, in practice, block durations were variable.

5.1.1 Behavioral shaping

The shaping procedure was divided into 8 stages. For stage 1, rats learned to maintain a nose poke in the center port, after which a 20μL reward volume was delivered from a random illuminated side port with no delay. Initially, rats needed to maintain a 5 ms center poke. The center poke time was incremented by 1 ms following each successful trial until the center poke time reached 1s, after which the rat moved to stage 2.

Stages 2-5 progressively introduced the full set of reward volumes and corresponding auditory cues. Rats continued to receive deterministic rewards with no delay after maintaining a 1 second center poke. Each stage added one additional reward that could be selected on each trial- stage 2 added 40 μL, stage 3 added 5 μL, stage 4 added 80 μL, and stage 5 added 10 μL. Each stage progressed after 400 successfully completed trials. All subsequent stages used all 5 reward volumes.

Stage 6 introduced variable center poke times, uniformly drawn from [0.8-1.2] s. Additionally, stage 6 introduced deterministic reward delays. Initially, rewards were delivered after a 0.1s delay, which was incremented by 2 ms after each successful trial. After the rat reached delays between 0.5 and 0.8 s, the reward delay was incremented by 5 ms following successful trials. Delays between 0.8 and 1 s were incremented by 10 ms, and delays between 1 and 1.5 s were incremented by 25 ms. Rats progressed to stage 7 after reaching a reward delay of 1.5 s.

In stage 7, rats experienced variable delays, drawn from an exponential distribution with mean of 2.5 seconds. Additionally, we introduced catch trials (see above), with a catch probability of 15%. Stage 7 terminated after 250 successfully completed trials. Finally, stage 8 introduced the block structure. We additionally increased the catch probably for the first 1000 trials to 35%, to encourage the rats to learn that they could opt-out of the trial. After 1000 completed trials, the catch probability was reduced to 15-20%. All animal data in Figure 2 paper was from training stage 8.

5.2 Behavioral analyses

Behavioral sessions from rats required at least 5 catch trials to be included. Additionally, sessions were excluded if a linear regression of wait time did not have positive slope in a two-day moving-window average, or if it lacked a statistically significant positive slope coefficient for linear sensitivity on that day (F test, $p < 0.05$). Trials were also excluded trials when inter-trial-intervals (ITIs) exceeded the 99th percentile of ITIs. Lastly, trials were excluded if the wait time was greater than one standard deviation from the mean wait time. These criteria were included primarily to exclude trials in which a rat was disengaged during the task.

For analysis of wait time ratio and regression of wait time over training, data were aggregated over groups of 2 sessions either at the beginning of block training, or the end of training ("first sessions" and "last sessions," respectively, in Fig. 2 e). The wait time ratio was calculated on these aggregated sessions, and a regression of wait time with regressors for current trial, one trial back, and current block were used to regress the wait time on that trial. For this, wait times for opt out trials were were z-scored, and an ordinal block was used ($B_{\text{low}} = 1$, $B_{\text{mixed}} = 2$, $B_{\text{high}} = 3$).

For block transition dynamics, we followed the approach in [18]. The wait times on catch trials of each rat and RNN were first z-scored separately for each volume, then the difference in these z-scored wait times were calculated for each volume, relative to the average z-scored wait time for that volume. This was calculated for each trial relative to an incongruent trial following a block transition. The change in wait time in Figs. 2E and 3F was an average of volume of these values. This approach was used to control for reward volume effects.

5.2.1 Approximate Bayesian inference of block

The Bayesian observer in Fig. 2 h was calculated based on methods from [18]. Briefly the posterior belief of block was calculated according to Bayes rule:

$$P(B_t|R_t) = \frac{P(R_t|B_t)P(B_t)}{P(R_t)} \tag{2}$$

where B_t is the block on trial t and R_t is the reward on trial t. The likelihood $P(R_t|B_t)$ is the probability of the reward for each block (1/5 for all offers in mixed blocks, and 1/3 or 0 for low and high blocks). The prior $P(B_t)$ is

approximated using a posterior from the last trial as

$$P(B_t) \approx P(B_t|R_{t-1}) = \sum_{B_{t-1}} P(B_t|B_{t-1})P(B_{t-1}|R_{t-1}), \tag{3}$$

where $P(B_t|B_{t-1})$ is referred to as the hazard rate, which incorporates knowledge of the task structure, including the block length and block transition probabilities. For example, for blocks of length H, the hazard rate for low blocks would be

$$P(Low|B_{t-1}) = \begin{cases} 1-H, & B_{t-1} = \text{low} \\ H, & B_{t-1} = \text{mix} \\ 0, & B_{t-1} = \text{high} \end{cases} \tag{4}$$

H = 1/40, to reflect the block length in this work. Extensive explanation motivating the assumption of a flat hazard rate is provided [18].

5.3 Details of Markov decision process (MDP) and RL formulation of task

The decision process (Fig. 2 g) and RL task environment (Fig. 3 b) were modeled as a simplified form of the animal task, without any left/right side choice information. Formally this is modeled as an episodic MDP with two potential actions (Wait, OptOut). The state at each point in a trial is minimally defined as a reward offer R and time within a trial, t: $S(R,t)$. Rewards are delivered probabilistically on each trial with probability $p_r = 0.8$. Within a trial, the reward is delivered at a time that is drawn from an exponential delay distribution (mean wait time $\lambda = 2.5s$). At every time step before reward is delivered, the agent received a small reward penalty $k = -0.05$, and when the opt-out penalty action is chosen, a reward penalty $R_{\text{oo}} = -2.0$ is provided. Rather than instantaneously award the reward offer or opt-out penalty, the reward was instead evenly divided out across the time bins of the inter-trial interval (ITI). ITIs were drawn from a normal distribution with with mean of 1s, and variance 450 ms. This form of ITI better mimicked the experience in the rat task, in which animals were drinking water throughout the ITI. It also creates longer timescale dependencies across trials, and receiving positive reward (rather than no reward) during the ITI improves value function estimates across trials. To avoid contaminating learning of the policies for Wait and OptOut, we enforced a required WaitITI action that was given unit probability of occurring during the ITI, and zero during the waiting epoch. We do not model violations in the RL formulation of the task, as we do not have a prolonged stimulus epoch analogous to when rats hear the offer tone.

Reward offers followed an alternating block structure, as in the rat task, with random transitions. The same alternating block structure was used, with sessions always beginning with a mixed block, and then transitioning into a low block. Note, in the rat task, there is equal probability of high or low blocks being the first adaptation block, though this difference is likely negligible to RNN behavior. Blocks were a minimum of 40 trials long, and after 40 trials block transitions occurred probabilistically, drawn from a binomial distribution with transition probability $p = 0.5$

5.4 Model architecture

The RL agent is comprised a two-layer network, each with 256 LSTM units (Fig. 3 a), with relevant gates, states and inputs denoted by (j) for layer j. The computational LSTM unit of the network uses the following gates to update the hidden states h_t and cell states c_t:

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \tag{5}$$
$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \tag{6}$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \tag{7}$$
$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \tag{8}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{9}$$
$$h_t = o_t \odot \tanh(c_t). \tag{10}$$

$\odot$ is the elementwise Hadamard product, σ is a sigmoid nonlinearity. $S_t^{(j)} \equiv [h_t^{(j)}, c_t^{(j)}]$ is a compact description of the activity of each unit, and successive states of the network are written here with a shorthand $S_{t+1} = \text{RNN}(S_t)$ or $S_{t+1} = \text{RNN}(\cdot)[(S_t)]$. Network inputs x_t are multidimensional, and will generally consist of external task stimuli s_{start} and s_{R}, past timestep reward r_{t-1}, past timestep action a_{t-1}, outputs from the network o_t, and states S_t from earlier LSTM layers. $s_{\text{start}} = 1$ and $s_{\text{R}} = \log(R)$ only at the start of the trial, and are otherwise zero. In contrast, the remaining inputs can in general by non-zero and time-varying over the course of an entire trial.

The first layer, a putative OFC layer, receives inputs about the task stimulus, trial start, reward offer, previous reward, and previous action: $x^{(\mathrm{OFC})} = [s_{\mathrm{start},t}, s_{\mathrm{R},t}, r_{t-1}, a_{t-1}]$. The OFC layer projects onto a 3-unit linear projection head that outputs the log-probability of each block $o_t^{(\mathrm{block})} = \log[p_{\mathrm{block},t}]$, as well as a linear projection head for an auxiliary task that outputs the average reward within a trial $o_t^{(\mathrm{int})}$. The second layer, a putative striatum layer (STR), receives OFC's block probabilities as inputs, as well as the stimulus inputs for trial start and reward offer: $x^{(\mathrm{STR})} = \left[s_{\mathrm{start},t}, s_{\mathrm{R},t}, o_t^{(\mathrm{block})}\right]$. The STR layer outputs onto linear projection heads for the i) policy π_t, ii) value of current state V_t, iii) prediction of reward offer for a supervised memory task $o_t^{(\mathrm{mem})}$, and iv) prediction of time within a task for a supervised memory task $o_t^{(\mathrm{count})}$. The policy head is a three-unit RNN with a softmax nonlinearity, corresponding to probabilities for waiting within a trial (waitTrial), opting out of a trial (OptOut), or waiting during the ITI (waitITI). During the trial, the only allowed options are waitTrial and Optout, with waitITI artificially set to low probability. During the the ITI period in which reward is delivered, the only option is waitITI. The remaining projection heads output scalar values.

Hidden and cell states of the LSTMs were initialized with random values drawn from a normal distribution ($\mathcal{N}(0,1)$). The weight parameters of the model were drawn from a uniform distribution ($\mathcal{U}(-1/N, 1/N)$, where N is the number of hidden layer units). States were reset at the end of each training epoch overall all data for kindergarten costs, and were reset every 160 trials during the temporal wagering task. This reset corresponds to roughly the same timescale as progressing through three blocks, though it is not exactly aligned with block transitions.

5.5 Training

All RNN model training was conducted in PyTorch (v 1.8.0). For all costs, weights were updated using backprop through time with an Adam minimizer. Hyperparameters for training are provided in the subsequent section.

5.5.1 Kindergarten training

Kindergarten tasks were comprised of supervised learning tasks: memory, counting, integration, and inference tasks. The memory, counting, and integration tasks were trained using supervised learning of a mean-squared error loss, where RNN ouputs o_t were trained to match a target output $o_{t,\mathrm{targ}}$. The memory task trained the network to output the initial reward offer throughout the duration of a trial ($o_{t,\mathrm{targ}} = s_{R,0}$). The counting task trained networks to count time elapsed within a trial ($o_{t,\mathrm{targ}} = t$). The integration task trained networks to calculate a running average of a the 'previous reward' stimulus input ($o_{t,\mathrm{targ}} = 1/t \sum_0^t r_{t-1}$). The inference task trained networks on a classification task using cross-entropy loss to categorize the latent block based on reward offer.

These tasks used similar input and trial time statistics as the target wait time task. Trial durations were on the same order of magnitude as the target task, and the inputs to the RNN used the same five target values as the log-reward offer stimulus input in the RL task. The target in the integration task was based off of typical rewards received from trials in the wait-time task. Importantly, while the input statistics were similar to the target task, these kindergarten tasks are supervised learning objectives in which the actions of the RL agent have no bearing. They are fundamentally a different class of learning than the final RL-based task.

We devised a cumulative curriculum for pretraining on the four tasks by introducing tasks into training, one at a time and adding onto previous tasks (Fig. S1). We first introduced simple, single-trial variants in order of memory, counting, and integration tasks (simple kindergarten). We then increased the complexity of these tasks by extending to multiple trials in a training batch, but continuing to train on the three tasks simultaneously (hard kindergarten). Finally, we added in the inference task (inference kindergarten).

The losses for each stage of kindergarten are the following:

$$\mathcal{L}_{\mathrm{simple}} = \sum_t^T \beta_{\mathrm{mem}} \|o_t^{(\mathrm{mem})} - o_{t,\mathrm{targ}}^{(\mathrm{mem})}\|^2 + \beta_{\mathrm{count}} \|o_t^{(\mathrm{count})} - o_{t,\mathrm{targ}}^{(\mathrm{count})}\|^2 + \beta_{\mathrm{int}} \|o_t^{(\mathrm{int})} - o_{t,\mathrm{targ}}^{(\mathrm{int})}\|^2, \tag{11}$$

$$\mathcal{L}_{\mathrm{hard}} = \sum_{m=1}^M \mathcal{L}_{\mathrm{simple}}^{(m)} \tag{12}$$

where $o_t^{(\cdot)}$ are output signals from the RNN, with $o_t^{(int)}$ coming from the OFC layer, and $o_t^{(mem)}$ and $o_t^{(count)}$ coming from the STR layer. Training data for simple kindergarten was performed in a single batch, with 20 timesteps of data per sample, ($T = 1s$, $\Delta t = 0.05$), and a batchsize of 1000 samples. Weight update occurred after every epoch over the data, and stopped when performance did not improve over 30 epochs. The second stage of kindergarten training (hard) simply expanded the time horizon of simple kindergarten, which optimized with target values from a single trial, m, to include multiple trials. We used 10 trials per sample, with variable trial times drawn from a uniform distribution between 1 and 5s. Training data was again a single batch, with batchsize = 1000. To maintain a single batch of data with the same amount of data per sample, each sample used 10 trials of the same durations,

but in random order for each sample. Weight update was performed after each epoch over the data, and completed after a threshold of $\mathcal{L}_{\text{hard}} < 0.001$ was reached, or until 10k epochs were performed. Rather than processing each contribution to the loss cumulatively as in simple kindergarten, all 3 costs were simultaneously optimized in hard kindergarten.

Following training on simple and hard kindergarten, which were tasks with squared error losses, the final stage of kindergarten (inference) was performed. Formally, this required the outputs of the OFC head, p_{block} to minimize a cross-entropy categorization loss at every time point,

$$\mathcal{L}_{\text{inf}} = -\sum_{m,t} I^{(m)} \log[p^{(m)}_{\text{block},t}], \tag{13}$$

where $I^{(m)}$ is an indicator function for the true block on each trial, m, taking the value $I^{(m)} = 1$ for the true block type, and 0 for the remaining block types. This final stage of kindergarten cumulatively optimized inference and the earlier kindergarten tasks:

$$\mathcal{L}_{\text{kind}} = \beta_{\text{hard}}\mathcal{L}_{\text{hard}} + \beta_{\text{inf}}\mathcal{L}_{\text{inf}}, \tag{14}$$

where $\beta_{\text{hard}} = 1.0$ and $\beta_{\text{inf}} = 0.5$ during kindergarten pre-training.

The sham delay-to-nonmatch task used an augmented RNN with two additional inputs to the OFC layer, as well as an additional output from the STR layer. The two inputs were chosen from the the potential offers provided during the task ($[5, 10, 20, 40, 80]$), and there was a temporal delay between the two offers drawn from a uniform distribution $\mathcal{U}(0.05s, 5s)$. There was equal probability of both inputs being the same or different on each trial. As with the other supervised learning tasks, trials were batched. The goal of the the agent was to output 'match' (1) if the two inputs matched, 'non-match' (2) if they were different , and 'no cue' (0) otherwise. The task was optimized using a cross-entropy loss as in 13.

5.5.2 Deep meta-RL loss

Following previous work [23], the temporal wagering tasks described above was optimized with a deep meta-learning framework that optimized an actor-critic loss that was regularized with an entropy loss to encourage explorations, as well as the full kindergarten loss,

$$\mathcal{L} = \beta_{\text{actor}} \sum_t -\log \pi(a_t|S_t)\,[G_t - V(S_t)] + \beta_{\text{critic}}[G_t - V(S_t)]^2 - \beta_H \pi_t log(\pi_t) + \mathcal{L}_{\text{kind}}, \tag{15}$$

where G_t is the empirical, total discounted future reward

$$G_t = r_t + \gamma G_{t+1} \tag{16}$$

5.6 Hyperparameters

Hyperparameters for all tasks were chosen with a combination of grid searches over groups of parameters that optimized performance and ideal behavior on the task (to balance good linear sensitivity, modest opt-out rates, low wait-time intercepts, and good inference performance), empirical searching to optimize behavior, and previous approaches as in [23]. We also utilized a heuristic approach for choosing some regularizer weights that is common in the literature, in which they are chosen in order to balance contributions from each term in eq. (15).

target task

parameter	*value*	*description*	*tuning method*
α	$1e^{-4}$	learning rate	based on [23]
γ	0.95	discount factor	grid search
β_{actor}	1	actor weight	-
β_{critic}	0.0005	critic weight	grid search
β_{H}	0.05	critic weight	heuristic
β_{kind}	10	hard kindergarten weight	empirical, heuristic
β_{inf}	0.5	inference kindergarten weight	grid search
β_{d2m}	0.5	delay-to-match sham weight	empirical
p_r	0.8	trial reward probability	based on [18]
Nt	500	# timesteps per gradient step	empirical
T	30s	max. time before trial timeout violation	empirical
Δt	0.05 s	length of time between RNN steps	empirical
T_{iti}	$\mathcal{N}(1, 0.15)s$	wait time distribution between trials	empirical
block length	40 trials + coin toss (p=0.5)	trials per block	based on [18]
state reset	160	# trials before reset of network	empirical

simple and hard kindergarten

parameter	*value*	*description*	*tuning method*
α_{kind}	$1e^{-4}$	learning rate	empirical
batchsize	1000	-	empirical
$\#epochs$	$1e^{4}$	-	empirical
Nf_{hard}	30	convergence horizon (in # trials), simple kind.	empirical
ϵ_{hard}	0.001	threshold cutoff, hard kind.	empirical
Nt_{simple}	20	# timesteps per trial, simple kind.	empirical
Nt_{hard}	$\mathcal{U}(20, 100)$	# timesteps per trial, hard kind.	empirical
Ns_{hard}	10	# trials per sample, hard kind.	empirical
batchsize	1000	-	empirical

inference kindergarten

parameter	*value*	*description*	*tuning method*
α_{inf}	0.005	learning rate	empirical
β_{inf}	1	loss weight, inference cross-entropy loss	empirical and heuristic
β_{hard}	0.5	loss weight, hard kind. supervised L2 loss	empirical and heuristic
Ns_{inf}	400	# trials per sample, inference	empirical
batchsize	20	-	empirical
# epochs	$1e^{4}$	-	empirical
ϵ_{inf}	0.2	threshold cost, inference	empirical

Delay-to-match kindergarten task

parameter	*value*	*description*	*tuning method*
α_{d2m}	$1e^{-3}$	learning rate	empirical
batchsize	1000	-	empirical
β_{d2m}	1	delay-to-match sham weight	-
β_{kind}	0.5	kindergarten weight	empirical

5.7 Network analyses

To determine which phases of training created large structural changes in the network, we calculated the change in weights ΔW over training. Specifically, we investigated the squared L2 norm of the difference of a concatenated recurrent weight matrix $W \equiv [W_{fh}, W_{gh}, W_{oh}, W_{ih}]$, over training. Different phases of training had different learning rates, so to meaningfully compare the magnitude of changes over training we sampled the network at different rates across training, proportional to their learning rates. Intuitively, this means we sampled training phases less frequently when only small parameter updates were possible (smaller learning rate), and conversely sampled more frequently in training epochs where large parameter updates were possible (larger learning rate). Specifically, the change $\|\Delta W\|^2$ in each stage is given by

$$\|\Delta W_s\|_L^2 = \|W_{s+N_L} - W_s\|_L^2, \tag{17}$$

$$N_L = \frac{0.1}{\alpha^{(e)}} \tag{18}$$

N_L is the number of gradient steps between samples, and $\alpha^{(e)}$ is the learning rate for a training epoch of the curriculum. For simple kindergarten, hard kindergarten, and the wait time task $\alpha^{(e)} = 1e^{-4}$ (N=1000 steps). For the inference stage of kindergarten, $\alpha^{(e)} = 0.005$ (N=2 steps).The wait-time task has variable numbers of gradient steps across trials, but empirically we observed that 1000 gradient steps spanned approximately 10k trials. Thus, we sampled the network every 10k trials for the wait time task.

5.8 Reduced dynamics flow fields

A low-dimensional manifold of activity was found through conventional means by performing PCA on $S^{(i)}$ from a session of 1000 trials from the wait time task. PCA was performed separately for each layer of the network to give a reduced set of of activity $P_t^{(i)} = \tilde{M}^{-1(i)}\left[S_t^{(i)} - \hat{S^{(i)}}\right]$, where $M^{(i)}$ are the principal components and $\hat{S^{(i)}}$ is the mean activity of network in layer i, respectively. $\tilde{M}^{-1(i)}$ denotes the pseudo -inverse of $M^{(i)}$. Networks fully trained with kindergarten curriculum learning had $\approx 90\%$ variance explained in the first two to three principal components. Thus, we visualized the dynamics in a two-dimensional space; however analysis of the low-dimensional dynamics was always performed in the N-D space that captured at least 90% variance.

Dynamical flow fields reflect the instantaneous change of the network state over time, due to network activity. We approximate this temporal gradient with an empirical difference F_t across successive states as

$$\dot{S}_t \equiv \frac{\partial S_t}{\partial t} \sim \frac{S_{t+1} - S_t}{\Delta t} \equiv F_t \tag{19}$$

$$F_t = \frac{(RNN(\cdot) - \mathbb{I})\,[S_t]}{\Delta t} \tag{20}$$

The flow field for behaviorally relevant low-dimensional dynamics in PC space is equivalently calculated by first projecting back into neural activity space with an inverse PCA transform, calculating F_t, then projecting this gradient back into PC space. Assuming $\hat{S^{(i)}} = 0$ for simplicity, this dynamical gradient $F_{PC,t}$ is given by

$$\dot{P}_t \equiv \frac{\partial P_t}{\partial t} \sim \frac{P_{t+1} - P_t}{\Delta t} \equiv F_{PC,t} \tag{21}$$

$$F_{PC,t} = \frac{\tilde{M}^{-1}\,(RNN(\cdot) - \mathbb{I})\,[MP_t]}{\Delta t}. \tag{22}$$

The dynamics have an intrinsic dimensionality that is typically greater than the space spanned by the variability of the network, as we noted some non-zero changes in activity in higher dimensions beyond our PCA projection (not shown). However, in practice, single-trial trajectories in these two dimensions tended to follow the flow fields of the 2-D dynamics quite well, and served as a sufficient space to analyze aspects of the dynamics that directly contribute to behavior.

Dynamics are only well-defined for static inputs, which for the wait-time epoch used in Figure 5 are $x^{(\mathrm{OFC})} = [s_{\mathrm{start},t} = 0, s_{\mathrm{R},t} = 0, r_{t-1} = -k, a_{t-1}] = 0$, and $x^{(\mathrm{STR})} = [s_{\mathrm{start},t} = 0, s_{\mathrm{R},t} = 0, o_{\mathrm{block}}]$. To calculate the reduced dynamics for stratum with a static input, we chose o_{block} values the corresponded to the block probability estimates output from the RNN during characteristic opt-out trials from each block, just before the RNN opted out.

5.9 Linearized dynamics

The dynamics of an RNN are in general nonlinear, and can be approximated by a Taylor series expansion around fixed point S_0 as

$$\dot{S}_t \approx \nabla[\dot{S}_t]^T\Big|_{S_0}(S_t - S_0) + \frac{1}{2}(S_t - S_0)^T\,\nabla^2[\dot{S}_t]\Big|_{S_0}(S_t - S_0) + \text{h.o.t.} \tag{23}$$

Analysis of the linearized dynamics matrix $J \equiv \nabla[\dot{S}_t]^T_{i,j} = \partial\dot{S}_i/\partial S_j$ (*i.e.,* Jacobian of the temporal gradient) provides a compact description of the underlying dynamical system driving behavior. The Jacobian for an LSTM only requires derivatives with respect to the cell state and hidden state. We again approximate the temporal gradient with an empirical difference, and calculate the Jacobian J as

$$\nabla[\dot{S}_t]^T \approx \nabla[F_t]^T \equiv J = \begin{pmatrix} \frac{\partial F_t}{\partial h_t}, & \frac{\partial F_t}{\partial c_t} \end{pmatrix} \tag{24}$$

$$= \frac{1}{\Delta t}\begin{pmatrix} \frac{\partial h_{t+1}}{\partial h_t} - \mathbb{I} & \frac{\partial h_{t+1}}{\partial c_t} \\ \frac{\partial c_{t+1}}{\partial h_t} & \frac{\partial c_{t+1}}{\partial c_t} - \mathbb{I} \end{pmatrix} \tag{25}$$

where we have used the fact that $S_{t+1} = \mathrm{RNN}(\cdot)[(S_t)]$. The Jacobian can be written more compactly as

$$J = \frac{1}{\Delta t}\begin{pmatrix} h_h^T - \mathbb{I} & h_c^T \\ c_h^T & c_c^T - \mathbb{I} \end{pmatrix} \tag{26}$$

We found that the Jacobian was typically non-normal (*i.e.,* when $JJ^T \neq J^TJ$), so we characterized the linearized dynamics by their spectral properties by using the Schur decomposition $J = WT$ [41]. The Schur decomposition is analogous to a standard eigenvalue decomposition, but returns orthogonal modes, even for non-normal matrices. We characterized fixed points of the dynamics by the eigenvalues of J (diag(T)), and the Schur modes W.

5.10 Locating and classifying dynamical fixed points

Analysis of the dynamics for each network numerically located fixed points (F_t or $F_t^{(PC)} = 0$); as well as "slow points" where the dynamics contained local minima, but $F_t^{(PC)} > 0$. These points were located through a minimization of the kinetic energy of the system, where kinetic energy is defined by

$$\mathrm{KE} = \frac{1}{2} F_{PC,t}^2. \tag{27}$$

To locate fixed points in the behaviorally relevant subspace of RNN activity, a constrained minimization of kinetic energy based on $F^{(PC)}$ was performed. Specifically, we found the low-dimensional manifold via PCA, and defined the dimensionality N as the number of principal components that captured $> 90\%$ variance This minimization searched for kinetic energy minima on the N-D PCA manifold. While a fully unconstrained minimization would identify all of the fixed points of the RNN dynamics, constraining to PC space restricts our analysis to the network dynamics driving behavior. Specifically, 1000 trials of the wait time task were simulated, then PCA was performed separately on the activity of STR and OFC defining the PC space for each layer of the network. To choose initial conditions to the minimization, we used a uniform sampling for 50 points per dimension when $N = 1$ or $N = 2$. For higher dimensionality, to avoid exponential scaling issued we used a biased grid search proportional to total variance explained by each dimension. The same network inputs that were used in calculating flow fields were also used for this analysis

Once minima were located, a clustering procedure (DBSCAN, scipy.clustering.DBSCAN) was performed to determine the effective number of fixed and slow points, as well as remove any outliers. The hyperparameters were min_samples = 10, and epsilon. Epsilon was chosen individually for each network, as 1% of the range of support of PC1 activity. Fixed points were defined as identified minima where $KE < 1e^{-4}$. All other minima were termed "slow points." The network dynamics were then linearized at the identified fixed and slow points, then a Schur decomposition was performed on the Jacobian to retrieve the eigenvalues and Schur modes of the system. Dynamical systems features were categorized in the top two dimensions of PC space. based on their eigenvalues λ_1 and λ_2 found by the Schur decompositions, we used the following scheme:

- $\lambda_1 < 0.999$, $\lambda_2 < 0.999$: point attractor
- $\lambda_1 \in [0.999, 1.001]$ and $\lambda_2 < 1.001$: stable line/plane attractor
- $\lambda_1 \in [0.999, 1.001]$ and $\lambda_2 > 1.001$: unstable line
- $\lambda_1 < 0.999$ and $\lambda_2 > 1.001$: saddle point

In order to characterize the dynamics more concretely, we identified a 'common motif' in the dynamics of OFC if the following features were observed: the dynamics contain a stabilizing feature (point or line attractor) in high and low-block regions of state space, as well as an unstable feature (typically a saddle) in the mixed-block region of state space. Additionally, 2-D PC space needed to contain three discrete and continuous regions of block confidence, assessed using o_{block}. Networks were considered to have 'rat-like' strategies if they possessed linear sensitivity to reward offers, as well sensitivity to block context for all blocks.

6 Supplementary Figures

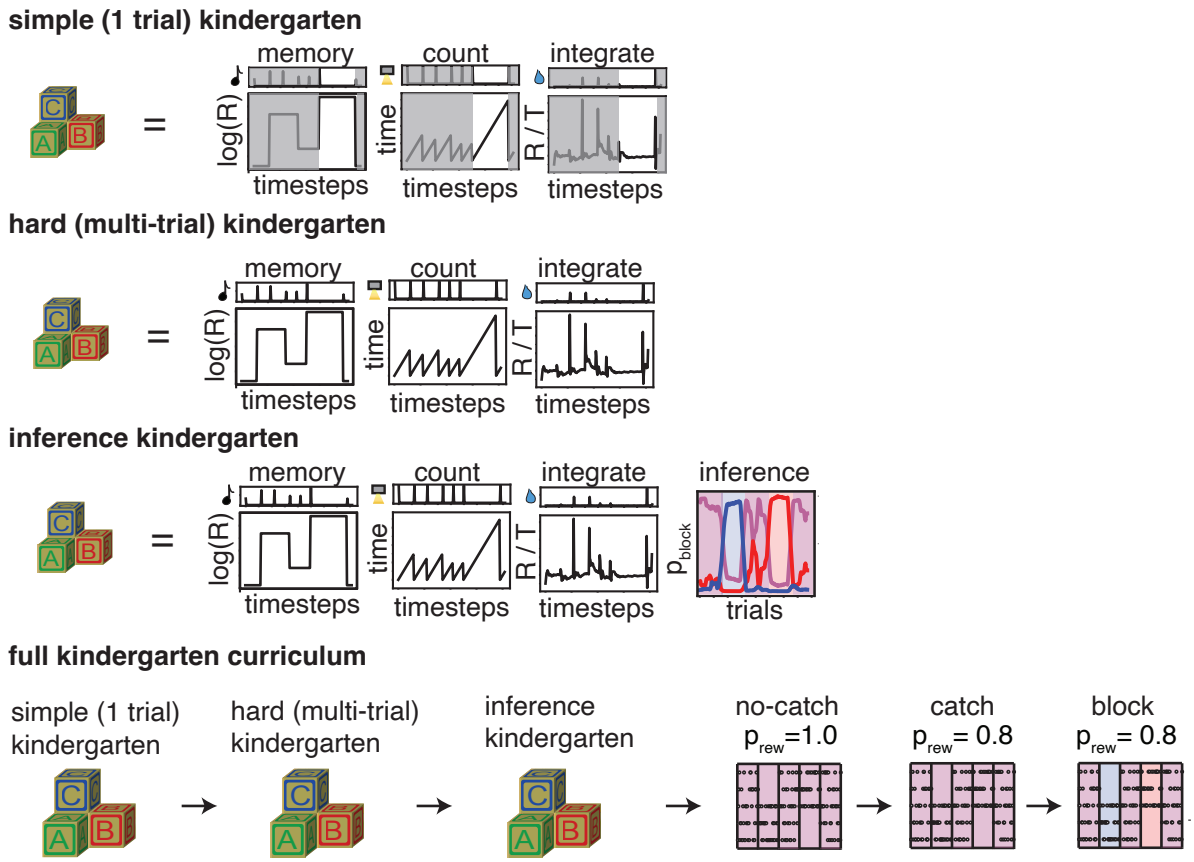

Figure S1: Full description of kindergarten curriculum learning. The kindergarten portion of training is also a curriculum of tasks, beginning with the memory, counting, and stimulus integration tasks over a single trial, denoted by non-masked regions (simple, 1-trial kindergarten). Following this, multiple trials are incorporated into training (Hard kindergarten). Once these tasks are performed well, the inference task is introduced (inference kindergarten). The complete kindergarten CL procedure performs these supervised tasks first, then begins training on the wait-time task using a classical CL approach of simpler task variants.

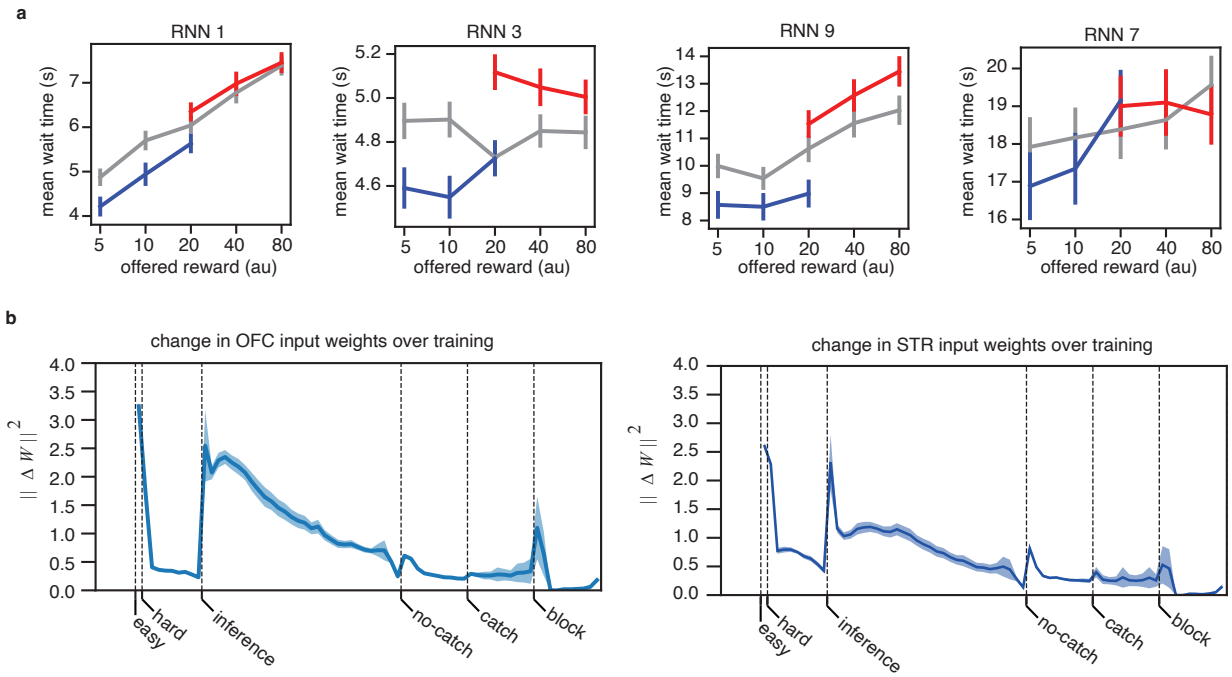

Figure S2: **a)** Representative wait time curves on catch trials for sample RNNs trained with classic CL. **b)** Change in network input weights across learning. Same timesteps over training used as in Fig. 4e. Error bars denote sem

7 Supplementary Materials

7.1 LSTM derivatives for linearized dynamics

Here we focus on derivatives of cell and hidden state in eqs. (24-26), although full treatment of the linearization of LSTM can be found in [42]. First, an important computational identity for handling the derivative of nonlinear functions of matrix-vector products $f(Wx)$, for $W \in \mathbb{R}^{nxm}, x \in \mathbb{R}^{mx1}$, is

$$\frac{\partial f(Wx)}{\partial x} = W^T \odot (\mathbf{1}_m \otimes f'^T), \tag{1}$$

where $f' = df(x)/dx$, and $\mathbf{1}_m \in \mathbb{R}^{mx1}$. $\otimes$ is the Kronecker outer product. Additionally, for $\partial(x \odot y)/\partial h$ we have via the chain rule,

$$\frac{\partial(x \odot y)}{\partial h} = x_h \odot (\mathbf{1}_m \otimes y^T) + (\mathbf{1}_m \otimes x^T) \odot y_h, \tag{2}$$

where $x_h \equiv \partial x/\partial h$, $y_h \equiv \partial y/\partial h$

Again, the LSTM gates are the following

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \tag{3}$$
$$f_t = \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \tag{4}$$
$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \tag{5}$$
$$g_t = \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \tag{6}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{7}$$
$$h_t = o_t \odot \tanh(c_t). \tag{8}$$

$\odot$ is the elementwise Hadamard product, σ is a sigmoid nonlinearity.

By denoting partial derivatives of gate a_t as $\partial a_{t+1}/\partial h_t \equiv a_h$, $\partial a_{t+1}/\partial c_t \equiv a_c$, the relevant gate derivatives for

the Jacobian are the following:

$$a_h = W_{ha}^T \odot \left[\mathbf{1}_m \otimes (a \odot (1-a))^T\right], \quad a \in [i, f, o] \tag{9}$$

$$g_h = W_{hg}^T \odot \left[\mathbf{1}_m \otimes (1-g^2)^T\right] \tag{10}$$

$$c_h = f_h \odot (\mathbf{1}_m \otimes c_t^T) + i_h \odot (\mathbf{1}_m \otimes g^T) + (\mathbf{1}_m \otimes i^T) \odot g_h \tag{11}$$

$$\frac{\partial h_t}{\partial c_t} = o_t \odot (1 - \tanh^2(c_t)) \tag{12}$$

$$h_h = o_h \odot (\mathbf{1}_m \otimes \tanh(c^T)) + (\mathbf{1}_m \otimes [o \odot (1 - \tanh^2(c)]^T) \odot c_h \tag{13}$$

$$a_c = W_{ha}^T \odot \left(\mathbf{1}_m \otimes \left[\frac{\partial h_t}{\partial c_t} \odot a \odot (1-a)\right]^T\right), \quad a \in [i, f, o] \tag{14}$$

$$g_c = W_{hg}^T \odot \left(\mathbf{1}_m \otimes \left[\frac{\partial h_t}{\partial c_t} \odot (1-g^2)\right]^T\right) \tag{15}$$

$$c_c = f_c \odot (\mathbf{1}_m \otimes c_{t-1}^T) + (\mathbf{1}_m \otimes f^T) + i_c \odot (\mathbf{1}_m \otimes g^T) + (\mathbf{1}_m \otimes i^T) \odot g_c \tag{16}$$

$$h_c = o_c \odot (\mathbf{1}_m \otimes \tanh(c^T)) + (\mathbf{1}_m \otimes (o \odot [1 - \tanh^2(c)]^T) \odot c_c \tag{17}$$

where we used eqs. (1, 2), $d\sigma(x)/dx = \sigma(x)(1-\sigma(x))$, and $d\tanh(x)/dx = 1 - \tanh^2(x)$.

7.2 Additional MDP analysis

In this section, we provide additional details about the Markov decision process for the temporal wagering task.

7.2.1 Derivation of optimal wait times

Specifically we consider the state-action values of waiting, and opting out. For an offer of size R, Based on the Bellman equation, these are given by [43]

$$\begin{aligned} Q(\text{wait}, S_t) &= r_{t+\Delta t} + \gamma \max_{a_{t+\Delta t}} Q(a_{t+\Delta t}, S_{t+\Delta t}) \\ &= p(t)R - k + \gamma \max_{a_{t+\Delta t}} Q(a_{t+\Delta t}, S_{t+\Delta t}) \end{aligned} \tag{1}$$

$$Q(\text{opt-out}) = -R_{\text{oo}}. \tag{2}$$

The point at which you should no longer wait is when the value of the choices are equivalent,

$$Q(\text{wait}, S_t) = Q(\text{opt-out}, S_t). \tag{3}$$

In the Bellman in eq. (1), at one step prior to this opting out timepoint, the next maximizing action should be to opt out, implying

$$\max_{a_{t+\Delta t}} Q(a_{t+\Delta t}, S_{t+\Delta t}) = Q(\text{opt-out}, S_{t+\Delta t}), \tag{4}$$

and for $\gamma = 1$ this yields an expression for determining optimal wait times (as in [20]):

$$p_r R_n p(t) - k + Q(\text{opt-out}, S_t) = Q(\text{opt-out}, S_t) \tag{5}$$

$$p_r R_n p(t) = k \tag{6}$$

For an exponential distribution $p(t) = \lambda^{-1}\exp(-t/\lambda)$, the optimal wait time t^* is log-linear in reward offer as

$$\frac{p_r R_n}{\lambda} \exp\left(-\frac{t^*}{\lambda}\right) = k \tag{7}$$

$$t^* = \lambda\left(\log[R_n p_r] - \log[k\lambda]\right) \tag{8}$$

Eq. (7) is the core result that describes optimal behavior: rewards should be linear in log-offer, with an intercept that is determined by the wait time penalty. In foraging theory, wait time penalties are often interpreted to be the opportunity cost, or the average reward rate of the environment, meaning that wait time decisions should be incorporate the value of what is being missed out on by continuing to wait [21, 22]). In the MDP formulation here,

If k is taken to be related to the opportunity cost as $k = \rho\Delta t$, then shifts in the intercept of optimal wait times will be related to this log of opportunity cost:

$$p_r \tilde{R} p(t) = \rho \Delta t \tag{9}$$

$$E[\tilde{R}] = \rho \Delta t. \tag{10}$$

$$t^* = \lambda \left(\log[R_n p_r] - \log[E[\tilde{R}]\lambda] \right). \tag{11}$$

Here, $\tilde{R}$ is the expected average reward from a given environment, which in the context of the wait time task are taken to be low, mixed, and high blocks. This demonstrates that optimal behavior for a given offer R from an environment with higher average rewards will be to wait *less* time than if it came from an environment with lower average rewards. Intuitively, this can be thought of as being willing to wait longer for an option from a reward-sparse an environment, and less time when it happens to be the lowest option from a reward-rich environment.

7.2.2 Asymmetry in wait times across blocks

As a consequence of this average reward dependence in eq. (9), there is a predicted asymmetry in the wait times across blocks in our task, where low blocks should have larger offsets from mixed blocks than compared to high blocks. Consider the difference in wait times for a given offer R, taken from two different environments with average rewards R_0 vs. R_1

$$t_1^* - t_0^* = \lambda \log \left[\frac{\tilde{R}_0}{\tilde{R}_1} \right] \tag{12}$$

The ratio of these differences between low(L) vs. mixed(M), and high(H) vs. mixed blocks captures this asymmetry,

$$\frac{t_{\mathrm{L}}^* - t_{\mathrm{M}}^*}{t_{\mathrm{M}}^* - t_{\mathrm{H}}^*} = \frac{\log[\tilde{R}_{\mathrm{M}}] - \log[\tilde{R}_L]}{\log[\tilde{R}_{\mathrm{H}}] - \log[\tilde{R}_M]} \tag{13}$$

In our task, this gives rise to low blocks being approximately 2.4 times higher above the mixed block wait times, compared high blocks. Simulated results from eq. (1) are shown in Figure 2 h.

7.2.3 Q values for deterministic policy

Let us consider the action-state value $Q(\text{wait}, R_n, t) \equiv Q(\text{wait}, S(R_n, t))$, and use a dynamic programming approach, beginning from the opt-out time of a policy, t_f. Here we assume that $\gamma = 1$. This opt-out point occurs when $Q(\text{wait}, R_n, t_f) = Q(R_n, \text{opt-out}) = -R_{oo}$. Working backwards from this using the Bellman equation, you can determine one timestep back:

$$Q(\text{wait}, t = t_f - \Delta t) = r_{t_f} + Q(\text{opt-out}, t_f) \tag{14}$$

$$= p_r p(t_f) R_n - k - R_{oo} \tag{15}$$

A second timestep back yields

$$Q(\text{wait}, t_f - 2\Delta t) = r_{t_f - \Delta t} + Q(\text{wait}, t_f - \Delta t) \tag{16}$$

$$= p_r (p(t_f) + p(t_f - \Delta t)) R_n - 2k - R_{oo} \tag{17}$$

You can see that in the limit of $\Delta t \to 0$, this is becoming an integration. The integration back to an arbitrary time point is then

$$Q(\text{wait}, t) = \int_t^{t_f} [p_r p(t') R_n - k] dt' - R_{oo} \tag{18}$$

For the exponential decay distribution this yields

$$Q(\text{wait}, S(R_n, t)) = p_r R_n \left(\exp(-t/\lambda) - \exp(-t_f/\lambda) \right) - (t_f - t)k - R_{oo} \tag{19}$$

If the policy in question is the optimal wait time policy, then $t_f = t^* = \lambda \log \frac{R_n p_r}{k\lambda}$, which simplifies the Q value to

$$Q^*(\text{wait}, S(R_n, t)) = p_r R_n e^{-t/\lambda} - k\lambda \left(1 + \frac{1}{k} \log \left(\frac{p_r R_n}{k\lambda} \right) - \frac{t}{\lambda} \right) - R_{oo} \tag{20}$$

This contains an expected exponential decay term, weighted by the reward probability, as well as a linear term and a bias. The linear term comes from the linear penalty in the MDP.

7.2.4 reward rate for deterministic policy

Reward rates for a given policy are analytically less tractable due to unbounded exponential integrals, but numerical solutions can be found when there are minimal delivery times for reward, or for calculating differences between reward rates of different policies. The reward rate here is calculated by the expected total reward in a trial, divided by duration of a trial. Here we consider a deterministic policy that waits until t for reward, then opts out. On catch trials, it will simply wait until t and then opt out.

$$\rho^{\pi_t}(R_n) = p_r R_n \int_0^t \frac{p(t')}{t'} dt' - p_r R_{oo} \int_t^\infty \frac{p(t')}{t'} dt' - \frac{(1-p_r)R_{oo}}{t} - p_r k \int_0^t p(t') dt' - (1-p_r)k \tag{21}$$

The terms above respectively accounts for i) reward arriving before time t and receiving reward, ii) reward arriving after t, and instead opting-out, iii) a catch trial, iv) the time penalty for a rewarded trial before t, and v) time penalty on a catch trial. Note, there is not a time penalty for rewards that would arrive after t, since the agent ops out, hence the presence of the integral on the wait time term for non-catch trials. The first two terms are exponential integral of the form $I(t) = \int_0^\infty e^{-x}/x dx$, and do not have a numerical solution due to integration from $t = 0$. However, numerical solutions exist for i) integrals with a nonzero lower limit, as well as ii) differences between two integrals $I(t_1) - I(t_2)$. The first case helps us understand the MVT, and the second is useful for comparing the performance different policies. We now tackle each case in turn.

If the deterministic policy has a minimal wait time t_0, then the lower limit of the integrals avoids numerical instability issues, and the reward rate can be calculated numerically.

$$\rho^{\pi_t}(R_n) = p_r R_n \int_{t_0}^t \frac{p(t')}{t'} dt' - p_r R_{oo} \int_t^\infty \frac{p(t')}{t'} dt' - \frac{(1-p_r)R_{oo}}{t} - p_r k \int_{t_0}^t p(t') dt' - (1-p_r)k \tag{22}$$

Interpreting t_0 is difficult. It is not the minimal wait time of the agent, since this is a deterministic policy. It is more like the minimal time before the award can arrive. In the context of the behavioral task, this could be related to a fundamental limitation in reward delivery from the operant chamber, or alternatively the minimum reaction time of the animal. Additionally, we can get numerically tractable integrals by comparing the reward rates between two policies that wait different times $t_2 > t_1$:

$$\rho^{\pi_{t_2}}(R_n) - \rho^{\pi_{t_1}}(R_n) = \frac{p_r}{\lambda}(R_n - R_{oo}) \int_{t_1}^{t_2} \frac{\exp(-t'/\lambda)}{t'} dt' - (1-p_r)R_{oo}\left(\frac{1}{t_2} - \frac{1}{t_1}\right) - p_r k \left(e^{-t_1/\lambda} - e^{-t_2/\lambda}\right) dt' \tag{23}$$

7.2.5 POMDP formulation: unknown time penalty

linear sensitivity is optimal in the case of one, unknown time penalty k, but in the full rat task, there are potentially multiple time penalties for the 3 difference blocks. When k, is an unknown quantity, a POMPDP formulation would augment the state of the system to incorporate the belief about each block occurring, p_{block}: $S(p_B, n, R_n, t)$. Optimal wait times would try to optimize for expected time penalty over potential time penalties, k_i

$$E[k] = \sum_i p_i k_i \tag{24}$$

Substituting this into eq. 1 gives the optimal wait time in the presence of unknown blocks:

$$t^*_{POMPDP} = \lambda \left(\log[R_n p_r] - \log[E[k].\lambda]\right) \tag{25}$$

The interpretation of this result is that an optimal agent would not infer being in a single block and then choose a wait time based upon that time penalty, which would be a heuristic to optimal decision making. Nor would it average wait times over the possible wait times for each block. Instead, it would wait based upon the probability-weighted average of the possible time penalties across blocks. In the case that beliefs about blocks are very high, the heuristic of inferring a single block would have strong correspondence, but instances in which beliefs are weak about any particular block (middle of mixed blocks, or block transitions to adaptation blocks) would be able to assess whether rats are behaving optimally, or using a heuristic.

7.2.6 Optimal MVT-driven decision making: self-consistent reward rate and optimal policy equations

If the time penalty is due the opportunity cost of possible other offers from the environment, then the optimal wait time and reward rate equations are found by replacing k with $\rho\Delta t$ the following:

$$t^*_{MVT} = \lambda \left(\log[R_n p_r] - \log[\rho \Delta t \lambda]\right) \tag{26}$$

$$\begin{aligned}\rho^{\pi_{t^*_{MVT}}}(R_n) &= p_r R_n \int_{t_0}^{t^*_{MVT}} \frac{p(t')}{t'} dt' - p_r R_{oo} \int_{t^*_{MVT}}^{\infty} \frac{p(t')}{t'} dt' - \frac{(1-p_r)R_{oo}}{t} - p_r \rho \Delta t \int_{t_0}^{t^*_{MVT}} p(t') dt' - (1-p_r)\rho\Delta t \\ &= \frac{1}{1 + \Delta t \int_{t_0}^{t^*_{MVT}} p(t')dt' + (1-p_r)\Delta t} \left(p_r R_n \int_{t_0}^{t^*_{MVT}} \frac{p(t')}{t'} dt' - p_r R_{oo} \int_{t^*_{MVT}}^{\infty} \frac{p(t')}{t'} dt' - \frac{(1-p_r)R_{oo}}{t} \right)\end{aligned} \tag{27}$$

Eqs. (26-27) are known as self-consistent equations for t^*_{MVT} and $\rho^{\pi_{t^*_{MVT}}}(R_n)$, meaning that they are a set of nonlinear equations that must be they must simultaneously satisfied. Self consistent equations can be solved by iteration, where an initial guess at ρ can be used to calculate t, which can be used to solve for a new reward rate, and so forth.

Results of this are shown in Figures 7.2.6. One important note is that the wait times can be negative unless the lower limit t_0 of the exponential integrals is set to be very low, in this case $t_0 = 0.01$. The motivation here is not to show the absolute optimal wait times, but rather the trends among blocks. It also suggests an iterative approach for developing a strategy that incorporates the marginal value theorem: Take some guess at the reward rate of the environment and then base your wait times from that. Based on the rewards from that policy, calculate a new reward rate and repeat the process of following a new set of optimal wait times.

7.2.7 Total reward, deterministic policies

Consider a deterministic policy that waits for a set time t' on each trial, denoted as $\pi_{t'}$. How much reward should you expect it to receive? This is the expected total return for a known offer starting from beginning of trial (i.e., value of initial state), given by $V^{\pi_{t'}}[S(R_n, t=0)]$

$$\begin{aligned}V^{\pi_{t'}}[S(R_n, t=0)] = p_r &\left[P(0,t')R_n - \int_0^{t'} (1 - P(0,t))[k + R_{oo}\delta(t-t')]dt \right] \\ &- (1-p_r) \int_0^{t'} [t'k + R_{oo}\delta(t-t')]\, dt\end{aligned} \tag{28}$$

$$P(0,t') = \int_0^{t'} p(t)dt = 1 - \exp(-t'/\lambda) \tag{29}$$

Here $\delta(t-t')$ is a Dirac delta function that signifies the opt out penalty occurs only at $t = t'$. On non-catch trials you will receive the reward R_n from the environment with probability $P(0,t')$, which is the cumulative probability of reward occurring between 0 and t'. You will also incur a wait time penalty at every time step (hence the integral) with probability that reward has not occurred yet: $1 - P(0,t) = \exp(-t'/\lambda)$. Additionally, on non-catch trials, you also incur a time-out if reward does not arrive by t'. On catch trials, you will simply accumulate a wait time penalty over the time interval $[0, t']$, and will incur the opt out penalty.

Simplifying this a bit, expected value for a known offer is then

$$\begin{aligned}V^{\pi_{t'}}[S(R_n, t=0)] = p_r &\left[(1 - e^{(-t'/\lambda)})R_n - \lambda \left(1 - e^{-t'/\lambda}\right) k - e^{-t'/\lambda} R_{oo} \right] \\ &- (1-p_r)\left[t'k + R_{oo}\right]\end{aligned} \tag{30}$$

The expected value for an unknown offer from a given block would simply substitute R_n with the average reward from that environment:

$$\begin{aligned}V^{\pi_{t'}}[\cdot, t=0] = p_r &\left[(1 - e^{(-t'/\lambda)})\hat{R}_b - \lambda \left(1 - e^{(-t'/\lambda)}\right) k - e^{-t'/\lambda} R_{oo} \right] \\ &- (1-p_r)\left[t'k + R_{oo}\right]\end{aligned} \tag{31}$$

However, for a policy that waits the "optimal" wait time t^* for each different offer value, the value of an unknown offer is then the average of values for each offer size

$$V^{\pi_{t^*}}[S(\cdot, t=0)] = \frac{1}{N} \sum_{i=1}^{N} V^{\pi_{t^*}}[R_i, t=0] \tag{32}$$

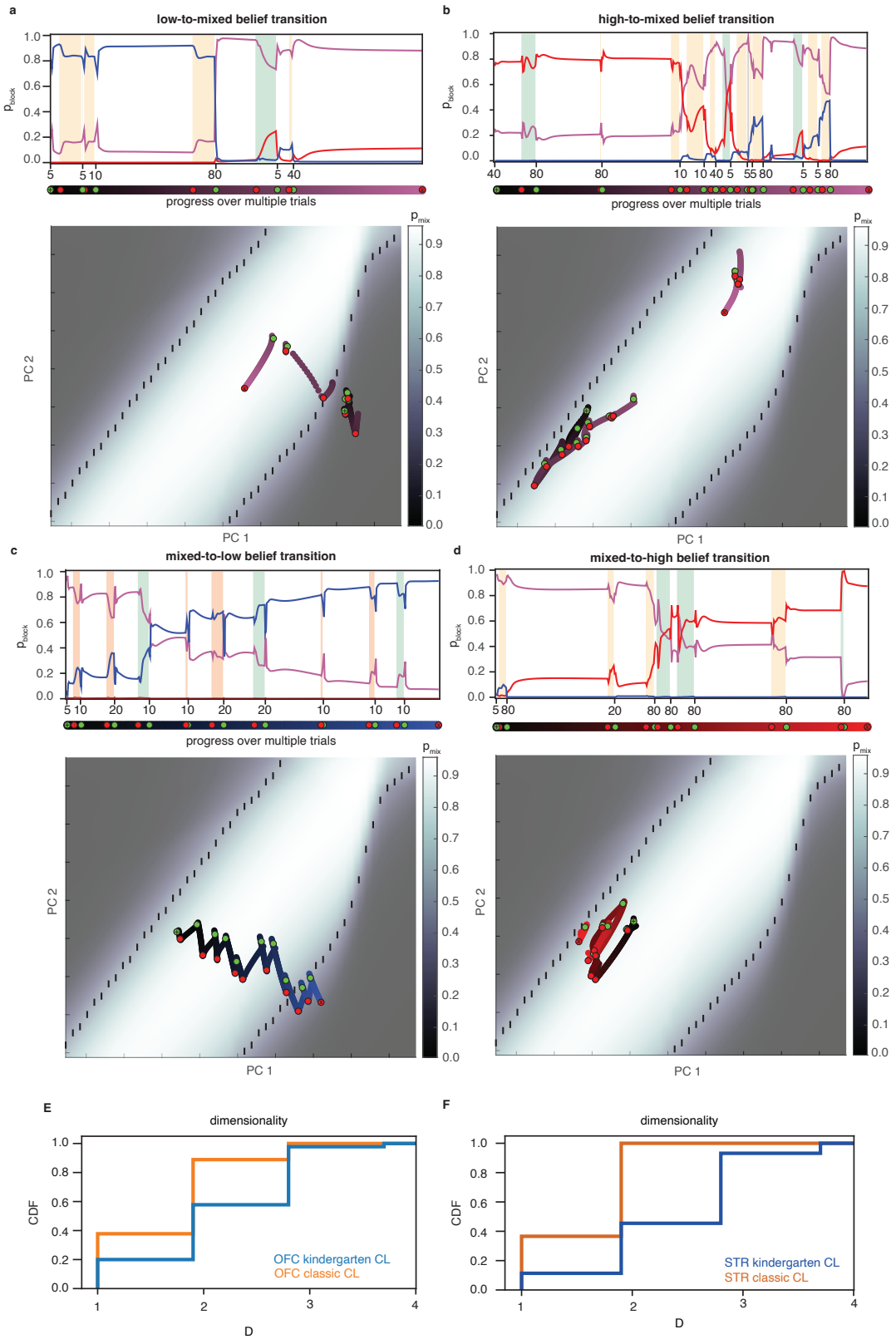

Figure S3: **a)-d)**Updating beliefs across all block transition types, analogous to Fig. 5 e. top: p_{block} over the block transition. Trial offers are on the x axis. Colored regions denote the ITI between trials when the agent either opted out (orange) or received reward (green). White regions are the waiting epoch. Progress across multiple trials is also shown as a colored line below, with start denoted as a green dot, and the ITI as a red dot. bottom: analogous evolution of state activity over the trials. The background color denotes p_{mix}, and the $p_{\text{mix}} = 0.5$ boundary is denoted with a dotted line. Trial progress uses the same convention from left plot. **e)-f)** CDF of dimensionality for different CL types in fully trained networks. in **e** *: p=0.0013, **f** *: p = 0.99, KS test.

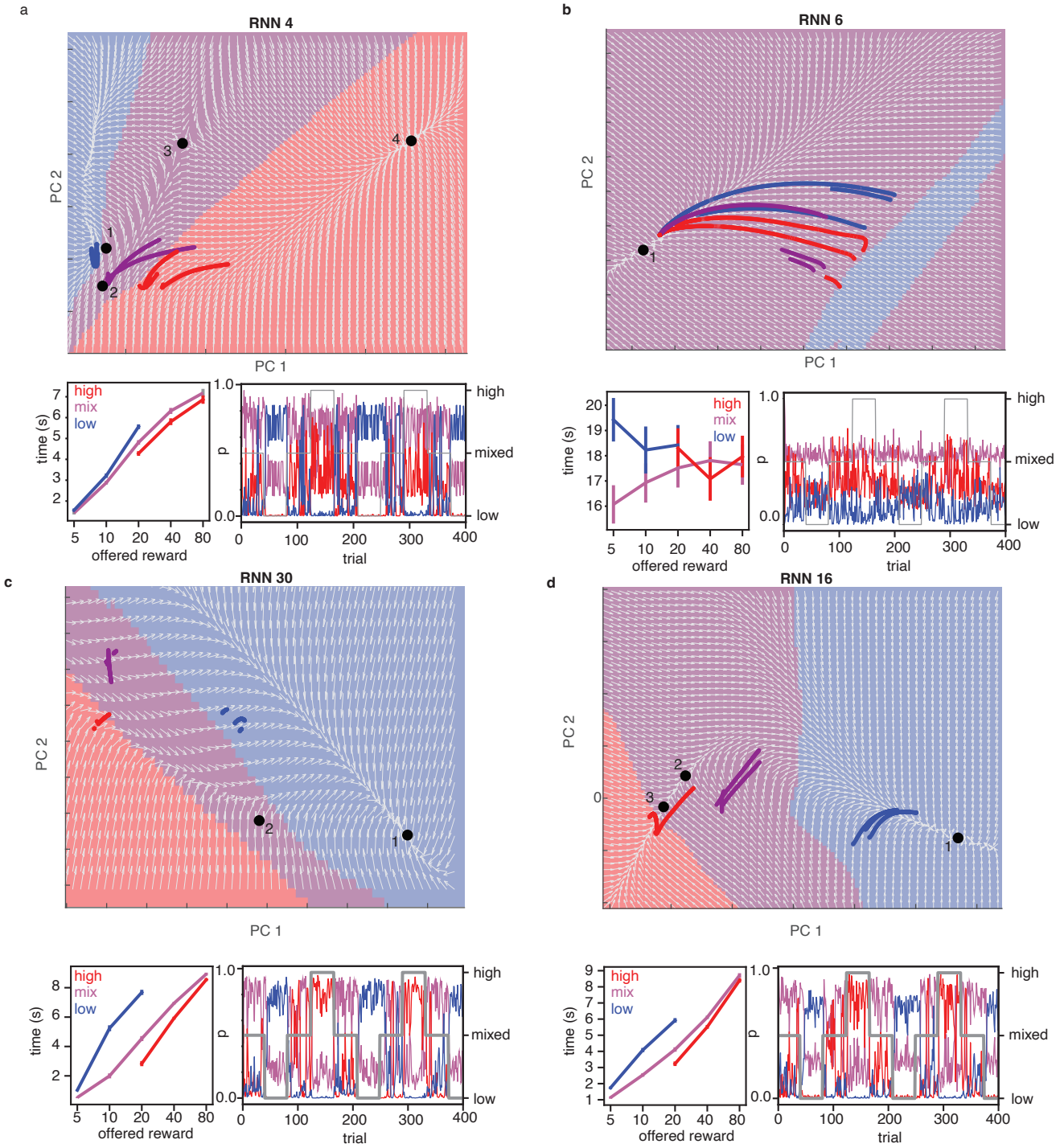

Figure S4: Examples of other dynamical systems motifs in top two PCS (top) found during training, along with their example wait time curves on catch trials (bottom left) and inference performance demonstrated by plotting p_{block} over trials (bottom right). In inference performance plots, blocks are denoted by gray line. **a)** sample network with same dynamics as in Fig. 5 a (e.g., stable attractor dynamics in high and low blocks, saddle point in mixed block). **b)** Network with poor inference performance that fails to partition state space, and has different dynamical systems features. **c)-d)** Alternative dynamical systems motifs supporting rat-like wait time behavior

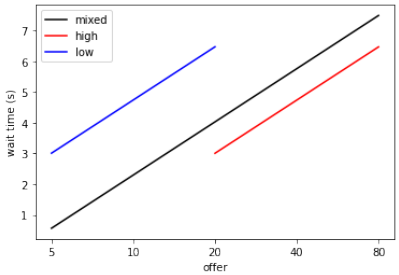

Figure S5: Results of SCF iteration. A) dt = 0.05 , B) dt = 0.01, C) convergence progress of the SCF iteration