Evidence accumulation in deep RL agents powered by a cognitive model

James Mochizuki-Freeman, Sahaj Singh Maini, Zoran Tiganj
{jmochizu, sahmaini, ztiganj}@iu.edu
Department of Computer Science, Indiana University Bloomington

Abstract

Evidence accumulation is thought to be fundamental for decision-making in humans and other mammals. Neuroscience studies suggest that the hippocampus encodes a low-dimensional ordered representation of evidence through sequential neural activity. Cognitive modelers have proposed a mechanism by which such sequential activity could emerge through the modulation of recurrent weights with a change in the amount of evidence. Here we integrated a cognitive science model inside a Reinforcement Learning (RL) agent and trained the agent to perform a simple evidence accumulation task inspired by the behavioral experiments on animals. We compared the agent's performance with the performance of agents equipped with GRUs and RNNs. We found that the agent based on a cognitive model was able to learn much faster and generalize better while having significantly fewer parameters. This study illustrates how integrating cognitive models and deep learning systems can lead to brain-like neural representations that can improve learning.

1 Introduction

Converging evidence from cognitive science and neuroscience suggests that the brain represents physical and abstract variables in a structured form, as mental or cognitive maps. A remarkable example of this is a recent study by Nieh et al. [11], in which mice were trained to perform an "accumulating towers task". In this task mice moved along a virtual track and observed objects (towers) on the left- and right-hand sides. When they arrived at the end of the track, to receive a reward, they had to turn left or right, depending on which side had more towers. The difference in the number of towers here corresponds to the amount of evidence for turning left *vs.* turning right. Nieh et al. [11] recorded the activity of hundreds of individual neurons from mice hippocampus. The results indicated the existence of cells tuned to a particular difference in the number of towers, such that a population of neurons tiles the entire *evidence* axis (see also [10]).

Cognitive scientists have developed elaborate models of evidence accumulation to explain the response time in a variety of behavioral tasks [7, 8, 14]. These models hypothesize that the brain contains an internal variable that represents the progress towards the decision. A neural-level cognitive model proposed that the brain could implement this process using a memory model based on the Laplace transform [6]. The Laplace framework gives rise to map-like representations, and it has been successful in describing the emergence of sequentially activated time cells [15] and place cells [4, 3].

Here we integrate the Laplace framework into Reinforcement Learning (RL) agents. The Laplace framework is based on recurrent neurons with analytically computed weights. We use the Laplace and inverse Laplace transform to generate a map-like representation of the amount of evidence. This representation is then fed into a trainable RL module based on the A2C architecture [9]. We compare map-based agents to standard RL agents that use simple Recurrent Neural Networks (RNNs) and Gated Recurrent Units (GRUs) [1]. The comparison is made in terms of performance and similarity of the neural activity to neural activity recorded in the brain.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

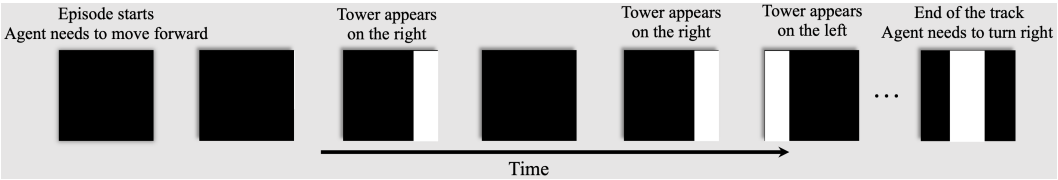

Figure 1: Schematic of the accumulating towers environment. In this simple example, two towers appeared on the right, and one tower appeared on the left, so the agent has to turn right once it reaches the end of the track.

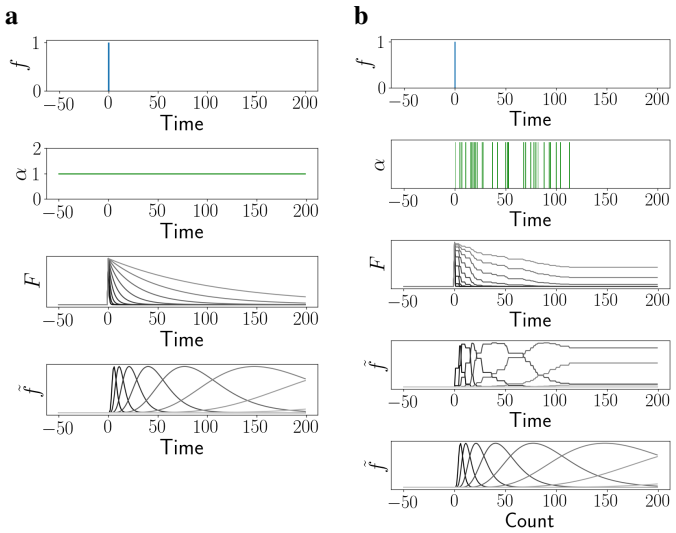

Figure 2: Example of the Laplace and inverse Laplace transform with and without modulatory input. (a) In the absence of modulatory input ($\alpha = 1$) the impulse response of the Laplace transform decays exponentially with decay rate s. The impulse response of the inverse Laplace transform has a unimodal shape. Note that if time t was shown on the log-scale, the unimodal curves would be equally wide and equidistant. (b) α modulates the decay rate of F and it is proportional to the change in the count. This makes units in $\tilde{f}$ develop unimodal basis functions that are tuned to count rather than to time and peak at n^*.

2 Methods

2.1 Accumulating towers task

We designed a simple version of the "accumulating towers task". Agents had to navigate down a virtual track composed of only three inputs: left, right and middle. In each episode, agents start from the beginning of the virtual track and observe towers (represented by the input value changing from 0 to 1) on each side of the environment (Fig. 1). Agents had four available actions: left, right, forward, and backward. Positions of towers were decided randomly in each episode. Similar to the neuroscience studies, the maximum number of towers on one side was 14. Once the agent arrived at the end of the track, the middle input changed from zero to one, signifying that it hit the wall. In order to receive the reward, the agent had to turn left or right, depending on which side had more towers. We set the magnitude of the reward to 10, penalize the agents for hitting the wall at the end of the track or going backward with a -1 negative reward and penalize the agent for hitting the side walls with a -0.1 negative reward.

2.2 RNN implementation of evidence accumulation using the Laplace domain

Evidence accumulation implemented through the Laplace domain is a key component of the cognitively-inspired RL agent. We write a discrete-time approximation of the Laplace transform as

an RNN with a diagonal connectivity matrix and a linear activation function:

$$F_{s;t} = WF_{s;t-1} + f_t, \tag{1}$$

where $W = \text{diag}(e^{-\alpha(t)s\Delta t})$. We define α as a change in numerosity: $\alpha = \Delta n/\Delta t$ and set $f_t = \delta(t = 0)$. Thus the rate of change of $F_{s;t}$ is modulated by α. If $\alpha = 1$, $F_{s;t}$ decays exponentially with rate constant s. This modulation converts a function of time into a function of numerosity. Consequently, the inverse Laplace transform $\tilde{f}_{n^*;n}$ with $n^* = k/s$, which can be computed by applying a k-th order derivative to $F_{s;t}$ (see Appendix for detailed derivation), will result in neurons tuned to log-spaced values of n^* (Fig. 2). Log-spaced n^* and scale-invariance of the inverse Laplace transform give rise to a log-compressed representation of numerosity, consistent with the Weber-Fechner law [2, 12].

In tasks where evidence corresponds to a difference of some quantities, such as the accumulating towers task, we can use the Laplace domain to perform the subtraction by computing a cross-correlation between the $\tilde{f}$ functions (which is equivalent to a product in the Laplace domain) since they represent the different counts. This will result in a new $\tilde{f}$ function with neurons tuned to the amount of evidence (see Appendix for detailed derivation).

2.3 Agent architecture

The agents received three inputs from the environment fed directly into the recurrent layer (Fig. A1). The recurrent layer was either RNN, GRU or an RNN based on the Laplace framework as described above. When the Laplace framework was used, we had three independent $\tilde{f}$ modules. Each module had 20 n^* values spaced logarithmically from 5 to 100. The value of parameter k was set to 8. Parameter k controls the sharpness of the unimodal basis functions, and this value was chosen to ensure no gaps between them. Note also that the input into the $\tilde{f}$ modules was delivered to α, which controls recurrent weights of the population of units F. This is a conceptual difference in comparison to other RNNs, where recurrent weights are tuned separately for each unit. The strength of the proposed approach is that the populations of neurons encode functions over relevant variables such that $\tilde{f}$ directly represents the count of the objects. In addition to computing $\tilde{f}$, we also computed the subtraction $\tilde{f}_{sub}$ of each pair of $\tilde{f}$. This was done by computing the product in the Laplace domain between each pair of F as described in the previous section. The total number of units was 180 (20 units per module, 3 independent modules and 6 subtraction modules). When other RNNs were used, the dense layer was mapped to 180 recurrent units. The output of the recurrent layer was passed to an actor network and a critic network. Both actor and critic consist of a single layer fully connected neural network. For all agents, we explored two different learning rates 0.001 and 0.0001.

3 Results

We trained and evaluated agents in an accumulating towers RL environment. We compared several agents based on the Laplace framework. One pair of agents used the inverse Laplace transform and either included the subtraction ($\tilde{f}_{sub}$) or excluded the subtraction ($\tilde{f}$). The other pair was without the inverse Laplace transform, again both including the subtraction (F_{sub}) and excluding it (F). We also compared agents based on a simple RNN and GRU, as well as versions of those agents with frozen recurrent weights.

The agents were trained and evaluated in 300 steps long environment. We trained four different agents for each of the eight models and performed 100 validation runs every 500 episodes. While other agents started increasing before $\tilde{f}_{sub}$, $\tilde{f}_{sub}$ agents were the first to learn the task and converge to a reward value of 10 (Fig. 3, Fig. A9 and the first column in Table A1). F_{sub} agents converged next, but they showed progress in learning faster than other agents. Taken together, these results indicate the importance of the subtraction operation. GRU agents managed to reach performance very close to 10, indicating that they can learn the task as well. On the other hand, the RNN agents did not learn the task in 100k episodes, indicating that gating was important for correct performance.

To test the ability of the agents to generalize, we also evaluated them on 3000 and 10000 steps long tracks without ever training them on tracks of that length (second and third column in Table A1). Agents based on the cognitive model showed great resilience to this kind of rescaling. This is not surprising since the representation was designed to change with the change in the amount of evidence.

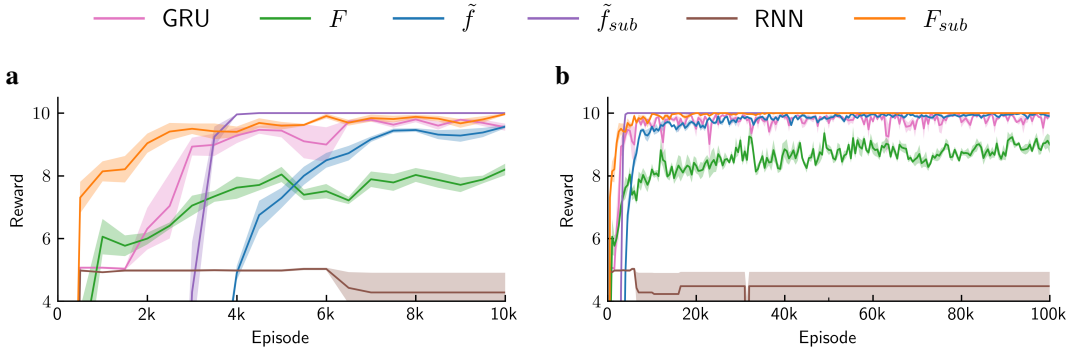

Figure 3: Agent performance on accumulating towers task. The two plots show two different zoom levels.

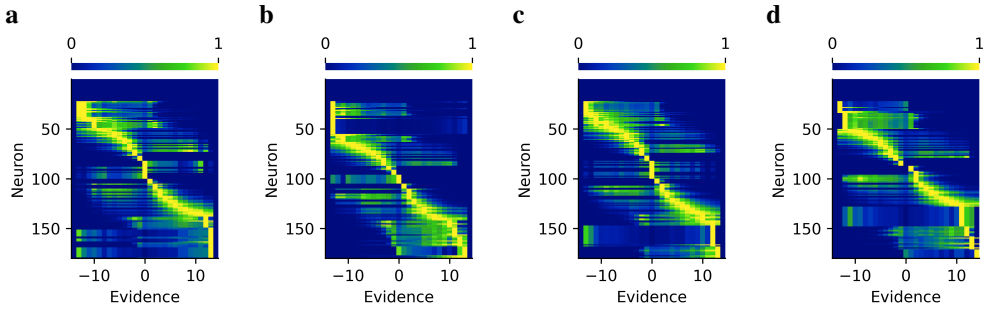

Figure 4: The activity of neurons in ($\tilde{f}_{sub}$) after 100k episodes of the accumulating towers task resembles the activity of neurons recorded in the hippocampus. Similar to plots in [11, 10], neurons are sorted by peak activity. Each row is normalized such that the activity ranges from 0 to 1.

On the other hand, the performance of GRU agents dropped at unseen track lengths but remained well above chance.

We visualized the neural activity of each of the four agents for each of eight models after 100k episodes of training. As expected, neurons in $\tilde{f}_{sub}$ agents activated sequentially as a function of evidence resembling the activity in neural recordings from the hippocampus [11, 10] (Fig. 4). Some of the neurons in GRU agents showed tuning to the magnitude of evidence with often prominent asymmetry between positive and negative amounts of evidence (Fig. A3).

4 Conclusions

This work provides an example of incorporating models from cognitive and computational neuroscience into artificial neural networks trained using error backpropagation. Agents based on the cognitive model were able to learn faster and generalize better despite having fewer parameters. This indicates that the A2C algorithm was able to use the neural representation from the cognitive model. This representation also resembled data from neural recordings in [11, 10] characterized with the sequential activation as a function of the amount of evidence.

Acknowledgment

We gratefully acknowledge support from the Defense Advanced Research Projects Agency (DARPA) under project Time-Aware Machine Intelligence (TAMI) and the National Institutes of Health's National Institute on Aging, grant 5R01AG076198-02. This content is solely the responsibility of the authors and does not necessarily represent the official views of DARPA or the National Institutes of Health's National Institute on Aging. This research was supported in part by Lilly Endowment, Inc., through its support for the Indiana University Pervasive Technology Institute.

References

[1] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[2] G. Fechner. *Elements of Psychophysics. Vol. I.* Houghton Mifflin, 1860/1912.

[3] M. W. Howard and M. E. Hasselmo. Cognitive computation using neural representations of time and space in the laplace domain. *arXiv preprint arXiv:2003.11668*, 2020.

[4] M. W. Howard, C. J. MacDonald, Z. Tiganj, K. H. Shankar, Q. Du, M. E. Hasselmo, and H. Eichenbaum. A unified mathematical framework for coding time, space, and sequences in the hippocampal region. *Journal of Neuroscience*, 34(13):4692–707, 2014. doi: 10.1523/JNEUROSCI.5808-12.2014.

[5] M. W. Howard, K. H. Shankar, and Z. Tiganj. Efficient neural computation in the laplace domain. In *Cognitive computations workshop at Advances in neural information processing systems*, 2015.

[6] M. W. Howard, A. Luzardo, and Z. Tiganj. Evidence accumulation in a laplace domain decision space. *Computational brain & behavior*, 1(3):237–251, 2018.

[7] D. R. J. Laming. Information theory of choice-reaction times. 1968.

[8] S. W. Link. The relative judgment theory of two choice response time. *Journal of Mathematical Psychology*, 12(1):114–135, 1975.

[9] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[10] A. S. Morcos and C. D. Harvey. History-dependent variability in population dynamics during evidence accumulation in cortex. *Nature neuroscience*, 19(12):1672–1681, 2016.

[11] E. H. Nieh, M. Schottdorf, N. W. Freeman, R. J. Low, S. Lewallen, S. A. Koay, L. Pinto, J. L. Gauthier, C. D. Brody, and D. W. Tank. Geometry of abstract learned knowledge in the hippocampus. *Nature*, pages 1–5, 2021.

[12] R. Portugal and B. F. Svaiter. Weber-fechner law and the optimality of the logarithmic scale. *Minds and Machines*, 21(1):73–81, 2011.

[13] E. Post. Generalized differentiation. *Transactions of the American Mathematical Society*, 32:723–781, 1930.

[14] R. Ratcliff. A theory of memory retrieval. *Psychological Review*, 85:59–108, 1978.

[15] K. H. Shankar and M. W. Howard. A scale-invariant internal representation of time. *Neural Computation*, 24(1):134–193, 2012.

A Appendix

Converting a Laplace-transform based memory model into a numerosity representation

We define the Laplace transform of function $f(t)$ from $-\infty$ to the present t:

$$F(s;t) = \int_0^t e^{-s\left(t-t'\right)} f(t')dt'. \tag{2}$$

We restrict variable s to real positive values. [1]

The above equation can be expressed in a differential form where s appears as a rate constant of a leaky integrator:

$$\frac{dF(s;t)}{dt} = -sF(s;t) + f(t). \tag{3}$$

Fig. 2a shows the impulse response of the above equation for several values of s.

To convert a representation of time into a representation of numerosity $n(t)$ (how many times some input was observed) we modulate s with a rate of change α expressed as a time derivative of numerosity ($\alpha = dn/dt$):

$$\frac{dF(s;t)}{dt} = \frac{dn}{dt}\left(-sF(s;t) + f(t)\right). \tag{4}$$

By reorganizing terms in the above equation and applying the chain rule we can rewrite the equation as a function of n, instead of t (Fig. 2b):

$$\frac{dF(s;n)}{dn} = -sF(s;n) + f(n), \tag{5}$$

and we set $f(n) = \delta(t = 0)$.

Inverting the Laplace transform reconstructs the input as a function of the internal variable n^*, which corresponds to n. The inverse, which we denote as $\tilde{f}(n^*;t)$ can be computed using the Post inversion formula [13]:

$$\tilde{f}(n^*;n) = \mathbf{L}_k^{-1} F(s;n) = \frac{(-1)^k}{k!} s^{k+1} \frac{d^k}{ds^k} F(s;n), \tag{6}$$

where $n^* := k/s$ and $k \to \infty$. As we show below, the reconstruction gives rise to units tuned to a particular n. By solving $\partial \tilde{f}_{n^*;n}/\partial n = 0$ we see that $\tilde{f}(n^*;n)$ peaks at $n^* = n$. For s being a continuous variable and $k \to \infty$, the width of the peak is infinitesimally small, providing a perfect reconstruction of the observed quantity.

Discrete implementation

For a neural network implementation, we discretize the Laplace and inverse Laplace transform for both s and t. To select values s in an informed way, we compute the impulse response of $\tilde{f}_{n^*;n}$:

$$\tilde{f}_{n^*;n} = \frac{1}{u(t)} \frac{k^{k+1}}{k!} \left(\frac{u(t)}{n^*}\right)^{k+1} e^{-k\frac{u(t)}{n^*}}, \tag{7}$$

where $u = \sum_{i=0}^{t} \alpha(t_i)$. When s is discrete and k is finite, $\tilde{f}_{n^*;n}$ is a set of unimodal basis functions (when s is continuous and $k \to \infty$, those unimodal basis functions turn into delta functions with spacing $\to 0$). The coefficient of variation of $\tilde{f}_{n^*;n}$ is independent of n^* and n: $c = 1/\sqrt{k+1}$. This implies that the width of the unimodal basis functions increases linearly with their peak time. When observed as a function of $\log(n)$, the width of the unimodal basis functions is constant. This property of the Post inversion formula is relevant for modeling human perception due to the Weber-Fechner law [2, 12]. This law states that the relationship between the perceived magnitude of the stimulus and its true magnitude is logarithmic, motivating the use of logarithmic units such as decibel and candela. To ensure equidistant spacing of unimodal basis functions along the log-axis we space n^* logarithmically. This results in dramatic conservation of resources, especially when representing large quantities since the number of units in $\tilde{f}_{n^*;n}$ grows as a function of $log(n)$ rather than n. Note that fixing the values of n^* and choosing k also fixes values of s since $s = k/n^*$.

[1] The Laplace transform defines s as a complex variable. This choice would result in exponentially growing and oscillatory neural activity, causing numerical instabilities when computing the inverse Laplace transform.

Subtraction of functions using the Laplace domain

In the accumulating towers task, Eq. 5 can enable the agent to learn to represent the number of towers on each side. However, the latent variable that should determine the agent's decision is not the number of towers on each side but the difference between those numbers (the agent needs to turn towards the side which had more towers). This is a non-trivial problem since the number of towers is not represented as a scalar but as a function over n. Fortuitously, the Laplace domain enables access to a number of useful operations, including subtraction of two functions [5]. To show this, let us define $f(a)$ and $g(a)$ as functions representing two distributions of possible values for the number a in the range 0 to a_{max}. Outside this range, the functions are assumed to vanish. We define the operation of subtraction of these two distributions $[f-g](a)$ to be the cross-correlation of the two functions:

$$[f-g](a) \equiv \int_0^\infty f(x')g(a+x')dx'. \tag{8}$$

To illustrate that the above operation results in subtraction of two functions, consider a simple case where each of the functions is a delta function: $f=\delta(a_1)$ and $g=\delta(a_2)$. Then $[f-g]$ is a delta function at a_1-a_2. To implement cross-correlation in the Laplace domain we can turn Eq. 8 into convolution by reflecting $g(a)$ around a_{max}: $g_r=g(a_{max}-a)$. Point-wise product of the Laplace transforms of two functions corresponds to their convolution in the time domain. Point-wise multiplication of the Laplace transform of $f(a)$ and $g_r(a)$ corresponds to cross-correlation of $f(a)$ and $g(a)$ in the time domain, which is equivalent to their subtraction $[f-g](a)$. Note that for subtraction we need to consider both positive and negative values. Since we only use positive values of s, we are not able to directly represent the negative axis. To work around this, we compute both $[f-g](a)$ and $[g-f](a)$.

	d=300	d=3000	d=10000	# Parameters
$\tilde{f}_{sub}$	**10.000 ± 0.000**	**10.000 ± 0.000**	**10.000 ± 0.000**	724
$\tilde{f}$	9.903 ± 0.066	9.925 ± 0.065	9.925 ± 0.041	**244**
F_{sub}	**10.000 ± 0.000**	**10.000 ± 0.000**	**10.000 ± 0.000**	724
F	8.995 ± 0.287	8.675 ± 0.361	8.900 ± 0.272	**244**
RNN	4.475 ± 0.456	4.600 ± 0.281	4.700 ± 0.576	34024
GRU	9.980 ± 0.000	9.600 ± 0.146	8.425 ± 0.504	100624
RNN_{FROZEN}	−21.0 ± 0.000	−201.0 ± 0.000	−601.0 ± 0.000	724
GRU_{FROZEN}	−14.473 ± 5.653	−149.3 ± 44.77	−449.4 ± 131.3	724

Table A1: Mean reward +/- standard error across four runs after 100k episodes of training in d=300 steps long environment. Validation was done in 300, 3000 and 10000 steps long environments.

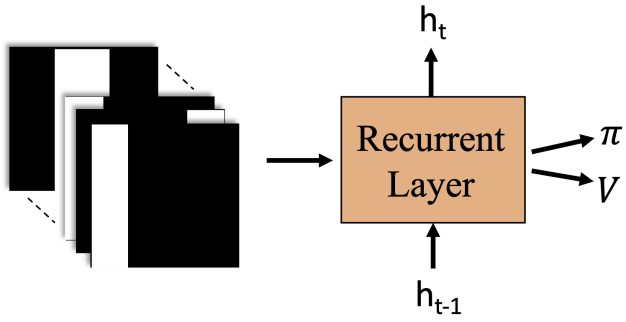

Figure A1: The agent architecture for accumulating towers task. We compare simple RNN, GRU and Laplace-based RNN described here.

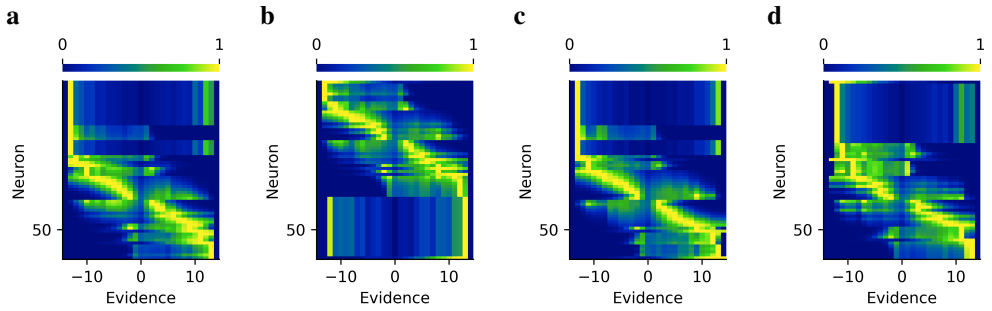

Figure A2: Same as Fig. 4 but for $\tilde{f}$ agents

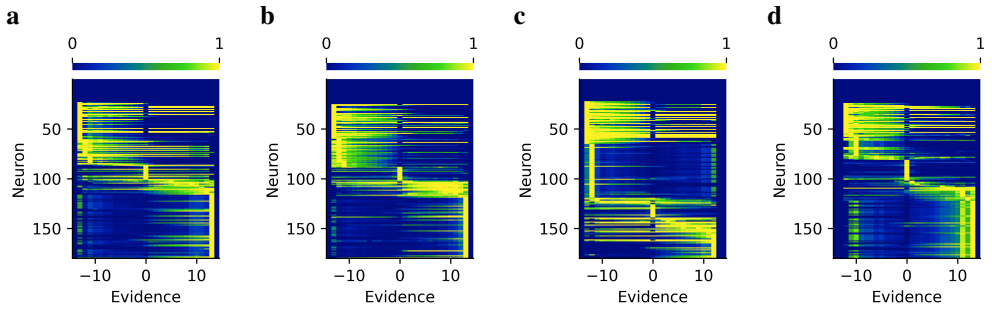

Figure A3: Same as Fig. 4 but for GRU agents.

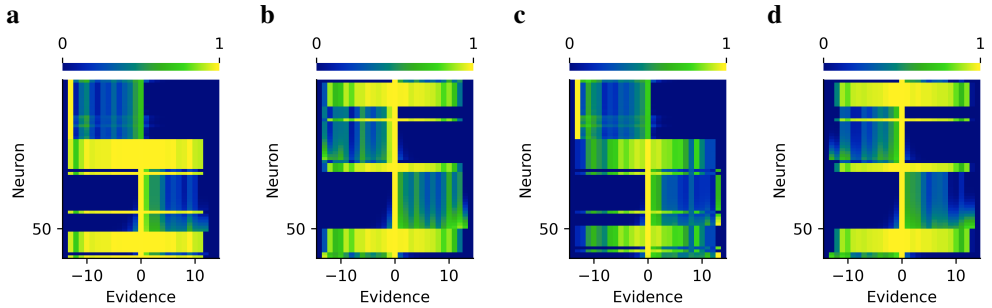

Figure A4: Same as Fig. 4 but for F agents

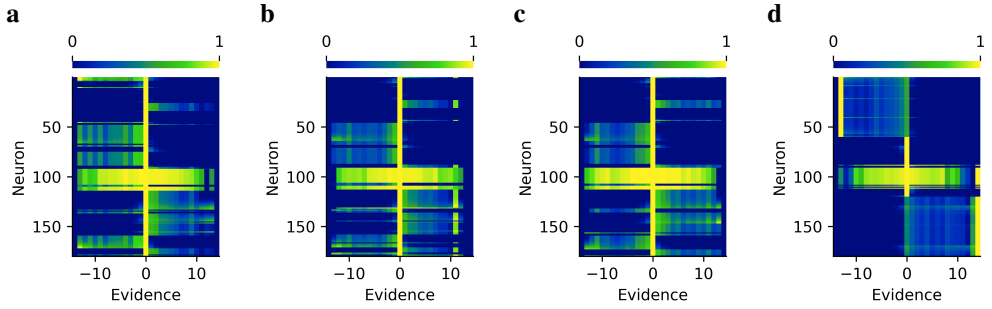

Figure A5: Same as Fig. 4 but for F_{sub} agents.

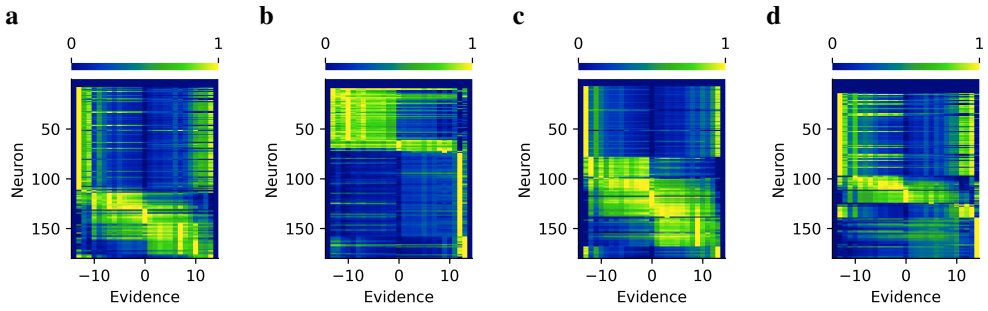

Figure A6: Same as Fig. 4 but for GRU$_{frozen}$ agents.

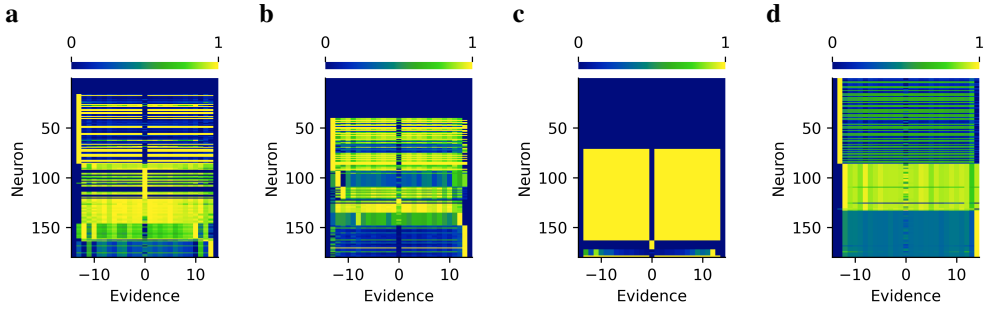

Figure A7: Same as Fig. 4 but for RNN agents

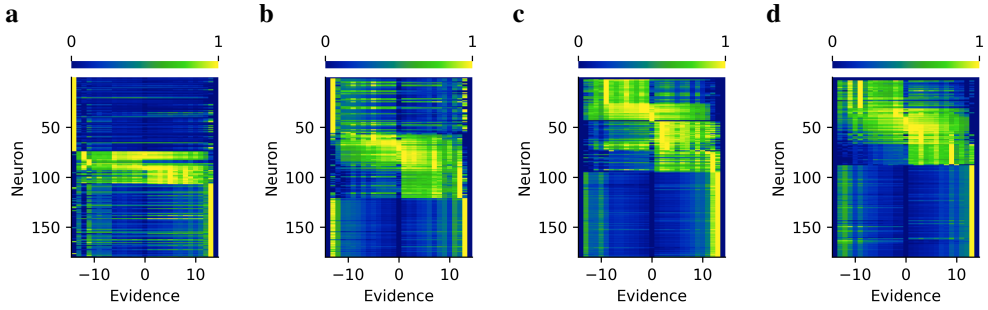

Figure A8: Same as Fig. 4 but for RNN$_{frozen}$ agents.

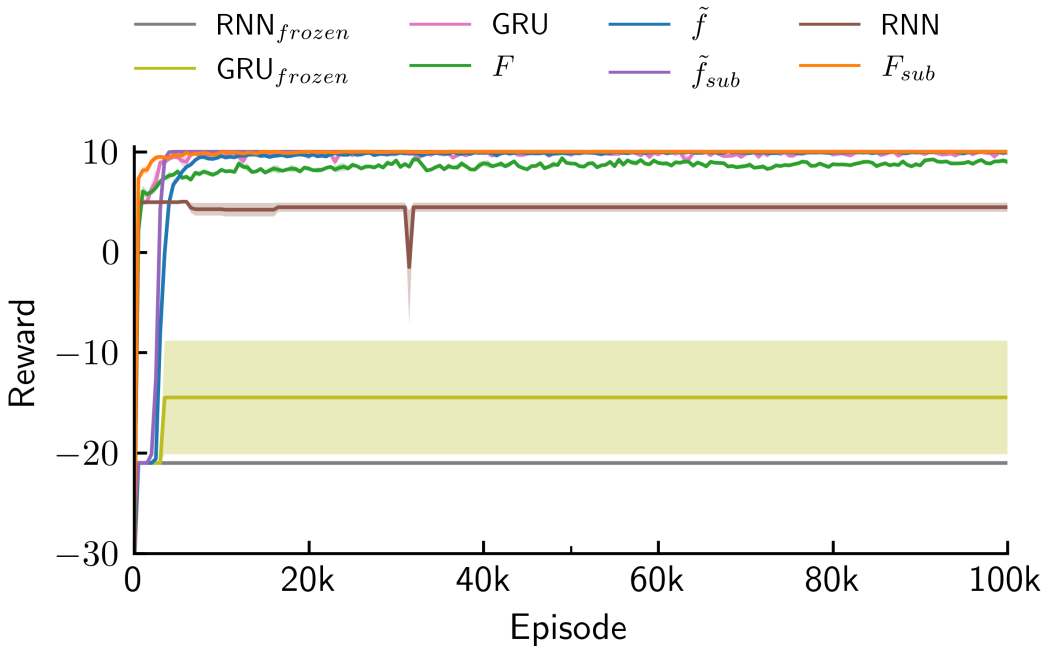

Figure A9: Agent performance on the accumulating towers task (same data as in Fig. 3 but different zoom level).