Article

Adaptive Behavior

From eye-blinks to state construction: Diagnostic benchmarks for online representation learning

Adaptive Behavior
2022, Vol. 0(0) 1–17

DOI: 10.1177/10597123221085039
journals.sagepub.com/home/adb

Banafsheh Rafiee[1], Zaheer Abbas[2], Sina Ghiassian[1], Raksha Kumaraswamy[1], Richard S Sutton[1,2], Elliot A Ludvig[3] and Adam White[1,2]

Abstract

We present three new diagnostic prediction problems inspired by classical-conditioning experiments to facilitate research in online prediction learning. Experiments in classical conditioning show that animals such as rabbits, pigeons, and dogs can make long temporal associations that enable multi-step prediction. To replicate this remarkable ability, an agent must construct an internal state representation that summarizes its interaction history. Recurrent neural networks can automatically construct state and learn temporal associations. However, the current training methods are prohibitively expensive for *online prediction*—continual learning on every time step—which is the focus of this paper. Our proposed problems test the learning capabilities that animals readily exhibit and highlight the limitations of the current recurrent learning methods. While the proposed problems are nontrivial, they are still amenable to extensive testing and analysis in the small-compute regime, thereby enabling researchers to study issues in isolation, ultimately accelerating progress towards scalable online representation learning methods.

Keywords

State construction, classical conditioning, diagnostic benchmarks, reinforcement learning

Handling Editor: Verena Hafner

1. Introduction

We consider the problem of multi-step prediction learning in a partially observable setting. In the multi-step prediction learning problem, the agent's objective is to use its sensory experience to predict signals of interest multiple steps into the future, just like when a reinforcement learning agent must predict future reward. In the partially observable setting, the agent must also construct an internal representation that summarizes its experience, as the immediate sensory information may not be sufficient for making accurate long-term predictions. Consider, for example, a rabbit trained to preemptively close its eyes by predicting a puff of air using another predictive stimulus, such as a tone, as shown in Figure 1. To appropriately time the eyeblink, the rabbit needs an internal representation of the elapsed time since the tone sounded. Neural network solution methods can be used for such problems (Tallec & Ollivier, 2018; Jaderberg et al., 2017; Dehghani et al., 2019; Gehring et al., 2017; Nath et al., 2019). Researchers use a variety of benchmarks to evaluate the progress of the neural network solution methods—toy problems, time-series data sets, NLP tasks, and large-scale navigation problems. We focus on the case in which the agent learns *online*: making and updating its predictions on every time step, even when the prediction target is not immediately available, as in temporal-difference (TD) learning (Sutton, 1988).

Benchmarks in reinforcement learning are relevant for evaluating multi-step predictions, but most are based on the fully observable setting. The Arcade Learning Environment (ALE) exhibits minor partial observability, but frame-stacking can be used to construct a state that can achieve good performance (Bellemare et al., 2013; Machado et al., 2018). OpenAI-Gym (Brockman et al., 2016) and MuJoCo (Todorov et al., 2012) offer a wide variety of tasks inspired by problems in robotics that are partially observable when

[1]Department of Computing Science and the Alberta Machine Intelligence Institute (Amii), University of Alberta, Edmonton, AB, Canada
[2]DeepMind Alberta, Edmonton, AB, Canada
[3]Department of Psychology, University of Warwick, Coventry, UK

Corresponding author:
Banafsheh Rafiee, Department of Computing Science and the Alberta Machine Intelligence Institute (Amii), University of Alberta, 116 St & 85 Ave, Edmonton, AB T6G 2R3, Canada.
Email: rafiee@ualberta.ca

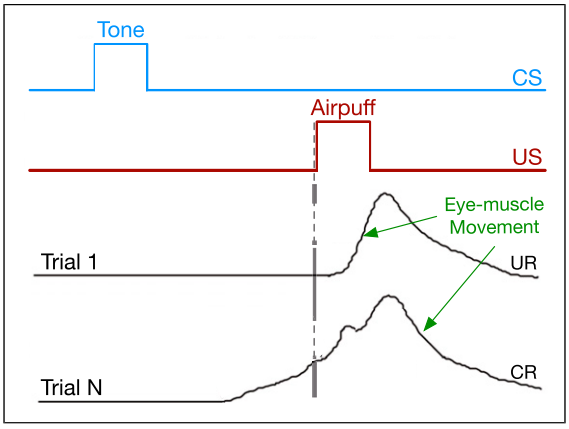


Figure 1. Eyeblink conditioning. After many pairings of the tone with the puff of air, the rabbit learns to close its inner eyelid (nictating membrane) before the puff of air is presented.

using only visual inputs. However, the focus is mostly on continuous actions and high-dimensional inputs from joint angles and velocities. The DeepMind Lab contains several 3D simulation problems inspired by experiments in neuroscience (Beattie et al., 2016; Wayne et al., 2018). Researchers have used these problems to benchmark large-scale learning systems; unfortunately, such experiments require several billion steps of interaction and cloud-scale compute (Beattie et al., 2016; Wayne et al., 2018; Parisotto et al., 2020a; Fortunato et al., 2019; Espeholt et al., 2018).

Diagnostic issue-oriented benchmarks serve different purposes than large-scale challenge problems. While the diagnostic benchmarks are simple, they still illuminate fundamental limitations of the existing methods. For example, the eight-state Black and White problem highlights the need for tracking in partially observable problems (Sutton et al., 2007), and DeepSea highlights how dithering exploration can be arbitrarily inefficient even in a grid world (Osband et al., 2019). Such diagnostic problems isolate specific algorithmic issues, and progress on these problems represents progress on the specific issues. Additionally, if a diagnostic benchmark has small compute requirements, then researchers can quickly evaluate new ideas and avoid the additional engineering complexity required to build high-performance, state-of-the-art architectures. Large problems often require complex architectures that can be difficult to analyze, and small implementation details can lead to incorrect conclusions (Engstrom et al., 2019; Tucker et al., 2018). Robust statistical analysis, experiment repetition, and ablations can be challenging in large-scale benchmarks because of the excessive computational requirements (see Machado et al. (2018); Henderson et al. (2018); Colas et al. (2018)).

Inspired by animal learning, this paper contributes a set of diagnostic benchmarks for the partially observable online prediction problem.[1] Our first problem, *trace conditioning*, requires an agent to predict a distal stimulus from a previously observed cue, just as a rabbit predicts an air puff based on a tone. The challenge here is representational: how does the agent bridge the gap between the tone and the air puff in a way that is not specific to the particular arrangement or timing of the stimuli (Ludvig et al., 2012; Sutton & Barto, 2018). Our second problem, *noisy patterning*, is inspired by biconditional patterning experiments (Mackintosh, 1974; Harris et al., 2008). This problem tests the agent's ability to determine which observation signals to pay attention to, in the presence of noise and distracting stimuli. Finally, our third benchmark, *trace patterning*, combines trace conditioning and noisy patterning and requires the agent to simultaneously discover the relevant observation signals and build their temporal representations.

Our second contribution is empirical. We use the proposed diagnostic problems to conduct a comprehensive empirical study of several state-of-the-art recurrent learning architectures, including Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) and related Gated Recurrent Units (GRU) (Cho et al., 2014), trained via Truncated Back-prop Through Time (T-BPTT) (Williams & Peng, 1990) and Real Time Recurrent Learning (RTRL) (Williams & Zipser, 1989). We systematically investigate each method's performance as we vary the key problem parameters. We also introduce a simple input augmentation scheme based on memory traces, improving both T-BPTT and RTRL based methods. In total, our results show that the proposed diagnostic problems can effectively isolate the limitations of the current training methods and help stimulate research in online representation learning.

2. Related work

In partially observable problems, the agent must construct an internal state to summarize the history of interaction in order to predict the future. This is often done by recurrent networks. An RNN uses hidden layers with recurrent connections trained via BPTT (Hopfield, 1982; Elman, 1990), in order to summarize the history of interaction. Storing network activations from the beginning of time is expensive, and so the update can be truncated T steps back in time (i.e., T-BPTT) (Williams & Peng, 1990). This presents a trade-off. If the truncation window is short, the agent cannot learn long temporal dependencies. If the truncation window is long, however, the agent can learn long temporal associations, but computation and memory costs grow with T. If the truncation window is shortened, then most recurrent systems including basic RNNs and LSTMs (and GRUs) cannot learn temporal relationships longer than T (Williams & Peng, 1990). This trade-off is particularly challenging in the online prediction setting where the agent's objective is to update and make a new prediction on each time step. Ideally, our state construction

methods would be able to learn dependencies greater than T without requiring proportional computation—as humans do.[2]

There are alternatives to T-BPTT, many based on RTRL; which is itself an approximation of the true gradient. For a fully connected network, RTRL requires quartic computation in the number of hidden states per step which makes online implementation with even modestly sized networks challenging (Williams & Zipser, 1989). Approximations of RTRL such as Unbiased Online Recurrent Optimization (UORO) (Tallec & Ollivier, 2018), synthetic gradient methods (Jaderberg et al., 2017), and SnAp (Menick et al., 2020) approximate the gradient back in time and thus suffer from the representability/computation trade-off of T-BPTT. We did not include UORO and SnAp as baselines in our experiments; we instead included the results from RTRL which both these methods approximate. We showed that the performance of RTRL significantly deteriorates as the temporal associations become longer, suggesting that its recent approximations will also have difficulty with the proposed benchmarks. In addition, prior work (Nath et al., 2019) found UORO to perform significantly worse than simpler T-BPTT variants in the related online predict k-steps ahead problem setting, suggesting that our benchmarks would be challenging for UORO.

Recent work has explored alternatives to overcome the trade-off, including alternative optimization schemes for RNNs (Nath et al., 2019), and learned sparse attention mechanisms combined with feedforward networks (Dehghani et al., 2019; Gehring et al., 2017). Fixed Point Propagation (Nath et al., 2019) has not been extended to our discounted multi-step prediction setting (estimating value functions).

Learned sparse attention mechanisms combined with feed-forward neural networks represent exciting alternatives for training RNNs. The best way to use attention strategies for partially observable reinforcement learning is still evolving (Parisotto et al., 2020b; Parisotto & Salakhutdinov, 2021; Loynd et al., 2020; Chen et al., 2021; Janner et al., 2021). Chen et al. (2021) and Janner et al. (2021) use transformers in the offline reinforcement learning setting. Parisotto et al. (2020b) and Parisotto and Salakhutdinov (2021) stack long sequences of past observations in order to learn long temporal dependencies. Therefore, they require at least linearly more resources as the span of temporal dependencies increases, which reintroduces the truncation trade-off. Combining transformers with mini-batches skewed more towards recent experiences (as shown to be effective in RL (Zhang & Sutton, 2017)) represents an interesting next step. However, more work is required to extend it to our online multi-step prediction learning setting. As these strategies are still beginning to be explored by the community, we leave these comparisons to future work.

Small diagnostic benchmarks like ours have a long history in online learning and reinforcement learning. Prior work on online supervised representation learning (Sutton & Whitehead, 1993; Mahmood & Sutton, 2013), step-size adaption methods (Sutton, 1992; Jacobsen et al., 2019), and divergence in temporal difference learning (Baird, 1995; Sutton & Barto, 2018) all make use of small diagnostic test problems to evaluate progress. More generally, small issue-focused problems are used pervasively in reinforcement learning to isolate and study research questions (see Sutton and Barto (2018)). The Deepmind Behavior Suite in many ways represents a modern attempt to organize and standardize a collection of interesting diagnostic test problems in reinforcement learning (Osband et al., 2020), similar in spirit to the Reinforcement Learning Competitions of old (Whiteson et al., 2010). Recent work has shown that classic toy problems like Mountain Car and Acrobot can be used to highlight the advantages of fairly complex modern architectures like Rainbow (Obando-Ceron & Castro, 2020), with a fraction of the computation typically required to run ALE experiments. Our diagnostic benchmarks can be accurately thought of as Prediction Suite.

3. Classical conditioning as representation learning

The study of multi-step prediction learning in the face of partial observability dates back to the origins of classical conditioning. Pavlov was perhaps the first to observe that animals form predictive relationships between sensory cues while training dogs to associate the sound of a metronome with the presentation of food (Pavlov, 1927). The animal uses the sound of a metronome (which is never associated with food in nature) to predict when the food will arrive, inducing a hardwired behavioral response. The ability of animals to learn the predictive relationship between stimuli is critical for survival. These responses could be preparatory like a dogs' salivation before food presentation or protective in case of anticipating danger like blinking to protect the eyes. Such predictions in the face of limited information are useful to humans too. You predict when the bus might stop next—and perhaps get off—based on the distal memory of the bell. You predict when the water from the tap might get too hot and move your hand in advance. The study of prediction, timing, and memory in natural systems remains of chief interest to those that wish to replicate it in artificial systems.

Some of the most relevant theories on multi-step prediction in animals have been explored in *trace conditioning*. In the classical setup, two stimuli are presented to the animal in sequence as shown in Figure 1. The first is called the conditioned stimulus or CS (the predictive trigger) which usually takes the form of a light or tone. Then an unconditioned stimulus (US), such as a puff of air to the animal's

eye, is presented which generates a behavioral response called the unconditioned response (UR)—the rabbit closes its inner eyelid. After enough pairings of the CS and US, the animal produces a conditioned response (e.g., closing the inner eyelid) after the CS—behaving in advance of the US. This arrangement is interesting because there is a gap, called the trace interval, between the offset of the CS and onset of the US where no stimuli are presented. Empirically we can only reliably measure the strength and timing of the animal's anticipatory behavior: the muscles controlling the inner eyelid. However, the common view is that the rabbit is making a multi-step prediction of the US triggered by the onset of the CS that grows in strength closer to the onset of the US (Schneiderman, 1966; Sutton & Barto, 1990, 2018), similar to the conditioned response in Figure 1.

The mystery for both animal learning and Artificial Intelligence (AI) is how does the agent fill the gap? No stimuli occur during the gap and yet the prediction of the US rises on each time step. There must be some temporal generalization of the stimuli occurring inside the animal. Additionally, what is the form of the prediction being made, and what algorithm is used to update it? Previous work has suggested that the predictions resemble *discounted returns* used in reinforcement learning (Dickinson, 1980; Wagner, 1978), sometimes called nexting predictions (Modayil et al., 2014), which can be learned using temporal difference learning and eligibility traces (i.e., TD(λ)). Indeed the TD-model of classical conditioning has been shown to emulate several phenomena observed in animals (Ludvig et al., 2008, 2012; Sutton & Barto, 1990).

On the question of representation or agent state, the answer is less clear. TD-models can generate predictions consistent with the animal data, but only if the state representation fills the gap between the CS and US in the right way (Ludvig et al., 2009, 2012; Williams et al., 2017). A flag indicating the CS just happened, called the *presence representation*, will not induce predictions that increase over time, and a clock is not plausible given the range of timescales, the presence of other non-relevant distracting signals, and the massive number of predictive relationships an agent must learn in its lifetime [3] (Gallistel & King, 2011). Hand-designed temporal representations do reproduce the animal data well (Ludvig et al., 2008, 2009, 2012; Williams et al., 2017), but their generality remains unclear. Ideally, the learning system could discover for itself how to represent different stimuli over-time in a way that (1) is useful across a variety of prediction tasks, and (2) requires computation and storage independent of the size of the trace interval. Animals do require more training to learn trace conditioning tasks with longer and longer trace intervals, but there is no evidence that the update mechanisms or representations fundamentally change as a function of the trace interval (Howard & Eichenbaum, 2013). Prior work by Rivest et al. has investigated an LSTM driven by temporal-difference errors as a model of cortical and dopaminergic neurons during trace conditioning (Rivest et al., 2014), but Rivest's work did not focus on the impact of problem parameters like the trace interval on learning performance. The Rescorla-Wagner drift-diffusion model provides a reasonable account of trace conditioning (Luzardo, 2018), but does not update predictions during the trial.

Trace conditioning represents a family of diagnostic problems with many potential variations. There could be several additional stimuli which are unrelated to the CS and US, called *distractors*. The CS and US could occur for different lengths of time and overlap in different ways. There can be multiple CSs and the US might only occur for particular ordering and configurations of the CSs. In *patterning* or *biconditional discrimination* experiments, for example, the CSs all occur at the same time step, but only a particular pattern of active and inactive CSs trigger the US (see Harris et al. (2008)). Finally, we can combine these problems in a variety of ways mixing multi-step dependencies, distractors, and patterning. In this paper, we propose three variations as diagnostic benchmark problems for evaluating online multi-step prediction and state construction.

4. From animal learning to online multi-step prediction

We model our multi-step prediction task as an uncontrolled dynamical system. At every time step t, the agent observes stimuli $\mathbf{o}_t \in \mathbb{R}^d$, which includes CS_t and US_t, and makes a prediction $v_t \in \mathbb{R}$ about the future value of the US. The CS at time t may be relevant to the prediction of the US in the future, and the observation $\mathbf{o}_t$ may contain distractors that are unrelated to the US—regardless $\mathbf{o}_t$ does not fully capture the current state of the system. As discussed in Section 3, a suitable choice for formulating the US predictions is the expected discounted return or value function: $v_t \doteq \mathbb{E}[G_t|S_t]$ where

$$G_t \doteq \sum_{k=0}^{\infty} \gamma^k \text{US}_{t+k+1} \tag{1}$$

is called the *return* and S_t is the unobserved state. $\gamma \in [0, 1)$ is called the discount factor and defines the horizon of the prediction of the US.

We will incrementally estimate v_t on each time step with semi-gradient temporal difference (TD) learning (Sutton, 1988). Semi-gradient TD is the most commonly used algorithm for these online prediction tasks and has appealing features relevant to our setting. TD is (1) simple and computationally frugal (linear complexity), and (2) efficient and accurate for learning multi-step predictions online from real data (see Modayil et al. (2014)). Semi-gradient TD learns a parametric approximation $V_t \in \mathbb{R} \approx v_t$ by updating a vector of parameters. $\mathbf{w}_t \in \mathbb{R}^d$ as follows

$$\begin{aligned}\mathbf{w}_{t+1} &\leftarrow \mathbf{w}_t + \alpha(\mathrm{US}_{t+1} + \gamma V_{t+1} - V_t)\mathbf{z}_t \\ \mathbf{z}_t &\leftarrow \gamma\lambda\mathbf{z}_{t-1} + \nabla_{\mathbf{w}} V_t\end{aligned} \tag{2}$$

where $\alpha \in (0, 1]$ is the learning rate and $\lambda \in [0, 1]$ controls the decay of eligibility trace $\mathbf{z}_t \in \mathbb{R}^d$. The precise form of V_t depends on the parameterization scheme. In the linear case, $V_t \doteq \mathbf{x}_t^T \mathbf{w}_t$ and $\nabla_w V_t = \mathbf{x}_t$, where $\mathbf{x}_t \in \mathbb{R}^d$ is a vector of features constructed from $\mathbf{o}_t$. In the non-linear case, V_t can be computed by a neural network and $\nabla_w V_t$ by backpropagation. More generally, US_{t+1} can be any component of $\mathbf{o}_t$ in equations (1) and (2) allowing prediction of any component of the observations (as in Modayil et al. (2014)).

In this paper, we investigate different approaches to constructing $\mathbf{x}_t$. One approach is to simply form an exponentially-weighted decaying memory of each component of $\mathbf{o}_t$, or *stimulating trace*,[4] and then apply a non-linear mapping to produce $\mathbf{x}_t$. Each component of stimulating trace, $\mathbf{y}_t$, corresponds to one component of $\mathbf{o}_t$ and is set to 1 at the onset of the corresponding observation and decays immediately after the observation onset following $y_{t+1} = \tau y_t$ where $0 < \tau < 1$ is the decay parameter. Our tile-coded traces representation applies tile coding[5] to the stimulating traces of $\mathbf{o}_t$. In this case, the quality of $\mathbf{x}_t$ depends on both the tile coding parameters and the exponential decay rate of the stimulating trace. The so-called microstimulus representation, used in prior computational modeling of trace conditioning, is also a fixed feature construction approach dependent on hyper-parameters set by the designer. The microstimulus is formed from a set of overlapping Gaussian basis functions with the heights forming an exponential decay of $\mathbf{o}_t$ achieved by using larger standard deviations for each gaussian (Ludvig et al., 2008, 2012; Hull, 1939). Figure 2 shows an example of the stimulating trace of the CS and how the representation constructed by tile-coded traces (1 tiling 8 tiles) and microstimulus (8 Guassians) for the CS unfold over time.

Alternatively, $\mathbf{x}_t$ can be constructed recursively from $\mathbf{o}_t$ and $\mathbf{x}_{t-1}$ using a non-linear state update function $\mathbf{x}_t \doteq u(\mathbf{x}_{t-1}, \mathbf{o}_t)$. See Figure 3. The tile-coded traces and microstimulus representations represent particular instantiations of u that never change during learning. We can also think of u constructing $\mathbf{x}_t$ as a recurrent neural network. We consider both the case where u is fixed at the beginning of learning, also known as echo state networks (Jaeger, 2001), as well as the case where T-BPTT or RTRL changes u on each time step. In the case of echo state network, there are three groups of incoming weights to the hidden layer: (1) the input weights from the input to the hidden layer (2) the internal weights from the hidden layer to itself and (3) the feedback weights from the output layer to the hidden layer. All the incoming weights to the hidden layer are fixed at the beginning of learning and only the weights from the features to the output are learned. In contrast, in the case of learning with T-BPTT and RTRL not only the agent's predictions of v_t are updated, but also the function u_t is learned.

Using T-BPTT and RTRL to train RNNs and their variants in an online setting is not new, nor is the application of such architectures to multi-step TD prediction targets. We followed standard practice in implementing these methods. For T-BPTT with truncation length T, when making an update at time t, we unroll the RNN for T steps. We set the initial hidden state to $\mathbf{x}_{t-T-1}$. Then we compute the hidden states and the value predictions along the observation sequence $\mathbf{o}_{t-T}, \ldots, \mathbf{o}_{t-1}$. After computing the value predictions V_{t-T} to V_{t-1}, we use them as a mini-batch to update the parameters of the network using backpropagation.

For RTRL, on the other hand, we update the parameters throughout the training sequence on every time step, while still carrying forward a stale Jacobian that tracks sensitivity to the old parameters (See Menick et al. (2020)).

5. Trace conditioning: Learning to fill the gap

Our first diagnostic problem, *trace conditioning*, is inspired by classical conditioning experiments described in Section 3. The problem is made up of a series of trials in each of which a sequence of stimuli are presented: the CS followed by the US. On each trial, the CS lasts for 4 time steps, and is followed by a long gap and then the US which lasts for 2 time steps. The time from the CS onset to the US onset is called the *inter-stimulus interval* (ISI). In this problem, the ISI is drawn from a uniform distribution. The time from the US onset to the start of the next trial is called the *inter-trial interval* (ITI). The ITI is uniformly sampled from (80, 120). γ is set according to the ISI: $\gamma = 1 - 1/\mathbb{E}(\mathrm{ISI})$. This allows the time horizon of the return to match the ISI. Figure 4 provides an example trial including the CS, US, and return for a case where $\mathrm{ISI} \sim \mathrm{Unif}(7, 13)$.

We also include several binary distractor stimuli that do not contain any information about the US. The distractors are drawn from a Poisson distribution with different frequencies and each lasts for 4 time steps. The frequency varies from distractor to distractor. One distractor occurs on average every 10 steps, another every 20 steps, and so on, up to one distractor that occurs every 100 steps on average. Note that they also occur during the ITI.

To understand why this problem could be challenging for a learning system, consider learning to predict using the presence representation. This representation contains one binary feature per stimulus which is activated only when the corresponding stimulus is present. The presence feature corresponding to the CS is active during the CS activation as shown in Figure 2. However, during the trace interval, between the offset of the CS and the onset of the US, no feature is active (only the bias feature, which has a small weight associated with it is active) and therefore, the trace

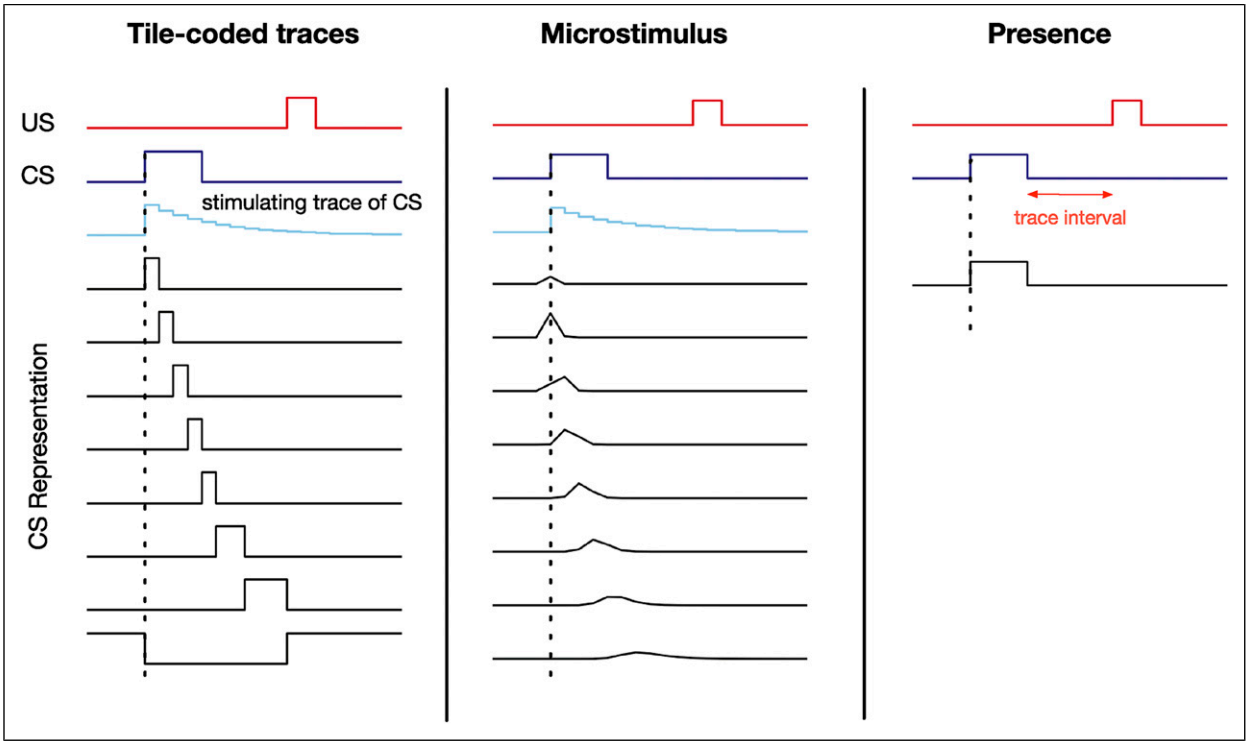


Figure 2. The stimulus representation for the tile-coded traces, microstimulus, and presence representations. The presence representation does not have any active features during the trace interval. This figure is adapted from Ludvig et al. (2012).

interval is not represented by the presence representation. As a result, as shown in Figure 4, the presence representation has a close to zero prediction during the trace interval.

To understand what a good prediction looks like, consider the microstimulus and tile-coded traces representations. During the empty gap between the CS offset and the US onset, the microstimulus and tile-coded traces representations have active features constructed from a trace of the CS (see Figure 3). As a result, they successfully associate the CS with the US (see the predictions for the microstimulus representation in Figure 4). Note that the return reaches its maximum just before the US onset and steps downward after. This happens because the discounted sum of future USs is maximal just before the US onset: at this instant in time the US is multiplied by the largest possible values of the discount factor, γ. This temporal profile is consistent with previous work on *Nexting* (Modayil et al., 2014) and computational modeling in animal learning (Ludvig et al., 2012).

Note that the prediction increases only after the CS onset whereas the return has non-zero values before the CS onset. This makes sense because there is a significant time between each trial and thus the onset of the CS is unpredictable by design—just like in trace conditioning experiments with animals.

In the trace conditioning benchmark, we experimented with two groups of representations as baselines. The first

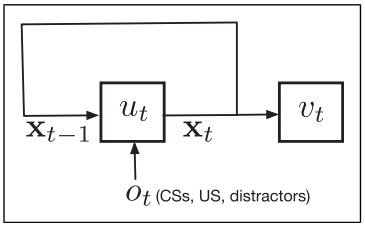


Figure 3. Recursive construction of $\mathbf{x}_t$ from o_t and $\mathbf{x}_{t-1}$ where o_t includes the CSs, the US, and the distractors, if any.

group includes fixed representations: microstimulus, tile-coded traces, and echo state network (See Section 4 for the explanation about these representations). Microstimulus and tile-coded traces are expert-designed representations and include a bias feature that is always 1. We adjusted the stimulating trace decay parameter for microstimulus and tile-coded traces according to the ISI: $1/\mathbb{E}(\text{ISI})$. For echo state networks, all the three sets of weights contributing to constructing the hidden state were initialized and fixed at the beginning of learning. The input weights and feedback weights were initialized using a binomial distribution and scaled by an input scaling parameter. The internal weights were initialized in such a way that the spectral radius of the corresponding matrix is less than 1 and its density is small.

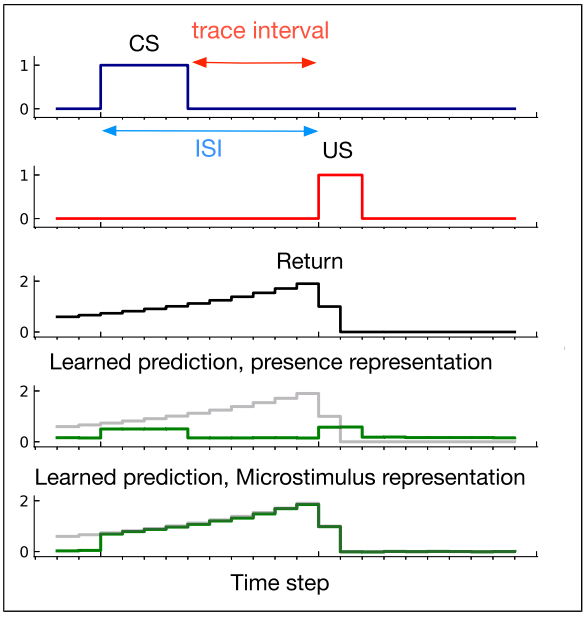


Figure 4. An example of learned predictions in *trace conditioning*. The *return* defined in equation (1) is the target of prediction. Rows 4 and 5 show predictions using the presence and microstimulus representations after 200,000 time steps learning. Microstimulus successfully predicted the US—matching the return—the presence representation failed to predict the US. The predictions never go to zero like the return because all representations use a bias feature and even after 200,000 steps the predictions continue to update.

The second group of representations includes those learned by recurrent neural networks: Vanilla-RNN, LSTM, and GRU. We used Haiku library for implementing the Vanilla-RNN, LSTM, and GRU architectures. We evaluated both T-BPTT and RTRL for computing the gradient of the value function with respect to the network's weights.

For each of these representations, we used semi-gradient TD(λ) and ADAM optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$ (Kingma & Ba, 2014). For the fixed representations, we set the eligibility traces parameter, λ, to 0.9. For the RNNs, we used $\lambda = 0$.

To evaluate the performance, we computed the Squared Return Error (SRE): $(\widehat{v}(S_t, w_t) - G_t)^2$. We then averaged the SRE over all time steps resulting in a Mean Squared Return Error (MSRE). We studied the effect of the ISI on the performance of the baseline representation methods considering three cases: (1) short: ISI $\sim$ Unif$(7, 13)$, (2) medium: ISI $\sim$ Unif$(14, 26)$, and (3) long: ISI $\sim$ Unif$(20, 40)$, with expected ISI equal to 10, 20, 30 for the 3 settings, respectively.

We swept over the parameters of each representation method. See Table 1. The parameter sweeps included the step-size for all the methods, the number of Tile/RBFs for tile-coded-traces/microstimulus, the hidden layer size for the RNNs and echo state network, and the spectral radius, input scaling, and internal connections density for the echo state network. For tile-coded traces, we used 2 tilings and for microstimulus, we set the standard deviation of the RBFs to 0.8. For RNNs trained with T-BPTT, we swept over T-

Table 1. Parameter sweeps for the three benchmarks.

Problem	Representation Method	Number of Tiles/RBFs	Hidden layer size	Spectral Radius	Input scaling	W h density	Truncation Length	Step-size
Trace Conditioning and Trace Patterning	Presence	—	—	—	—	—	—	3×10^{-6}, 10^{-5} 3×10^{-5}, 10^{-4} 3×10^{-4}, 10^{-3}
	Microstimulus	4, 8, 16, 32	—	—	—	—	—	
	Tile-coded-traces	2, 4, 8, 16	—	—	—	—	—	
	Vanilla-RNN	—	10, 20, 40	—	—	—	5, 10, 20, 40	
	GRU	—	10, 20, 40	—	—	—	5, 10, 20, 40	
	LSTM	—	10, 20, 40	—	—	—	5, 10, 20, 40	
	ESN	—	100, 1000	0.9, 0.99, 0.999	0.1, 0.5	0.05, 0.1	—	
Noisy Patterning	Presence	—	—	—	—	—	—	
	Vanilla-RNN	—	10, 20, 40	—	—	—	5	
	GRU	—	10, 20, 40	—	—	—	5	
	LSTM	—	10, 20, 40	—	—	—	5	
	ESN	—	100, 1000	0.9, 0.99, 0.999	0.1, 0.5	0.05, 0.1	—	

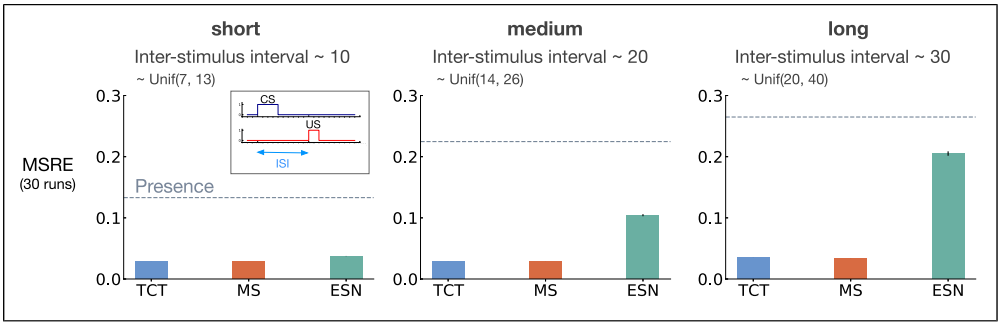


Figure 5. The interaction between ISI and truncation level in *trace conditioning* for fixed representations: tile-coded traces (TCT), microstimulus (MS), and echo state network (ESN). Each subplot corresponds to one setting of short, medium, and long ISI. A mini picture of the CS and US timings is included in the leftmost subplot. The y-axis is the MSRE. Lower is better. The results are calculated over 2 million steps and averaged over 30 runs. (Standard error bars are plotted but in some cases are not visible due to being small). The error level for the presence representation is plotted in each subplot as a dotted line for comparison. In the short setting, all methods performed well. Microstimulus and tile-coded traces performed well across all settings. The performance of echo state network, however, deteriorated as ISI got larger.

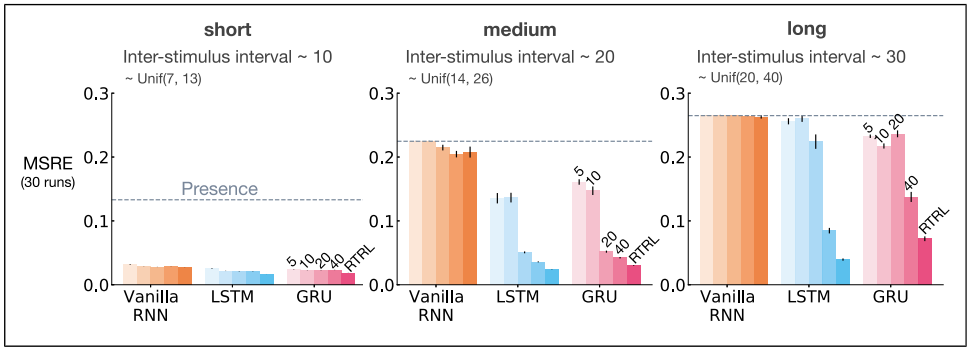


Figure 6. The interaction between ISI and truncation level in *trace conditioning* for representations learned by T-BPTT and RTRL. Each subplot corresponds to one setting of ISI. In each subplot, multiple bars are plotted for Vanilla RNN, LSTM, and GRU. For each architecture, the left four bars correspond to T-BPTT with different truncation levels and the right bar corresponds to RTRL. The y-axis is the MSRE with lower better. The results are calculated over 2 million steps and averaged over 30 runs. Standard error bars are included in the plot. With short ISI all methods performed well and the T-BPTT based methods worked with all T's. In the medium setting, we see basic RNNs performed poorly, and LSTMs and GRUs required truncation at or greater than expected ISI (20) to perform well. In the long setting, we see that none of the T-BPTT based methods performed well, even with T greater than expected ISI. Across all three problem settings, RTRL-based LSTMs learned accurate predictions.

BPTT truncation length. For all RNNs, the number of hidden layers was set to one.

We ran each method with each of its parameter settings for 5 runs and 2 million time steps. We then computed MSRE averaged over the 5 runs and selected the parameter setting that resulted in the lowest level of MSRE. After optimizing the parameters, we ran each method with its best parameter setting for 30 runs and averaged the result. We calculated standard errors for each method to measure how far the sample means are from the true population means. We then plotted the MSRE averaged over 30 runs and standard error bars with non overlapping standard error bars for two methods suggesting significant difference in their performance.

Figure 5 shows MSRE for fixed representations for short, medium, and long ISI. The y-axis is MSRE averaged over 30 runs. The level of error for the presence representation is shown with a dotted gray line for comparison.

The expert designed fixed representations of microstimulus and tile-coded traces performed well across all ISI settings; however, echo state network failed to capture longer temporal dependencies. In the short setting, all fixed representations performed well. As ISI got larger, echo state network performed worse and approached the level of error of the presence representation. This is likely due to the fact that echo state networks trade-off prediction accuracy for computation.

Figure 6 shows MSRE for representations learned by T-BPTT and RTRL for short, medium, and long ISI. In each subplot, multiple bars are shown for each of Vanilla RNN,

LSTM, and GRU architectures. For each architecture, the four left bars correspond to T-BPTT with $T = 5$, $T = 10$, $T = 20$, and $T = 40$. The right bar corresponds to the result for RTRL.

In the short setting, the representations learned by both T-BPTT and RTRL performed well for all architectures, reaching a much lower level of error compared to the presence representation.

RNNs trained with T-BPTT were sensitive to the length of the truncation window, and the sensitivity became more pronounced as ISI got larger (Figure 6). To better understand this, let us contrast the performance of T-BPTT with that of the RTRL variants, which are roughly equivalent to T-BPTT for $T = \infty$ (since when $T = \infty$, T-BPTT computes the gradient all the way back in time, resulting in a gradient roughly the same as the one computed by RTRL). In the medium setting, the T-BPTT variants for LSTMs and GRUs performed similarly to the RTRL counterparts only when the truncation window was greater than or equal to 20—the expected ISI (Figure 6, middle column). This effect was even stronger in the long setting (Figure 6, right column). This result is one example of the efficacy of trace conditioning as a diagnostic benchmark—it clearly isolates the trade-off introduced by the T-BPTT algorithm.

There was a significant drop in the performance of Vanilla RNNs as we increased the expected ISI and larger truncation window did not help improve performance much. This is likely due to the vanishing gradient problem (Hochreiter et al., 2001). Vanilla RNN trained with RTRL also failed to capture longer dependencies. This is in contrast to the LSTM and GRU variants trained with RTRL.

Our results suggest that further algorithmic improvements are required for solving the trace conditioning problem. While the expert designed fixed representations perform robustly across all ISI settings, they do not automatically discover useful features, and thus are not scalable. RTRL also performs well in all cases; however, it is not computationally feasible. Finally, T-BPTT's performance is highly sensitive to the truncation parameter, requiring much more computation for learning longer temporal dependencies. Later we will discuss a simple algorithm that we tried to improve performance.

6. Noisy patterning

The trace conditioning benchmark is an idealization because there is only one signal of interest: the CS. The agent need not figure out which parts of its input stream to focus on—it is purely a temporal memory problem. Our second diagnostic benchmark, *noisy patterning*, does not make this assumption. In noisy patterning, the agent must predict a binary outcome which only occurs if a particular pattern of stimuli is presented (Mackintosh, 1974; Harris et al., 2008). To do so, it has to both figure out which parts of the input to pay attention to in the presence of noise and distractors and also make nonlinear features to identify the patterns of interest. This is similar to the "Blooming, Buzzing Confusion" visual stream that infants experience—they must learn what to pay attention to and ignore (James, 1890). In robot terms, the equivalent would be which sensors the agent should pay attention to, to avoid damage or gain additional reward. Similar problems have been studied in supervised learning (Sutton, 1992; Sutton & Whitehead, 1993; Mahmood & Sutton, 2013).

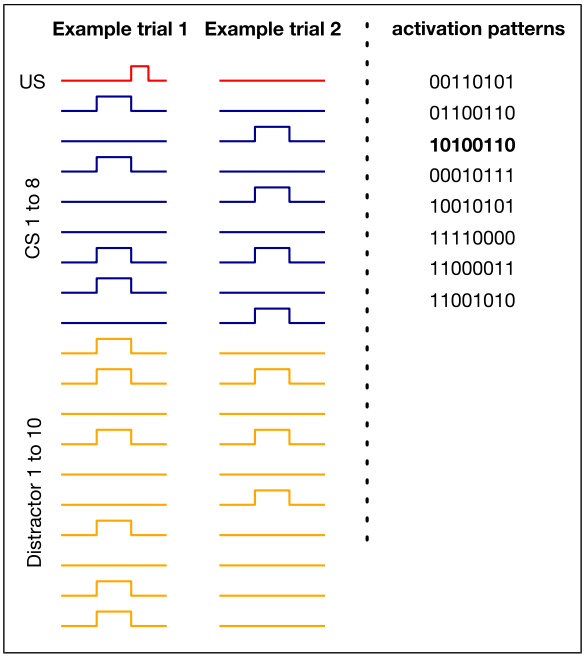


Figure 7. Example trials for noisy patterning in the case of 8 CSs, 8 activation patterns, 10 distractors, and 10 percent noise. 10100110 is one of the 8 activation patterns. In the example trial on the left, the pattern of the CSs matches this pattern and the US gets activated as a result. In the example trial on the right, however, the pattern of the CSs does not match any of the activation patterns resulting in US remaining 0.

Noisy patterning is analogous to positive/negative patterning in psychology. It considers a situation where nonlinear combinations of CSs activate the US. As we discussed in Section 3, in negative patterning each CS in isolation activates the US but their combination does not. Interestingly, these problems correspond to famous logical operations like XOR, which are famously unsolvable by single-layer neural networks. While neural networks with more than one layer can easily learn patterning problems like XOR, some of the approaches considered in this paper, such as microstimulus, fail to solve them. To make the benchmark more challenging, we designed the benchmark such that multiple configurations of the CSs activate the US and added distractors and noise.

This benchmark includes n CSs and one US. There are k configurations of the CSs that activate the US. We refer to

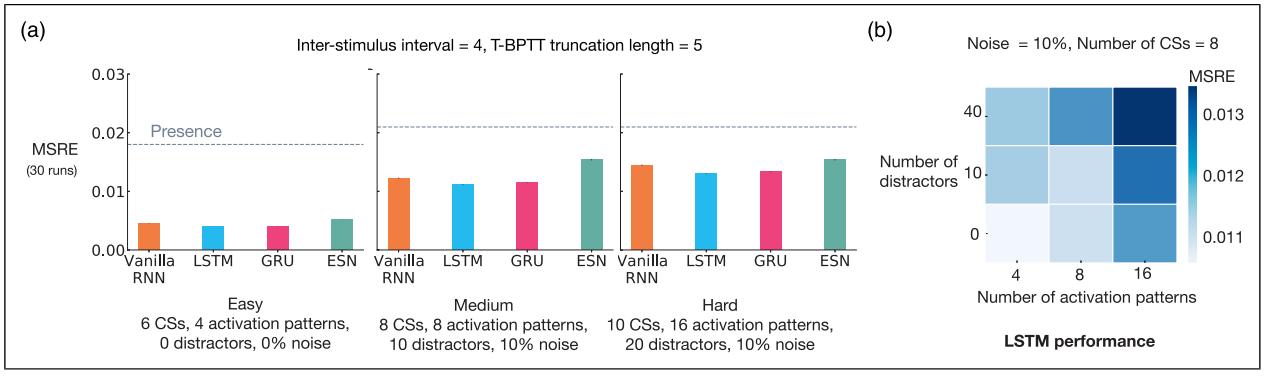


Figure 8. *Noisy patterning* with varying difficulty levels. The 3 bar plots (a) show the MSRE of Vanilla-RNN, GRU, and LSTM trained with T-BPTT as well as the MSRE of echo state network for three different configurations of the problem: easy, medium, and hard. The results are for 2 million steps of training and averaged over 30 runs. The standard error bars are included. In the leftmost plot, we see a consistent drop in performance, across all methods, from the easy setting to the hard one. The heat map on the right (b), illustrates that the performance of LSTM degraded as the the number of distractors and activation patterns increased.

these configurations as activation patterns. Each trial starts with the CSs getting a value of 0 or 1. If the value of the CSs matches an activation pattern, the US becomes 1 in 4 time steps (i.e., ISI equals 4). (In contrast to trace conditioning, the ISI is fixed.) The ITI is uniformly sampled from (80, 120). We designed the benchmark such that in half of the trials, one of the activation patterns occurs each of which includes $n/2$ activated CSs and $n/2$ non-activated CSs. The benchmark also includes m distractors, which occur at the same time as the CSs but do not contribute to the US activation. We also add noise such that in x percent of the trials, an activation pattern occurs but the US remains 0 or a non-activating pattern occurs and the US gets activated. γ is set to $1 - 1/\text{ISI} = 0.75$. Two example trials for a case with 8 CSs, 8 activation patterns, 10 distractors, and 10 percent noise are shown in Figure 7. In the example on the left, the pattern of the CSs matches one of the 8 activation patterns. Therefore, the US gets activated. In the example on the left, however, the pattern of the CSs does not match any of the activation patterns. As a result, the US remains 0.

Just as we can control the difficulty level of trace conditioning by changing, for example, the ISI, we can also control the difficulty level of noisy patterning by changing the key problem parameters—the number of CSs, the number of activation patterns, the number of distractors, and the level of noise. Using this flexibility, we experimented with noisy patterning in two ways. First, we evaluated echo state network and several T-BPTT variants with truncation length 5 on three different levels of difficulty that we refer to as easy, medium, and hard.

We did not experiment with RTRL because with small ISI (= 4), T-BPTT with $T = 5$ performs as well as the idealized RTRL baseline. We also did not experiment with tile-coded traces and microstimulus because they independently represent each input and cannot predict patterns of CSs as they are combined with linear function approximation.

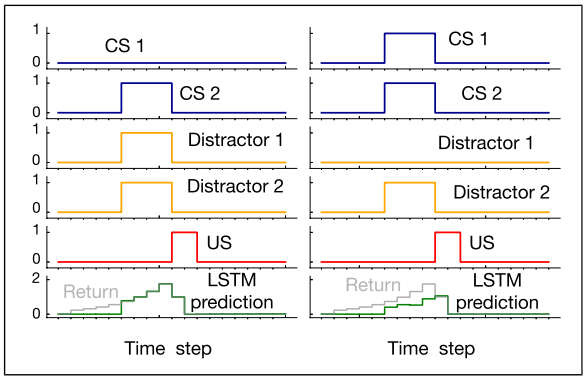


Figure 9. Example prediction profile plots for noisy patterning in the medium setting and hard setting. Unlike Figure 7 where all the CSs and distractors were shown, in this figure only two of the CSs and distractors are shown as examples. In both cases, an activation pattern occurred as a result of which the US got activated. In the the medium setting, LSTM prediction matched the return. In the hard setting, however, LSTM did not predict the US accurately.

There was a consistent drop in performance, across all methods, as the level of difficulty was increased (Figure 8(a)). Echo state network performed worse than all three recurrent variants trained with T-BPTT in all three configurations of the problem. This is likely due to the fact that echo state network's representation, which is randomly determined and fixed at the beginning of learning, is not suitable for capturing the activation patterns.

Example prediction profile plots for noisy patterning are provided in Figure 9 for the medium and hard levels of difficulty. We are only showing 2 of the CSs and 2 of the distractors as examples. In both examples, an activation pattern occurred and the US got activated (i.e., the US activation was not due to noise). In the medium setting, LSTM successfully predicted the US, matching the return after the

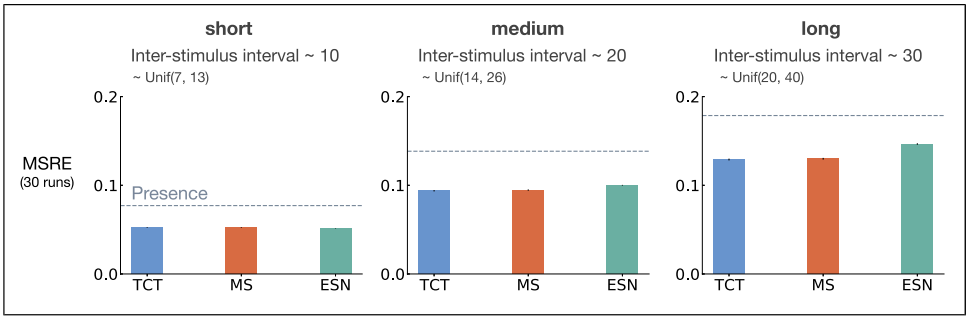


Figure 10. The impact of truncation level in *trace patterning* for fixed representations. We used the exact same scheme as Figure 5 to visualize the performance in trace patterning. Each plot corresponds to one setting of short, medium, and long ISI. Each bar reports the MSRE averaged over 30 runs. All methods were trained for 5 million steps. All fixed representations performed poorly. Tile-coded traces and microstimulus independently represent each input (not combinations) and thus cannot learn accurate predictions.

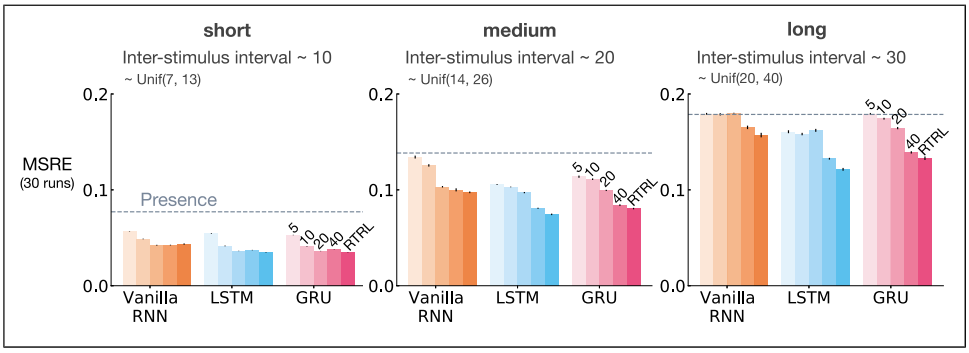


Figure 11. The impact of truncation level in *trace patterning* for representations learned by T-BPTT and RTRL. Each subplot corresponds to one setting of short, medium, and long ISI and includes the error for Vanilla-RNN, LSTM, and GRU. For each architecture, multiple bars are shown with the left four bars corresponding to T-BPTT with different T's and the right bar corresponding to RTRL. The results are calculated over 5 million steps and averaged over 30 runs. Similar to trace conditioning, the T-BPTT based methods showed sensitivity to the truncation parameter. The use of RTRL always improved performance; however, except for ISI ∼10 no methods learned accurate predictions.

onset of the CS. However, in the hard setting, there was a mismatch between LSTM's prediction and the return.

To further highlight the configurability of noisy patterning, we evaluated the T-BPTT variant of LSTM across two dimensions: the number of activation patterns and the number of distractors. The results, presented as a heatmap of MSRE in Figure 8(b), show that the performance deteriorated as we made the problem more difficult across either dimension.

Taken together, these results demonstrate that noisy patterning can be useful for systematically studying the scaling properties of the algorithms in isolation from the temporal dimension, by simply increasing the number of signals from half a dozen to tens of thousands.

7. Trace patterning: Putting it all together

We put together the challenge of bridging the temporal gap, as posed by trace conditioning, and the challenge of recognizing important patterns, as formulated in noisy patterning, in a unified diagnostic problem that we refer to as *trace patterning*. For a learner to do well on this problem, it has to both fill the trace interval and construct non-linear representations of the CSs.

Similar to the results presented in Section 5, we evaluated the baseline methods as we increased the ISI while keeping the rest of the problem parameters constant (8 CSs, 8 activation patterns, 10 distractors, and 10% noise). The results for fixed representations and representations learned by T-BPTT and RTRL are provided in Figures 10 and 11, respectively.

The fixed representations performed poorly in all cases of short, medium, and long ISI and their performance got worse as ISI got larger (Figure 10). The expert designed fixed representations of microstimulus and tile-coded traces independently represent each input (and not their combinations) and thus cannot learn accurate predictions; contextualizing the failure of the echo state network in this problem.

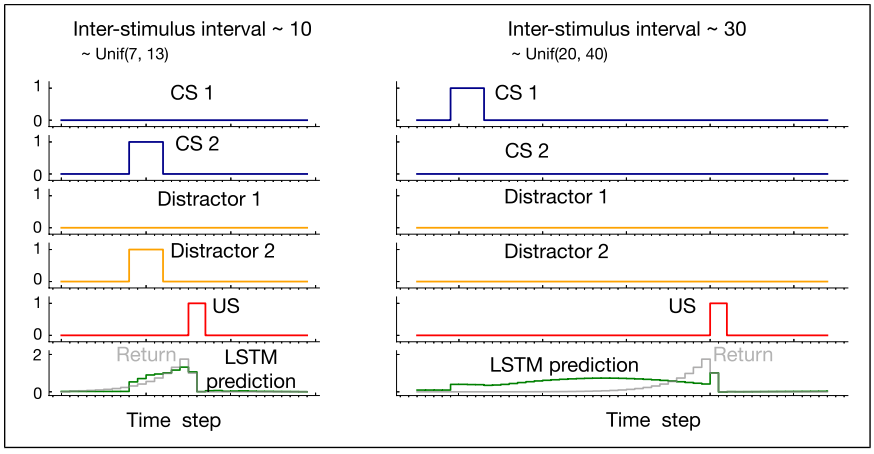


Figure 12. Example prediction profile plots for LSTM in *trace patterning* in the the case of expected ISI 10 and 30. LSTM was trained with T-BPTT and a truncation length of 40. Only two of the CS and distractors are shown as examples. In both cases, an activation pattern occurred as a result of which the US got activated. In the the case of expected ISI of 10, LSTM prediction resembled the return. In the case of longer ISI with expectation of 30, however, LSTM did not predict the US accurately.

The T-BPTT algorithms showed sensitivity to the length of the truncation window (Figure 11). This is consistent with the findings from the trace conditioning experiments. One key difference, however, is that longer truncation parameter for the LSTM and GRU variants did not help as much as in trace conditioning. Moreover, in contrast to trace conditioning, the performance of the idealized RTRL baselines for the LSTM and GRU variants got worse considerably as we increased the ISI.

Example prediction plots for LSTM trained with T-BPTT are shown in Figure 12 in the case of expected ISI of 10 and 30. In both cases, a truncation length of 40 was used. While LSTM prediction profiles resemble the return in the case of expected ISI of 10, they fail to match the return in the case of expected ISI of 30.

This result emphasizes the difficulty of trace patterning—the tested recurrent networks struggle to achieve low error, even when they have access to better gradient approximations, as in the case of training with RTRL.

8. Combining stimulating traces with RNNs

Our experimental results highlight the limitations of the current learning methods. While the linear trace-based methods successfully bridge the temporal gap in trace conditioning, their performance deteriorates when we introduce nonlinearities in trace patterning. On the other hand, the recurrent learning algorithms can simultaneously bridge the temporal gap and handle nonlinearities, but they can be expensive in computational and memory requirements

In the case of T-BPTT, the memory requirements of RNNs grow with the length of the truncation window, and learning long-term dependencies, as in trace conditioning, requires a comparably long truncation window. In the case of RTRL, the computational complexity of RNNs grows quartically in the size of the hidden state, and learning patterns from a large number of signals, as in noisy patterning, requires a large hidden state. Ideally, we need training methods that scale well in computation and memory simultaneously.

As an example, we present a simple approach that scales well in computation and memory. We augment the RNNs with the stimulating memory traces of the observation. In particular, we feed an exponentially decaying trace of each stimulus, as described in Section 4, as part of the input observation to the recurrent network.

Figures 13 and 14 show the effect of augmenting the RNNs with the stimulating memory traces of the observation, respectively, in trace conditioning and trace patterning. The results for RNNs fed with only the observation is also included in lighter shades for comparison., as does the general conclusion that stimulating traces improve performance but less so than in trace conditioning.

When trained with T-BPTT, feeding the RNNs with the stimulating traces significantly improved the performance for the Vanilla RNN, LSTM, and GRU variants in trace conditioning. Moreover, it made the T-BPTT variants robust to the truncation length, achieving a similar level of error for all T's. This effect was more pronounced in the long setting (Figure 13).

When trained with RTRL, feeding the RNNs with the stimulating traces helped improve the performance (Figure 13). The improvement was larger for Vanilla RNN than the LSTM and GRU variants.

In trace patterning, also feeding the RNNs with the stimulating traces improved performance in both T-BPTT and RTRL variants but less so than in trace conditioning.

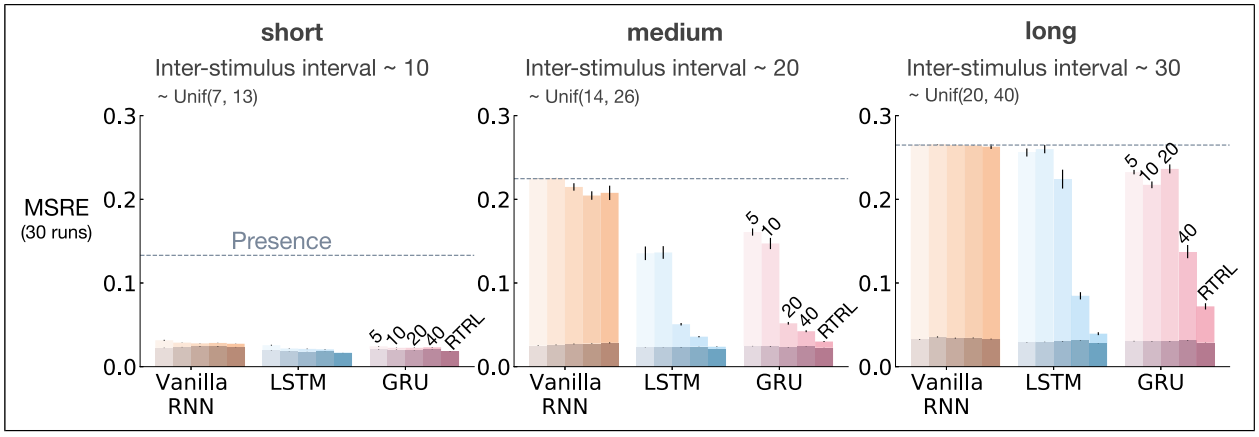


Figure 13. Results for combining stimulating *traces* with RNNs in *trace conditioning*. We used the exact same scheme as Figure 6. Darker colors denote the combination of stimulating traces with the recurrent methods and lighter shades denote the recurrent methods. Each bar reports the MSRE averaged over 30 runs. The methods were trained for 2 million steps. The error bars denote the standard errors. Adding stimulating traces to the input of the Vanilla-RNN, GRU, and LSTM improved their performance in both T-BPTT and RTRL cases and made them less sensitive to the truncation length in the case of training with T-BPTT.

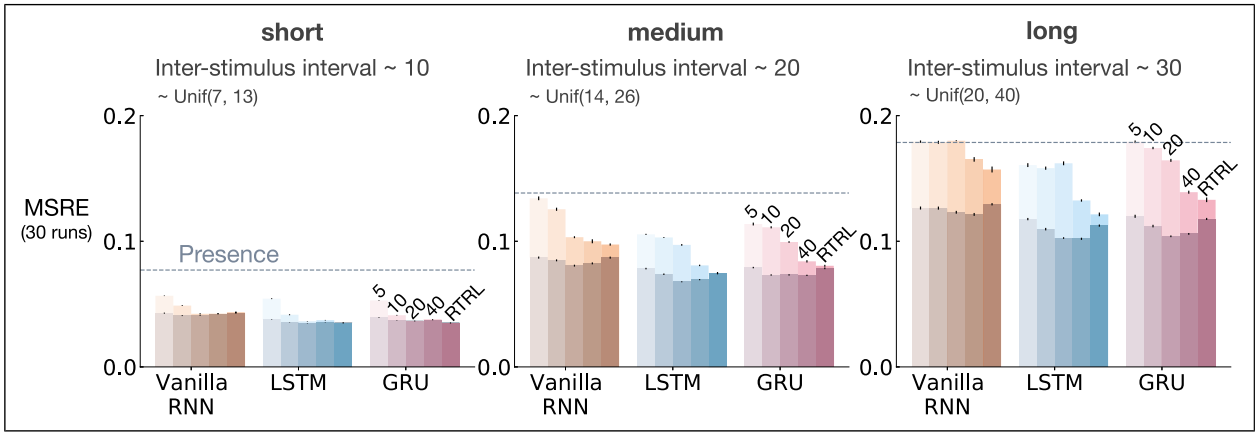


Figure 14. Results for combining stimulating traces with RNNs in *trace patterning*. The naming conventions exactly match Figure 13, as does the general conclusion that stimulating traces improve performance but less so than in trace conditioning.

While the space of ideas for fruitfully combining memory traces and RNNs needs further investigation, this result shows how the proposed diagnostic benchmarks can help us search for general and scalable ideas for the online prediction problem.

9. Discussion

Our diagnostic benchmarks can be used to isolate and investigate fundamental algorithmic issues in recurrent learning. In trace conditioning, we found that basic recurrent architectures could not handle significant temporal dependencies. Gated architectures exhibited significant sensitivity to truncation level (needing to unroll beyond the onset of the relevant cue) but did not perform as well as RTRL variants. In our trace patterning experiments, all methods struggled when confronted with the combination of long temporal dependencies and the need to extract configuration patterns.

In this paper, we investigated the online prediction setting, but more stringent computational restrictions might be useful for future work. Many RL algorithms, like TD, can make and update long-horizon predictions with computation significantly less than the length of the prediction's horizon (van Hasselt & Sutton, 2015). This might also be possible in representation learning. Can the agent construct representations capable of overcoming dependencies back in time with computation and storage less than the length of the gap? While recurrent learning algorithms based solely on T-BPTT do not meet this requirement, our results show that some combination of stimulating traces and recurrent architectures may reduce the agent's dependency on the

truncation level. Moreover, there is a discrepancy between the speed of learning for natural and artificial systems; while animals learn eyeblink conditioning in about a few hundred trials, our baseline methods require thousands of trials to learn the task. Future research should investigate reasonable computational restrictions if we hope to discover representations as efficient as those used by animals. Work on more efficient update rules (Nath et al., 2019) and attention mechanisms (Dehghani et al., 2019) represent promising directions toward this ambitious goal.

Acknowledgements

We would like to thank Martha White for many helpful discussions. We also thank Patrick Pilarski, Doina Precup, and Khurram Javed for providing helpful comments for improving the paper.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: The work was supported by Google DeepMind, the Alberta Machine Intelligence Institute, Natural Sciences and Engineering Research Council of Canada (NSERC), Canadian Institute for Advanced Research (CIFAR), Alberta Innovates—Technology Futures (AITF), NSERC discovery program, and CIFAR Canada AI Chair program.

ORCID iD

Banafsheh Rafiee https://orcid.org/0000-0003-4641-7349

Notes

1. The source code for our three benchmark problems is available here.
2. Humans do not appear to perform more computations to remember further back in time, rather people appear to employ abstractions that lose precision the further back they remember.
3. Ludvig's microstimulus representation can be viewed as a clock whose resolution gets worse over time (Ludvig et al., 2012).
4. A stimulating trace of the observation is different from the eligibility trace vector z. z is part of the update mechanism and does not impact the representational capacity of x. Mozer was the first to investigate stimulating traces as input to neural network representation learning (Mozer, 1989).
5. See (Sutton & Barto, 2018) for an in-depth treatment of tile coding.

References

Baird, L. (1995). Residual algorithms: Reinforcement learning with function approximation. In Machine learning proceedings 1995, Tahoe City, California, 9–12 July 1995, pp. 30–37, Elsevier.

Beattie, C., Leibo, J. Z., Teplyashin, D., Ward, T., Wainwright, M., Küttler, H., Lefrancq, A., Green, S., Valdés, V., Sadik, A., Schrittwieser, J., Anderson, K., York, S., Cant, M., Cain, A., Bolton, A., Gaffney, S., King, H., Hassabis, D., & Petersen, S. (2016). Deepmind lab. arXiv preprint arXiv:1612.03801.

Bellemare, M. G., Naddaf, Y., Veness, J., & Bowling, M. (2013). The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, *47*, 253–279.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). Openai gym. arXiv preprint arXiv:1606.01540.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., & Mordatch, I. (2021). Decision transformer: Reinforcement learning via sequence modeling. arXiv preprint arXiv:2106.01345.

Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259.

Colas, C., Sigaud, O., & Oudeyer, P. (2018). How many random seeds? statistical power analysis in deep reinforcement learning experiments. arXiv preprint arXiv:1806.08295.

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., & Kaiser, L. (2019). Universal transformers. In International conference on learning representations.

Dickinson, A. (1980). *Contemporary animal learning theory* (Vol. 1). CUP Archive.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*(2), 179–211. https://doi.org/10.1016/0364-0213(90)90002-E

Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., & Madry, A. (2019). Implementation matters in deep rl: A case study on ppo and trpo. In International conference on learning representations.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., & Kavukcuoglu, K. (2018). IMPALA: Scalable distributed deep-RL with importance weighted actor-learner architectures. In J. Dy & A. Krause (Eds.), Proceedings of the 35th International conference on machine learning, proceedings of machine learning research, Stockholmsmässan, Stockholm Sweden, 10-15 July 2018, (Vol. 80, pp. 1407–1416). PMLR.

Fortunato, M., Tan, M., Faulkner, R., Hansen, S., Badia, A. P., Buttimore, G., Deck, C., Leibo, J. Z., & Blundell, C. (2019). Generalization of reinforcement learners with working and episodic memory. In Advances in neural information processing systems (pp. 12469–12478).

Gallistel, C. R. & King, A. P. (2011). *Memory and the computational brain: Why cognitive science will transform neuroscience* (Vol. 6). John Wiley & Sons.

Gehring, J., Auli, M., Grangier, D., Yarats, D., & Dauphin, Y. N. (2017). Convolutional sequence to sequence learning. In International conference on machine learning (pp. 1243–1252). PMLR.

Harris, J. A., Livesey, E. J., Gharaei, S., & Westbrook, R. F. (2008). Negative patterning is easier than a biconditional

discrimination. *Journal of Experimental Psychology: Animal Behavior Processes*, *34*(4), 494–500. https://doi.org/10.1037/0097-7403.34.4.494

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., & Meger, D. (2018). Deep reinforcement learning that matters. In Proceedings of the AAAI conference on artificial intelligence (Vol. 32).

Hochreiter, S., Bengio, Y., Frasconi, P., & Schmidhuber, J. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In A field guide to dynamical recurrent neural networks.

Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, *79*(8), 2554–2558. https://doi.org/10.1073/pnas.79.8.2554

Howard, M. W. & Eichenbaum, H. (2013). The hippocampus, time, and memory across scales. *Journal of Experimental Psychology: General*, *142*(4), 1211–1230. https://doi.org/10.1037/a0033621

Hull, C. L. (1939). The problem of stimulus equivalence in behavior theory. *Psychological Review*, *46*(1), 9–30. https://doi.org/10.1037/h0054032

Jacobsen, A., Schlegel, M., Linke, C., Degris, T., White, A., & White, M. (2019). Meta-descent for online, continual prediction. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(1), 3943–3950.

Jaderberg, M., Czarnecki, W. M., Osindero, S., Vinyals, O., Graves, A., Silver, D., & Kavukcuoglu, K. (2017). Decoupled neural interfaces using synthetic gradients. In International conference on machine learning (pp. 1627–1635). PMLR.

Jaeger, H. (2001). The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. Bonn, Germany: German National Research Center for Information Technology GMD Technical Report 148(34): 13.

James, W. (1890). *The Principles of psychology* (Vol. 1). Henry Holt and Company.

Janner, M., Li, Q., & Levine, S. (2021). Reinforcement learning as one big sequence modeling problem. arXiv preprint arXiv:2106.02039.

Kingma, D. P. & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Loynd, R., Fernandez, R., Celikyilmaz, A., Swaminathan, A., & Hausknecht, M. (2020). Working memory graphs. In International conference on machine learning (pp. 6404–6414). PMLR.

Ludvig, E. A., Sutton, R. S., & Kehoe, E. J. (2008). Stimulus representation and the timing of reward-prediction errors in models of the dopamine system. *Neural Computation*, *20*(12), 3034–3054. https://doi.org/10.1162/neco.2008.11-07-654

Ludvig, E. A., Sutton, R. S., & Kehoe, E. J. (2012). Evaluating the td model of classical conditioning. *Learning & Behavior*, *40*(3), 305–319. https://doi.org/10.3758/s13420-012-0082-6

Ludvig, E. A., Sutton, R. S., Verbeek, E., & Kehoe, E. J. (2009). A computational model of hippocampal function in trace conditioning. In Advances in neural information processing systems (pp. 993–1000).

Luzardo, A. (2018). *The Rescorla-Wagner drift-diffusion model*. PhD Thesis, City, University of London.

Machado, M. C., Bellemare, M. G., Talvitie, E., Veness, J., Hausknecht, M., & Bowling, M. (2018). Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, *61*, 523–562.

Mackintosh, N. J. (1974). *The psychology of animal learning*. Academic Press.

Mahmood, A. R. & Sutton, R. S. (2013). Representation search through generate and test. In Workshops at the Twenty-Seventh AAAI conference on artificial intelligence.

Menick, J., Elsen, E., Evci, U., Osindero, S., Simonyan, K., & Graves, A. (2020). Practical real time recurrent learning with a sparse approximation. In International conference on learning representations.

Modayil, J., White, A., & Sutton, R. S. (2014). Multi-timescale nexting in a reinforcement learning robot. *Adaptive Behavior*, *22*(2), 146–160. https://doi.org/10.1177/1059712313511648

Mozer, M. C. (1989). A focused back-propagation algorithm for temporal pattern recognition. *Complex Systems*, *3*(4), 349–381.

Nath, S., Liu, V., Chan, A., Li, X., White, A., & White, M. (2019). Training recurrent neural networks online by learning explicit state variables. In International conference on learning representations.

Obando-Ceron, J. S. & Castro, P. S. (2020). Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research. arXiv preprint arXiv:2011.14826.

Osband, I., Doron, Y., Hessel, M., Aslanides, J., Sezener, E., Saraiva, A., McKinney, K., Lattimore, T., Szepesvari, C., Singh, S., Roy, B. V., Sutton, R., Silver, D., & Hasselt, H. V. (2020). Behaviour suite for reinforcement learning.In International conference on learning representations.

Osband, I., Van Roy, B., Russo, D. J., & Wen, Z. (2019). Deep exploration via randomized value functions. *Journal of Machine Learning Research*, *20*(124), 1–62.

Parisotto, E. & Salakhutdinov, R. (2021). Efficient transformers in reinforcement learning using actor-learner distillation. arXiv preprint arXiv:2104.01655.

Parisotto, E., Song, F., Rae, J., Pascanu, R., Gulcehre, C., Jayakumar, S., Jaderberg, M., Kaufman, R. L., Clark, A., & Noury, S. (2020b). Stabilizing transformers for reinforcement learning. In International conference on machine learning (pp. 7487–7498). PMLR.

Parisotto, E., Song, F., Rae, J., Pascanu, R., Gulcehre, C., Jayakumar, S., Jaderberg, M., Kaufman, R. L., Clark, A., Noury, S., Botvinick, M., Heess, N., & Hadsell, R. (2020a).

Stabilizing transformers for reinforcement learning. In H. Daumé III & A. Singh (Eds.), Proceedings of the 37th International conference on machine learning, proceedings of machine learning research (Vol. 119, pp. 7487–7498). PMLR.

Pavlov, I. P. (1927). *Conditioned reflexes: An investigation of the physiological activity of the cerebral cortex* (Vol. 3). Oxford University Press.

Rivest, F., Kalaska, J. F., & Bengio, Y. (2014). Conditioning and time representation in long short-term memory networks. *Biological Cybernetics*, *108*(1), 23–48. https://doi.org/10.1007/s00422-013-0575-1

Schneiderman, N. (1966). Interstimulus interval function of the nictitating membrane response of the rabbit under delay versus trace conditioning. *Journal of Comparative and Physiological Psychology*, *62*(3), 397–402. https://doi.org/10.1037/h0023946

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, *3*(1), 9–44. https://doi.org/10.1007/BF00115009

Sutton, R. S. (1992). Adapting bias by gradient descent: An incremental version of delta-bar-delta. In Proceedings of the Tenth National conference on artificial intelligence (pp. 171–176). MIT Press.

Sutton, R. S. & Barto, A. G. (1990). Time-derivative models of pavlovian reinforcement. In M. Gabriel & J. Moore (Eds.), *Learning and computational neuroscience: Foundations of adaptive networks* (pp. 497–537). MIT Press.

Sutton, R. S. & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Sutton, R. S., Koop, A., & Silver, D. (2007). On the role of tracking in stationary environments. In Proceedings of the 24th International conference on machine learning (pp. 871–878).

Sutton, R. S. & Whitehead, S. D. (1993). Online learning with random representations. In Proceedings of the Tenth International conference on machine learning (pp. 314–321).

Tallec, C. & Ollivier, Y. (2018). Unbiased online recurrent optimization. In International conference on learning representations.

Todorov, E., Erez, T., & Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In IEEE/RSJ International conference on intelligent robots and systems (pp. 5026–5033). IEEE.

Tucker, G., Bhupatiraju, S., Gu, S., Turner, R., Ghahramani, Z., & Levine, S. (2018). The mirage of action-dependent baselines in reinforcement learning. In International conference on machine learning (pp. 5015–5024). PMLR.

van Hasselt, H. & Sutton, R. S. (2015). Learning to predict independent of span. arXiv preprint arXiv:1508.04582.

Wagner, A. R. (1978). Expectancies and the priming of stm. In *Cognitive processes in animal behavior* (pp. 177–209). Routledge.

Wayne, G., Hung, C., Amos, D., Mirza, M., Ahuja, A., Grabska-Barwinska, A., Rae, J., Mirowski, P., Leibo, J. Z., Santoro, A., Gemici, M., Reynolds, M., Harley, T., Abramson, J., Mohamed, S., Rezende, D., Saxton, D., Cain, A., Hillier, C., & Lillicrap, T. (2018). Unsupervised predictive memory in a goal-directed agent. arXiv preprint arXiv:1803.10760.

Whiteson, S., Tanner, B., & White, A. (2010). Report on the 2008 reinforcement learning competition. *AI Magazine*, *31*(2), 81–81.

Williams, R. J. & Peng, J. (1990). An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural Computation*, *2*(4), 490–501. https://doi.org/10.1162/neco.1990.2.4.490

Williams, D. A., Todd, T. P., Chubala, C. M., & Ludvig, E. A. (2017). Intertrial unconditioned stimuli differentially impact trace conditioning. *Learning & Behavior*, *45*(1), 49–61. https://doi.org/10.3758/s13420-016-0240-3

Williams, R. J. & Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, *1*(2), 270–280. https://doi.org/10.1162/neco.1989.1.2.270

Zhang, S. & Sutton, R. S. (2017). A deeper look at experience replay. arXiv preprint arXiv:1712.01275.

About the Authors

Banafsheh Rafiee is a PhD student at the University of Alberta. Her research interest is to investigate the problem of representation learning in reinforcement learning. Banafsheh received her MSc in computer science from the University of Alberta in 2018.

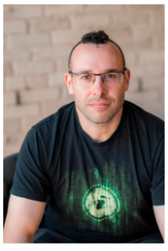
Zaheer Abbas is a research engineer at DeepMind Alberta, where he investigates scalable reinforcement learning methods. Before joining DeepMind, Zaheer completed a Master in Computing Science degree at the University of Alberta.

Sina Ghiassian is a PhD student at the University of Alberta. His research interests are scalable off-policy and Emphatic learning algorithms. Previously, Sina finished his MSc at the University of Alberta where he studied Supervised Learning.

Raksha Kumaraswamy is a Senior Researcher at Huawei Technologies Canada Co. Ltd., and a recent graduate of University of Alberta. Her research interests are directed towards improving sample-efficiency of reinforcement learning algorithms, with a focus on exploration and representation learning.

Richard S. Sutton is a distinguished research scientist at DeepMind, a professor in the Department of Computing Science at the University of Alberta, and a fellow of the Royal Society (UK), the Royal Society of Canada, the Association for the Advancement of Artificial Intelligence, the Alberta Machine Intelligence Institute (Amii), and CIFAR. He received a PhD in computer science from the University of Massachusetts in 1984 and a BA in psychology from Stanford University in 1978. Prior to joining the University of Alberta in 2003, he worked in industry at AT&T Labs and GTE Labs, and in academia at the University of Massachusetts. In Alberta, he founded the Reinforcement Learning and Artificial Intelligence Lab, which now consists of ten principal investigators and about 100 people altogether. He joined DeepMind in 2017 to co-found their first satellite research lab, in Alberta.

Elliot A. Ludvig is a Professor in the Department of Psychology at the University of Warwick, UK. His research aims to understand how humans and other animals learn to make better decisions, using computational models and behavioral experiments. He completed his Ph.D. in Psychological and Brain Sciences at Duke University in 2003 on timing in pigeons. During his career, he has also studied the behavior of humans, animals, and machines at Princeton University, the Technion, the University of Alberta, PsychoGenics Inc., and Rutgers University.



Adam White is an Assistant Professor at the University of Alberta and a Staff Research Scientist at DeepMind. He is a principal investigator of the Alberta Machine Intelligence Institute and the Reinforcement Learning and Artificial Intelligence group at the University of Alberta. Adam is a Canada CIFAR Chair in Artificial Intelligence. His research program explores how the problem of intelligence can be modeled as a reinforcement learning agent interacting with some unknown environment, learning from a scalar reward signal rather than explicit feedback.