Improving Reinforcement Learning with Biologically Motivated Continuous State Representations

Madeleine Bartlett[1†] (madeleine.bartlett@uwaterloo.ca),
Kathryn Simone[1†] (kpsimone@uwaterloo.ca),
Nicole Sandra-Yaffa Dumont[2†] (ns2dumont@uwaterloo.ca),
P. Michael Furlong[2] (michael.furlong@uwaterloo.ca),
Chris Eliasmith[2] (celiasmith@uwaterloo.ca),
Jeff Orchard[1] (jorchard@uwaterloo.ca),
and Terrence C. Stewart[3] (terrence.stewart@nrc-cnrc.gc.ca)

[1] Cheriton School of Computer Science, University of Waterloo, Waterloo, ON N2L 3G1, Canada
[2] Centre for Theoretical Neuroscience, University of Waterloo, Waterloo, ON N2L 3G1, Canada
[3] National Research Council of Canada, University of Waterloo Collaboration Centre, Waterloo, ON N2L 3G1, Canada

Abstract

Learning from experience, often formalized as Reinforcement Learning (RL), is a vital means for agents to develop successful behaviours in natural environments. However, while biological organisms are embedded in continuous spaces and continuous time, many artificial agents use RL algorithms that implicitly assume some form of discretization of the state space, which can lead to inefficient resource use and improper learning. In this paper we show that biologically motivated representations of continuous spaces form a valuable state representation for RL. We use models of grid and place cells in the Medial Entorhinal Cortex and hippocampus, respectively, to represent continuous states in a navigation task and in the CartPole control task. Specifically, we model the hexagonal grid structures found in the brain using Hexagonal Spatial Semantic Pointers, and combine this state representation with single-hidden-layer neural networks to learn action policies in an Actor-Critic framework. We demonstrate our approach provides significantly increased robustness to changes in environment parameters (travel velocity), and learns to stabilize the dynamics of the CartPole system with comparable mean performance to a deep neural network, while decreasing the terminal reward variance by more than 150x across trials. These findings at once point to the utility of leveraging biologically motivated representations for RL problems, and suggest a more general role for hexagonally-structured representations in cognition.

Keywords: Reinforcement Learning; Grid Cells; Spatial Semantic Pointers; Representations

Introduction

Humans and animals are able to learn how to interact with their environment through a process of trial-and-error, repeating behaviours that lead to high rewards, and avoiding behaviors that lead to punishments. This process is known as conditioning and has inspired the development of Reinforcement Learning (RL) algorithms for training computational systems. RL algorithms, in turn, have provided further insights into the nature of learning in biological agents.

Classical RL algorithms discretize state and action spaces, and assume that time can be divided into discrete time steps. However, biological agents exist and evolve in continuous time and space, and therefore their learning mechanisms must also operate in continuous domains. While discretized representations are convenient when working with standard computers and can often produce good results in RL, they have limitations. For instance, a coarse discretization can result in non-smooth control output and poor performance, while a fine discretization can lead to an explosion in the number of states, memory resources, and time required to learn. Additionally, selecting a discretization that does not match the "correct discretization" of the environment may result in either poor representation or inefficient resource use, or both. Obtaining a good discretization scheme often requires prior knowledge or trial and error. Furthermore, the environment itself may not remain stable during its operational lifespan. This could cause the selected and optimal discretizations to diverge over time, at the expense of either performance or wasted representational resources.

In RL, feature representation plays a crucial role in performance. Various feature encoding methods have been proposed, including discretization of the state space through techniques like tile coding (Sutton, 1996) and using deep auto-encoders to obtain latent state representations (Lange & Riedmiller, 2010). In deep RL networks, a linear output layer is typically used, and so the majority of the neural network can be viewed as a state encoding network followed by linear value function approximation. Additionally, biologically inspired state representations have been explored. For instance, RL agents using grid-cell-like representations have outperformed both deep AC models and agents with place-cell-like representations in 2D navigation tasks (Banino et al., 2018). This supports the idea that grid cells provide a useful basis for RL tasks. However, to our knowledge these benefits have been established only for navigation tasks. The extent to which these benefit generalize to other types of tasks remains an open question.

The use of grid or place cell-like encodings of spatial information in RL networks has been demonstrated to facilitate faster learning on spatial navigation tasks (Gustafson & Daw, 2011; Banino et al., 2018; Dumont & Eliasmith, 2020; Bartlett et al., 2022a,b). The method of modelling grid cells

[†] These authors contributed equally.

used in the present work, first presented by Dumont & Eliasmith (2020), builds on Spatial Semantic Pointers (SSPs; Dumont et al., 2023; Komer & Eliasmith, 2020; Komer et al., 2019), a high-dimensional representation of continuous values used in cognitive models that employ vector symbolic architectures (VSAs). This method allows us to represent continuous state information as a specific type of SSP (hexagonal SSPs, or HexSSPs). The resulting representations can be mapped to individual neurons giving rise to grid cells. HexSSPs have been demonstrated to be flexible, computationally efficient and, being a VSA, highly interpretable (Dumont et al., 2023; Komer & Eliasmith, 2020; Bartlett et al., 2022a,b). In general, SSPs can be generated for any spatial environment regardless of size or shape, and scaled to accommodate changes in the environment (Komer & Eliasmith, 2020). Komer & Eliasmith (2020) further demonstrated that HexSSPs encoding spatial location can support supervised learning of policies to navigate through complex, continuous-space environments containing obstacles. HexSSPs have also proven useful as representations for semantic mapping, Bayesian optimization, and neural representations of probability (Dumont et al., 2023; Furlong et al., 2022; Furlong & Eliasmith, 2022). Due to their ability to represent structured data as vectors (*e.g.*, Voelker et al., 2021), these representations may permit extending simple algorithms to more complex spaces. In particular, SSPs can be used to represent sequences or trajectories through continuous spaces, along with hierarchical representations that mix discrete and continuous data. Consequently, algorithms designed to use SSP input can be applied to tasks with complex feature data.

Recent research has illustrated the usefulness of HexSSPs when learning navigation policies in an online fashion using RL (Bartlett et al., 2022a,b). However, this past work was limited to a task defined in discrete space (Gymnasium's MiniGrid: Chevalier-Boisvert et al., 2018). In this paper, we present the results of a series of simulations demonstrating the benefit of using HexSSPs to represent continuous state to solve tasks with an Advantage Actor-Critic (A2C) network. The A2C network is first tested on a novel spatial navigation task 'RatBox', designed as a continuous-space variant of MiniGrid. Additionally, as the representational capacity of HexSSPs generalizes to representing continuous *feature spaces* (Dumont & Eliasmith, 2020), we also run simulations on a continuous state RL benchmark task, CartPole (Brockman et al., 2016). This problem is analogous to balancing an inverted pendulum, relevant for numerous animal behaviors, such as walking or perching on a branch. CartPole has previously been solved efficiently with continuous representations of the state in an actor-critic model with neural networks (Anderson, 1989) and spiking neural networks (Frémaux et al., 2013). Furthermore, grid cell-like representations of the state have been shown to improve the performance of a Deep Q network on the CartPole problem (Yu et al., 2020).

Methods

Hexagonal SSPs

Spatial Semantic Pointers (SSPs) are a high-dimensional vector representation of lower-dimensional continuous spaces (Plate, 1995; Komer et al., 2019; Komer & Eliasmith, 2020), developed within the framework of the Semantic Pointer Architecture (Eliasmith, 2013). SSPs represent state variables by selecting frequency components in the Fourier domain and using those components to project continuous state variables into the high-dimensional frequency space, followed by an inverse Fourier transform. Specifically, to represent m-dimensional data, $\mathbf{x} \in \mathbb{R}^m$, we generate an encoding matrix, $\Theta \in \mathbb{R}^{d \times m}$, and define the SSP representation of $\mathbf{x}$ as:

$$\phi(\mathbf{x}) = \mathcal{F}^{-1}\{e^{i\Theta\mathbf{x}}\}, \tag{1}$$

where elements of Θ are sampled uniformly from the interval $[-\pi, \pi]$ and d is the dimensionality of the SSP representation. We further constrain Θ so that $e^{i\Theta\mathbf{x}}$ has conjugate symmetry, to ensure the inverse Fourier transform does not generate imaginary components. This method for representing continuous values is also known as *fractional binding* (Komer et al., 2019), *fractional convolution powers* (Plate, 1994), or *fractional power encoding* (Frady et al., 2022).

In this work we use Hexagonal SSPs (HexSSPs), a variant of SSPs in which the encoding matrix is specifically structured to model grid cell activity. The encoding matrix is constructed so that the dot product with an encoded point and other points in the domain mimics the activity of grid cell neurons in the medial entorhinal cortex (MEC) of the hippocampus (Dumont & Eliasmith, 2020).

Algorithm 1 Hexagonal SSP Generator. Given data $\mathbf{x}$ with dimensionality m, this returns its SSP encoding $\phi(\mathbf{x})$. The input scales, $\mathcal{S}$, are scalar values, and rotations, $\mathcal{R}$, are a set of m-dimensional rotation matrices.

procedure HEX-SSP($\mathbf{x}$, m, $\mathcal{S}$, $\mathcal{R}$)
  $\mathbf{v}_1, \ldots, \mathbf{v}_{m+1} \leftarrow$ Coordinates of regular m-dim simplex
  $\mathbf{V} \leftarrow \begin{pmatrix} | & & | \\ \mathbf{v}_1 & \cdots & \mathbf{v}_{m+1} \\ | & & | \end{pmatrix}^T$
  $\Theta \leftarrow \text{stack}\left(\{s\mathbf{R}\mathbf{V} \,|\, s \in \mathcal{S}, \mathbf{R} \in \mathcal{R}\}\right)$
  $\phi(\mathbf{x}) = \mathcal{F}^{-1}\left\{e^{i\Theta\mathbf{x}}\right\}$
  return $\phi(\mathbf{x})$
end procedure

The algorithm for constructing HexSSPs is given in Algorithm 1. HexSSP encoding matrices are constructed from $m+1$ vectors that form a regular m-simplex in m-dimensional space. The simplex is determined by minimizing the expression $\sum_i^m \sum_{j=1, i\neq j}^m \mathbf{v}_i \cdot \mathbf{v}_j$, where $\mathbf{v}_i, \mathbf{v}_j$ are unit vectors that make up the simplex. Stacking these vectors produces an initial $(m+1) \times m$ encoding matrix, $\mathbf{V}$. We can specify the kernel function, $k(\mathbf{x}, \mathbf{x}')$ that is approximated by the dot product between two SSPs, $\phi(\mathbf{x})$ and $\phi(\mathbf{x}')$. With this initial encoding

matrix, $\mathbf{V}$, the resulting kernel function will have a hexagonally tiled pattern.

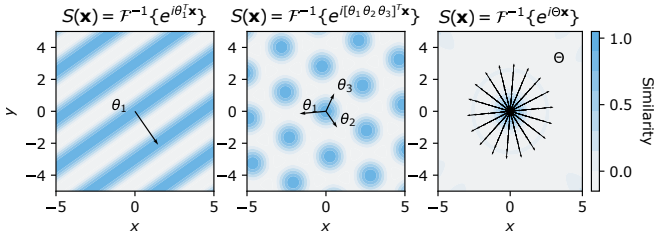

Figure 1: **Construction of HexSSPs from Fourier basis functions.** Left: An encoding matrix consisting of a single Fourier basis function, $e^{i\theta_1^T x}$, results in a kernel function, $k(\mathbf{x},\mathbf{x}')$, with oscillations in 2D space characteristic of spatial frequency representation. $k(\mathbf{x},\mathbf{x}')$, in turn, is approximated by the dot product between HexSSPs, each representing 2-dimensional variables. Middle: When m+1 such Fourier basis functions are included (3 for the 2-dimensional space depicted here, and spaced 120° apart), the interference pattern results in the hexagonally-patterned kernel function. Right: As more rotations and scales of these vectors are used to generate the encoding matrix, the kernel function becomes smoother, with one centralized peak and more shallow local optima.

The firing patterns of grid neurons in the MEC of the hippocampus are characterized by different orientations and sizes. To mimic this features, the complete encoding matrix, Θ, used to construct HexSSPs is composed of multiple rotations and scalings of $\mathbf{V}$. This choice also has useful practical implications: as more rotations and scales are added to the representation, the kernel function becomes smoother, with one centralized peak and more shallow local optima, as shown in Figure 1. SSPs created with such encoding matrices are also more robust to noise than randomly generated representations, and so can be more accurately encoded in spiking neural networks via grid cells (Dumont & Eliasmith, 2020).

Advantage Actor-Critic Network

The Advantage Actor-Critic (A2C) network implemented for these simulations was the same as that presented in (Bartlett et al., 2022a). The network architecture is shown in Figure 2. It was implemented in Python and Nengo (Bekolay et al., 2014) using the principles of the Neural Engineering Framework (NEF; Eliasmith & Anderson, 2003). The code is available at `https://github.com/maddybartlett/ImprovedRLContinuousStateReps`.

States were represented either as a one-hot vector or by a population of rectified linear neurons. When solving the RatBox task, encoders were sampled from the regions of SSP space associated with the observation space using the Sobel sampling method, thus generating a population of grid cell neurons. The number of neurons in this case was calculated such that the number of neurons was at least 10 times the dimensionality of the HexSSP and a power of 2 (a requirement of the Sobel sampling method). For the CartPole task, the encoders were randomly sampled from the whole SSP space, as the observation space is unbounded for some state variables. Other than the state representation layer, no other part of the network used neurons. Learning was performed on the connection weights from the state representation layer to the output. Connection weights were initialized to zero.

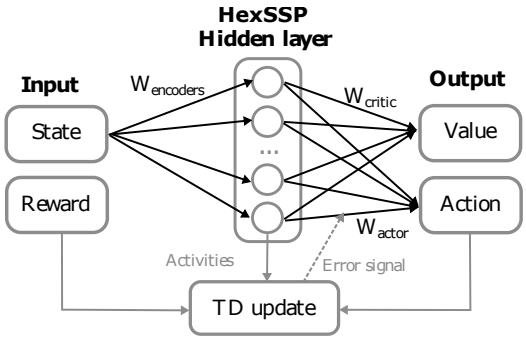

Figure 2: A schematic of the Advantage Actor-Critic (A2C) network. $W_{encoders}$ project a HexSSP representation of the state to neurons in the Hidden Layer.

Hyperparameter optimization

The performance of RL networks and algorithms is sensitive to the selection of hyperparameters (Sutton & Barto, 2018). To identify the hyperparameter configuration that maximizes performance, we first defined the performance metric as the terminal return, in turn computed as the mean reward received in the last 100 episodes of a single trial. We then searched for the configuration of hyperparameters that maximize performance, a process referred to as hyperparameter optimization, using the simulated annealing algorithm implemented in the Neural Network Intelligence (NNI; Microsoft, 2021) Python package. This algorithm begins by randomly sampling from the hyperparameter space, and progresses by sampling from regions that achieved higher performance. Each NNI experiment therefore determines the performance achievable given the stochastic selection of hyperparameters. For both the HexSSP network and all state discretizations, 100 such NNI experiments were conducted. The random seed was fixed across all NNI experiments. Hyperparameter optimization was performed over the parameter ranges specified in Table 1, which were selected based on performance results obtained through systematic exploration of the hyperparameter space (Bartlett et al., 2022a) on a navigation task similar to that used in this present work. In cases where identical, optimal performance could be achieved with multiple sets of hyperparameters, the set of hyperparameters was selected based on the most temporally stable terminal behavior. For discrete representations of the state, the number of bins per state was set for each NNI experiment and the remaining hyperparameters were left free.

Symbol	Variable	Range - RatBox	Range - CartPole
ε	probability of an off-policy action	$[0.3, 0.6]$	$[0.2, 0.6]$
α	learning rate	$[0.001, 0.5]$	$[0.001, 0.5]$
β	action value discount	$[0.8, 1.0]$	0.9
γ	state value discount	$[0.8, 1.0]$	0.99
η	proportion of active cells	⬡ $[0.01, 0.5]$	⬡ $[0.01, 0.5]$
N	number of neurons		⬡ $\{1024, 2048, 4096\}$
$\mathcal{R}$	rotations of V	⬡ $[4, 5, 6, 7, 8, 9, 10]$	⬡ $[4, 5, 6, 7]$
$\mathcal{S}$	scalings of V	⬡ $[4, 5, 6, 7, 8, 9, 10]$	⬡ 8
l	length scale of representation	⬡ $[1, 100]$	⬡ $[0.01, 1.0]$

Table 1: Hyperparameter values tested during optimization of networks solving each task. Hyperparameters marked by ⬡ apply only to models using HexSSPs for state representation.

Evaluation on continuous space navigation with obstacles (RatBox)

Discrete state representations are incapable of perfectly capturing the boundaries of irregularly shaped objects. A novel 2D environment, 'RatBox', containing 4 obstacles that an agent must navigate around to reach a goal location was developed for these experiments (see Figure 3), to assess the ability of HexSSPs to learn an efficient policy in this scenario. The state of the agent in this environment is its 2D position and heading, $s = (x, y, w)$. The state space within the environment is continuous in that the agent can be in any location within the 600×600 space. Additionally the agent is able to face any direction, $w \in [0, 2\pi]$.

The discrete agent's action space consists of a set of vectors, $\{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_{n_a}\}$. In this task there are 4 action primitives, each corresponding with a 'compass' direction (North, South, East, West). The discrete A2C network learns a policy over this discrete action space. To allow for a continuous action space, the action taken is a weighted sum of the action primitives, *i.e.*, $\mathbf{a}(t) = \sum_i^{n_a} c_i \mathbf{a}_i$. In this case, the output from the actor portion of the network is a 4-vector, c, consisting of the learned value for each action primitive. This weighted sum is the direction vector for the agent moving at a fixed speed. The agent's maximum speed was set to 10,000 pps (pixels per second), which equates to 100 pixels per timestep.

To use this method of representing a continuous action space, we formulate our policy as an isotropic Gaussian distribution over the action vector, $\pi_W(s) = \mathcal{N}(\mu_W(s), \sigma^2 I)$, with small isotropic noise, $\sigma^2 << 1$, and where $\mu_W(s)$ is the weighted sum of the discrete action vectors,

$$\mu_W(s) = \text{softmax}(W\phi(s))^T [\mathbf{a}_1, \ldots, \mathbf{a}_{n_a}]^T \tag{2}$$

This parameterization of the mean action vector is the softmax of a linear decoding from the state population, $W\phi(s)$.

Figure 3: The RatBox environment

We assume the isotropic Gaussian noise added to this action is small to obtain an approximately-deterministic stochastic policy. Then we can derive the approximate policy gradient from the expected rate of reward as a function of the policy parameters, $J(W)$:

$$\nabla_W J(W) = \mathbb{E}[\nabla_W \log \pi_W(s)\, A(s,a)] \tag{3}$$

$$\nabla_w \log \pi_W(s) = (I - \pi_W(s))\pi_W(s)\phi(s)^T \tag{4}$$

$$W_{\text{new}} = W_{\text{old}} + \alpha \nabla_W J(W), \tag{5}$$

where $A(s,a)$ is the advantage function, α is the learning rate, and I is the $n_a \times n_a$ identity matrix. With this update, we can improve the policy – parameterized by the decoding weight matrix, $W \in \mathbb{R}^{n_a \times d}$ – with the TD(0) actor-critic learning rule.

The network was tested under two different conditions. In the baseline condition, the agent's state was represented using a tabular representation. We generated several resolutions by applying 6, 8, 10 or 12 partitions to each of the three state dimensions. In the second condition, the state was represented using HexSSPs and a population of grid cell neurons. In the 10- and 12-bin discrete conditions, none of the hyperparameter combinations tested by NNI were able to solve the task. We therefore instead used the hyperparameters found for the 6-bin condition. The optimal network for each condition was then trained 10 times using 10 different random seeds. Interestingly, in the 10-bin condition, the final network was able to learn the task on some of the random seeds.

The average reward was calculated over a 100-trial window, and then averaged across the 10 random seeds. The results, shown in Figure 4, illustrate that the HexSSP and 6-bin solutions were able to learn the task. Performance declines as the resolution becomes finer in the discrete condition.

Learning in non-stationary environments can be challenging for RL algorithms that use a tabular representation of the state, as changes in the environment can potentially cause incompatibilities between the optimal and actual discretizations. In general, continuous state representations avoid this limitation by allowing for generalization between states. This holds for the HexSSP approach we use here, where the extent of generalization over the state space is specified by the length scale parameter. Assuming an appropriate length scale, we would therefore expect our algorithm to exhibit robustness to changes in the environment, as compared to the state discretization approaches. To test this prediction, we performed hyperparameter optimization on the network with the agent's speed set to 10,000 pixels per second (pix/sec), and then assessed performance with the agent's speed set to either 10,000 pix/sec or reduced to 5,000 pix/sec. No other changes were made and the networks were tested again with 10 random seeds. The results are shown in Figure 4. While the performance of the network using HexSSPs to represent the state drops slightly following the change in speed, closer inspection revealed that this was due to two of the ten random seeds resulting in no learning, while the rest of the seeds led to performance as good as the original (data not shown). In contrast, none of the baseline networks were able to solve the

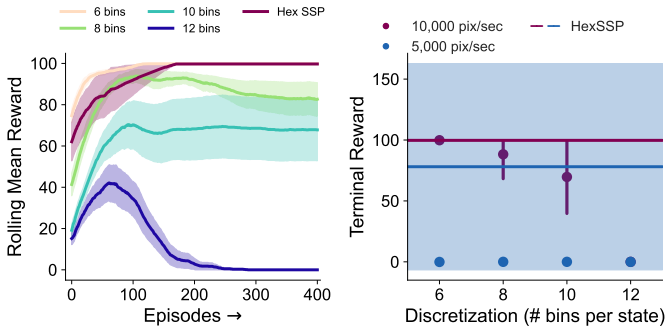

Figure 4: **Performance on RatBox.** Left: Learning curves using HexSSPs or tabular approaches for state representations, averaged across 10 random seeds. The shaded area depicts the standard error of the mean. Right: Average terminal reward (with 95% confidence interval) across the different representations when the agent's maximum speed was 10,000 pix/sec vs. 5,000 pix/sec.

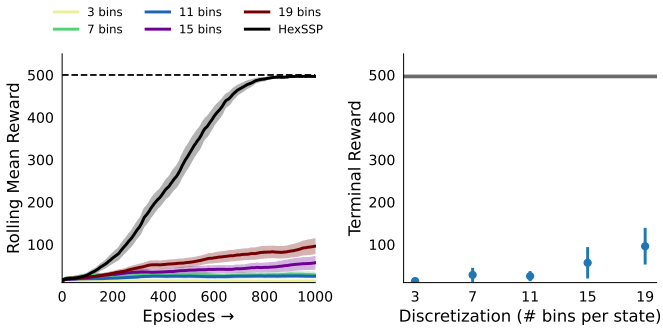

Figure 5: **Performance on CartPole.** Left: Learning curves using HexSSPs or tabular approaches for state representation. Shown is the mean across 10 runs of the model. The shaded area depicts the standard error of the mean. The dotted line indicates the maximum possible episodic reward achievable. Right: Terminal rewards observed in 10 runs of each model (and 95% confidence intervals) for each discretization condition. The gray bar denotes the 95% confidence interval for performance using HexSSPs.

task following the change in agent speed, suggesting that the networks would need a different discretization, or to be re-optimized, in order to adapt. This result shows that continuous HexSSPs are a more general solution that is less sensitive to changes in the task or environment compared to standard discrete representations.

Evaluation on the inverted pendulum (CartPole) problem

Here we characterize learning on the standard CartPole task from OpenAI's Gym Library (Brockman et al., 2016). The CartPole's dynamics are unstable, making performance particularly sensitive to state representation accuracy. Errors introduced due to state discretization should therefore cause a decrease in performance. Indeed, prior work investigating learning with continuous and discrete control schemes report fewer trials to learn with a continuous algorithm compared to a discrete algorithm on the similar CartPole Swingup task (Doya, 2000). We therefore first sought to validate that the proposed continuous representation confers a learning advantage over the state discretization approach. We characterized learning across 5 discretizations corresponding to a distinct number of partitions applied to each of the 4 state variables. Hyperparameter optimization was performed for each discretization, and performance characterized across 10 runs of the model with different seeds.

As can be seen in Figure 5, terminal reward grows slowly as the resolution of the discretization increases, but is well below the terminal reward achieved by the HexSSP model. Crucially, and in contrast to performance with most of the tabular approaches, terminal performance using HexSSPs to represent state exhibits relatively small variation across seeds (95% confidence interval: [496.88,499.97]). This is especially surprising as the only source of randomness using tabular approaches is the initial conditions given by the environment; HexSSPs are subject to this source of randomness in addition to that incurred by the sampling of encoders for each model run. Of course, hand-tuned discretizations may facilitate superior or more consistent performance, but this requires trial and error or *a priori* knowledge of the problem. HexSSPs reliably produce high terminal performance regardless of initial environment or network conditions in the tested scenario.

Performance comparison against a deep network baseline

So far we have described two advantages of representing state information as a continuous variable with HexSSPs, as measured against state discretization approaches. On the navigation task with RatBox, HexSSP representations conferred greater robustness to changes in model parameters. On the CartPole control task, we observed better terminal performance and lower sensitivity to initial conditions. However, neural networks can also represent continuous state information, and multi-layer networks, in particular, can learn effective representations for decoding value and policy functions. This can include representations similar to that which we have used here in their hidden layers. What specific advantage, if any, is offered by using HexSSP representations on RL problems over the state-of-the-art?

To address this question, we compare the performance of our algorithm to an A2C method that uses a multi-layer perceptron policy (Raffin et al., 2021) on the CartPole task. Figure 6 shows learning curves for both the proposed HexSSP single-layer network model and the deep network baseline model. On the CartPole task, the two methods exhibit comparable performance, as shown by the overlap in the medians and interdecile range produced by a range of initial seeds. Interestingly, the HexSSP method produces more re-

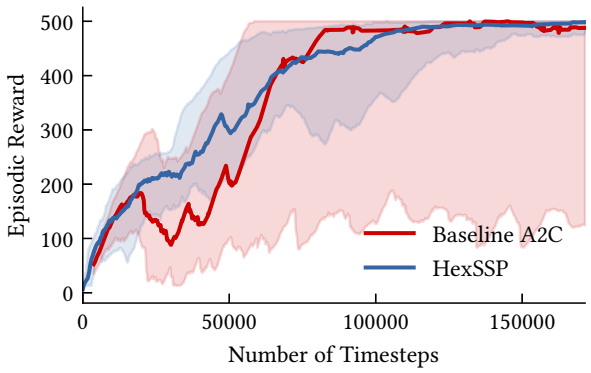

Figure 6: **Performance of a Single layer A2C network with HexSSPs and a deep A2C network on CartPole.** Comparing the moving average over episodic rewards between our proposed method using HexSSP representations and a baseline implementation of A2C with a multilayer perceptron policy from (Raffin et al., 2021). Solid lines are the medians and shaded areas are the interdecile range, taken over 20 randomly chosen seeds.

liable terminal performance, indicating less sensitivity to the initial seed as shown by lower variance in the terminal reward (Baseline: σ^2 = 20966.52, HexSSP: σ^2 = 127.84, L = 6.19, P = 0.017, Levene's test of equal variances using group medians). The high reliability of performance with HexSSP representations as compared to that observed with a state-of-the-art approach suggests a fundamental robustness across different types of continuous feature spaces. We speculate that since the baseline model must learn the state representation, it is therefore able to modify this representation late in training when it may no longer be advantageous to do so. The representation for the HexSSP model is fixed, so additional training is not able to degrade its performance in this way.

Discussion

In this paper, we presented HexSSPs as a method for representing continuous states when solving tasks using RL. We evaluated the HexSSPs on a spatial navigation task and compared performance to networks using tabular representations, and to a multi-layer perceptron where the state representation is learned via backpropagation. While a discrete representation could solve the RatBox task as well as the HexSSP method, we found that the discrete method was not robust to changes in the task; the HexSSPs proved to be very robust. We also found that the HexSSP representation was able to achieve a final performance greater than any of the tabular representations on the standard benchmark CartPole task. The HexSSP solution's performance was also comparable to that of the Deep A2C network, but it is noteworthy that the HexSSP network produced a more reliable terminal performance suggesting that it is more robust to changes in network initialisation. Notably, the performance on CartPole was achieved while adopting a relatively inefficient method of sampling encoders from the subspace spanned by the problem domain. An important direction for future work is therefore to explore the impact of more efficient sampling methods on this task.

Experiments comparing the HexSSP method and tabular approaches were designed to assess differences in average performance. Assuming the performance metric follows a Gaussian distribution, the sample size of 10 opted for in this work confers to us a 99.8 % probability that the mean performance falls within the span of the sampled data points (computed as $P = (1 - \frac{1}{2^{N-1}}) \times 100\%$, where N is the number of sample points). However, the observation that 2 (20%) of the random seeds yield particularly poor performance on the RatBox task suggest that more sampling would be needed if one were to move beyond characterizing average performance and towards understanding the full distribution of model behavior.

A key characteristic of the HexSSP method is that it allows one to build neuron populations whose firing patterns mimic those of grid and place cells found in MEC and hippocampus across many animal species. The performance improvement on the RatBox task gained by leveraging these hexagonally-patterned representations is expected considering their role in spatial navigation in biological agents. These firing patterns are readily comparable to neural recordings from animal models used in neuroscience, such as rodents. However, to our knowledge, these representations have not been implicated in motor control analogous to that required to solve the CartPole task. In this case, comparing the performance of our network against the behavior of a biological agent may yield insight into the specific role of hexagonally-patterned representations in biological cognition.

On the RL problems explored in this work, we observed the benefits of HexSSPs being more robust to noise and better able to generalize to changes in the environment. These features are essential for autonomous agents needing to learn online, potentially in dynamic or unstable environments. Thus an interesting avenue for future work would be to explore the extent to which the HexSSP method proves useful in solving non-stationary problems. Additionally, the robustness to noise is significant for cognitive models, and this representation supports encoding in spiking neural networks. Moreover, these spiking implementations can be implemented straightforwardly on neuromorphic hardware.

Online Resources

All code necessary to reproduce these results are hosted at: `https://github.com/maddybartlett/ImprovedRLContinuousStateReps`

Acknowledgments

This project was supported in part by collaborative research funding from the National Research Council of Canada's Artificial Intelligence for Logistics program (AI4L-116), as well as by CFI (52479-10006) and OIT (35768) infrastructure funding, the Canada Research Chairs program, and NSERC Discovery grant 261453.

References

Anderson, C. W. (1989). Learning to control an inverted pendulum using neural networks. *IEEE Control Systems Magazine*, *9*(3), 31–37.

Banino, A., Barry, C., Uria, B., Blundell, C., Lillicrap, T., Mirowski, P., ... others (2018). Vector-based navigation using grid-like representations in artificial agents. *Nature*, *557*(7705), 429–433.

Bartlett, M. E., Stewart, T. C., & Orchard, J. (2022a). Biologically-based neural representations enable fast online shallow reinforcement learning. In *Proceedings of the annual meeting of the cognitive science society* (Vol. 44).

Bartlett, M. E., Stewart, T. C., & Orchard, J. (2022b). Fast online reinforcement learning with biologically-based state representations. In *Proceedings of the 20th international conference on cognitive modeling.*

Bekolay, T., Bergstra, J., Hunsberger, E., DeWolf, T., Stewart, T. C., Rasmussen, D., ... Eliasmith, C. (2014). Nengo: a Python tool for building large-scale functional brain models. *Frontiers in neuroinformatics*, *7*, 48.

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

Chevalier-Boisvert, M., Willems, L., & Pal, S. (2018). *Minimalistic gridworld environment for gymnasium.* Retrieved from `https://github.com/Farama-Foundation/Minigrid`

Doya, K. (2000). Reinforcement learning in continuous time and space. *Neural computation*, *12*(1), 219–245.

Dumont, N. S.-Y., & Eliasmith, C. (2020). Accurate representation for spatial cognition using grid cells. In *Cogsci.*

Dumont, N. S.-Y., Stöckel, A., Furlong, P. M., Bartlett, M. E., Eliasmith, C., & Stewart, T. C. (2023). Biologically-based computation: How neural details and dynamics are suited for implementing a variety of algorithms. *Brain Sciences*, *13*(2), 245.

Eliasmith, C. (2013). *How to build a brain: A neural architecture for biological cognition*. Oxford University Press.

Eliasmith, C., & Anderson, C. H. (2003). *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT press.

Frady, E. P., Kleyko, D., Kymn, C. J., Olshausen, B. A., & Sommer, F. T. (2022). Computing on functions using randomized vector representations (in brief). In *Neuro-inspired computational elements conference* (p. 115-122).

Frémaux, N., Sprekeler, H., & Gerstner, W. (2013). Reinforcement learning using a continuous time actor-critic framework with spiking neurons. *PLoS computational biology*, *9*(4), e1003024.

Furlong, P. M., & Eliasmith, C. (2022). Fractional binding in vector symbolic architectures as quasi-probability statements. In *Proceedings of the annual meeting of the cognitive science society* (Vol. 44).

Furlong, P. M., Stewart, T. C., & Eliasmith, C. (2022). Fractional binding in vector symbolic representations for efficient mutual information exploration. In *Proc. icra workshop, towards curious robots, mod. approaches intrinsically-motivated intell. behav.* (pp. 1–5).

Gustafson, N. J., & Daw, N. D. (2011). Grid cells, place cells, and geodesic generalization for spatial reinforcement learning. *PLoS Computational Biology*, *7*(10), e1002235.

Komer, B., & Eliasmith, C. (2020). Efficient navigation using a scalable, biologically inspired spatial representation. In *Cogsci.*

Komer, B., Stewart, T. C., Voelker, A. R., & Eliasmith, C. (2019). A neural representation of continuous space using fractional binding. In *41st annual meeting of the cognitive science society.* Montreal, QC: Cognitive Science Society.

Lange, S., & Riedmiller, M. (2010). Deep auto-encoder neural networks in reinforcement learning. In *The 2010 international joint conference on neural networks (ijcnn)* (pp. 1–8).

Microsoft. (2021, 1). *Neural Network Intelligence.* Retrieved from `https://github.com/microsoft/nni`

Plate, T. A. (1994). *Distributed representations and nested compositional structure*. Citeseer.

Plate, T. A. (1995). Holographic reduced representations. *IEEE Transactions on Neural Networks*, *6*, 623-641.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., & Dormann, N. (2021). Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, *22*(268), 1-8.

Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. In D. Touretzky, M. Mozer, & M. Hasselmo (Eds.), *Advances in neural information processing systems* (Vol. 8). MIT Press.

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

Voelker, A. R., Blouw, P., Choo, X., Dumont, N. S.-Y., Stewart, T. C., & Eliasmith, C. (2021). Simulating and predicting dynamical systems with spatial semantic pointers. *Neural Computation*, *33*(8), 2033–2067.

Yu, C., Behrens, T. E., & Burgess, N. (2020). Prediction and generalisation over directed actions by grid cells. *arXiv preprint arXiv:2006.03355*.