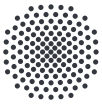


University of Stuttgart
Germany

Institute for Visualization and Interactive Systems
Universitätsstraße 38
70569 Stuttgart

Bachelorarbeit

Biologically Plausible Reinforcement Learning

Tobias Stegmaier

Course of Study:	Informatik
Examiner:	Prof. Dr. Andreas Bulling
Supervisor:	Anna Penzkofer, M.Sc.
Commenced:	May 4, 2023
Completed:	November 4, 2023

Abstract

The fundamental idea of Reinforcement Learning (RL) is learning by interacting with an environment through trial and error and is inspired by the way animals learn in nature. However, standard RL approaches still struggle with seemingly simple tasks, where humans would excel after a few trials. In detail, there is a high volatility between different runs of the RL training process, in addition to a slow convergence rate for each trial.

Previous work has shown that using a biologically inspired approach to RL significantly improves this learning speed and increases the robustness of training across different runs. The biologically based approach uses a state representation, called Spatial Semantic Pointers (SSPs), where embeddings of the state are encoded in a biologically plausible vector space representation. Experiments on a simple two-dimensional navigation task have shown that introducing a grid-like structure into the vector space further increases the learning speed. However, it remains unclear, whether these findings scale to different environments with more complex inputs. Specifically, we are interested in comparing this approach in environments with state spaces to environments with more complex inputs, such as RGB images. Furthermore, the approach is also compared to common artificial neural networks with the state-of-the-art Advantage Actor-Critic (A2C) agent.

Our results suggest that using biologically based state representations leads to a faster learning speed in some environments while causing slower learning in others. Further, the state representations do not extend very well on larger or more complex inputs like images, causing worse performance in both learning speed and overall training time needed.

Kurzfassung

Die grundlegende Idee von Reinforcement Learning (RL), von lernen, indem man mit seiner Umgebung durch Trial-and-Error (Versuch und Irrtum) interagiert, ist ähnlich zu der Art, wie Tiere in der Natur lernen. RL Ansätze haben jedoch Probleme scheinbar einfache Aufgaben zu lösen, in denen Menschen nach nur einigen Testläufen exzellent Leistung zeigen. In mehr Detail, der RL Lernvorgang zeichnet sich durch eine Sprunghaftigkeit über Trainingsdurchläufe aus, zusätzlich zu einer langsamen Konvergenz rate für jeden Trainingsdurchlauf.

Vorherige Arbeit hat gezeigt, dass die Nutzung eines durch die Biologie inspirierten Ansatz zu RL die Lerngeschwindigkeit und die Stabilität über Trainingsdurchläufen erhöht. Die biologisch inspirierten Ansätze nutzen eine Repräsentation für den State (Zustand), die sich Spatial Semantic Pointer (SSP) nennt, die den State durch die Nutzung eines biologisch plausiblen Vektorraum kodiert. Experimente mit einer zweidimensionalen Navigationsaufgabe haben gezeigt, dass das Einführen einer rasterähnlichen Struktur in den Vektorraum die Lerngeschwindigkeit noch weiter erhöht. Es ist jedoch nicht sicher, ob diese Ergebnisse sich auch so in anderen Umgebungen mit komplexeren Werten ausdehnen lassen. Diese Arbeit vergleicht diesen Ansatz in Umgebungen mit komplexeren Inputs (Werte, die in ein neuronales Netz gegeben werden) wie zum Beispiel RGB-Bilder. Des Weiteren wird der Ansatz, mit einem regulären künstlichen neuronalen Netz (ANN), mit einem Advantage Actor-Critic (A2C) verglichen, einem Model des aktuellen Technikstands.

Die Ergebnisse legen nahe, dass biologisch inspirierte Repräsentationen für den Zustand zu einer höheren Lerngeschwindigkeit in einigen Umgebungen führt, aber zu langsameres Lernen in anderen Umgebungen. Zusätzlich lassen sich die Repräsentationen für den Zustand nicht gut auf größere oder komplexere Inputs erweitern, da sie schlechtere Ergebnisse in Lerngeschwindigkeit und benötigte Trainingszeit haben.

Contents

List of Figures

List of Tables

Acronyms

A2C Advantage Actor-Critic. 21

A3C Asynchronous Advantage Actor-Critic. 21, 23

ANN artificial neural network. 15

LIF Leaky Integrate-And-Fire. 19

RL Reinforcement Learning. 15

SNN spiking neural network. 19

SPA Semantic Pointer Architecture. 17

Spaun Semantic Pointer Architecture Unified Network. 17

SSP Spatial Semantic Pointer. 15

TD temporal-difference. 20

1 Introduction

The goal of Reinforcement Learning (RL) is learning how to choose from different actions, to maximize some reward signal [SB18]. To achieve this reward maximization, RL methods use trial and error to determine which actions lead to a better outcome in a given situation. In that sense, RL uses a similar approach to the way humans and animals learn. General deep learning approaches to RL often use the state of the current system as its direct input for decision-making. In fact, the state is usually encoded in artificial representations with no foundation in biology. However, biological systems, as demonstrated by humans, can learn general RL tasks more quickly than artificial neural networks (ANNs). One hypothesis of why this could be the case is that biological systems do not have to learn a new way to represent information for novel tasks, but repurpose learned representations [BSO22].

Recently, Komer et al. [KSVE19] proposed a way to encode continuous values in a biologically plausible vector space called Spatial Semantic Pointers (SSPs). This work has been extended by Dumont and Eliasmith [DE20] to construct a neural representation that mimics grid cells, a special type of neuron in the brain. Prior work of Bartlett et al. [BSO22] and Gustafson and Daw [GD11] have demonstrated, that using the biologically inspired grid cells to encode the state of a RL model improves the learning speed and robustness in comparison to other representations, such as one-hot encoding.

While the results of Bartlett et al. and Gustafson and Daw suggest that using grid cells to encode the state leads to a faster learning speed, they only tested them in a discrete two-dimensional navigation task. This is a type of task grid cells naturally excel at because they are optimized to encode spatial information [BSO22]. This makes it uncertain if an increase in learning speed can also be observed for different kinds of tasks or environments with continuous state spaces.

This work aims to test how well the biologically inspired state representations scale to different kinds of tasks and environments with continuous state space. Further, we test, if the representations can be extended for larger and more complex inputs like images. We test the representations in two different RL models: An artificial model using established RL methods without biological foundation, and a biologically inspired model that uses biologically plausible approaches. This allows further insight, into how well biologically inspired state representations work in models without biological plausibility.

The main novelty of this work is the testing of the biologically inspired state representation with new kinds of tasks and the exploration of ways, to encode more complex inputs like two-dimensional images into SSP and grid cell representation. Specifically, we propose the use of an image regressor and the use of a custom neural network layer to encode images to SSP and grid cell representations.

The main contribution of this work is three-fold:

- Testing of biologically plausible state representations for new tasks.

- Testing of how well biologically plausible state representations work in models without biological foundations.
- Two novel approaches, on how to extend biologically plausible state representations for image inputs.

First, we describe the relevant literature with a focus on the theoretical background for the following sections. Then we explain the architecture of the models, the testing environments, and their input modalities. Next, we present the results of the experiments conducted, followed by a discussion of said results. Finally, we draw some conclusions from the experience of using the different models and state representations as well as the results that were found.

2 Related Work

In this work, we want to test whether a biologically based approach can improve the learning speed of reinforcement learning models. The main focus of this work is to test state representations, that are founded in biology. Besides the state representations, other aspects of the tested models are biologically inspired. We use temporal difference learning and the actor-critic approach, which have similarities to learning processes in the brain. Additionally, we use a spiking and a non-spiking model, to test if the biologically based approach of spiking neural networks can further increase learning speed and robustness. The following section discusses these topics and gives some background information about reinforcement learning.

2.1 Biologically Plausible State Representations

In general RL models, the state representation has no foundation in biology. This section discusses biologically plausible state representations that are based on the Semantic Pointer Architecture (SPA) proposed by Eliasmith [Eli13], which is founded on findings in theoretical neuroscience. The SPA uses semantic pointers, vectors of high-dimensional vector spaces that result from compressed information, to represent anything from low-level visual features to high-level concepts [DE20]. They are called semantic *pointers*, because similar to pointers in computer science they can be dereferenced or queried to access information that they do not carry themselves directly. They are called *semantic* because their vector representations show their relations to other semantic pointers through their distances and similarities. The SPA has been used to create Semantic Pointer Architecture Unified Network (Spaun) a large-scale brain model that can solve several different tasks like image recognition, working memory, or motor control [ESC+12].

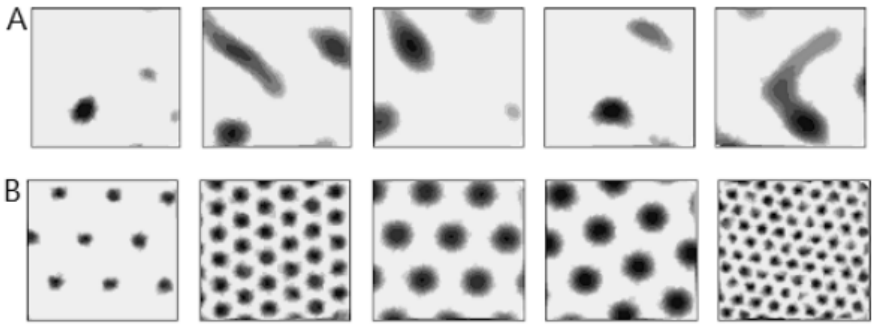


Figure 2.1: Receptive fields of A: Neurons with random encoders and no global order, B: Grid cell representation with a hexagonal grid pattern. Image source: [BSO22]

2.1.1 Spatial Semantic Pointer Representation

As described by Eliasmith [Eli13], the Semantic Pointer Architecture focuses on tasks with discrete structure, making it difficult to use in general RL tasks with continuous state spaces. This is remedied by the SSP representation proposed by Komer et al., which uses the SPA to create an encoding of cognitive structures over continuous spaces [KSVE19].

To encode a natural number $k \in \mathbb{N}$ in a vector binding with d-dimensional vector $B \in \mathbb{R}^d$ the vector is bound to itself $k-1$ times, as can be seen in the following equation:

$$B^k = \underbrace{B \circledast B \circledast ... \circledast B}_{\text{B appears k times}}, k \in \mathbb{N} \tag{2.1}$$

To allow the encoding of a real number$k \in \mathbb{R}$ Komer et al. use fractional binding with circular convolution and choose $B \in \mathbb{R}^d$ as a random semantic pointer of the unit sphere resulting in the following equation:

$$B^k = \mathcal{F}^{-1}(\mathcal{F}(B)^k), k \in \mathbb{R} \tag{2.2}$$

The $\mathcal{F}$ of Equation (2.2) refers to the Fourier transform and $\mathcal{F}(B)^k$ to the element-wise exponentiation. Notable here is, that the multiplications from the exponentiation are circular convolutions because they are done in the Fourier domain.

To extend this representation for multiple dimensions, Komer et al. repeatedly use Equation (2.2) with different semantic pointers for each dimension. A small example of the encoding of coordinates in a two-dimensional space can be seen in the following equation:

$$\mathcal{S} = X^x \circledast Y^y = \mathcal{F}^{-1}(\mathcal{F}(X)^x \cdot \mathcal{F}(Y)^y), (x, y) \in \mathbb{R}^2 \tag{2.3}$$

The encoder weights, used to transform this result from the representational space to neuron activity can be chosen at random. The resulting encoding produces neurons with a random receptive field with no global order, as can be seen in Figure 2.1 A [BSO22].

To summarize, the Spatial Semantic Pointer (SSP) representation allows the encoding of an arbitrary number of real-valued variables into a single fixed-dimensional vector. This allows the use of the SSP representation in all environments, in which the state space consists of any number of continuous one-dimensional variables without modification.

2.1.2 Grid Cell Representation

The grid cell representation is an extension of the SSP representation. It was developed by Dumont and Eliasmith [DE20] to mimic the functionality of the grid cells that can be found in some parts of the brain. The encoding for grid cell representation is calculated the same way as for the SSP representation in Equation (2.3), the only difference being the way the unitary vectors X and Y are chosen. Instead of choosing them randomly, they are instead calculated according to the following equation:

$$X = \sum_{n=0}^{N-1} W_{6N+1}^{-1} \bar{B}_n W_7 X_n \tag{2.4}$$

In the equation W_{6N+1}^{-1} refers to the 6N + 1 × 6N + 1 inverse discrete Fourier transform matrix and W_7 to the 7 × 7 discrete Fourier transform matrix. Further, $\bar{B}_n$ and X_n are a matrix and a vector respectively that are constructed in a certain way. When using the grid cell representation, the encoder weights used to transform from the representational space to neuron activity also have to be chosen in a certain way. for further details refer to the original work of Dumont and Eliasmith [DE20].

Using the grid cell representation results in neurons that are active in a hexagonal grid pattern, as can be seen in Figure 2.1 B. Bartlett et al. [BSO22] compared the learning speed of the SSP and grid cell representation to state representations without foundations in biology like the one-hot representation. They found that using grid cells as state representation performed significantly better than any other state representation in both models that were tested. The SSP representation only performed slightly better in one model and markedly worse in the other model, compared to the other representations without foundations in biology.

2.2 Spiking Neural Networks

Spiking neural networks (SNNs) are more similar to natural neural networks compared to ANN because they simulate neurons in a biologically inspired way. This causes SNNs to encode values over time rather than discrete computation steps. The reason for that is that spiking neurons do not necessarily propagate their values through the network at every computational step. Rather, a spiking neuron only sends an output signal when its action potential reaches a certain threshold caused by an incoming stimulus. This signal then in turn might cause the action potential of other connected neurons to change, causing them to also reach their threshold [GK02].

Spiking Neuron Models

Spiking neuron models can vary from very complex and computationally expensive models with a lot of different parameters like the Hodgkin–Huxley model [HH52] to relatively simple models like Lapicque's Leaky Integrate-And-Fire (LIF) neuron model [BV07] that can be modeled like a simple electrical circuit consisting of a resistor and a capacitor. Even though the LIF neuron model is not as complex, it still allows a realistic simulation of parts of the brain. It was shown for example, that it is possible to simulate a rat's basal ganglia, a decision-making system of mammals, by using LIF neurons [SBE12]. Furthermore, LIF neurons have also been used by Eliasmith et al. [ESC+12] inside the large-scale brain model Spaun, which can solve several different tasks like motor control and memorization of visual inputs.

Other than neuron models that are based on biology, it is also possible to convert established activation functions of regular ANN methods like ReLU or sigmoid into a spiking variant. This can be done using libraries like KerasSpiking[1] or PyTorchSpiking[2], which allow the conversion of regular ANNs of Keras or PyTorch into a spiking model with only a little additional work.

[1] https://www.nengo.ai/keras-spiking/
[2] https://www.nengo.ai/pytorch-spiking/

2.3 Background – Reinforcement Learning

The theoretical foundation of the following section is based on the work of Sutton and Barto [SB18]. RL is an area of machine learning, in which an agent chooses from different possible actions and is either rewarded or penalized, depending on whether it was a good or a bad action to take, in the state the agent was in when choosing this action. The aim of RL is to maximize the future cumulative reward by continually refining its policy, a mapping of the current state of the RL agent, and the action the agent should take in said state. To achieve this the agent has to balance exploration and exploitation. The agent has to explore the environment to search for better actions to take, that promise a higher reward and exploitation because the agent has to act on its experience and choose the best actions that it knows. To do this the agent has to learn either the state-value function or the action-value function, which maps either states or state-value pairs to future cumulative rewards the agent can obtain starting from this state. The action-value function Q_π of given policy π is calculated by simulating the environment and taking actions according to π. If the agent ends up in a goal state S_t and takes action a_1, it receives a reward, which leads to an increase of the action-value function of $Q_\pi(S_t, a_1)$. If the agent reaches state S_t again by taking action a_2 from state S_1, the value of $Q_\pi(S_1, a_2)$ also increases. This is done until the reward has propagated through the state and action spaces, the values of state-action pairs leading to the goal state being higher than state-action pairs that lead away from the goal state. The action-value function can then be used to define a new policy $\pi^* = argmax_a Q(S, a)$, the optimal policy that always uses the action a, which maximizes the future cumulative reward. In more complex cases, the action-value function might not be able to be calculated directly, rather it has to be approximated leading to a policy that gets increasingly better with each iteration but does not necessarily reach the optimal policy [SB18].

2.3.1 Temporal Difference Learning

Sutton and Barto describe temporal-difference (TD) learning as the combination of the sampling of experience like in Monte-Carlo methods with bootstrapping of dynamic programming, the performing of updates of the current estimates based on other estimates. The following paragraph shows how TD(0) is used to calculate the value function of a given policy.

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \tag{2.5}$$

When calculating the value function of some policy π with TD learning, the environment is simulated and actions are taken according to the policy π. Then the estimated value function V of v_π is updated for each non-terminal state S_t occurring during the simulation of the environment according to Equation (2.5). The α of the equation is the learning rate, a number between zero and one that specifies how easy it is for a taken action to influence the value of the value function. The γ refers to the discount value, also a number between zero and one, that specifies how much future rewards should be reduced to prioritize more immediate rewards. The value function is then updated until a terminal state has been reached. Then further episodes can be simulated for the estimated value function V to more accurately approximate the real value function v_π.

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \tag{2.6}$$

The last part inside of the brackets of Equation (2.5) is a value called the TD error (see Equation (2.6)) that measures the difference of the estimated value for state S_t and the more accurate estimate $R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$. Important to note here is that the calculation of the TD error at time t is dependent on the state and the received reward at time $t + 1$, meaning the TD error at time step t can only be calculated at time step $t + 1$ [SB18].

2.3.2 Actor-Critic Methods

The Fundamental concepts of the actor-critic approach were introduced by Barto et al. [BSA83]. This work introduces the idea of using two different modules, an Associative Search Element (ASE), commonly referred to as actor today and an Adaptive Critic Element (ACE), today commonly referred to as critic. The actor is a module, that calculates the policy, and the critic is a module, which calculates the value function of the actor's policy and 'criticizes' the actions the actor takes. The critic uses a TD algorithm to calculate the TD error and uses this error to encourage actions, which leads to states with a higher value compared to the estimated value, and discourage actions, which lead to states with a lower value compared to the estimated value [SB18].

The Advantage Actor-Critic (A2C) algorithm used in this work is a synchronous and deterministic variant of the Asynchronous Advantage Actor-Critic (A3C) proposed by Mnih et al. [MBM+16]. The main difference between Advantage Actor-Critic (A2C) to the regular actor-critic approach is to use the critic to calculate the advantage function rather than the value function. The advantage function is defined as the difference between the action-value function and the value function: $A(s, a) = Q(s, a) - V(s)$. The advantage function essentially determines how better or worse it is to take action a while in state s compared to other available actions.

The actor-critic approach is especially interesting regarding biological plausibility because the two components, the actor and the critic, work similarly to parts of the brain, the dorsal and ventral subdivisions, which are important for reward-based learning. Sutton and Barto also describe, that the TD error being used for the learning for both the actor and the critic is comparable to the properties of biological neurons, where the axons of dopamine-producing neurons also target the dorsal and ventral subdivision.

3 Methods

Previous work has shown that using biologically plausible state representations can increase the learning speed of reinforcement learning models in a simple grid-based two-dimensional navigation task [BSO22] [GD11]. In this work, we test how well this approach scales to different and more complex tasks and compare the results to a model using standard deep learning approaches. In detail, we use two different models for testing: An artificial model, using established reinforcement learning methods, and a biological model, using biologically based methods like spiking neurons. For testing, we use three different and more complex environments, with a continuous state space. Further, we also test the models with image inputs instead of the state variables for an even more complex and difficult to learn task. To that end, we explore different possibilities to expand the biologically plausible state representations for use with images.

3.1 Artificial Model

The first RL model implemented to test the biologically plausible state representations in this work is an A2C. The A2C is a synchronous version of the Asynchronous Advantage Actor-Critic (A3C) proposed by Mnih et al. [MBM+16]. The main difference to the regular actor-critic approach is to use the critic to calculate the advantage function rather than the value function. The advantage function is defined as the difference between the action-value function and the value function: $A(s, a) = Q(s, a) - V(s)$. The advantage function essentially determines how better or worse it is to take action a while in state s compared to other available actions. The artificial model will further be abbreviated with *Art-Critic*: Artificial Advantage Actor-Critic.

While *Art-Critic* uses an actor-critic approach, which has been hypothesized to be biologically plausible (see Section 2.3.2), it uses different established RL methods and neural network layers which are not biologically plausible. Some examples of this are the use of backpropagation in the learning process, the use of batch-normalization layers, and the use of non-spiking neurons with the ReLU activation function. Because *Art-Critic* works like a conventional RL model it allows further examination, of whether the biologically plausible state representations can work well in models without biological foundations. The basic structure of *Art-Critic* can be seen in Figure 3.1. The observed state of the environment is mapped to the used state representation, which will then be used as the input for the actor and the critic network. The actor network calculates the action preference for each action, while the critic calculates the value function for the current state, both of which are then used in conjunction with the observed reward for the TD update. The TD update adjusts the weights for both networks and uses Adam as the optimizer.

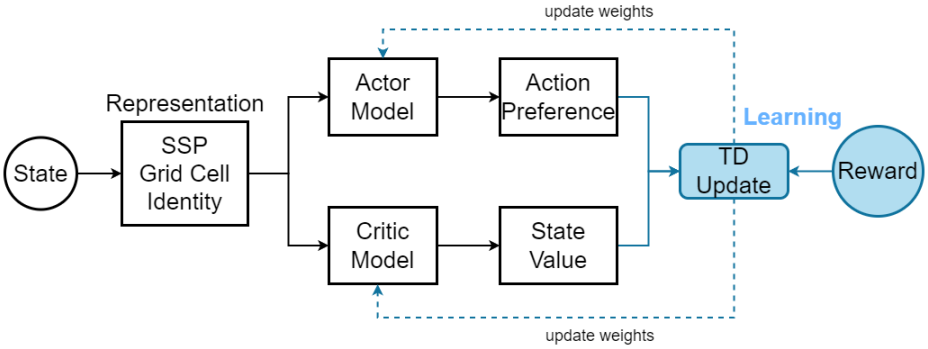


Figure 3.1: Overview of the artificial PyTorch model (*Art-Critic*).

3.2 Biologically plausible Model

For the biologically plausible model, we use the implementation of Bartlett et al. [BSO22] as our base and adapt it to our use case. The model will further be referred to as *Bio-Mod*: Biologically plausible model. The basic overview of the model can be seen in Figure 3.2. Identical to *Art-Critic*, *Bio-Mod* maps the observed state of the environment to a vector representation, which is then used as input for the model. The weights $w_{encoders}$ refer to the encoders, that are used to transform the input for the neurons from the representational space to neuron activity, while $w_{decoders}$ refers to the learned weights, that are updated during training. *Bio-Mod* only consists of a single layer of 4000 spiking LIF neurons. The learning is done by directly using the TD error to update the weights of the only layer, allowing learning without the need for backpropagation. Bartlett et al. implemented and tested different learning rules like TD(0) and TD(λ). For simplicity's sake and to reduce the number of training runs we only use the TD(0) learning rule. For further details regarding the implementation of the biologically plausible model we refer the reader to the repository of the original implementation[1].

Hyperparameter	α_{actor}	α_{critic}	γ	Entropy
Art-Critic	0.00025	0.00025	0.98	0.1

Hyperparameter	α	γ_{action}	γ_{value}	# Neurons	Sparsity	Δt
Bio-Mod	0.8	0.75	0.94	4000	0.35	0.05s

Table 3.1: Hyperparameters of the models *Art-Critic* and *Bio-Mod* and example of chosen parameters for the Cart Pole environment where the state is used directly as input.

[1] https://github.com/maddybartlett/Bio_Based_Reps_for_RL, last accessed 14.07.2023

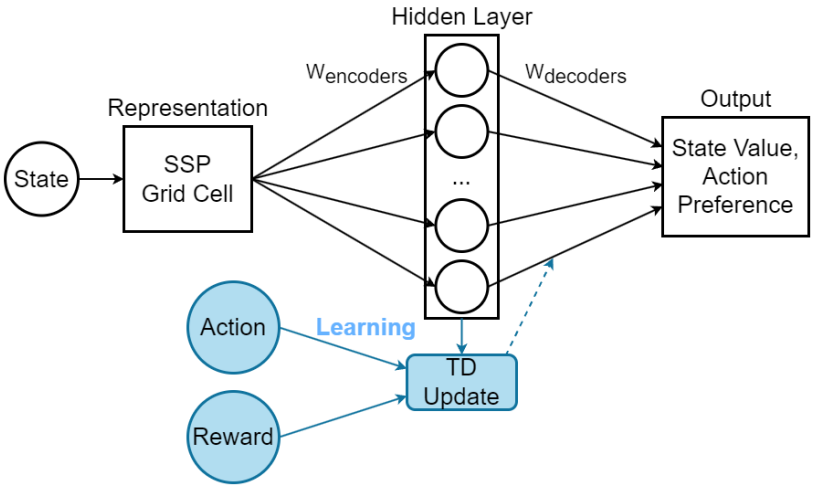


Figure 3.2: Overview of the biologically plausible Nengo model adapted from [BSO22] (*Bio-Mod*).

The hyperparameters for *Bio-Mod* consist of the learning rate α, the discount factors for the action and the value function $\gamma_{action}, \gamma_{value}$, the number of LIF neurons in the hidden layer, the sparsity, that determines where the activity of a neuron is zero and the length of a time step of the spiking neural network in seconds Δt.

3.3 Observation As Input

We test our model in two different cases. In the first and the more easily learnable case, we use the state as observed from the environment to calculate the encoding of the representation. The state usually contains variables that are very helpful to learn the task effectively like the position of the agent or its current velocity. In the second and more complex case, we instead use an image of the environment to encode the state. That means the model has to learn the information it was given in the previous case directly by itself.

Bartlett et al. [BSO22] used one-hot encoding and a tabular look-up method as a baseline in comparison to the biologically based state representations. However, we test the representations in environments with continuous state spaces, making discrete representations a bad fit. Instead, we use the state observed from the environment directly and use the result as our baseline.

For observation as input, *Art-Critic* consists of a simple neural network of three linear layers with ReLU activation function. The input of the first layer has the size of the state representation and 128 output channels. The second layer has 256 output channels and the third layer has either one output, for the critic network, or an output for each action of the environment for the actor. The actor additionally contains a (logarithmic) softmax layer, to calculate the action preference of the agent. For *Art-Critic* there are a total of 34,178 learnable parameters for the Cart Pole, 34,691 for the Acrobot, and 35,204 learnable parameters for the Lunar Lander environment. The number of learnable parameters for the critic network is slightly lower, with 33,921, 34,177, and 34,433 respectively. In comparison *Bio-Mod* uses only a single layer with 4000 spiking LIF neurons.

3.4 Image as Input

Using the image as the input for the model instead of the state poses various challenges. For one, the amount of raw data increases significantly from four to eight floating point numbers to a 400 by 600 RGB image. The image is not only significantly larger but also has very low information density. There is only a very small part of the image, which contains information relevant to learning a better policy. To reduce the amount of raw data, we preprocess the image every frame before passing it to the model. More specifically we use the grey-scale of the image and sample down the image size to 84 by 84. For the downsizing of the image, we use OpenCV library's inter-area interpolation method to preserve structures in the image, with only a few pixels in width, like the flagpoles in the Lunar Lander environment, that can be seen in Figure 3.3c.

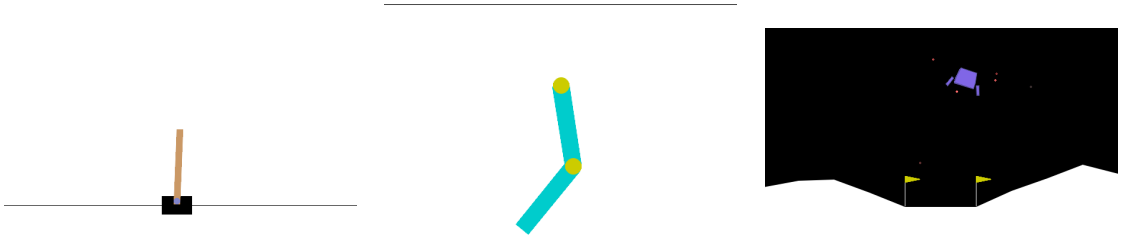
Another challenge when using images as input is the missing information, that cannot be inferred from a single image. Information, like the current speed, the trajectory of an object, or angular velocity cannot be determined from a single static image. Because of that we instead batch the last four frames of the environment and use the batch as input for the models, allowing the agent to learn such information by detecting the change between frames. More specifically, we use the difference between the previous and the current image to determine what changed between frames. Although it is a very rudimentary approach to detect the optical flow, the motion of objects in a visual scene, it still allows the detection of differences between frames.

Thirdly, translating a two-dimensional image into a biologically plausible state representation poses an additional challenge. In the following we will take a look at the two main approaches tested, none of which led to satisfactory results:

- Training an image regressor that uses the image as input and predicts the corresponding state given from the environment. The training data for the regressor was generated by simulating the environment and taking actions randomly. From this seeded data, only a shuffled subset was used, to prevent the regressor from learning patterns from the sequence of the input. The training set consisted of 25,000 images and the validation set of 5,000 images. While the regressor was very accurate for some state variables like the current position, it had trouble consistently predicting variables like the current speed accurately. Different architectures for the regressor were tested using AutoKeras's ImageRegressor[2]. The best performing of 120 tested regressors consists of six convolutional layers and one dense layer with a total of 1.185.988 trainable parameters. This model achieved a mean squared error of 0.0341 on the training data and an error of 0.0386 on the validation data. This approach was tested for both *Bio-Mod* and *Art-Critic*. While this regressor is not a SNN and therefore not biologically plausible, it would be possible to convert the Keras into a SNN, by for example using the NengoDL library [3]. For simplicity, we use the same regressor in both cases.
- Converting the result of the convolution layers to the state representation. As converting the flattened image directly to the state representations did not seem to work very well, a convoluted image has the potential to lead to better results, as every pixel contains information about other pixels around it. This can be interpreted as an increase in the size of the receptive fields of the resulting encoding of the biologically plausible state representations, as now

[2] https://autokeras.com/image_regressor/

[3] https://www.nengo.ai/nengo-dl/examples/keras-to-snn.html



(a) Cart Pole environment **(b)** Acrobot environment **(c)** Lunar Lander environment

Figure 3.3: Example image of a single frame of each tested environment.

every semantic pointer also contains information on neighboring pixels. Because of the large number of values produced by the convolutions, and the need to constantly calculate the Fourier transform and inverse Fourier transform, this was one of the slowest implementations regarding training time needed. For a training run of the Cart Pole environment with 1000 episodes, this approach needed about 27 and a half hours of training time. Considering the model did not learn very well, this time could further be double or tripled if it learned a better policy, because that would increase the length of each episode. This approach was only tested for *Art-Critic*.

When using images as input, *Art-Critic* consists of three convolutional layers with batch-normalization and ReLU activation between each layer. The layers have 32, 64, and 64 input channels respectively. The result is flattened into a linear layer with 512 input channels and again for the actor network followed by a (logarithmic) softmax layer.

Further, when using a biologically plausible state representation, an additional layer is used between the flattened and the linear layer, which turns the output of the convolutional layers to that representation.

3.5 Testing Environments

The environments that were used for testing the RL models are from the Gymnasium library[4]. A total of three different environments were used for testing, namely the Cart Pole-, Acrobot- and the Lunar Lander environment. We use Cart Pole as a relatively simple environment to validate the functionality of the models, Acrobot to test the models on an environment with sparse reward distribution, and Lunar Lander because it is significantly more complex and difficult to learn than the other environments. The technical details of the environments can be seen in Table 3.2

[4] https://gymnasium.farama.org/

Cart Pole Environment

As depicted in Figure 3.3a on the preceding page, the Cart Pole environment consists of a cart represented by a black square, which is connected to a pole by a joint. The goal of the agent is to balance the pole on top of the cart as long as possible by moving the cart either left or right on a frictionless track. The agent is given a reward of one for every simulation step including the step which terminates the episode. The episode is truncated after 500 simulation steps, meaning an agent with an optimal policy achieves a total reward of 500 in every episode. The state contains four values describing the cart's position and velocity and the pole's angle and angular velocity. The Cart Pole environment's simplicity makes it a fitting candidate to validate the correctness of a RL model.

Acrobot Environment

Figure 3.3b on the previous page shows the Acrobot environment. The environment is comprised of two blue links which are connected by an actuated joint. The upper-link further is connected to a fixed joint, holding the links in place. The goal of the agent in this environment is to reach a certain height with one of the links by applying torque to the actuated joint. The agent is allotted a reward of minus one every simulation step, except when reaching the target height, where it is awarded a reward of zero. The state for the Acrobot environment consists of six values describing the Sine and Cosine of the angle of the first joint, as well as the relative angle of the first link and their angular velocity. The sparse rewards of this environment can make it difficult to learn a good policy effectively.

Lunar Lander Environment

Figure 3.3c on the preceding page shows the Lunar Lander environment. The environment is composed of a mountainous terrain and two flagpoles marking the landing area for the lunar lander. The terrain around the landing area changes every episode. The reward consists of multiple different factors like the distance to the landing area or whether the lander is tilted, encouraging the agent to land in the landing area as softly as possible. In Lunar Lander, the state consists of eight values:

Environment	# States	# Actions	Termination	Truncation
Cart Pole	4	2	• Pole angle greater than $\pm$ 12° • Cart reaches screen border	500 steps
Acrobot	6	3	• The target height is reached	500 steps
Lunar Lander	8	4	• The lander crashes • The lander is outside of the viewport • The lander landed safely	1000 steps

Table 3.2: Technical details of the different testing environments.

Two for the location of the lander and its linear velocity in the x- and y-axis, one for the angle and the angular velocity of the lander, and two booleans for each of the legs, if they are in contact with the ground. The Lunar Lander environment is the most complex of the environments tested in this work.

4 Results

The result chapter is split into two sections. The first section discusses the results, where the observation of the environment is used as the input for the model directly. The second section covers the results, for which an image of the current state of the environment is the input for the model. Using images as input significantly increases the difficulty of learning and training time needed, since the model has to abstract the information that was previously given directly by itself.

The figures in this section show the running mean reward of the last 100 episodes to represent the current performance of the model and to make changes in performance more notable. The hyperparameters of the models were chosen by a manual search in most cases. For model and representation configurations, where a manual search led to no good learning agent, a Bayesian search of the parameter space was conducted, and the best performing configuration was chosen.

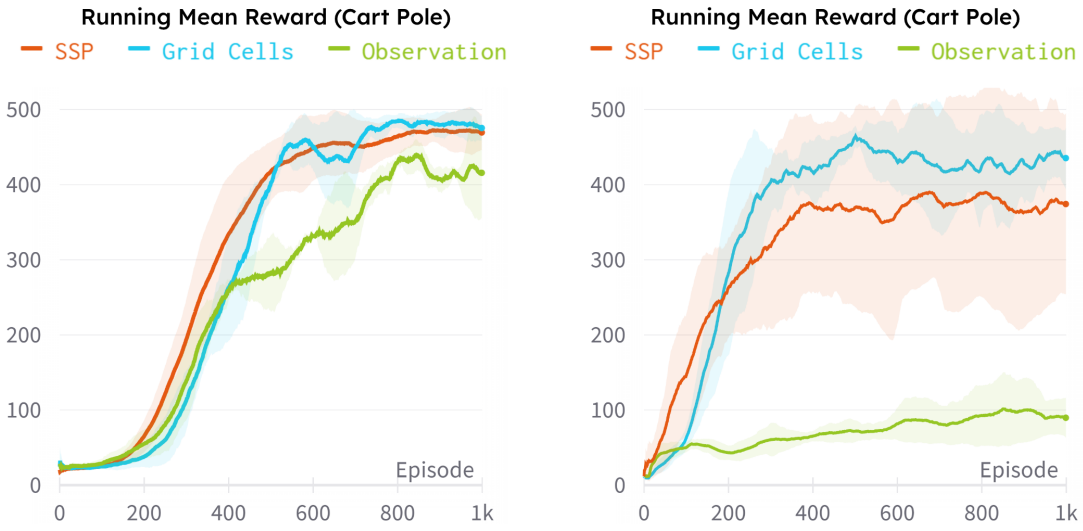
This section shows the results of the two tested models: *Art-Critic*, a model using regular approaches to RL with no foundation in biology, and *Bio-Mod*, a model with biological plausibility. The models were tested in the three different environments described in Section 3.5

4.1 Observation As Input

This section presents the results, where the observation from the environment is used as the input for the model, either directly for the baseline or encoded in the form of SSPs or grid cells. The observation consists of four to eight values representing different information between the three different testing environments. The values represent information like the current position, speed, and angular velocity in most cases. For a more detailed description of the environments and their corresponding state space refer to Section 3.5

Cart Pole Environment

The results for the Cart Pole environment can be seen in Figure 4.1. In the Cart Pole environment, both the SSP and grid cell representation outperform the baseline of the models. Especially so for (**b**) *Bio-Mod* where the baseline is significantly slower than not only the biologically plausible state representations but also the baseline of (**a**) *Art-Critic*. SSP and grid cell representation perform very similarly, with grid cells being slightly better for *Bio-Mod*. That is however only the case, because of a single badly performing run for the SSP representation, which can be seen by the significantly bigger standard deviation. With that run omitted, the SSP representation even outperforms grid cells slightly in the later half of the training time.



(a) *Art-Critic* **(b)** *Bio-Mod*

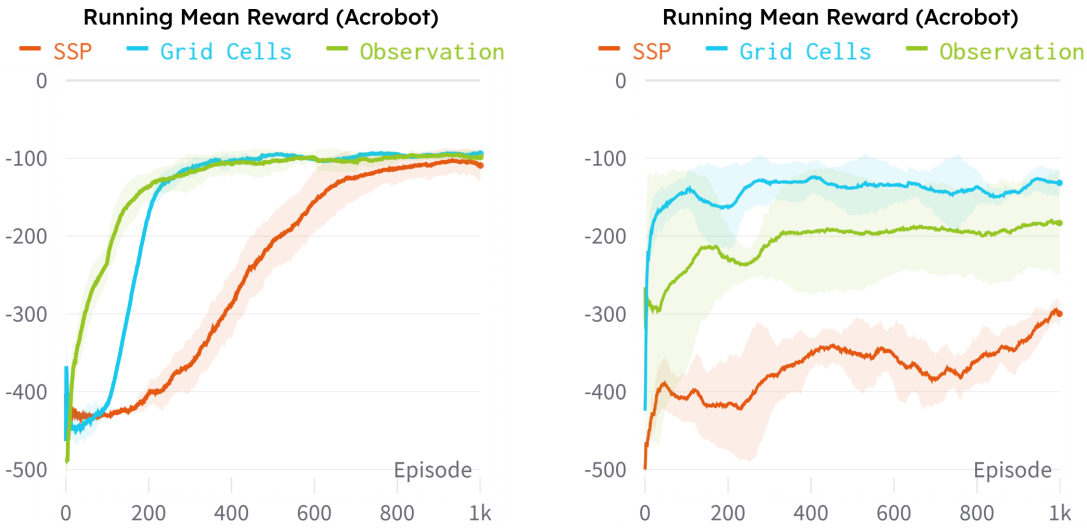
Figure 4.1: Running mean reward and standard deviation of the last 100 episodes for the Cart Pole environment with the observation of the environment used as input directly.

Except for using the observation with *Bio-Mod*, the training process proceeds very similarly between *Bio-Mod* and *Art-Critic*. *Bio-Mod* seems to learn a bit faster at the beginning of the training period but does not learn as good a policy in the end compared to *Art-Critic*. The bad performance of the observation for *Bio-Mod* does not seem to be a problem of the model, as the model still worked well in the other environments. Further, it is also unlikely a problem with the parameters of the model, because even a Bayesian sweep over the parameter space did not reveal any better-performing configurations. The range of tested values for each parameter are as follows: Learning rate and the discount factors were tested in a range of 0.2 to 0.99, sparsity from 0.05 to 0.75 and the simulation time per step was chosen discretely from 0.05, 0.075, and 0.1 seconds. The number of neurons was set to 4000. For further details of the parameters refer to Section 3.2 The Bayesian sweep was conducted using Weights & Biases (W&B)[1] and contained a total of 560 tested configurations.

Acrobot Environment

For the Acrobot environment, using the observation directly performed surprisingly well for both models, as can be seen in Figure 4.2. For (**a**) *Art-Critic* using direct observations is initially the fastest but is later tied with the grid cell representation. For (**b**) *Bio-Mod* using the observation is moderately slower than grid cells. SSP representation learned significantly slower for both models. While *Art-Critic* using SSP representation learned a policy comparable to the other representations, *Bio-Mod* remained worse than the other representations at the end of training. The slower learning in both models using SSP representation suggests, that the SSP representation might not be a suitable use case for the Acrobot environment. A reason for that might be, that the random receptive patterns

[1] https://wandb.ai/wandb_fc/articles/reports/Bayesian-Hyperparameter-Optimization-A-Primer--Vmlldzo1NDQyNzcw



(a) *Art-Critic* **(b)** *Bio-Mod*

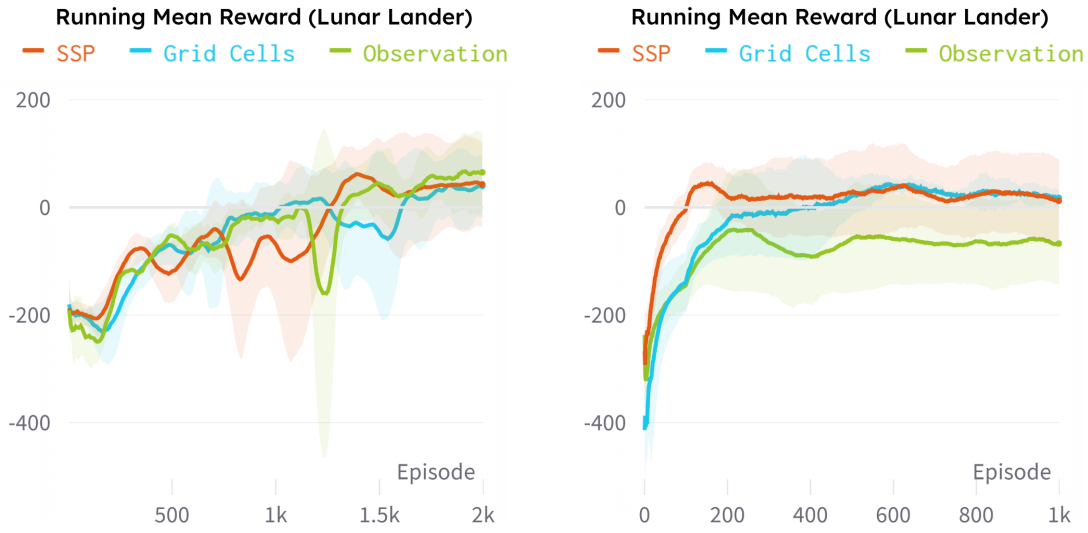
Figure 4.2: Running mean reward and standard deviation of the last 100 episodes for the Acrobot environment with the observation of the environment used as input directly.

of the SSP representation make it harder to learn the Acrobot environment. Similar to the results for the Cart Pole environment, *Bio-Mod* seems to learn very fast at the beginning of training of the training, but does not learn as good a policy as *Art-Critic* at the end of training. In this case, it seems to be even more extreme, as *Bio-Mod* needs only very few episodes (10-20) to learn a good policy. After the initial fast learning, *Bio-Mod* fluctuates around the region it reached during the beginning. *Art-Critic* also learns very fast but still takes about 300-400 episodes of learning to reach its maximum for both the SSP and grid cell representations.

Lunar Lander Environment

The training process of the Lunar Lander environment was very unstable for *Art-Critic*, as is shown in Figure 4.3. While the running mean reward is trending upward during training, there are episodes where the model drastically worsens for a while, before recovering and learning a better policy. A reason for that might be a change in strategy of the model, that results in a temporary worsening of the model. Another reason could be, that slight changes in the layer weights can drastically change the reward in the Lunar Lander environment, because of the way rewards are calculated. For example, when the lander crashes, the model is penalized with a reward of -100, while it is rewarded with a reward of 100 if the lander lands successfully. If because of some change in strategy the lander now crashes more often, it can lead to a sudden decrease in the total episode reward for some time. Even though the training performance of *Art-Critic* was very volatile for all representations, aggregated over the whole training process all representations performed very similarly.

Bio-Mod did not have problems with stability during training, but had problems similar to what we observed in the other testing environments: While it had a very promising and fast initial learning speed, it stopped improving after about 200 episodes. Because of that *Art-Critic* learned a better



(a) *Art-Critic* **(b)** *Bio-Mod*

Figure 4.3: Running mean reward and standard deviation of the last 100 episodes for the Lunar Lander environment with the observation of the environment used as input directly.

policy at the end of training, although it has to be noted, that *Art-Critic* was trained an additional 1000 episodes. But since *Bio-Mod* stopped improving after 200 episodes, it is reasonable to assume 1000 additional training episodes would not have led to a different result. The models that stop learning, often do so because they are caught in a local maximum, where the model learned the strategy to take no action in most cases. This causes the agent to fly in the general direction of the goal, which allows the agent to gain a higher reward than using actions incorrectly, which could cause the lander to crash and be penalized with a negative reward. Overall, the grid cell and SSP representations perform slightly better than direct observation input for *Bio-Mod*.

Summary

The results suggest that the performance of the state representation depends on various factors. As shown in Figures 4.1, 4.2, and 4.3, every representation sometimes performed very well and sometimes worse than the other state representations, depending on the model architecture used and the testing environment. Even though grid cell representation is sometimes slower than the other representation at the start of training, it is always better or matches the performance of the best-performing representation by the end of training. This suggests, that using grid cell state representations is generally a good choice when training for a reasonable amount of time. The variance between runs is generally low, implying high robustness of the approach. There are some exceptions to this, like a run with the SSP representation with *Bio-Mod* or the problems of *Art-Critic* with the Lunar Lander environment. The results mainly support the results of the previous work by Bartlett et al. [BSO22] and Gustafson and Daw [GD11], where biologically inspired methods like grid cells and temporal-difference learning have been found to increase learning speed in a two-dimensional navigation task, but the effect does not seem to be as pronounced in the here tested environments.

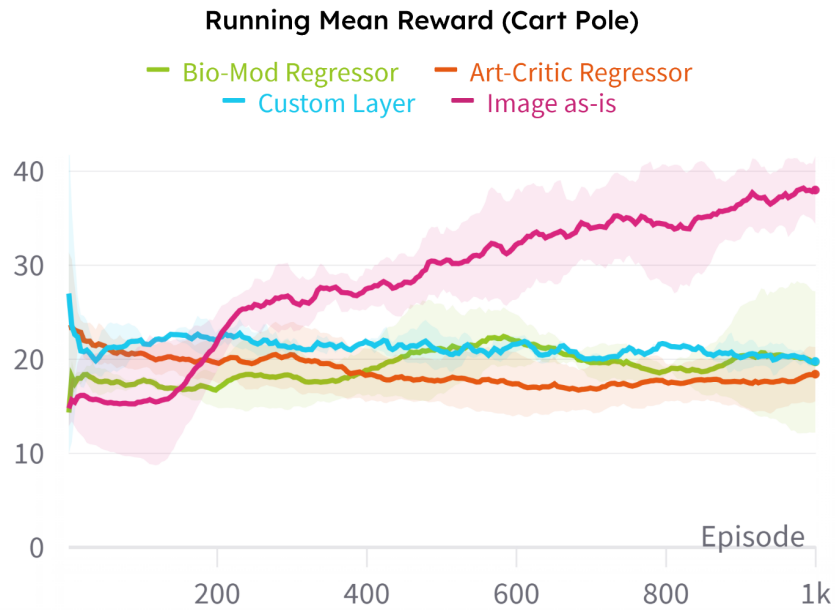


Figure 4.4: Running mean reward and standard deviation of the last 100 episodes for the Cart Pole environment with an image of the environment used as input. Custom Layer and Image as-is refer to *Art-Critic*.

Bio-Mod always learned very fast at the beginning of training but stagnated soon after. The reason for the stagnation in learning might be, because only a single layer of neurons is used for that model, restricting its capabilities. Another possibility is, that the use of spiking neurons leads to the stagnation in learning. When using *Bio-Mod* with a non-spiking rectified linear neuron model (comparable to using ReLU activation function), the problem of the stagnation of learning did not occur, while still learning fast in the beginning. This was only tested for the Cart Pole environment, meaning it is not certain it would also perform as well in other environments. A noteworthy result from using the rectified linear neuron model is that using it in conjunction with the grid cell representation allowed *Bio-Mod* to learn the optimal policy, where it reached the maximum episode length of 500 simulation steps in the running mean of the last 100 episodes. While it was not rare for the models to regularly reach the maximum episode length, no other model could reach it consistently to make the running mean reward of the episode reach 500. An ideal approach would combine both the fast learning speed at the beginning of *Bio-Mod* and the ability of *Art-Critic* to constantly improve the performance of the model during training. While using ReLU neurons with *Bio-Mod* makes the model lose biological plausibility, it combines both the mentioned properties, i.e. learning fast and continuous improvement.

4.2 Image as Input

For image as input, we test the approaches of converting the two-dimensional image to a biologically plausible state representation. The approaches we compare are using an image regressor, that predicts the state from the image and using a custom neural network layer, which converts the

flattened output from the convolutional layers to the state representation. The image regressor approach was tested for both *Art-Critic* and *Bio-Mod*, while the custom layer was only tested for *Art-Critic*. The results of these approaches are compared to the baseline, where the image from the environment is used as-is as the input for *Art-Critic*. In this case, as described in Section 3.4 *Art-Critic* also uses convolutional layers. Because of time constraints, we limit our experiments to the Cart Pole environment.

The results of the different approaches can be seen in Figure 4.4. The results suggest, that the biologically plausible state representations do not scale very well on large or complex inputs. The baseline, where the image is used directly in a convolutional neural network worked better than any of the tested approaches, which convert the input to a biologically plausible state representation. All of the tested approaches have similar performance and as training progresses even slightly trend downwards.

For the image regressor approach, it is likely a problem with the accuracy of the regressor. As stated in Section 3.4, the regressor has problems learning state variables, which pertain to movement, such as speed or angular velocity. While it is quite accurate for other variables, such as the current position, this information is not as valuable in the Cart Pole environment, as the speed and current angle of the pole are more critical in most cases. Incidentally, the current position would only be important in case the cart is in danger of leaving the viewport.

The bad performance when using the custom layer is likely caused by the large number of parameters produced by the convolution. The input for the custom layer consists of 3136 values with the configuration used described in Section 3.4. With such a large number of values, we have to choose a very small dimensionality for the semantic pointer, to prevent running out of memory and allowing reasonably fast calculations. The tests were made with 32-dimensional SSPs, which is significantly less than other experiments, where 512 dimensions were used. With a small number of dimensions and a high amount of SSPs, a lot of the semantic pointers inevitably have high cosine similarity, making it difficult to distinguish between them. Additionally to the high memory demand, the approach is also very computationally demanding, as the training took over 27 hours to complete a single run, about five to six times longer than the next slowest approach. Such a long training time for so few simulated steps makes this approach not very suitable to use unless a more efficient way of calculating the representation is found. Even then, it could be difficult to use with higher dimensions because of the memory demand: 3126 SSPs with 512 float or double variables with 4 or 8 Byte would sum up to 6.4GB and 12.8GB, respectively. While there are GPUs with enough VRAM for using 512 dimensions, this might still not be enough, considering we also use 512 dimensions for four- to eight-dimensional inputs. Further, keeping in mind that the images used for the simple Cart Pole environment are relatively small with a size of 84 by 84, this approach does not seem to be scalable.

5 Discussion

The goal of this thesis is to test, if a biologically plausible approach to reinforcement learning, with a main focus on biologically based state representations, leads to a faster learning speed compared to established RL approaches. The results suggest that using biologically based state representations, especially grid cells, generally leads to an increase in learning speed or at least matches the performance of other tested approaches. Although there are some exceptions, where biologically plausible representations learn at a slower rate, i.e. we found that biological-based approaches do not always have a positive or neutral effect on learning speed. For example, the use of spiking LIF neurons led to a decrease in learning speed.

While the use of spiking neurons does not necessarily lead to worse performance, it increases the complexity of the model and introduces a time component into the neural network, that can make it difficult to achieve performance similar to a model using non-spiking neurons. We therefore posit, that if the goal is not to create a biologically realistic model but to achieve a faster learning speed, it might be better to forgo some biological plausibility for a simpler and faster approach, e.g. by using artificial ReLU activations. The results further show that biologically plausible state representations do not extend very well to complex or large inputs like images, causing slower learning speed and an overall increase in training time needed. Two main approaches for this were explored, both of which performed worse compared to the baseline using a simple convolutional neural network.

The results presented in this work mainly support the results of Bartlett et al. [BSO22] and Gustafson and Daw [GD11], where biologically inspired methods like grid cells and temporal-difference learning have been found to increase learning speed in a two-dimensional navigation task. The effect of the biologically inspired methods used in this work does not seem to be as pronounced as in the works of Bartlett et al. and Gustafson and Daw. The reason for that difference can mainly be attributed to the use of tasks other than a navigation task in a different environment. Evidence suggests grid cells are well suited to encode spatial information [BSO22], which indicates that they are a good choice as a representation to encode location data, which is useful in a navigation task. However, in the tested environments other information was more valuable. This would also explain the not as pronounced effect of using grid cells in our different kinds of tasks.

The approaches used to extend the biologically based state representations to image inputs explored in this work did not work as well as expected. However, it still holds promise for future work to explore new and extend upon the presented approaches. The custom layer approach, as tested in this work, is not feasible to use as-is, but the image regressor could work very well with a more accurate regressor model. Since the regressor had trouble predicting values pertaining to movement, one extending idea would be incorporating optical flow estimation more concretely. If determining the exact value is a problem, another possible addition could be to change the regression problem into a classification problem, by disregarding the actual values and binning them into a discrete structure. For the Cart Pole environment, it would for example be possible to use a variable, that determines whether the cart is moving left or right, rather than the actual speed it is currently traveling at.

Other information, that a regressor can learn well, like the current position or values that have to be available as continuous values can still be determined using a regressor. Finally, adding the last predicted state and the taken action to the input of the regressor could improve accuracy. It could allow the model to learn the connection between a taken action and the effect it has on the state, e.g. acceleration to the left while moving right, decreases the speed value.

While this work tested different tasks and environments with different model architectures, it is still limited to a very small sample of tested models, configurations, and environments/tasks. The biologically based representations performed very well with the tested environments and models of this work but did not always perform the best. This could mean that environment and model configuration might not be uncommon, where the biologically based state representations perform worse than other representations, similar to the results of Figure 4.2b, where the SSP representation performed worse than just using the state directly. This could make it more ambiguous if the difference in performance can be traced back to the choice of the state representation.

6 Conclusion

In this work, the performance of biologically plausible state representations was tested in a biologically based and in an artificial RL model. We explored two ways how the biologically plausible state representations could be extended for use with larger and more complex inputs like images. We found, that biologically plausible representations generally lead to a faster learning speed, especially for the grid cell representation. Further, the tested approaches to extend the representations for use with image inputs, led to worse performance compared to using the images as input directly, meaning further work is required to effectively use them in this use case.

Spiking neural networks (SNNs) are very interesting from a research standpoint. They can be used to simulate biological neural circuits and build realistic brain models, to gain more insight into the brain and how learning works. While this is very interesting to learn more about how the brain works, it is often not necessary in standard RL tasks to simulate a brain realistically. Most people are likely more interested in efficiently implementing a well-performing model to solve the task they work on. While SNNs can technically be used in the same way as artificial neural networks, it is often harder and more work necessary to achieve similar performance with SNNs compared to ANNs. That means it is reasonable to forgo some of the biological plausibility for a simpler and faster approach. The biologically based approaches, which have been found to increase the performance can then still be used, like possibly the grid cell state representation.

While the biologically plausible state representations did not extend well to image inputs, they worked well on environments, in which state space consists of a collection of floating point numbers. Consequently, we suggest established neural network libraries like PyTorch and Tensorflow/Keras to implement a version of SSP and grid cell representations. This could allow the use of biologically plausible methods without the need for extensive background knowledge in neuroscience and the math behind it, allowing a further examination of the evaluated advantages of biologically inspired state representations.

Bibliography

[BSA83] A. G. Barto, R. S. Sutton, C. W. Anderson. "Neuronlike adaptive elements that can solve difficult learning control problems". In: *IEEE transactions on systems, man, and cybernetics* 5 (1983), pp. 834–846. DOI: 10.7551/mitpress/4943.003.0033 (cit. on p. 21).

[BSO22] M. Bartlett, T. C. Stewart, J. Orchard. "Biologically-based neural representations enable fast online shallow reinforcement learning". In: *Proceedings of the Annual Meeting of the Cognitive Science Society*. Vol. 44. 2022. URL: https://escholarship.org/content/qt49v0x3rz/qt49v0x3rz.pdf (cit. on pp. 15, 17–19, 23–25, 34, 37).

[BV07] N. Brunel, M. C. Van Rossum. "Lapicque's 1907 paper: from frogs to integrate-and-fire". In: *Biological cybernetics* 97.5-6 (2007), pp. 337–339. URL: https://www.researchgate.net/profile/Mark-Van-Rossum/publication/5876908_Lapicque%27s_1907_paper_From_frogs_to_integrate-and-fire/links/0fcfd50d0573111697000000/Lapicques-1907-paper-From-frogs-to-integrate-and-fire.pdf (cit. on p. 19).

[DE20] N. Dumont, C. Eliasmith. "Accurate representation for spatial cognition using grid cells." In: *CogSci*. 2020. URL: https://www.cognitivesciencesociety.org/cogsci20/papers/0562/0562.pdf (cit. on pp. 15, 17–19).

[Eli13] C. Eliasmith. *How to build a brain: A neural architecture for biological cognition*. OUP USA, 2013. DOI: 10.1093/acprof:oso/9780199794546.001.0001 (cit. on pp. 17, 18).

[ESC+12] C. Eliasmith, T. C. Stewart, X. Choo, T. Bekolay, T. DeWolf, Y. Tang, D. Rasmussen. "A large-scale model of the functioning brain". In: *science* 338.6111 (2012), pp. 1202–1205. DOI: 10.1126/science.1225266 (cit. on pp. 17, 19).

[GD11] N. J. Gustafson, N. D. Daw. "Grid Cells, Place Cells, and Geodesic Generalization for Spatial Reinforcement Learning". In: *PLOS Computational Biology* 7.10 (Oct. 2011), pp. 1–14. DOI: 10.1371/journal.pcbi.1002235. URL: https://doi.org/10.1371/journal.pcbi.1002235 (cit. on pp. 15, 23, 34, 37).

[GK02] W. Gerstner, W. M. Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge university press, 2002. DOI: 10.1017/CBO9780511815706. URL: https://doi.org/10.1017/CBO9780511815706 (cit. on p. 19).

[HH52] A. L. Hodgkin, A. F. Huxley. "A quantitative description of membrane current and its application to conduction and excitation in nerve". In: *The Journal of physiology* 117.4 (1952), p. 500. DOI: 10.1007/BF02459568 (cit. on p. 19).

[KSVE19] B. Komer, T. C. Stewart, A. Voelker, C. Eliasmith. "A neural representation of continuous space using fractional binding." In: *CogSci*. 2019, pp. 2038–2043. URL: https://www.researchgate.net/publication/337984933_A_neural_representation_of_continuous_space_using_fractional_binding (cit. on pp. 15, 18).

[MBM+16] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu. "Asynchronous methods for deep reinforcement learning". In: *International conference on machine learning*. PMLR. 2016, pp. 1928–1937. URL: https://proceedings.mlr.press/v48/mniha16.pdf (cit. on pp. 21, 23).

[SB18] R. S. Sutton, A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018. DOI: 10.1109/tnn.1998.712192 (cit. on pp. 15, 20, 21).

[SBE12] T. C. Stewart, T. Bekolay, C. Eliasmith. "Learning to select actions with spiking neurons in the basal ganglia". In: *Frontiers in neuroscience* 6 (2012), p. 2. URL: https://www.frontiersin.org/articles/10.3389/fnins.2012.00002/full (cit. on p. 19).

Declaration

I hereby declare that the work presented in this thesis is entirely my own. I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted hard copies.

---

place, date, signature