Biologically Plausible Deep Reinforcement Learning

Darrien McKenzie
Department of Computer Science
Missouri University of Science and Technology

Abstract

Biological neurons communicate primarily via a spiking process. Recurrently connected spiking neural networks (RSNNs) more realistically model the brain, compared to their non-spiking counterparts. It is of great interest to discover a biologically realistic learning rule to achieve optimal levels of performance on machine learning tasks. Experimental data describe a phenomenon known as spike-timing-dependent-plasticity (STDP), which integrates local firing coincidences between neurons to learn. STDP is believed to underlie memory formation and storage within the brain. When a reward signal modulates STDP, it enables forming associative memories via operant conditioning. Neuromodulators like dopamine operate similarly in the brain. We employ processes like synaptic scaling to support R-STDP in large, unstructured RSNNs as a means to produce an agent that achieves adequate performance on reinforcement learning tasks.

1 Motivation

Deep learning and the usage of neural networks have seen usage in many different fields over the years due to their universal ability to generalize and approximate any arbitrary function–even if said function is unknown to the experimenter. This ability makes neural networks able to recognize patterns that involve massive amounts of data, which can then be utilized to make accurate predictions and enhance general decision making. However, the biological plausibility of artificial neural networks has seen criticism since their inception due to their neglect of neuroscientific principles. Actual neurons communicate through spikes, which are essentially discrete signals in time that occur on a timescale of milliseconds, which is in contrast to artificial neural networks that communicate by transforming and sending numerical values down to neighboring layers. The backpropagation algorithm in particular, despite it being the primary reason neural networks are so widely used, is not at all biologically plausible, as it is an operation that requires a certain structure and global knowledge of specific details such as the calculated error of the output layer and the exact amount of contribution each neuron has for said error (Bengio et. al, 2015). This error driven approach assumes that there are designated neurons who are aware of what their exact target values should be, and that many other neurons know this as well, which is itself an unreasonable assumption. The backpropagation also imposes a structural limitation, as neurons must be organized in feedforward layers—which is in great contrast to the brain's reentrant structures (Izhikevich 2004, Edelman et. al, 2013). These are but a few details that indicate that artificial neural networks do not correspond to neurobiological behavior.

There can be great benefit in determining how the brain performs such computations if it is not done by backpropagation. By respecting and abiding by behavior seen in actual neurons

we could, hypothetically, develop neural networks that approach the performance of the actual brain or parts of it, for which we can then reap great benefits and approach the construction of actual artificial intelligence. In this work, we will utilize a spiking neural network whose computations allow for the emergence of forms of associative learning, which allows a neural network to perform similarly to that of a reinforcement learning algorithm.

2 Background

Spiking neural networks are considered to be the third generation of artificial neural networks (Maass 1997). Their main feature over artificial neural networks is that the neurons that make up the network communicate through discrete spiking signals. Because these signals happen over milliseconds, these neural networks inherently take the aspect of time into their consideration. Neurons spike by accumulating membrane potential over time as it receives electrical signals from other neurons (Vreeken 2003). Once a neuron is excited enough, it releases the action potential, which is what we refer to as the spike. This action potential, generally, will last for around 1 millisecond (Paugam-Moisy & Bohte 2012). The signal, like in artificial neural networks, will then be transmitted along its axon, which is the part of the neural anatomy that connects one neuron to another. The synapses of a neuron lie at the receiving end of the axon, and are analogously equivalent to the weights in an artificial neural network in that the synapses can strengthen or weaken signals received from other neurons. It is generally accepted that synaptic plasticity, or the way in which synapses either strengthen or weaken signals received from other neurons, is the core function that allows for the storage of memories, and overall learning to occur (Abott et. al, 2000; Martin et. al, 2000).

Most observed forms of synaptic plasticity are in accordance with Hebbian Theory, which claims that if one neuron tends to contribute to the firing of another neuron, then that relationship is more likely to be enhanced (Hebb, 2005). This theory was further supported with the observation of the phenomenon known as Spike-Timing-Dependent-Plasticity (STDP). STDP is a biological process in which neurons adjust their synaptic connections based on precise spike timing coincidences (Markram 1997; Bi & Poo 1998). STDP can be broken up into two rules: a Hebbian rule and a Anti-Hebbian rule (Florian, 2007; Paugam-Moisy & Bohte 2012). If a presynaptic (input) neuron fires right before a postsynaptic (output) neuron, then the synapse between these neurons will be strengthened. This strengthening effect can be considered the Hebbian part of STDP, since a neuron takes part in causing the firing of another neuron and the connection is enhanced as a result. If a presynaptic neuron fires right after a postsynaptic neuron, however, then the synaptic connection between the neurons decreases in strength. This dampening effect is considered to be the Anti-Hebbian part of STDP (Florian, 2007; Paugam-Moisy & Bohte 2012). Intuitively, if a presynaptic neuron fires right before a postsynaptic neuron (the Hebbian rule), then it is highly likely that the presynaptic neuron contributed to the firing of the postsynaptic neuron, and so this particular connection will strengthen and interaction is thus more likely to happen in the future. Likewise, it should be the case that if a presynaptic neuron fires right after a postsynaptic neuron does (the Anti-Hebbian rule), then it is unlikely that the presynaptic neuron has contributed to the firing of the postsynaptic neuron. As a result, this connection will be weakened to make it even less likely that the presynaptic neuron will cause the postsynaptic neuron to fire in the future. In both cases, an observed temporal coincidence will cause the same coincidence to be more likely to occur in the future–the synaptic plasticity of the neurons is dependent on their timing—hence the naming

of STDP. When these coincidences occur, there must be some leftover trace of that interaction, which is traditionally called the eligibility trace, that indicates that the synapse between the presynaptic and the postsynaptic neuron should be either strengthened or weakened depending on their particular temporal relationship. If both neurons fire at the exact same time, then it implies that some other neuron(s) contributed to their firing, and so no synapses will be affected as there is no temporal coincidence. It should also be noted that neurons that have no relationship at all, in that their spikes are too distant in time to be related, will undergo no plasticity.

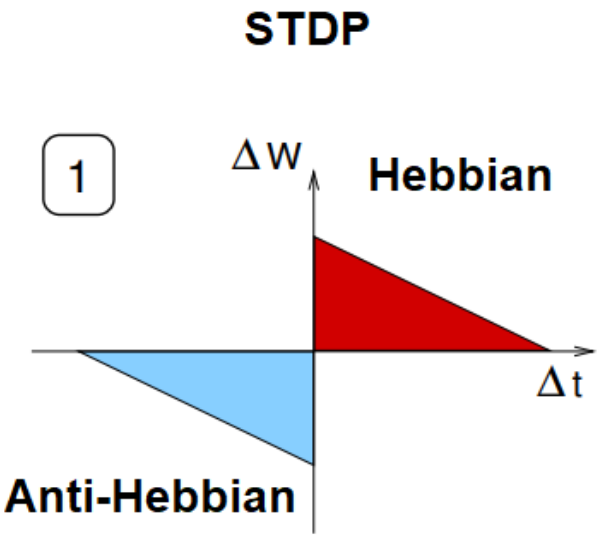


Figure 1: A graph representing STDP between an arbitrary presynaptic and postsynaptic neuron. *Δt* represents the difference between the firing of the postsynaptic neuron and the presynaptic neuron. *Δw* represents the change in synaptic weight strength between the presynaptic and postsynaptic neuron.

Reward modulated spike timing dependent plasticity (R-STDP) is an additional form of STDP in which the temporal coincidences will only be exaggerated if those coincidences led to some reward, which is usually signaled in the brain by the dopamine neuromodulator (Florian 2005; Izhikevich 2007). By itself, STDP can be considered an unsupervised rule in that the network of spiking neurons will change their weights based purely on the stimuli (data) given, as opposed to a supervised or reinforcement learning rule, which gives the network feedback on the network's overall performance in regards to some desired task (Paugam-Moisy & Bohte, 2012). This feedback can then be used by the network in some way to further improve its ability to solve said task. In the case of R-STDP, some form of reward can be given that serves as a gating signal that must be present if synaptic plasticity is to occur. Generally R-STDP is applied by multiplying a reward by the weight changes calculated by STDP (Florian, 2005; Florian, 2007; Izhikevich, 2007). The weight changes in this case are referred to as the eligibility trace (Florian 2007, Sutton & Barto 2018). Intuitively, STDP will calculate which synapses are due for strengthening and dampening, and by how much, which is stored as a trace of weight changes. It is these synapses (weights) that are eligible to change. But in the case of STDP, the eligibility

trace is multiplied by some sort of numerical reward, meaning that the absence of reward will result in the canceling out of the eligibility trace. Alternatively, the presence of reward will not change the network if STDP determines that no synapses are eligible for change. While this is almost certainly a simplification of the brain's rewarding mechanisms, it does allow the network to distinguish between which temporal coincidences matter as a means of increasing the amount of rewards it receives, and which do not. If actions are driven by interactions between neurons, and those actions lead to reward, then by reinforcing the coincidences that occurred when reward was received, the neurons are more likely to cause the same pattern that led to the reward to happen again in some capacity. For simplicity, we assume this reward factor is global and washed over the entire network or some region of the brain responsible for decision making. This should lead to some sort of global optimization, on average, within the network.

It is our main interest to use R-STDP, which requires the use of spiking neural networks, to construct an agent that solves reinforcement learning tasks. With states being encoded as stimuli that are able to then cause neurons that encode actions to fire, we can have a spiking neural network that will exaggerate the spike coincidences that lead to reward. Over time, the agent will improve over time and choose the optimal actions in each state just like that of a typical reinforcement learning algorithm. To simulate the spiking neural network, we use our own Spikey python library, which is a malleable spiking neural network simulator that provides basic spiking neuron operations and extends naturally to reinforcement learning tasks. We refer to the approach of taking the neurobiological learning rule R-STDP, and extending it to be able to solve reinforcement learning tasks as Reinforcement Learning-Spiking-Timing-Dependent-Plasticity (RL-STDP).

3 The RL-STDP Learning Rule

The main purpose of RL-STDP is ultimately to produce an agent that adjusts its behavior to achieve the optimal amount of reward within some arbitrary environment. To be more specific, the agent abides by some policy that maps the states of the environment to probabilities of taking actions within said environment. The evaluation of the policy is dependent on some reward signal that is either received from the environment (external reward) or generated by the agent itself (intrinsic reward). At this time, RL-STDP only utilizes external rewards provided by the reinforcement learning environment's reward scheme. In this iteration of RL-STDP, we primarily utilize a simple feedforward architecture, with one input layer and one output layer, though hypothetically the structure could be extended to include hidden layers and or recurrent connections (see Discussion). Neurons in the input layer correspond to specific states in the environment, which are activated when the agent arrives in said state. Neurons in the output layer correspond to actions that can be taken in the environment, whose activations are dependent on the synaptic weights that connect the presynaptic state neurons to the postsynaptic action neurons.

The RL-STDP algorithm can be best described as continuously progressing through three different timescales or stages: the network stage, the game stage, and the episodic stage (see Figure 3). We will thus describe what happens upon each step in each of these stages, where a step generally refers to a single increment of time in a particular stage.

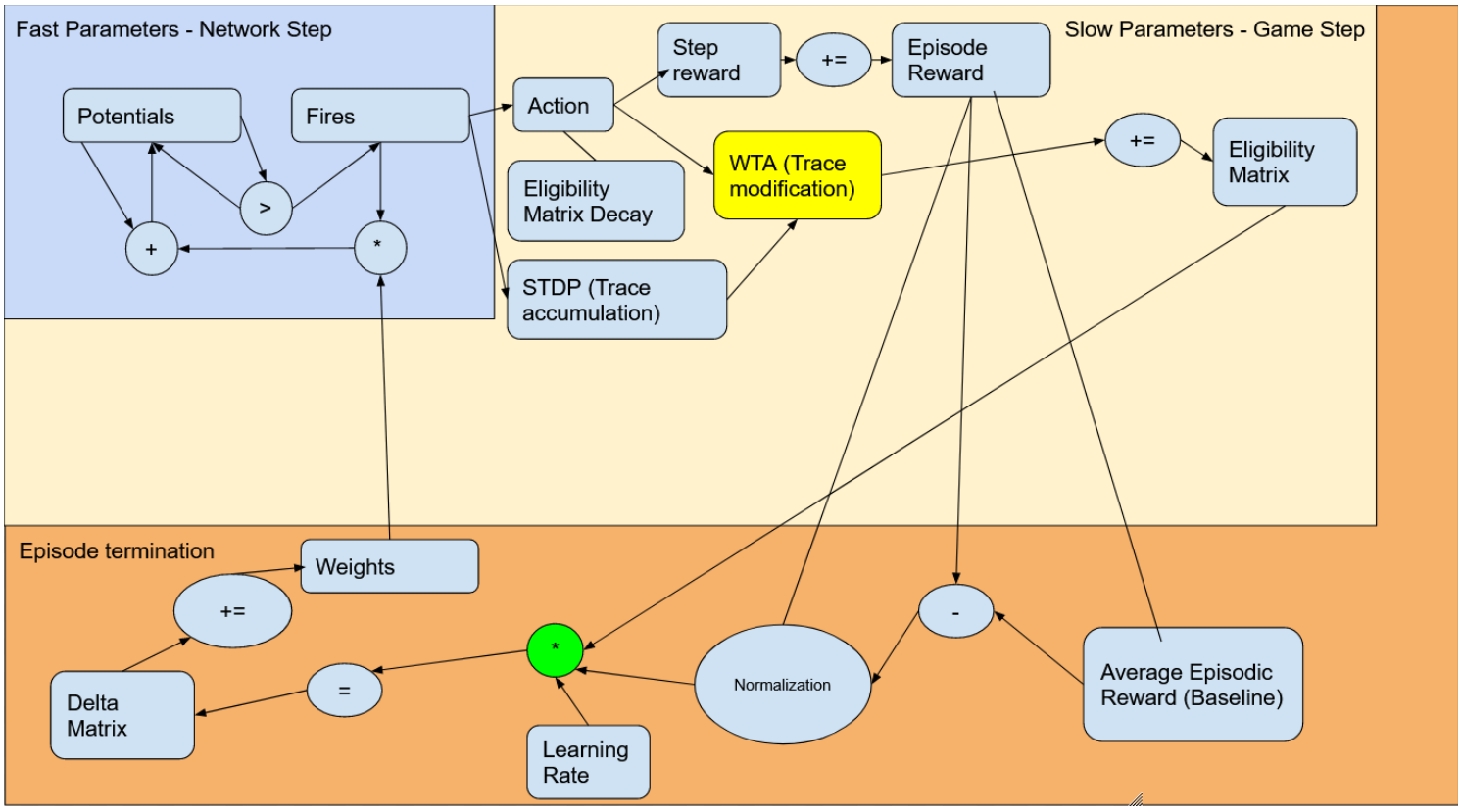


Figure 3: A high level overview of the RL-STDP algorithm. Each operation occurs in one of three phases: the network phase, game phase, and episode phase. The operations of one phase lead into another.

Network Timescale (Perception):

During the network stage, the input neurons that correspond to the agent's current state fire according to some parameterized firing rate. In our models, we assume that a spiking neuron can only fire once per millisecond. This millisecond serves as a single network step, or tick. The spikes from the input neurons are connected to the action neurons, and will build up charge and cause them to spike once their potential surpasses the firing threshold (see Figure 3). The state neurons will continue firing for a certain amount of milliseconds that we refer to as the processing time. A processing time of 20 ms means that the state neurons will fire over the course of 20 network steps, and that the conclusion of the network timescale phase will conclude at the end of those 20 steps. Informally, this process could be thought of perceiving the state stimuli--with the spikes corresponding to the agent "thinking" about the state, before committing an action. The network is subject to spike-timing-dependent-plasticity, which stores a potential weight change traditionally referred to as the eligibility trace that depends on the timing of presynaptic state and postsynaptic output neurons, or STDP.

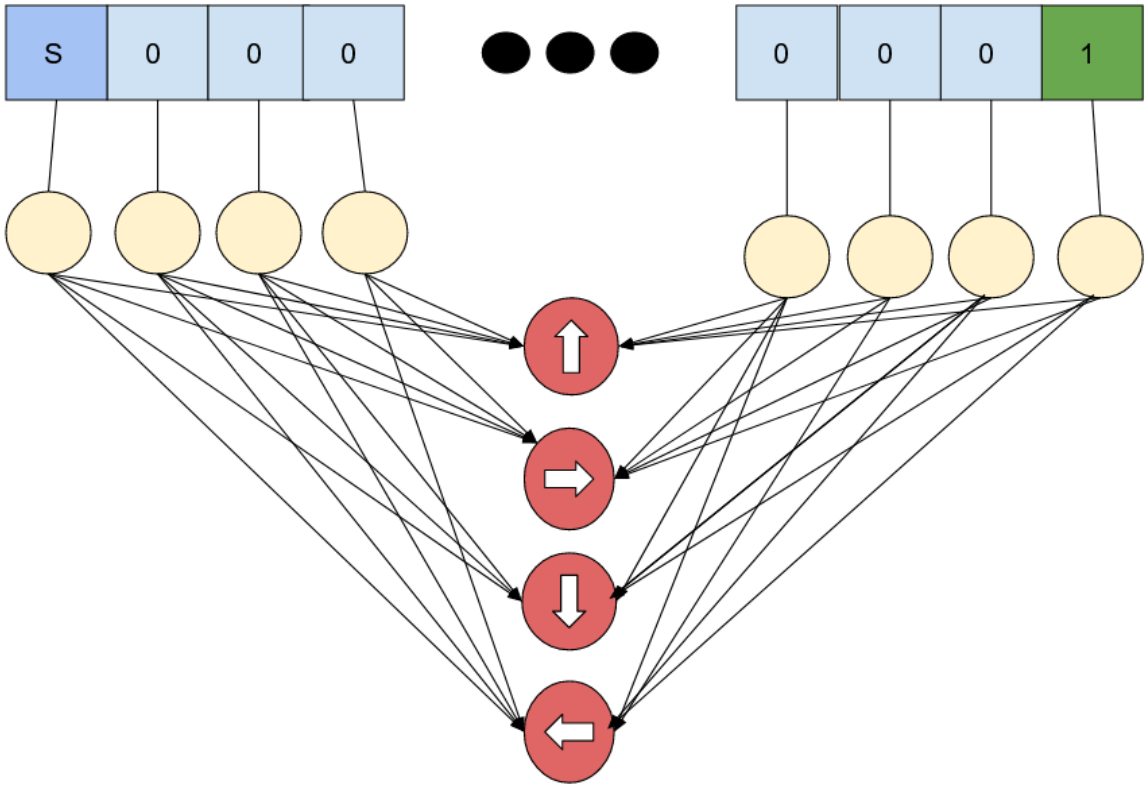


Figure 3: The general structure of the network in the case of a gridword. Each tile represents a state, which has a corresponding neuron that will spike upon transitioning into that state. All of the state neurons are connected to a group of action neurons. In this case, the agent can move in all cardinal directions. State neurons will only fire when their respective state is visited.

Game Timescale (Action):

Once the network phase has completed, we use a readout function on the network to dictate what action the agent has decided to take as a result of perceiving the state over the processing time. Each neuron in the output layer corresponds to an action within the environment's action space. Neurons that correspond to the same action can be thought of belonging to the same action group. The readout mechanism used simply takes the action that corresponds to the action group that exhibited the most activity–specifically the amount spikes over the processing time. If two action groups have the same amount of activity, then one of those actions is chosen randomly. Once the action is taken, the reward that is given by the environment's step function is added to the total reward for that episode.

Alongside this action is optional the activation of the winner-take-all (WTA) mechanism, which is some arbitrary procedure that aims to increase the likelihood that the eligibility trace accumulated over the game step corresponds to the actual action taken. Since the reward is global and washed over the entire network, other neurons from other action groups will also be affected by rewards and punishments in the future. Thus, it is possible that an action can be taken and rewarded, but the underlying spike timings are such that another action group accumulated more of an eligibility for that game step. This will result in the wrong action group getting more

credit than the action group that corresponds to the action that was actually taken. This is an instance of the structural credit assignment problem, in which a decision is formed from the interaction of multiple parts, and the credit for said decision must ultimately be distributed to the parts that contributed most to said decision.

While reward modulated spike-timing-dependent plasticity is capable of structural credit assignment, it has yet to demonstrate the same capability as something like backpropagation, which is comparatively a much more fine-tuned computation. Winner-take-all mechanisms are usually introduced as a means to accelerate learning by making the structural credit assignment problem easier. A somewhat soft but common approach is to utilize the idea of lateral inhibition, in which neurons that spike subsequently inhibit the firing of their neighbors. Specifically, a neuron in one action group could be inhibitory and also be connected to all output neurons that are not within its action group. An extremely forceful approach towards the structural credit assignment is to simply erase the eligibility traces that correspond to the other action groups that were not taken. This essentially removes the structural credit assignment problem from the picture entirely, as the action neurons that truly contributed to the action taken will be the only neurons that are marked as eligible upon receiving a reward or punishment. It should be noted that this particular form of winner-take-all is only applicable to a simple feedforward network that consists of only two layers. While we choose to utilize this approach so that we may focus on the temporal credit assignment problem, it is highly desirable that we eventually develop an approach or network configuration that emergently and effectively resolves the total credit assignment problem. Under this winner-take-all, one neuron per state and one neuron per action is sufficient to see performance on reinforcement learning tasks. We note that it is somewhat similar to something like Q-learning, in that Q-learning will take the action that is most associated with reward, and it does not need to concern itself with the structural credit assignment problem due to the nature of the Q-table.

Regardless of whether a winner-take-all mechanism modifies the eligibility trace for a single game step, or no winner-take-all mechanism is implemented at all, the eligibility trace that is accumulated by spike-timing-dependent-plasticity will be added to the eligibility matrix that is the same shape as the weight matrix. Upon taking any action, this eligibility matrix is decayed by some parameter determined by the experimenter. The trace decay is the primary feature that allows for temporal credit assignment, as actions that happened earlier in time will receive less credit for a reward than those actions that happened more recently. The eligibility matrix, which stores the timing coincidences between the state and action neurons, essentially represents the history of actions taken over an entire episode. Without any trace decay, all actions would be rewarded equally regardless of when they occurred in time. And with complete trace decay, the network would only be able to remember the last action taken. Once the change has been added to the eligibility matrix and the action readout from the network is taken by the agent, the game step concludes and another step begins. Note that every game step consists of multiple network steps which operate on the timescale of milliseconds.

Episode Timescale (R-STDP):

Once the episode has been terminated, the reward used in the final R-STDP product needs to be calculated. The incoming episode reward is first normalized, which requires the network to keep track of the maximum reward and the minimum reward received. The network keeps track of a list of rewards that represent the agent's memory of past episodic returns. This history will be used to both calculate the agent's average reward, and determine the maximum

and minimum reward used for the normalization. The optimal length of the history, or the amount of past episodes to consider, tends to vary between tasks and is thus a parameter that needs to be tuned. The length of the history is akin to the reward discount parameter used in other standard reinforcement learning algorithms like Q-learning and Sarsa. Once an episode terminates, the network reward is calculated by taking the normalized difference between the incoming episode reward, and the average episode reward. With the network reward calculated, the weight delta matrix will be formed by taking the product of the network reward, eligibility matrix, and a parameterized learning rate. This delta matrix will then be added to the weight matrix. This concludes the episode time step. Note that every episode time step consists of multiple game steps, which themselves consist of multiple network steps. The interactions that occur on a scale of milliseconds leads to the formation of behavior on a much longer time scale.

A special case occurs if the average has not been assigned–meaning that this is the agent's first episode. In this case the reward average will be initialized to the first episode's returned reward, and the episode time step will end with no modification of the weight matrix. No learning will occur on the first episode as the return of the first episode, in which the agent acts randomly, serves as a comparative basis for future returns.

Upon completion of an episode, the modified synaptic weights will then be used in the next episode, starting again in the network timescale. The transitions between the network, game, and episode timescales repeat for a given amount of episodes. The overall training period ends when the last episode has concluded.

4 Applications

While the general idea of R-STDP could certainly be applied in other tasks that benefit from classical or instrumental conditioning, it is a primary concern to use the learning rule to solve reinforcement learning tasks. The tasks focused on in this work are referred to as the Cartpole, Mountain Car, and Acrobot tasks. These environments represent some of the most popular problems in the reinforcement learning literature, and are thus perceived by many as a benchmark used to determine the ability of a reinforcement learning algorithm. Though there exist many variations of these types of tasks, the OpenAI implementation represents the most popular versions of the benchmark. All of these environments have continuous state spaces, made up of multiple dimensions or attributes that the agent must observe before taking an action. The current implementation of RL-STDP relies on what is referred to as 'place cells'. Place cells map discrete states to a group of input neurons. When the agent transitions into a state, only the corresponding place cells will fire. Since these tasks have continuous state spaces, each state dimension must be partitioned and enumerated in such a way that each combination of partitions corresponds to a discrete state, which is a standard approach for dealing with continuous state spaces. This approach is not practical for complex tasks with large dimensional state spaces, but this will suffice for the target tasks.

Some of these environments (Cartpole, MountainCar) have formal solve conditions, which indicate that the agent has solved the environment if it meets a certain average reward over a certain amount of episodes. Other environments lack these solve conditions, and so we must impose our own benchmarks based on how typical reinforcement learning algorithms perform in comparison to ours.

4.1 Cartpole Task

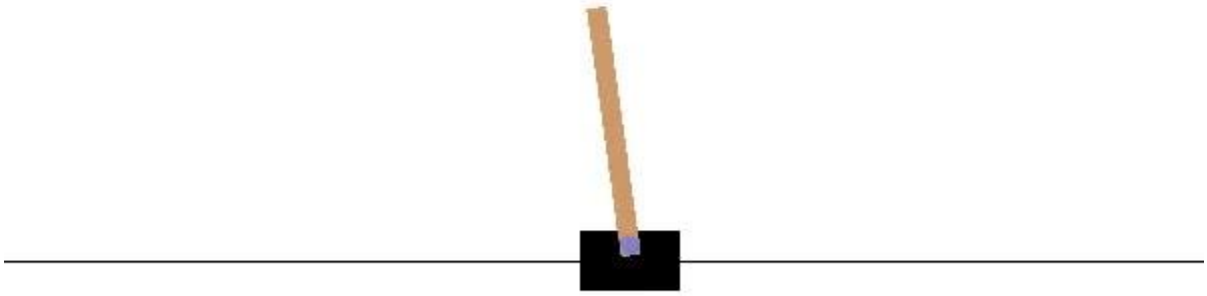


Figure 4: A render of OpenAI's CartPole-v1 environment

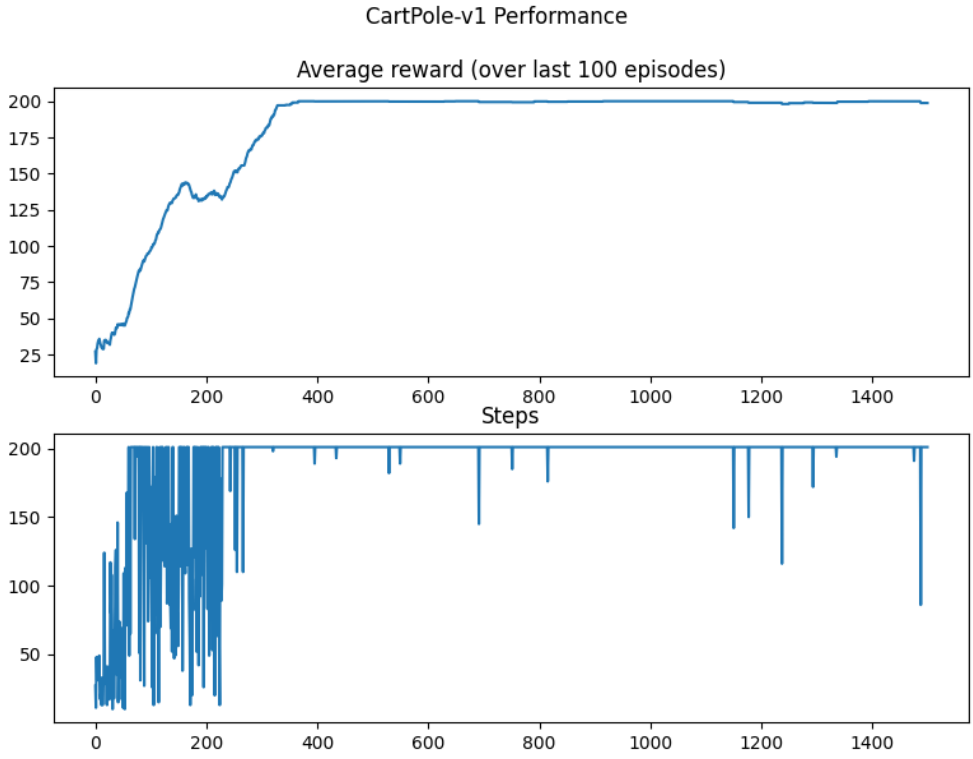
Definition: "A pole is attached by an un-actuated joint to a cart, which moves along a frictionless track. The system is controlled by applying a force of +1 (right) or -1 (left) to the cart. The pendulum starts upright, and the goal is to prevent it from falling over. A reward of +1 is provided for every timestep that the pole remains upright. The episode ends when the pole is more than 15 degrees from vertical, or the cart moves more than 2.4 units from the center." (Brockman et. al 2016)

Environment Spaces:
Action Space: [-1 (left), 1 (right)]
Observation Space: [Cart Position, Cart Velocity, Pole Angle, Pole Angular Velocity]

Solve condition:
"Considered solved when the average return is greater than or equal to 195.0 over 100 consecutive trials."

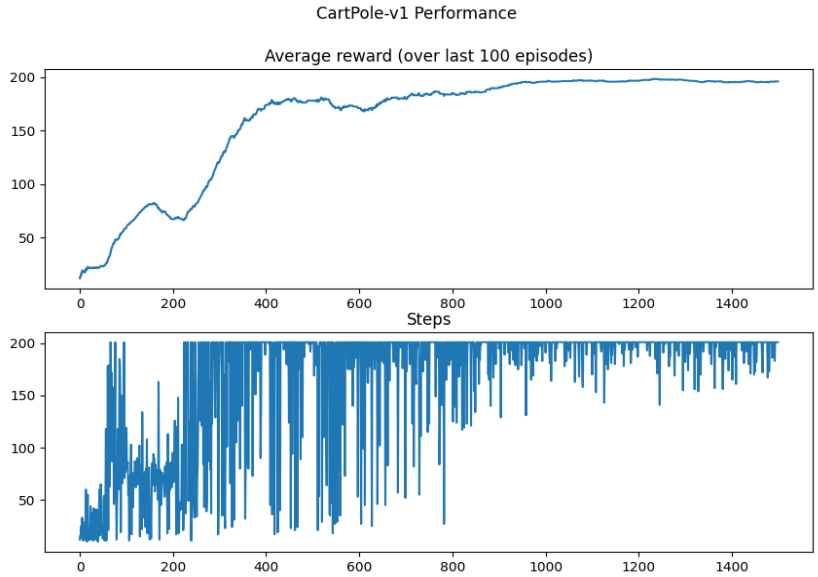
Results:

Experiment 1:



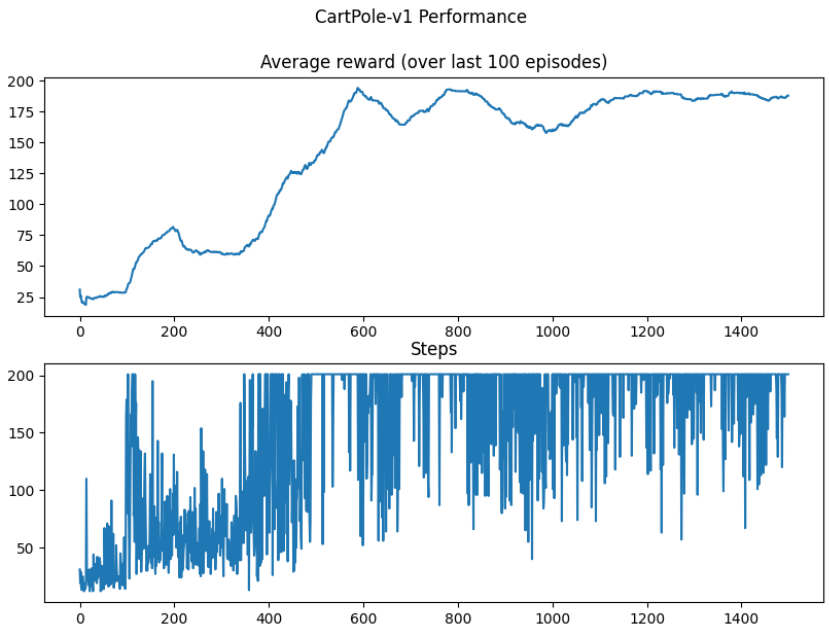
Highest Target Average Reward (over 100 episodes): **199.4**

Experiment 2:



Highest Target Average Reward (over 100 episodes): **197.2**

Experiment 3:



Highest Target Average Reward (over 100 episodes): **194.17**

Observations:

It can be observed that the performance across multiple runs is able to meet the solve condition for the task. Compared to the other environments, this performance is fairly consistent. However, it should be noted that we do not consider all state attributes when we are reading in the state. We found that performance is much better when we only consider the angle of the pole, and the velocity of the pole. This is likely because there are less states to keep track of, and thus it is easier for the agent to recognize that it needs to focus on keeping the pole upright. Despite it going out of bounds occasionally, the agent still recognizes that it receives more reward when it keeps its pole's velocity to a minimum, which inherently prevents the pole from tipping over and forcing the cart to increase its speed to keep it upright. This gives it the appearance that it is deliberately avoiding the bounds, but in reality the agent has no perception of said boundaries.

4.2 MountainCar Task

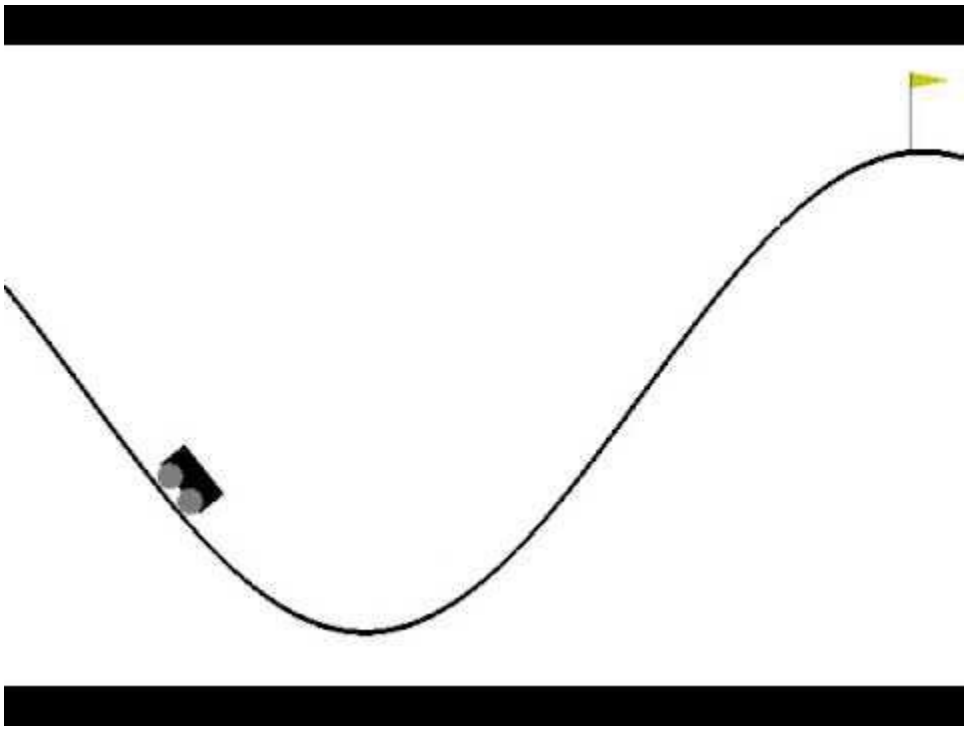


Figure 5: A render of the MountainCar-v0 environment

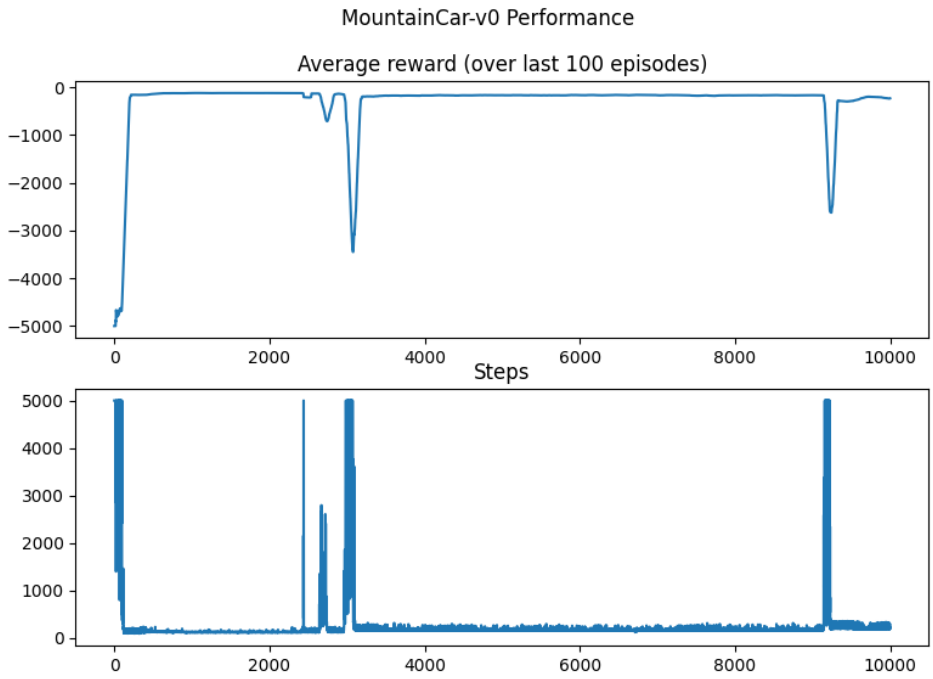
Description: “A car is on a one-dimensional track, positioned between two ‘mountains’. The goal is to drive up the mountain on the right; however, the car's engine is not strong enough to scale the mountain in a single pass. Therefore, the only way to succeed is to drive back and forth to build up momentum.” (Brockman et. al 2016)

Environment Spaces:
Action Space: [left acceleration, no acceleration, right acceleration]
Observation Space: [Car Position, Car Velocity]

Solve condition*:*
“MountainCar-v0 defines ‘solving’ as getting an average reward of -110.0 over 100 consecutive trials.

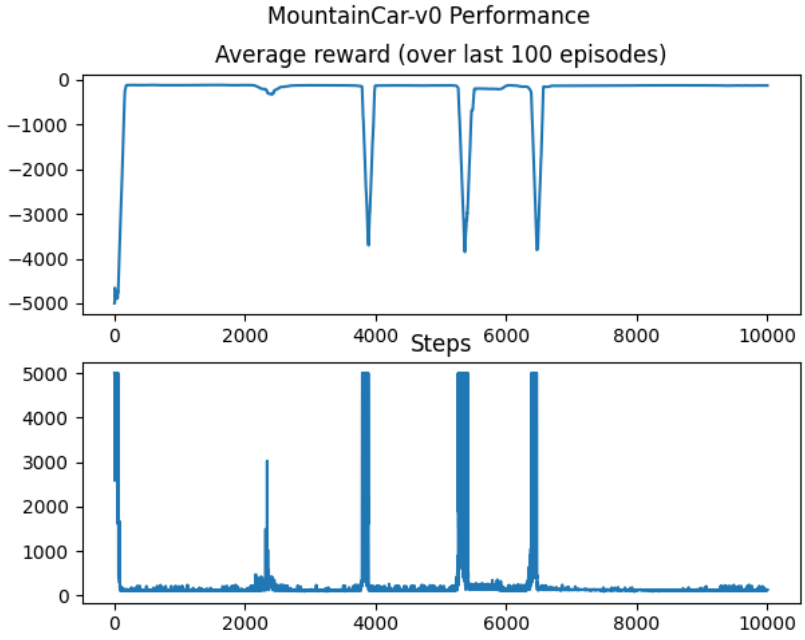
Results:

Experiment 1:



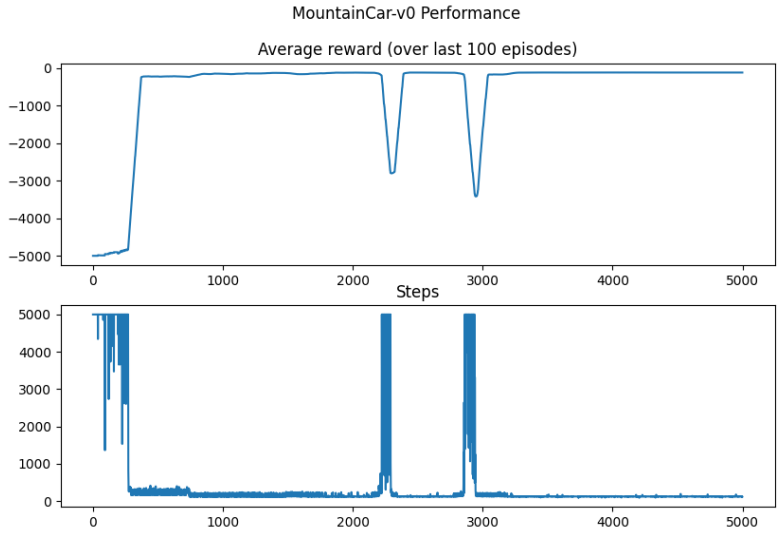
Highest Target Average Reward (over 100 episodes): **-119.79**
Highest Episode Score: **-89**

Experiment 2:



Highest Target Average Reward (over 100 episodes): **-115.69**
Highest Episode Score: **-90**

Experiment 3:

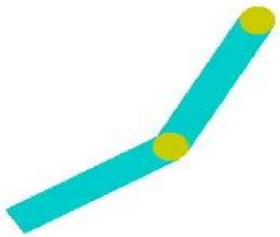


Highest Target Average Reward (over 100 episodes): **-121.28**
Highest Episode Score: **-87**

Observations*:*

Though not all runs are shown, the patterns witnessed on each run are fairly similar. Given enough time to explore, the agent is able to resolve the temporal credit assignment problem and identify which steps led to the reward--resulting in a drastic increase in average reward over time. We consider all state attributes when reading in the state. This problem is slow to improve at first, because it will spend a lot of time moving randomly until it stumbles upon the goal. Even when it does discover the goal for the first time, it may take a few more wins in order to recognize which actions led to that reward. It will then enter a phase in which it consistently reaches the goal in about less than 130 timesteps. Sometimes, if it gets unlucky and takes too long to get back up to the goal, a string of punishments will occur and the agent will forget and take a few episodes to stumble back into the ideal pattern. Unfortunately, we were not able to meet the solve condition for this task.

4.3 Acrobot Task



A render of OpenAI's Acrobot-v1 environment

Description: "The acrobot system includes two joints and two links, where the joint between the two links is actuated. Initially, the links are hanging downwards, and the goal is to swing the end of the lower link up to a given height" (Brockman 2016). A reward of -1 is given for every step until the lower link surpasses this height."

Environment Spaces:

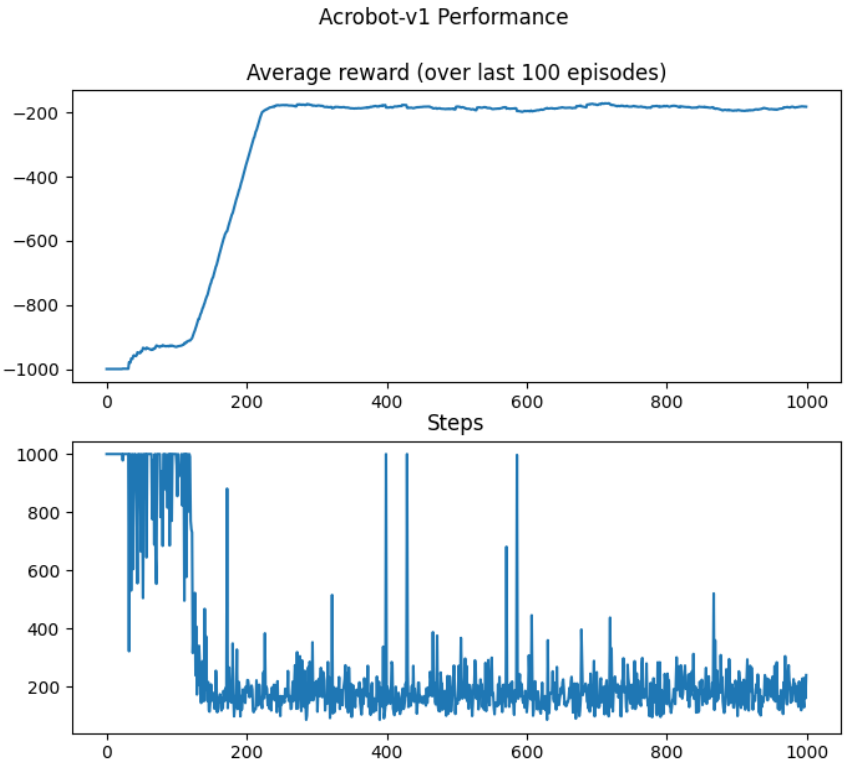
Action Space: [-1, 0 (no torque), +1] (these refer to the torque applied between the two links)
Observation Space: [cos(theta1), sin(theta1), cos(theta2), sin(theta2), angular velocity of first joint, angular velocity of second joint]

(theta1 and theta2 relate to the rotational positions of the first and second link, respectively)

Solve condition*:*
There is no formal solve condition for this environment.

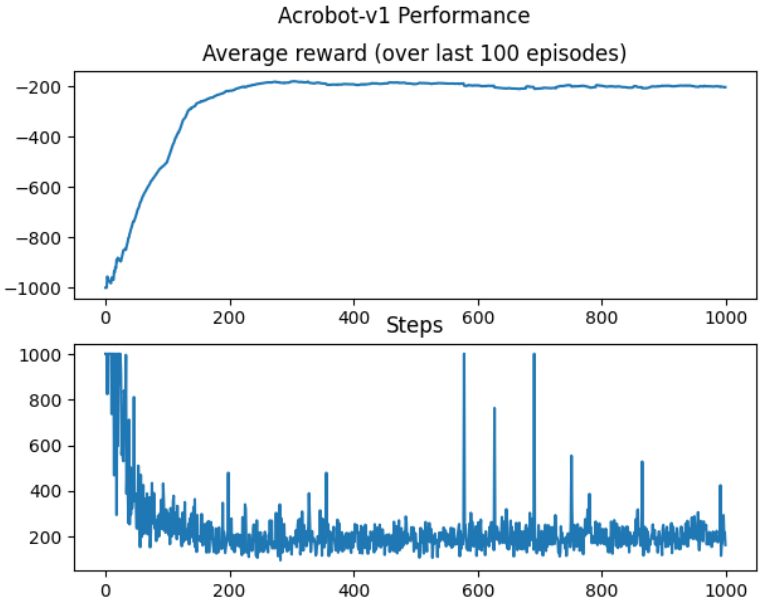
Results:

Experiment 1:



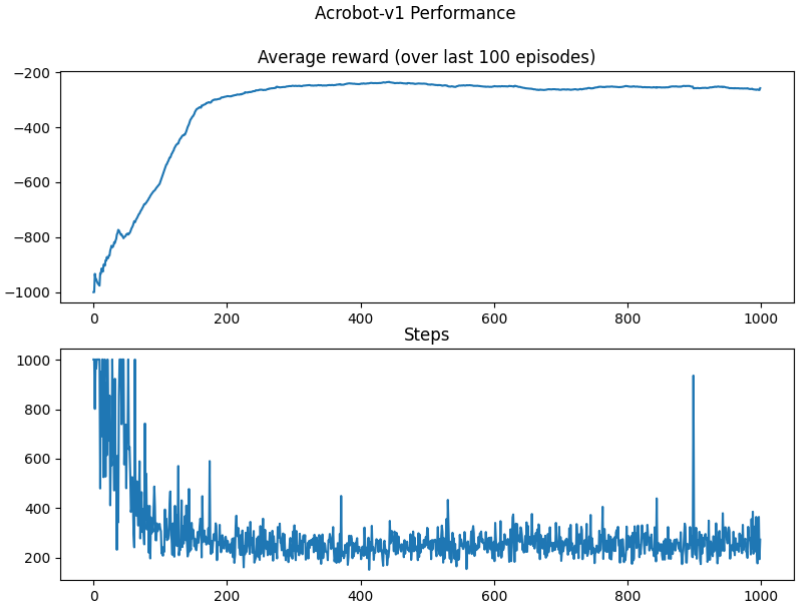
Highest Target Average Reward (over 100 episodes): **-170.3**
Highest Episode Score: **-85**

Experiment 2:



Highest Target Average Reward (over 100 episodes): **-178.97**
Highest Episode Score: **-94**

Experiment 3:



Highest Target Average Reward (over 100 episodes): **-234.58**
Highest Episode Score: **-149**

Observations:
This task is similar to the mountain car task in that it requires the agent to maneuver between states and eventually discover the goal. While the learning is seemingly more gradual, it tends to peak at a suboptimal level of performance. Even though there is no formal solve condition, it is preferable that our agent is able to at the very least solve the task in around 100 time steps. This task also features the most variability, as the quality of experimental runs vary widely. We consider this task to be the most difficult of the three, as the task requires precise movements like in the cartpole task to move the links upward, and the agent must also stumble around and discover the goal in the first place, similar to the mountain car task.

5 Discussion

It is our main intention to have a spiking neural network perform optimally in tasks similar to these without having to resort to forcefully solving the structural credit assignment problem by utilizing a winner-take-all mechanism. While this was attempted during the creation of this algorithm, it soon became a developmental bottleneck. We have seen some success in

simpler reinforcement learning tasks, where the temporal credit assignment problem is taken out of the picture, and the problem is reduced to taking actions in a few states. We have also shown, primarily through the results in the aforementioned tasks, that the decaying eligibility traces are able to mostly solve the temporal credit assignment problem. It is perhaps not surprising to find trying to solve both credit problems at once, which is the total credit assignment problem, is far more difficult than solving just one or the other. We still believe, however, that reward modulated spike-timing-dependent-plasticity is still a main component of the solution that will solve this problem. More experimentation will need to be done to demonstrate this claim.

We also intend to have a rule that strays away from structure dependence. Hypothetically, we do not need feedforward networks to solve reinforcement learning problems. Spike-timing-dependent plasticity only considers the temporal coincidences of neurons, and so any structure in which neurons are connected can be subject to STDP and R-STDP. However, a feedforward structure is the most simple and the most familiar to us given our previous experiences dealing with artificial neural networks. Homeostatic processes that occur alongside STDP and R-STDP, like synaptic scaling, cannot be applied to a feedforward network. At the very least they rely on hidden layers. We have considered that trying to use a biologically unrealistic structure with a biologically realistic rule is not an approach that should be further explored. Perhaps more of a focus should be placed on recurrent connections or other arbitrary network structures. Nonetheless, more experimentation will be needed to argue for or against such claims.

We hope to use this implementation of the RL-STDP algorithm as a basis to improve on and eventually give rise to a rule that can solve the total credit assignment problem in a biologically realistic way–such that the behavior is emergent rather than being forced by the experimenter.

6 Acknowledgements

I would like to thank Dr. Taylor for providing me with many resources and the opportunity to participate in a project that has greatly developed my skills and intuition as a researcher.

References

Abbott, L. F., & Nelson, S. B. (2000). Synaptic plasticity: taming the beast. *Nature neuroscience*, *3*(11), 1178-1183.

Bengio, Y., Lee, D. H., Bornschein, J., Mesnard, T., & Lin, Z. (2015). Towards biologically plausible deep learning. *arXiv preprint arXiv:1502.04156*.

Bi, G. Q., & Poo, M. M. (1998). Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. *Journal of neuroscience*, *18*(24), 10464-10472.

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

Edelman, G. M., & Gally, J. A. (2013). Reentry: a key mechanism for integration of brain function. *Frontiers in integrative neuroscience*, 63.

Florian, R. V. (2005, September). A reinforcement learning algorithm for spiking neural networks. In *Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC'05)* (pp. 8-pp). IEEE.

Florian, R. V. (2007). Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity. *Neural computation*, *19*(6), 1468-1502.

Izhikevich, E. M., Gally, J. A., & Edelman, G. M. (2004). Spike-timing dynamics of neuronal groups. *Cerebral cortex*, *14*(8), 933-944.

Izhikevich, E. M. (2007). Solving the distal reward problem through linkage of STDP and dopamine signaling. *Cerebral cortex*, *17*(10), 2443-2452.

Maass, W. (1997). Networks of spiking neurons: the third generation of neural network models. *Neural networks*, *10*(9), 1659-1671.

Markram, H., Lübke, J., Frotscher, M., & Sakmann, B. (1997). Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs. *Science*, *275*(5297), 213-215.

Martin, S. J., Grimwood, P. D., & Morris, R. G. (2000). Synaptic plasticity and memory: an evaluation of the hypothesis. *Annual review of neuroscience*, *23*(1), 649-711.

Paugam-Moisy, H., & Bohte, S. M. (2012). Computing with spiking neuron networks. *Handbook of natural computing*, *1*, 1-47.

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

Biologically Plausible Deep Reinforcement Learning

Darrien McKenzie
Department of Computer Science
Missouri University of Science and Technology

Reflection

1.) Describe your foundational understanding of how research is conducted in your discipline

Though we are working in a niche intersection between computational neuroscience and reinforcement learning, our experiments bear more resemblance to how research is conducted in reinforcement learning as opposed to computational neuroscience. Before OpenAI Gym, the library used to simulate environments (or "games") that an agent can solve, experimenters had to construct their own environments with their own dynamics and reward schemes in order to demonstrate the performance of their method. Not only is constructing environments like Acrobot from scratch a non-trivial task, but it would differ from someone else's simulation unless the source code for the environment was shared–which was rarely done. OpenAI gyms provide universal benchmarks that any reinforcement learning practitioner should be familiar with if they wish to convey the efficiency of their method. Being able to perform on these tasks was therefore the end goal, and from there, we constructed less complex tasks like gridworlds as a means to build up to these milestones. Were we to eventually solve the 'classic' tasks in OpenAI gym–we could chase the more complex environments within the library, and every consecutively difficult environment solved represents another accomplishment that is easily demonstrated and understood by anyone who is familiar with the reinforcement learning discipline.

2.) How have you expanded your understanding of the informational resources available and how to best use these resources?

In the beginning, I was given many resources that could have proved useful, though at first this led me to missing the forest for the trees. I attempted to read and understand every paper thoroughly, which proved to be an arduous task given that the research papers were written at a very high level and as such were not the most digestible materials–nor were many of those complex details necessary for me to pursue our goal overall. It took talking with one of my colleagues, Cole Dieckhaus, who had been in this situation before, to simplify things and get me to see the big picture. I now no longer agonize over every detail–only digging in when it is of relevance to the development of our research. I imagine that this skill will strengthen over time with practice, and I do not claim that it is perfect.

3.) Describe the knowledge you have gained regarding the fundamentals of experimental design

It is important to start small and progress iteratively when designing experiments. In the case of reinforcement learning, there are many environments that we are interested in performing in, but it would not have been practical to pursue those environments to start with. I believe there was great value in starting with the simple Gridworld environments that could be solved with a few actions and did not require some method of simplifying the observations. If this simple environment could not be solved–it usually indicated that the issue was with the network itself, rather than the complexity of the environment. Problems were better able to be isolated and resolved using this iterative approach, and I couldn't imagine the headache I would acquire trying to manage many complexities at once.

4.) Describe how you have learned to interpret the results of your research project

In this work, I did not acquire the ideal result of a completely randomly interconnected neural network of many different neurons being able to organize itself using neurobiological rules to solve complex problems. We were able to achieve some performance using a specific structure of the network, and I do believe that in doing so I was able to resolve some of the problems that will eventually contribute to us being able to develop a rule that does not have such restrictive structures in the future. We can build upon these results by taking away restrictive elements, resolving performance problems that arise from removing said elements, and then aim to remove another element one-by-one until we arrive at a form of network that has the neurobiologically inspired self-organizing properties that we desire. Our results are but one step towards this goal.