POLITECNICO DI MILANO
DEPARTMENT OF ELECTRONICS, INFORMATION, AND BIOENGINEERING
DOCTORAL PROGRAM IN INFORMATION TECHNOLOGY

PERCEPTION AS BEHAVIOUR INDUCING MECHANISM: A REINFORCEMENT LEARNING PERSPECTIVE

Doctoral Dissertation of:
Mirza Ramicic

Supervisor:
Prof. Andrea Bonarini

Tutor:
Prof. Francesco Amigoni

The Chair of the Doctoral Program:
Prof. Andrea Bonarini

Year 2018 – Cycle XXX

A great thanks to Professor Bonarini for an amazing journey this PhD has been. Thank you for trusting me with your bow tie ⋈ It has been an honour.

To my parents. To my uncle Feruh who passed away during the writing of this work. You always believed in me.

I appreciate the help from Professor Dusan Starcevic and Professor Tim Miller taking the time to review the work and provide feedback on how it could be improved.

Abstract

Rapid advancement of machine learning makes it possible to consider large amounts of data to learn from. Learning agents may get data ranging on real intervals directly from the environment they interact with, in a process that is usually time-expensive. To improve learning and manage these data, approximated models and memory mechanisms are adopted. In most of the implementations of reinforcement learning facing this type of data, approximation is obtained by neural networks and the process of drawing information from data is mediated by a short-term memory that stores the previous experiences for additional re-learning, to speed-up the learning process, mimicking what is done by people. In this work, a multitude of techniques are presented, each of them concerned not just with data collected by the agent, but with how the feedback from its environment is encoded and managed by the learning mechanism trough a mediating replay memory structure. This opens up a possibility of implementing different replay memory architectures forming different modes of agent's artificial perception. The techniques presented in this work try to exploit the influence that an artificial perception brings on the learning process. Their application ranges from improving performance to inducing behavioral characteristics of learning agents.

Reinforcement learning agents can use artificial perception in order to support the emergence of personality traits commonly found in humans. Experimental results show that different personality traits help the agent adapt and thrive in different variations of the environment. Two novel algorithms are presented; one models the main personality dimension of Introversion-Extraversion in a multi-agent social environment while the other is concerned with the emergence of the dimension of Openness to Experience.

Reinforcement learning agents are able to evolve their artificial perception over generations in order to be more efficient in receiving the feedback from the environment. Experimental results show that the novel algorithm is capable of evolving the state of artificial perception not only to improve performance but also change the overall

agents behavior and prevent oversaturation of replay memory.

Reinforcement learning agents can make use of the artificial perception to improve the overall perfomance of learning with two novel algorithms. Experimental results show that both context augmented machine learning algorithm and a novel sampling criteria based on Shannon's entropy can outperform the standard uniform sampling and improved sampling based on the temporal difference error prioritization.

Reinforcement learning agents can benefit from artificial perception in order to learn temporally extended complex actions. A novel approach is presented.

Summary

The introduction presented in chapter 1 provides a motivational basis for the general theme of this work and outlines in short the underlying principles of the main mechanism that this work focuses on: reinforcement learning.
Chapter 2 introduces the main meta-algorithm of *Artificial Perception* on which most of the algorithms presented in this work are based by first presenting its inspiration and origins in biological mechanisms.
The mechanism of *cognitive filter* is used in chapter 3 to model the human personality traits by modifying the agent's *perception* dynamics, specifically *openness to experience* in section 3.1 and the main axis of personality *extraversion* in section 3.2.
The same dynamics of *perception* is artificially evolved using *genetic algorithm* as presented in chapter 4 in order to provide the agents with the better adaptation to their environment.
Chapter 5 extends the use of *cognitive filter* beyond modeling the behavioral characteristics and, instead, presents a novel sampling dynamics and a novel sampling criterion that combined with the algorithm improve the convergence rate of the main reinforcement learning algorithm.
Chapter 6 focuses on making use of the *replay memory* component from the *cognitive filter* in order to enable efficient learning of the complex hierarchical actions that span across multiple learning steps by providing a delayed reinforcement.
The discussion of the implications of this work as well as the general overview of the presented material is found in chapter 7.

Contents

CHAPTER 1

Introduction

Machine learning has established itself as one of the important technologies with a wide application domain that is still about to reach its golden age of expansion. *Reinfocement learning* or RL as a branch of machine learning doesn't depend on the human expert training information since it generates the all of the required information from the consequences of its decisions that are a product of its interaction with immediate environment. This gives reinforcement learning an advantage over the problems where is difficult to obtain a reliable expert training. There are already interesting areas on applications and industrial products that rely on RL. It has been used to design first reinforcement learning DRAM controller that could outperform speed of process executions of the state-of-the-art ones by taking into account of long-term consequences of task scheduling [28]. Researchers from Google DeepMind played 49 different Atari 2600 games using a RL approach called *Deep Q-learning* or DQL [39, 40] and were able to perform beyond human playing level on a good number of games. A year later, another project was presented by the same team called *AplhaGo* [55], a combination of deep convolutional neural networks, reinforcement learning, supervised learning and Monte Carlo tree search achieved amazing results by winning over professional players in the complex game of Go.
In the process of reinforcement learning [59] , agents perform a sequence of interactions with their immediate environment defined over a *Markov Decision Process* or MDP [59]. During this process the agent constantly receives a specific feedback from its environment in the form of a scalar reward that provides the information used in creating a policy π. A policy is a function that maps the perceived state of the environment that the agent is currently sensing to the *Value* of each action available to the agent. Value functions in *Temporal Difference* or TD based learning [62] can be viewed as higher order rewards or their predictor as they represent an estimate of

the amount of expected reward in the future after that state is visited and the specific action was performed. Creating and maintaining policy is the goal of the reinforcement learning, which is used to determine the agent's behavior or, more precisely, which action will potentially yield a better sequence of rewards in the future given the current state of the agent. We call as *optimal policy* π^* the policy that maximizes the reward in the long run: it provides an agent with optimal behavior based on the state of its environment.

1.1 The curse of dimensionality part I: Information overload

If we look at this learning approach in a broader sense, it consists in a constant process of reducing *uncertainty*, or *entropy*, of the set of possible actions that the agent is able to perform in each given state, as perceived from the environment, in order to maximize its reward in the long run. At the very beginning of the learning activity, the probabilities of selecting actions as given by their state-action pair values can be set as similar, because the agent has not learned the proper action for the state, yet. This corresponds to a high uncertainty of the agent's belief about its environment. As the agent proceeds in the learning process, the estimated values for the actions become more differentiated from each other, the probabilities shift towards the preferred actions and the uncertainty or entropy of the agent's belief about its environment decreases. This process effectively compresses the high amount of information received from its constant interaction with environment into a much higher density representation of *values* of the possible actions. The compressibility of the information decreases with the increase of state space dimensionality or amount of variables that the agent perceives to interpret the environment it is acting. In a typical learning episode, an agent performs hundreds of thousands of transitions, each described by a starting state s_t, ending state s_{t+1} a reward received and an action taken which, when it is facing a high dimensional state space, drastically increases the amount of information it needs to compress into the value function. While the state space dimension increases with the advancement of new approaches, the amount of actions that are available to the agents in most cases is, for practical reasons, very low in comparison. Often, we expect an agent to navigate a 2D space by taking 4 primitive actions: left,right,down,right. This brings to an inefficiency of trying to compress ever growing amount of data received from the environment with a constantly limited dictionary that is encoding a great deal of information into a low number of possible actions.

1.2 The Perception Paradox

We have seen that the reinforcement learning process can be viewed as a compression of information received by a multitude of agent's experiences into a limited set of functions that govern its behavior for the purpose of better adaptation to the environment. This implies also that if an agent has to improve its adaptation, it needs to extract as much useful information as possible from its environment in the form of transitions. This is achievable by designing an optimal state space able to encompass all relevant characteristics of the environment, and also by motivating the agent into

transitioning to more "interesting" states by additionally rewarding the transitions that potentially carry more informational content. This brings us to a paradoxical situation in which a well designed agent is *information hungry* about its environment, but it is also limited when it comes to effectively compress this information into its value function. With the promise of better function approximation techniques used by recent developments, agents are able to perceive and adapt to an increasingly complex environments by increasing the dimensionality of their state space. This is especially evident in the use of DQL approaches, which, for instance, are able to play Atari games where the state space representation is including every pixel of the viewport [39] . Despite this growth in the amount of information perceived from the environment, the agents are still faced with a small action space, which also accounts to a few atomic actions in Atari games.
Similiarly to the above mentioned computational solution, the evolutionary development of a human brain was followed by its ability to perceive and elaborate an increasing amount of external stimuli from its immediate environment. However, no matter how much the brain processing power increased, its ability to process these sensory data still remained very much limited [16] . Contrasting this limitation, the same paradoxical need for the information is present in the brain's constant need for external stimuli. The preference for novel experiences is also found in the developing human brain, as babies not only prefer, but are driven to focus attention on situations that include novelty and surprise [6, 27, 63] in order to acquire the needed stimulation. The brain during this phase of the development depends so much on the entropy of its perceived environment that if the brain of a newborn is sensory deprived of its much important stimuli, it will not develop. Like the computational approach of reinforcement learning, it needs to be constantly surprised in order to develop. However, recent discoveries in neuro-plasticity also show that the fully developed brain is not that static as previously thought, since it is constantly changed and shaped by perceived information, or the lack of it. As the information flows in, the neural pathways that get stimulated make new connections and grow, and the ones that are not stimulated will shrink over time. The brain is being modeled by information that is perceiving and this modification affects its further perception. This hunger for information is so evident that the most of our insights in the workings of this complex organ comes from the studies that examine what happens when the person is deprived of it. For example, if a person had lost its ability to see, the part of the brain in which the visual information is processed will shrink and some other competing sense, such as hearing, will expand over this area making new connections between neurons [21]. Along with the sensory deprivation, too much activation of one specific brain area can influence its growth. This is the case, for example, when we try to learn how to master a musical instrument: the part of the brain in charge of processing notes significantly expands to cope with the overflow of this specific information.

1.3 The need for selection

At the current stage of development our brains are bombed with millions of bits of information each second, but we are able to consciously process only about 126 bits

over that time interval [16]. The existence of this cognitive bottleneck brought the need for another mechanism, capable of focusing, at each time, on the subset of information actually useful for the cognitive process. This cognitive filter that protects us from the sensory overload caused by perceiving millions of bits every second is called *attention* [16, 23]. The studies using FMRI confirmed that this filtering process is happening on the level of neurons. In one of the studies [33], monkeys have been shown a green circle followed by green circle accompanied by a red one. What is interesting is that the activity in the visual cortex was exactly the same when the green circle is shown besides a red one as when the green circle was displayed on its own. This effect is called *biased competition* as it seems that the similar type of neurons compete with each other for the specific stimuli that can provide an activation. This also means that the amount of perceived, competing information determines the amount of attention needed.
There are two kinds of selective attention that the human brain is capable of performing. One is *controlled attention* and its mechanisms are active when we want to consciously force our attention to an object; the other one is *stimulus-driven attention* when our brain is focusing on a surprising event in our environment. Both of them evolved separately in different parts of the brain in order to improve our chances of survival. First type of attention helped us hunt for sources of food with the ability to focus on the pray while the second one protected us from predators by shifting the attention towards potential source of danger.

1.3.1 The Protective layer

A major role in this filter is played by a system named *working memory* or WM, which acts as a buffer between our perception and its conscious processing [3]. The *working memory* consists of a temporary memory storage along with specialized mechanisms for replaying its contents. This seemingly simple mechanism is essential for many mental tasks such as controlling the attention to solve logical tasks. It also relies on the central or executive attention that is in charge of regulating its active contents therefore providing a specific memory-related context for the higher order cognitive processing [22]. By now, a growing number of psychological and neuroscience studies have confirmed that the *working memory* is selective when it comes to storing stimuli: sensory stimuli that are important to the goal show enhanced activity while the other irrelevant stimuli are suppressed [18, 20, 26]. This buffer not only serves as a protective barrier against sensory overload, but provides a cognitive context by holding on to the memories that are supportive of our current goal. This is especially important from the standpoint of reinforcement learning in which the behaviour is fully goal-oriented and could benefit from this type of selective directed behavior.
Along with the types of experiences stored, the WM capacity can greatly affect the higher cognitive processes or our ability to solve problems [22]; the higher the capacity the more context we have for reasoning and solving demanding tasks. As we can see from the simplified illustration Figure 1.1 the capacity is not everything when it comes to perception. Cognitive bandwidth also referred as *breadth of attention* [31] signifies how much different sensory information can a person focus at a given time,

it can be viewed also as a bandwidth of a channel from perception to the working memory.

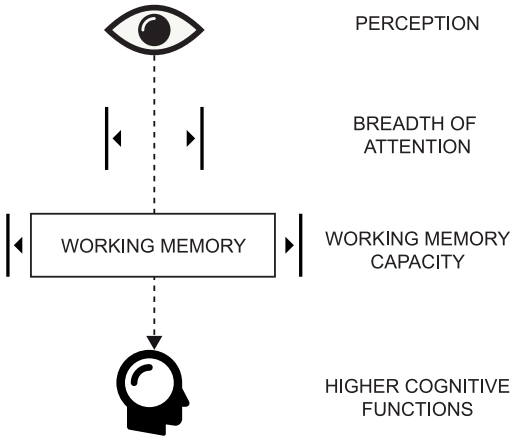


Figure 1.1: *Simplified model of the WM ability of mediating perception by its capacity and breadth of attention*

In fact, machine learning algorithms are now more than ever taking inspiration from the physiology of human brain that is especially evident in approximation techniques of Deep Q-learning or DQL [39] and its replay memory mechanism which is functionally related to the concept of *working memory* found in humans. Similarly, in modern approaches to DQN the stream of experiences are not directly propagated to the learning mechanism in order to create a TD error that is used to update the *Artificial Neural Network* ANN function approximator. Instead, they are mediated by a specific mechanism called *Experience Replay* [36, 39] that is used to store some of the experiences in a sliding window memory in order to make it possible for an agent to learn from its past experiences.

1.3.2 An Autistic Robot

When faced with sensory overload due to the lack of mediating cognitive filter that *working memory* provides in humans, the presented oversimplified model of reinforcement learning certainly would not exhibit complex pathological behaviors, but its learning or compression abilities will be affected by an increased level of competing, redundant and inadequate information that is receiving from its environment. As the learning progresses, the function approximator used to predict the value of the state-action pairs $Q(s, a) = f_\theta(s, a)$ is constantly changed by shifting its coefficients in order to determine the optimal action. The experiences that agent perceives vary according to their ability to convey information to the approximator or they relatedness to a specific goal that the agents is trying to perform. Because of this constantly shifting dynamics of the approximator an agent that is being equally exposed to experiences in in risk of constantly unlearning the effects of important experiences from the exposure to noisy experiences that are not likely to influence learning as much. The signal formed by the important experiences that is crucial for our approxima-

tion is being lost in the noise of the cognitive stream.Due to the training algorithm based on *gradient descent* in order to theoretically achieve the correct approximation at each learning iteration our function approximator should be trained on all of the dataset or in our case the cumulative experience history up to that point. This being highly inefficient, we instead sample a small batch of experiences from our *replay memory* for the re-training which should be our representative sample of the whole experience set. This also accounts for sensory overload because each experience is considered of an equal value for the approximator and therefore the re-training of the more important experiences will be less effective at the cost of noise that is created by revisiting less important transitions. The inability to separate the important information from the sensory overload of our perception can lead to serious problems in humans. Due to the overwhelming complexity of human brains most of our insights come from the situations where some specific part of it is damaged or malfunctioning. This is how we identify which area of the brain is essential for processing which type of activity by identifying which higher-order complex brain function is reduced or lost by the lack of the missing part.
One of these situations appears when a combination of genetic predispositions and certain toxic events in the early embryonic development form a series of hypersensitive neural circuits in the brain that can later affect its functional connectivity, and lead to the most common symptoms of autism. Recent studies [24] show that the main drive for autistic behaviours is the social and sensory over stimulation which can lead to difficulties in development. Brains affected with autism are hypersensitive and prefer predictable, well-structured and safe stimulations from environment because of their innate inability to process high amounts of information that is provided by their perception. In other words, people with autism have difficulty sorting out important information from the noise of the less important stimuli. Since the ability to prioritize on the information is affected, people with this condition often tend to isolate themselves from the massive sensory overload, by seeking relief in environments that can provide them with sensory deprivation. The bias towards lower entropy environments that are not able to provide enough information for development destroys the subject's ability to learn and feel, and leaves them disconnected from the reality of ever changing social and sensory environment of their daily life. The inability to process the sensory perception in autism is often attributed to a specific condition of *Sensory Processing Disorder* or SPD introduced by sensory integration theory in early Eighties [2]. This condition accounts for the difficulty in discriminating sensation and the feeling of being overwhelmed with the sensory stimulation. Some patients with SPD respond normally when only one sensory modality is presented, but when two or more stimuli types are presented, as in normal daily life they are not able to provide an appropriate response.

1.4 The curse of dimensionality part II: Approximation

1.4.1 Starting simple

One of the simplest environments that we can define over MDP is the so-called *grid-world* shown in Figure 1.2. It consists of a rectangular grid in which every cell

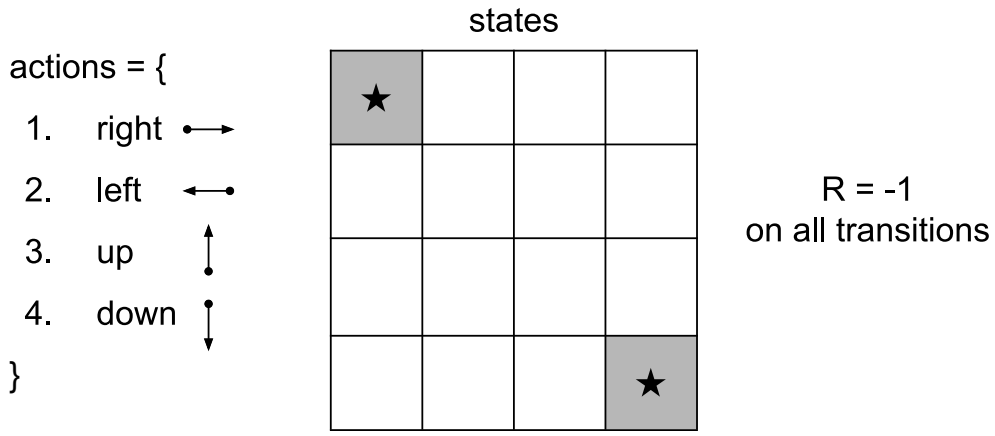


Figure 1.2: *A simple Gridworld showing the possible states and set of available actions*

represents either a single state defined as a transitioning one, shown in white, or a terminal state represented by a star. The action space is consisting of four atomic actions that move the agents deterministically into the adjacent cells as a consequence of each transition. In the terminal state the agent is rewarded by a scalar value of $r = -1$. As we have seen before, the goal of the learning process is the creation of a policy π which maps the current state to a probability of selecting an action that would take the agent to a next state. More precisely, we seek out a specific policy called *optimal policy* that will achieve a maximum reward over the long run. To help us determine the optimal policy we can calculate *value functions* v for the possible states. These value functions represent an estimation on how good is the given state or how much return in the form of reward we can expect from being in that specific state and following the policy π.

To obtain an optimal policy we follow two *Dynamic Programming* or DP algorithms. The first one enables the system to obtain the values for each state by multiple iterations of Equation 1.1 over all states summing up the values all of the possible actions that lead the agent to the next states. This process is called *iterative policy evaluation* and as we can see from Algorithm 1 it requires tedious computation in which the value for each state is updated in multiple sweeps using Equation 1.1 until the value of the state is reaching a stability.

$$v_\pi(s_t) = \sum_{a_t} \pi(a_t|s_t) \sum_{s_{t+1}, r_t} p(s_{t+1}, r|s_t, a_t) \left[r_t + \gamma v_\pi(s_{t+1})\right] \tag{1.1}$$

The starting point of this algorithm is shown in Figure 1.3 which initializes the value of the states to equal 0.0 and defines an equiprobable random policy that doesn't favour any action and this represents the situation with the maximum unpredictability or entropy.

After three sweeps of Algorithm 1 we can see on the left grid of Figure 1.4 that the value functions are differentiating from each other and the agent has learned to value of some states more than others. All that is left is the process of updating the policy π to represent an optimal policy π^* w.r.t the new values for the states using the

Algorithm 1 Iterative Policy Evaluation

Input π, the policy to be evaluated
Initialize and array $v(s_t) = 0$ for all $s_t \in S^+$
repeat
  $\Delta \leftarrow 0$
  for each $s_t \in S$ **do**
    $b \leftarrow v(s_t)$
    $v(s_t) \leftarrow \sum_{a_t} \pi(a_t|s_t) \sum_{s_{t+1}, r_t} p(s_{t+1}, r|s_t, a_t) \left[r_t + \gamma v(s_{t+1})\right]$
    $\Delta \leftarrow max(\Delta, |b - v(s_t)|)$
  end for
until $\Delta < \Theta$ (small positive number)

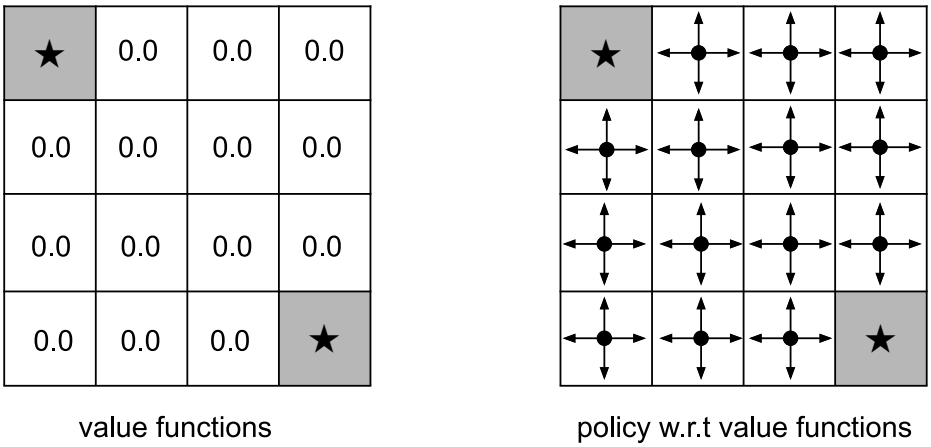


Figure 1.3: *A random policy π w.r.t value of the states*

other part of the process which is *policy improvement*. Since the optimal policy π^* is the policy that will bring the most reward in the long run it is simply the one that is value-function-greedy in its choice of actions. So, it is possible to perform *policy improvement* by selecting actions that lead to highest values of the next states and we can see the final result in the right part of Figure 1.4 that shows the optimal policy taking into the consideration the value functions calculated on the left.

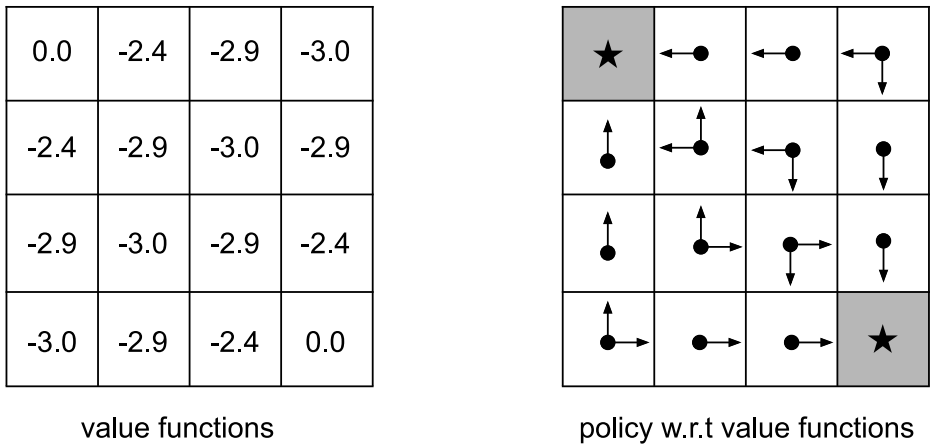


Figure 1.4: *An optimal policy π^* w.r.t value of the states*

1.4.2 Temporal Difference

One of the disadvantages of DP methods is that they require a model of an environment, in other words we need to know the probabilities of landing in a next states s_{t+1} by performing an action a_t on current state s_t which we can see from the $p(s_{t+1}, r|s_t, a_t)$ part of Equation 1.1. *Temporal difference* or TD approaches on the other hand don't require any knowledge of the model as they update the estimates based on the previous learned estimates without waiting for the final outcome. This bootstraping method is shown in Equation 1.2 as the value function for the state $v(s_t)$ is shifted by a small amount α in the direction of TD error which is represented by a difference of our target expectation for the value after the transition which is $r_t + \gamma v(s_{t+1})$ and our previous estimate of the value function given by $v(s_t)$. Our target value for the update is defined by *Bellman equations* which defines the expected value of the state to be an immediate reward received r_t plus the value of the next state $v(s_{t+1})$ discounted by parameter γ.

$$v(s_t) \leftarrow v(s_t) + \alpha \left[r_t + \gamma v(s_{t+1}) - v(s_t) \right] \tag{1.2}$$

Taking it a step further *Q-learning* is a off-policy approach approximating q values instead of value functions. Similar to value functions q values take state-actions pairs instead of states only and determine how good is to be in the state s_t and perform an action a_t. In this way, the target value for the $q(s, a)$ becomes immediate reward received plus the discounted q value of the expected next state and the action that maximizes it. Expectation value is crucial to the learning process as it can tell us

Algorithm 2 Q-learning: and off-policy TD algorithm

Initialize $q(s, a)$ for each combination, and $q(terminal - state, \cdot) = 0$
repeat
  Initialize S
  repeat
    Choose an action a using a policy derived from q (usually ϵ-greedy)
    Take action a and observe reward r and the next state s_{t+1}
    $q(s, a) \leftarrow q(s, a) + \alpha \left[r + \gamma \max_a q(s_{t+1}, a) - q(s, a)\right]$
  until S is terminal
until Final episode is reached

how wrong we were in the previous estimate of the q value. This difference between our previous estimation and our expectation is called *temporal difference* or TD error and is indicative of how far the system is wrong with the current estimate of the value of the state-action pair. Knowing this error, It is possible to perform the main learning loop outlined in Algorithm 2 by updating the predictions for the state-action pairs making them closer to the expectation.

$$q(s, a) = \mathbb{E}\left[r_t + \gamma \max_{a_t} q(s_{t+1}, a_t) | s, a\right] \tag{1.3}$$

1.4.3 Scaling Up

The techniques for solving learning problems using iterative updates of DP have their limitations when faced with high dimensional or continuous state spaces because of the need to memorize a table of the value functions for all of the states and action combinations. This becomes infeasible if our state input is for example grayscale values of every pixel of atari game; although not continuous, the gray scale values representing a single pixel are integers ranging from 0 to 255 and this leads to an enormous number of possible states.
To cope with this we borrow a method commonly used in supervised learning which uses a function to approximate the value function based on the state and action values on its input. The simplest form of the approximation function is to represent a value of state-action pairs by a weighted linear combination of the state features like in Equation 1.4 where w_n represents a weight and f_n represents the n-th feature of the state input. This represents the simplest model of an *artificial neuron* called *perceptron* which was introduced by Rosenblatt as early as 1957. The basic idea is simple, a weight signifies how much, and in which direction the corresponding state feature influences our approximated q, and tweaking those weights enables the system to improve the approximation or representation of the input. All of the weights are updated according to Equation 1.5 by similarly nudging it towards the TD error in the proportion of its feature value $f_i(s, a)$.

$$q(s, a) = w_1 f_1(s, a) + w_2 f_2(s, a) + ... + w_n f_n(s_a) \tag{1.4}$$

If we were to use *temporal difference* method, we would simply look up the state-action pair in our table and nudge it by α in the direction of the TD error in order to update our belief about the value and perform learning. However, when using a function approximator like the simple linear combination of weighted features, learning consists of updating its weights in order to better represent q value at the output.

$$w_i \leftarrow w_i + \alpha \left[r_t + \gamma \max_{a_t} q(s_{t+1}, a_t) - q(s, a) \right] f_i(s, a) \tag{1.5}$$

The basic principle is to minimize the total error defined in Equation 1.7 as the square difference between target value or observation after transition i given by $y = r_t + \gamma \max_{a_t} q(s_{t+1}, a_t)$ and our prediction $\hat{y} = q(s, a)$.

$$error = \sum_i (y_i - \hat{y_i})^2 = \sum_i (y_i - \sum_k w_k f_k(x_i))^2 \tag{1.6}$$

To find out how we can minimize the received error on just one point x we can take a derivative with respect to some specific weight w_m as shown in Equation 1.7. From this we can infer that if we want to minimize the error function we should change the weights in the direction away from the derivative of the error function w.r.t the weight w_m which corresponds to the learning update already defined in Equation 1.5.

$$\frac{\partial error(w)}{\partial w_m} = - \left(y - \sum_k w_k f_k(x) \right) f_m(x) \tag{1.7}$$

1.4.4 Beyond the Perceptron

In order to form an *artificial neural network* or ANN a non-linearity can be introduced in the linear representation of *perceptron* given in Equation 1.4; the easiest way to do this is to apply a simple non-linear function on top of it like *Rectified Linear Unit* or ReLU shown in Equation 1.8. This is called *activation function*.

$$ReLU(x) = max(x, 0) \tag{1.8}$$

A neuron is defined in Equation 1.9 as the function that takes some vector on the input and returns its non-linear *activation function* defined over the linear combination of weighted input vector.

$$f(x_1, x_2, ...x_n) = max(0, w_1 x_1 + w_2 x_2 + ... + w_n x_n) \tag{1.9}$$

A simple neural network architecture is shown in Figure 1.5, it consists of three fully connected layers, one of which represent the input of two variables x_1 and x_2, which do not perform any calculation. We can apply a different activation function to the last neuron such as non-linear sigmoid for example. The final result is a complex, non linear function that is parametrized by a set of weights Θ as shown in Equation 1.10.

$$f(x_1, x_2) = Sigmoid\big(w_1^3 ReLU(w_1^1 x_1 + w_2^1 x_2) + w_2^3 ReLU(w_1^2 x_1 + w_2^2 x_2)\big) \tag{1.10}$$

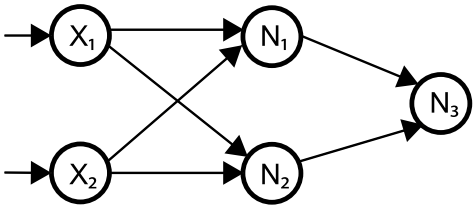


Figure 1.5: *A simple example of an artificial neural network*

In order to train the network to approximate our new observation for q value given by Bellman equation given by $y = r_t + \gamma v(s_{t+1})$ and make our prediction $\hat{y}$ closer to the real y, a parameter update on Θ is performed so that it will minimize the loss function $L(\Theta)$ shown in Equation 1.11.

$$L(\Theta) = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2 \tag{1.11}$$

$$\nabla_{\Theta_i} L_i(\Theta_i) = (y_i - Q(s, a; \Theta_i))\, \nabla_{\Theta_i} Q(s, a; \Theta_i), \tag{1.12}$$

where $y_i = r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \Theta_{i-1})$ is in fact the Bellman equation defining the target value.

Equation 1.12 formally represents the full update using *Stochastic Gradient Descent* or SGD for approximating $q(s, a, \Theta) = q^*(s, a)$. A SGD represents a method for solving a minimization problem by updating the parameter vector Θ consisting of weights w. In order to find out how will an update on a specific weight parameter w_i influence the change in the loss function L, it is possible to consider its derivative with respect to that parameter like in Equation 1.13.

$$\Delta L \approx \partial \frac{\partial L}{\partial w_i} \Delta w_i \tag{1.13}$$

We would like to minimize the loss function L, so the weight w_i can be modified in the direction that is opposite of the derivative by a small number α called *learning rate*.

$$w_i = w_i - \alpha \partial \frac{L}{\partial w_i} \tag{1.14}$$

This method is called gradient descent because we move away from the gradient which is the vector of partial derivatives which is pointing to the maximum increase of the function. Gradient elements are the derivatives with respect to the weights as shown in Equation 1.15.

$$\left(\frac{\partial f(\Theta_t)}{\partial \Theta_{t,1}}, \frac{\partial f(\Theta_t)}{\partial \Theta_{t,2}}, ..., \frac{\partial f(\Theta_t)}{\partial \Theta_{t,n}} \right) \tag{1.15}$$

An intuitive approach to use a neural network as approximator for the Q function in reinforcement learning would be to use a neural network that takes state s and

action a as the input and predicts $q(s, a)$, but due to usually small amount of actions it would be, in fact, better to predict the q values for all available actions as shown in Figure 1.6. The improved architecture shown on the right will make it possible to get all possible q values for the given state in one forward pass of the approximator; these q values can be used to predict the next action using ϵ-greedy policy and the part of y target value taking the discounted $\max_{a_t} q(s_{t+1}, a_t)$.

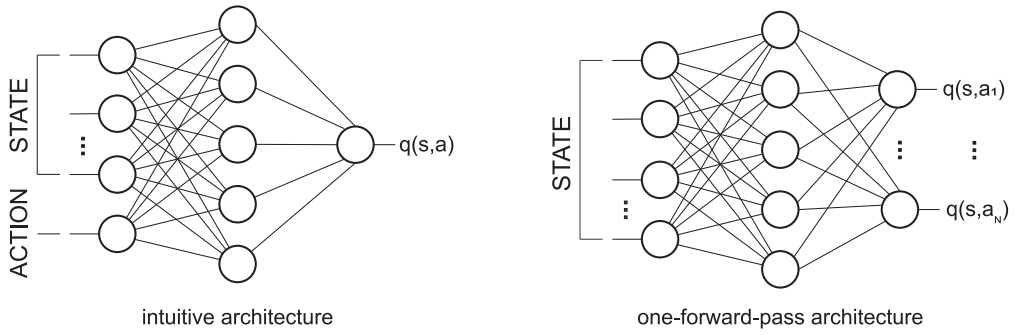


Figure 1.6: *Comparison between different ways to approximate* $q(s, a)$

As we can see from Algorithm 3 the learner is iteratively performing the transition in the form of a tuple (s_t, a_t, r_t, s_{t+1}) and updates our q value approximator. This represents a main *learning loop* shown in part (b) of the model outlined in Figure 1.7. It also showcases a mechanism of *experience replay* that is the focus of our discussion.

Algorithm 3 Q-learning with function approximation

Initialize replay memory D with capacity N and sampling rate S
Initialize action-value function q with random weights
for episode = 1, M **do**
  for t = 1, T **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \arg\max_a q^*(s_t, a; \Theta)$
    Execute action a_t, observe reward r_t and state s_{t+1}
    if $i \bmod S = 0$ **then**
      Store transition (s_t, a_t, r_t, s_{t+1}) in D
    end if
    Sample random batch of transitions (s_t, a_t, r_t, s_{t+1}) from D

$$\text{set } y_i = \begin{cases} r_i, & \text{terminal } s_{i+1} \\ r_i + \gamma \max_{a'} q(s_{i+1}, a'; \Theta), & \text{non terminal} \end{cases}$$

    Perform a gradient descent step on $(y_i - Q(s_i, a_i; \Theta))^2$ according to Equation 1.12
  end for
end for

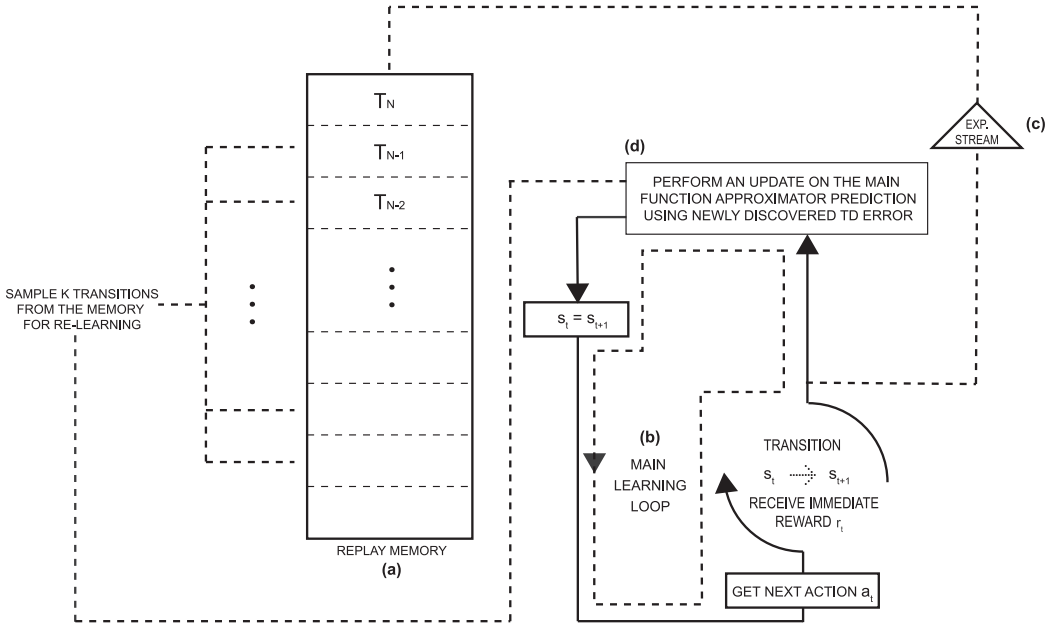


Figure 1.7: *General learning model architecture of q-learning with function approximation:* (a) *Replay memory stores the last* N *experiences in a sliding window buffer for later replay;* (b) *Main learning loop consists of:* $1)$ *the transition in which the agent performs an action, receives an immediate reward* r_t*, while transitioning to a next state* s_{t+1}*;* $2)$ *performing an update on main function approximator ANN* (d) *by performing a gradient descent on the loss function shown in Equation 1.12;* $3)$ *shifting the states for the next iteration in which the* s_t *becomes our* s_{t+1}*;* $4)$ *forwarding the current state through a function approximator in order to find out the best action* a_t *candidate based on its* q *value for* ϵ*-greedy policy;* (d) *A block implementing q-value function approximator detailed in Figure 2.2 taking the starting state* s_t *on the input and predicting q-values for each of the available actions on its output;* (c) *Raw stream of the experiences that are perceived representing unfiltered cognition of an agent.*

CHAPTER 2

The Perception Model

2.1 Origins

The main idea of *active learning* is that a learner is not bound just to process the data that it perceives in order to update its knowledge about the environment, but it is also capable of choosing the data from which to learn, therefore achieving greater efficiency in learning with fewer training data. This opens an possibility for the learning agent to not only control data processing, but also data perception from the environment in order to achieve different outcomes. The main hypothesis presented here is that the agent's learning process is not shaped only by the data, but also by the way they are perceived, and then used, much like the perception process in humans can greatly affect the structure of the brain, given its important property of *neuroplasticity*.

Advances in Artificial Intelligence have sparked the interest in developing systems that perform, learn, and think similarly to human beings. Furthermore, modern machine learning algorithms are now more than ever taking inspiration from the physiology of human brain, as it is particularly evident in approximation techniques of Deep Q-learning and its replay memory mechanism, which is functionally related to the hippocampus, a center for memories in the mammalian brain. Similarly, in modern approaches to DQL, the stream of experiences are not directly propagated to the learning mechanism in order to create *Temporal Difference* (TD) error that is used to update the *Artificial Neural Network* (ANN) function approximator. Instead, they are mediated by a specific mechanism, called *Experience Replay* [36, 39], which we can see in Figure 1.7 that is used to store some of the past experiences in a sliding window memory in order to make it possible for an agent to learn by replaying them, without the need of accumulating further direct experience, a particularly time con-

suming process. This reinforcement learning mechanism is functionally similar to what happens in mammalian hippocampus, which is deemed to store the most valuable experiences during periods of activity, in order to be able to learn from them while the person is sleeping or relaxing [34].
Recent discoveries in neurosciences and cognitive sciences [34] show that the human neocortex is not a slave to the stream of the experiences from the environment, and that the hippocampus is goal-depending and biased in replaying stored experience for re-learning. We exploit this specific property of the hippocampal memory mechanism in order to create an analogue structure called *Cognitive Filter*. This filter structure will act as a mediator between the agent's raw sequential stream of experiences and its replay memory by selectively choosing the memories that are stored using memory prioritization [50].
Since the introduction of the replay memory mechanism in Deep Q-Learning or DQL [39] many works have been aimed at further improving the efficiency of learning by focusing, or giving priority, to certain types of experiences over others, both in sampling and replay. One of the first successful approaches [50] used stochastic prioritizing on the experiences stored in replay memory with high *Temporal Difference* (TD) error under the assumption that high TD error of experience transition would make the training faster because of its higher deviation from the current approximated q-value for the state-action pair.
Another approach [64] further argued that uniform sampling performed by [50] may suffer from loss of potentially valuable transitions with higher TD error especially in the beginning of the learning process when the transitions with rewards are mainly the ones that account for high TD error levels. Instead of uniform sampling, their approach was to sample all the transitions into two separate memory replay buffers: one containing the transitions with the immediate reward, and the other the rest of the transitions. Stochastically sampling from the two memory buffers with different priorities allowed them to reach a learning speed higher than in [50].
The influence of focusing on the transitions that will increase the speed of convergence to the optimal policy π^* is significant and scales well to all types of problems, but if we want to explore the possibility of allowing the attention focus change the behavioral characteristics of a learning agent, we need to adopt a more biologically inspired point of view. Focusing attention is an expensive process for humans, as it requires spending a certain amount of psychic energy, since our brain tries to filter out the noise from the narrow signal that is the focal point. Our brains try to conserve as much of the psychic energy as they can while there's no specific need to spend it. The perception as an attention focusing mechanism relies on this principle. When we are in a state of hunger for example, we will tend to focus our attention on the specific sources of food in our environment, much more than when are in a specific homeostasis and our needs are satisfied. We can thus consider attention as a function of an agent's state, or mood, that is driven by a specific hierarchically extended or higher goal.

2.2 Adaptation Through Artificial Perception

2.2.1 Introduction

Reinforcement learning is a process aimed at reducing uncertainty about which actions an agent is required to take at each time step in order to better adapt to its environment. This process of adaptation is driven by a specific feedback the agent receives from the environment, usually provided by a reinforcement function ranging on some signals from the environment. The main goal of the agent is to create a mapping from its sensed state to the probability of selecting an action that would represent the best choice to maximize the received reinforcement in the long run. This probability distribution is called an optimal policy π^* and it is a result of an iterative update of the previous policies π over the learning process.

Given that we may have multiple optimal policies π^* all of them achieving optimal behavior in different ways, through selecting different actions,it might be possible to influence the agent behavior by further narrowing the action selection, whilst still keeping it focused on the main goal of maximizing the reinforcement. This work exploits this possibility by modifying the agent's behavioral characteristics by changing the way the agent perceives the feedback it receives from its immediate environment; this creates an additional secondary drive that augments the main one given by the reinforcement function itself. The perception itself is not represented by the data collected by the agent but as the way the feedback from its immediate environment is encoded and managed by the learning mechanism. Influencing the way that the data is encoded we can modify the dynamics of the agent's perception and further influence its behavioral characteristics. Due to the increase in state space dimension, modern approaches to reinforcement learning rely on a function approximator, such as artificial neural networks, to approximate the state-action values. The proposed approach takes an advantage of the dynamics of the *stochastic gradient descent* algorithm which is used to perform the learning update on the weights of the artificial neural network at each time step in order to better approximate the values. An agent learns from a sequential stream of experiences and after each perception it updates its belief about the optimal state-action value $Q^*(s, a)$ by performing a *gradient descent* on the parameters or weights of neural network Θ in order to minimize the square error of the previous estimate and the expected value of Q for the perceived state s and the taken action a. For the accurate approximation a *gradient descent* should be performed on every dataset point and, in our case, this means computing an algorithm not just on the current experience, but on the whole history of experiences the agent perceived so far, which is computationally impractical for most implementations. To alleviate this approximation problem a *minibatch stochastic gradient descent* [36] is used which instead of re-learning from all of the experiences selects a batch of few random experiences from a sliding window buffer of experiences called *replay memory* and re-learns from this batch at each time step. The dynamics of determining which experiences to store and replay using minibatch approach form the basis for the implementation of the *artificial perception* mechanism that represents the focal point of this work.

A reinforcement learning mechanism becomes concerned not only with the infor-

mation that its environment provides, but also with the way this information is perceived and this perception factor becomes the additional secondary drive that could further reduce the agent's uncertainty during the learning process. The process of a secondary drive driven by the perception mechanism allows for a further and more subtle modification of an agent's behavior without interfering with the primary reward maximization behavior. Similarly in humans we see different dispositions to a range of behaviors that alter they way they achieve their primary goals. These dispositions are known as emotional states and they are a product of the different physical characteristics of human brains altering the chemical compositions that govern our behavior. For example, extroverted and introverted individuals can achieve the same primary goal of enjoying, but in a different way; the first will most probably engage in social activity, while the latter will prefer to stay home, away from people. This section proposes an algorithm to manage perception able to model agents with a specific set of *emotional states*, similar to human ones, that will enable them to better adapt and thrive in environments with different or changing characteristics.

2.2.2 Related Work

A work by Rumbel et al. [49] provides a good introduction and review of the work that dealt with the emergence of emotional states in artificial learning agent's.
One of the first approaches in modeling emotion in reinforcement learning is presented in work by Gadanho and Hallam [25] implementing a bottom-up model with four basic emotions of sadness, happiness, fear and anger that are induced by a combination of feelings and a simple artificial hormone system. Instead of providing a secondary drive goal as outlined in the approach presented here, the main role of the emotions in this work is the decomposition of the main drive into more basic parts which can be activated by emotions and thus become more manageable separately.
In the work by Ahn et al. [1] the emotions are induced by their components of valence and arousal, which, similarly to our approach, takes into account the cognition flow of the agent's experience stream. The valence component takes into consideration the history of previous experience to calculate the anticipation of the reward, and it is positive when a choice is expected to give a reward that is higher than expected, and negative if that is not the case. The arousal component increases with the cognitive uncertainty associated with the decision.

2.2.3 Model Architecture and Learning Algorithm

An agent performs a transition in the environment by moving from state s_t to next state s_{t+1} using action a_t and experiencing a scalar reward r_t in the process. These observed values contain enough information to perform an iteration of a learning process by updating our previous belief about the value $Q(s, a)$ of the state-action pair. This update is performed at (d) block of Figure 1.7 as a part of the main learning loop (b) and it consists of updating the weights vector Θ of artificial neural network detailed in Figure 1.6 making our approximation of Q a step closer to the target value as given by Equation 1.3. The process creates an experience stream of sequential transitions the agent has experienced from the beginning of the learning process. Each of these experiences carries a potential for re-learning that would be lost if

they are discarded after the initial update; so, instead, they are recycled through a perceptive buffer called *replay memory*. Due to the limited capacity of the memory buffer this form of perception, just like in humans, should be able to be selective over types of experiences that are being sampled. This type of a "reality filter" is able to influence how does the agent gather information from its environment and further interacts with it to reach its goal. The premise of this approach is that the contents of the *replay memory* that represents a subset of the *experience stream* form the perceptive layer that is able to provide a secondary drive that in turn influences the agent's behaviour. The idea of perception being a filter of experiences that protects the agent from oversaturation is implemented through stochastic sampling shown in the *artificial perception* box included in block (f) of Figure 2.1. As we can see from Algorithm 4 the *artificial perception* block is implemented as a probability of sampling the transition $P(i)$ into *replay memory* structure D.

Algorithm 4 Q-learning with artificial perception block

Initialize replay memory D with capacity N
Initialize action-value function Q with random weights
for episode = 1, M **do**
  for t = 1, T **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \arg\max_a Q^*(s_t, a; \Theta)$
    Execute action a_t, observe reward r_t and state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in D according to the probability $P(i)$
    Sample random batch of transitions (s_t, a_t, r_t, s_{t+1}) from D

$$\text{set } y_i = \begin{cases} r_i, & \text{terminal } s_{i+1} \\ r_i + \gamma \max_{a'} Q(s_{i+1}, a'; \Theta), & \text{non terminal} \end{cases}$$

    Perform a gradient descent step on $(y_i - Q(s_i, a_i; \Theta))^2$ according to Equation 1.12
  end for
end for

2.2.4 Experimental setup

Environment

To evaluate our model we have applied in Algorithm 4 in a relatively complex environment called *Waterworld*, from the ReinforceJs framework [30].
The environment consists of food pieces that are generated as they are consumed, with a random position and velocity vector. They move freely in the environment. There are two types of food: good and bad. We have performed different sets of experiments, exploring the effect of different amounts of food. In the first one, the number of bad and good food pieces in the environment are the same, namely 25 of each type. In the second one, the proportion of good and bad food is brought to 2:1 and 1:2. In the third one, the amount of food generated is two pieces for each of the two types of food.

The agents aim at discovering an optimal policy that would allow them to eat (touch) the most good food pieces, while avoiding as much as possible the bad ones. The reinforcement is, respectively, +1 for the good, and -1 for the bad food sources that the agent can reach. The environment contains equal amounts of good and bad food pieces and the ratio is kept constant by regenerating the consumed piece of food. The agent perception includes 30 directional sensors, each of which can detect 5 continuous variables: distance and speed in x and y direction of the different perceived objects that include good food, bad food, and distance to wall or the boundaries. This, with an agent's own speed components in x and y direction accounts for a quite high dimensional state space of 152 continuous variables.
Approximation of $Q(s, a; \Theta) \approx Q^*(s, a)$ is done using an ANN with one hidden fully connected layer of 100 neurons which are producing as output the Q values of all five actions available to the agent: up, down, left, right, stay. The learning rate of an approximator α is set to a low value (0.05) and the capacity of the replay memory buffer was 5000 elements. The value of ϵ was set to 0.2 at the beginning and adjusted to 0.1 at the mid-point of the learning to exploit more of the learned behavior. The discount factor γ is set to 0.9.

Perception Dynamics

In order to see how does the *artificial perception* component contribute to behavioral characteristics and adaptation, a total of six emotional states are outlined in Table 2.1 each of them corresponding to a different perception dynamics given by their experience sampling strategy. The sampling strategies are inducing dynamics by modifying two dimensions of the experience stream: the type of the transition given by the reward obtained and the amount of transitions sampled which is illustrated in Table 2.2. The rows of the table are showing the transition type which is *good* if the reinforcement is positive, *bad* if it's negative and *random* if the agent's doesn't care about the reinforcement. A low or high amount of each type of experience can be sampled according to what reported on the columns of Table 2.2. This gives us a total of six combinations of perception dynamics; each one corresponding to an agent's emotional disposition.

2.2.5 Experimental Results

Experiments compared the adaptation ability of six agent types implementing different perception dynamics or sampling strategies in three variations of environment ranging from *hostile* to *benevolent*. The adaptability of each agent was measured by a total cumulative reinforcement received, or how well it performed in the given environment.
Normal environment which contained equal amounts of positive and negative reinforcement sources showed a divergence in agents score implementing different perception dynamics as we can see from Figure 2.3. We can also notice that in this type of balanced environment the conservative sampling strategy of the *provident* agent type proved to be most fruitful to adaptation, while the optimistic *happy* agent was the worst. Agents with *cautious* and *greedy* attitude performed slightly better than the conservative one, but random sampling strategy of *sad* one still underperformed

Table 2.1: *Mapping of agent's emotional states and sampling strategies.*

Emotion state	Strategy
happy	There is something good in any experience. It selects a lot of experiences, randomly.
sad	There is always something bad in new things. It selects randomly few experiences.
fearful	It is afraid to get bad things. It selects a good number of negative experiences, so to learn to avoid them.
greedy	Always get as much as possible. It selects a lot of good experiences, discarding the bad ones.
cautious	It aims for good, but it tends to be conservative. It selects few good experiences.
provident	This is ready to face problems, also conservative. It selects few negative experiences to learn from them.

Table 2.2: *Agent emotional state types represented as combinations of two sampling dimensions with columns representing the quantity of sampled transitions and rows represent the type.*

amount/type	few	many
good	cautious	greedy
bad	provident	fearful
random	sad	happy

when compared to the second best *fearful* strategy.
Hostile environment showcased in Figure 2.4 which was mostly populated with negative reinforcement (ratio good/bad food is 1:2) shows even more divergence of the agent types than the normal one. This harsh environment have proved to be best faced by the agents that adapted a *fearful* strategy allowing them to thrive and perform far more than the other types. The conservative strategies of *provident* agents also showed adaptability scoring second best followed by an ultra-optimistic *happy* one. The strategies that haven't adapted to this kind of environment were the one's that focused on the positive experiences such as *cautious* and *greedy* and the conservative random approach of *sad* agent.
Figure 2.5 compares the performance in a benevolent environment containing more positive than negative reinforcement (ratio 2:1). This environment variation was faced in a similar way by all of the perception dynamics, except the optimistic *happy* agent that underperformed despite the high availability of the positive food sources.

2.2.6 Discussion

Looking at the all-round performance across the environment variations it seems that conservative, cautious strategies like *provident* and *fearful* perform best when it comes to adaptation. The fearful approach to perception seems to give a good consistent performance in all of the experiments in the setup, suggesting that the negative experiences cognitively play a more important role than the positive ones in the process of learning. Even in environments that are scarce in positive reinforcement like

the hostile one, focusing on the negative experience still gives a performance that is equal or better than the other sampling strategies. Not surprisingly, the *fearful* agent performs the best in an environment that is hostile by focusing on avoidance rather than exploration but interestingly provides an effective strategy also for the environments that are by nature more supportive.
The sampling strategy implemented through *provident* agent's narrower selection of negative experiences is insightful of the nature of perception itself. Its superior performance in the baseline, well balanced Normal environment gives us a clue about the fact that perception is inherently highly selective of experiences and that it can be best represented as a filter with respect to the environment. Acting as a mediator of agent's cognition it can protect its cognitive gateway of limited capacity, the replay memory, from being oversaturated with experiences. The oversaturation having a negative effect on the learning process can be seen in the low performance of the *happy* strategy across all of the environment variations. The "happy" selection of a lot of random experiences oversaturates the replay memory quickly and this has a great effect on the agent's performance.
Cautious and *greedy* strategies of focusing on the good experiences proved to be a good approach in the Normal environment, where they outperformed the random ones of *happy* and *sad*, but due to the lack of positive reinforcement in hostile world they lacked in the performance.
Overall the agents that focused their perception on the positive experiences showed a more exploratory behavior, while the ones that focused on the negative were more conservative and static.

2.2.7 Conclusions

We presented a model of *artificial perception* capable of modifying the way an agent processes information from its environment during learning. The perception layer creates additional possibility for influencing the agent's behavior which can be used as a secondary goal oriented drive. Using this technique, an agent is able to better adapt to learning in a specific environment only by changing the dynamics of its perception without modifying its reinforcement function.
Looking beyond the adaptation *artificial perception* creates a contextual filtering buffer between the agent's cognition and the learning algorithm, possibly providing more efficient data exploitation from the environment and preventing the mechanism of replay memory to be oversaturated and ineffective.

Table 2.3: *Agent sampling strategies probability values used in experiments.*

amount/type	few	many
good	p(.05) or (if(r=1) and p(.5))	p(.05) or (if(r=1) and p(1))
bad	p(.05) or (if(r=-1) and p(.5))	p(.05) or (if(r=-1) and p(1))
random	p(.15)	p(.5)

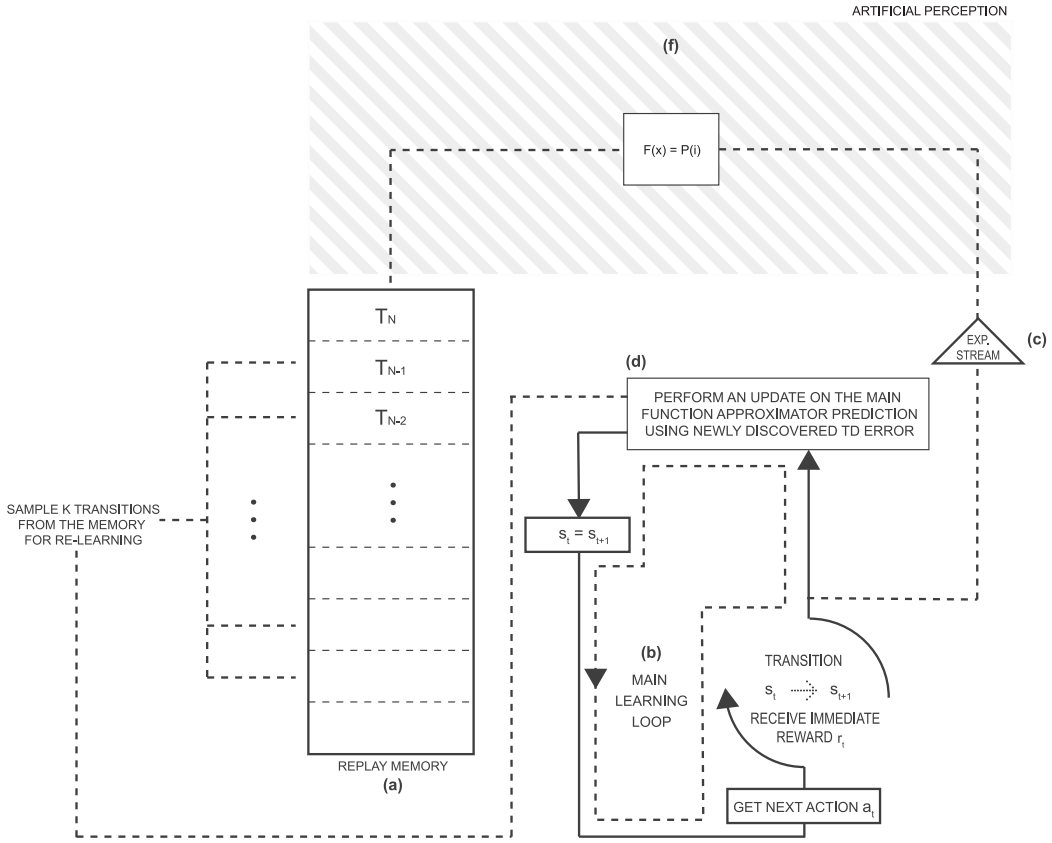


Figure 2.1: *General learning model architecture including attention focus block:* (a) *Replay memory stores the last* N *experiences in a sliding window buffer for later replay;* (b) *Main learning loop consists of:* $1)$ *the transition in which the agent performs an action, receives an immediate reward* r_t*, while transitioning to a next state* s_{t+1}*;* $2)$ *performing an update on main function approximator ANN* (d) *by backpropagating the TD error as a gradient of the* a_t *output;* $3)$ *shifting the states for the next iteration in which the* s_t *becomes our* s_{t+1}*;* $4)$ *forwarding the current state through a function approximator in order to find out the best action* a_t *candidate based on its Q value for* ϵ*-greedy policy;* (d) *A block implementing Q-value function approximator detailed in Figure 2.2 taking the starting state* s_t *on the input and predicting Q-values for each of the available actions on its output;* (c) *Raw stream of the experiences that are perceived representing unfiltered cognition of an agent;* (f) *Artificial perception block implemented as a function that determines whether* i*th transition will be sampled into replay memory taking the sampling probability as a parameter*

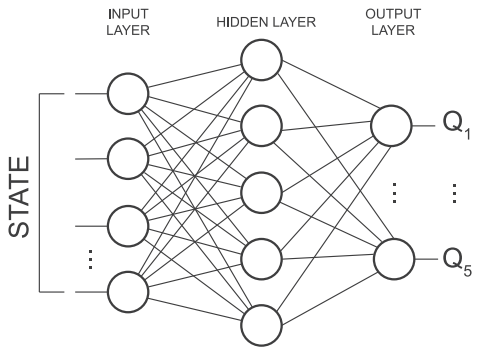


Figure 2.2: *Main function approximator ANN implemented in the (d) block of Figure 2.1: it receives a 152-dimensional state as its input and approximates it to Q values of each 5 possible actions available to an agent at its output, therefore providing an approximation for $Q(s, a)$ pairs.*

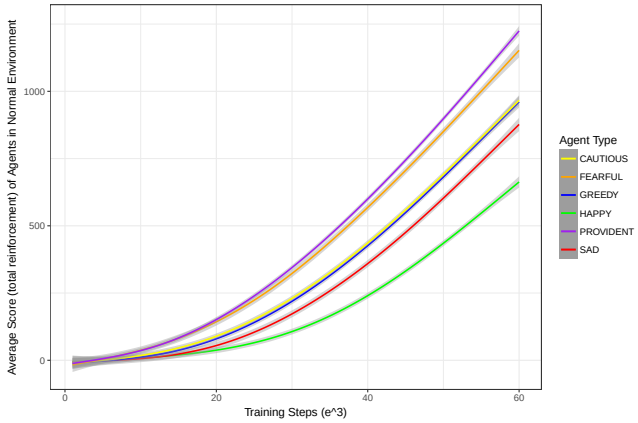


Figure 2.3: *Average score or total reinforcement received over 50 trials with agents with different perception dynamics in Normal environment type during first 60.000 learning steps.*

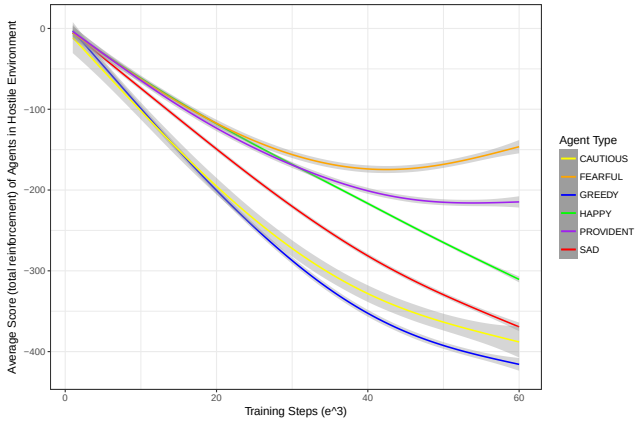


Figure 2.4: *Average score or total reinforcement received over 50 trials with agents with different perception dynamics in Hostile environment type during first 60.000 learning steps.*

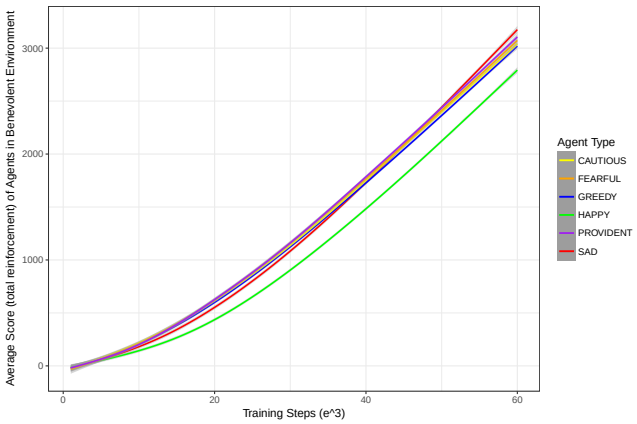


Figure 2.5: *Average score or total reinforcement received over 50 trials with agents with different perception dynamics in Benevolent environment type during first 60.000 learning steps.*

CHAPTER 3

Character Matters

Human individuals differ in their emotional, attitudinal, experiential and motivational styles. Personality psychology helps us to better understand the scope and the multitude of these variations using a central principle called *trait*. A *trait* is a single continuum over which individuals differ and each of them can be interpreted as a continuous variable. The main idea behind this simplification is the possibility of describing some possible drivers for the complex generation of human behaviors with a finite number of discrete types, or traits, which levels vary from human to human. In our machine learning model, we would use the term continuous valued *feature* instead of *trait* to denote the same concept. Measuring the traits is not a straightforward, direct process. Instead, we infer the level of the trait by observing whether the individual is consistently inclined towards a type of behavior over time.
Throughout the history, many theory candidates for the traits have been offered, but at the beginning of the Eighties of the last century, many researchers from many different backgrounds agreed that there are five basic factors, or personality dimensions, found in natural language, theoretically based questionnaires, in self-reports and in ratings [19, 38]. The proposed five factor model organizes the personality traits in five dimensions, reported in Table 3.1, each of them associated to a set of Adjectives and Scales. The Adjective column contains a list of items defining the factor in a study of 280 men and women during an assessment weekend at the Institute of Personality Assessment and Research [29]. Similarly, from the Scales column we can see Revised *Neuroticism Extraversion Openness*, or NEO, inventory facet scales from self reports of 1536 adult men and women [15].
Recent discoveries brought forward by advancement of neural imaging techniques such as *fMRI* made evident that the five basic personality traits have physiological origin in the structure of an individual's brain. It allowed us to detect that the areas

for controlling emotional responses vary in size, shape, baseline activity and more important activation intensity among individuals with different traits. The correlation between the activation of brain regions and the personality traits has been most evident in personality dimensions of Neuroticism, Extraversion and to a lesser extent Conscientiousness. The brains of individuals with different personality traits are actually differently hardwired and this physiological differences lead to the variation found in the higher-level order psychological functions that in terms shape the behavioral response [42]. The reason why people react differently when faced with the same situation lies in the different activation threshold of the actual physical regions of the brain that together contribute to the emergence of varying behaviors. This chapter shows how the differences in the psychological mechanism of attention can lead to reinforcement learning agents responding differently to the situations, or states and this can lead to an overall behavioral changes commonly found withing the different personalty dispositions without modifying the main reinforcement function.

Table 3.1: *Adjectives and Questionnaire scales defining the Five Factors Model [38]*

Name	Number	Adjectives	Scales
Extraversion (E)	I	Active Assertive Energetic Enthusiastic	Warmth Gregariousness Assertiveness Activity
Agreeableaness (A)	II	Appreciative Forgiving Generous Kind	Trust Straightforwardness Altruism Compliance
Conscientiousness (C)	III	Efficient Organized Planfull Reliable	Competence Order Dutifulness Achievement Striving
Neuroticism (N)	IV	Anxious Self-pitying Tense Touchy	Anxiety Hostility Depression Self-Consciousness
Openness (0)	V	Artistic Curious Imaginative Insightful	Fantasy Aestetics Feelings Actions

We make use of the insights about the way a mammalian brain processes information by modifying the underlying mechanisms of Reinforcement Learning (RL) accordingly, in order to mimic the differences in brains that may account for emergence of different personality types in humans. As a reference model of human personality traits we take one of the most widely accepted frameworks, the Five Factor Model or FMM, which organizes personality traits in terms of five basic dimensions: Extrover-

sion, Agreeableness, Conscientiousness, Neuroticism, and Openness to Experience.

3.1 Modeling Openness to Experience

In the work we present in this section, we focus only on the facet possibly most related to human cognition: Openness to Experience or OTE. Individuals that score higher in the Openness to Experience scale have greater permeability of consciousness and perceptive cognition, are more curious and insightful about their surroundings and more motivated to seek variety and experience.
As a core part of the *Cognitive Filter* structure, we implement a novel prioritization criteria that is using specific properties of agent's sensed state space called Information Gain, or IG, given by relative Shannon's entropy of the two transitioning states s_t and s_{t+1}. Agents that are more Open to Experience will favor the experience transition that lead to the increasing of the relative entropy between two states, while agents that are low on the scale will favor the transitions that reduce it. This approach differs from the prioritization techniques found in [50, 64] in the sense that its sampling criteria takes into account important properties of the agents sensed space during the transition instead of the *TD error* that a specific transition yields.
We present results of experiments where this relatively simple selection mechanism of experiences to be replayed can evolve agents showing different personality traits, more or less suited to learn and perform in different types of environments.

3.1.1 Model Architecture and Learning Algorithm

In this section, we present the structure of the learning model we are proposing, whose schema is reported in Figure 3.1.
In section (a) of Figure 3.1, we can see the main learning loop, which represents the core of our Q-learning mechanism. It performs the transition from state s_t to state s_{t+1} taking an action a_t and possibly receiving an immediate reward r_t. The whole transition is defined by a tuple vector (s_t, a_t, r_t, s_{t+1}) .
Once the action is performed and the reward is obtained, an agent performs the learning process on a single transition (s_t, a_t, r_t, s_{t+1}) by generating its new estimate of the Q-value for being in a state s_t and taking an action a_t. This process performs a forward pass on the approximator (b) with s_t on input, after which we select the predicted Q-value on the output a_t. After this, the states are shifted so our old s_{t+1} becomes the new s_t and the agent is ready for the next iteration. TD error represents the discrepancy between the previous estimate and the expected target Q-value after the transition, which is given by Equation 1.3, considering its received reinforcement value r_t and the discounted maximum Q-value of the next state s_{t+1}. The learning process consists in the update of the estimate for $Q^*(s, a)$ produced by the function approximator by performing an SGD on the weights Θ in order to minimize the loss function $L(\Theta)$, which, in this case, is represented by a squared TD error.

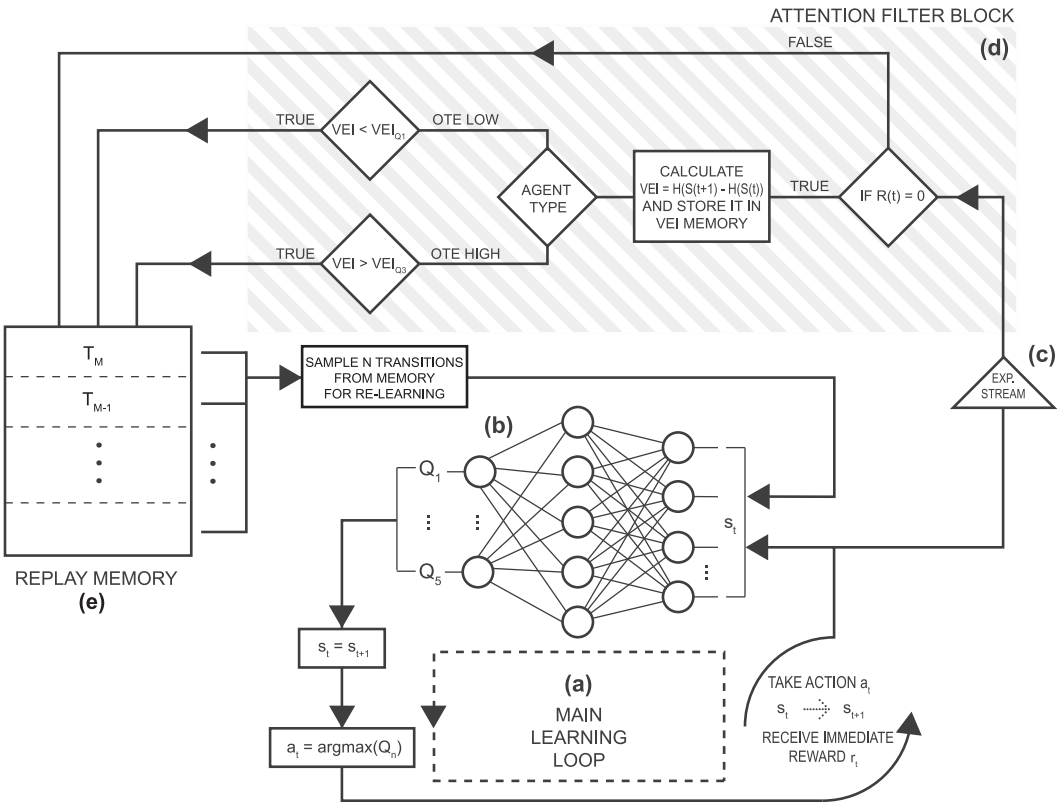


Figure 3.1: *Architecture of the proposed model, showing the role of the cognitive filter block component.*

Since the nature of the learning loop (a) is sequential, we may risk to produce highly temporally correlated transitions that could not satisfy the assumptions of most gradient descent-based, non-linear function approximators, like ANN [39]. Some of the transitions after each learning iteration are stored into sliding-window memory shown in section (e) of Figure 3.1. From the mechanism of replay memory the past transitions can later be replayed allowing the agent to re-learn from its previous experiences. This solves the temporal-correlation problem and it also guarantees a more stable training of the approximator.

Cognitive filter

Element (c) from Figure 3.1 represents the sequential stream of experiences produced by the learning loop. It would be highly inefficient to sample the whole stream into the replay memory (a), especially because some of the transitions may be more valuable than others [48, 64]. Instead of sampling the whole stream of experiences, we introduce a mechanism called *Cognitive Filter*, shown in section (d) of Figure 3.1. This is in charge of selecting the elements that are stored for replay. It consists of two main parts, governed by two different selection criteria. The first one is simple: it selects all the transitions with a non-zero reward that will give rise to the highest TD error, especially at the beginning of the learning process [64]. The rest of the transitions are then evaluated by our newly introduced, cognitively inspired, *Variety*

of Experience Index or VEI criterion, which takes into account the properties of the agent's transitioning states. The introduced criterion (VEI) represents the tendency of the agent to gain variety in experience. We define it as a difference of Variety of Experience between the starting state s_t and the state that an agent has transitioned to, s_{t+1}.

Quantifying the Variety of Experience of the states

In order to quantify the possible information gain that a state space vector can carry, we have adopted Shannon's entropy as a measure of diversity, also called Shannon's index. The state space vector is represented by a number M of variables that are continuous and normalized in $[0..1]$. In order to measure the entropy, each of the M state space variables are discretized into N bins, calculated by Equation 3.1, where p_i is the frequency of values belonging to the ith bin.

$$H(s_t) = -\sum_{i=1}^{N} p_i log_2 p_i \tag{3.1}$$

Learning Algorithm

Previous prioritization algorithms [64] used a stochastic sampling method in order to make the learning faster and more efficient, which falls between uniform sampling and greedy sampling based on the TD error.
In our approach, instead of focusing on the TD error, we introduce a prioritization based on the *Variety of Experience Index* (VEI) criterion, able to model the behavioural characteristics of learning agents. As we can see from Equation 3.2, VEI criterion represents the difference between the entropy of the two states involved in a transition: s_{t+1} and s_t; it is defined as a relative entropy or Kullback-Leibler distance between the posterior and prior state. This implies that, when an agent is transitioning from a lower entropy state to a higher entropy state, the VEI is positive, while if the transition is reducing the entropy between states its VEI is negative.

$$VEI = H(s_{t+1}) - H(s_t) \tag{3.2}$$

We can now define two types of behaviorally modified agents, depending on the fact that their prioritization criteria respectively focus on positive or negative extremes of the VEI parameter. The positive values of VEI correspond to the agent that is high on the Openness to Experience scale, because it is preferentially using to replay, and thus to learn from, transitions that bring it towards the more "interesting" states, the ones that may provide a higher information potential; this makes it behave more curiously. The agent that lies on the low end of the Openness to Experience scale is doing the VEI prioritization in the opposite way: it tends to focus on experiences with negative, smaller values of the parameter, so it tends to be more conservative and less curious.

Instead of greedy sampling on VEI values, which may make the system prone to over-fitting because of lack of diversity [50], we define a stochastic prioritization based on the *Variety of Experience Index* VEI where the probability of sampling

the $P(i)$ transition from the sliding window experience memory D is determined by Equation 3.3.

$$P(i) = \frac{VEI_i^\beta}{\sum_{j=1}^{j=size(E)} VEI_j^\beta} \tag{3.3}$$

In this case,VEI represents the priority of the transition and the parameter β, ranging on $[0..1]$, determines how much this prioritization is used; in the uniform case, $\beta = 0$.

To alleviate the selection of a value for β, which would need to be tweaked for the specific application, we introduce a more general prioritization technique based on the descriptive statistical property of quartiles that can be used in a broader sense with no additional adjustments.
In order to sample basing on the VEI criterion, instead than on the stochastic approach given by Equation 3.3, we use a descriptive statistic approach that considers quantiles, in particular, in the experiments reported in this paper, the first and third quartiles (respectively referred to as VEI_{Q1} and VEI_{Q3}) of the VEI values of agent's experiences stored in a sliding window memory E of capacity n.

Algorithm 5 selectively stores the transitions after each update step based on VEI criterion. The criterion is used to determine the transitions that have the *Variety of Experience Index* VEI contained in the specific quartiles of the sample depending on the type of the agent. In our experiments, the agent with higher *Openness to Experience* will sample the transitions with VEI higher than the third quartile, or Q3, of the n latest samples from E, given by $VEI_i > VEI_{Q3}$. In contrast, the agent that is low on the *Openness to Experience* scale will sample the transitions lower then the first quartile determined by the conditional $VEI_i < VEI_{Q1}$.
After each transition a random batch of the so-selected transitions is selected from the replay memory D where samples were stored, in order to perform additional training on the approximator.

Algorithm 5 Deep Q-learning with cognitive filter block component

Initialize replay memory D with capacity N and VEI experience memory E
Initialize action-value function Q with random weights and agent type $OTE_a = (LOW, HIGH)$
for episode = 1, M **do**
  for t = 1, T **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \arg\max_a Q^*(s_t, a; \Theta)$
    Execute action a_t, observe reward r_t and state s_{t+1}
    Calculate the transition value VEI_i based on Equation 3.2 and add it to the sliding window VEI memory E
    if $OTE_a = LOW$ **then**
      Identify VEI_{Q1}
      if $VEI < VEI_{Q1}$ **then**
        Store transition (s_t, a_t, r_t, s_{t+1}) in D
      end if
    end if
    if $OTE_a = HIGH$ **then**
      Identify VEI_{Q3}
      if $VEI > VEI_{Q3}$ **then**
        Store transition (s_t, a_t, r_t, s_{t+1}) in D
      end if
    end if
    Sample random batch of transitions (s_t, a_t, r_t, s_{t+1}) from D

$$\text{set } y_i = \begin{cases} r_i, & \text{terminal } s_{i+1} \\ r_i + \gamma \max_{a'} Q(s_{i+1}, a'; \Theta), & \text{non terminal} \end{cases}$$

    Perform a gradient descent step on $(y_i - Q(s_i, a_i; \Theta))^2$ according to Equation 1.12
  end for
end for

3.1.2 Experimental setup

To evaluate our model we have applied in Algorithm 5 on two relatively complex and different environments, selected among the ones used in the machine learning community. The first one is representative of a high dimensional state space with sparse reward feedback called *Waterworld*, from ReinforceJs framework [30], while the second is a more realistic setup with a more evenly distributed reinforcement feedback called *LunarLander-v2* from *OpenAI Gym* framework [10].
In the experiments, we have compared two types of prioritized sampling algorithms with two different prioritization criteria, OTE_{LOW} and OTE_{HIGH}, associated to agents that are on the lower and higher end of *Openness to Experience* axis, respectively. We have performed two different batches of 50 independent learning trials, for each agent type, OTE_{LOW} and OTE_{HIGH}, and we show the results as the arithmetic mean of the scores over the 50 trials.

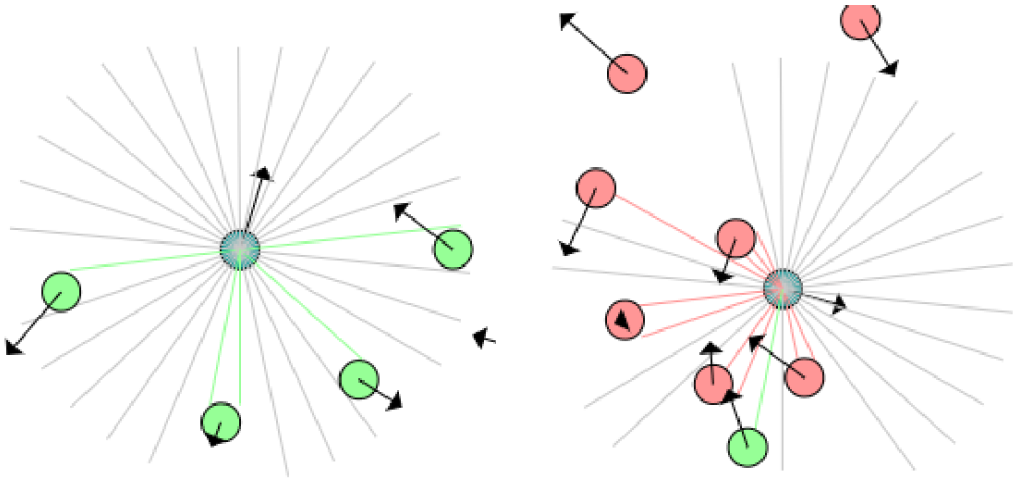
Waterworld Environment

The *Waterworld* environment consists of food pieces that are generated as they are consumed, with a random position and velocity vector. They move freely in the environment. There are two types of food: good and bad. The agents aim at discovering an optimal policy that would allow them to eat (touch) the most good food pieces, while avoiding as much as possible the bad ones. The reinforcement is, respectively, +1 for the good, and -1 for the bad food sources that the agent can reach. The environment contains always the same amount of good and bad food pieces: their ratio is kept constant by regenerating the consumed piece of food. We have performed different sets of experiments, exploring the effect of different amounts of food. In the first one, the number of bad and good food pieces in the environment are the same, namely 25 of each type, giving an average density of 1.78% of the maximum possible amount.In the second one, split in two sub-experiments, the proportion of good and bad food was brought respectively to 2:1 and 1:2, while keeping the same density. In the third one, the amount of food generated was two pieces for each of the two types of food, thus providing a quite sparse reward.
The agent perception consists of 30 directional sensors, each of which can detect 5 continuous variables: distance and speed in x and y directions of the closer perceived object, together with its identification as good food, bad food, and distance to wall or the boundaries. This, together with an agent's own speed components in x and y direction accounts for a quite high dimensional state space of 152 continuous variables.
Approximation of $Q(s, a; \Theta) \approx Q^*(s, a)$ is done using an ANN with one hidden fully connected layer of 100 neurons which are producing as output the Q values of all five actions available to the agent: up, down, left, right, stay. The learning rate of an approximator α was set to a low 0.05 and the capacity of the replay memory buffer was 5000. The value of ϵ was set to 0.2 at the beginning and adjusted to 0.1 at the mid-point of the learning process to exploit more of the learned behavior. The discount factor γ was set to 0.9.

Variety of Experience criterion comparisons

In order to evaluate how does *Variety of Experience Index* of the transition VEI relate to the behavioural characteristics of the *Openness to Experience* personality trait we are comparing transition examples from *Waterworld* environment on the two extremes of this parameter, which correspond to the prioritization focus of the two agent types: OTE_{LOW} and OTE_{HIGH}, respectively with low and high Openness to Experience. In Figure 3.2 and Figure 3.3, each of the detected objects and agents are depicted, together with their speed vector, which represents the composition of their x and y speed components as described in the state space.
Figure 3.2 shows some of the transitions with a low (negative) value of VEI used as the prioritization criterion for the type of agent associated with low level of *Openness to Experience* or OTE_{LOW}. This type of agents favor the transitions which have entropy values of the starting state s_t higher than that of the end state s_{t+1}. This is evident in the behaviours shown in Figures 3.2a and 3.2b, where we can see that the agent has the tendency to move away from the experience that is represented by the

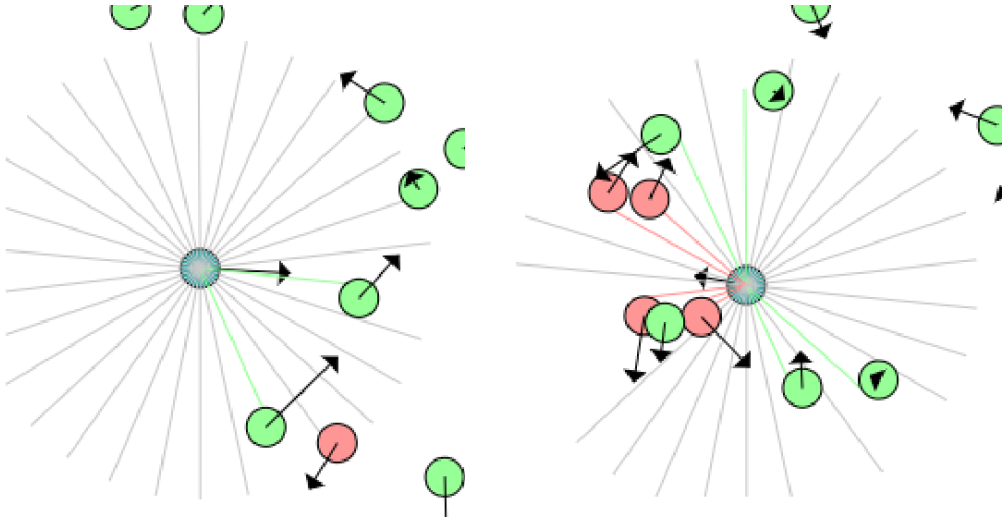
moving food clusters.



(a) $VEI = -0,1528$ **(b)** $VEI = -0,2083$

Figure 3.2: *Transition examples in Waterworld environment with VEI values lower than the average. The agent is represented by the blue circle, from where the sensing directions depart, food is represented by green circles (good food), or red circles (bad food), together with their speed vectors.*

On the higher (positive) end of the VEI spectrum (OTE_{HIGH}), we can see from Figures 3.3a and 3.3b the behavior of the agent, which has a tendency to move toward the experience since the entropy values of the end state, s_{t+1}, are higher than those of the previous one, s_t.



(a) $VEI = 0,2872$ **(b)** $VEI = 0,3064$

Figure 3.3: *Transition examples with agents having VEI values considerably higher then the average. It is possible to notice how the agent tends to go towards food.*

Lunar Lander Environment

In this environment, a craft is attempting to land on a designated landing area defined over an rugged lunar terrain by firing three of its thrusters as depicted in Figure 3.4. The craft has four discrete actions at its disposal: do nothing, fire main thruster, fire left orientation thruster and fire right orientation thruster.
In contrast to the *Waterworld* environment the Lunar Lander is an episodic task; an episode ends if a craft crashes or comes to rest receiving an additional reward of -100 and +100 respectively. The continuous reinforcement received is proportional to how close the craft is to the zero speed and inversely proportional to the distance from the landing area. Firing main thruster results in -0.3 additional reinforcement each frame but the fuel available to the craft is infinite. Each leg contact with the ground is rewarded with +10.

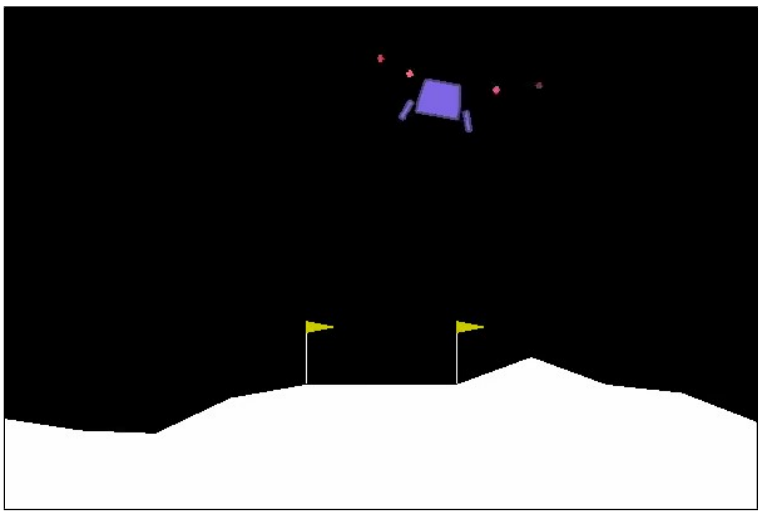


Figure 3.4: *A single rendered frame from Lunar Lander environment depicting a craft firing its thrusters in order to land on the designated area marked by two yellow flags.*

3.1.3 Experimental results

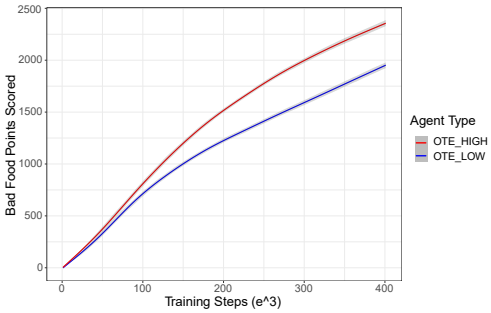
In this section, present the experimental results for the experiments done in the two environments.

Waterwold Environment

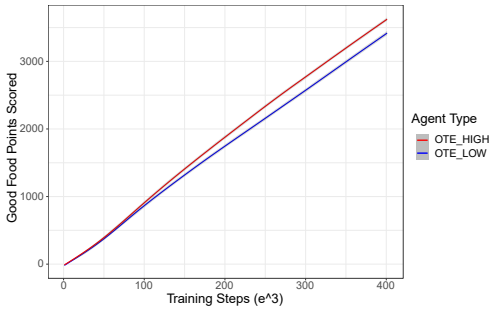
In the Waterworld environment, we have performed different experiments to evaluate the performance of the two types of agents in the different environmental conditions mentioned in subsubsection 3.1.2.

Negative vs. Positive reinforcement

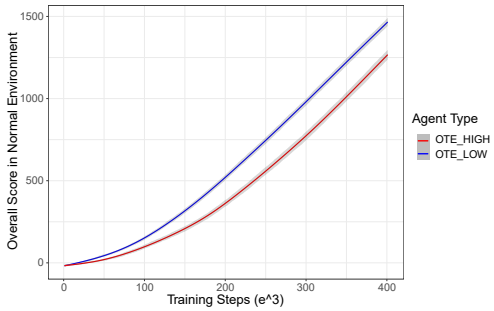
In this first set of trials, we wanted to compare the possibility to collect positive and negative reinforcement of the two agent types, given respectively by the amount of good and bad food collected. Figure 3.5 shows the comparison between the results



(a) *Total number of bad food pieces collected corresponding to the cumulative negative reinforcement received.*



(b) *Total number of good food pieces collected corresponding to the cumulative positive reinforcement received.*



(c) *Total scores, i.e., sum of positive and negative reinforcement received.*

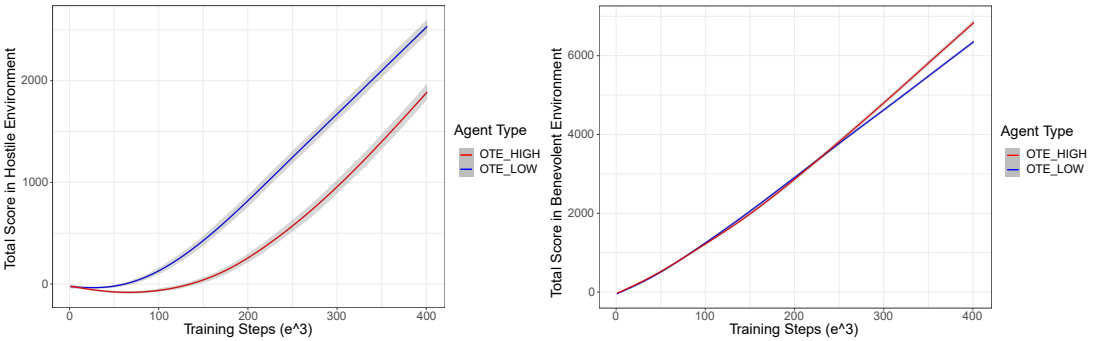
Figure 3.5: *Comparison between cumulative rewards obtained the first 400.000 learning steps, respectively, by OTE_{HIGH} and OTE_{LOW} agents for bad (3.5a) and good (3.5b) food sources, in the Waterworld environment. The plots report the average of the values over 50 trials, In these and the following plots, the standard deviation of the plotted values over the 50 trials is also plotted, in gray, around the average lines. Scaling varies for each plot in order to make differences more evident.*

obtained by the two different prioritization criteria applied to the *Waterworld* experimental setup.

We can see that the two agent types have developed different behaviours from the total scores shown in Figure 3.5a for bad food pieces and in Figure 3.5b for the good ones. Agents that are high on the *Openness to Experience* scale, marked as OTE_{HIGH}, score more on both good and bad food points, thus demonstrating the behaviour characteristics of taking more risk due to the tendency of moving towards areas where it is possible to accumulate more experience, in this case, food pieces. Agents that are lower in *Openness to Experience* OTE_{LOW} behave differently from their counterparts, by scoring much lower values of bad food sources but also lower values of good food sources, so they behave in a more cautious way. The personality trait of lower *Openness to Experience* gives them advantage in this setup as their overall score (i.e., the difference between the number of good food pieces collected, and the number of bad food pieces) is higher than that of the agents that are conditioned higher on the trait as we can see from Figure 3.5c.
Mean standard deviations for the batch of trials whose rewards are reported in Figures 3.5a and 3.5b are presented in Table 3.2, in order to give a better feeling of the

Table 3.2: *Average standard deviation σ values for experimental results shown in Figure 3.5*

	OTE_{LOW}	OTE_{HIGH}
Bad food sources	10,674	10,796
Good food sources	14,751	13,325



(a) *Cumulative, total reinforcement received in Hostile environment.*

(b) *Cumulative, total reinforcement received in Benevolent environment.*

Figure 3.6: *Comparison between total cumulative reinforcements, obtained in the first 400.000 learning steps in two variations of environment, respectively by OTE_{HIGH} and OTE_{LOW} agents, averaged over 50 learning trials of the Waterworld environments. Scaling varies for each plot in order to make differences more evident.*

variability that was found in the trials, since it may be difficult to appreciate it from the gray shades in the plots.

Adaptation to different environments

To study the suitability of the behaviors developed by agents with different traits to different environments, we have performed trials on two extreme variations of the *Balanced* environment used for the first experiment, reported in subsubsection 3.1.3. While the *Balanced* environment was characterized by an even distribution of positive and negative reinforcement, on one end of the extreme we have a *Hostile* environment which contains 2:1 ratio of bad to good food pieces, and on the other extreme a *Benevolent* environment in which the distribution of food is the inverse in favor of good pieces. Here, we are focusing on the agent's overall performance, which is defined as the cumulative reinforcement received over the trial, both negative, corresponding to the number of bad food pieces consumed, and positive, corresponding to the consumed good ones. We take these results as an indicator for the adaptability of an agent to a specific type of environment.
The overall performance of different types of agents, OTE_{LOW} and OTE_{HIGH}, is compared in both environment variations and outlined in Figure 3.6; the corresponding standard deviation values are reported in Table 3.3.

From Figure 3.6a we can see that the cautious nature of the OTE_{LOW} agent has shown a significant degree of adaptability to the *Hostile* type of environment, which is evident by the difference of 25.2% of the mean total score compared to curious OTE_{HIGH}.

On the other side, considering the overall performance of the different agent types

Table 3.3: *Average standard deviation σ of values of total, cumulative reinforcement obtained in 50 trials in Hostile and Benevolent environments, whose trend is shown in Figure 3.6.*

	OTE_{LOW}	OTE_{HIGH}
Hostile environment	30,834	26,936
Benevolent environment	23,495	27,076

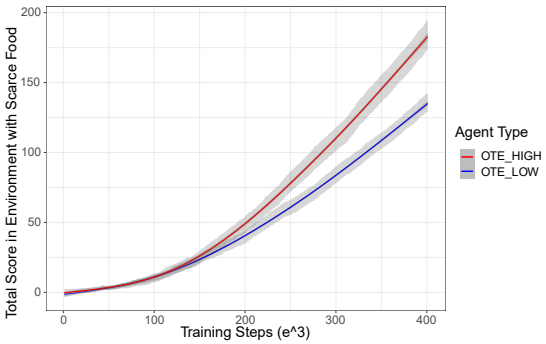


Figure 3.7: *Comparison between total, cumulative scores (sum of positive and negative reinforcement), obtained respectively by OTE_{HIGH} and OTE_{LOW} agents in the first 400.000 learning steps, averaged over 40 learning trials in the Waterworld environment with Scarce Food.*

in the *Benevolent* environment, we can see in Figure 3.6b that this type of environment gave a curiosity oriented OTE_{HIGH} agent a difference of 7.1% of the mean total score over the cautious OTE_{LOW} type.

Environment with Scarce Reinforcement

One of the common obstacles in reinforcement learning is the scarcity of reinforcement: in many cases the agent doesn't receive enough feedback from its environment, in the form of reward, in order to converge to its optimal policy in an acceptable time. We have tested the adaptability of the two agent types in a variation of the environment that provides a significantly lower amount of food than the *Balanced* one. This environment consists of only four food sources: two of them provide a negative reinforcement (bad food), while the other two give a positive reinforcement (good food). Scarce reinforcement also meant that the agents had to adopt a more exploratory approach in order to maximize their cumulative reward in the long run. Figure 3.7 shows the comparison of overall performance for the OTE_{LOW} and OTE_{HIGH} agent types, given by the cumulative reinforcement received in the scarce food environment. The corresponding standard deviation values are reported in Table 3.4. These results show that the agents having an OTE_{HIGH} trait, given their curious nature, performed better and shown more ability to adapt to an environment with scarce reinforcement then OTE_{LOW} type of agents.

Table 3.4: *Average standard deviation σ of values of total reinforcement in environment with scarce food, whose trend is shown in Figure 3.7.*

	OTE_{LOW}	OTE_{HIGH}
Environment with scarce food	4,0191	5,6944

Lunar Lander Environment

Total of 20 trials for each of the OTE types were performed over first 32 episodes in a more realistic Lunar Lander environment with the same learning parameters and memory capacity as the previous ones. Experimental results plotted in Figure 3.8 indicate that the agent characterized by a cautious nature of OTE_{LOW} allowed for an overall better adaptation than the more curious OTE_{HIGH}, which had some advantage only at the beginning of the learning process.

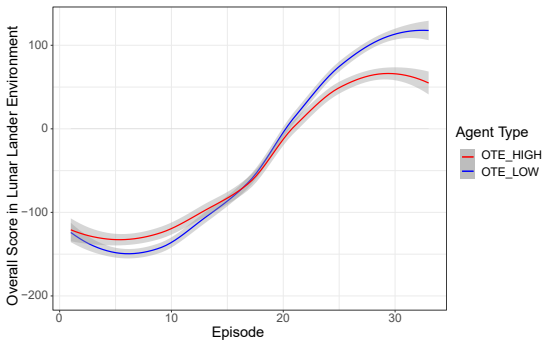


Figure 3.8: *Comparison between total, cumulative scores (sum of positive and negative reinforcement), obtained respectively by OTE_{HIGH} and OTE_{LOW} agents in the first 32 learning episodes, averaged over 20 learning trials in the Lunar Lander environment*

3.1.4 Discussion

Experiments were performed on different variations of the *Balanced* environment, to investigate about the possibility for an agent to gain a better performance by showcasing a specific, behaviorally conditioned strategy for each environment type. Results obtained from these experiments show that our algorithm is capable of conditioning a better adaptation strategy to common scenarios we encounter in machine learning, which include environment variations with different distributions of positive and negative reinforcement seen in subsubsection 3.1.3 and environment variation with reduced amount of available reinforcement, as reported in subsubsection 3.1.3. Our *Balanced* environment provided an initial baseline benchmark since it contained an even distribution of positive and negative reinforcement in a good amount. Results reported in subsubsection 3.1.3 show that this environment seems to support more the cautious nature of an agent that is low in *Openness to Experience* trait (OTE_{LOW}), which seems to develop a behavior that relies more on avoiding the bad food pieces, rather then aiming at collecting the good ones. On the other side of the *Openness to Experience* spectrum, the OTE_{HIGH} trait seems to drive the agent to focus on getting the most of food, either good or bad.

As it appears from the results in the environments with a different food generation rate, the curious OTE_{HIGH} agent thrived better in an environment providing twice the amount of positive reinforcement, while the cautious tactics of OTE_{LOW} took advantage of the environment that contained more bad than good food sources giving it a far better performance (a difference of 25.2%) than that obtained by the other type.

We can see another significant difference of performance in the environment with scarce reinforcement, as reported in subsubsection 3.1.3. Here the agent that is high on the *Openness to Experience* scale outperformed the low one by a total 26.3%. Curious OTE_{HIGH} agent seemed to take more risk in this scenario by actively seeking out the food instead than being primarily focused on avoiding negative food like the OTE_{LOW}, and this proved to be the best behavior for this environment.
The risk taking strategy proved to be a disadvantage in a Lunar Lander environment, which enables different dynamics, and, instead, supported the more conservative approach of agents that are on the low scale of the *Openness to Experience*. Although OTE_{HIGH} was able to gain an early advantage during the first episodes by exploring the state space more efficiently, the OTE_{LOW} strategy proved to be a better one overall by cautiously reducing the amount of engine firing which was expensive in negative reinforcement.

3.2 Modeling Extraversion

3.2.1 Cognitively Inspired Architectures

Studies have showed that human cognitive processes utilized during the interaction with the environment are mediated by a memory buffer called working memory [3]. The working memory keeps a temporary storage of the perceived information needed to perform a complex cognitive task: it acts as a connecting mechanism between perception and long term memory.
Experiments have identified that the differences between individuals in the capacity of working memory [22] and the breadth of attention generally influence the way they are focusing their attention and creative abilities [31]. The term "breadth of attention", in this context, refers to a sort of cognitive bandwidth, i.e., the number and scope of stimuli that one is attending at a time.
Extroverted individuals tend to have an attention broader than the introverted ones, which, in turn, tend to focus their attention to a narrower subset of stimuli in order to reduce the cognitive load of having a higher basal arousal level [23, 35].

3.2.2 Model Architecture and Learning Algorithm

Using only uniform sampling as a way to store experiences in the replay memory proved to have limitations such as that some of the valuable experiences might never be replayed [64]. Breadth of Attention-based replay memory keeps the uniform sampling and extends it by additionally sampling the experiences that emerged from a specific type of interaction. For the purpose of mapping the transition to a specific, goal-oriented interaction, we extend the experience description tuple with a transition type indicator or c_t $e_t = (s_t, a_t, r_t, c_t, s_{t+1})$.
The modified replay memory uniform sampling algorithm that we propose, in addition to sampling every Sth sample, samples the experiences that match the subset of transition types, called F (for focus of attention), as shown in Algorithm 6.
Primary contrubution of this algorithm is twofold. The first one is goal focusing, which enables the agent to exibit a secondary goal, one that does not necessarily match the reinforcement function; this may lead to a more complex and dynamic behavioral pattern. The second contribution is the possibility to simulate the differences between Extraverted and Introverted personality types evident in the *breadth of the attention*, by altering the maximum amount of stimuli that is collected into *replay memory* buffer at each given time. We define *Breadth of Attention Sampling* or BAS as a type of *Cognitive Filter* that is able to modify the scope of F and therefore modeling the agents with different behavioral characteristics in both goal- and trait-oriented way, thus making them more adapted to learn in different environments.

3.2.3 Experimental setup

To evaluate the proposed model we have adopted a learning environment that consists of moving good/bad food pieces and multiple agents as described in Section 2.2.4.
As a function approximator we are using a neural network to approximate $Q(s, a; \Theta) \approx Q^*(s, a)$. To reduce the computational complexity of having multiple forward passes

Algorithm 6 Q-learning with Breadth of Attention Sampling

Initialize replay memory D with capacity N and sampling frequency S
Initialize and set transition types index $C = \{c_1, c_2, ..., c_n\}$ and attention focus index $F \subset C$
Initialize action-value function Q with random weights
for episode = 1, M **do**
  Initialize sequence $s_1 = \{x_1\}$ and pre-processed sequenced $\phi_1 = \phi(s_1)$
  for t = 1, T **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \Theta)$
    Execute action a_t, observe reward r_t type of transition t_t and image x_{t+1}
    Set $s_{t+1} = s_t$,a_t,x_{t+1} and pre-process $\phi_{t+1} = \phi(s_{t+1})$
    if $i \bmod S = 0$ **then**
      Store transition $(\phi_t, a_t, r_t, c_t, \phi_{t+1})$ in D
    end if
    for each f in F **do**
      if $c_t = f$ **then**
        Store transition $(\phi_t, a_t, r_t, c_t, \phi_{t+1})$ in D
      end if
    end for each
    Sample random batch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from D

$$\text{set } y_i = \begin{cases} r_j, & \text{terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \Theta), & \text{non terminal} \end{cases}$$

    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \Theta))^2$ according to Equation 1.12
  end for
end for

each time, we want to find an action that maximizes the state-action function $Q(s, a)$; the network takes the state vector s as an input and predicts $Q(s, a)$ for each possible action.
We have adopted the original Q-learning update with a learning rate α set to a low value (0.05) because of the nature of the approximator, and discount factor $\gamma = 0.9$. The default capacity of the replay memory buffer D included 9000 experiences. For comparison with our proposed algorithm, we performed reference experiments where we uniformly sampled experiences every 7th transition. With regards to our experimental environment this sampling frequency provided a balance between the transitions that were sampled uniformly and the ones that were sampled on the basis of attention focus.

We also performed experiments in a multi-agent setting. The multi-agent environment differed from the single-agent one in size and amount of food generated to accommodate up to 7 agents learning simultaneously. Agents in a multi-agent environment had a possibility of social interaction by sharing food with other agents in proximity, as detected by their sensors. If a single agent consumed a positive food piece it shared the full reinforcement reward of +1 to each of the agents found within its range.

3.2.4 Experimental results

In the experiments, we have compared three types of agents implementing different types of focus of *BAS*, with the baseline uniform sampling already proposed in literature, under three different configurations of the environment. The transitions were given a focus type only if they resulted in an interaction, i.e., either a food piece had been consumed or an agent had been perceived. To differentiate between the interactions, we have defined three focus types in $C = \{$*consume-good, consume-bad, social*$\}$. If the transition resulted in a consumption of good food, it was labeled as *consume-good*, if bad food was consumed it was labeled as *consume-bad*, and if it resulted in either sharing or receiving food through social interaction it was labeled as *social*.
Table 3.5 shows which agent personality type is associated with which subset of C.We call this subset *Attention focus* F as it represents the set of type labels on which Algorithm 6 additionally focuses while sampling from the stream of experiences. For instance, the *Introverted - Cautious* agent focuses on consuming good food, i.e., it samples experiences labeled as *consume-good*. Analogously for the others.

Efficiency comparison

In this section we evaluate the efficiency of agents with different configurations of *BAS* with respect to the ability to consume good food pieces and avoid the bad ones in the environment with an equal distribution of good and bad food pieces. The aim is to compare the behavioral differences of the agents and their effect on the performance by two different criteria: ability to avoid the bad pieces of food and the ability to consume the good ones.
In a multi-agent setting, agents with the ability to communicate had the possibility of choosing whether to engage more in social communication, which may provide an indirect reinforcement, or focus more on the exploration of the environment in

Table 3.5: *Personality types and attention focus*

Single Agent Environment	
Agent personality type	*Attention focus (F)*
Introverted - Cautious	consume-good
Introverted - Brave	consume-bad
Extroverted	consume-good, consume-bad
Baseline	-
Multi Agent Environment	
Agent personality type	*Attention focus (F)*
Introverted - Social	social
Introverted - Explore	consume-good, consume-bad
Extroverted	social, consume-good, consume-bad
Baseline	-

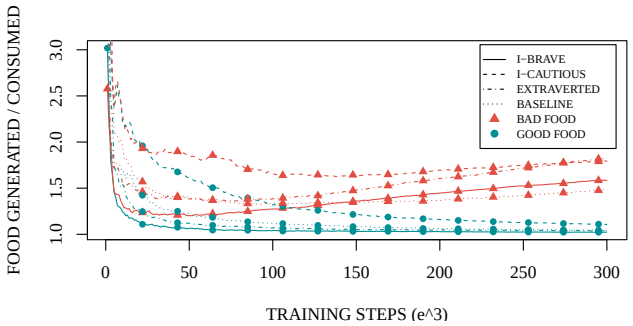
search for more direct reinforcement. Experimental results have shown that agents prioritizing on the transitions that led to social contact can modify their behaviour, making them more inclined to social experiences, while agents that focused on the transitions that are related to exploration showed tendency to stay away from other agents. These two types of agents are used to model the two extremes in the main five factor personality dimension of Extraversion; on the higher end of the scale there is the agent type that gravitates more towards social interaction and, similarly, on the low scale we have the more selfish, explorative agent type. In Figure 3.9 we compare these criteria for each type of agent as a ratio between generated and consumed food pieces. Figure 3.9a shows the average results of 10 experiments done under the same settings for each of the defined agent type, while Figure 3.9b depicts analogous results averaged over 7 agents of the same type interacting in a multi-agent environment.
From Figure 3.9 we can notice that the efficiency of the agents differs depending of the agent type in both single and multi-agent environment. Introverted-Cautious agent type showed to be the most efficient in avoiding bad food sources followed by Extroverted type, while Introverted-Brave outperformed every other type in consuming good food sources. From these results, it seems that focusing on a given aspect pushes to efficiently develop a policy that takes better into account that aspect. We can also notice that BAS agents generally perform better than the non-focused ones.

Performance in different environmental conditions

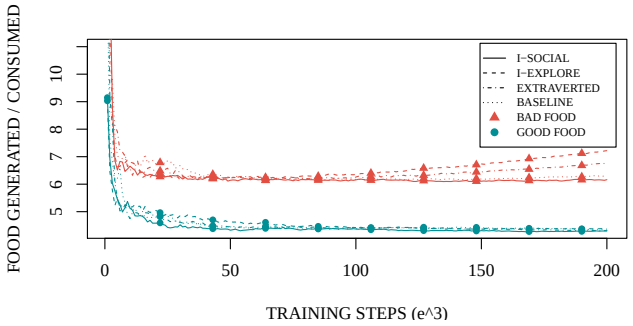
The intention was to explore how can differences in agent personality type impact on the performance under different environmental conditions.

We wanted to answer the question: Can some personality type be more capable than others to learn in a specific environment?
We have modified the equal ratio between the generated good and bad food pieces for the purpose of creating more hostile or more benevolent environment. Benevolent environment generated 2/3 of good food pieces and 1/3 of bad, while the hostile environment had a distribution of 2/3 bad food pieces and only 1/3 good. Results from single agent simulation as depicted in Figure 3.10 show that the Extroverted agent was performing best in both normal and hostile environments, while Introverted-

Figure 3.9: *Differences in ratio between generated and consumed food sources amounts, between BAS focus variations over first 300K learning steps.*



(a) *Single agent environment*



(b) *Multi-agent environment with food sharing*

Brave type better adapted to the environment that contained more good food. It seems that the broader attention span of the Extroverted agent gave it an advantage in the environments that contained higher amount of bad food points. Focusing on both positive and negative experiences allowed the Extroverted agent to learn a policy that was equally efficient in avoiding the bad food points as it was in consuming the positive ones.
Figure 3.11 shows the results from a simulation that included 7 agents interacting by sharing food sources, each of them learning separately. For the normal environment configuration Introverted-Social and Extroverted types were best performing probably because their social focus allowed them to make better use of the available good food points by sharing. Introverted-Explore type outperformed others in a hostile environment mostly because its narrow focusing on the food points rather than social interaction allowed it to be more efficient in avoiding the bad food points.

Implicit vs. explicit goal directed behavior

In the next batch of experiments, we wanted to compare the difference between goal-oriented behavior that is modulated implicitly by *BAS* and the behavior that was explicitly influenced by different reinforcement values. Two additional "baseline" agent types were defined that used only uniform sampling replay memory and dif-

fered only in their reinforcement functions. Baseline social agent was given double value of reinforcement for making a social contact relative to the food, while the baseline exploratory type had double reinforcement for food consumption. In Figure 3.12, we can see the difference in performance between attention-based approaches of modeling social and exploratory behaviors (I-SOCIAL,I-EXPLORE) and the baseline ones (BASE-SOCIAL,BASE-EXPLORE). It is evident that in the hostile environment the BAS exploring agent is better suited to learn faster to avoid bad food, while in the other situations the performance of the different agents is comparable, which means that, at least for these experiments, attention-based replay memory gives the agents the possibility to successfully face different environments, without requiring any special design of the reinforcement function. In particular, in at least one combination, the BAS agents where even able to perform better than the one with modified reinforcement function.

3.2.5 Discussion

Experimental results show that *BAS* can either outperform state of the art approaches on at least some of the environment variations, or have a similar performance. The *BAS* approach makes thus possible to define the focus of attention for an agent and have it performing well in different environments, without the need of re-designing the reinforcement function.
Being able to select the focal experiences by different criteria opens a lot of possibilities for modeling a stream of replay experiences that can potentially give rise to complex behavioral patterns.

3.3 Related Work

3.3.1 Prioritized sampling and replay

Since the introduction of the replay memory mechanism in DQL [39] many works were aimed at improving the efficiency of learning by focusing or giving priority to certain types of experiences over others, both in sampling and replay. One of the first successful approaches [50] used stochastic prioritizing on the experiences stored in replay memory with high *Temporal Difference* (TD) error under the assumption that high TD error of experience transition would make the training faster because of its higher deviation from the current approximated Q-value for the state.
Another approach [64] further argued that uniform sampling performed by [50] suffers from loss of potentially valuable transitions with higher TD error especially in the beginning of the learning process when the transitions with rewards are mainly the ones that account for high TD error levels. Instead of uniform sampling, their approach was to sample all the transitions into two separate memory replay buffers: one containing the transitions with the immediate reward, and the other the rest of the transitions. Stochastically sampling from the two memory buffers with different priorities allowed them to reach a learning speed higher than in [50].

Modeling curiosity

First works about modeling the agent's drive towards more interesting situations appeared in the early Nineties of last century, with the reinforcement learning framework of artificial curiosity [52–54, 58], one of which implementations that are based on prior-posterior entropy [58] is important from the standpoint of our approach. This implementation generates an additional intrinsic reward that is proportional to the predictor informational gain measured by the relative entropy, or Kullback-Leibler divergence, between the learning predictor subjective probability distributions before and after new observations.
Mostly inspired by the psychology of optimal experience or $flow$ [17], arguing that human beings are intrinsically motivated to seek out situations that are just above their skill level therefore satisfying their drive for curiosity, a meta-framework called Intelligent Adaptive Curiosity was proposed in [43]. This approach relies on the selection of the actions from which the agent expects the maximal learning potential.
Some of the lower-level frameworks as [5] take a more bottom-up approach and are as well inspired by the *flow theory* [17], focusing on the fact that it demands to an agent to gravitate towards an equilibrium between its skills and the goal demands. This proposal relies on calculating the novelty of the experience as an order of deviation from the average sensorimotor behavior so far learned by the model.
One the most low level frameworks is presented in [32], an approach that behaviorally modifies the agent by changing its exploration rate in order to better adapt to the new conditions. The exploration rate parameter is influenced directly by the *vigilance*, i.e. the degree of effort to make a decision, which is estimated by a low level predictor.

3.4 Conclusions

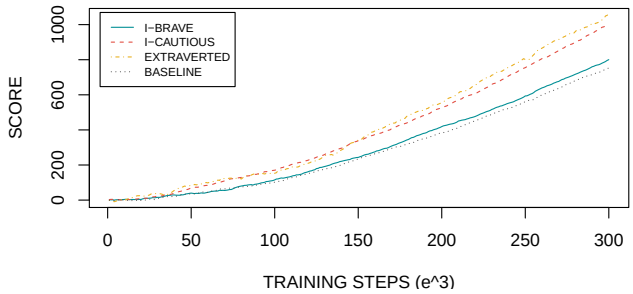
This chapter shows how altering the information processing mechanism of selective attention may lead to behavioral differences that are consistent with the human personality traits found in the widely accepted *Big Five index*. The interest in modeling the traits found in human beings sprang from their evolutionary basis of providing the needed variety of dispositions when forming a group of individuals [42] that, as combined together, elicit a fitness function that transcends the individual ones. For example, there may not be an evolutionary fitness associated with an individual with a high level of *neuroticism*, but, as a component integrated into the group of individuals, it achieves its purpose of supporting the fitness of the whole group. This concept that in human communities the whole is more than a sum of its parts can be supported using agents that are modeled along the personality axis of *openness to experience* showcased in the first part of chapter 3. On one end of the *openness to experience* spectrum we have an agent with a *cautious* disposition that tends to focus attention to the experiences that are leading away from potential information and this accounts for its behavior to be more calm and reserved by not being intrinsically motivated for exploration. Its more lively counterpart on the other extreme of the trait will behave in the opposite way, being *brave* in actively searching for the information gain that would lead it to more interesting, new and unpredictable states while exhibiting much

more aggressive and dynamic motion around its environment. This brings us to the well known trade-off found in reinforcement learning of exploitation to exploration ratio. Sometimes we would like to behave like the "shy" cautious agent type in *exploiting* the more familiar states that we are more certain will bring us more reward in the long run, and sometimes we would like to be like the "brave" one motivated to try out new situations that haven't been fully explored thus it doesn't know whether they are beneficial or not; they have to be visited in order to find out their values. This, conventionally is most often done by adopting an ϵ-greedy approach that basically chooses a random action instead of an action selected by policy π every ϵ times in order to promote exploration of the unexplored states. However, if we are faced with a multi-agent implementation an alternative made possible by agents with modeled *openness to experience* is to use a mixed population with a ratio of ϵ of brave explorative agents combined with the $1 - \epsilon$ of the cautious ones in order to bring us closer to the desired collective behavior. This allows us to define exploration-exploitation properties of the learning process in a much more dynamic and subtle way than using simple ϵ-greedy tactics, with the combined dynamics of the explorative extreme side working along the exploitative cautions one. It also alleviates the need for adjustment of ϵ parameter over time because both types of agent are actually converging to their optimal policies π^* just in different ways.
The concept of application of traits in modeling the collective behavior is taken further in section 3.2 where we apply the same concept of *cognitive filter* to a more complex multi-agent setting in order to elicit both goal-oriented and trait-oriented behaviors. The potential for dynamic behaviors arises from the two equally rewarding ways of getting food represented by a positive reinforcement value: one directly collects reward by aiming at food, and the other interacts with other agents that share their collected food. The first, induced behavioral change is the goal-oriented one, by focusing artificial attention on either experiences that resulted in collecting food on its own by exploring, or focusing on the ones that resulted in getting food from another agent. The first disposition is introversion and the latter extroversion. Both types of agents, the extraverted social exploiter and the introverted explorer are motivated by their primary drive of collecting good food while avoiding the bad one, as given by their reinforcement function, but their behavior as a group is affected by the secondary, more subtle drive of goal-orientation influenced by their attention preference.

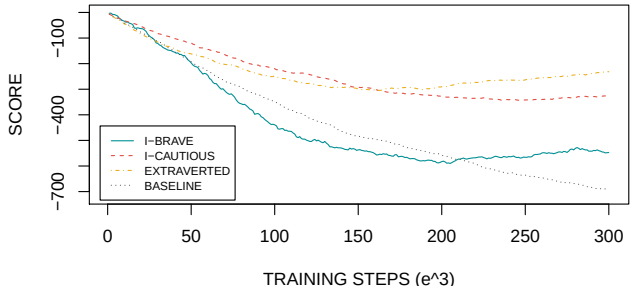
A good evolutionary example of the type separation in a group is the ant colony, where every ant is exhibiting a main drive of hunger and self-sustainability only to be able to perform another specialized function that benefits the whole colony. The example proposed in the previous sections is closer to the situation of a human group than to that of an ant colony as the separations of individual types is not crisp just like the human personality traits are not; however, their dynamics are noticeable in the collective sense. The need of this secondary personality diversity can possibly be evolutionary justified by looking at the two extreme compositions of the presented multi-agent environment. If we had a population consisting entirely of the social exploiters we would soon be faced with the lack of the food to share among the agents and if we had an explorers only population we wouldn't use all of the collected food in the best way by not being generous about it. It can be said that there exists a ratio

of the two types being combined in a specific multi-agent setting that would function as both individually and collectively optimal, by making a better use of the available food resources.
In this chapter, two novel approaches were presented that enable agents to learn behaviors corresponding to different personality traits just by favoring specific types of experiences when sampling them into the *replay memory*. From the reported experiments showcased in both section 3.1 and section 3.2 emerges that this technique seems to be an efficient way of developing a specific personality trait in a learning agent without modifying any other property of the algorithm, or reinforcement function, but only selecting the experiences to be replayed according to a formal model that mimics the concept of *attitude*. The novelty of these approaches mainly consists in the use of the replay memory in a biologically inspired, goal-oriented approach, showing that the way an agent selects experiences to be used for the main massive learning activity can influence the development of a personality trait.
From the reinforcement learning perspective, this type of biologically inspired modification of the selection of the experiences to be replayed opens the path to the development of agents that could be more or less effective in different types of environments, both in learning and in the quality of the obtained behavior, eventually providing a bio-inspired dimension to consider to better adapt anytime learning agents to dynamic environments.

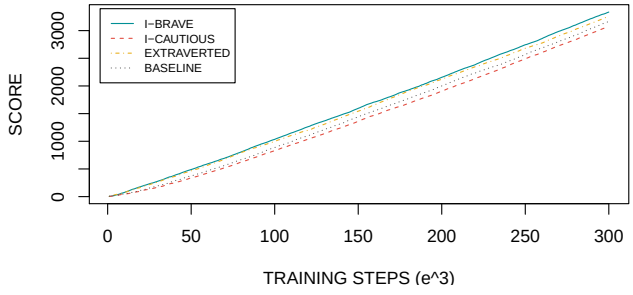
Figure 3.10: *Differences in average score/reward between agents with BAS focus variations, learning in a single agent environment over first 300K learning steps.*



(a) *Normal environment: even number of good and bad food sources*

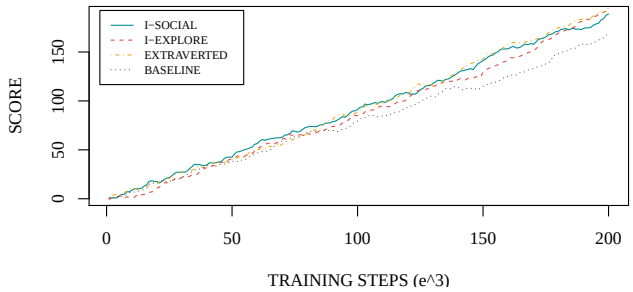


(b) *Hostile enviroment: bad food sources 66.66%, good food sources 33.33%*

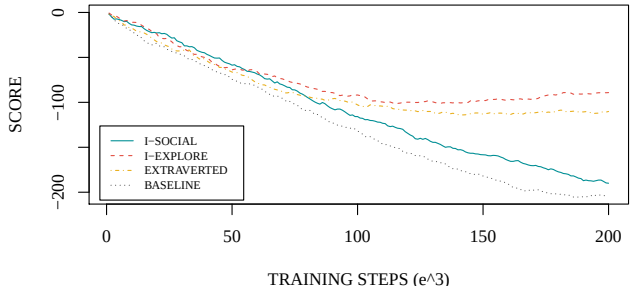


(c) *Benevolent environment: bad food sources 33.33%, good food sources 66.66%*

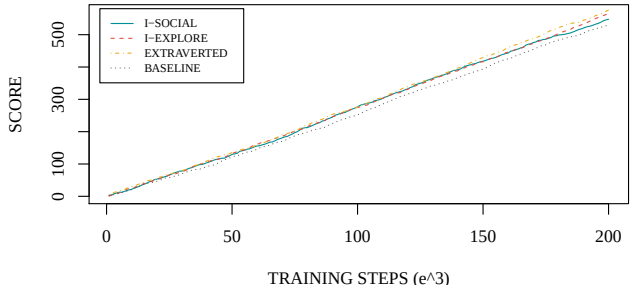
Figure 3.11: *Differences in average score/reward between agents with BAS focus variations learning in a multi-agent environment over first 300K learning steps.*



(a) *Normal environment: even number of good and bad food sources*

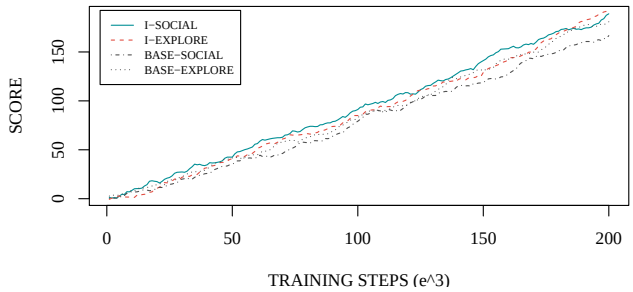


(b) *Hostile environment: Bad food points 66.66%, good food sources 33.33%*

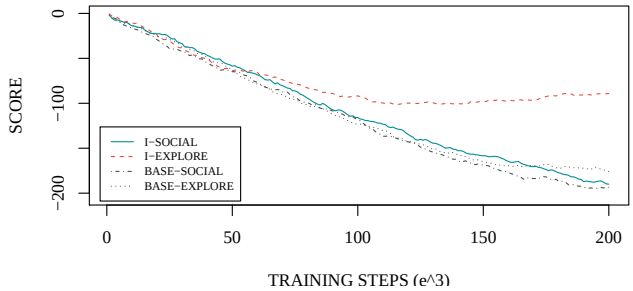


(c) *Benevolent environment: Bad food points 33.33%, good food sources 66.66%*

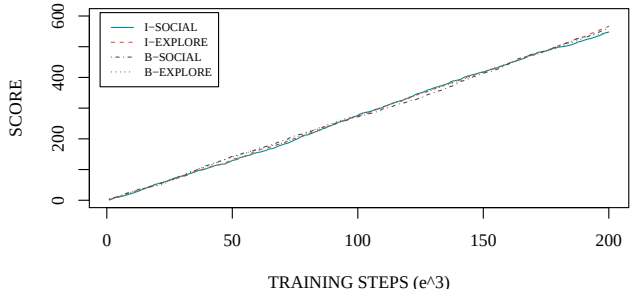
Figure 3.12: *Differences in average score/reward of agents with behaviour modulated by BAS focus types (I-SOCIAL,I-EXPLORE) and behaviours induced by implicit modification of the reinforcement function (BASE-SOCIAL,BASE-EXPLORE).*



(a) *Normal environment: even number of good and bad food sources*



(b) *Hostile environment: Bad food sources 66.66%, good food sources 33.33%*



(c) *Benevolent environment: Bad food sources 33.33%, good food sources 66.66%*

CHAPTER 4

Evolution

As the human being is the product of an adaptive evolutionary mechanism that spans over million of years, so are the psychological mechanisms that shape its behavior [14]. The lack of the insight of the science about the evolutionary basis for our behaviors is partly due to the expectation that the inherent complexity of human brains obscures the adaptive patterns of the underlying behaviors, and partly to the fear that is impossible to explain something with a tool that has the same complexity This being said, many have dismissed the search for the evolutionary origin of behaviors on the basis of them showing a great amount of complexity and variety among human communities and social constructs. However, we can argue that this complexity emerges from our nature of evolving in groups instead of individuals; like primates, we rely on the group interaction for resources and protection, so our individual fitness functions are often represented by adaptation of the whole group [14]. Regardless of being evaluated as a single or in a group setting, the behavioral variations found in humans are a product of an adaptation to the environment aimed at guaranteeing the survival of the group, which, in turn, is expected to guarantee the survival of its individual member. This need for variation can also account for emergence of meta-variations in the population that are discussed in the previous chapter: the personality traits. The personality dispositions may not be directly affecting the individual's chances of adaptation, or its fitness function, but their variations distributed through the group of individuals will support the fitness of the group. Since these low-level physical structures in the brain are a direct product of our genetic markup and complex psychological mechanisms are actually built around them and shaped by our evolutionary adaptation, they are a good representation of the forces that have created them. Since the brain is a large information processing mechanism, the best way to understand the functioning of proximate mechanisms is to look at the

way they process information by focusing on its cognitive level. This informational centered approach is good from a standpoint of evolutionary adaptation because, in order to elicit a specific behavior, a living being needs to gain information and more importantly process it in order to make mental models that inform other parts that are responsible for behavior about how to act. It is argued that for the purpose of eliciting behaviors animals have evolved *Darwinian algorithms* capable of forming adaptively meaningful frames [11–13] that are capable of focusing attention and bringing forward procedural knowledge that can lead to higher order psychological functions. For these reasons it is suggested by [14] that *"the evolutionary function of the brain is to process information in ways that lead to adaptive behavior"*. In this chapter, a biologically inspired artificial mechanism of perception is proposed which is able to evolve its characteristics in terms of processing and filtering information that can lead to emergence of adaptive behavior found in humans. This also represents an experiment in exploring how we can generate complex behavioral patterns by evolving a simple cognitive filtering of the information that is coming from the agents environment.

4.1 Perception as Attention Focusing Mechanism: An Evolutionary Perspective

We present a computational model of *attention* that includes the mechanisms for selectively storing experiences from the agent's cognitive stream into its *replay memory*, therefore providing a goal-related context buffer from which the past experiences can be sampled for re-learning. Furthermore, we explore how it is possible for an artificial agent to evolve this attention mechanism over generations so that it would make it possible to learn more efficiently by selectively focusing on experiences more valuable than others. As a focusing mechanism we are using an *artificial neural network*, which receives as input the characterizing parameters of the transition between states and produces as output a crisp decision about whether to sample the transition into the replay memory, or to discard it. After multiple trials over 100 generations, we have found that the focusing mechanism becomes more selective; this is consistent with the concept of attention, and the *replay memory* contents get permeated with transitions that are high in the values of curiosity indicative parameter *informational gain* [44]. Evolutionary motivated, higher level goal superseded the primary reinforcement-based one making the agent to adopt a new long-term strategy of intrinsic curiosity, while exhibiting a behavioral change of being more brave or aggressive towards acquiring new sources of stimuli in the given environment.

4.1.1 Model Architecture and Learning Algorithm

In this section, the structure of the learning model we are proposing is presented in its two main parts: evaluation and evolution.
The first part, shown in Figure 4.1, represents the main evaluation reinforcement model, where the proposed *attention focus block* plays a primary role. The main part of this block is the ANN function approximator (f), whose evolutionary process is shown in Figure 4.2.

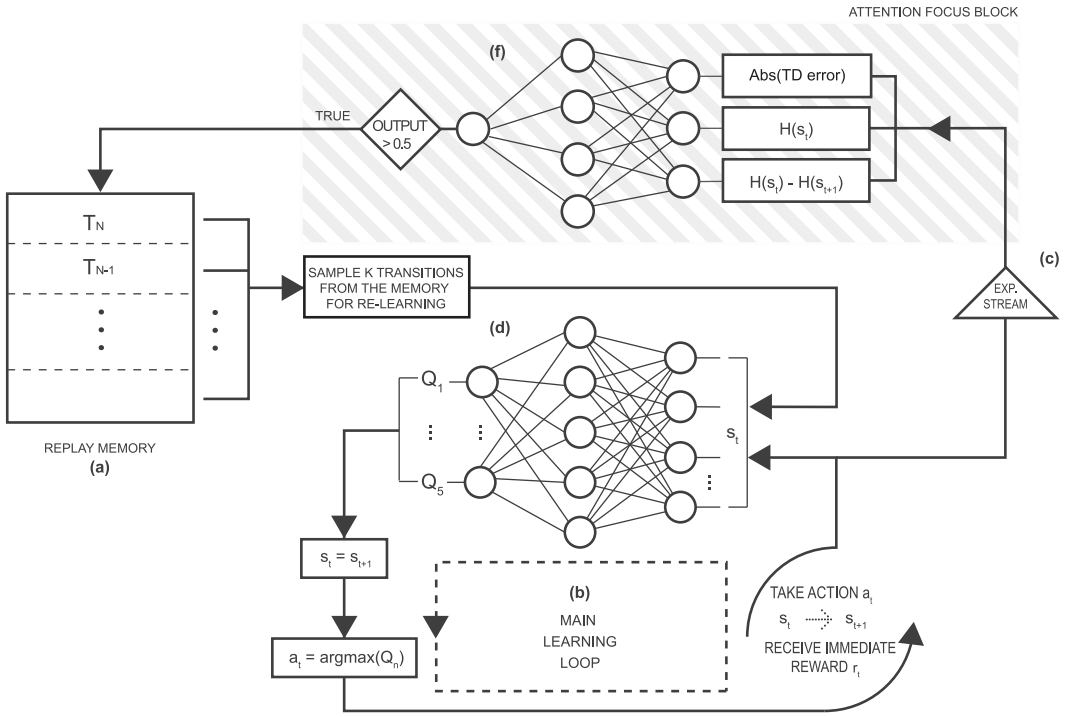


Figure 4.1: *General learning model architecture including attention focus block:* (a) *Replay memory;* (b) *Main learning loop;* (d) *A block implementing main Q-value function approximator neural network ;* (c) *Raw stream of the experiences;* (f) *ANN function approximator as a part of attention focus block*

The main part of the learning algorithm is the *learning loop*, represented as the (b) section in Figure 4.1, where the agent actually takes an action a that brings it from state s_t to the next state s_{t+1}, which is also providing an immediate reinforcement r_t. The actual learning part of the loop is supported by a main function approximator ANN shown in (d) block of Figure 4.1, which performs a backpropagation update at each learning step in order to provide a better approximation of the Q values of the state-action pairs. This ANN takes a multi-dimensional state on its input and provides the estimated Q values for each possible action available for the agent as its output layer. Since the target value for $Q(s, a)$ is given by the Bellman equation, it is possible to calculate it, taking into account the immediate reinforcement and the discounted Q value of the next state, and to compare it with the current estimate of the function approximator ANN in order to figure out how wrong it was with respect to what obtained by the last transition. This difference, also known as *TD error* provides enough information to update the new estimation of $Q(s, a)$. A *backpropagation* is performed on the approximator with the state s_t on the input layer and the gradient on the a_t output is set to *TD error* while all other gradients on the action outputs are set to 0. After the update, the transition is actuated so that s_t becomes s_{t+1} and the *loop* restarts.
The *learning loop* process provides the raw sequential experience stream marked as (c) in Figure 4.1 from which it is possible to sample some of the experiences into

a buffer structure called *replay memory* (a), which stores the experience transitions from which the agent can selectively learn. In this proposal, this process is mediated by the *attention focus block*, including another ANN function approximator (f) able to predict whether or not it is the case to sample the specific transition in the *replay memory*, depending on its properties provided in input. The properties forwarded through the ANN encompass the predictive power, given by its *TD error*, along with its information potential factors such as the Shannon's entropy of the state s_t and the *information gain* potential of the transition given by Kullback-Leibler difference between state s_{t+1} and s_t.

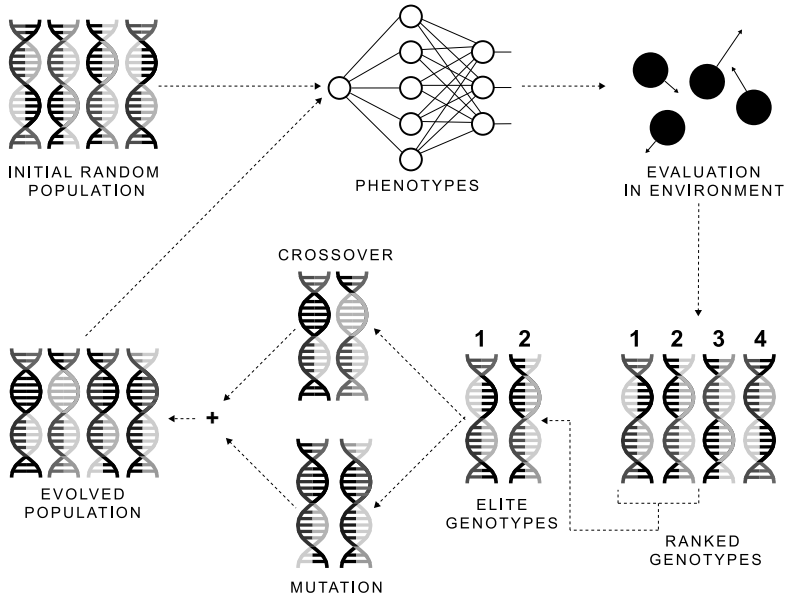


Figure 4.2: *Schema of the evolution of the Attention Filter*

The evolution of the proposed *attention focus block* is obtained by a *Genetic Algorithm* or GA, where the members of the population encode the information about the weights of the ANN approximator (f), used to control the sampling dynamics of the *attention focus block* structure. The starting point for the algorithm is a random population, and each configuration of parameters for an *attention focus block* dives learning for an agent operating in the environment. Each agent performs a learning process mediated by its *genetically* altered *attention* in an environment that provides both positive and negative reinforcement. Over a given number of learning steps, we may observe that agents learn to gravitate towards positive and to avoid negative reinforcement sources. The total reward collected at the end of a trial represents the agent's score, which is indicative of its adaptability to the specific environment.
After the evaluation phase, the agents are ranked by the total reinforcement they received, which is taken as their respective fitness. The *mating* phase starts by selecting the individuals with the higher fitness that are going to be the base for the next population. The selected genotypes undergo the genetic operations of crossover and mutation. In crossover, two *parent* genotypes randomly combine their genetic information to produce an offspring. Then, mutation randomly modifies the genetic

material. The resulting genotypes produced by these two operations join to form a new evolved generation from which the process of evaluation can start again.

4.1.2 Experimental Setup

The proposed approach was tested in different environments. Here are presented the results obtained in two environments, derived from some of the environments used in literature, namely *Waterworld* [30], and *LunarLander-v2* from *OpenAI Gym* framework [10]. In both cases we have a complex and continuous state spaces which can support their diversity.

Waterworld Environment

In the first example the agents operate in an environment inspired by the *Waterworld* setting already mentioned in Section 2.2.4 and showcased in [30].
The Q value function approximator ANN (marked as (d) in Figure 4.1 is implemented as three layers: the input layer consists of 152 nodes fully connected to an inner layer of 100 nodes, that is in turn connected with an output layer of 5 nodes (the possible actions) and trained using a learning rate $\alpha = 0.005$. It is able to approximate the 152 dimensions of state on the input to the Q values of 5 actions available to the agent. In the genetic algorithm adapting the approximator ANN, the genes were modified with a heuristically determined probability of 0.25 and the modification was implemented as the addition of a number between -0.1 and 0.1 to the respective parameter value. The evaluation trial lasted 160,000 steps. Reinforcement learning expectation is computed with a discount rate $\gamma = 0.9$, and $\epsilon-$greedy policy is used with the starting $\epsilon = 0.2$, then adjusted to 0.1 at the mid-point of the trial, after 80.000 steps for a better convergence towards the end. Replay memory buffer capacity was set to 3000 experience transitions. A population of 4 agents was evolved and after each evaluation the best performing two were selected for crossover and mutation. The new population was made from a multiple mating of the two best performing agents.

Lunar Lander Environment

The second set of experiments was performed in a more realistic setup, such as *LunarLander-v2* from *OpenAI Gym* framework [10]. The goal is to land a craft in a designated landing place indicated by two flags while countering the gravity pull using three thrusters: main, left and right orientations. While *Waterworld* is representative of tasks with continuous variables and sparse, random reinforcement, *Lunar Lander* is representative of rocket trajectory optimization which is a classic topic in area of optimal control featuring a more dispersed and constant reinforcement feedback. An episode concludes when a lander crashes or comes to a rest, in which case it receives additional -100 or +100 of reinforcement respectively. Additional reinforcement is provided, inversely proportional to the craft distance from the landing area and deviation from zero speed; it comes in the range of +100 to +140. Firing main thruster results in a -0.3 reinforcement while each leg contact with a ground is rewarded by +10. The craft has an unlimited amount of fuel at its disposal and can also land outside the designated area.

State space is 8-dimensional and consists of 4 continuous variables sensing the x position of the craft, its y position relative to the land area, craft's angle and its angular velocity along with 2 boolean variables indicating a land contact for each of the craft's leg. Four discrete actions are available to the craft: do nothing, fire main engine, fire left orientation engine, fire right orientation engine.
Reinforcement learning parameters were set to be the same as in previous batch of experiments along with an adjustment of ϵ. Elitism was implemented allowing two best scoring agents to propagate their genotypes unchanged into the next generation while the other 8 phenotypes of the next generation were generated by crossover of genotypes selected with a probability proportional to their respective scores. Mutation rate was also 0.25 and again performed by adding a number between -0.1 and $+0.1$ to the parameter value and attention filter block ANN architecture is the same as in previous batches with an exception of a hidden layer containing 6 neurons which slightly increased the variety of genotypes.

4.1.3 Experimental Results

The experiments performed on both environments, *Waterworld* and *Lunar Lander*, were compared using three settings. A genetic algorithm evolutionary implementation of the proposed *attention focus block* sampling, or *GA-AFBS*, was compared with a non-evolutionary case *R-AFBS* of generations consisting of randomly selected weight parameters, and a baseline *NO-AFBS* in which agents used no *AFBS* filtering and sampled every experienced transition into the replay memory buffer.

Waterworld Environment

A total of 6 trials were performed in the *Waterworld* environment, each of them evolved a 100 generations of *attention focus block* phenotypes evaluated by reinforcement learning phases for $160,000$ learning steps.
In Figure 4.3 we can observe the evolution of the number of experiences sampled by the *attention focus block* over 100 generations. Experimental data show that the *attention focus block* evolved in the direction of being more selective about the sampled experiences, from inefficiently taking almost 88% of the raw experience stream in the replay memory at the beginning, to a much more selective selection of 12% experiences at the 100th generation, which represents a great difference with respect to the random *R-AFBS* percentage which was constantly kept around 40%.

Figure 4.4 shows how the evolutionary model influenced the performance of the agents given by their total score, or total reinforcement received over the evaluation phase. An approach that used evolving phenotypes of *attention focus block* (*GA-AFBS*) greatly outperformed the no filter one (*NO-AFBS*) by over 400% and a random parametrization (*R-AFBS*) by more than 200%, and came to a stable point in about 75 generations.
Figure 4.5 shows the sampling preference of the approaches in terms of types of transitions as determined by average *informational gain* values of the sampled experiences in the *replay memory*. We can see that the genetically supported evolution of *GA-AFBS* evolved a high tendency to sample the experiences with positive values of the *informational gain* property in contrast with the no filter *NO-AFBS*, whose aver-

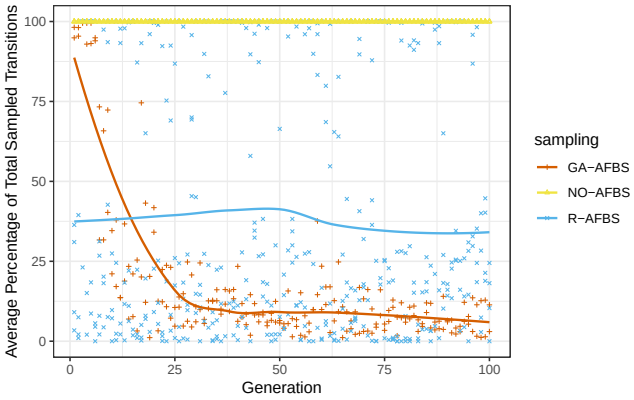


Figure 4.3: *Average total sampling percentage in the Waterworld environment of the genetic algorithm supported evolution of the attention focus block (GA-AFBS) compared with a non-evolutionary sampling implemented as a random attention filter neural network (R-AFBS) and a baseline approach without any cognitive filter (NO-AFBS), over the first 100 generations of 6 trials (solid lines) and the respective variance (gray areas).*

age accumulated to 0 as it preferred the experiences with both positive and negative values in the same proportion. The evolutionary approach settled to a 0.1 average informational gain which gave rise to a more curious agents than a random *R-AFBS* one, which was rather consistent with an average of 0.25 throughout the generations.

Lunar Lander Environment

Evaluation phase in *Lunar Lander* environment in *GA-AFBS* consisted of a generation of 10 agents competing with each other based on the average reward received over 60 consecutive learning episodes.
Changes in sampling preferences can be seen from Figure 4.6, where the proposed evolutionary approach *GA-AFBS* evolved again to be more restrictive to select experiences for replay memory. Although not as selective as *Waterworld*, in the *Lunar Lander* environment *GA-AFBS* resulted in a more conservative 25% sampling percentage which is a significant change compared to the expected 50% average sampling of the random network *R-AFBS*.
Figure 4.7 shows the amount of improvement that *GA-AFBS* brought to the more realistic *Lunar Lander* environment. We can see that *GA-AFBS* took about 12 generations to outperform the baseline *NO-AFBS* by 150% and provide a 125% increase over the random *R-AFBS*.
We can also notice that *GA-AFBS* evolved a preference for sampling experiences that manifest a higher value of the starting state entropy $H(s_t)$ outlined by the Figure 4.8 and a high preference for experiences with lower informational gain level IG which can be seen from Figure 4.9.

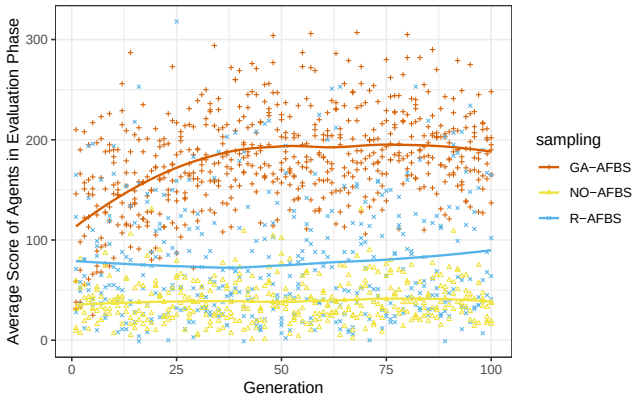


Figure 4.4: *Average fitness (total reinforcement) received in Waterworld environment by the proposed genetic algorithm supported evolution of the attention focus block GA-AFBS sampling, compared with a non-evolutionary sampling implemented as random attention filter neural network R-AFBS, and a baseline approach without any cognitive filter NO-AFBS over first 100 generations of 6 trials.*

4.1.4 Discussion

Informational Gain parameter as a measure of Curiosity

Informational gain or *IG* parameter is defined as Kullback-Leibler difference or relative entropy between posterior state s_{t+1} and anterior one s_t, as we can see from Equation 4.1. It is especially important for a discussion about the emergence of intrinsically motivated evolved behavioral traits of agents in the *AFBS* sampling method. It can provide an insight about the agent preference to move towards a state of higher informational content which is indicative of intrinsic curiosity if it is positive, or, on the other end of the spectrum, negative values are indicating a more cautious move in which the agent is moving away from the state of high informational potential.

$$IG = H(s_{t+1}) - H(s_t) \tag{4.1}$$

Implications

From Figure 4.3 and Figure 4.6, we can conclude that the evolutionary approach of *GA-AFBS* can reduce the cognitive load on the agent induced by a highly saturated, high-dimensional state space, by selecting more interesting experiences that are stored for learning in the *replay memory*. Besides evolving an optimal cognitive load for each of the considered environments of 12% for the *Waterworld* and 25% for the more realistic *Lunar Lander*, this approach also improved the selection of experiences that are more valuable for machine learning as evident from Figure 4.4 and Figure 4.7, which show a significant improvement of the total reinforcement received in both environments. The fact that the different optimal sampling percentages were evolved in adaptation to the environments bring us to a conclusion that the different environments present a varying level of cognitive load for the learning agent. The more chaotic nature of the *Waterworld* gave rise to more interesting, information sat-

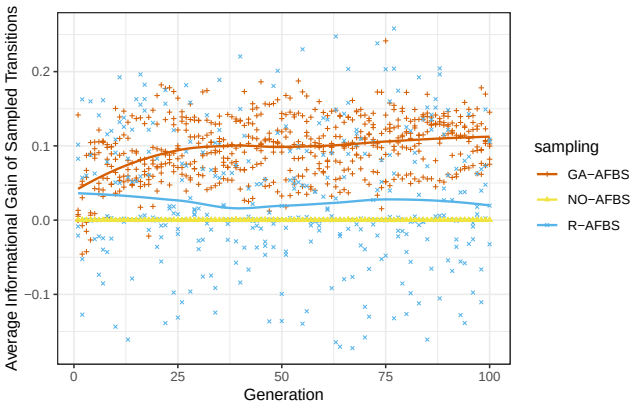


Figure 4.5: *Average values of informational gain parameter of experiences contained in working memory at the end of evaluation in Waterworld environment for the proposed genetic algorithm supported evolution of attention focus block GA-AFBS sampling, compared with a non-evolutionary sampling implemented as random attention filter neural network R-AFBS and a baseline approach without any cognitive filter NO-AFBS over first 100 generations of 6 trials.*

urated transitions, which resulted in a more selective perception compared to the not so saturated one in *Lunar Lander* environment.
Some behavioral characteristics such as curiosity were intrinsically evolved to better adapt to the specific dynamics of the environment. From Figure 4.5 we can see that in the *Waterworld* environment curiosity or positive *IG* was evolved as an adaptation trait that arose from the need of an agent to be more engaged in the environment with a scarce reward and more focused in finding transitions that lead to situations expected to provide positive reinforcement.
Contrary to the scarce reinforcement feedback of the *Waterworld* environment, *Lunar Lander* provided a totally different and more dynamic reward mechanism which included constant adjustment of the reinforcement function based on the agent's state. Confronted with the dynamics of the *Lunar Lander* environment, *GA-AFBS* evolved a trait of being cautious given its preference for the transitions with negative *IG* as shown in Figure 4.9. Also interesting to note is that the evolved perception mechanism in *Lunar Lander* displayed a preference for the transitions that have a higher entropy of the starting state, as can be see in Figure 4.8, which possibly contained more informational potential for learning, but at the same time cautiously preferred low entropy of the next state s_{t+1} given by the negative *IG* displayed in Figure 4.9.
From displayed results it is possible to notice that an evolved artificial perception using the proposed *GA-AFBS* algorithm was able to alter the behavioural characteristics of the agents by producing agents with specific traits or tactics that provide a better adaptation to a specific environment without the need to alter the reinforcement function.

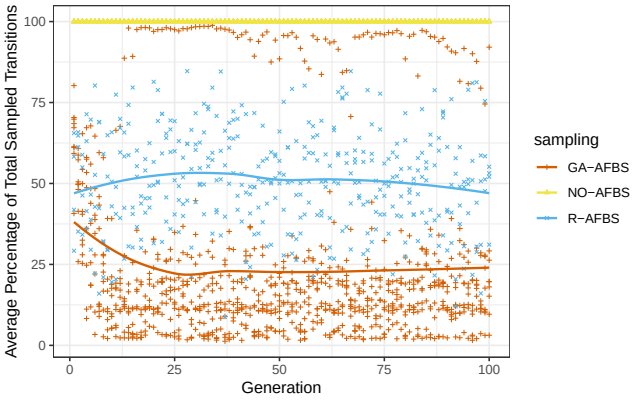


Figure 4.6: *Average total sampling percentage in Lunar Lander environment for the proposed genetic algorithm supported evolution of attention focus block GA-AFBS sampling, compared with a non-evolutionary sampling implemented as random attention filter neural network R-AFBS and a baseline approach without any cognitive filter NO-AFBS over first 100 generations of 6 trials.*

4.2 Related Work

4.2.1 Artificial attention as a behavior inducing mechanism

The idea that selective focusing of the memories that enter working-term memory analogue of *replay memory* can behaviorally influence the artificial learning agents was explored in [47]. In this work, a computational model of main personality trait axis including introversion-extroversion dichotomy was presented. The model was based only on changing the dynamics of the attention span of the *replay memory*, which was shown to be different between introverted and extroverted individuals, as the latter exhibited a broader attention span [22]. The two types of agents were tested in different variations of the environment respectively providing positive and negative reinforcement ratios. *Normal* environment provided an equal amount of positive and negative reinforcement and represented the baseline for the experiment. *Hostile* environment provided more negative reinforcement, while the *benevolent* one provided more positive reinforcement. Curiosity-driven, extroverted agents performed better in a *benevolent* type of environment while the cautious, introverted ones managed to learn better in the *hostile* environment.
Attention based working memory approach was used also in [48] which proposed a selective focusing of the experiences based on their informational potential or Shannon's entropy of the perceived state space. Although the aim of this approach was primarily to increase the learning performance of the agents, compared to an uniform sampling baseline the entropy-based sampling also seemed to have induced an intrinsically motivated exploration that became an important part of their tactics to increase the overall performance.

4.2.2 Evolutionary Adaptive Approaches

In [45] basic emotions such as fear were evolved as motivational drives involved in adaptation of learning agents to their immediate environment. At each generation,

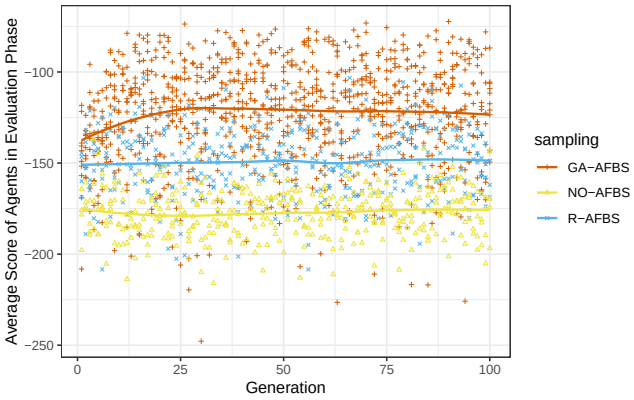


Figure 4.7: *Average fitness or total reinforcement received in Lunar Lander environment for the proposed genetic algorithm supported evolution of attention focus block GA-AFBS sampling, compared with a non-evolutionary sampling implemented as random attention filter neural network R-AFBS and a baseline approach without any cognitive filter NO-AFBS over first 100 generations of 6 trials.*

a new population of virtual agents was tested, each of them evolving a neural network that mapped its input, which included time, good and bad sensation neurons, and visual perception, to its output that was used to focus on the visual stimuli and select the agent's actions. Over time, the selection of agents based on elitism w.r.t. the ability to adapt to the environment gave rise to a specific drive of being cautious or fearful as a survival strategy.
Another evolutionary perspective is presented in [56] in which the reward functions of the agents are evolved taken in consideration their fitness. This forms an idea of *optimal reward function* that builds upon the basic reward function in order to maximize the expected fitness over distribution of environments. The presented experiments support the notion of emergence of intrinsic reward for specific actions such as playing and manipulating objects in their immediate environment that are not meeting any primary need of the agent. [51] introduced a combination of evolution and machine learning allowing the agents to intrinsically evolve a basic reinforcer for atomic building-block skills in the *childhood* learning phase, which are later used in the *adulthood* phase.

4.3 Conclusion

As the machine learning mechanisms evolve, we are now aware that along the advancement of the learning algorithms focused on how to use data received from the environment in a most efficient manner to support learning, we also need to be concerned about the way those data are *perceived* in the first place. In spite of being vastly unexplored, a good source of inspiration for new computational and evolutionary approaches of *perception* is a computational organ that is a product of a million of years of evolution: the human brain. In chapter 4 we tried to exploit the insights received by the areas of psychology and neuroscience, which describe the higher order complex functions that our brain is using in order to make its perception more efficient. Since these functions were developed in humans by an evolutionary pro-

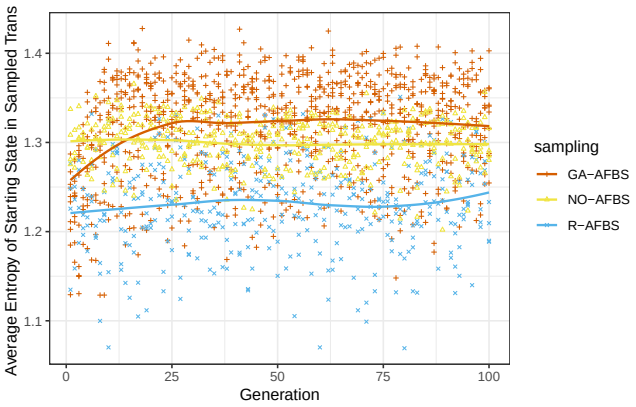


Figure 4.8: *Average values of entropy of the starting state of experiences contained in working memory at the end of evaluation in Lunar Lander environment for the proposed genetic algorithm supported evolution of attention focus block GA-AFBS sampling, compared with a non-evolutionary sampling implemented as random attention filter neural network R-AFBS and a baseline approach without any cognitive filter NO-AFBS over first 100 generations of 6 trials.*

cess of natural selection, it seemed that a similar process in a computational sense would also do the work.

Nature doesn't like to reinvent things, instead the basis of the primordial functional mechanisms is found in all of the complex organisms up to date in the primal bulk of the genetic material. Evolution is all about keeping what works and building upon it and this is the reason we share a great deal of genome with a simple organism such as fruit fly and our brains are built upon features found in reptiles only and those primal parts can still take control over the higher order neocortex if we are found in a dangerous situation. Different species have developed perception mechanisms with varying complexity and scope in order to better adapt to the different environments but they all share the same primordial building block. For example, the fish that have adapted to living in a hostile environment of deep oceans without any light have adapted their perception mechanism to rely almost solely on tactile and olfactory domains while most of them still have now obsolete functional visual system reminiscent of their common evolutionary base. In the approach presented in chapter 4 an evolutionary algorithm was used in order to adapt the basic functional artificial mechanism of perception called *attention filter block* to a specific learning environment. In this way the artificially evolved mechanism took a role of a primitive perception that enhanced and supported the agents learning process. It is interesting that over the generations the mechanism took a "protective" stance towards the *working memory* of a limited capacity by gradually reducing the cognitive load to a more manageable $\frac{1}{10}$ of the sensory input. The environment dynamics required that for an agent to be more efficient in acquiring food it needed to explore and move in an aggressive way and this affected the algorithm to evolve a more curious agents by favoring experiences that lead to an increase of information potential in a sensed state.

Evolution of the artificial perception proved to be an important part of a long-term adaptation that is able to optimize a short-term adaptation algorithm of the reinforcement learning itself. In this way, it is possible to achieve both types of adaptations by

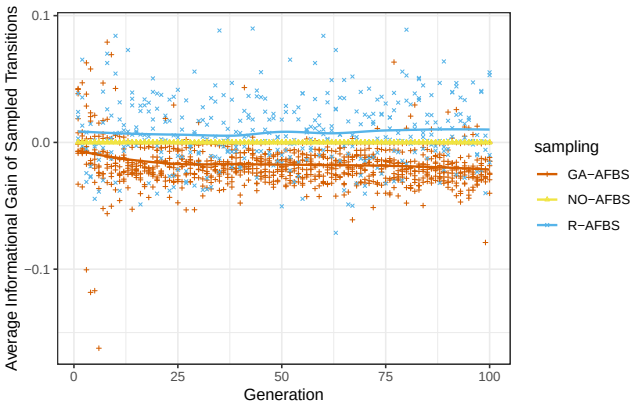


Figure 4.9: *Average values of informational gain parameter of experiences contained in working memory at the end of evaluation in Lunar Lander environment for the proposed genetic algorithm supported evolution of attention focus block GA-AFBS sampling, compared with a non-evolutionary sampling implemented as random attention filter neural network R-AFBS and a baseline approach without any cognitive filter NO-AFBS over first 100 generations of 6 trials.*

having an artificial agent learn how to perceive the environment and use that experience in order to perform a more efficient final adaptation in terms of maximization of an expected reward. Although oversimplified compared to the human brain, the proposed approach was able to develop an applicable filtering mechanism able to reduce the cognitive load and to induce an intrinsic behavior, therefore simulating the arising of the process of *attention* just by changing the dynamics of experience sampling.

CHAPTER 5

The Performance

The application of the mechanism of *replay memory* is not limited to eliminating problems with highly time-correlated samples that gradient descent approximators don't prefer; from previous chapters we have seen that it can be used in order to behaviorally modify the learning process. In this chapter, the dynamics of *replay memory* structure is changed in order to improve the overall performance of the learning agents. The first part of the chapter takes a biologically inspired approach while the second focuses on improving the criteria for prioritization.

5.1 Context Augmented Reinforcement Learning

5.1.1 Introduction

Studies concerning human and animal memory storage and retrieval as a part of a basic cognitive information processing show that the *context* in which a specific memory is retrieved highly influences its perception and expression [61]. Furthermore, it has been evident that the contextual retrieval nature of the memory is very much similar to the learning process [57]. This nature of processing memories is evident in our everyday life; we are more likely to remember the person if the person appears again in the same environment in which we first met that individual as opposed to a totally new, out of context, location. Context, in this case a location act as a sort of amplifier for the similar and connected memories thus supporting their retrieval. *Contextual memory retrieval* dynamics can be found in complex human declarative memory and also in a more simple form of animal learning known as *classical conditioning* [7, 8].
Similiary to the biological organisms some online reinforcement agents use the mem-

ory of the past experiences which can be replayed in order to support the learning process [36,39,40]. These learning mechanisms benefit from a *replay memory* structure which is efficient in optimizing the *stochastic gradient descent* or SGD algorithm used in training the artificial neural network approximator which represents the core of the reinforcement learning mechanism. The approach presented here further exploits the biological learning mechanisms such as *contextual memory retrieval* in order to increase the efficiency of the learning process in function approximated reinforcement learning. Instead of using a single memory replay buffer and treating all experiences equally, *Context Augmented Reinforcement Learning* or *CARL* introduces a finite number of contextual replay memory buffers, each of which holds the experiences corresponding to its specific context. In order to achieve this, each experience of an agent is clustered into the same amount of dimensions or classes as the number of contextual memory buffers based on their similarity. The more that experience belongs to a specific class the more are the chances of it being sampled into the contextual memory buffer corresponding to this class. The main principle of the *context augmenting* algorithm is found in replay of the clustered memories; each experience is supported or augmented by replaying the past experiences from its corresponding memory replay memory buffers instead of a random out of context replay used in [36, 39, 40]. Experimental results show that *CARL* shows an improvement in learning performance over the standard q-learning with uniform sampling and q-learning with improved prioritized sampling based on temporal difference error [50].

5.1.2 Related Work

In the approach presented by Shaul et al. [50] the replay memory mechanism was improved by prioritizing on the experiences with higher temporal-difference error. These experiences are potentially more valuable for the training of the approximator in a way that they carry more surprise over the predictions than other.

5.1.3 Model Architecture and Learning Algorithm

The main part of the learning mechanism represents the *learning loop* outlined in section (b) of Figure 5.1. Here the agent iteratively performs an interaction within its environment and uses the newly obtained experience to update its belief about the best action to take depending on the state of its surroundings. During the transition stage, an agent performs an action a_t which transitions the agent from the starting s_t state to the next s_{t+1} while receiving the immediate feedback from its environment in the form of a scalar reward r_t. These parameters form a tuple $e_t = (s_t, a_t, r_t, s_{t+1})$ which fully determines a transition and contains every information we need to perform a learning step. After an agent has transitioned and we have gained the parameters of the tuple we can use Equation 1.3 in order to calculate our expected value for the state-action pairs $Q(s_t, a_t)$. The update is performed in section (d) of the *learning loop* by adjusting the weights of the approximator neural network using SGD in such a way that we can minimize the squared error or difference between our newly calculated expected Q-value and our current estimate of it. After a learning update the states are shifted and a new iteration of the *learning loop* is performed. This loop

creates a stream of experiences (c) represented by a sequence of transition tuples e_t. In the base approach of Q-learning those experiences are sampled in a sliding window *replay memory* buffer from which a number of them are selected at random to be reused after each iteration in order to perform an additional training on approximator neural network. With $CARL$ approach instead of having one *replay memory* buffer we have a M amount of *contextual replay memory* buffers shown in section (a) of Figure 5.1, each of which is corresponding to one of the M classes of the experiences coming from *experience stream*. Before storing each experience is being clustered into M dimensions using an unsupervised learning method in order to determine into which *contextual replay memory* it will be sampled. To determine the classes of the experiences an *autoencoder* neural network shown in (f) is used to predict the state s_t part of an experience from the same s_t on the input. The *autoencoder* performs a reduction of the dimension of the perceived state space from its original size to M dimensions or classes using a bottleneck hidden layer in the middle consisting out of M fully connected neurons. The process similarly performs SGD in order to make the weights of a neural network a better predictor of itself by minimizing the difference between the predicted and the actual state given by the experience. The M dimensions or classes are effectively encoded in the M neuron layer and in order to find the classes of a single experience we forward the state s_t through the network and obtain a vector of the M activations of the bottleneck layer $(C_1, C_2, ..., C_M)$. We define the probability of the experience being sampled into and consequently replayed from the i-th *contextual replay memory* or CRM_i as the class or the activation of the i-th layer of autoencoder bottleneck layer as defined in Equation 5.1.

$$P(CRM_i) = C_i \tag{5.1}$$

The more the experience belongs to a specific class the more the probability it will be sampled into and replayed from the CRM of that class. Algorithm 7 showcases the details about the process of *context augmentation* in a way that each experience is supported or augmented by the experiences that are similar by replaying experiences from the contextual buffers that are determined by the classes the experience belongs to.

5.1.4 Experimental Setup

Environment

The agents performed a learning process in an *Waterworld* environment that is part of ReinforceJS machine learning framework [30]. The environment is described in Section 2.2.4.

Approximation of $Q(s, a; \Theta) \approx Q^*(s, a)$ is done using an ANN with one hidden fully connected layer of 100 neurons which are producing as output the Q values of all five actions available to the agent: up, down, left, right, stay at the output given the state space of 152 dimensions on the input. The learning rate of an approximator α is set to a low 0.05 and the capacity of the each contextual replay memory buffers N was 1000. The value of ϵ was set to 0.2 at the beginning and adjusted to 0.1 at the mid-point of the learning to exploit more of the learned behavior. The discount factor γ is set to 0.9.

Algorithm 7 Q-learning with Contextual Augmentation

Initialize M instances of replay memory D of capacity N
Initialize and pre-train autoencoder neural network A with M neurons in the middle layer
Initialize action-value function Q with random weights
for episode = 1, E **do**
  for t = 1, T **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \arg\max_a Q^*(s_t, a; \Theta)$
    Execute action a_t, observe reward r_t and state s_{t+1}
    Train A by backpropagating s_t with same s_t on the input
    Forward s_t through A and obtain a M-dimensional vector of activation values from neurons on the middle layer (C_1, C_2 ... C_M)
    for i = 1, M **do**
      Store transition (s_t, a_t, r_t, s_{t+1}) in i-th replay memory D_i according to the probability of the i-th activation $P(C_i)$
      Sample random batch of transitions (s_t, a_t, r_t, s_{t+1}) from i-th replay memory D_i according to the probability of the i-th activation $P(C_i)$
    end for

$$\text{set } y_i = \begin{cases} r_i, & \text{terminal } s_{i+1} \\ r_i + \gamma \max_{a'} Q(s_{i+1}, a'; \Theta), & \text{non terminal} \end{cases}$$

    Perform a gradient descent step on $(y_i - Q(s_i, a_i; \Theta))^2$ according to Equation 1.12
  end for
end for

Unsupervised learning

Autoencoder neural network was able to cluster the 152 dimensional states into $M = 6$ dimensional array using architecture of three hidden layers. The input and output layers were implemented using 152 neurons capable of handling the high-dimensional state array on both sides. The hidden part consisted of a bottleneck middle layer containing $M = 6$ neurons surrounded by two additional fully connected layers implemented with 100 neurons each, able to smooth out the transition from high to low dimensional space. Each of the output activations of the bottleneck layer was paired with a $M = 6$ CRM buffers. The learning rate of the autoencoder was set to a higher $\alpha = 1$ because of a different nature of the approximation.

5.1.5 Experimental Results

Experiments compared the average performance of the agents using three different types of memory replay sampling in 50 learning trials consisting of 180 learning steps each. Figure 5.2 compares the baseline q-learning with uniform memory sampling and replay USQ with enhanced sampling method of TD-error prioritization TDQ and the novel approach of *context augmented reinforcement learning* or

CARL. The $CARL$ sampling method shows superior performance over the baseline uniform sampling and it even outperforms the optimized sampling based on the TD-error prioritization.

5.1.6 Discussion

The presented approach tries to bridge the gap between biological and artificial systems by implementing a biologically inspired technique of grouping and replaying the similar memories together in order to increase the learning potential of experiences. Like the discussed *contextual memory retrieval* an artificial learning agent is able to support and amplify the effect that an experience brings to the learning process by grouping it with past memories that correspond to the same context. In this way the replay memory structure acts as a secondary drive that alters the agent's cognition by supporting the current experiences which in terms improves its overall performance.

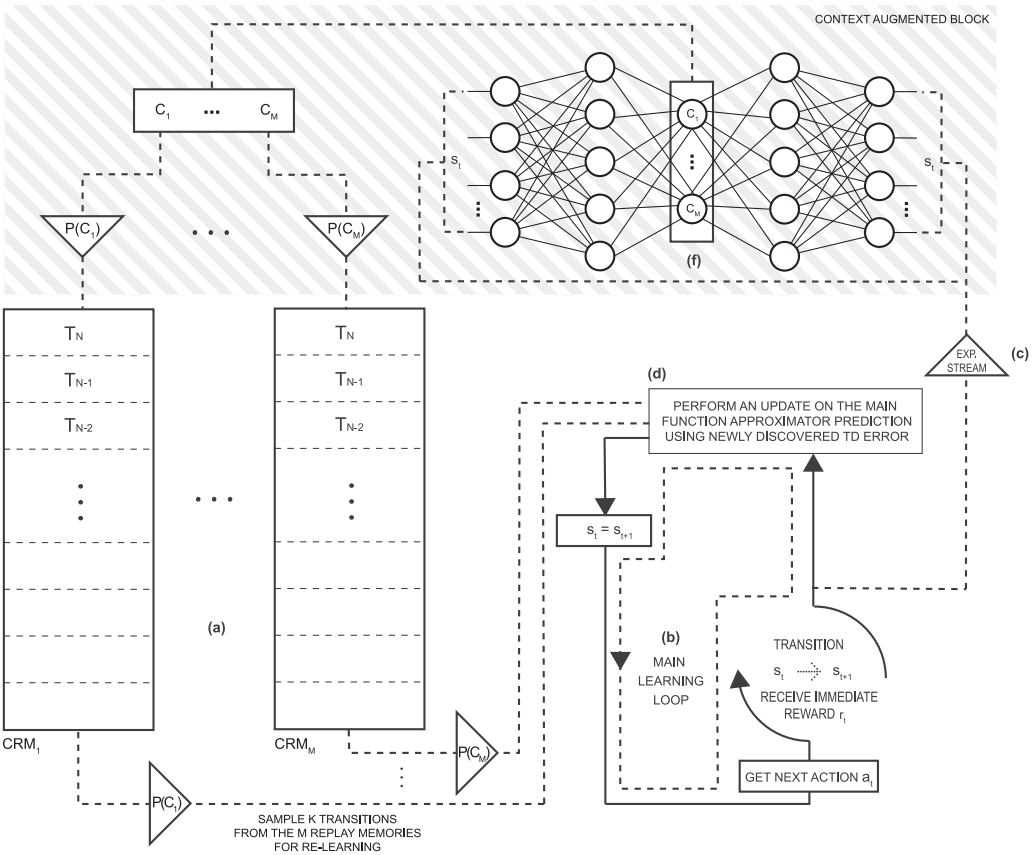


Figure 5.1: *General learning model architecture including attention focus block:* (a) *Array of* M *contextual replay memories, each of which stores the last* N *experiences of its own class in a sliding window buffer for later replay;* (b) *Main learning loop consists of:* $1)$ *the transition in which the agent performs an action, receives an immediate reward* r_t*, while transitioning to a next state* s_{t+1}*;* $2)$ *performing an update on main function approximator ANN* (d) *by backpropagating the TD error as a gradient of the* a_t *output;* $3)$ *shifting the states for the next iteration in which the* s_t *becomes our* s_{t+1}*;* $4)$ *forwarding the current state through a function approximator in order to find out the best action* a_t *candidate based on its Q value for* ϵ*-greedy policy;* (d) *A block implementing Q-value function approximator taking the starting state* s_t *on the input and predicting Q-values for each of the available actions on its output;* (c) *Raw stream of the experiences that are perceived representing unfiltered cognition of an agent;* (f) *Context augmented block implemented as an autoencoder neural network performing unsupervised clustering of the experiences into* M *dimensions or classes, each of which determines the probability of the selected experience being sampled into and replayed from the corresponding contextual replay memory buffer*

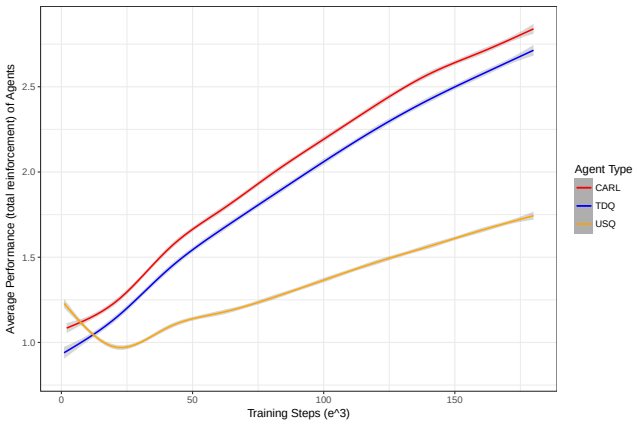


Figure 5.2: *Average score or total reinforcement received over 50 trials with agents with different sampling dynamics during first 180.000 learning steps. Gray ribbon represents the standard deviation of the samples.*

5.2 Entropy-based Prioritization

Since some transitions are more valuable for learning than others, especially in the early stages, prioritizing on experience transitions was introduced in order to improve the general performance of learning.
Successful approaches dealt with prioritized experience sampling [64] and experience replay [50], but their prioritization criterion was limited to one property of the transition, TD error, which is mostly conditioned by the reinforcement that the transition gave rise to. This inference was exploited by prioritized experience sampling [64], which prioritized on a specific property of the transition that in general yields higher absolute values of TD error: its immediate reward value being non-zero.
In this work, we propose an additional criterion for prioritizing using *Cognitive Filter* which takes into account the specific property of the sensed state space, measured by Shannon's entropy. We further show that prioritizing on transitions that include the state with higher entropy values can additionally improve the performance in some learning scenarios.

5.2.1 State space entropy prioritization

Going beyond TD error

An agent performs the learning process on a single transition (s_t, a_t, r_t, s_{t+1}) by first predicting its previous estimate of the Q-value for being in a state s_t and taking an action a_t. This process performs a forward pass on the neural network approximator with s_t on input, after which we select the predicted Q-value on the output a_t. TD error represents the discrepancy between the previous estimate and the expected target Q-value after the transition which is given by its newly discovered reinforcement value r_t and the discounted maximum Q-value of the next state s_{t+1}. The learning process represents an update on the estimate of the function approximator by using backpropagation rule with perceived state space features s_t on the input and TD error difference on the a_t output.
Previous prioritization approaches consider that the magnitude of TD error that a specific transition generates can directly influence the speed of the learning process given the nature of the learning update. Transitions with high magnitude of TD error that are usually produced by non-zero reinforcement are especially valuable in the early stages of learning because they carry more training information and thus can make the learning process faster.
Because the backpropagation update takes into account not only the TD error at the output, but a state space vector at the input, while performing a gradient descent we can analogously postulate that the nature of the state space vector can also have an effect on the learning process itself. Some of the state space vectors are potentially carrying more training information and can be favored by prioritizing, in order to improve the learning performance. For example: if an agent perceives a predictable or simple environment during the transition, the state space vector will potentially carry less information for training than when the transition is made in a dynamic and highly unpredictable environment.

We can thus say that the learning utility of the transition not only depends on the transition TD error, but also on the potential information that is carried by the perceived state space vector.

Quantifying the unpredictability of the state space

In order to quantify the amount of uncertainty and possible information gain that a state space vector can carry we have applied Shannon's entropy as a measure of diversity, also called Shannon's index. The state space vector is represented by a number M of variables that are in most cases continuous and normalized in

$$0..1$$

. In order to measure the entropy, each of the M state space variables are discretized into N bins and calculated using Equation 5.2, where p_i is the frequency of values belonging to the ith bin.

$$H(s_t) = -\sum_{i=1}^{M} p_i log_2 p_i \tag{5.2}$$

Model Architecture and Learning Algorithm

Previous prioritization algorithms [64] used a stochastic sampling method that falls between uniform sampling and greedy sampling based on the TD error.
In our approach, we introduce an additional criterion based on the diversity of the state space that can further prioritize on the uniform sampling part of the algorithm.
For the purpose of prioritizing we extend the experience description tuple with an entropy value $H(s_t)$ of the state space that is calculated using Equation 5.2, so that our stored transition takes the form of $e_t = (s_t, a_t, r_t, H(s_t), s_{t+1})$.
Instead of greedy sampling on $H(s_t)$ values which can make the 'system prone to over-fitting because of the lack of diversity [50], we define a stochastic prioritization based on the entropy criterion $H(s_t)$ where the probability of sampling the $P(i)$ transition from the sliding window experience memory D is determined from Equation 5.3. $H(s_t)$ in this case represents the priority of the transition and the β parameter determines how much prioritization is used; in the uniform case $\beta = 0$.

$$P(i) = \frac{H(s_t)_i^{\beta}}{\sum_{j=1}^{j=size(E)} H(s_t)_j^{\beta}} \tag{5.3}$$

To alleviate the selection of the values for the β parameter, which would need to be tweaked for the specific application, we introduce a more general prioritization technique based on the descriptive statistical property of quartiles that can be used in a broader sense with no additional adjustments.
In order to sample basing on the $H(s_t)$ criterion, in Algorithm 8, instead of the stochastic approach given by Equation 5.3 we use a descriptive statistic approach which takes into account the upper interquartile mean of the data stream or the third quartile value ($Q3$) of the H_t values of agent's experiences stored in a sliding window memory E of capacity n. This is computed by Equation 5.4.

$$H(s_t)_{Q3} = \frac{3(n+1)}{4} thH(s_t) \tag{5.4}$$

Given this, we sample only the transitions with $H(s_t)$ higher than the upper interquartile mean $H(s_t)_{Q3}$ of the entropy experience memory E as shown in Algorithm 8.

Algorithm 8 selectively stores the transitions after each update step based on two criteria. The first one is based on the TD error, and simply stores the transitions that result in a reinforcement r_t different from null. The second criterion stores the transitions that have the entropy state space value $H(s_t)$ higher than the upper interquartile mean of the n latest entropy samples from E given by the $H(s_t) > H(s_t)_{Q3}$ conditional.
After each transition a random batch of the previous transitions is selected from the replay memory D in order to perform additional training on the approximator.

Algorithm 8 Deep Q-learning with entropy-based prioritization

Initialize replay memory D with capacity N and entropy experience memory E
Initialize action-value function Q with random weights
for episode = 1, M **do**
  for t = 1, T **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \arg\max_a Q^*(s_t, a; \Theta)$
    Execute action a_t, observe reward r_t and state s_{t+1}
    Calculate the entropy value $H(s_t)$ of the state space based on Equation 5.2 and add it to the sliding window memory E
    if $r_t! = 0$ **then**
      Store transition $(s_t, a_t, r_t, H(s_t), s_{t+1})$ in D
    end if
    Calculate upper interquartile mean $H(s_t)_{Q3}$ of the last n samples from E using Equation 5.4
    if $H(s_t) > H(s_t)_{Q3}$ **then**
      Store transition $(s_t, a_t, r_t, H(s_t), s_{t+1})$ in D
    end if
    Sample random batch of transitions $(s_t, a_t, r_t, H(s_t), s_{t+1})$ from D

$$\text{set } y_i = \begin{cases} r_i, & \text{terminal } s_{i+1} \\ r_i + \gamma \max_{a'} Q(s_{i+1}, a'; \Theta), & \text{non terminal} \end{cases}$$

    Perform a gradient descent step on $(y_i - Q(s_i, a_i; \Theta))^2$ according to Equation 1.12
  end for
end for

5.2.2 Experimental setup

To evaluate the proposed model we have adopted a learning environment that consists of moving good/bad food pieces [30] as described in Section 2.2.4.

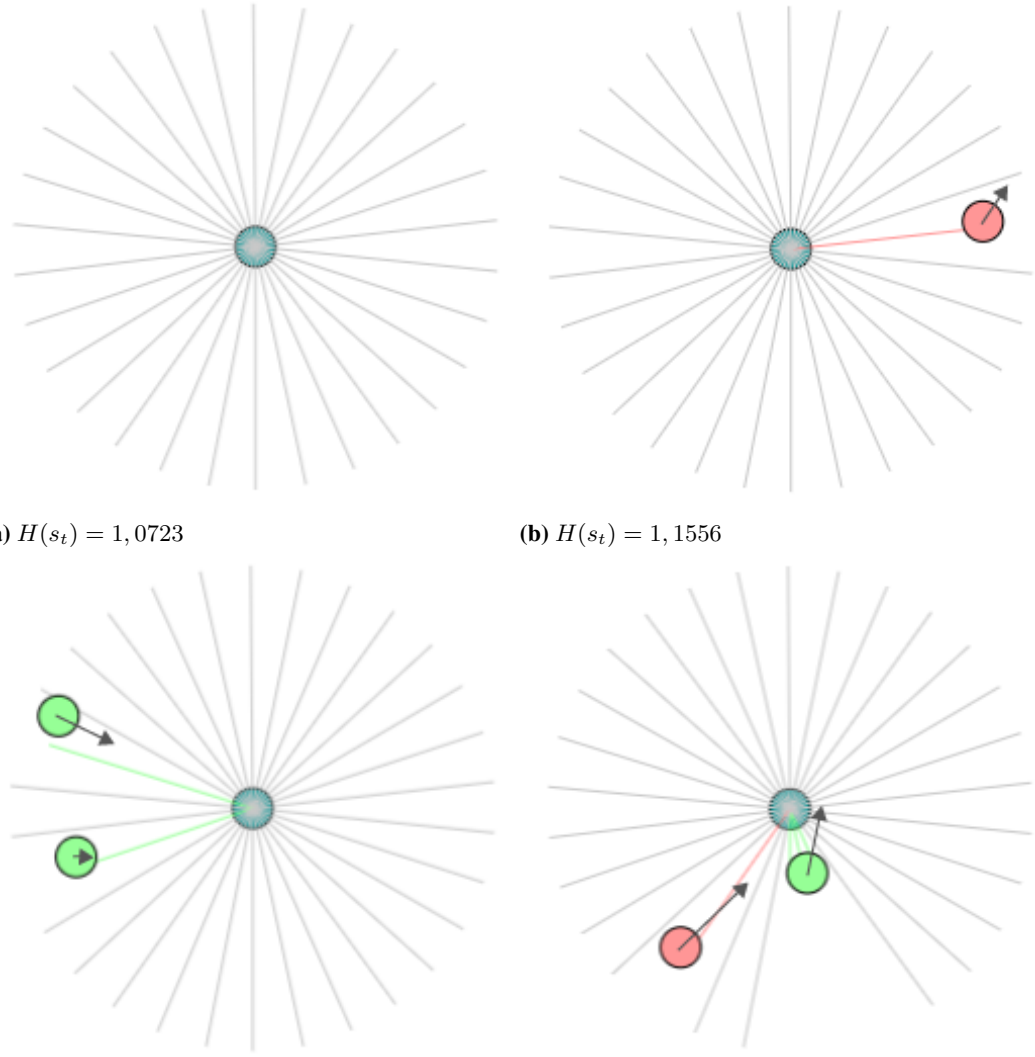
The state space is continuous and intentionally high-dimensional for the purpose of increasing the entropy and consequently the diversity of possible experience transitions.
We have adopted the original Q-learning update formula with a learning rate α set to a low value (0.05) because of the nature of the approximator, and discount factor $\gamma = 0.9$. The default capacity of the replay memory buffer D included 7000 experiences and the entropy experience memory capacity n was set to 500.

Entropy criterion comparisons

In order to evaluate how does entropy value of the state space vector $H(s_t)$ relate to the diversity of the agents' perception and further to its learning potential we are comparing state examples from the experimental setup grouped in low, medium and high entropy levels. For the purpose of evaluation each of the detected objects is depicted with its speed vector that represents the composition of its x and y speed components as described in the state space.
Figure 5.3 showcases some of the low entropy states, ranging from $H(s_t)$ 1.0 to 1.4. The center circle represents the agent, the lines its 30 detectors, the other circles food pieces. From Figure 5.3a we can see that if an agent is not perceiving any object the entropy of the state vector has the lowest value of $H(s_t) = 1,0723$. This represents the transition having the least value for learning process even if the transition results in a reinforcement reward that makes the TD error potentially high.
Figure 5.3 also shows that the $H(s_t)$ rises with the number of perceived objects but this is not always the case; both Figure 5.3c and Figure 5.3d have the same number of objects in the range but in Figure 5.3d we see more differences in distance from the objects and in their respective speed vectors, which account for higher diversity and consequently higher entropy values.
Medium entropy states shown in Figure 5.4 confirm the previous observation, but also include the food sources that are triggering more detectors because they are closer to the agent and this results in even higher entropy values. The previous assumption of the diversity in distance and speed vector is especially evident in the entropy difference between the situations respectively depicted in Figure 5.4a and Figure 5.4b.
Figure 5.5 shows the states with the highest entropy and diversity. We can notice that these states involve a high number of objects with diverse distances and speed vectors, which have greater potential for the learning process because they inherently carry more information.

5.2.3 Experimental results

In the experiments, we have compared two types of prioritized sampling algorithms with the baseline one, which uses only uniform sampling *DQ-U*. First prioritized sampling algorithm *DQ-TD* combined uniform sampling and prioritization based on the immediate reinforcement component of TD error value being non-zero, while the *DQ-ETD* combined two criteria for prioritization: entropy of the state space vector $H(s_t)$ and the reinforcement value one.
Figure 5.6 shows the comparison between the three different algorithms applied to

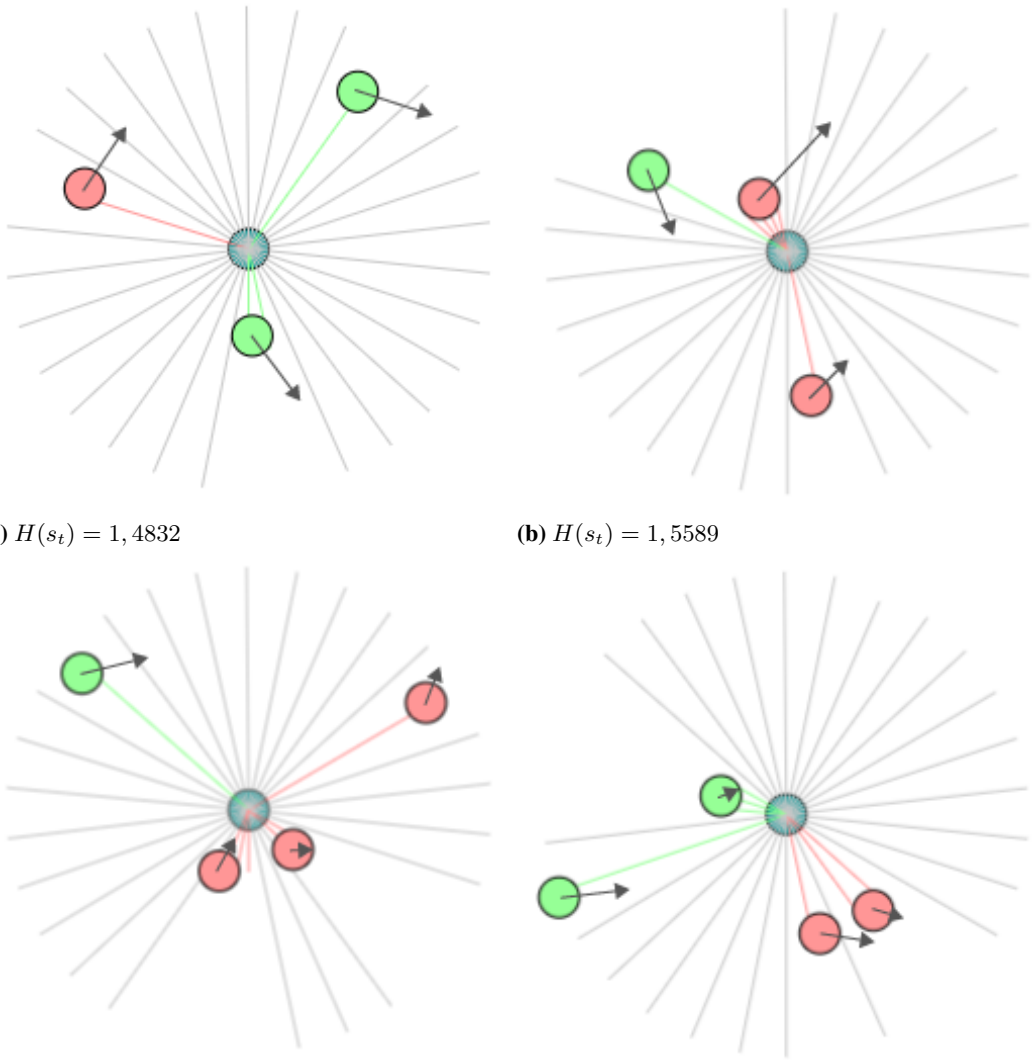


(a) $H(s_t) = 1,0723$ **(b)** $H(s_t) = 1,1556$

(c) $H(s_t) = 1,2035$ **(d)** $H(s_t) = 1,3711$

Figure 5.3: *State spaces with low entropy*

our experimental setup; First algorithm *DQ-U* represents the baseline as it utilizes only uniform sampling, with no prioritization. Algorithm *DQ-TD* uses prioritization based on reinforcement value only, while *DQ-ETD* combines the entropy and TD error criteria based on the reinforcement value being non-zero as shown in 8.
From Figure 5.6 we can see that the *DQ-ETD* method outperforms both the baseline *DQ-U* and *DQ-TD* method based on TD error prioritization only.
From these results, we can notice that adopting an additional prioritization criterion based on the state space properties can significantly improve the efficiency of the prioritization mechanism.



(a) $H(s_t) = 1,4832$ **(b)** $H(s_t) = 1,5589$

(c) $H(s_t) = 1,6985$ **(d)** $H(s_t) = 1,7895$

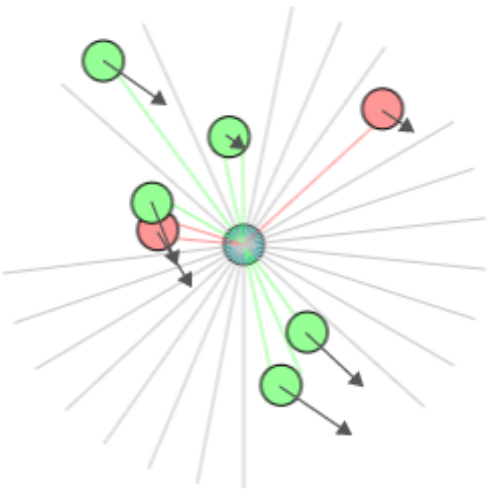
Figure 5.4: *State spaces with medium entropy*

5.2.4 Limitations

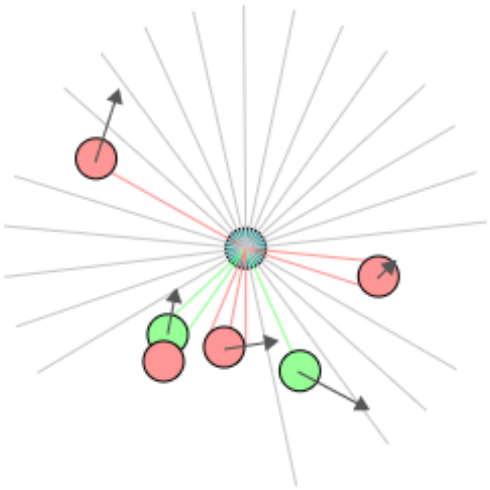
Greedy sampling on the prioritization criteria in both *DQ-TD* and *DQ-ETD* introduces a bias which is tolerable in the early stages of learning, but it may violate the convergence guarantee of the Equation 1.3, and therefore may prevent the agent to obtain an optimal policy π in the long run. For this reason, adjusted annealed importance sampling [41] can be used to compensate for the bias.

5.3 Conclusion

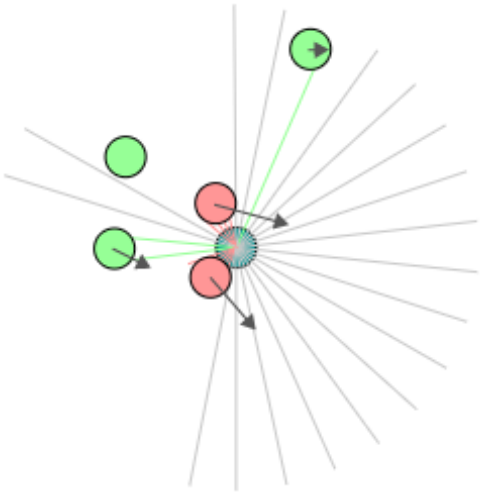
A novel approach was presented to sample replay memory which includes a new criterion for prioritization based on the entropy of the state space vector called *DQ-ETD*. Experimental results have shown that *DQ-ETD* can outperform the prioritized



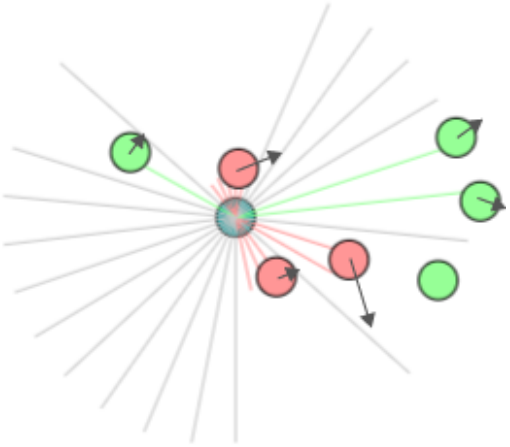
(a) $H(s_t) = 1,8633$



(b) $H(s_t) = 1,9608$



(c) $H(s_t) = 2,0610$



(d) $H(s_t) = 2,1244$

Figure 5.5: *State spaces with high entropy*

sampling approaches based on the TD error component criterion only such as *DQ-TD* in the early stages of the learning process.

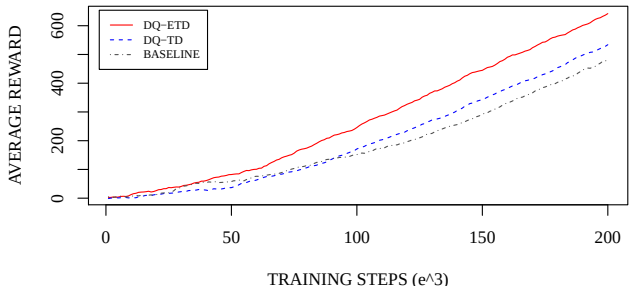


Figure 5.6: *Comparison between average reward values in 20 learning epochs respectively with DQ-ETD, DQ-TD, and the baseline DQ-U, over first 200K learning steps*

CHAPTER 6

Temporally Extended Actions

In the previous chapter we have seen that the selective sampling into replay memory using *Cognitive Filter* can be used to improve the speed of convergence of the reinforcement learning algorithm. However, in this chapter the mechanism of *replay memory* is exploited in a very different way making it possible to efficiently learn temporally extended actions found in hierarchical learning.

6.1 Delayed Memory Reward

This work takes the advantage of the *Replay Memory* structure by introducing an approach called *Delayed Memory Reward* or DMR that enables agents to learn the preference for initiating temporally extended actions while performing interactions with agents of different social dispositions. Since the outcome of interacting socially with other agents can be beneficial or not depending on their dispositions the interacting agent is able to learn to avoid or flock together with agents of specific social dispositions. *Delayed Memory Reward* achieves the hierarchical learning by *a posteriori* rewarding the specific primitive transitions stored in its memory. This enables an agent to learn the preference for the temporally extended actions based on their final outcomes simply by changing the reinforcement of the primitive stored transitions that led to their instantiation.
We demonstrate the advantage of *DRM* method by evaluating it on a complex multi-agent environment in which the agents perform Q-learning at different levels of temporal abstractions while consuming the food sources directly by exploring or through social interaction with other learning agents.

6.1.1 Theoretical Background

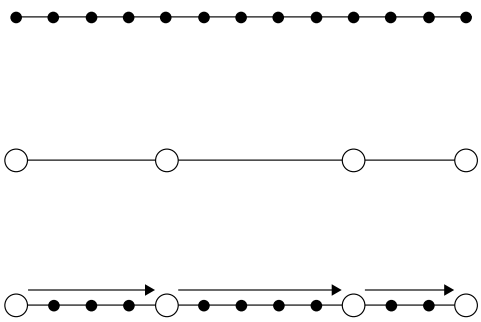
Temporal abstractions

Defining complex actions that span across sequences of primitive ones is the main idea behind Hierarchical Machine Learning approach [4]. For example, a temporally extended action could be *open the doors* or *go to the kitchen* and the primitive ones could be *pull handle, push the door, go left, go right, go back, go forward*. To perform a complex action an agent has to select the right sequence of primitive actions that bring it to achieve the goal, e.g., the right combination of the primitive movements in order to *go to the kitchen*.
Usually, an agent selects the primitive actions based on the learned policy π in order to maximize the expected reinforcement return in the long run, but during a temporally extended action it selects primitive ones based on a different policy π_g that could be, for example, the most efficient way to *go to the kitchen*.

MDPs and SMDPs

In order to introduce temporal abstractions as a minimal extension of the learning framework we can build upon a theory of *semi-Markov Decision Processes* or SMDP which represents a special kind of *Markov Decision Process* MDP adapted to model continuous-time discrete-event systems [9, 37, 46].
Since temporally extended actions defined over SMDPs are treated as indivisible and cannot be dissected into smaller, more elemental actions [60] introduced an *Options Framework* in which the actions are defined over both SMDPs and more basic MDPs. Figure 6.1a shows state trajectories of a simple MDP where state changes are defined as discrete time steps over atomic actions while Figure 6.1b represents temporally extended state transitions of variable length defined over more complex SMDPs. We can see that the temporally extended actions, shown in Figure 6.1c as arrows, are defined over both MDP and SDMP, making it possible to represent complex actions defined over variable time spans as a collection of a smaller discrete-time atomic actions defined in simple MDPs.
Temporally extended actions are not seen anymore as indivisible black-boxes but as something that can be broken down into more elemental pieces and analyzed. Moreover, during the execution of a *option* time-extended action an agent can perform learning steps on a lower MDP level and on a higher SMDP one.

In *Options Framework* [60] a temporally extended action is defined by three components: a policy $\pi : S \times A \rightarrow [0, 1]$, a termination condition $\beta : S^+ \rightarrow [0, 1]$ and an initiation set $I \subset S$.When an *option* is taken the actions are chosen according to π until the *option* is terminated stochastically according to the probability β.
Executing a *Markov option* starts by selecting the next action a_t according to probability distribution $\pi(s_t, \cdot)$. After this step the agent makes the transition to the next state s_{t+1} where the option either terminates with probability $\beta(s_{t+1})$ or continues determining a_{t+1} according to $\pi(s_{t+1}, \cdot)$, and so on.



(a) *MDP*

(b) *SMDP*

(c) *Options over MDP*

Figure 6.1: *State trajectories of discrete time steps MDPs compared with SMDPs and Temporally extended actions defined over both MDPs and SMDPs.*

6.1.2 Delayed Memory Reward Hierarchical Learning

Rewarding complex temporally extended actions proved to be a challenge since they span across multiple primitive actions and their outcome is known only after its termination. Most common techniques include a cumulative discounted reward of primitive actions taken from the initiation to the termination step of the temporally extended action [60].

Model Architecture and Learning Algorithm

We have approached the temporally extended actions in a different way, taking into consideration only the primitive transition that instantiated it and whether the complex goal was achieved at the termination condition. For example, if our action was *open the door*, depending on the final outcome *door opened* or *not (door open)*, we can provide either positive or negative reinforcement to the instantiated transition. Since the N latest primitive transitions are stored in the agent's *Replay Memory*, we can modify the reinforcement values of the specific instantiation transitions after the termination state of the temporally extended actions, and thus enable the hierarchically higher reinforcement to propagate back to the primitive atomic transitions as shown in Algorithm 9. For the purpose of identifying the primitive transition that instantiated the complex action in the *Replay Memory*, we have modified the standard transition tuple (s_t, a_t, r_t, s_{t+1}) to include a unique identifier u_{id}.
By default, an agent selects its next action that maximizes the state-action value $Q(s, a)$ until the initiation state of the temporally extended action. After it behaves according to the complex action policy: primitive actions are selected according to the probability distribution $\pi_g(s_t, \cdot)$. During the course of the complex action, the agent still performs primitive learning steps according to the immediate rewards r_t. When a termination condition of the complex action is met and the outcome is known, a complex action reward r_g is determined and propagated back to the primitive instantiation transition identified by u_{id}.

6.1.3 Experimental setup

To evaluate the proposed model we have created a multi-agent learning environment that consists of multiple agents along with randomly moving good/bad food pieces, based on the environment described in Section 2.2.4. Food pieces are generated at a random position with random speed and direction, and move in a constrained environment by bouncing on the walls. Agents move in the same environment and should learn to touch (eat) good food pieces and to avoid bad food pieces either directly or by initiating a *social contact* when they touch (communicate) with other agents. *Social contact* also represents an initiation state for a temporally extended *social action* with a termination condition of consuming the target food source shared by the agent with whom the contact was made.

There are two classes of agents defined in the environment: good agents that upon social contact share a good food source as a goal to the agent with which it communicated, and bad agents that share bad food, initiating a non beneficial temporally extended action for the other agent.
We define r_g used in Algorithm 9 as the outcome reinforcement of temporally extended action which is equal to the reinforcement of termination condition transition: the transition that results by consumption of the goal food source which can be positive or negative depending on the disposition of the agent that initiated it.

The goal of each agent is to consume as much good food pieces as possible, while, in turn, try to avoid the bad food sources using direct exploration or indirect complex social contact action. After being consumed, new food pieces of the same type of the consumed ones are re-generated with a random position, speed, and direction, thus keeping the distribution of food constant. Agents receive reinforcement +1 for consuming good food pieces and -1 for consuming bad ones.

The state space is continuous and intentionally high-dimensional for the purpose of increasing the entropy and consequently the diversity of possible experience transitions. Each agent has 30 directional sensors and each of them can perceive 7 continuous variables: distance of sensed object (good food, bad food, good agent, bad agent, wall), the first two of which have the two additional attributes: speed in x direction and speed in y direction; this gives a total of 210 state inputs for each agent.
As a function approximator we adopt a deep neural network with two fully connected hidden layers of 120 and 30 neurons, respectively, with weights Θ to approximate $Q(s, a; \Theta) \approx Q^*(s, a)$. To reduce the computational complexity of having multiple forward steps each time, we want to find an action that maximizes the state-action function $\arg\max_a Q(s, a)$; the network takes the state vector s as an input and predicts $Q(s, a)$ for each possible action.
We have adopted the original Q-learning update formula with a learning rate α set to a low value (0.05) because of the nature of the approximator, and discount factor $\gamma = 0.9$. The default capacity of the replay memory buffer D included 7000 experiences. The experimental setup performed *Q-learning* in a multi-agent environment consisting of 4 agents for each disposition (bad, good) learning simultaneously while interacting with each other.

6.1.4 Experimental results

In the experiments, we have compared two identical setups: one used *Delayed Memory Reward* method for learning temporal abstractions (*DRM-Q*) while the other one (*baseline*) performed temporally extended actions without propagating r_g value back to the first transition. From Figure 6.2 and Figure 6.3, we can see that experiments that used the *Delayed Memory Reward* technique gave better overall performance in both agent disposition types, which is particularly evident in the average total score of the good agent type shown in Figure 6.3.
Influence of the *DRM* method on the behavioral characteristics of different agent types is more noticeable when we take into consideration the difference between the social score which represents the cumulative reward that agent scored through temporally extended action of social contact and exploration score which is the cumulative reward of eating food pieces directly using the basic primitive actions.
Figure 6.4 shows that in the experiments that were using the *DMR* agents moved from relying more on exploration after about 150K steps to heavily focusing on social, while they learned to gravitate to the agents of good disposition which provided temporally extended actions with positive outcome. Agents with the good disposition also learned to take more temporally extended social action when *DMR* was active, but in a more conservative amount compared to their bad counterparts since they were the learning focus of both dispositions given their positive nature of the social contact outcome.

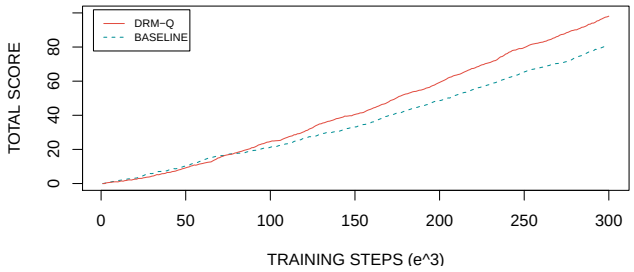


Figure 6.2: *Comparison between average total reward score of bad agent types in 50 learning epochs of Q-learning with Delayed Memory Reward (DMR-Q), and the baseline without the DMR technique, over first 300K learning steps*

6.2 Conclusion

An approach for learning temporal abstractions *DMR-Q* is presented that takes into advantage the structure widely used in deep Q-learning called *Replay Memory*. Experimental results have shown that *DMR-Q* is effective in learning dispositions towards temporally extended actions with different outcomes only by modifying the reinforcement of the primitive transitions in agents *Replay Memory*.
An advantage of using *DMR-Q* is that it does not require re-defining or extending the action and state spaces in order to accommodate the temporal abstractions since it targets only the structure of a primitive action transitions.

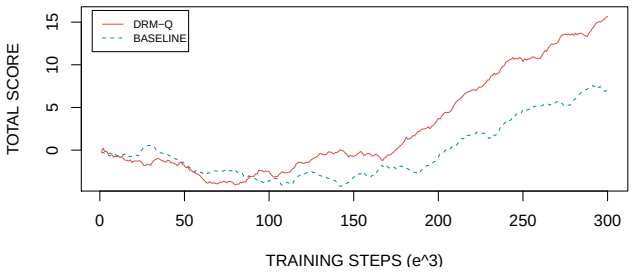


Figure 6.3: *Comparison between average total reward score of good agent types in 50 learning epochs of Q-learning with Delayed Memory Reward (DMR-Q), and the baseline without the DMR technique, over first 300K learning steps*

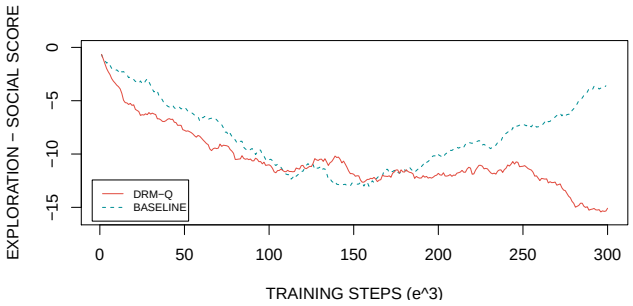


Figure 6.4: *Comparison between average difference of exploration and social rewards from bad agent types in 50 learning epochs of Q-learning with Delayed Memory Reward (DMR-Q), and the baseline without the DMR technique, over first 300K learning steps*

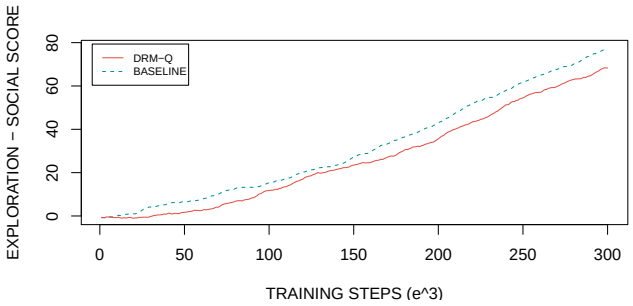


Figure 6.5: *Comparison between average difference of exploration and social rewards from good agent types in 50 learning epochs of Q-learning with Delayed Memory Reward (DMR-Q), and the baseline without the DMR technique, over first 300K learning steps*

Algorithm 9 Deep Q-learning with delayed-memory reward

Initialize replay memory D with capacity N. Complex action policy $\pi_g : S \times A \to [0, 1]$, a termination condition $\beta : S^+ \to [0, 1]$, initiation set $I \subset S$ and goal flag $F_g = false$
Initialize action-value function Q with random weights
for episode = 1, M **do**
  for t = 1, T **do**
    if $F_g = false$ **then**
      With probability ϵ select a random action a_t
      otherwise select $a_t = \arg\max_a Q^*(s_t, a; \Theta)$
    else
      Select a_t according to the probability distribution $\pi_g(s_t, \cdot)$
    end if
    Execute action a_t, observe reward r_t and state s_{t+1}
    if $s \in I$ **then**
      Set $F_g = true$
      Store U_{id} of the transition
    end if
    if Termination condition according to probability β **then**
      Set r_g according to the outcome
      Set $r_t = r_g$ of the transition from D where $u_{id} = U_{id}$
      Set $F_g = false$
    end if
    Store transition $(s_t, a_t, r_t, u_{id}, s_{t+1})$ in D
    Sample random batch of transitions $(s_t, a_t, r_t, u_{id}, s_{t+1})$ from D

$$\text{set } y_i = \begin{cases} r_i, & \text{terminal } s_{i+1} \\ r_i + \gamma \max_{a'} Q(s_{i+1}, a'; \Theta), & \text{non terminal} \end{cases}$$

    Perform a gradient descent step on $(y_i - Q(s_i, a_i; \Theta))^2$ according to Equation 1.12
  end for
end for

CHAPTER 7

Conclusion

Environment is everything; we constantly remind ourselves of this fact when it comes to designing reinforcement learning problems where the ultimate goal is to enable an artificial agent to best adapt to its surrounding reality by learning to take better choices over time. We are often so focused on this reality that we neglect the fact that when we replace it with a finite amount of features that are either handcrafted or extracted using some algorithm we are actually modeling a *perception* of reality. This is also the case with the evolution shaping our brains in order to become a very efficient information processing system by enabling us to internally model the unfathomable informational potential given by the entropy of the surrounding external world. Each occurrence in nature has an almost unlimited informational potential, the perception of it alone is determining with how much entropy will we perceive, will the potential information received from it be highly compressed or more raw. If we take for example a human being analyzing a car, there is a limited amount of information one can notice, for example, the car is blue, it is a sports car, and it is currently moving at a low speed. What if, we possessed an x-ray vision that would allow us to see the inside of the car, the amount of information we can gather now is higher; for example, we can notice the gas level in the tank, etc.. This would require our brain to engage more mental processing power. And on an even more extreme level, if we could perceive every atom of that car or more precisely the spin of its electrons, every one of them would have an entropy of 1 bit (taking aside the quantum effects), which would amount for an enormous amount of information our brain would never be able to process given a simple car. To fight this potential information overload the evolutionary process allowed us to develop perceptive mechanisms that are able to process an event that had theoretically infinite potential for information into a 1 bit of a single neuron firing in our brain in a perfect oversimplified case,

starting a complex activation procedure that would at the end produce some form of behavior.
Our mental state, traits, mood, emotion that we are feeling affect the way we perceive reality as they alter the way the information is processed in our brains. For example, if we are in a state of hunger everything becomes a potential source of food because our perception tends to focus its resources on the current goal of satisfying its physiological needs. This cognitive process is so important that it actually generates information even if its not actually there, for example people depraved of food and water hallucinating and seeing an oasis in the desert. The process of perception is so crucial that for those people an oasis becomes part of their reality. The fact that perception is affected by our mental state and its constantly changing how we perceive the reality brings us to the main premise of this work: the perception modeling in reinforcement learning can be seen as a more than a rigid one-time feature design, but represented by a dynamic and state responsive mechanism that can by itself be capable of eliciting behavioral changes during an agent learning process. Because the agent actions depend solely on the information gained through its state representation, the dynamic processing of the information by a dynamic perceptual mechanism is bound to alter the action selection in a way that is responsive to the logic of the selection. This creates a secondary drive that complements the main meta-drive that is driven by the sheer reinforcement. The learning process is still driven by the main drive of maximizing its reinforcement reward in order to reach an optimal behavior, but the way of getting to it is affected by the dynamic, subtle perception. This is possible because the agent in any given time may have multiple options in terms of actions that are each optimal to the main reinforcement drive so the final choice of a specific action has the possibility of being driven by the perception layer instead. The premise is that a simple difference of selecting information in the perception layer of the agent can lead to an emergence of complex and multi-layered behavioral patterns.
This simplicity is seen also in the mechanisms of our brain when our mental processing energy is focused using a cognitive buffer called *working memory*, just by permeating it with experiences that are the focal point. It is a key ingredient in conscious reasoning as it contains the "working material" or the objects that we use to reason about and its capacity can determine the underlying difference in people's performance. The agent we considered, instead of reasoning, is performing a learning process, by iteratively updating the ANN function approximator using *stochastic gradient descent* or SGD. In theory, the *gradient descent* should be at each iteration performed on all of the agent experiences in order to ensure the best approximation, but, since this is unpractical in most real world, huge state spaces, we use a SGD which is an extreme case of *gradient descent* in which we perform an update on only one data sample. Most algorithms use a *mini batch gradient descent*, which provides a compromise between the two extremes by randomly selecting a certain amount of the data samples from the dataset and using them to update the gradients. This is exactly what happens in Q-learning, as presented in this work; the difference from standard *mini batch* approach is that the samples are not selected randomly from the whole dataset of experiences, but from a sliding window *replay memory*. This allows us to dynamically re-train our approximator on the contents of selected samples from

replay memory in order to achieve an artificial attention focusing mechanism that we call *cognitive filter*. The attention process alters the way the approximator is trained in a way to allow the focal memories to be better approximated at the expense of the rest of the memory and lead us to modified behavior of an agent.
The experiments that were performed using *cognitive filter* presented in chapter 3, chapter 4 and chapter 5 differed in the selection of criteria that in term modified the sample probabilities and led to an emergence of a behavioral changing secondary drivers. Experiments in chapter 3 were motivated by modeling some of the important human personality traits found in FFM just by changing the perception of the agent using *cognitive filter* mechanism. First experiment outlined in section 3.1 is using a sampling criterion of *information gain* in order to create a secondary behavioral drive that would represent the characteristics commonly associated with *Openness to Experience* personality spectrum. Second experimental setup from the section 3.2 of chapter 3 used the same mechanism of *cognitive filter* in a multi-agent social setting and the sampling priorities were determined by the type of transition or whether it resulted in a social or exploration reinforcement. The sampling also simulated the difference in *cognitive bandwidth* that correlated with the main personality axis of Extraversion-Introversion. Both experimental setups provided the results that have confirmed that the secondary drive effects are noticeable in the way the agents perform in the environment. The effect of a change in the agents perception induced complex behavioral patterns that significantly improved the adaptability to a variations in the environment. Some of the traits adapted better to a variation that contained more negative reinforcement while others better adapted to the environment with scarce reinforcement that contained less amount of both positive and negative one. A social multi-agent sister where agents are able to communicate among each other bring us to the opportunity of being able to model agents that are more socially engaged and those that prefer to explore the environment without so much interaction with others. By tweaking the ratio between the two types we can achieve the intrinsic dynamics of a group social structure.
Experiments done with *genetic algorithms* performed in chapter 4 allowed us to evolve a perception that was best adapted to a specific environment and this showed us that the properly developed perception mechanism can greatly improve the efficiency of learning while producing complex secondary behavioral drives that complement the main one. Furthermore, there's an opportunity to develop different modes of perception that are adapted to a different variations of the environment and then use them as the specific variation occurs, for example environment in day and night mode. If we are found in an environment that is over-saturated with informational content, chaotic and high in entropy such as our day mode an artificial perception driven by *cognitive filter* mechanism is especially important when reducing the cognitive load by focusing on the experiences that matter and filtering out the background noise. While in the night mode where there's not much going on the *cognitive filter* is not focused on filtering out noise but making use of the low sensory input the best it can.
chapter 5 and chapter 6 instead of focusing on the behavioral aspect deal with the general improvement of the reinforcement learning process. Experimental results from chapter 5 show how can *cognitive filter* with addition of a new criterion based on the

entropy of the state improve the speed of convergence of the algorithm and chapter 6 shows us how can an algorithm make use of the mechanism of *replay memory* in order to be able to learn complex actions that span across multiple learning steps.
The main contribution of this work is an underlying mechanism that is present across the chapters; the novel approach of modeling artificial perception in reinforcement learning by selectively filtering the agents raw cognitive stream. Experimental results show us that this seemingly simple mechanism is able to make a great difference to a reinforcement learning process by making it more dynamic and behaviorally complex.

List of Figures

Acronyms

ANN Artificial Neural Network.

BAS Breadth of Attention Sampling.

CARL Context Augmented Reinforcement Learning.

DMR Delayed Memory Reward.

DP Dynamic Programming.

DQL Deep Q-learning.

DRAM Dynamic Random Access Memory.

FMM Five Factor Model.

FMRI Functional Magnetic Resonance Imaging.

GA Genetic Algorithm.

IG Informational Gain.

MDP Markov Decision Process.

NEO Neuroticism Extraversion Openness.

OTE Openness to Experience.

ReLU Rectified Linear Unit.

RL Reinforcement Learning.

SGD Stochastic Gradient Descent.

SMDP Semi-Markov Decision Process.

SPD Sensory Processing Disorder.

TD Temporal Difference.

VEI Variety of Experience Index.

WM Working Memory.

Bibliography

[1] Hyungil Ahn and Rosalind W Picard. Affective-cognitive learning and decision making: A motivational reward framework for affective agents. In *International Conference on Affective Computing and Intelligent Interaction*, pages 866–873. Springer, 2005.

[2] A Jean Ayres and Jeff Robbins. *Sensory integration and the child: Understanding hidden sensory challenges*. Western Psychological Services, 2005.

[3] Alan D Baddeley and Graham Hitch. Working memory. *Psychology of learning and motivation*, 8:47–89, 1974.

[4] Andrew G Barto and Sridhar Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(4):341–379, 2003.

[5] Marwen Belkaid, Nicolas Cuperlier, and Philippe Gaussier. Emotional metacontrol of attention: Top-down modulation of sensorimotor processes in a robotic visual search task. *PloS one*, 12(9):e0184960, 2017.

[6] Daniel E Berlyne. Conflict, arousal, and curiosity. 1960.

[7] Mark E Bouton. Context, time, and memory retrieval in the interference paradigms of pavlovian learning. *Psychological bulletin*, 114(1):80, 1993.

[8] Mark E Bouton. Context, ambiguity, and classical conditioning. *Current directions in psychological science*, 3(2):49–53, 1994.

[9] Steven J Bradtke and Michael O Duff. Reinforcement learning methods for continuous-time markov decision problems. In *Advances in neural information processing systems*, pages 393–400, 1995.

[10] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[11] L Cosmides. Deduction or darwinian algorithms? an explanation of the" elusive" content effect on the wason selection task (evolution, sociobiology, logic, cooperation, modularity). 1987.

[12] Leda Cosmides. The logic of social exchange: Has natural selection shaped how humans reason? studies with the wason selection task. *Cognition*, 31(3):187–276, 1989.

[13] Leda Cosmides and John Tooby. From evolution to behavior: Evolutionary psychology as the missing link. 1987.

[14] Leda Cosmides and John Tooby. *From evolution to adaptations to behavior: Toward an integrated evolutionary psychology.* Ablex Publishing, 1995.

[15] Paul T Costa Jr, Robert R McCrae, and David A Dye. Facet scales for agreeableness and conscientiousness: A revision of the neo personality inventory. *Personality and individual Differences*, 12(9):887–898, 1991.

[16] Mihaly Csikszentmihalyi. Flow: The psychology of optimal performance. *NY: Cambridge UniversityPress*, 40, 1990.

[17] Mihaly Csikszentmihalyi. Toward a psychology of optimal experience. In *Flow and the foundations of positive psychology*, pages 209–226. Springer, 2014.

[18] Jan W de Fockert, Geraint Rees, Christopher D Frith, and Nilli Lavie. The role of working memory in visual selective attention. *Science*, 291(5509):1803–1806, 2001.

[19] John M Digman. Personality structure: Emergence of the five-factor model. *Annual review of psychology*, 41(1):417–440, 1990.

[20] Paul E Downing. Interactions between visual working memory and selective attention. *Psychological science*, 11(6):467–473, 2000.

[21] Bogdan Draganski, Christian Gaser, Volker Busch, Gerhard Schuierer, Ulrich Bogdahn, and Arne May. Neuroplasticity: changes in grey matter induced by training. *Nature*, 427(6972):311, 2004.

[22] Randall W Engle. Working memory capacity as executive attention. *Current directions in psychological science*, 11(1):19–23, 2002.

[23] M. W. Eysenck. *Attention and Arousal, Cognition and Performance*. Springer, 1982.

[24] Mônica R Favre, Deborah La Mendola, Julie Meystre, Dimitri Christodoulou, Melissa J Cochrane, Henry Markram, and Kamila Markram. Predictable enriched environment prevents development of hyper-emotionality in the vpa rat model of autism. *Frontiers in neuroscience*, 9:127, 2015.

[25] Sandra Clara Gadanho and John Hallam. Exploring the role of emotions in autonomous robot learning. *DAI RESEARCH PAPER*, 1998.

[26] Adam Gazzaley and Anna C Nobre. Top-down modulation: bridging selective attention and working memory. *Trends in cognitive sciences*, 16(2):129–135, 2012.

[27] Eleanor J Gibson. Exploratory behavior in the development of perceiving, acting, and the acquiring of knowledge. *Annual review of psychology*, 39(1):1–42, 1988.

[28] Engin Ipek, Onur Mutlu, José F Martínez, and Rich Caruana. Self-optimizing memory controllers: A reinforcement learning approach. In *ACM SIGARCH Computer Architecture News*, volume 36, pages 39–50. IEEE Computer Society, 2008.

[29] OP John. Big five prototypes for the adjective check list using observer data. In *OP John (Chair), The Big Five: Historical perspective and current research. Symposium conducted at the annual meeting of the Society for Multivariate Experimental Psychology, Honolulu*, 1989.

[30] Andrej Karpathy. Reinforcejs framework. `https://github.com/karpathy/reinforcejs`, 2013. Accessed: 2018-09-06.

[31] Joseph Kasof. Creativity and breadth of attention. *Creativity Research Journal*, 10(4):303–315, 1997.

[32] Mehdi Khamassi, Stéphane Lallée, Pierre Enel, Emmanuel Procyk, and Peter F Dominey. Robot cognitive control with a neurophysiologically inspired reinforcement learning model. *Frontiers in neurorobotics*, 5:1, 2011.

[33] Torkel Klingberg. *The overflowing brain: Information overload and the limits of working memory*. Oxford University Press, 2009.

[34] Dharshan Kumaran, Demis Hassabis, and James L McClelland. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in Cognitive Sciences*, 20(7):512–534, 2016.

[35] Matthew D Lieberman. Introversion and working memory: Central executive differences. *Personality and Individual Differences*, 28(3):479–486, 2000.

[36] Long-Ji Lin. Reinforcement learning for robots using neural networks. Technical report, DTIC Document, 1993.

[37] Sridhar Mahadevan, Nicholas Marchalleck, Tapas K Das, and Abhijit Gosavi. Self-improving factory simulation using continuous-time average-reward reinforcement learning. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, pages 202–210. MORGAN KAUFMANN PUBLISHERS, INC., 1997.

[38] Robert R McCrae and Oliver P John. An introduction to the five-factor model and its applications. *Journal of personality*, 60(2):175–215, 1992.

[39] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[40] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[41] Radford M Neal. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, 2001.

[42] Daniel Nettle. *Personality: What makes you the way you are*. Oxford University Press, 2009.

[43] Pierre-Yves Oudeyer, Frdric Kaplan, and Verena V Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2):265–286, 2007.

[44] Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:6, 2009.

[45] Daniela Pacella, Michela Ponticorvo, Onofrio Gigliotta, and Orazio Miglino. Basic emotions and adaptation. a computational and evolutionary model. *PLoS one*, 12(11):e0187463, 2017.

[46] Ronald Parr and Stuart J Russell. Reinforcement learning with hierarchies of machines. In *Advances in neural information processing systems*, pages 1043–1049, 1998.

[47] Mirza Ramicic and Andrea Bonarini. Attention-based experience replay in deep q-learning. In *Proceedings of the 9th International Conference on Machine Learning and Computing*, pages 476–481. ACM, 2017.

[48] Mirza Ramicic and Andrea Bonarini. Entropy-based prioritized sampling in deep q-learning. In *Image, Vision and Computing (ICIVC), 2017 2nd International Conference on*, pages 1068–1072. IEEE, 2017.

[49] Timothy Rumbell, John Barnden, Susan Denham, and Thomas Wennekers. Emotions in autonomous agents: comparative analysis of mechanisms and functions. *Autonomous Agents and Multi-Agent Systems*, 25(1):1–45, 2012.

[50] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[51] Massimiliano Schembri, Marco Mirolli, and Gianluca Baldassarre. Evolution and learning in an intrinsically motivated reinforcement learning robot. In *European Conference on Artificial Life*, pages 294–303. Springer, 2007.

[52] Jürgen Schmidhuber. Curious model-building control systems. In *Neural Networks, 1991. 1991 IEEE International Joint Conference on*, pages 1458–1463. IEEE, 1991.

[53] Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pages 222–227, 1991.

[54] Jürgen Schmidhuber. *What's interesting?* Istituto Dalle Molle Di Studi Sull'Intelligenza Artificiale, 1997. Technical report IDSIA-35-97.

[55] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[56] Satinder Singh, Richard L Lewis, Andrew G Barto, and Jonathan Sorg. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*, 2(2):70–82, 2010.

[57] Norman E Spear. Retrieval of memory in animals. *Psychological Review*, 80(3):163, 1973.

[58] Jan Storck, Sepp Hochreiter, and Jürgen Schmidhuber. Reinforcement driven information acquisition in non-deterministic environments. In *Proceedings of the international conference on artificial neural networks, Paris*, volume 2, pages 159–164. Citeseer, 1995.

[59] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[60] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[61] Endel Tulving and Donald M Thomson. Encoding specificity and retrieval processes in episodic memory. *Psychological review*, 80(5):352, 1973.

[62] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[63] Robert W White. Motivation reconsidered: The concept of competence. *Psychological review*, 66(5):297, 1959.

[64] Jianwei Zhai, Quan Liu, Zongzhang Zhang, Shan Zhong, Haijun Zhu, Peng Zhang, and Cijia Sun. Deep q-learning with prioritized sampling. In *International Conference on Neural Information Processing*, pages 13–22. Springer, 2016.