A Broad-persistent Advising Approach for Deep Interactive Reinforcement Learning in Robotic Environments

Submitted as Master Dissertation in SIT724

SUBMISSION DATE

T2-2021

Hung Son Nguyen

STUDENT ID 220069106

COURSE - Master of Applied Artificial Intelligence (Professional) (S737)

Supervised by: Dr. Francisco Cruz, A/Prof Richard Dazeley

Abstract

The current market for robots in domestic environments is growing nowadays. Robots are expected to do simple household tasks to help people release themselves from their personal jobs. While industrial robots are said to compete with humans, as one of the serious reasons that cause job losses from workers, domestic robots are strongly supported because of the purpose of unleashing human labor. However, in order to do these simple tasks, robots need to complete a lot of related sub-tasks such as knowledge, understanding of the environment, decision making, learning strategies, and various human behaviours. Reinforcement Learning (RL) is an area of machine learning where an agent interacts with the environment to find an optimal policy to perform a particular task. The agent tries to learn how to maximise the obtained reward by choosing the best action in every time step. It is recently widely used in robotics to learn about the environment and acquire behaviors autonomously. DeepRL includes Reinforcement Learning (RL) with neural networks approaches, to solve the problems in continuous action-state spaces and more complex real-world domains. Moreover, interactive feedback, where an external trainer or expert gives the advice to help learners choosing actions to speed up the learning process, will be also included in the DeepRL approach for speeding up the learning process by including a trainer providing extra information to the robot in real time. However, current research has been limited to interactions that offer actionable advice to the state of the agent only. Additionally, the information is discarded by the agent after a single use that causes a duplicate process at the same state for a revisit. In this paper, we present BPA, a broad-persistent advising approach that retains and reuses the processed information. It not only helps trainers to give more general advice relevant to similar states instead of only the current state but also allows the agent to speed up the learning process. We test the proposed approach in two continuous robotic scenarios, namely, a cart-pole balancing task and a simulated robot navigation task. The obtained results show that the performance of the agent using BPA improves while keeping the number of interactions required for the trainer in comparison to the DeepIRL approach.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Robot development has been achieved big steps of improvement and gains more attention in recent years. This success does not only come from industrial areas where robots are gradually replacing humans [16], but also known in the domestic areas. Their presence in domestic environments is still limited, mainly due to the presence of many dynamic variables [15] and safety requirements [37]. Intelligence robots in the future should be able to know and detect users, learn action objects, select opportunities, and learn to behave in domestic scenarios. To successfully perform these complex tasks, robots face many challenges such as pattern recognition, navigation and object manipulation all in different environmental conditions. That is, robots in the domestic environment need to be able to continuously acquire and learn new skills.

Reinforcement Learning (RL) is a method applied for a robot controller in order to learn optimal policy through interaction with the environment, through trial and error [36]. The policy defines which action the agent should take when it is in a certain state. With the current policy, after the agent tries to select and execute an action, it will receive a reward signal provided by the reward function defined in advance by the environment designer. This reward reflects the quality of the actions performed by the agent and then the policy will be updated after doing these steps. The final goal is to learn a policy that maximises the total cumulative reward.

DeepRL is an alternative based on the RL structure but also adds Deep Learning (DL) to supply the function which approximates the value of state in continuous action-state spaces.

DeepRL combines the advantages of DL with RL and can realise the end-to-end autonomous learning and control with the raw high-dimensional environment input information that is mapped to the behaviour actions in real-world domains [19].

The applying of using RL in the context of domestic robots has also been surveyed in the full context of scenarios with different learning algorithms like Q-learning and SARSA with exploration strategies like ϵ-greedy, softmax, VDBE, and VDBE-Softmax [15]. The result showed that there is great potential for using RL in robots. Especially, Deep Reinforcement Learning (DeepRL) has also achieved promising results in manipulation skills [40, 31], and on how to grasp as well as legged locomotion [21]. However, there is an open issue relating to the performance in the RL and DeepRL algorithms, which is the excessive time and resources required by the agent to achieve acceptable outcomes [13, 3]. The larger and complex the state space is, the more computational costs will be spent to find the optimal policy.

Among of different approaches to speed up this process, there is one promising method named Interactive Reinforcement Learning (IRL) that can improve convergence speed and has shown its feasibility. IRL allows a trainer to give advice or evaluate a learning agent's behaviour [11]. There are two techniques to giving advice in IRL. First, reward shaping uses which is additional rewards used to guide the agents and it has shown that can accelerate the learning process [30]. However, it may create a different scenario compare with the context of the target. The solution for the context with reward shaping no longer as same as for context without reward. Second, policy shaping tries to maximise the information gained from human advice feedback by using it to modify policy. It can work more effectively with unusually and inconsistently human feedback in the real world [20]. Combining IRL with DeepRL gives a model of Deep Interactive Reinforcement Learning (DeepIRL) which can be used in continuous space with improved learning speed [29]. The approaches using external trainers allow the learning agent to archive more rewards, faster learning time as well as fewer mistakes in comparison to the traditional DeepRL approach. However, current techniques using DeepIRL allow trainers to evaluate or recommend actions based only on the current state of the environment. The advice from the trainer has discarded causes by the agent after a single use, which leads to duplicate processes at the same state for reuse.

On the other hand, persistent advice is a new approach [6] to speed up the learning process by retaining the value from human advice for the next revisit of the state and using policy shaping. Then, the performance of the agent may be improved and the number of interactions in the learning process also appreciably reduced. Although persistent advice is a promising approach at the current, it has been only used on discrete domains like mountain car and self driving car experiments.

Recent advances in the field of domestic robots mentioned above together with their current limitations are the motivation of this thesis. We want to give agents running on DeepIRL

environments the ability to remember the advice and reuse it in the future of decision making.

1.2 Objectives

In this thesis, we explore approaches by combining persistent feedback with DeepIRL techniques. We expect to further improve the learning performance of the agent and facilitate a new method that can apply to continuous domains. We wanted to dig into Deep Interactive Reinforcement Learning (DeepIRL) domain and find answers to these research questions:

- Is possible to extend the persistent IRL approach to continuous representation?
- To what extend can persistent feedback speed up the learning process in continuous RL scenarios?

These questions will be addressed in this document throughout the investigation and experiment about the combination of deep reinforcement learning model and persistent feedback in domestic robots environment. This work introduces the Broad-persistent Advising (BPA) approach for DeepIRL to provide the agent a method for information retention and reuse of previous advice from a trainer. This approach includes two components: broad advice and persistent advice. In this article, we used k-means algorithm and Probabilistic Policy Reuse (PPR) for each component, respectively. Agents using the BPA approach have better results than their non-using counterparts while keeping the number of interactions required for the trainer.

1.3 Structure

The present thesis is organised into four main parts, each one of them is described as follows:

- Introduction: This is the current chapter which briefly describes what motivates this thesis, states the problem along with defining the main research questions, and shows a short brief about contribution as well.
- Literature Review: We introduce the background knowledge which is needed to understand the state of the art and related research around it as well. All of the knowledge will be used throughout the development of the work.
- Research Design: In this chapter, we present the methodology to solve the problem and research questions. This method includes environment setup a domestic scenario

for the robotic agents, implement interactive feedback, the criteria for verifying the results, and project risk as well.

- Artefact Development Approach: We describe in detail the proposed Broad-persistent Advising (BPA) that includes a generalisation model along with a persistent approach to answering the research question: *Is possible to extend the persistent IRL approach to continuous representation?*. We built a generalisation model to solve the problem of applying the persistent approach to continuous domains.

- Empirical Evaluation: We present the above experimental results presented in the research plan chapter to test the effectiveness of the BPA approach. The presentation and explanation will be discussed in each experiment.

- Conclusion: This chapter summarises the contributions of this thesis, discusses some of the research limitations, and identifies some potential topics for further research.

Chapter 2

Literature Review

2.1 Reinforcement Learning Introduction

We introduce the area of RL in general before looking at the definition of further technique. RL is a branch of machine learning in which artificially intelligent agents learn behaviours by interacting with their surroundings [36]. Reinforcement learning tools learn through trial and error by repeatedly interacting with the surrounding environment and learning which actions do and which actions will not produce the expected results. The main idea is inspired by nature itself and based on the learning methods of humans and animals [32]. In cognitive beings, there is a tendency to reapply good behavior, otherwise, there is a tendency to avoid negative behavior in the future. Reinforcement learning was first introduced by psychologist B. F. Skinner [35] in behavioral psychology. Many decades later, reinforcement learning has expanded to computer sciences that provide agents to receive digital rewards based on the value of their actions. The goal of the agents is to maximise the digital return that they get from interacting with the environment. Figure 2.1 shows the traditional learning loop between an RL agent and its environment.

The three basic components ofRL are states, actions, and rewards. A state is a collection of observable information at a specific point in time. For example, in the environment of a movement for a robot, the robot is considered an agent. State information may include speed, position, nearby obstacles are described through the robot's front camera. In general, state space is also known as environment information that is the collection of all things that a robot or agent can receive or encounter at a certain point in time.

An action is performed by the agent that has an impact changes on the environment. The

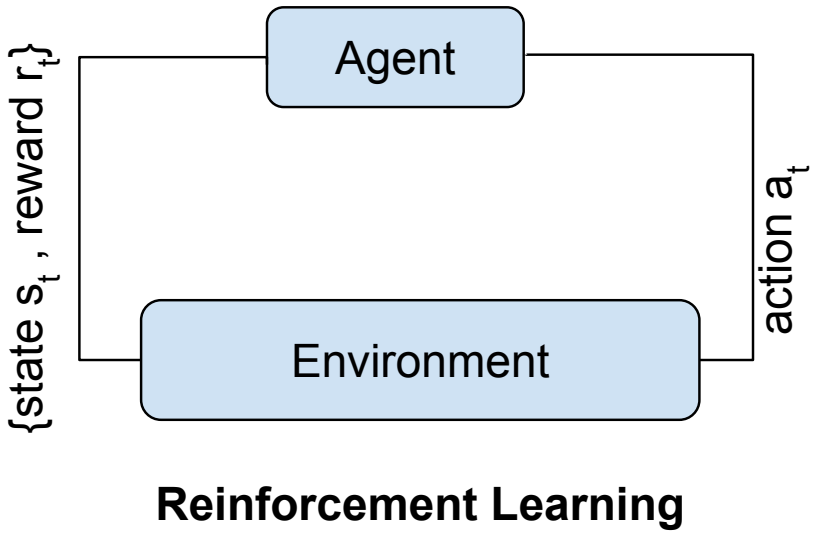


Figure 2.1: Reinforcement Learning framework. At state s_t, the agent performs action a_t and obtaining reward and reaching the next state s_{t+1}.

agent chooses an action depending on the information provided in the current state. For example, after detecting an obstacle in front, the robot can turn left, turn right to avoid, or continue to go straight to hit that block. The action space is a collection of all actions that the agent can perform at a point in time. The action space may differ from the current state to the next state.

At any point in time, the agent monitors the environment and uses state information to decide what action to take. After completing this step, the agent will take a new snapshot of the environment (new states and rewards). The reward is used for measuring the benefits of previous actions to the goal. The agent's goal is to learn which action is optimal for each state. The agent's policy or behavior is the way to choose action at each state. The optimal strategy or optimal policy that will lead to yields the most long-term profit, even if it has to do some action that incurs temporary penalties.

In the RL setup, a machine learning algorithm-controlled agent observes a state s_t from its environment at timestep t. In state s_t, the agent communicates with the environment by performing action a_t. Then the agent moves to a new state s_{t+1} and receive reward r_{t+1} as feedback from environment based on the previous state and the chosen action. Therefore, the reward collected by policy π at timestep t is shown in Equation (2.1). Where r_t is the reward at timestep t. The discount rate γ stands for the importance of rewards in the future. The agent's aim is to find out a policy π that maximises anticipated profit (reward).

$$r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ... = \sum_{k=0} \gamma^k r_{t+k} \tag{2.1}$$

The special advantage of RL lies in the ability to learn without prior information. RL methods make it possible to learn complete and accurate behaviors without the agent having prior knowledge of environmental dynamics or required behaviors [4]. It differs from the supervised learning method in that it does not require clear and correct examples to learn behaviour, but instead uses a reward function to influence the learned behavior.

2.1.1 Marcov Decision Process

Reinforcement Learning is appropriate for studying tasks that may be modelled as Markov Decision Processes (MDP) [36]. An MDP is specified by the tuple (S,A,T,R, γ) where:

- S is a finite set of states in the environment,
- A is a set of actions available in each state,
- T is the transition function $T : S_n \times A \rightarrow S_{n+1}$,
- R is the reward function $R : S \times A \rightarrow R$, and
- γ is a discount factor which is $0 \leq \gamma \leq 1$

As mentioned above, the agent perceives the current state $s_t \in S$ at timestep t and chooses an action $A_t \in A$ to execute it. The environment returns the reward $R_t = R(S_t, A_t)$, and the agent will change to the state $s_{t+1} = T(s_t, a_t)$. The goal of the agent is to learn the mapping state, which is the policy π: $S \rightarrow A$, which can maximise the expected benefits from the environment. Therefore, the equation (2.1) can be denoted as follows:

$$Q^{\pi}(s_t, a_t) = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ... = \sum_{k=0} \gamma^k r_{t+k} \tag{2.2}$$

where $Q^{\pi}(s_t, a_t)$ is the accumulated reward by following the policy π from an initial state s_t. γ is a constant value in range $0 \leq \gamma \leq 1$, that defines the relative value of immediate rewards compared to potential rewards in the future. When $\gamma = 0$, the agent is short-sighted and only seeks to optimise immediate rewards. When γ reach to 1, the agent will be more foresighted and consider future returns.

Otherwise, there is an indicator named learning rate α, $0 \leq \alpha \leq 1$ which defines the step size for the agent's optimisation in an RL agent. The agent will learn faster if the learning rate is higher, but the resulting behaviour may not be optimal. Small learning speeds will result in optimal behaviour, but it will take a long time to achieve. In almost all agents, a constant or decaying learning rate is sufficient.

2.2 Deep Reinforcement Learning

2.2.1 Deep Learning

Deep learning is a type of machine learning technique that employs an artificial neural network to turn a collection of inputs into a set of outputs. A study [25] has shown that deep learning technology usually uses supervised learning with labeled data sets, which can process complex and multi-dimensional original input, such as images. In deep learning, the term ”deep” refers to the amount of layers in the network that the data is transformed into. Figure 2.2 below represents an example of a basic artificial neural network that has one hidden layer.

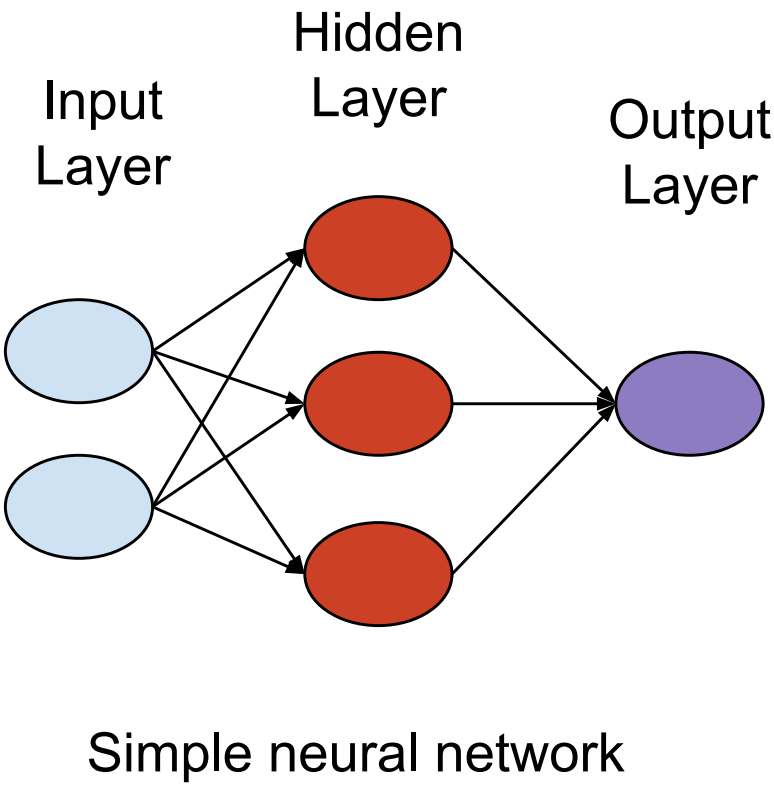


Figure 2.2: An example of a basic artificial neural network. The input has two attributes, go through a hidden layer with three nodes and come to the output layer.

Each neuron has three main parameters: the input data, the output data and the activation function. The neurons of the input layer receive information. The data from the previous layers is passed on to the layers that follow. The activation function transforms input data to output data. Each of these connections between neurons is assigned a weight w that represents the importance of the connection that corresponds to the final result. Figure 2.3 below shows the simple architecture of neural network with input, output and activation.

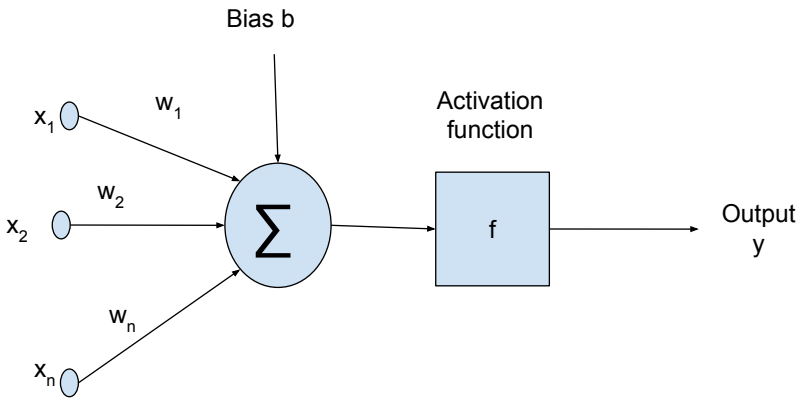


Figure 2.3: The simple architecture of a neural network. Neural network tries to transform from a screenshot image into the next action need to perform for playing Mario game automatically.

After calculating the output with inputs and initial weights, an error function known as the Loss Function is used to define how far the outcome is from the actual value. Then we update each network weight in such a way that the Loss Function is minimised.

Among various deep learning models, the most famous architecture is Convolutional Neural Network (CNN), which is a type of artificial neural network. Since surprising results were presented in recognition competitions [34], it has been the dominant technology for computer vision problems.

2.2.2 Deep Learning with Reinforcement Learning

In conventional RL algorithms, most of the time, are only considered MDP with discrete states and actions space. However, in many real-world applications, the state space is not really discrete, but rather a continuous domain [17]. Therefore, to be usable in the continuous state space, neural networks are also considered as function approximators that are especially useful in RL when the state space or action space is too broad to fully comprehend [28]. Neural nets can discover ways to map states to values in this way. When the problem state space is too big or considered as continuous space, we cannot use a lookup table to store and update all possible states and actions. In that case, one alternative is to train a neural network with samples from the state and the environment and expect them to predict the value of the next action as our target in RL.

We will take an example about a deep learning agent which is shown in Figure 2.4. A CNN will rate the acts that can be performed in a given state given an image of the environment. It may predict that running right will return 1 point, jumping will return 2, and running left will return zero.

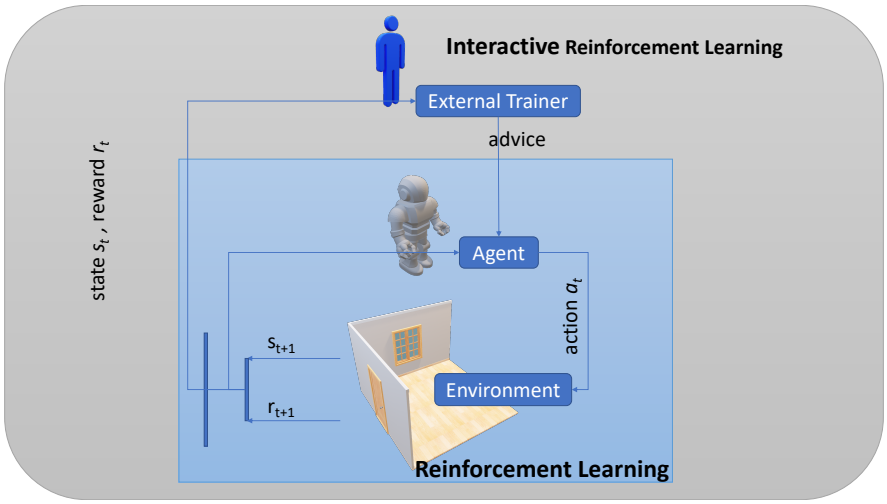


Figure 2.4: An example of what a policy deep learning agent map a state to the best action.

More formally, we use a neural network to approximate the optimal action-value function which is the maximum sum of rewards in Equation (2.3)

$$Q^*(s_t, a_t) = \max_{\pi}(\mathbb{E}[\sum_{0} \gamma^k r_{t+k} | s_t = s, a_t = a, \pi]) \tag{2.3}$$

A vast variety of recent advanced robot applications have been accomplished using deep reinforcement learning to teach agent complex activities including cube play [2], ambidextrous robot gripping [40], categorised objects [29] and cleaning table task [12]. For instance, Cruz et al. [12] used an associative neural architecture to learn the available action possibilities of agents with the objects in the current context. Levine et al. [26] proposed a learning-based approach to hand-eye coordination for robotic grasping from monocular images using a large CNN to learn the way to grasp objects.

2.3 Interactive Reinforcement Learning

Reinforcement Learning, like other machine learning approaches, has a problem learning in large state spaces. The agent's training time increases significantly and making finding a solution become impractical [8]. Interactive Reinforcement Learning (IRL) is an extension of RL which motivates to scale up the agent problems and improve the training speed by using external information. The human knowledge will be transferred to agent by guidance while retaining the advantages of RL. In Interactive Reinforcement Learning (IRL), there is an external trainer involved in the agent's learning process [11]. Figure 2.5 depicts the IRL solution, which includes an advisor who observes the learning process and offers guidance on the way to improve decision-making [22]. The advisor can be an expert human or an artificial

agent.

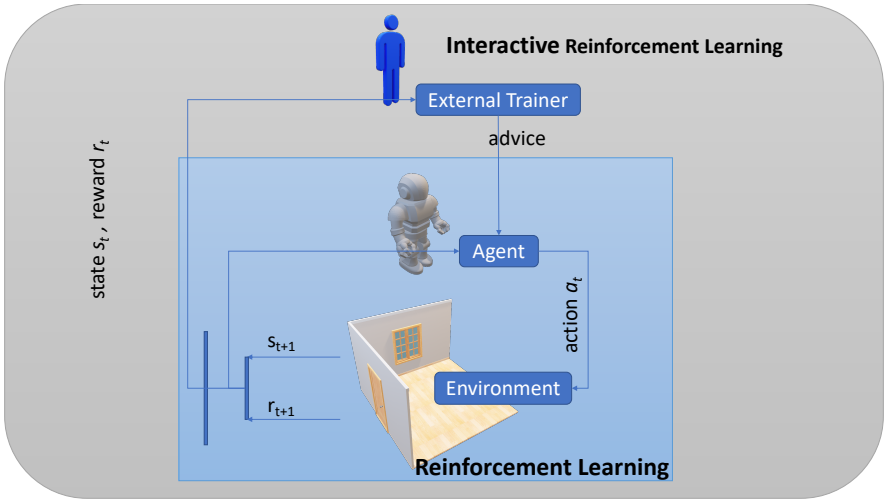


Figure 2.5: The involvement of external trainer to the traditional reinforcement learning process. At state s_t, the agent performs action a_t and obtaining reward r_{t+1} and reaching the next state s_{t+1} in conventional RL. In IRL model, the involvement of an external trainer gives the agent more options to chose which action to perform next in the following iteration.

Adaptive agent behaviour is needed in domestic environments. IRL enables a parent-like tutor to facilitate learning by providing useful guidance in some particular situation, allowing the apprenticeship process to be accelerated. In contrast to an agent exploring completely autonomously, this makes for a smaller search space and hence quicker learning of the mission [14].

When operating alone, the next step is chosen by selecting the better known action at the current time, defined by the highest state-action pair. While IRL accelerates the learning process by incorporating additional guidance into the apprenticeship loop. Using IRL, a trainer with prior experience of the target goal is required [39].

There is a difference between the two main methods dedicated to feedback learning: reward shaping and policy shaping. While in the reward shaping, external trainers can assess the quality of the actions performed by the RL agent, as good or bad [39]. Using policy shaping, the actions proposed by the RL agent can be replaced by more appropriate actions selected by the external trainer before implementation [9].

An open problem that can significantly affect the agent's performance is inaccurate advice from the trainer [11], since lack of accuracy and repetitive mistakes will result in a longer training time. Human advice, on the other hand, is not 100% correct [5]. When an advisor gives so much guidance, the agent will have limited experience in exploration because the trainer makes almost all of the decisions [38]. To address the problem, a prior study [6] applied to the agent a strategy of discarding or refusing advice after an amount of time, endowing an agent with the ability to work with potentially incorrect information.

Chapter 3

Research Design

With the goal of solving the research questions and assessing the recommended techniques, we describe the research methodology in detail as follows.

3.1 Problem Statement

Deep Interactive Reinforcement Learning (DeepIRL) includes Deep Reinforcement Learning (DeepRL) and interactive feedback from an external trainer or expert giving advice to help learners choosing actions to speed up the learning process. However, current research has been limited to interactions that offer actionable advice to only the current state of the agent. Additionally, the information is discarded by the agent after a single use that causes a duplicate process at the same state for a revisit. The aim of the project is to retains and reuses the processed information in continuous environment. It not only helps trainers to give more general advice relevant to similar states instead of only the current state but also allows the agent to speed up the learning process.

3.2 Review and Analysis Existing Approaches

A thorough analysis of the theoretical context and recent studies conducted relating to topics of Reinforcement Learning, Deep Learning, Deep Reinforcement Learning, Interactive Reinforcement Learning and persistence feedback. This work is done in chapter 2

3.3 Environment setup

3.3.1 Cart pole gym environment

The deep reinforcement learning environment is implemented using the well-known library of AI gym environments [7]. First, we build the Cart Pole environment. In this environment, there is a pole that is attached to the cart. The carriage can move by applying force to the left or right. The purpose of this problem is to prolong the time while avoiding the pole falling down. The terminal condition is that the pole deviates more than 12 degrees from the vertical or the wagon moves 2.4 units from the center. The cart pole MDP is defined as follow:

- State: The state vector has a continuous representation with four attribute which is as follows
  - The cart position on the track with a range from -2.4 to 2.4
  - The cart velocity along the track with a range from -inf to inf
  - The pole angle with a range of -24 degrees to 24 degrees
  - The pole velocity at the tip with a range of -inf to inf
- Action: The cart can perform two actions on the track: go to left or right.
- Reward function: Reward is 1 for every step taken, including the termination step

Figure 3.1 below denotes a graphic of the Cart Pole in the AI-gym environment.

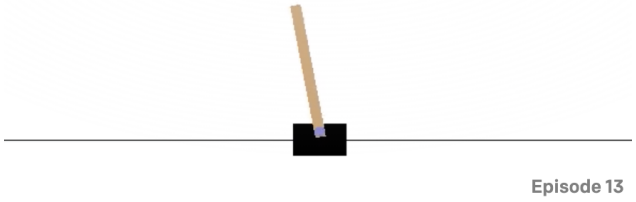


Figure 3.1: A graphical representation of the Cart Pole environment. The goal is to keep the pole balanced while applying forces to the carriage. The terminal condition is that the pole deviates more than 12 degrees from the vertical or the wagon moves 2.4 units from the center.

This is a continuous environment, we build a simple neural network consisting of two dense layers to learn optimal policy. Each layer has 24 fully connected nodes. The input layer has four nodes that represent four attributes in state space. The output layer has two nodes corresponding to two actions: go left and go right. The network architecture is described in Figure 3.2.

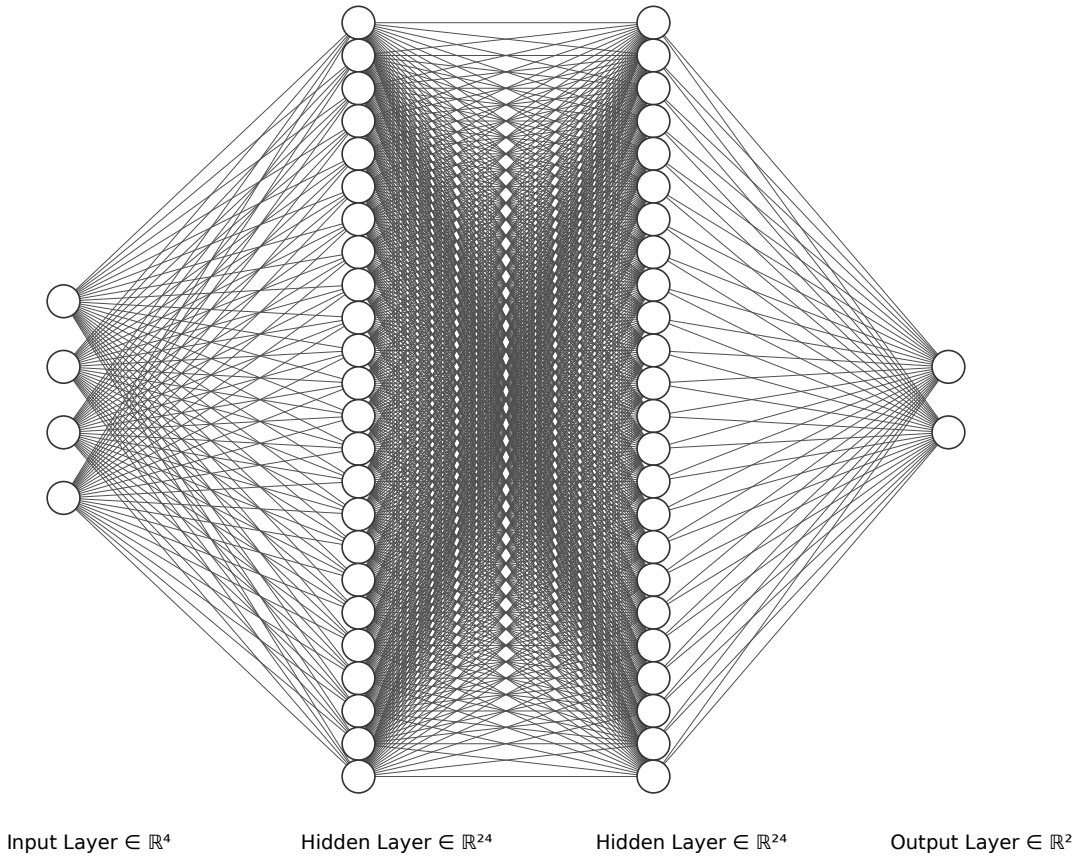


Figure 3.2: A neuron architecture architecture with input 4 node, two dense full connected layers 24 nodes, and 2 nodes at the output

3.3.2 Domestic robot environment

Additionally, we also build an environment for domestic robots using Webots. In this environment, the goal is to train the robot to go from the initial position to the target position. Figure 3.3 denotes a graphic of our experimental environment in Webots.

The robot is equipped with distance sensors on its left and right eyes. The robot is completely unaware of its current position in the environment. The robot can only choose one of three actions: go straight at 3m/s, turn left, or turn right. At each step, the robot will be deducted 0.1 points if it uses the action of turning left or right, no points will be deducted if it chooses

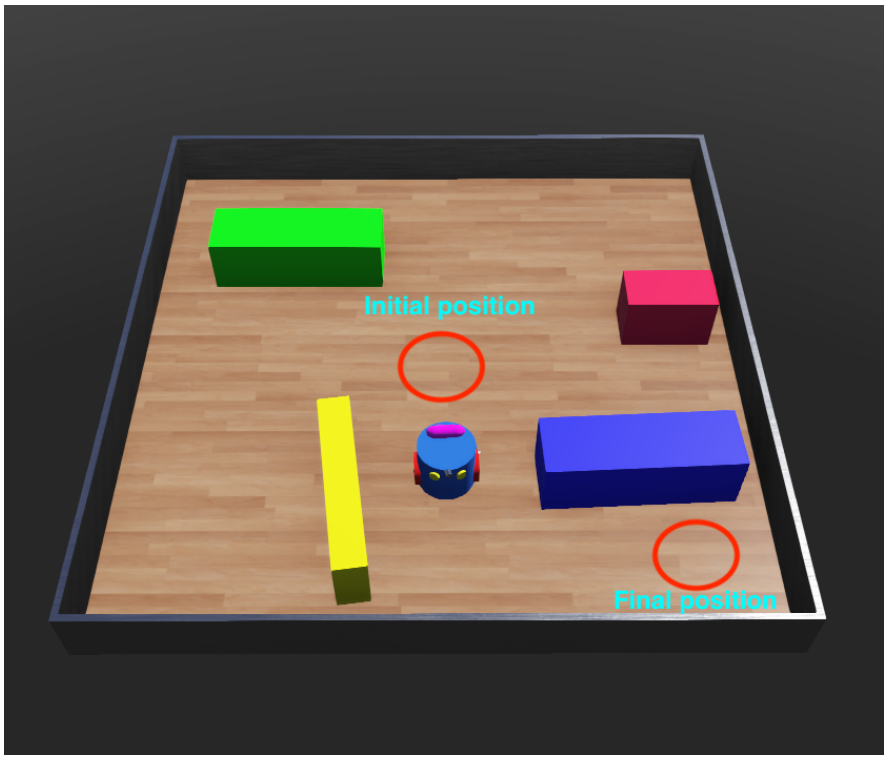


Figure 3.3: A example of Webots environment with initial position and final position. The robot has goals to go from initial position to final position while avoiding obstacles. The robot will be returned to its initial position after any collision.

to go straight. This is to optimise the robot's straight movement and avoid the robot running in circles by turning left or right continuously. The robot is equipped with a few touch sensors next to it, to detect the collision with the environment. The robot will be returned to its initial position and receive 100 penalty points every time it collides on the way. The robot does not know where the touch sensor is located relative to itself, the only information it receives is whether it is a collision with obstacles or not. When the robot goes to the finish position located in the lower right corner of the environment, the robot is considered to complete the task and be rewarded with 1000 points.

To decide on the next action the robot will choose, the robot's supervisor will use the image taken from the top of the environment to enter the Convolutional Neural Network (CNN) system to decide. The CNN system built in this environment will be a system whose input is 64x64 image RGB channels. This architecture is inspired by similar networks used in other DeepRL works [29, 23]. In more detail, we use 4 kernels with size 8x8. The second layer is 8 kernels with size 4x4, and the last layer convolution is 16 kernels with size 2x2. Following each convolution network layer is a 2x2 max-pooling layer. Finally, there is a flatten and dense layer with 256 neurons fully connected with the output layer. The network architecture is described in Figure 3.4.

The environment MDP is defined as follow:

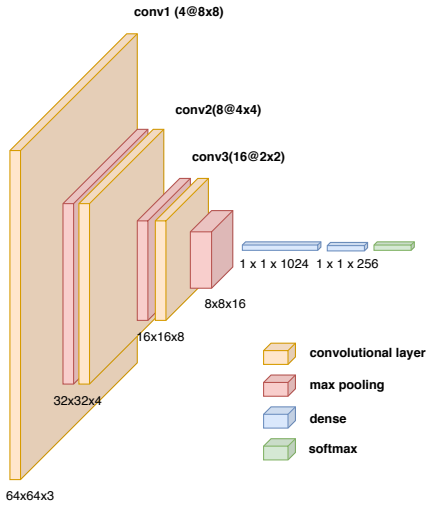


Figure 3.4: CNN architecture with 64x64 RGB image as input, group of three convolution layers, three max-pooling layers, two dense full connected layers, and a softmax function at the output

- State: RGB image size 64x64 taken from the top of the environment.
- Action: Three actions: go straight at 3m/s, turn left, or turn right
- Reward function: Turn left, right: -0.1; Go straight: 0; Collision: -100; Reach to final position: 1000

3.4 Interactive feedback

While the interactive agent's human-related approach to learning is one of its greatest strengths, it may also be its greatest weakness [1, 10]. Advice with good accuracy given in the proper time will help the agent a lot in speeding up the speed of finding the optimal solution. However, in the case when the agent only gives advice with low accuracy and in high frequency, that not only does not help the agent but also brings it to a dead road and is much more time-consuming than no interaction situation. Furthermore, human experiments are costly, time-consuming, have problems of repeatability, and can be difficult to recruit volunteers. Therefore, during the early stages of the agent, we suggested that simulating human interactions would be much more convenient.

Each use case of the simulated user will have different advice's accuracy and frequency. Accuracy is a measure of the precision of advice provided by an advisor. When the advisor's precision is high, the action would be proposed precisely as the advisor's knowledge of the environment. On the contrary, the advisor would propose not optimal action based on how it

knows about the environment. Frequency is the availability of the interaction of the advisor at the given time step. The higher frequency, the advisor has more rate for giving advice to the agent. Accuracy is a measure of the precision of advice provided by an advisor. When the advisor's accuracy is high, the action would be proposed precisely as the advisor's knowledge. On the contrary, the action proposed is different from the advisor's knowledge. Accuracy and frequency of three kinds of agents are used with value described in Table 3.1. Optimistic simulated agents have 100% accurate advice and always provide advice on every time step. Realistic simulated agents use accuracy and frequency value from results in a human trial [5, 6]. The pessimistic value of frequency is 0%, however, it works the same as in the case without interactive feedback. Therefore, we use half of the realistic value for the case with the least interaction of the advisor. The accuracy and frequency value advice is beyond the scope of this study.

Agent	**Frequency**	**Accuracy**
Pessimistic Advisor	23.658%	47.435%
Realistic Advisor	47.316%	94.87%
Optimistic Advisor	100%	100%

Table 3.1: The three simulated users designed for the experiments. These users will are not intended to be compared against each other, rather than comparing with persistent counterpart

Experiments are performed for each case and the indicator of how accumulated reward can achieve the optimal policy will be recorded to compare the result between many approaches. The more reward the agent takes, the better result of the method is. The concept of persistent feedback and the detailed description that needs to be used for experiments will be described in the following chapter, Chapter 4.

The experiment is used to test the effectiveness of applying the persistent model to the DeepIRL algorithm. Therefore, the results of the algorithm running due to the three types of agents created in Table 1 are not intended to compare against each other, rather than compare with their non-persistent counterparts. The test results after running, the solution will be displayed in the form of a graph. Based on that to determine whether persistent agents learn faster, have fewer interactions, or are worse off than non-persistent. Then, conclusions can be drawn about the effectiveness of the new method.

3.5 Project Risk

Risk Potential	Detail	Mitigation
Identify the scope of thesis	Sometimes, the scope of the thesis is significantly extended beyond its initial limits during redaction. This risk depends on the student experience	The student should follow the general plan of the thesis has been agreed upon with the supervisor.
Misallocation of time resources	Students do not allocate their time appropriately followed the plan	Students should often talk to their supervisors about what they are interested in and what they are doing to have timely appropriate intervention from their supervisors.
Physical and Psychological	Including physical discomfort, pain, injury, illness or disease or anxiety, depression.	Students should pay attention to stay healthy in order to achieve good academic result
Tool using adaptability	Not familiar with using tools of robot simulators	Students take time for understanding how to use it in basically, need to schedule carefully

3.6 Ethics

This is a robot-related project, the information the robot stores is data created by itself about its actions and surrounding environments, so do not minimise it ethically. What we need to be concerned about is data from actually, if there are people involved, there much more to take care about interacting with robots. We need to ensure the data we collect and store do not contain any sensitive information which not reveal any information to identify people, or potential harm to humans. Moreover, it is necessary to ensure that the data storage place is only internally, not public to avoid unwanted harm. In the other hand, in the future regarding the application of this research and the booming of research about domestic robots, there will be more new robot laws coming, and moreover, the robot will be connected to the smart home ecosystem. Robots can be share with the system to get more information about user behaviours, which can be harmful if exploiting data.

Chapter 4

Artefact Development Approach

This chapter continues the discussion of the approach described in the research design section. While including the rationale and the specific research procedure that supported the study, this chapter also includes a description of the methods and procedures used to develop a minimum-viable artefact to find the answer to the research question. In this section, we give more details about the proposed Broad-persistent Advising (BPA) approach that includes a generalisation model along with a persistent approach. These details are described next.

4.1 Persistent Advice

4.1.1 Probabilistic Policy Reuse

Probabilistic Policy Reuse (PPR) is a learning strategy used in reinforcement learning to enhance training speed with guidance from previously learned similar policies [18]. At every step, the learning agent needs to balance between three choices: past policy, exploitation of the new policy being learned, or random unexplored actions. Reusing a previous policy necessitates integrating past policy knowledge into the current learning process. We call Policy Library to the set of past policies as follow $L = \{\pi_1, \pi_2, ..., \pi_n\}$. Each policy $\pi_i \in L$ solves a task at a certain state s_i with an action a_i. The goal of the π—PPR technique is to balance random exploration, exploitation of previous policies, and exploitation of the new

policy that is presently being learnt, as represented by Equation 4.1

$$a_t = \begin{cases} \pi_{past}(s_t) & \text{with probability } \psi \\ \epsilon - greedy(\pi_{new}) & \text{with probability 1 - } \psi \end{cases} \tag{4.1}$$

The π—PPR strategy uses the past policy with a probability of ψ. However, with a probability of 1 - ψ, it exploits the new policy. Obviously, random exploration is always required, so when exploiting the new policy, it follows an ϵ-greedy strategy. As aforementioned, there is an issue relating to inaccurate advice. After a certain amount of time, a mechanism for discarding or ignoring advice is needed. Therefore, the value of ψ decay in each trial to deal with this problem. The flow PPR used in this work is present in Algorithm 1

Algorithm 1 PPR in RL algorithm

for all (episodes) **do**
    Initialise environment with s_t
    for all (step) **do**
        With a probability of ψ, a = $\pi_{old}(s)$
        With a probability of $1 - \psi$, a = ϵ-greedy $\pi_{new}(s)$
        Update $\pi_{new}(s)$ to policy library
        Update ψ with new value decayed
        Perform action a
        Observe next state s'
    end for
end for

4.1.2 Persistent Advice for IRL

A recent study [6] suggests a permanent agent that records each interaction and the circumstances around particular states. The actions are re-picked when the conditions are met again in the future. As a consequence, the recommendations from the advisor are used more effectively, and the agent's performance improves. Furthermore, as the training step is no need to provide advice for each repeated state, less interaction with the advisor is required. However, in this experiment, we limit the research to keeping the same number of interactions to the trainer to investigate the effectiveness of our approach in continuous domain.

Figure 4.1 denotes an example of IRL using PPR. The advising user has the opportunity to engage with the agent at each time point. When there is an interaction, the model is updated. At the time advice is firstly recommended, it is assumed that the agent will carry it out the suggested action, regardless of the setting of PPR. PPR is used the time step when

the agent did not receive advice from the trainer, which flow is denoted by red arrows. First, the agent's policy is examined to see whether any advice is applicable to the existing state. If the current policy suggests an action, the action is taken with the determined by the PPR selection policy.

PPR is used where an agent chooses an action in a time step where the user has not recommended a prior action, which is denoted by red arrows. First, the agent's policy is examined to see whether any advice is applicable to the existing state. If the current policy suggests a action, the action is taken with the determined by the PPR selection policy.

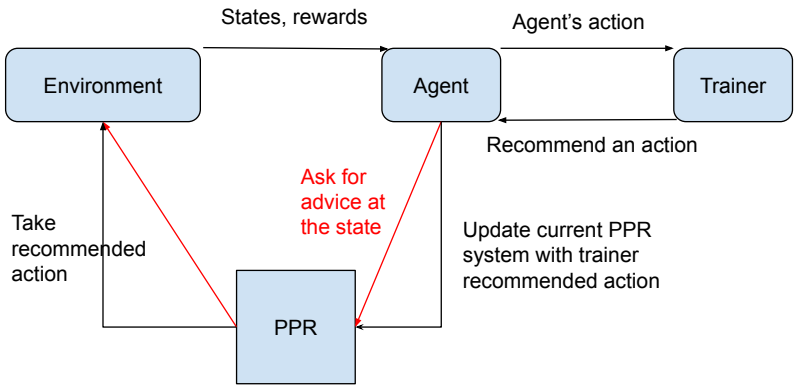


Figure 4.1: Process flow of an interactive reinforcement learning agent using PPR system. After receiving advice from the trainer, the agent will store the action into the PPR model. In the iteration not receiving advice from the trainer, the agent will check and reuse the old advice (red arrows) from the past.

4.2 Broad Advice

4.2.1 Broad Advice for Large State Space

To use PPR, we need a system to store the used pairs of state-action. When the agent arrives at a certain state at a time step, agents using PPR need to check with the system if this state has been suggested by the trainer in the past. If there is advice in the memory of the model, the agent can use the option to reuse the action. However, there is a problem when using PPR in infinite domains. We cannot build a system that stores state-action pairs with infinite state values. In addition, when the amount of state becomes too large in space, which is equivalent to infinity, the possibility that agents revisit exactly the same state will be very small. Therefore, building this model will become cumbersome and inefficient in large spaces.

BPA includes a model for clustering states and then building a system for cluster-action pairs

instead of traditional state and action pairs. The proposed model is shown in Figure 4.2. When the agent receives current state information from the environment and it does not receive any advice from the trainer, the agent will use PPR by injecting the state into the generalisation model and defining its cluster. Then proceed to consider whether any advice pertains to the current cluster. If there is an action recommend in the past, the agent can reuse it with the PPR selection probability, or use default action as ϵ-greedy.

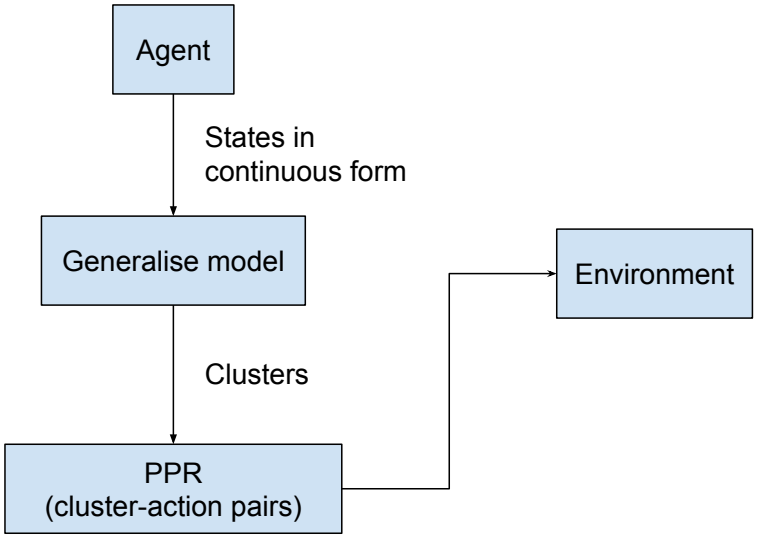


Figure 4.2: Broad advice transform continuous states into finite clusters. Hence, the state-action pair becomes cluster-action used in the PPR model.

The generalisation model we use in this paper is the k-means algorithm. k-means is one of the most popular clustering methods [33]. k-means is simple to implement, and its complexity scales well with a higher number of data. However, the user must decide on the number of clusters beforehand [27]. We used the elbow technique which is described in next section.

4.2.2 Elbow Method

The elbow method is a method that looks at the percentage of variance explained to specify the number of clusters [**?**]. It is the visual graphic approach that was generated from the Sum Square Error (SSE) computation. This technique is based on the idea that the number of clusters should be chosen so that adding another cluster does not cause significantly improved modeling. The early clusters will provide a lot of information, but at some point, the marginal gain will drop drastically, giving the graph an angle. At this angle, the correct k-number of clusters is determined, thus called ”elbow criteria”. Detail about algorithm is describe in Algorithm 2

Algorithm 2 Elbow method

1: Initialise environment with $k = 1$
2: **repeat**
3: Increase the value of k
4: Measure the cost of quality solution
5: At some points the cost drop dramatically. After that, the cost don't drop more when you increase k.
6: That is the value k we need
7: **until** (k is found)

The idea starts with $K = 2$, calculating the clusters and the cost that comes with the training: SSE. The value of SSE goes down rapidly with K increasing from 2 to 3 and 4, and after that (for example), it goes down very slowly. It looks like maybe 4 is the right number of clusters because that is the elbow of this curve. The rationale is that the number of clusters is increased but the new cluster is very near some of the existing ones. Therefore the new cluster does not cause effective improvement for the algorithm.

4.2.3 Implementation

Next, we demonstrate the use of and broad advice and persistent advice using Probabilistic Policy Reuse (PPR). The flow of using PPR is depicted as shown in Figure 4.3. Initially, the agent reuses the action using PPR with a certain chance if the current state has been recommended by the trainer in the past. At the current work, we use chance value at 80% which is also used in previous research [6]. This probability decreases by 5% for each step. With the remaining 20%, the greedy action policy is selected.

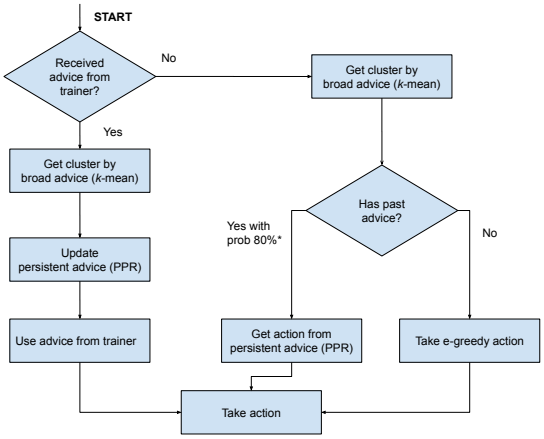


Figure 4.3: Flow of using broad-persistent advising. The agent will reuse previously obtained advice with 80% chance (decays over time) and perform its exploration policy for the remaining change (20%).

Algorithm 3 shows the process flow for selecting an action using BPA approach to assist a learning agent. In the algorithm, c_t represents the cluster which is categorised by broad advice by using k-means with the input state s_t, the pair (c_t, a_t) is memorised in policy reuse model.

Algorithm 3 Interactive reinforcement learning with a BPA

```
1: Built k-means model with states from trainer
2: Initialise environment selecting s_t
3: for all (episodes) do
4:     repeat
5:         if (have feedback) then
6:             Get recommend action a_t
7:             Get cluster c_t by using k-means
8:             Add pair (c_t, a_t) to ppr
9:         else
10:            if (rand(0,1) < ε) then
11:                Get c_t by using k-means
12:                if (c_t is available in ppr) then
13:                    Get a_t is reuse action from ppr
14:                else
15:                    Random action c_t from environment
16:                end if
17:            else
18:                Choose action a_t using π
19:            end if
20:        end if
21:        Perform action a_t
22:        Observe next state s_{t+1}
23:    until (s is terminal)
24:    Update policy π
25: end for
```

The work will be tested three learning agents have been designed: baseline RL, non-persistent RL, and persistent RL. They are described as below:

- Baseline Reinforcement Learning: The model will be trained without using any interactive feedback or evaluation from trainer. It is used as a benchmark
- Non-persistent Reinforcement Learning: The agent is assisted by multiple type of users with in mentioned before in Table 3.1. After taking recommendation from trainer and

execute the action, the agent will discard the advice. When the agent come to the similar state again in the future, it cannot recall the previous recommendation and performs an ϵ-greedy action instead.

- Persistent Reinforcement Learning: This agent is supported by a trainer and PPR system. The trainer can suggest an action in each time step for the agent to take. If recommended, the learning agent will perform on that time step and retain the recommendation for reusing when it visits the similar state in the future. When an agent accesses similar state it has previously suggested, it will perform that action with the probability determined by the PPR action selection rate.

Chapter 5

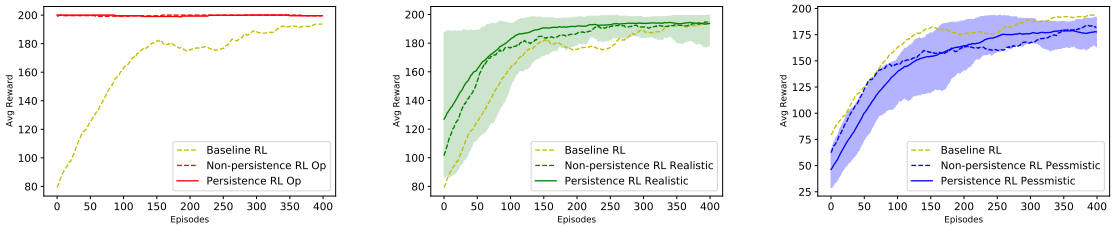
Empirical Evaluation

5.1 Cart pole domain

In this section, we proceed to display the results three types of agents proposed above, including: baseline RL for bench marking, non-persistent RL, persistent RL. For agents of the type non-persistent RL and persistent RL, we conduct tests on different frequencies and accuracy of feedback, called optimistic user, realistic user and pessimistic user. The method for all the agents are tested with the same hyper-parameters as follows: initial value of $\epsilon = 1$, ϵ decay rate of 0.99, learning rate $\alpha = 0.01$, and discount factor $\gamma = 0.99$ during 500 episodes. To better display, we computed the average value of the last 100 rewards instead of the current episode reward. We inspired the idea done on this article with the same result from cart pole environment [24].

The results obtained are shown in Figure 5.1. Optimistic, realistic, and pessimistic agents are run five times and are represented by red, green, and blue lines respectively. The shaded area indicates the standard deviation of the agent's reward after multiple training. Overall, all interactive agents outperformed the autonomous one (baseline RL in yellow), except the pessimistic agents. Agents which receive advice from the instructor make fewer mistakes, especially in the early stages of the learning process, and can learn the task in fewer episodes. However, in this work, we want to compare pairs of non-persistent and persistent agents with the same style delivered advice to verify if the BPA approach implementation is indeed effective.

The agents assisted by optimistic achieved the maximum score of DeepIRL algorithms at very early after a few episodes. Because the trainer always makes decisions for the agent (100%),



(a) Optimistic agents (b) Realistic agents (c) Pessimistic agents

Figure 5.1: The comparison for persistent agents, non-persistent agents, and baseline each kind of agent in deep reinforcement learning built with cart pole environment. The shaded area indicates the deviation between the minimum and maximum values of the agent value after multiple training.

and this decision is absolutely correct (100%). In this experiment, the agents did not even have any chance to make their own decisions or use PPR, the trainer made all decisions.

On the contrary, agents supported by pessimistic users have different results, but in fact, neither of them can solve the problem. On many runs in both non-persistent and persistent cases, the agent failed to achieve convergence. Both cases are considered worse than the baseline. This can be explained because the accuracy of advice for pessimistic agents is only 23.658%

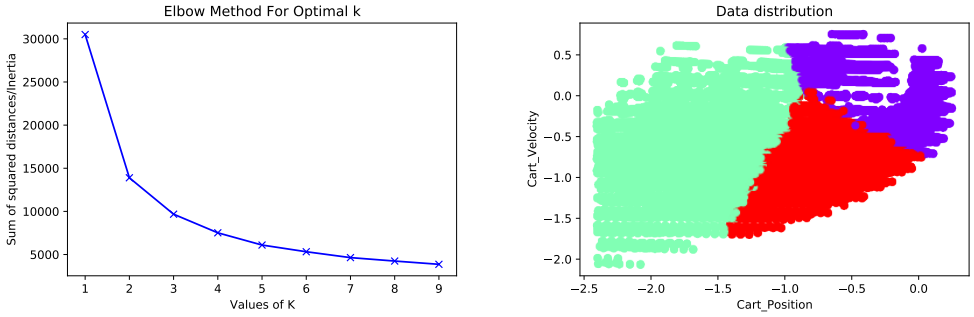
On the graph of the agents being helped by realistic trainers, we can see that using BPA produces slightly better results than the non-using counterpart. Persistent agents not only have a better initial reward but also can achieve convergence results 100 episodes earlier than non-persistent agents. This difference in learning rates is due to the fact that the agent retains and reuses advice. In this experiment, the realistic agent has a 47.3% chance to interact with the trainer and the agent will withhold or not withhold the advice from the trainer depending on whether it is a persistent agent or non-persistent agent. However, the persistent agent will retain and reuse the advice with an 80% probability (decreasing over time) for any state in which it has received the advice in the past. As long as the stored advice is accurate enough, the persistent agent will learn faster because they use the advice more often. In this experiment, we focused on implementing persistent advice in a continuous environment. The ratio of the number of interactions with our trainers remains the same: for example 47.316% with the realistic agent. When receiving advice from the trainer, the agent will always prioritise executing this recommended action. Therefore, the number of interactions using the BPA method is equivalent to not using it. Table 5.1 shows the average number and percentage of interactions that occurred for each agent. Both non-persistent and persistent agents use the interaction rate according to Table 3.1. We can see that the number of interactions of every pair of optimistic, realistic and pessimistic agents are similar in the experiment.

Figure 5.2 shows a graph using the elbow method to specify the number of clusters as a

Agent	Interaction	
	Non-persistent	**Persistent**
Optimistic Advisor	99796 (100%)	99846 (100%)
Realistic Advisor	40976 (47.15%)	41832 (47.1%)
Pessimistic Advisor	18034 (23.62%)	16685 (23.76%)

Table 5.1: The average number of interactions in experiment for each kind of agent, and the percent compare with total steps taken

parameter of k-means, using the data of 50000 states, the same number as the experiment of cart pole, in the actual running environment. The elbow method shows the best value for using k at the value 3. Figure 5.2b displays the data distribution in cart position and cart velocity attributes axes at the value k = 3.



(a) Elbow method (b) Distribution

Figure 5.2: Total of squared distance for value k from 1-9 and distribution for 50000 states with value k = 3 at cart position and cart velocity attributes

5.2 Webots domain

In this scenario, we focus only to examine the results for the realistic agent, because this can be transferred to the real-world scenarios in a more rational manner. The method is tested with the following hyper-parameters: initial value of $\epsilon = 1$, ϵ decay rate of 0.99, learning rate $\alpha = 0.01$, and discount factor $\gamma = 0.99$ during 500 episodes. We use the average value of the last 100 rewards instead of the current reward only.

The results obtained are shown in Figure 5.3. Non-persistent RL agent is shown by a green dashed line while persistent RL agent is shown by the green solid line. Baseline RL is drawn with yellow lines used for bench marking. Similar to the cart pole environment, both agents supported by the trainer, regardless of whether or not they used PPR, obtain better results than baseline RL. Then, the persistent agent achieves convergence results slightly earlier than its non-persistent counterpart. The trainer's accuracy and frequency feedback are used the

same as in the cart pole environment, so the results are reflected for use in the domestic robot environment as well and, not just the ideal hypothetical environment like cart pole in AI gym.

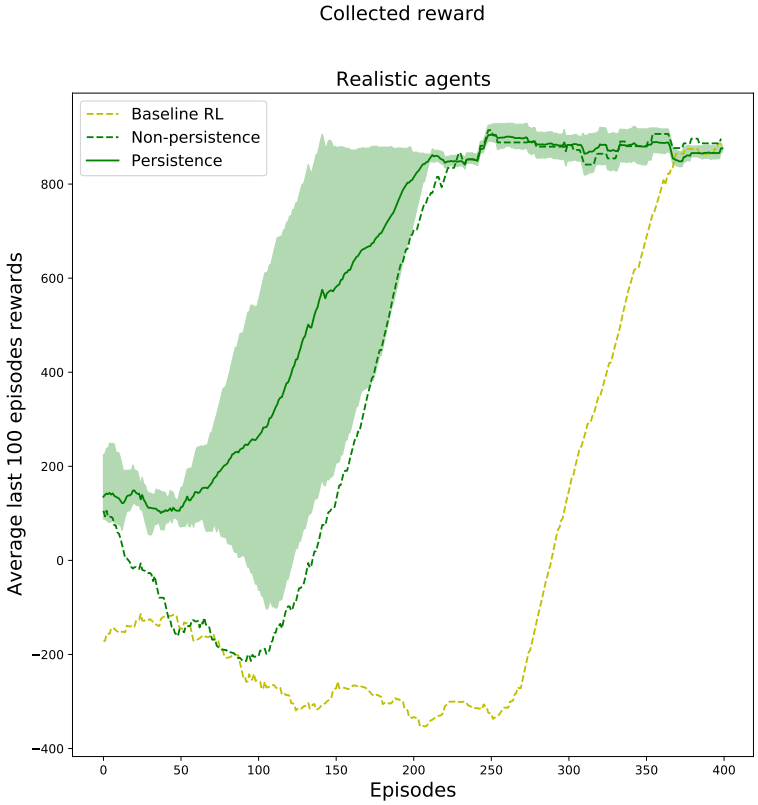


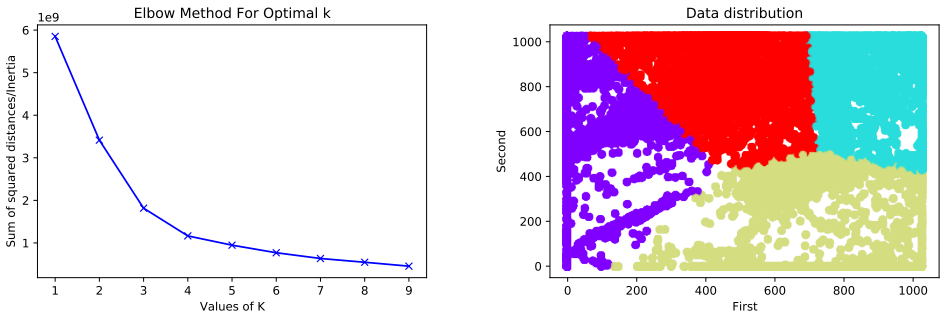
Figure 5.3: Result for deep reinforcement learning with autonomous agent, non-persistent agent and persistent agent built with Webots domestic robot environment.

Table 5.2 shows the average number and percentage of interactions that occurred for each agent. We can see that the number of interactions is similar in the experiment.

Agent	**Interaction**	
	Non-persistent	**Persistent**
Realistic Advisor	9077(47.64%)	8241(47.18%)

Table 5.2: The average number of interactions in experiment and the percent compare with total steps taken for realistic agent in Webots environment

Figure 5.4a shows a graph using the elbow method to specify the number of clusters as a parameter of k-means, using the data of 50000 states in the actual running environment. The elbow method shows the best value for using k at the value 4. Figure 5.4b displays the data distribution in two axes of distance sensor value at the value k = 3.



(a) Elbow method (b) Distribution

Figure 5.4: Total of squared distance for value k from 1-9 and distribution for 50000 states with value k = 3 at two value attributes of distance sensor.

Chapter 6

Conclusion

6.1 Summary of the Thesis

In this work, we experimented with incorporating persistent feedback into the DeepIRL model. To do this, we have researched and presented the underlying theories as well as recent surrounding studies on Reinforcement Learning, Deep Learning, Deep Reinforcement Learning, Interactive Reinforcement Learning, and persistence feedback as well.

For the contribution, we proposed BPA, a broad-persistent Advising approach to implement the use of PPR and generalised advice in continuous-state environments. We then set up an experiment on cart poles and webots to investigate the impact that the BPA approach had on the measured performance.

In addition, to simulate the environment of interaction feedback, we also present studies and includes optimistic agents, realistic agent, and pessimistic agents into our experiment. Finally, we carried out with three different agents set-ups were : (i) baseline RL, (ii) non-persistent RL, and (iii) persistent RL.

6.2 Discussion

The research presented in this thesis aimed at addressing the following question: *Is possible to extend the persistent IRL approach to continuous representation?* These results prove that we can apply the persistent IRL approach into continuous domains by using broad advice.

Broad advice is a generalised model to transform a variety of pairs state-action into a limited number of pairs cluster-action. This process is based on assumption that the advice from trainers in similar states is the same action.

The results from the experiment also answer for the question: *To what extend can persistent feedback speed up the learning process in continuous RL scenarios?* Experiments show the effectiveness of persistent feedback to speed up the learning process in continuous RL scenarios. Overall, results obtained show that the BPA approach with k-means as a generalise model and PPR as a model of persistence performed slightly faster when the advice is withheld. The more accuracy for the advice and the longer time to retain it increase learning speed significantly. Our research shows that the first step for applying persistent approach in a continuous state-space environment is feasible and effective. In addition, k-means used as broad advice inherits advantages based on its characteristic: runs fast, and scales very well over a large state space. Therefore, it is very suitable for the model to run in a real-world environment.

In general, the use of K-means in this study is considered the simplest example for implementing a generalised advice model. The k number of clusters determined by the elbows method based on 50,000 states on the trainers' run are quite small, only 3 for the cart pole environment and 4 for the webots environment. Therefore, the accuracy of the generalisation model is quite low, many different states have the same action that is remembered in the persistent model. Therefore we only allow agents to remember advice within a few steps and keep the number of interactions from the trainer to ensure that the advice from the trainer to the agent is still accurate. If the agent tries to reuse incorrect advice and store it for a long time that effectively affects the accuracy of BPA models as well as the speed and convergence of the RL algorithm. As future work, we are planning to further investigate the number of clusters for the k-means algorithm or a more extensive study of the generalisation model to obtain a better model that can give the best performance in BPA approach.

Additionally, we suggest reducing the number of interactions with the trainer by reusing the action in the persistent model more often. When the agent reaches a new state that is already in memory, the agent reuses the recommended action immediately without interacting with the trainer. The number of interactions between the agent and the trainer will be reduced to reduce the pressure when used in a real environment, where the signal transmission time between the trainer and the agent is significant. However, this should only be done when we have a good enough generalisation model.

6.3 Future Work

This section discusses possible future directions for the research presented in this thesis. Overall, this thesis is the first step in applying the use of persistent advice to continuous domains. There are still many things that need to be dug into in order for this research approach to become more effective or transferring and test the proposed approach on a real-world scenario with human and robot interaction.

In this experiment, user simulation trainers were used instead of the real user as a limitation of the thesis. The user usage data above is simply accuracy and frequency. A larger and more comprehensive simulated model or an inquiry about the performance of simulation of human behaviors are required. In the further future, both voice recognition and knowledge acquisition will be incorporated to be used in the model of the agent.

A future study might focus on expanding interactive Reinforcement Learning to enable many advice-giving users. A vision of a group of users with different expertise and varying degrees of precision providing support. The more advice the agent receives, the faster the agent will be able to solve the problem. However, this entails a lot of future work to be resolved such as conflicting advice, or choosing the optimal advisor.

Appendix A

List of acronyms

Acronyms

BPA Broad-persistent Advising. i, 3, 4, 19, 21, 24, 26, 27, 31, 32

CNN Convolutional Neural Network. 9, 10, 15, 16

DeepIRL Deep Interactive Reinforcement Learning. i, 2, 3, 12, 17, 26, 31

DeepRL Deep Reinforcement Learning. i, 1, 2, 12, 15

DL Deep Learning. 1, 2

IRL Interactive Reinforcement Learning. i, 2–4, 10, 11, 20, 31

MDP Markov Decision Processes. 7, 9, 13, 15

PPR Probabilistic Policy Reuse. iii, 3, 19–23, 25, 28, 31, 32

RL Reinforcement Learning. i, 1–3, 5–7, 9–11, 24–26, 28, 31, 32

Bibliography

[1] A. Bignold, F. Cruz, R. Dazeley, P. Vamplew, and C. Foale, *An evaluation methodology for interactive reinforcement learning with simulated users*, Biomimetics, 6 (2021), p. 13.

[2] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, et al., *Solving rubik's cube with a robot hand*, arXiv preprint arXiv:1910.07113, (2019).

[3] A. Ayala, C. Henríquez, and F. Cruz, *Reinforcement learning using continuous states and interactive feedback*, ACM International Conference Proceeding Series, (2019).

[4] R. Bellman, *A markovian decision process*, Journal of mathematics and mechanics, 6 (1957), pp. 679–684.

[5] A. Bignold, F. Cruz, R. Dazeley, P. Vamplew, and C. Foale, *Human engagement providing evaluative and informative advice for interactive reinforcement learning*, arXiv preprint arXiv:2009.09575, (2020).

[6] A. Bignold, F. Cruz, R. Dazeley, P. Vamplew, and C. Foale, *Persistent rule-based interactive reinforcement learning*, 2021.

[7] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, *Openai gym*, 2016.

[8] A. R. Cassandra and L. P. Kaelbling, *Learning policies for partially observable environments: Scaling up*, in Machine Learning Proceedings 1995: Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, July 9-12 1995, vol. 362, Morgan Kaufmann, 2016.

[9] T. Cederborg, I. Grover, C. L. Isbell, and A. L. Thomaz, *Policy shaping with human teachers*, in Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.

[10] F. Cruz, S. Magg, Y. Nagai, and S. Wermter, *Improving interactive reinforcement learning: What makes a good teacher?*, Connection Science, 30 (2018), pp. 306–325.

[11] F. Cruz, S. Magg, C. Weber, and S. Wermter, *Training Agents With Interactive Reinforcement Learning and Contextual Affordances*, IEEE Transactions on Cognitive and Developmental Systems, 8 (2016), pp. 271–284.

[12] F. Cruz, G. I. Parisi, and S. Wermter, *Learning contextual affordances with an associative neural architecture.*, in ESANN, 2016.

[13] F. Cruz, G. I. Parisi, and S. Wermter, *Multi-modal feedback for affordance-driven interactive reinforcement learning*, Proceedings of the International Joint Conference on Neural Networks, 2018-July (2018).

[14] F. Cruz, J. Twiefel, S. Magg, C. Weber, and S. Wermter, *Interactive reinforcement learning through speech guidance in a domestic scenario*, in 2015 international joint conference on neural networks (IJCNN), IEEE, 2015, pp. 1–8.

[15] F. Cruz, P. Wuppen, A. Fazrie, C. Weber, and S. Wermter, *Action Selection Methods in a Robotic Reinforcement Learning Scenario*, 2018 IEEE Latin American Conference on Computational Intelligence, LA-CCI 2018, (2019).

[16] E. Dahlin, *Are Robots Stealing Our Jobs?*, Socius: Sociological Research for a Dynamic World, 5 (2019), p. 237802311984624.

[17] G. Dulac-Arnold, D. Mankowitz, and T. Hester, *Challenges of real-world reinforcement learning*, arXiv preprint arXiv:1904.12901, (2019).

[18] F. Fernández and M. Veloso, *Probabilistic policy reuse in a reinforcement learning agent*, in Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems, 2006, pp. 720–727.

[19] V. François-lavet, P. Henderson, R. Islam, M. G. Bellemare, V. François-lavet, J. Pineau, and M. G. Bellemare, *An Introduction to Deep Reinforcement Learning. (arXiv:1811.12560v1 [cs.LG]) http://arxiv.org/abs/1811.12560*, Foundations and trends in machine learning, II (2018), pp. 1–140.

[20] S. Griffith, K. Subramanian, J. Scholz, C. L. Isbell, and A. Thomaz, *Policy shaping: Integrating human feedback with Reinforcement Learning*, Advances in Neural Information Processing Systems, (2013), pp. 1–9.

[21] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, *How to train your robot with deep reinforcement learning: lessons we have learned*, International Journal of Robotics Research, (2021), pp. 1–24.

[22] W. B. Knox and P. Stone, *Interactively shaping agents via human reinforcement: The tamer framework*, in Proceedings of the fifth international conference on Knowledge capture, 2009, pp. 9–16.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, *Imagenet classification with deep convolutional neural networks*, Advances in neural information processing systems, 25 (2012), pp. 1097–1105.

[24] S. Kumar, *Balancing a cartpole system with reinforcement learning–a tutorial*, arXiv preprint arXiv:2006.04938, (2020).

[25] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning*, nature, 521 (2015), pp. 436–444.

[26] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, *Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection*, The International Journal of Robotics Research, 37 (2018), pp. 421–436.

[27] T. S. Madhulatha, *An overview on clustering methods*, arXiv preprint arXiv:1205.1117, (2012).

[28] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, *Playing atari with deep reinforcement learning*, arXiv preprint arXiv:1312.5602, (2013).

[29] I. Moreira, J. Rivas, F. Cruz, R. Dazeley, A. Ayala, and B. Fernandes, *Deep reinforcement learning with interactive feedback in a human-robot environment*, Applied Sciences (Switzerland), 10 (2020).

[30] A. Y. Ng, D. Harada, and S. Russell, *Policy invariance under reward transformations : Theory and application to reward shaping*, Sixteenth International Conference on Machine Learning, 3 (1999), pp. 278–287.

[31] H. Nguyen and H. La, *Review of Deep Reinforcement Learning for Robot Manipulation*, Proceedings - 3rd IEEE International Conference on Robotic Computing, IRC 2019, (2019), pp. 590–595.

[32] Y. Niv, *Reinforcement learning in the brain*, Journal of Mathematical Psychology, 53 (2009), pp. 139–154.

[33] G. Y. Park, H. Kim, H. W. Jeong, and H. Y. Youn, *A novel cluster head selection method based on k-means algorithm for energy efficient wireless sensor network*, in 2013 27th international conference on advanced information networking and applications workshops, IEEE, 2013, pp. 910–915.

[34] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., *Imagenet large scale visual recognition challenge*, International journal of computer vision, 115 (2015), pp. 211–252.

[35] B. F. Skinner, *The behavior of organisms: An experimental analysis*, BF Skinner Foundation, 2019.

[36] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 2018.

[37] T. S. Tadele, T. De Vries, and S. Stramigioli, *The safety of domestic robotics: A survey of various safety-related publications*, IEEE Robotics and Automation Magazine, 21 (2014), pp. 134–142.

[38] M. E. Taylor, N. Carboni, A. Fachantidis, I. Vlahavas, and L. Torrey, *Reinforcement learning agents providing advice in complex video games*, Connection Science, 26 (2014), pp. 45–63.

[39] A. L. Thomaz, G. Hoffman, and C. Breazeal, *Real-time interactive reinforcement learning for robots*, in AAAI 2005 workshop on human comprehensible machine learning, 2005.

[40] C. Wang, Q. Zhang, Q. Tian, S. Li, X. Wang, D. Lane, Y. Petillot, and S. Wang, *Learning mobile manipulation through deep reinforcement learning*, Sensors (Switzerland), 20 (2020), pp. 1–18.