Persistence Rule-based approach in continues domestic robot domain

Submitted as Master Dissertation in SIT723

SUBMISSION DATE
T1-2021

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 causes 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 maximize the obtained reward by choosing the best action in every time step. It recently widely used in robotics to learn about the environment and autonomously acquired behaviors. By combining RL with neural networks approaches, a new method is known as Deep Reinforcement Learning (DeepRL), was born to aim at solving continuous action-state spaces in more complex real-world domains. Moreover, interactive feedback, where an external trainer or expert gives advice to help learner 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 realtime. 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 work, we propose using a persistent rule-based approach on Deep Interactive Reinforcement Learning (DeepIRL). This method will retain the processed information as well as provided knowledge. It not only helps trainers give general advice relevant to similar states instead of only the current state but also allows the agent to speed up the learning process. We expect persistent advice will improve the learning performance of the agent while reducing the need for a number of iteration required for trainers. It will open a new and more potential development direction in the future of domestic robots.

Contents

List of Figures

List of Tables

1 Introduction

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 [13], but also known in the domestic areas. Their presence in domestic environments is still limited, mainly due to the presence of many dynamic variables [12] and safety requirements [28]. Moreover, it is expected that agents require active human participation for better responding to find the solution and execute tasks effectively.

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 [27]. 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 maximizes the total cumulative reward. DeepRL is an alternative based on the RL structure but also adds Deep Learning (DL) to supply the function which approximate for the value of state in continuous action-state spaces. DeepRL combines the advantages of DL with RL and can realize 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 [15].

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 [12]. The result showed that there is great potential for using RL in robot. Moreover, DeepRL has also achieved promising results, especially in manipulation skills [31, 23], and on how to grasp as well as legged locomotion [18]. However, there is an open issue relating to performance in the RL and DeepRL algorithms, which is the excessive time and resources required by the agent to achieve acceptable outcomes [10, 3]. The larger and complex of 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 showed its feasibility. IRL allows a trainer to give advice or evaluate a learning agent's behaviour [9]. There are two techniques to giving the 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 [22]. 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 maximize 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 [16]. Combining IRL with DeepRL gives a model of Deep Interactive Reinforcement Learning (DeepIRL) which can be used in continuous space with improved speed [21]. 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.

1.1 Aim & Objectives

Persistence rule-based is a new approach [2] to speed up the learning process by retaining the value from human advice for the next revisit of the state and using rule-based to policy shaping. Then, the performance of the agent may be improved and the number of interactions in the learning process also appreciably reduce. Although persistence rule-based is a promising approach at the current, it has been only used on discrete domains like mountain car and self driving car experiments. By combining persistence rule-based with DeepIRL techniques, we expect to further improve the learning performance of the agent and facilitate a method that can apply to continuous domains.

1.2 Structure

Section 2 reviews the literature. Section 3 presents the research design and methodology. Section 4 describes the approach and the technical details of artefact development. Section 5 evaluates the artefacts and discusses the RQs in Section ??. Section ?? discusses threats to validity and Section 6 concludes the report.

2 Literature Review

2.1 Reinforcement Learning Introduction

We introduce the area of RL in general before looking at definition of further technique. RL is a branch of machine learning in which artificially intelligent agents learn behaviours by interacting with their surroundings [27]. 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 [24]. 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 [26] 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 maximize the digital return that they get from interacting with the environment. Figure 1 shows the traditional learning loop between an RL agent and its environment.

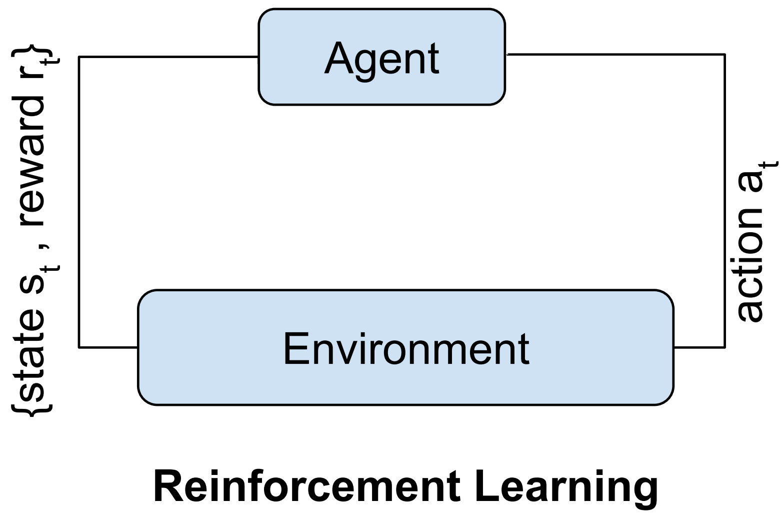


Figure 1: Reinforcement Learning framework

At any point in time, the agent monitors the environment and uses this 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 goal of the RL agent is to maximize this reward in the long run.

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 (1)

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

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).

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) [27]. 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 maximize the expected benefits from the environment. Therefore, the equation (1) can be denoted as follows:

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

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 $\gamma \rightarrow 1$, 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 [20] 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 below represents an example of a basic artificial neural network that has one hidden 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 3 below shows the simple architecture of neuron with input, output and

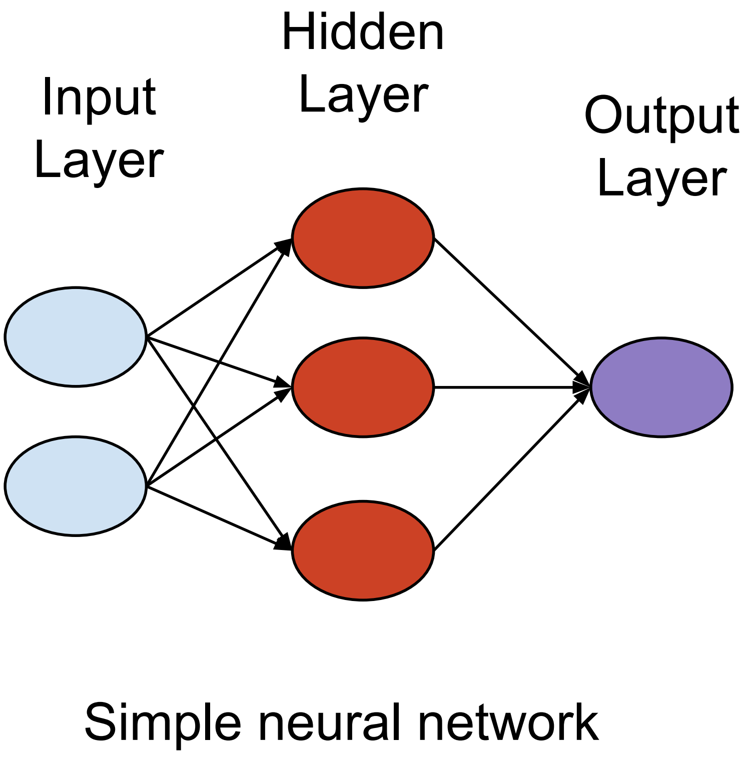


Figure 2: An example of a basic artificial neural network

activation.

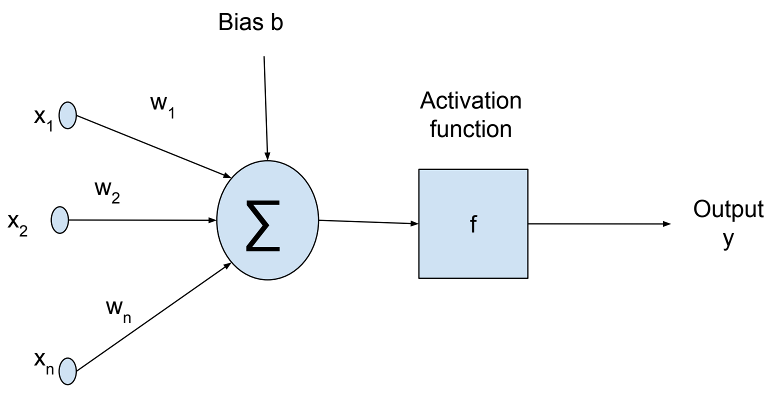


Figure 3: Simple architecture of neuron

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 minimized.

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 [25], it has been the dominant technology for computer vision problems.

2.2.2 Deep Learning with Reinforcement Learning

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. 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 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.

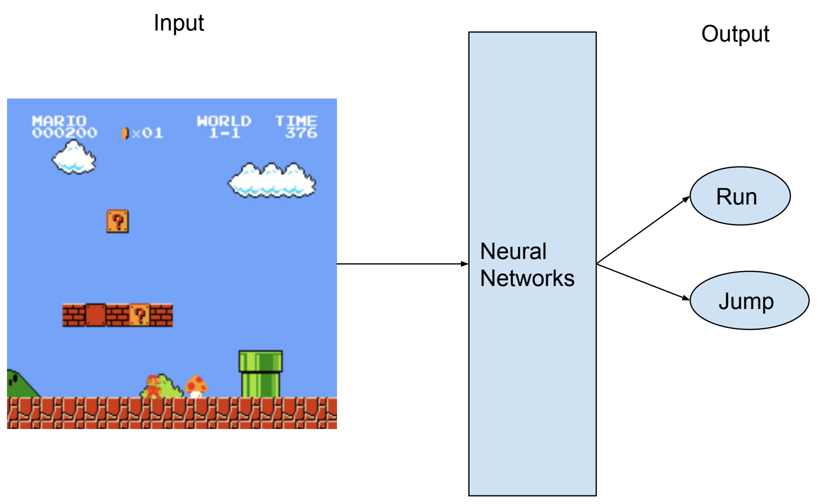


Figure 4: An example of what a policy deep learning agent map a state to the best action.

Using a deep convolutional neural network, the optimal action-value function (see Eq. (2)) that enables interaction with high-dimensional spaces extended as follows:

$$Q^*(s_t, a_t) = \max_{\pi}(\mathbb{E}[\sum_{0} \gamma^k r_{t+k} | s_t = s, a_t = a, \pi]) \tag{3}$$

A vast variety of recent advanced robot applications have been accomplished using deep reinforcement learning to teach agent complex activities including cube play [?], learning ambidextrous robot gripping [31], or categorized objects [21].

2.3 Interactive Reinforcement Learning

In Interactive Reinforcement Learning (IRL), there is an external trainer involved in the agent's learning process [9]. Figure 5 depicts the IRL solution, which includes a advisor who observes the learning process and offers guidance on the way to improve decision-making [19]. The advisor can be an expert human, or an artificial agent.

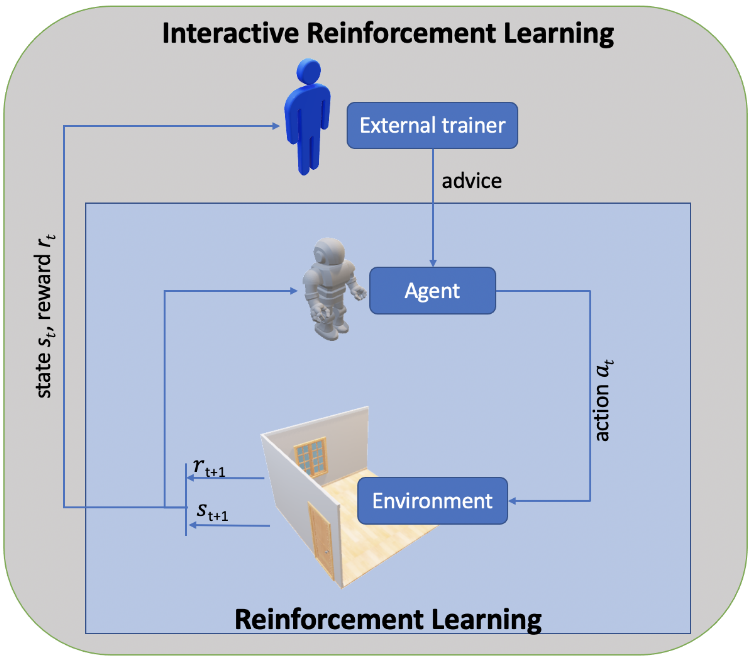


Figure 5: The involvement of external trainer to the traditional reinforcement learning process.

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 [11].

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 [30].

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 [30]. Using policy shaping, the actions proposed by the RL agent can be replaced by more appropriate actions selected by the external trainer before implementation [7].

An open problem that can significantly affect the agent's performance is inaccurate advice from the trainer [9], 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 [29]. To address the problem, a prior study [2] 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.

2.4 Persistence Rule-based Feedback

2.4.1 Ripple-down Rules

Ripple-down rules (RDR) are a method of acquiring information. It consist a kind of a data structure, well-known technique for building binary decision trees, especially suitable with ever-changing system [17]. A rule is contained in each node of an RDR tree. This RDR tree will be automatically created without understanding the meaning of the whole scheme or how new laws can affect its composition. Since no rules are written by the user, no supervision of the tree's creation is needed. Figure 6 shows an instance of conditional clause-based RDR model tree.

2.4.2 Persistence Rule-based Interaction

A recent study [2] suggests a permanent agent that records each interaction and the circumstances around particular states. The actions are re-picked when the conditions

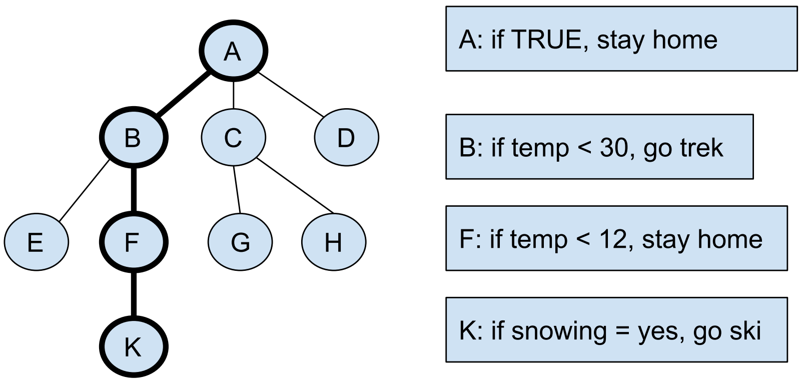


Figure 6: An example of RDR tree model.

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.

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. Probabilistic Policy Reuse (PPR) is a strategy for improving RL agents that use advice [14]. Where various exploration policies are available, PPR uses probabilistic bias to decide which one to choose, with the intention of balancing between random exploration, the use of a guideline policy, and the use of the existing policy.

Figure 12 denotes an example of rule-based interactive reinforcement learning. 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 where an agent chooses an action in a time step where the user has not recommended a prior action. 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.

3 Research Design & Methodology

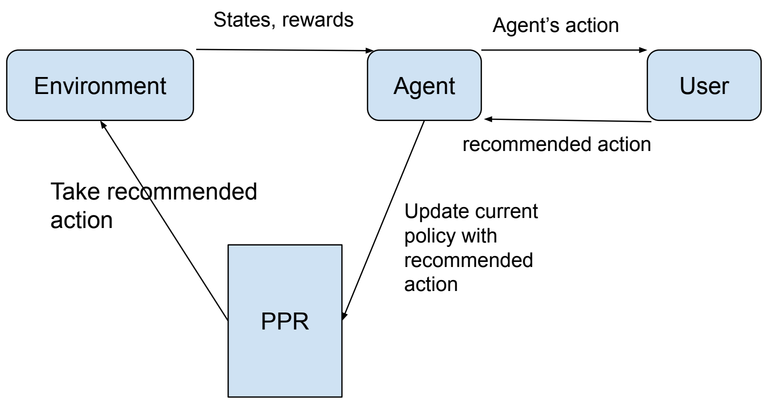


Figure 7: Process flow of a rule-based interactive reinforcement learning agent.

3.1 Research designs

- Problem statement:

  Intelligent robots have lately made their first steps into home environments. As a result, it is anticipated that robots can learn to perform tasks that are mostly considered simple for humans. However, in order for a robot to achieve human-like results, a variety of subtasks must be completed in order to satisfactorily complete a given task like perception and interpretation. To fill this gap, recent advances of IRL are shown that IRL is the suitable algorithm applied to a domestic robot scenario until now [16], despite it remains to take a long time to identify a suitable policy. This combination between with IRL with persistence rule-based approach to speed up the process is also reviewed. However, the prior study about persistence rule-based [2] is only used in a discrete domain. The aim of the project is to extend to continuous space to apply in the real complex world of domestic robots.

- Research questions:

– Is possible to extend the rule-based persistent IRL approach to continuous representation?

– To what extend can rule-based persistent feedback speed up the learning process in continuous RL scenarios?

3.2 Research methodology

- 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.

- Environment setup:

  The deep reinforcement learning environment will be implemented using the well-known library of AI gym environments [6] in reinforcement learning today. We will build a famous environment Cart Pole. There is a pole in the environment that is attached to join 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 15 degrees from the vertical or the wagon moves 2.4 units from the center. A state can be the current image (in pixels) or it can be other information that can represent the vehicle's pole, for example, the vehicle's speed and position, its angle and its velocity. Figure 10 below denotes a graphic of the Cart Pole in the AI-gym environment.

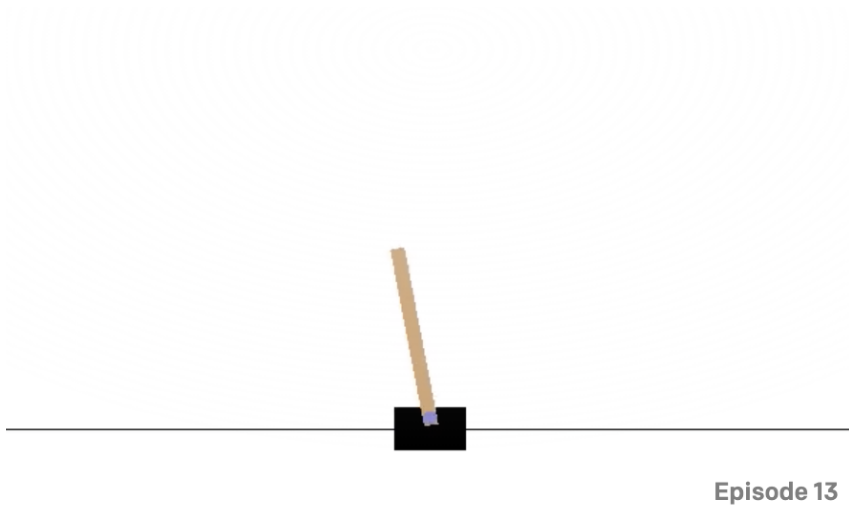
  

  Figure 8: A graphical representation of the Cart Pole environment.

  Then an enclosed space for robot movement will be set up to illustrate the example of domestic robots. We will simulate walls, obstacles such as chairs or tables as parts of the environment. The agent will be trained on how to go from initial position to target position. Software that will be used in this project is Webots or CoppeliaSim. The Figure 9 below denotes a graphic of enclosed space that will be developed in this dissertation.

- Results verification and analysis:

Figure 9: A example of Webots environment (will be replaced it in future)

While the interactive agent's human-related approach to learning is one of its greatest strengths, it may also be its greatest weakness [1, 8]. 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.

To compare agent performance, information about interactions, agent steps, rewards and interactions are recorded. To identify the efficiency of ruled-based persistence, we need to test the experiment with three cases: No interactive action, interactive actions without rule-based persistence and interactive actions with rule-based persistence.

In addition, 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 intention dependent on the advisor's knowledge of the environment. When the accuracy goes down, the advisor intends to propose not optimal action for the current state which is also based on how it knows about the environment. Frequency is the availability of the interaction of the advisor at the given time step. The accuracy and usability of the real-world simulation were simulated using data from a human test [5] and the article [2]. The value of informative and evaluative advice is beyond the scope of this study.

Frequency: 13.43%, 26.86%, 100%

Accuracy: 24.235%, 48.47%, 100%

The corresponding frequency and accuracy values above are pessimistic value, realistic value and optimistic value, respectively. The pessimistic value of frequency is 0%, however it works the same as in the case without interactive feedback. So I put the value half of realistic value as the case with the least interaction of the advisor.

The experiment will be examined in 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.

3.3 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

3.4 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 minimize 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.

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. It also includes ethical considerations, as well as an explanation of the analysis process and the means of establishing validity and reliability.

The purpose of the study is to extend the rule-based persistent IRL approach to continuous domains. This is the main driving force of the analytical experiment described in this section.

There are many environments that are used in the Reinforcement Learning community for testing develop new methods. The experiment will execute as follow:

4.1 Develop a built-in deep reinforcement learning environment

We will use Python, a very flexible programming language that is used in a wide range of applications today. This language is extremely extensible, with libraries and other add-ons that enable it to work in a wide range of applications. It is also a simple language to pick up and read.

Deep reinforcement learning environment will be deployed using AI gym environment library that is very famous nowadays in reinforcement learning. The environment with continuous state will be used. Cart Pole, also known as an inverted pendulum, is a game in which the aim is to keep the pole balanced for as long as possible. It is thought that there is something at the pole's tip that makes it easy to topple over. In this mission, the aim is to drive the cart left and right so that the pole can stand for as long as possible. We will experiment in this environment.

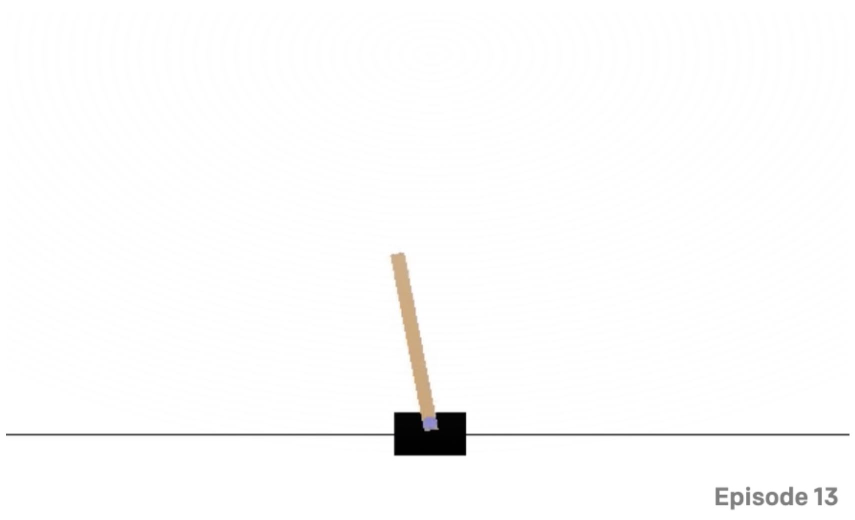


Figure 10: A graphical representation of the Cart Pole environment.

4.2 Implement interactive feedback into the built environment

In the first step, we will train simulation agents. Because agents are considered to have knowledge about the environment, multiple agents are pre-trained. We will test the number of runs until the result converges. We will choose the agent which has a certain good outcome after conduct the proposed three-time test run.

In the second step, the simulation advisor will be tested and give the comment to a fresh

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

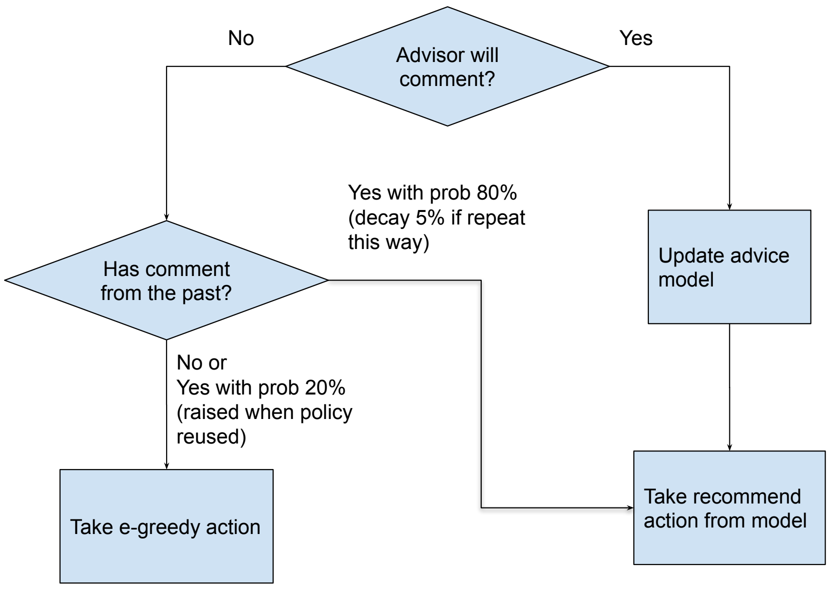
Table 1: The three simulated users designed for the experiments. These users will are not intended to be compared against each other, rather than comparing with persistence counterpart

new agent, which has no experience about the environment. The comment from the simulation advisor will be tested with different frequencies and different accuracy. This simulation of different accuracy will be executed by letting the advisor make decisions with accuracy according to the knowledge it knows. For example, an advisor with an accuracy of 70% will have 70% giving the advice and 30% will give the wrong advice. The true advice and wrong advice are based on the advisor's understanding and knowledge of the environment. The comment from the simulation advisor will be tested with different frequencies and different accuracy [2, 1]. From the chosen simulation advisor above, I will create three agents with different frequencies and different accuracy that stand for pessimistic advisors, real advisors, optimistic advisors. Table 1 shows the accuracy and frequency values for each of the three simulated users

4.3 Implement persistent approach

Agents adopt a persistence model of using Probabilistic Policy Reuse (PPR) for action selection. In probabilistic policy reuse, as shown in Figure 11, the selection of action begins with an 80% chance of reusing advice given to the agent in the past. The probability of reusing that policy is reduced by 5% for every episode (decay over time). For the remaining 20%, or if no advice is given, the ϵ-greedy action choice policy will be used.

Two kind of agent will be create of experiments: no persistence agent and persistence agent. Therefore there are six kind of agent will be created for the experiments. The table 2 lists all of six simulated user combinations tested. It also includes agents' short name, that will be used for next parts of this paper for the convenient.



No
Advisor will comment?
Yes
Yes with prob 80% (decay 5% if repeat this way)
Update advice model
Has comment from the past?
No or Yes with prob 20% (raised when policy reused)
Take e-greedy action
Take recommend action from model

Figure 11: Probabilistic policy reuse (PPR) for an IntRL agent with 80% probability of reuse the past advice

Agent	Short name
Non Persistence Pessimistic Advisor	NPP
Non Persistence Real Advisor	NRP
Non Persistence Optimistic Advisor	NPO
Persistence Pessimistic Advisor	PP
Persistence Real Advisor	PR
Persistence Optimistic Advisor	PO

Table 2: Simulation advisor combinations for persistent agent testing

4.4 Implement the rule-based approach

In the discrete environment, a ripple down tree has been built for the rule-based model. Here we need to build a similar rule-based model to use for the continued space state. However, the continuous space state has unlimited states. If we try to create a rule-based rule to keep previously visited states, there is a high chance that the generated tree will be very complex because the space of deep reinforcement learning is usually one image or multiple features. The approach is that we will use feature extraction to reduce the number of spatial dimensions in a certain way. For example in CartPole, features would be the position of the cart, the velocity of the cart, the angle of the pole, and the rotation rate of the pole, a total of four features. In this environment, we may need to do some feature extraction if it is necessary. In the input environment is an image, we need to do feature extraction to reduce the number of spatial dimensions to about ten. After feature extraction, we will conduct the Ripple-down Rule Tree to complete the rule-based model. Figure 12 indicates the flow of rule-based approach in our model

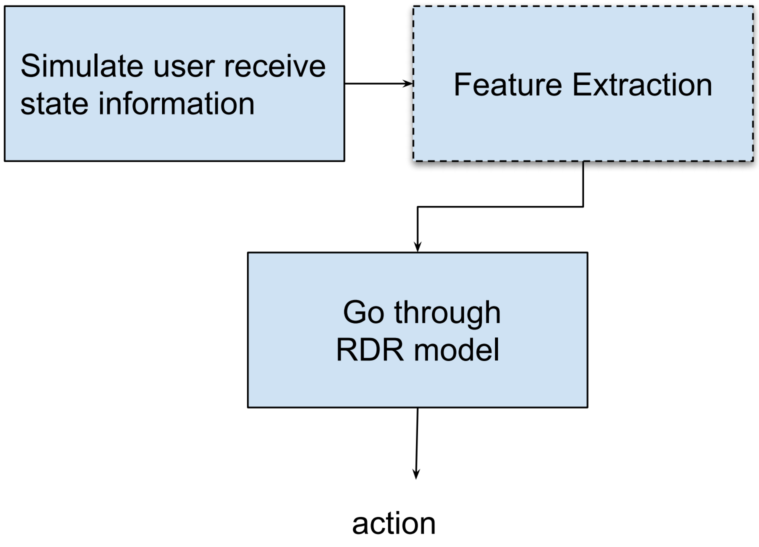


Figure 12: Flow of rule-based approach

4.5 Build an environment for domestic robots

We will build an environment for domestic robots: a closed environment built by Webots or CoppeliaSim. The environment with the goal to train the robot to go from the initial position to the target position. The Figure 9 denotes a graphic of domestic robot that will be developed by Webots. Then, we apply the algorithms developed in above step in this environment.

5 Empirical Evaluation

5.1 Develop a built-in deep reinforcement learning environment

- Evaluation method:

  Code with python and AI gym environment. The code will be submitted to Github. (show link)

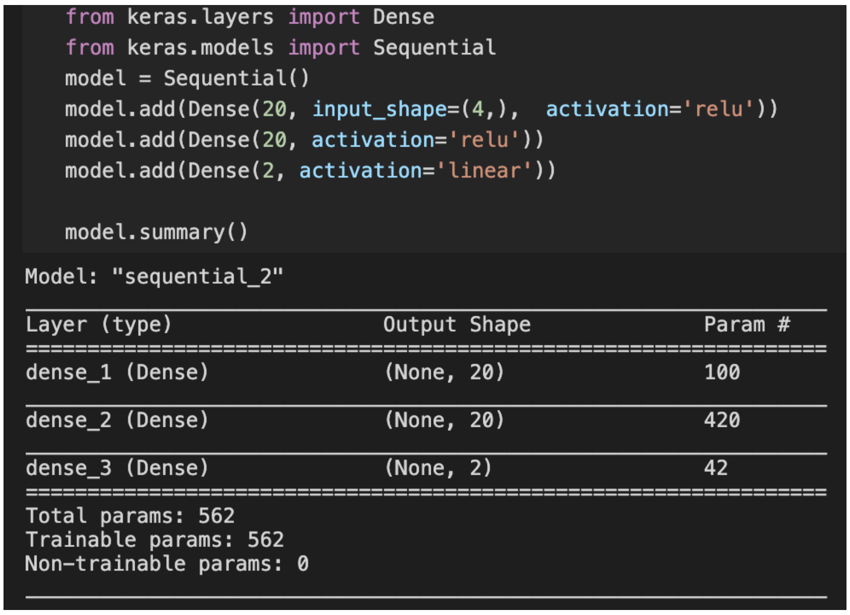
The environment will be CartPole in a continuous environment. Need to show a model of network for action choosing (neural network model)

One image of first three hundreds iteration about average reward will be displayed

- Result:

  Code is implemented in Python AI gym environment with the link is below https://github.com/mwizard1010/robot-control/tree/main/Code

  The Figure 13 shows the structure of the implemented neural network model for deep learning which uses for picking the action of agent



```
from keras.layers import Dense
from keras.models import Sequential
model = Sequential()
model.add(Dense(20, input_shape=(4,),  activation='relu'))
model.add(Dense(20, activation='relu'))
model.add(Dense(2, activation='linear'))

model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 20)                100
_________________________________________________________________
dense_2 (Dense)              (None, 20)                420
_________________________________________________________________
dense_3 (Dense)              (None, 2)                 42
=================================================================
Total params: 562
Trainable params: 562
Non-trainable params: 0
_________________________________________________________________
```

Figure 13: Implementing neural network model

The experiment is tested with the hyper parameters as follows: initial value of ϵ = 1, ϵ decay rate of 0.995, learning rate $\alpha = 0.01$, and discount factor $\gamma = 0.99$. The average collection reward shown in Figure 14 is represented by the yellow line after three hundreds episodes in the Carl Polar environment. The value of rewards begins to converge and stabilize after 250 episodes. It shows that the time required for the training process is rather high to obtain a stable behavior by RL algorithms.

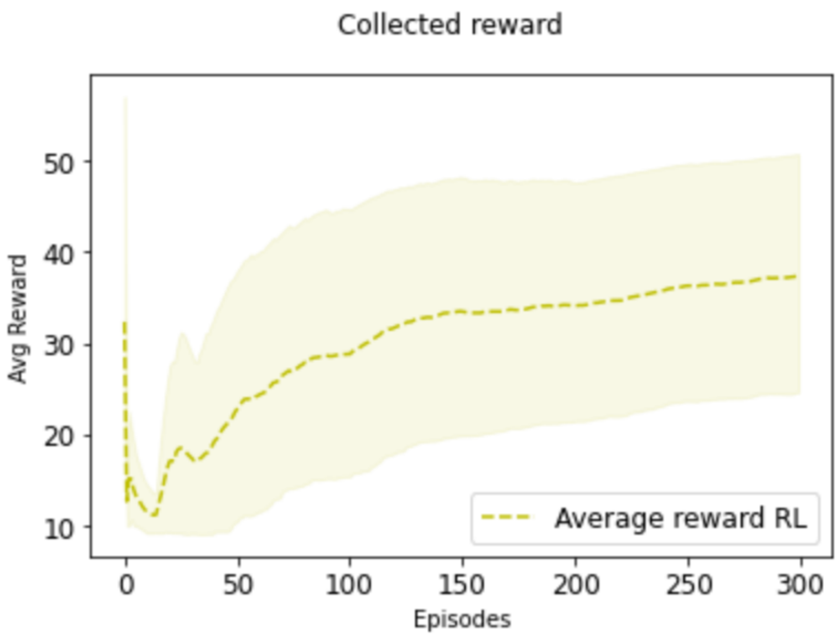


Figure 14: Three hundreds episodes for deep reinforcement learning built with CarlPole environment. The shaded area indicates the deviation between the minimum and maximum values of the agent value after multiple training.

5.2 Implement interactive feedback into the built environment

- Evaluation method:

  Built simple interactive feedback advisor to help agent on learning step. Show the result after 300 step, compare with case of without interactive feedback

  Built three kind of agents with different accuracy and availability described in Table 1

  Show the result after 300 step

- Result:

  We built an interactive feedback advisor with 100% availability and 100% accuracy at the first step. After training the first agent, the second agent was trained using the IRL method to manipulate selected actions. Figure 15 shows a comparison between two algorithm RL and IRL in Cart Pole continuous environment, where the reward value of IRL and RL is represented by the red line and the yellow line, respectively. In this experiment, the IRL algorithm achieved the absolute score very early after a few episodes. It is clear that an agent with an advisor leads to better results compared to without an advisor. The learning speed greatly improved compared to the traditional RL very fast due to the use of an optimistic agent. However, the reality can be very different, we will conduct more experiments with real agents and pessimistic agents in further experiments.

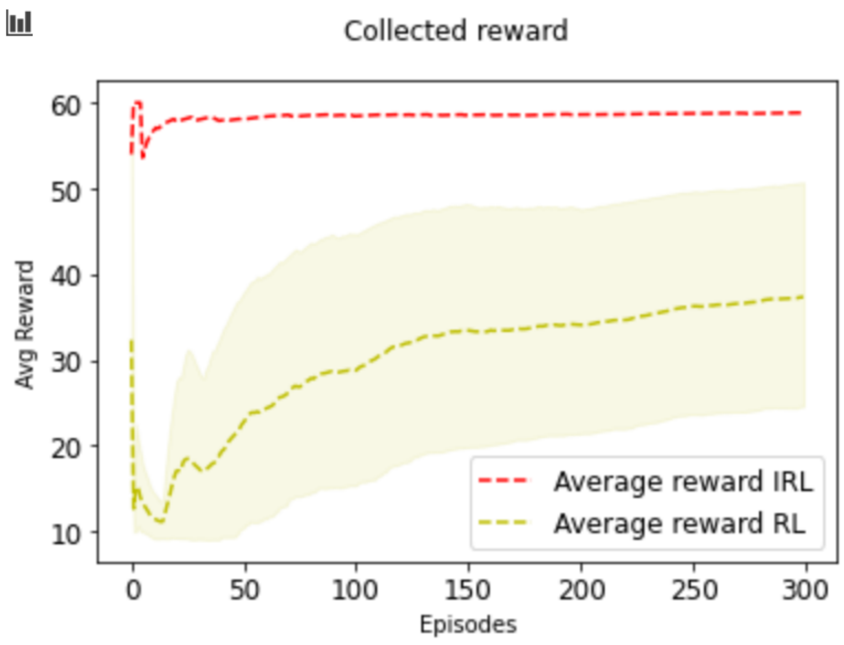


Figure 15: Three hundreds episode for IRL built with CarlPole environment, compared with RL

5.3 Implement persistent approach

- Implement flow of PPR and show implementation code
- Design six kind of agents which is described in Table 2 and show implementation code

5.4 Implement the rule-based approach

- Implement flow of rule-based approach and feature extraction and show code
- Show the result after 100 step, compare with case of of all six advisors

5.5 Implement environment for domestic robots

- Built the new environment for domestic robots using Webots or CoppeliaSim. Show an image of environment here

- Rerun the all cases training with six advisors in new environment. Show the result after 100 step or more

6 Conclusion & Future Work

6.1 Future Work

In this work, we have finished a literature review on basic knowledge information required for the project, made a research design and research methodology to deal with the problem. The artefact development is designed in five sprints: develop a built-in environment, implement interactive feedback, implement a persistent approach, implement the rule-based approach, and build an environment for domestic robots. We have completed the first spring for development of the deep reinforcement learning built-in environment and a part of the second sprint to built an advice interaction system.

Current results show that the learning speed of IRL algorithms tested with optimistic agent improved significantly compared to the traditional RL. In the future, we need to take further experiments with real agents and pessimistic agents. In addition, the next sprints also need to set up and follow.

References

[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] A. Bignold, F. Cruz, R. Dazeley, P. Vamplew, and C. Foale, Persistent Rule-based Interactive Reinforcement Learning, (2021).

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

[7] T. Cederborg, I. Grover, C. L. Isbell Jr, and A. L. Thomaz, Policy shaping with human teachers., (2015), pp. 3366–3372.

[8] 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.

[9] 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.

[10] 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).

[11] F. Cruz, J. Twiefel, S. Magg, C. Weber, and S. Wermter, Interactive reinforcement learning through speech guidance in a domestic scenario, (2015), pp. 1–8.

[12] 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).

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

[14] F. Fernández and M. Veloso, Probabilistic policy reuse in a reinforcement learning agent, (2006), pp. 720–727.

[15] 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.

[16] 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.

[17] D. Herbert and B. H. Kang, Intelligent conversation system using multiple classification ripple down rules and conversational context, Expert Systems with Applications, 112 (2018), pp. 342–352.

[18] 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.

[19] W. B. Knox and P. Stone, Interactively shaping agents via human reinforcement: The tamer framework, (2009), pp. 9–16.

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

[21] 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).

[22] 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.

[23] 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.

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

[25] 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.

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

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

[28] 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.

[29] 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.

[30] A. L. Thomaz, G. Hoffman, and C. Breazeal, Real-time interactive reinforcement learning for robots, (2005).

[31] 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.