EFFECT OF HUMAN PRIOR KNOWLEDGE ON GAME SUCCESS AND COMPARISON WITH REINFORCEMENT LEARNING

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MERT HASANOĞLU

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COGNITIVE SCIENCE

DECEMBER 2019

EFFECT OF HUMAN PRIOR KNOWLEDGE ON GAME SUCCESS AND COMPARISON WITH REINFORCEMENT LEARNING

Submitted by **MERT HASANOĞLU** in partial fulfillment of the requirements for the degree of **Master of Science in Cognitive Science Department, Middle East Technical University** by,

Prof. Dr. Deniz Zeyrek Bozşahin
Dean, **Graduate School of Informatics Institute** _____

Prof. Dr. Cem Bozşahin
Head of Department, **Cognitive Science** _____

Assist. Prof. Dr. Murat Perit Çakır
Supervisor, **Cognitive Science Dept.,METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Cengiz Acartürk
Cognitive Science Dept., METU _____

Assist. Prof. Dr. Murat Perit Çakır
Cognitive Science Dept., METU _____

Assist. Prof. Dr. Murat Ulubay
Business Dept., YBU _____

Date: **12 December 2019**

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: MERT HASANOĞLU

Signature :

ABSTRACT

EFFECT OF HUMAN PRIOR KNOWLEDGE ON GAME SUCCESS AND COMPARISON WITH REINFORCEMENT LEARNING

Hasanoğlu, Mert

M.S., Department of Cognitive Science

Supervisor : Assist. Prof. Dr. Murat Perit Çakır

December 2019, 44 pages

This study aims to find out the effect of prior knowledge on the success of humans in a non-rewarding game environment, and then to compare human performance with a reinforcement learning method in an effort to observe to what extent this method can be brought closer to human behavior and performance with the data obtained. For this purpose, different versions of a simple 2D game were used, and data were collected from 32 participants. At the end of the experiment, it is concluded that prior knowledge, such as the meaning of objects and colors, have an impact on human performance. It was observed that the reinforcement learning agent failed to finish the same game. Various attempts have been made to improve performance and to achieve human-like behavior. In one of these, mini-games were prepared to introduce prior knowledge of the objects and the interaction with them. In another trial, a model is created with the game data collected from participants, and the agent is trained using this model as an exploration strategy. Only when the human data is used as an exploration strategy, the agent succeeded in finishing the game. Although the performance of the reinforcement algorithm is increased, human-like behavior is not observed. The conclusion is that it is more meaningful to consider prior knowledge within the context of exploration strategy, and having prior knowledge is not enough in achieving human-like behavior.

Keywords: reinforcement learning, human priors, artificial neural networks, machine learning, deep learning

ÖZ

İNSAN ÖN BİLGİSİNİN OYUN BAŞARISINA ETKİSİ VE PEKİŞTİRMELİ ÖĞRENME İLE KIYASLANMASI

Hasanoğlu, Mert

Yüksek Lisans, Bilişsel Bilimler Bölümü

Tez Yöneticisi : Dr. Öğr. Üyesi Murat Perit Çakır

Aralık 2019 , 44 sayfa

Bu çalışma, insanların önceki deneyimlerinden elde ettiği bilgilerin, ödül olmayan bir oyun ortamındaki insan başarısı üzerindeki etkilerini bulmayı ve daha sonra insan performansını bir pekiştirmeli öğrenme yöntemiyle karşılaştırmayı ve bu yöntemi insan davranışlarına ve performanslarına, elde edilen verilerin kullanılması ile, daha da yakınlaştırmayı amaçlamaktadır. Bu amaçla, basit bir 2B oyunun farklı versiyonları kullanılmış ve 32 katılımcıdan veri toplanmıştır. Deney sonunda, nesnelerin ve renklerin anlamı gibi önceki bilgilerin insan performansı üzerinde etkisi olduğu sonucuna varılmıştır. Pekiştirmeli öğrenme yöntemleri ile eğitilen ajanın ise aynı oyunu bitiremediği görülmüştür. Performansı arttırmak ve insan benzeri davranışları elde etmek için çeşitli denemeler yapılmıştır. Bunlardan birinde, nesneler hakkında önceden bilgi edinilmesi ve onlarla etkileşimin sağlanması için mini oyunlar hazırlanmıştır. Bir başka denemede, katılımcılardan toplanan oyun verileriyle bir model oluşturulmuş ve bu model pekiştirmeli öğrenme algoritması için bir keşif stratejisi olarak kullanılmıştır. Sadece insan verilerinin bir keşif stratejisi olarak kullanıldığı durumda, pekiştirmeli öğrenme ajanı oyunu bitirmeyi başarmıştır. Pekiştirmeli öğrenme algoritmasının performansı arttırılmış olmasına rağmen, insan benzeri davranış gözlenmemiştir. Sonuç olarak insanların sahip olduğu önceki bilgilerin oyun performanslarını etkilediği görülmüştür. Pekiştirmeli öğrenme yöntemleri kullanılarak eğitilen ajanın ise bu tip bir bilgiye sahip olmasının uygulama noktasında kolay olmadığı görülmüştür. Son olarak ajanın eğitiminde insan verileri kullanıldığı durumda dahi insan benzeri davranış gözlemlenmemiştir.

Anahtar Kelimeler: pekiştirmeli öğrenme, insan ön bilgisi, yapay sinir ağları, makine öğrenmesi, derin öğrenme

to My Family and Best Friend who left us too early ...

ACKNOWLEDGMENTS

Firstly, I would like to thank my academic advisor, Asst. Prof. Dr. Murat Perit Çakır, for his patience and guidance throughout my years at the department. Without his guidance I could have lost my way between my day job and masters studies.

I would like to thank İbrahim Kaya for his huge help with the reinforcement learning parts. Without him it would be impossible to finish all the work done.

Last but not least I would like to thank my family for always supporting me and being there whenever I needed them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ANN	Artificial Neural Network
ANOVA	Analysis of Variance
CNN	Convolutional Neural Network
DDQN	Double Deep Q-Network
DL	Deep Learning
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
LG	LaddermaskedGame
MG	MultiplegoalGame
ML	Machine Learning
NG	NosemanticsGame
NRG	NosemanticsReversedGame
OG	OriginalGame
RL	Reinforcement Learning
RG	ReversedGame

CHAPTER 1

INTRODUCTION

Reinforcement learning is a long-known method, although it became popular not too long ago. The popularly known version dates back to the 1980s. Recently, reinforcement learning has met with deep learning, and deep reinforcement learning has emerged. Thanks to this development, problems that could not be solved in the past can now be solved. Here, of course, similar to the advances seen in machine learning in recent years, new and more powerful hardware is one of the main driving forces.

In its purest form, reinforcement learning involves an agent that learns how to behave optimally, given the rewards and an environment that provides state and reward information given the actions taken by the agent (Sutton & Barto, 2018). Figure 1.1 shows this framework. Games are often used as environments to test the performance of different algorithms.

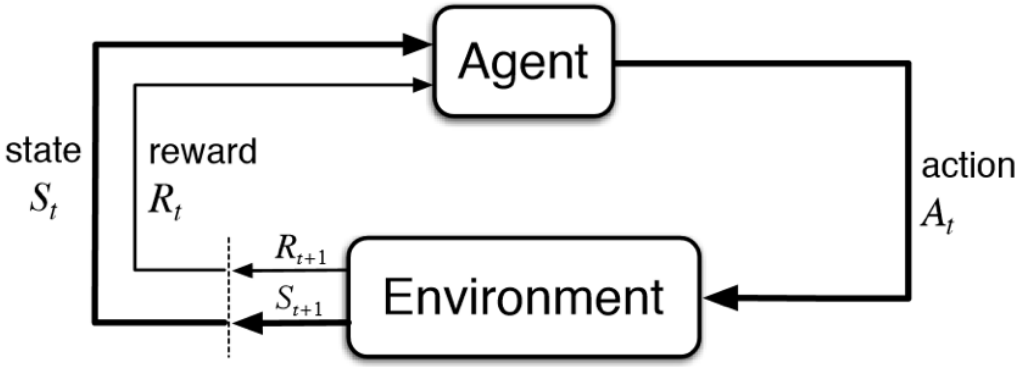


Figure 1.1: *Basic Reinforcement Learning Scheme (Sutton & Barto, 2018)*

Deep reinforcement learning is first used by Google's DeepMind branch, which demonstrated the algorithm using ATARI 2600 games as the environment (Mnih et al., 2015). Their paper called "Human-level control through deep reinforcement learning" is published in Nature. The popularity of deep reinforcement learning increased tremendously afterward. This increase can be observed from Figure 1.2, taken from Google Trends, which shows interest over time. Up to 2015, there is close to none interest for the keyword deep reinforcement learning, which starts to increase at that point and continue increasing steadily till now. At this point, it would be useful to specify a point

regarding naming. The only difference between reinforcement learning and deep reinforcement learning is that deep reinforcement learning uses a deep structured machine learning method. Due to its broader scope, the term reinforcement learning will be used in this study.

Figure 1.2: *Deep Reinforcement Learning Google Search Trend (Google Trends, 2019)*

This increasing interest in reinforcement learning can be attributed to its similarity to human learning, which also incorporates learning by trial and error. Google's DeepMind unit managed to come up with an algorithm that used deep reinforcement techniques in harmony with other, more conventional ones to finally conquer the popular Chinese game called Go. They demonstrated the success of the algorithm by defeating the Go World champion Lee Sedol (Silver et al., 2018; 2017). They also created an agent called AlphaGo Zero that learned Go by only using deep reinforcement techniques, which defeated AlphaGo by 100 to 0 (Hassabis & Silver, 2017). Apart from Go, generally, game environments are used to test different techniques. These environments can be more simple ones like Atari games or much more complex ones like DOTA or StarCraft II. For example, DeepMind recently introduced its agent called AlphaStar, which can win against human players in StarCraft II (DeepMind, 2019).

These accomplishments and interests are maybe promising, but apart from research studies and games, real-world applications are very hard to come by. This fact can be related to the problems that are faced using reinforcement learning which can be summarized as:

- Finding correct reward functions for complex behavior is very hard (Palan, Landolfi, Shevchuk, & Sadigh, 2019)
- Due to reward function or other phenomena (for example local minima) agent can gain reward but act not as intended (Everitt & Hutter, 2019)
- If reward is sparse or none, learning becomes harder (Pathak, Agrawal, Efros, & Darrell, 2017)
- Compared to humans, learning takes too much time and computational resource (Botvinick et al., 2019)
- Each new problem requires extensive training even if the concepts are similar (Yin & Pan, 2017)
- Exploration and exploitation dilemma (Kaelbling, Littman, & Moore, 1996)

As mentioned before, reinforcement learning is fundamentally close to human learning, thus studying humans can shed light on some of these problems. Moreover, these studies can yield agents that act like humans, and these agents can be used in cognitive studies. For this purpose, in this work, the effect of prior knowledge on game success is investigated in a setting where there is no reward provided by the environment.

1.1 Purpose and Scope of the Study

There are three aims of this study. The first one is to show that prior knowledge has an impact on game success. Prior knowledge is a very broad concept, but for the sake of this study, the ones deemed most important were selected. Then, in order to observe the effect of these prior knowledge elements, six simple games were developed, and experimental data was collected. Details are presented in the subsequent sections. The second aim of this study is to show the similarities and contrasts of human gameplay and reinforcement learning algorithms. This was done by implementing a popular reinforcement learning algorithm, namely double Q-learning (DDQN), and comparing it with human gameplay data. As part of the study, the performance of the method mentioned above was also improved by using human gameplay data. Finally, the agent was evaluated in terms of game performance and similarity to human behavior.

Hypotheses associated with these aims are as follows:

1. Human prior knowledge plays an important role in game success and this knowledge is used extensively while playing games
2. Known popular reinforcement algorithms fail at games that has no or sparse reward signals. Even if a reward signal is designed it would not be enough to surpass human performance.
3. With the introduction of prior knowledge, reinforcement algorithms can behave more like humans and their performance can get better.

1.2 Thesis Outline

The following chapters of this thesis were organized as follows. In Chapter 2, a literature review regarding prior knowledge and reinforcement learning is provided. Moreover, a brief information regarding reinforcement learning, focusing on the DDQN algorithm used (Van Hasselt, Guez, & Silver, 2016), is also provided. In the first half of Chapter 3, the methodological approach followed in the experiments regarding prior knowledge is given. The remaining half of the chapter is dedicated to the results of these experiments. In Chapter 4, the performance of the vanilla DDQN algorithm is presented. Then the chapter is finalized with the results obtained from our attempts to improve

agent performance. Finally, in Chapter 5, a general discussion and conclusion are provided.

CHAPTER 2

LITERATURE REVIEW

In this section, literature information about the topics related to this thesis study will be given. The topics related to this study can be organized under two mutually informing threads, one focusing on insights from cognitive science related to notions of reinforcement and learning, and the latter from computer science regarding advances in implementing such algorithms. In the first part, the basics of reinforcement learning in cognitive sciences and the literature on the concept of prior knowledge will be discussed. In the second part, the development of reinforcement learning algorithms from the past to the present in the computer science literature will be provided.

Reinforcement learning is basically a trial and error phenomenon, as the learning process requires the agent to interact with the environment through trial and error. Another central element is feedback from the environment, which optimally involves a reward, although there are cases where no reward is provided by the environment like the game used in this study. If the concept of trial and error is considered, the oldest and foundational research is regarding animal behavior, not human. Thorndike (1911) is one of the first studies on this subject. Edward Thorndike explains learning in animals by two laws, "the Law of Effect" and "the Law of Exercise" (Thorndike, 1911). The law of effect says that the animal will exhibit behavior with negative feedback less, and with positive feedback more. Thorndike uses the word discomfort for negative feedback and satisfaction for positive feedback. The law of effect also laid the foundation for operant conditioning (Skinner, 1938). In operant conditioning, the term reinforcement consists of two types; negative reinforcement and positive reinforcement. Both of these types shape the behavior. The difference is that negative reinforcement encourages the behavior by removing negative feedback, whereas positive reinforcement does this with positive feedback. There is also a concept of punishment which is a negative feedback discouraging the behavior. Comparing with reinforcement learning, concepts of positive reinforcement and punishment can be considered similar to the reward function. Negative reinforcement is the main difference here, as it is not used in reinforcement learning.

Although Skinner uses the term reinforcement, he is not the one using it first. The notion of reinforcement was first used in classical conditioning studies pioneered by Pavlov (Pavlov & Gantt, 1928). In classical conditioning, the re-

inforced behavior is not directly associated with the stimulus; it is somewhat indirect. The best-known example of classical conditioning is the establishment of a link between a bell and a dog's saliva by ringing a bell each time the dog is given food. Here, a link is established between the bell and animal behavior by giving food, but this is an indirect effect. In this respect, classical conditioning differs from operant conditioning. It can be said that reinforcement algorithms, as applied in the context of this work, is more similar to operant conditioning.

In this study, the prior knowledge refers to the object knowledge that the person has before playing the game, knowledge about how to interact with these objects and the game environments and physics. In the game environment used in this study, fire pits symbolized by flames were used. An example of object information is that the player knows that he should escape from this object when he sees this image of the flame. It is known that even 3 month infants can recognize shapes that are three dimensional (Kraebel, West, & Gerhardstein, 2007). Infants can even recognize objects in visually cluttered scenes which is clearly a more demanding task (Spelke, 1990). There are many studies on how object recognition works, its development with age and its underlying biological principles (Baillargeon et al., 2012; Diamond, 1991; Kraebel et al., 2007; Piaget & Piaget, 2006; Reynolds, 2015; Wilcox, Stubbs, Hirshkowitz, & Boas, 2012). Together with object recognition, how to interact with these objects also becomes something to be learned. Although things done by the infants may seem meaningless from an outsider's perspective, on the contrary it can be said that these actions are their way of exploring and making sense of the physical world. To acquire knowledge of objects and how to interact with them infants continually explore and even change their behavior to be better explorers (Lobo, Kokkoni, de Campos, & Galloway, 2014). Finally, with experience humans gather information regarding game environments. Even if the person did not play any game the exposure to a gaming environment cannot be evaded. Moreover, our understanding of the real physical world is reflected on to the game environment. For example, it becomes harder to play a game when direction of the gravity is changed from up to down to left to right (Dubey, Agrawal, Pathak, Griffiths, & Efros, 2018).

Although the concept of reinforcement learning has similarities with learning in humans and animals, as mentioned earlier, it is clear that the two areas can benefit from each other more. There is an increasing number of research done recently to highlight how human cognition can help reinforcement learning algorithms (Doshi-Velez & Ghahramani, 2011; Dubey et al., 2018; Lake, Ullman, Tenenbaum, & Gershman, 2017; Rosenfeld & Tsotsos, 2018; Tsividis, Pouncy, Xu, Tenenbaum, & Gershman, 2017). There are also algorithms based on human cognition. It is known that exploration is a problem for reinforcement learning algorithms. A recent algorithm developed tried to solve this problem by using the concept of human curiosity (Pathak, Agrawal, Efros, & Darrell, 2017). Here an intrinsic reward based on curiosity is defined in addition to the external reward from the environment. This way authors tried to have success in environments with sparse or no rewards. In another study human game play data is used to create a reward function (Ibarz et al., 2018).

Nine different Atari games are played by human demonstrators and a network providing reward information is constructed. Instead of rewards from the environment, rewards from the built network is used with a Deep Q-Network. Better performance compared to imitation learning is reportedly achieved (Ho & Ermon, 2016). For complex behavior, finding the correct reward function is a very hard task. If there are bugs in the reward function, the agent can manipulate these back doors not learning the desired behavior. This is also known as reward hacking (Hadfield-Menell, Milli, Abbeel, Russell, & Dragan, 2017). A solution for this is to actively use human feedback (Mindermann, Shah, Gleave, & Hadfield-Menell, 2018).

2.1 Double Deep Q-Learning (DDQN)

Before describing the DDQN algorithm, it will be useful to go back to Figure 1.1 and make some basic definitions.

- Agent: Entity that learns how to behave in the given environment.
- Environment: Can be regarded as the simulation environment. Outputs state and reward information given the previous state of the agent and the chosen action.
- Reward (R, r): Value gained by the agent by taking a specific action at time = t, which can be negative, positive or zero. The agent tries to maximize the total gained reward. As rewards directly affect the training it is very crucial to determine it. For complex behavior it is very hard to come up with a correct reward function.
- State (S, s): Information regarding the environment known to the agent. This information is then used to determine the action to be taken.
- Policy (π): Policy determines the agent's actions given the state.
- State-Value Function (V, v): Expected return of an agent starting from a state following a policy.
- Action-Value Function (Q, q): Expected return of an agent starting from a state taking an action and following a policy.
- Discounted Reward (G): It is the reward obtained by the agent following a policy that is discounted with a parameter that is between 1 and 0. It depicts the importance of previous rewards in agent's decisions. If it is 0 only the most recent reward is used.

In the light of these basic concepts, first of all, the questions of what reinforcement learning is and what it aims will be answered. Then briefly, the Q-learning algorithm (Watkins & Dayan, 1992) that is used in the popular reinforcement learning method deep Q-learning (DQN) (Mnih et al., 2015) will be explained. DQN algorithm is the first example of deep reinforcement

learning and is also the method DDQN (Van Hasselt et al., 2016), algorithm used in this study, based on. Finally, the DDQN algorithm will be summarized.

As already mentioned, the fundamental goal in reinforcement learning is to obtain an agent that will receive the maximum reward provided by the environment. In the simplest case of the problem, the action taken by the agent always gives the same result for each state, and a reward is provided for each action. In reality, problems can be much more complicated. For example, the reward may be given at intervals, not always, or only when the target state is reached. In the most extreme case, no reward is provided by the environment. Another difficulty is that the result of action for a state may not always be the same. For example, in a grid world game, when the player presses the right button, the probability of going to the right could be 80 percent, and the probability of going up could be 20 percent.

The difference between the discounted reward and reward is also critical. When R is reward, G is total discounted reward and t is the time step, discounted reward equation can be written as follows (Sutton & Barto, 2018);

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+1+k} \tag{2.1}$$

Here γ is defined as the discount rate and can take values between 0 and 1. In Figure 2.1, a grid world game environment is given. The value in each box indicates the reward that the agent will receive when the agent reaches that box. The goal state is shown in green, and the starting state is shown in blue. In the case of instant reward being maximized (γ = 0), the agent will not be able to reach the goal state because it must first go through the states that give a negative reward. In such a simple environment, the discount rate can be easily assessed, but in more complex and realistic environments, it is not that easy, and different values should be tested for best performance.

The task that the reinforcement learning agent tries to learn in the environment can be of two different types. As in the grid world game given in Figure 2.1, tasks with a start and end states are defined as episodic tasks, and the course of engagement between these two terminal states is called an episode. Another type of task is a continuous task. In this case, there is no specific start and end state. The agent continues to run unless the user terminates it. An agent trained in a car simulation will be called an episodic task if it is trained to travel between two specific points, but it will be a continuous task if it is expected to travel freely in the city.

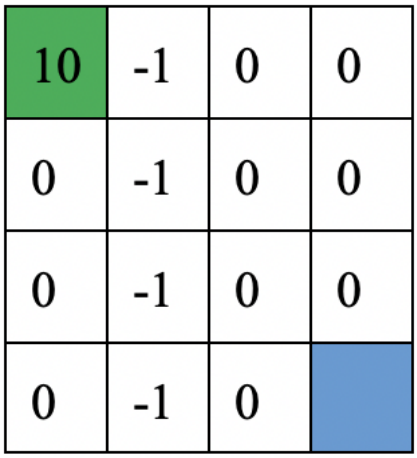


Figure 2.1: *An Example Grid World Environment*

Reinforcement learning methods can be classified in many different ways. If the method involves the modeling of the environment, such methods are called model-based methods. If there is no such modeling, they are called model-free methods. If a method depends on the policy used during learning, such methods are classified as on-policy methods. If the policy followed by the agent during training is not vital for the method, such methods are called off-policy methods. Another classification is made regarding what is learned during training. The methods that directly learn a policy are called policy-based methods. If the value or action functions are learned, the method is called value-based. Finally, if a method takes advantage of both policy-based and value-based methods, it is classified as an actor-critic method. Q-learning, which is also at the heart of the algorithm used in this study, is a model-free, off-policy, and value-based method.

As the name Q-learning suggests, it is a method for obtaining the optimal values of the action-value function. When the values of the optimal action-value function are obtained, the agent can move within the environment using these values. Table 2.1 provides example action-value function values for an environment with eight states and four actions. Once the table is obtained, it is straightforward to have the agent select an action hence have a policy. The action with the highest q value is selected for any given state. For example, if the agent is in state S3, action A1 is selected, which yields the maximum q value for that state. When in the S8 state, the selected action is A3. At this point, there are two questions yet to be answered; how to obtain optimal action-values like given in the example table and what happens to the table when there are too many states and actions which usually is the case.

Table 2.1: Example q Value Table

	S1	**S2**	**S3**	**S4**	**S5**	**S6**	**S7**	**S8**
A1	-0.5	-0.2	0.4	-0.5	0.0	0.5	0.2	-0.4
A2	-0.6	0.3	0.3	0.1	0.5	0.2	0.5	0.0
A3	0.5	0.1	0.7	-0.1	0.2	-0.5	-0.4	0.2
A4	0.1	-0.1	-0.1	0.3	0.3	0.4	-0.2	0.1

In order to answer the first question, the Markov decision process (MDP) must be defined. MDP can be defined as an environment whose all states qualify as Markov states. Markov state, which bears the name of Russian mathematician A. A. Markov means that the future can only be explained by the present, independent of the past. For example, in the grid world environment in Figure 2.1, assume that the agent is in any state adjacent to the goal state. The location of the agent in the next step is understandable only by the state in which it is currently located and regardless of which states it has previously taken. Mathematically, this situation can be expressed as follows (Puterman, 1994; Silver, 2015; Sutton & Barto, 2018);

$$\mathbb{P}[S_{t+1}|S_t] = \mathbb{P}[S_{t+1}|S_1, S_2, S_3, ..., S_t] \tag{2.2}$$

If the problem at hand is a MDP then Bellman optimality equation can be used to calculate the q values. This equation is defined by the American mathematician Richard E. Bellman, who is also regarded as the founder of dynamic programming (Bellman, 1954). Bellman equation makes it possible to calculate action-value function or state-value function, which is given in Equation (2.3). Q-learning (Watkins & Dayan, 1992) makes it possible to iteratively solve Bellman equation for an approximate optimal q value. The equation for this iterative approximation is provided as Equation (2.4) (Sutton & Barto, 2018).

$$Q(s, a) = r + \gamma \max_{a'} Q(s', a') \tag{2.3}$$

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1}\gamma \max_{a} Q(S_{t+1}, a) - Q(S_t, A_t)] \tag{2.4}$$

In Equation (2.4), α is the learning rate which can get values from the range (0, 1]. The Q-learning algorithm cannot succeed with the so far defined version. The reason for this is that when the agent moves according to the existing q function, it will act in a loop visiting the same states. The solution, in this case, is the employment of an exploration algorithm. As mentioned before, Q-learning is an off-policy method; thus, this exploration algorithm would not directly affect the learning process, but it must ensure the exploration of the environment. The most popular method of exploration is the ϵ-greedy method (Sutton & Barto, 2018). In this method, an ϵ value is determined, and the agent is moved randomly in proportion to this value. For example, if ϵ is 0.1, 90 percent of the time, the agent will act according to the q function being approximated, the remaining 10 percent will be random. The problem known as the exploration-exploitation dilemma is faced at this point. It would make sense to act highly randomly at the beginning; in this way, the agent would experience more new states. On the other hand, as q function predictions improve, it will be more meaningful for the agent to perform much less random action. As there is no metric that can be calculated to know how much the environment is explored, determining ϵ value is not straightforward. A fixed ϵ value is, most of the time, not preferred. Instead, a greater value is chosen

at the beginning of the training, and the value is reduced depending on a specific function or heuristic decided before the training. As optimum values for ϵ cannot be determined analytically, during training, different values and schedules may be tried.

After explaining how to reach the values given as an example in Table 2.1, it is possible to answer the question of how to deal with situations where the number of states and actions are high. Since only tables or linear function approximators can be used for the approximation of the q function, Q-learning and most other reinforcement learning algorithms have been used in limited situations. The Deep Q-Network (DQN) algorithm changed this situation by approximating the q function by using a deep neural network (Mnih et al., 2015). By expressing the action-value function with a deep convolutional neural network (Hinton & Salakhutdinov, 2006; Krizhevsky, Sutskever, & Hinton, 2017), the q function becomes dependent not only on state and action but also on the network parameters used for approximation and is expressed as Q(s,a,θ_i). Here, θ signifies the deep convolutional network weights, and i indicates the iteration step.

DQN has succeeded with two critical applications. The first is called experience replay (Lin, 1992). For this method, similar to human memory, the experiences of the agent gathered while exploring the environment are stored. This stored data is shuffled prior to training. This ensures that the input is not corelated during training. Another application that made it possible to approximate the q function with a neural network was periodical updates to target values. Actually, there are two functions here one is the action-value function Q and the other is target action-value function $\hat{Q}$. Updates between these functions are made certain intervals instead of every iteration. This method is aimed to decrease the correlation with the target. DQN is tested using Atari 2600 games. Colored screen shots of the game screen are used as states after being processed by a preprocess routine and the score values provided by the game are used as rewards. From the 49 games tested, in 29 games DQN achieved more than 75 percent of the human scores. (Mnih et al., 2015)

Double DQN (DDQN) (Van Hasselt et al., 2016) algorithm has been obtained by making use of DQN (Mnih et al., 2015) and Double Q-learning (Van Hasselt, 2010) algorithms to achieve better performance. It has been preferred in this study because it provides improvements compared to DQN and is still a Q-learning based method retaining its properties.

The action-value function obtained when following a policy π can be denoted by Q_π. If it is desired to obtain the optimal version of it, Equation (2.5) can be used, and continuing further the optimal policy can be obtained by selecting the action that provides the maximum value for a given state as explained before (Van Hasselt et al., 2016).

$$Q_*(s,a) = max_\pi Q_\pi(s,a) \tag{2.5}$$

As demonstrated several times in the literature (Thrun & Schwartz, 1993; Van Hasselt, 2010; Van Hasselt et al., 2016), Q-learning has the problem of estimating overoptimistic values for the action-value function. The reason for this is simply the use of the action values estimated during the training for both the selection and evaluation processes of an action. DDQN provided an improvement in overestimation by separating the selection of the action and the evaluation of the action. In Equation (2.6) and Equation (2.7) for target values is provided for DQN and DDQN (Van Hasselt et al., 2016). Here it is clearly seen that DDQN not only uses target network parameters θ^- , but also the action value network parameters θ is used. Evaluation of the action is done by the target network whereas action selection is done by the action value network.

$$Y_t^{DDQN} \equiv R_{t+1} + \gamma Q(S_{t+1}, \underset{a}{\operatorname{argmax}}\, Q(S_{t+1}, a; \theta_t), \theta_t^-) \tag{2.6}$$

$$Y_t^{DQN} \equiv R_{t+1} + \max_a Q(S_{t+1}, a; \theta_t^-) \tag{2.7}$$

Starting from the most fundamental definitions a brief overview of reinforcement learning is provided in this section focusing on the algorithms that are relevant for the DDQN method.

CHAPTER 3

PRIOR KNOWLEDGE EXPERIMENTS

3.1 Methodology

In this section, information about the methodology of the experiment will be provided. First, the phenomena that are aimed to be observed in the experiment and the underlying motivation will be explained. Secondly information about the games developed in order to observe chosen phenomena will be provided. Finally, the section will end with information on the way the experiment is conducted, and the profiles of the participants.

3.1.1 Selection of Phenomena to be Observed

A lot of phenomena can be mentioned when it comes to human prior knowledge. Since it was impossible to address all of these in this study, there was a need for selection and simplifying assumptions. The following points were taken into consideration during the selection;

- Effect on game performance
- Applicability to reinforcement learning
- Likely to be observable in most people
- Easily applicable in the developed game environment

The selected phenomena were chosen to meet these criteria at the highest rate, if not completely. The first phenomenon is a preliminary information about the meaning of objects and colors. Trying to navigate to the princess object or the green object seen on the game screen is an example of the effect of this type of information. Another selected phenomenon was chosen as information about the relationship between the keys pressed and the direction. An example of this type of information is the expectation that the character moves to the right when the right arrow key is pressed. The last selected phenomenon is the direction information within the game. The hypothesis here is that the right and up sides are preferred primarily to advance in the game.

The application of selected cases in the game environment is explained in the next section.

3.1.2 Experiment Environment

In order for the experiment to be carried out as desired, the experiment environment must be easily modifiable, allow data recording during the experiment, and provide a suitable environment for participants who are not experienced in playing games. In addition, the experimental environment should be easy to work with algorithms for the studies to be done with reinforcement learning algorithms.

Considering the fact that most of the studies on reinforcement learning were developed in Python, it was decided to develop the experimental environment using Python. Another advantage of the Python programming language is that there are many open source packages. These packages include Pygame and Pygame Learning Environment (Tasfi, 2016) which were used to develop the games employed in this study.

In Figure 3.1 performance of Deep Q-Network algorithm across different Atari games is provided (Mnih et al., 2015). Performance of the algorithm is normalized using random play and human expert scores according to the formula 100*(DQN score - random play score) / (human score - random play score) (Mnih et al., 2015). DQN performed better than the human expert in most of the games but not all. It even got a score of zero in Montezuma's Revenge, which is a sparse reward game where in order to score points certain actions should be done in order. For example, player needs to get a key first, then find the door, and open it with this key to get a certain amount of reward. The environment developed for the present work is based on a simplified version of Montezuma's Revenge. The developed environment does not require the player to accomplish tasks before the goal state for the sake of simplification, but to make it harder for the reinforcement learning algorithms, the game does not assign any rewards. Moreoer, the game is not coded from scratch. Code developed for a similar study (Dubey et al., 2018) is used as a starting point. Several modifications are made to the code and new game maps are designed.

The game was designed after determining the infrastructure. First of all, a default version of the game which is used as a reference and which has not been changed regarding the phenomena to be observed is planned. In addition to the fact that the game does not contain any score, attention was paid for it to be playable by the participants in a five-minute period even if they are not experienced in computer games. The screenshot of the reference version, called the OriginalGame (OG), is shared in Figure 3.2.As can be seen from the figure, the game includes the following elements;

- Princess: Goal of the game. The game ends when the player reaches the princess.

- Fire pit: A trap that causes the game to end.
- Enemy: Contact causes the game to end. When jumped over, it disappears.
- Player
- Ladder

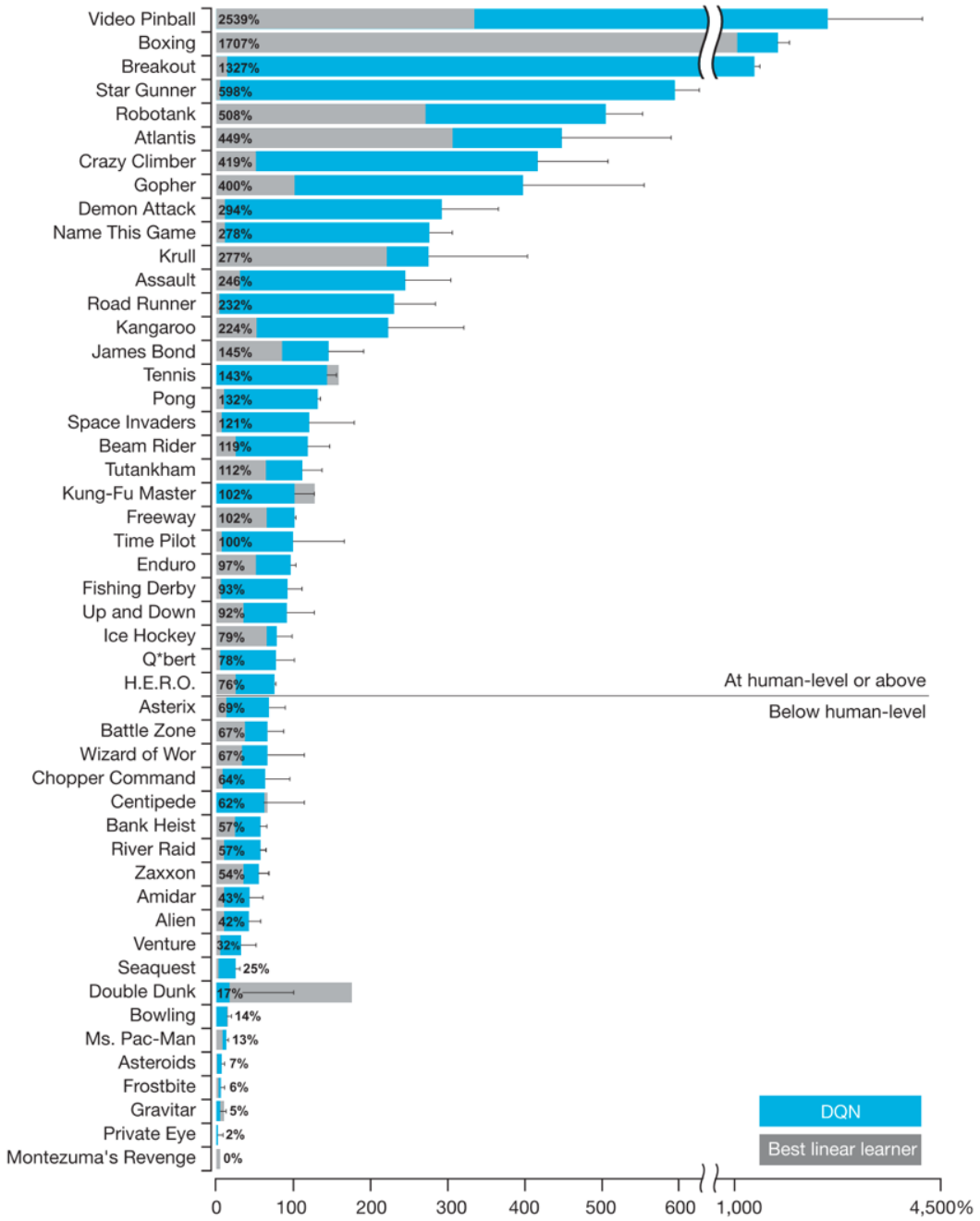


Figure 3.1: *Normalized Game Performance of DQN Algorithms Playing Atari Games (Mnih et al., 2015)*

In order to observe how objects and colors affect perception and game performance, the above-mentioned game elements are masked with colored boxes.

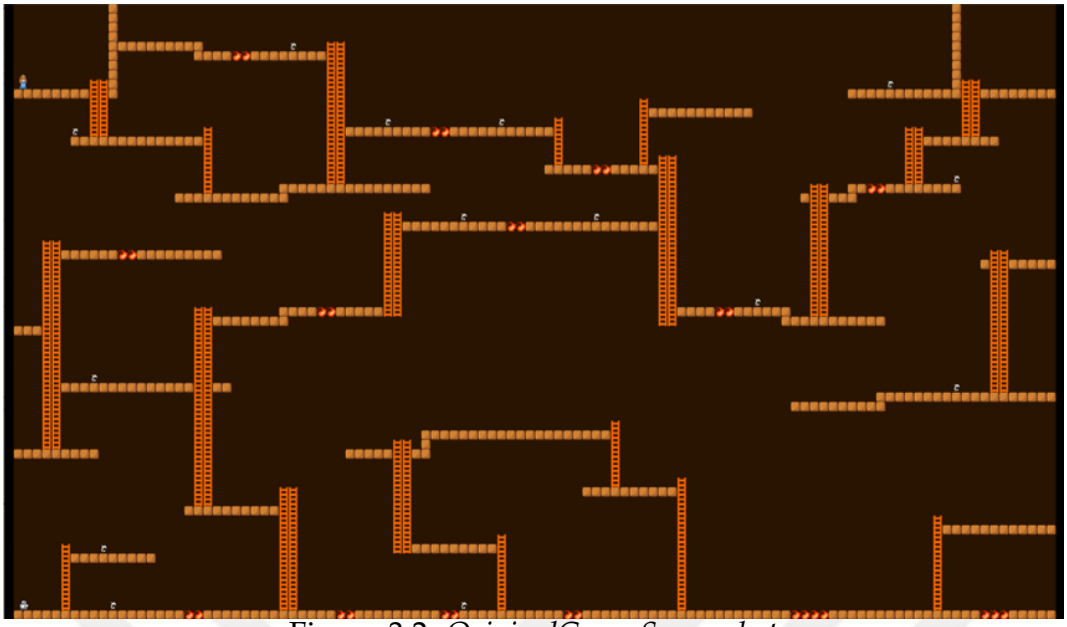


Figure 3.2: *OriginalGame Screenshot*

Two different strategies for masking were followed. In one version, negative objects are masked with negative colors, which is referred as the NosemanticsGame (NG). In another version negative objects are masked with positive colors, which is called the NosemanticsReversedGame (NRG). The player is assigned the same color in both versions. In order to choose colors for the objects, existing findings on the relationship between color and emotion (Naz & Epps, 2004) is used as a guideline, which led to the assignment of following colors to game entities:

- Player: White
- Princess: Green, Black (Reverse version)
- Ladder: Blue, Gray (Reverse version)
- Enemy: Black, Green (Reverse version)
- Fire pit: Gray, Blue (Reverse version)

Another version of the game, called LaddermaskedGame (LG), was created by masking the stairs in the game as a stone wall to see if using colored blocks instead of stairs would affect the participants' performance. The screenshots for these versions are shared in Figure 3.3, Figure 3.4 and Figure 3.5.

Game versions are designed to be played normally with the following keys;

- W: Up
- A: Left
- S: Down
- D: Right

- Space: Jumping

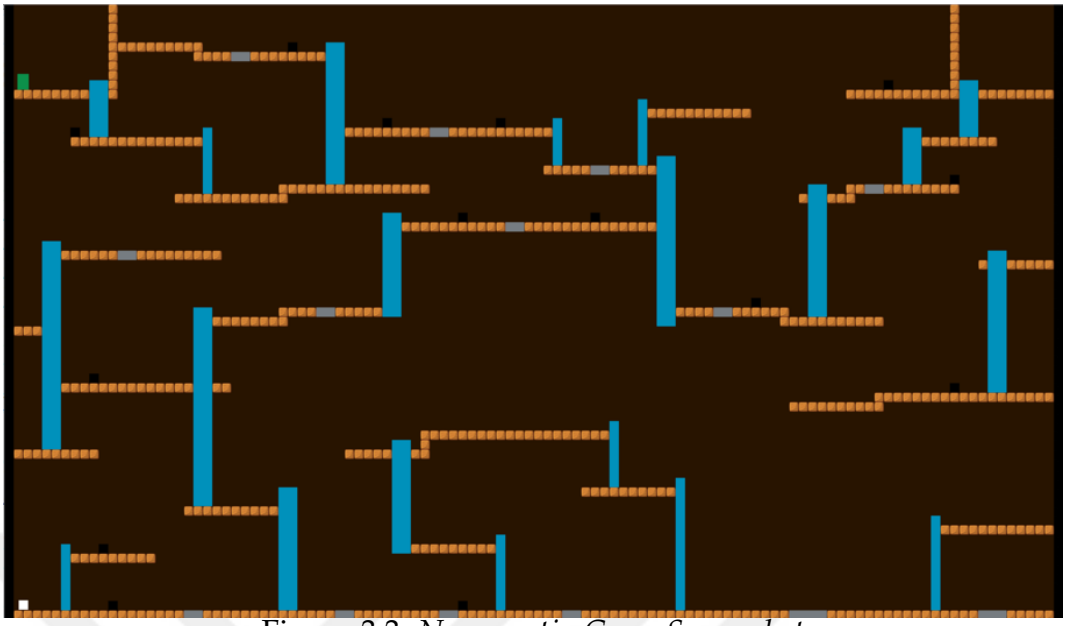


Figure 3.3: *NosemanticsGame Screenshot*

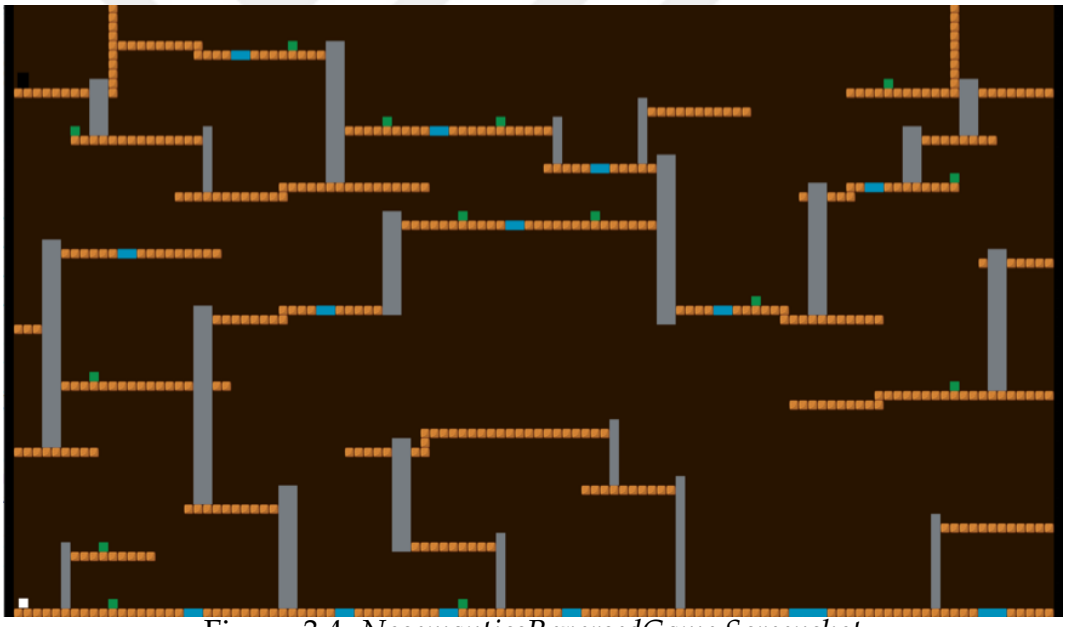


Figure 3.4: *NosemanticsReversedGame Screenshot*

When the positions of the keys are taken into consideration, expectations of the people regarding the functions of the keys are in accordance with the determined directions. In order to observe how the game performance will be affected if these expectations are not met, the keys are changed as given below. Other than the keys nothing is changed compared to the OriginalGame given in Figure 3.2. This version is called ReversedGame (RG).

- W: Down
- A: Right

- S: Up
- D: Left
- Space: Jumping

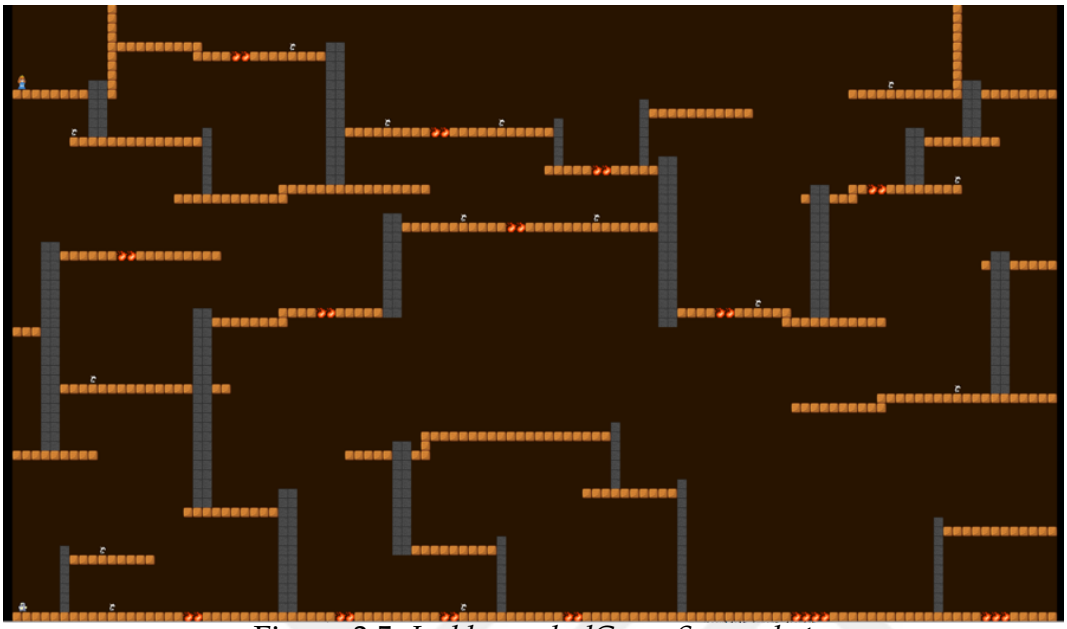


Figure 3.5: *LaddermaskedGame Screenshot*

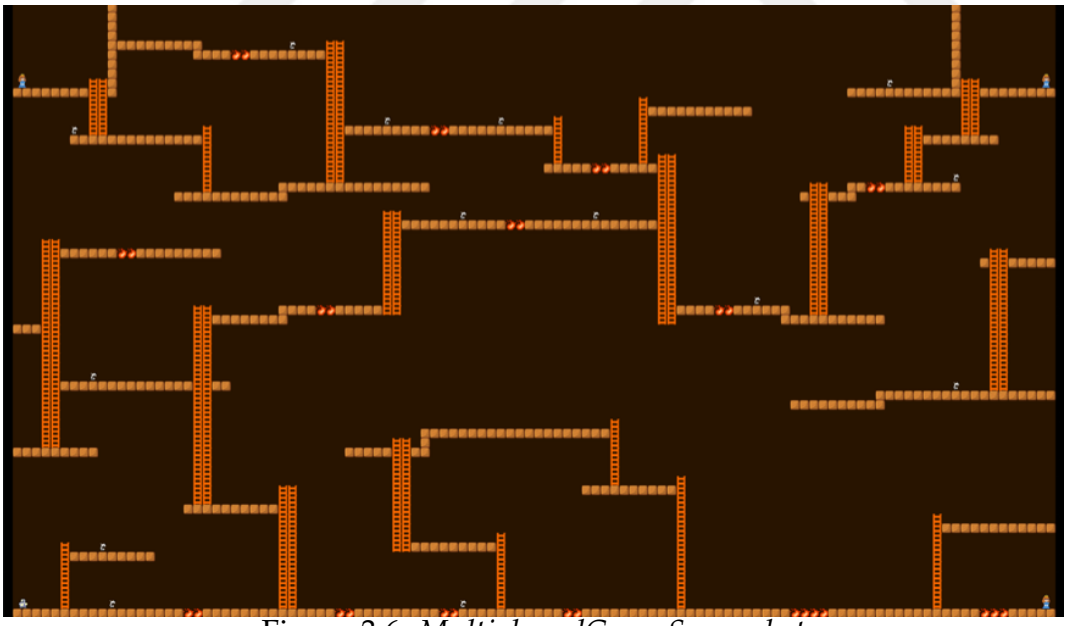


Figure 3.6: *MultiplegoalGame Screenshot*

A game version, named MultiplegoalGame (MG), with three goals, one real and two fake, was created in order to observe in which direction the participants would prefer to go first. The screenshot of the relevant version is shared in Figure 3.6. As can be seen from the figure, one of the targets is placed on the lower right, another is placed on the upper right, and finally, as in the other versions, a target is placed on the upper left. The closest right-hand target of

these targets is positioned so that it cannot be reached. For the upper right and upper left targets, they are intended to be accessible by traveling almost the same distance. To reach the end of the game, as in all other versions of the game, the upper left goal must be reached. Information about the participant profile and the implementation of the experiment is given in the next section.

3.1.3 Participants and Experiment Conduct

The experiments were carried out on a laptop. The participant was briefly informed about the experiment, but only the valid keys of the game were provided regarding gameplay; no information was provided regarding the purpose of the game or the functions of the keys. If the game could not be completed within five minutes, the game was automatically terminated to avoid prolonging the experiment.

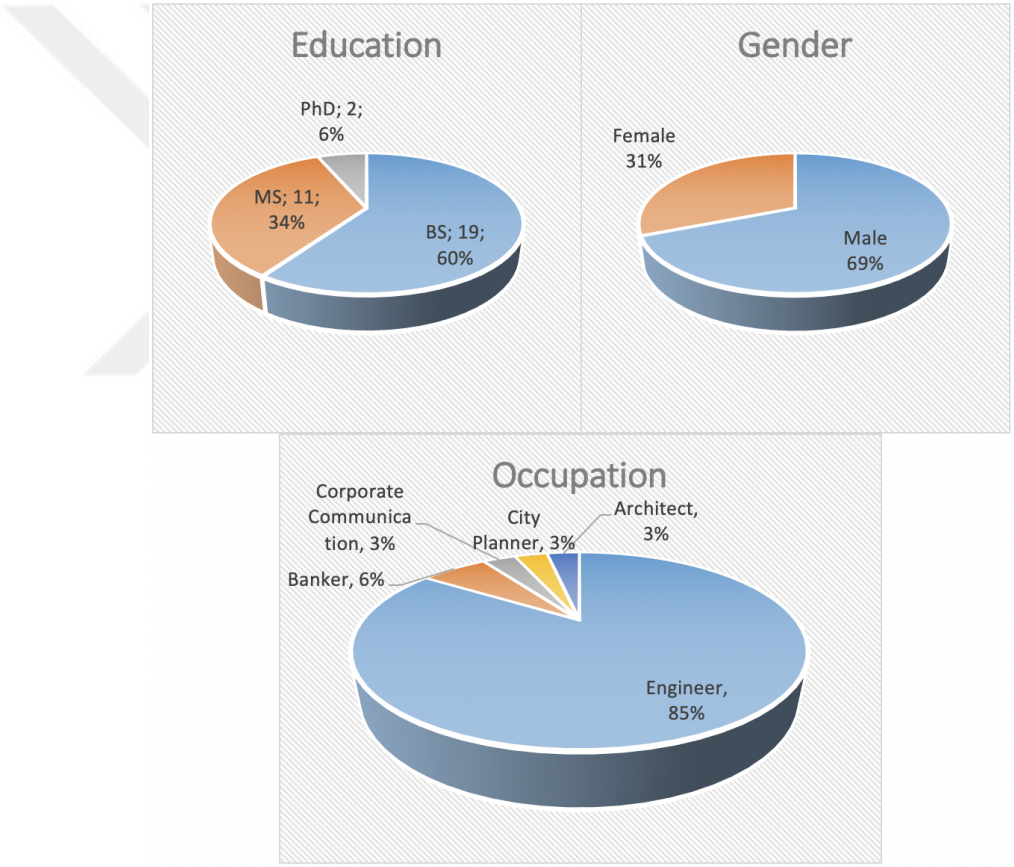


Figure 3.7: *Gender, Education and Occupation Profile of Participants*

Each participant was asked to play all six games, but the order of the games was changed to ensure that each game was played for the first time by an equal number of participants. At the end of the experiment, a short questionnaire was used to collect further information. The questionnaire consists of six questions. In order to obtain general information about the participants, age, gender, education and occupational information were collected. In addi-

tion, they were asked to rate their interest in computer games out of ten, and indicate the hardest of the six games they played. Graphs showing gender, education and occupation information gathered from the survey are given in Figure 3.7.

Average age of the participants was 29.6, youngest being 23 and oldest being 41 years old. As can be observed from Figure 3.7, majority of the participants were male (69%). With 19 out of 32 people, most of the participants were holding a BSc degree and the majority of the participants were engineers. The average interest of the participants in computer games was 6.19. 22 of the 32 participants indicated that the first game they played was the most difficult. This can be considered as an expected result considering that they can use what they have learned in the first game in other games.

3.2 Results

As mentioned in the previous section the experiment is conducted with 32 participants where six different games are played by each participant with different orders to ensure that each game is played for the first time by nearly equal number of participants. Every key press of the participant was logged making it easier to generate necessary data during post processing. Main parameters generated from the gameplay data were the following;

- Finishing time: The time spent by the participant to finish the game.
- Number of deaths: The number of deaths of the participant due to the enemy or fire pit until the game is over.
- Number of unique states: The number of unique positions the participant walked through until he/she finished the game.
- Success: If the participant reached the princess in less than five minutes, game is deed successful. If time limit is passed the game is unsuccessful.

The distribution of successes and failures across game types is summarized in Figure 3.8. Although the number of failures were slightly higher in the case of No Semantics and the Ladder Masked game conditions, a chi-square test conducted on the frequency distribution of success and failure cases did not find a significant difference, $X^2(5) = 2.10, p > 0.05$.

The boxplots, given in Figure 3.9, summarize the observed game finish times. The normality tests indicated significant positive skew in all conditions, so a non-parametric Friedman's ANOVA test was conducted to compare game type in terms of their completion times. Although the Reversed, Multiple Goals and No Semantics conditions had higher completion times, Friedman's ANOVA test result showed that these differences were not statistically significant, $X^2(5) = 5.73, p > 0.05$. Detailed statistical information is provided in Table 3.1 and mean rank is provided in Table 3.2.

Table 3.1: Descriptive Statistics for Finishing Time

						Percentiles		
Game	**N**	**Mean**	**Std. Deviation**	**Min.**	**Max.**	**25th**	**Median**	**75th**
OG	32	137.09	96.80	44.64	300	61.20	85.10	215.21
RG	32	150.44	89.45	55.19	300	67.38	126.65	233.47
MG	32	155.78	92.02	47.15	300	90.38	122.57	253.88
NG	32	159.31	92.33	53.35	300	79.94	118.81	278.80
NRG	32	131.81	85.96	46.21	300	65.48	101.30	182.77
LG	32	141.32	99.07	48.13	300	61.80	92.55	265.66

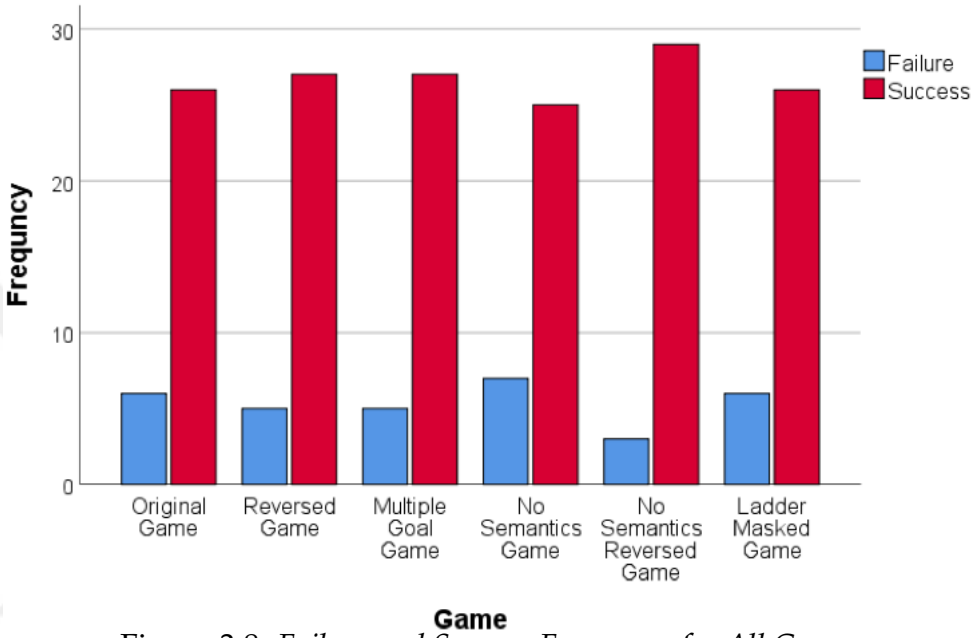


Figure 3.8: *Failure and Success Frequency for All Games*

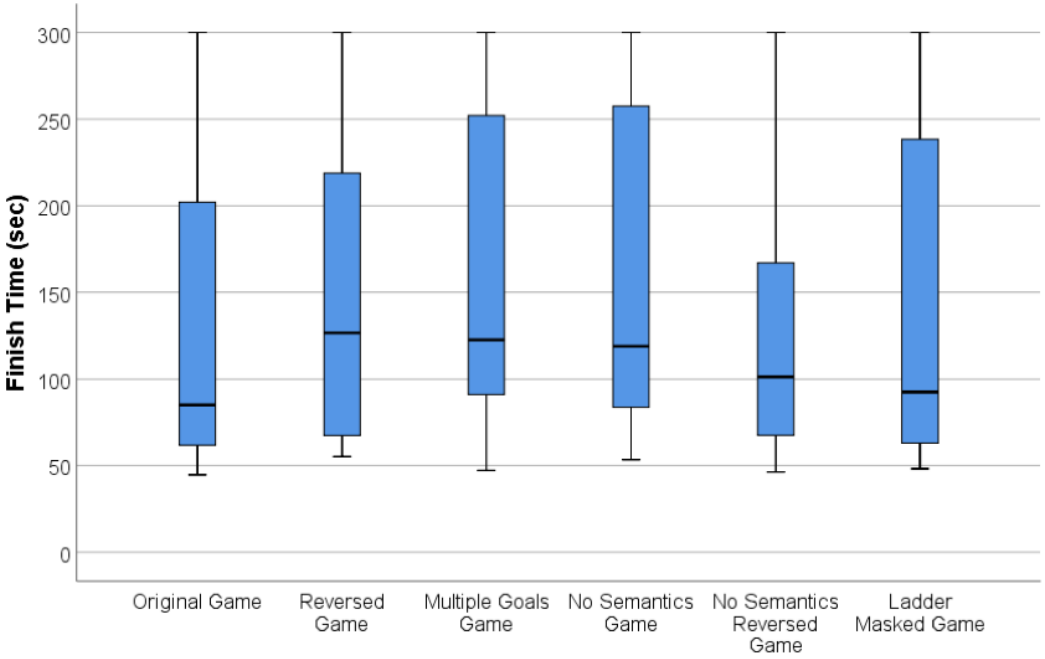


Figure 3.9: *Boxplots for Finish Times*

Table 3.2: Mean Rank for Finishing Time

Game	Mean Rank
OG	3.11
RG	3.73
MG	3.75
NG	3.92
NRG	3.17
LG	3.31

The boxplots, given in Figure 3.10, summarize the distribution of unique states the player ran into during each game condition. A parametric repeated measures ANOVA was conducted to test for difference across game conditions since the distributions were approximately normal for each game. Although the Multiple Goals and Reversed conditions had higher mean unique states, the ANOVA test did not detect any significant differences across game types, $F(5, 155) = 0.34, p > 0.05$.

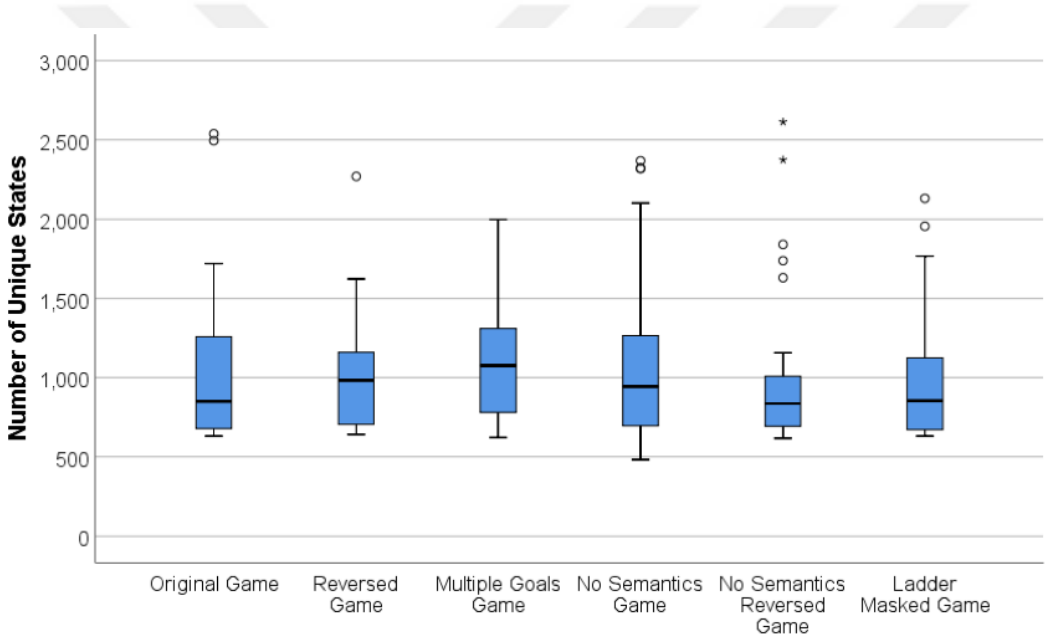


Figure 3.10: *Boxplots for Number of Unique States*

The boxplots, given in Figure 3.11, summarize the number of times players died (and restarted) during the game. The normality tests indicated significant positive skew in all conditions, so a non-parametric Friedman's ANOVA test was conducted to compare game type in terms of their completion times. Friedman's ANOVA test found no significant differences between the game conditions, $X^2(5) = 6.94, p > 0.05$. Detailed statistical information is provided in Table 3.4 and mean rank is provided in Table 3.5.

In Table 3.6, the average values of the examined parameters are given. The average values are given both for all participants played the game and only for those who played that game as the first. The main reason for this was that during the experiments it was observed that the participants had less difficulty in other versions of the game after playing the first game.

Table 3.3: Tests of Within-Subjects Effects

Source		Type III Sum of Squares	df	Mean Square	F	Sig.	Partial Eta Squared
Game	Sphericity Assumed	295723.19	5.00	59144.64	0.34	0.89	0.01
	Greenhouse-Geisser	295723.19	4.40	67243.20	0.34	0.87	0.01
	Huynh-Feldt	295723.19	5.0	59144.64	0.34	0.81	0.01
	Lower-bound	295723.19	1.00	295723.19	0.34	0.57	0.01
Error (Game)	Sphericity Assumed	27349662.48	155.00	176449.44			
	Greenhouse-Geisser	27349662.48	136.33	200610.31			
	Huynh-Feldt	27349662.48	155.00	176449.44			
	Lower-bound	27349662.48	31.00	882247.18			

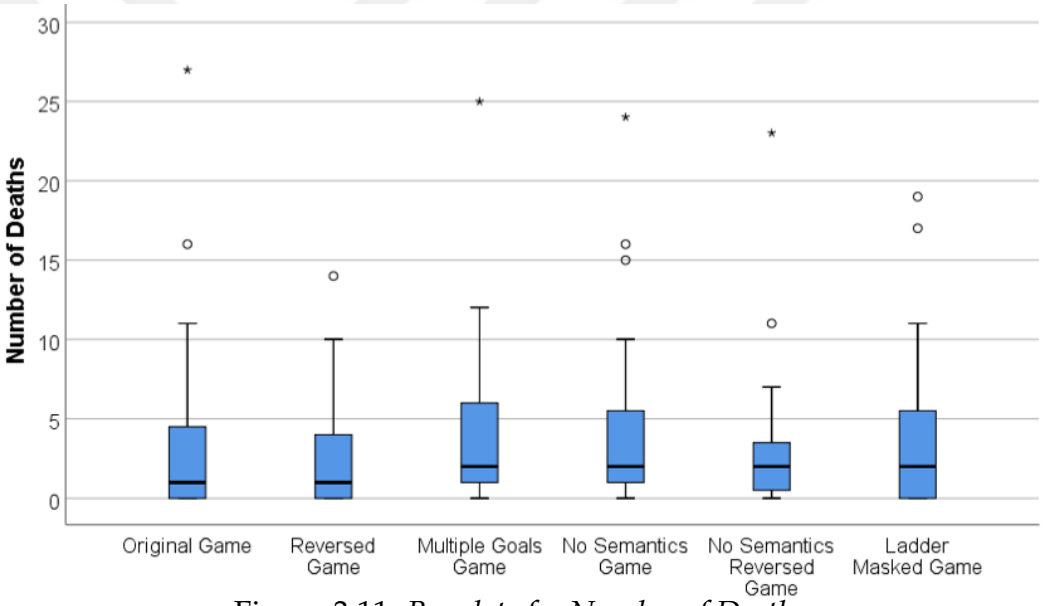


Figure 3.11: *Boxplots for Number of Deaths*

Table 3.4: Descriptive Statistics for Number of Deaths

						Percentiles		
Game	N	Mean	Std. Deviation	Min.	Max.	25th	Median	75th
OG	32	3.53	5.71	0.00	27	0.00	1.00	4.75
RG	32	2.75	3.91	0.00	14	0.00	1.00	5.00
MG	32	4.12	5.17	0.00	25	1.00	2.00	6.00
NG	32	4.41	5.49	0.00	24	1.00	2.00	5.75
NRG	32	2.97	4.40	0.00	23	0.25	2.00	3.75
LG	32	3.69	4.95	0.00	19	0.00	2.00	5.75

Considering the average completion time for the players playing the game for the first time, it is seen that OG is completed in the least amount of time. This is expected since it is the easiest game. Another expectation was NRG to

Table 3.5: Mean Rank for Number of Deaths

Game	Mean Rank
OG	3.38
RG	3.00
MG	3.98
NG	3.89
NRG	3.39
LG	3.36

be the game taking longest time to finish. The reason for this is that, in addition to masking the meaning of the objects, the participants are misdirected in a sense because colors are used in contrast to the meaning of the objects. NRG is also the game which has the lowest success rate when played first. When the average number of unique states is evaluated for the first timers, it can be observed that there is more exploration during NRG play. The main reason for this is that some participants think that enemies are something that should be gathered because the enemies are green and disappear when they jump onto them. In the light of the data in Table 3.6, it can be said that prior knowledge has a definite effect on game performance. This may be in contrast with the ANOVA analysis presented before. The problem here is that most of the participants learned the general structure of the games in their first game. As a result, games played after the first one ended in a similar fashion although these were different versions. For a single game, the number of other players is higher than the ones played the game first. For this reason, prior knowledge effect cannot be observed as expected in ANOVA analyzes.

Table 3.6: Average Values of Collected Parameters for Each Game

Games	Avg. Time	Avg. Time (First)	Avg. Unique States	Avg. Unique States (First)	Avg. # of Deaths	Avg. # of Deaths (First)	Success Rate	Success Rate (First)
OG	137.09	187.51	1068.22	1425.00	3.53	6.80	81%	60%
RG	150.44	226.26	1010.22	1362.80	2.75	7.00	84%	100%
MG	155.78	253.74	1079.00	1281.83	4.13	11.17	84%	50%
NG	159.31	255.43	1102.13	1333.83	4.41	12.67	78%	67%
NRG	131.81	261.22	1002.31	1670.60	2.97	9.60	91%	40%
LG	141.32	233.20	1009.63	1629.40	3.69	10.00	81%	60%

It can be assumed that some values given in Table 3.6 are contrary to the assessment of the effect of prior knowledge. For example, as expected, with regard to data of OriginalGame first timers, the game is completed in a shorter period of time, while NosemanticsReversedGame appears to be completed in a shorter period of time when all participants are evaluated. There are two main reasons for this and similar inconsistencies. The first is due to the averaging operation. A small number of large values (outliers) that can be added between many small values will increase the average value, but this will not actually provide accurate information in evaluating all data. For this reason consider boxplots given in Figure 3.9, Figure 3.10 and Figure 3.11 for the completion time, the number of unique states and the number of deaths. From

the finishing time graph, it can be seen that the OriginalGame finishing time varies over a much wider range than the NosemanticsreversedGame though the median value is smaller for OG. The second reason is the participants' interest in computer games. Table 3.7 shows the averages of the participants' interest in the computer games stated in the survey. When the data is analyzed, it is seen that the average of the OG and NG first timers is below all the participants and the average of the first RG, NRG and LG players is high. This type of imbalance also has an effect on the results.

Table 3.7: Average Gaming Interest

	Avg. Gaming Interest
All Participants	6.2
First OG	4.6
First RG	7.2
First MG	6.5
First NG	4.7
First NRG	7.0
First LG	7.4

As mentioned earlier, it was observed that the order of playing of the games had a significant effect on the performance. In order to better observe this point,Table 3.8 gives the average completion time of the games and average number of deaths for all participants, participants played the game first and last. Only OG, NG and NRG have participants playing them as the last game thus in the table only information collected from these games are provided.

Table 3.8: Average Values for All Participants, Participants Playing the Game First and Last

Games	**Avg. Time**	**Avg. Time (First)**	**Avg. Time (Last)**	**Avg. # of Deaths**	**Avg. # of Deaths (First)**	**Avg. # of Deaths (Last)**
OG	137.09	187.51	96.72	3.53	6.80	1.31
NG	159.31	255.43	98.28	4.41	12.67	1.50
NRG	131.81	261.22	84.10	2.97	9.60	1.45

As can be seen from the Table 3.8, there is a big difference between the results if the game is played first or last. Changing the playing order from first to last results in a shortening of the game completion time by 48% for OG, 62% for NG and 68% for NRG. A similar improvement is observed for the average number of deaths. Especially for NG there is a great improvement of 88%. All these data show the importance of the order of the games.Figure 3.12 shows the number of times each game was played. This also demonstrates the ability of people to use what they have learned in a short time then quickly adapt to changing keys. In addition, after learning and having to jump through the fire pit or the enemy in the first place, people did not try to learn them again, as expected.

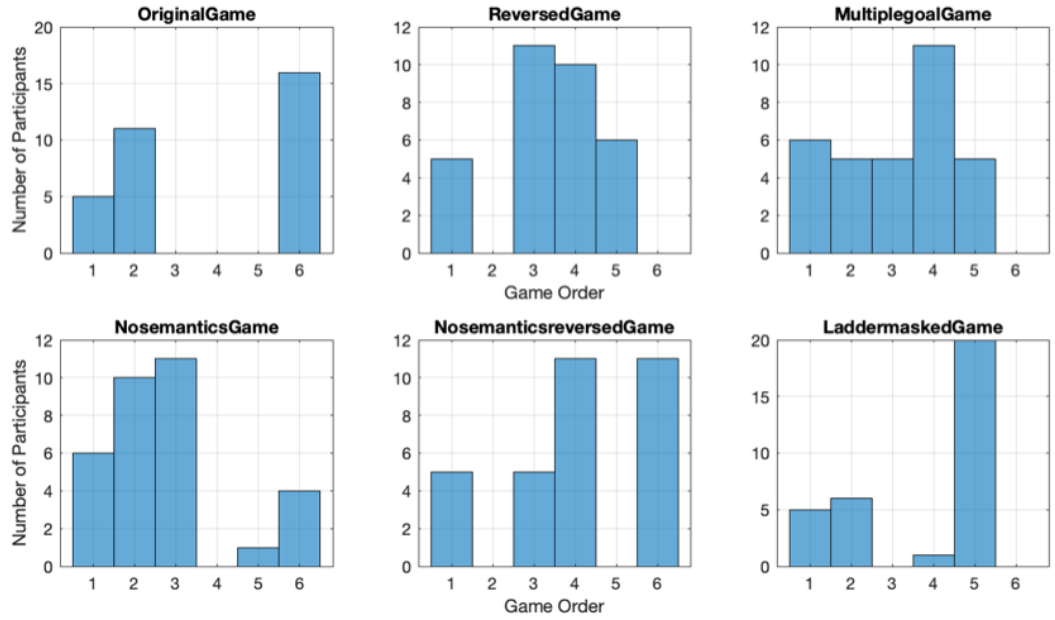
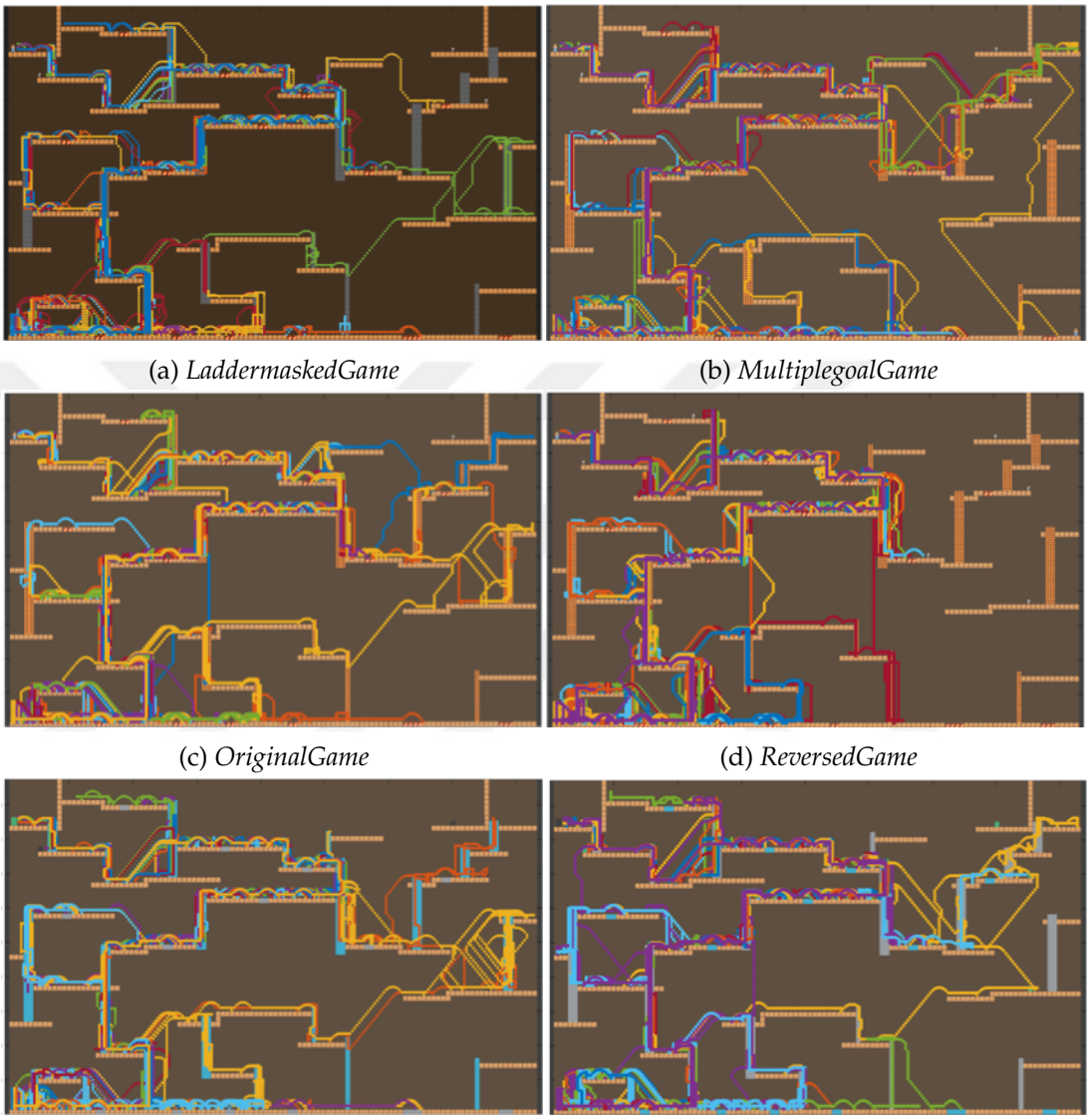


Figure 3.12: *Playing Order of Games*

Finally, for all games, how participants discover the game will be examined. For this purpose, the routes that all participants have followed for all the games are shown on the game screenshot in Figure 3.13. Paths followed tended to diverge less for the case of ReversedGame. Moreover, there are more failed attempts when climbing ladders due to the reversed keys. Except for the ReversedGame, it is clear that for all games some participants tried to go right at some point instead of going towards the goal. Although it was designed to be unreachable, even one participant found a bug and reached lower right princess in the MultiplegoalGame. Apart from these observations nothing concrete regarding human exploration strategy can be found from these charts. Further investigation is needed.

In addition to the numerical data examined, the participants' comments and observations made during the experiment are also crucial for the interpretation of the results. Initially, some of the participants who started with NosemanticsReversedGame thought that the enemies represented by the green box had to be gathered and tried to collect the green boxes instead of going to the princess, which was represented by a black box. Similar behavior was not observed for the participants starting with the NosemanticsGame. Participants playing the LaddermaskedGame first tried to go up later as compared to other games. When asked why they tried to go up, although stairs were wall-like structures, participants explained that there was no other way to go; therefore, they tried to go up. The participants first playing the MultiplegoalGame have seen all the targets, but most of those who have not played the game in the first place were not aware of the newly added targets, and they went to the target that they were already familiar with from the previous games. Most of the participants who saw all three goals in the game of MultiplegoalGame stated that they planned to go to the bottom right target because it was the closest but did not go because they considered it unreachable. The first players to play NosemanticsGame directly went to the princess, which is represented by a green box. When asked for the reason, they stated that it was

the only green object besides being green. This shows that apart from having a positive color, number of occurrences also have a relevance. Some participants did not notice the princess, or the object placed instead of a princess and tried to move up and to the right, thinking that the game would progress. As a result, it can be concluded that at least some participants have a prior that going right and up will make progress within the game.



(a) *LaddermaskedGame* (b) *MultiplegoalGame*

(c) *OriginalGame* (d) *ReversedGame*

(e) *NosemanticsGame* (f) *NosemanticsReversedGame*

Figure 3.13: *Paths Taken by All Participants for All Games*

CHAPTER 4

REINFORCEMENT LEARNING

In this section, the performance of vanilla DDQN algorithm will be presented. Afterwards, the improvement attempts using prior knowledge will be provided and the results of these experiments will be shared. Implementation of DDQN is done in Python language and Keras (Chollet, 2015) with Tensorflow (GoogleResearch, 2015) is used for training. Different network parameters, state representations and reward structures are used for the experiments. These parameters along with the aims and results of the experiments are provided in the subsequent sections.

4.1 DDQN Game Performance

In order to observe the game performance of the vanilla DDQN algorithm, different state and rewards were used. In the first attempt, the game screen was used as the state representation. In order to minimize the number of network parameters, the image was first converted to grayscale, and then a 201x201 pixel area was cut around the player. Similar to the original DDQN application, four consecutive frames were buffered and used as input. Thus, the network input was 4x201x201x1. The network structure used is given in Figure 4.1. The optimization employed to train the network is Adam (Kingma & Ba, 2015). The discount parameter was set to $\gamma = 0.90$, and the learning rate was 0.0001. The number of steps between target network updates was 2000. The size of the experience replay was 20000 and it got sampled to update the network every 4 steps with minibatches of size 20. ϵ-greedy policy was used with the ϵ starting from 1.0 dropping down to 0.05. Reward is constructed proportional to the change of state. As the only moving entity on the screen is the player, change between frames is only due to its movement. An episode ends if a terminal state is reached or after 300 seconds of gameplay. Reaching princess and death by falling into a firepit or colliding with an enemy are the terminal states. If the agent reaches a terminal state or time limit is reached game is restarted. If the terminal state reached is princess a positive reward otherwise a negative one is assigned. These rewards assigned are scaled such that they are magnitude wise comparatively larger than the rewards assigned due to agent movement.

The agent was unable to reach the princess at the end of the training. In addition to not reaching the princess, it did not learn any meaningful policy. It was observed that the agent was constantly jumping to collect reward as jumping all the time was also causing a difference between frames. A second trial is done changing the state representation. This time the whole screen was used as the state. In order to keep the number of parameters under control, image is scaled down by a factor of 5. Changing the state representation did not change the end result as the agent did not learn any meaningful policy.

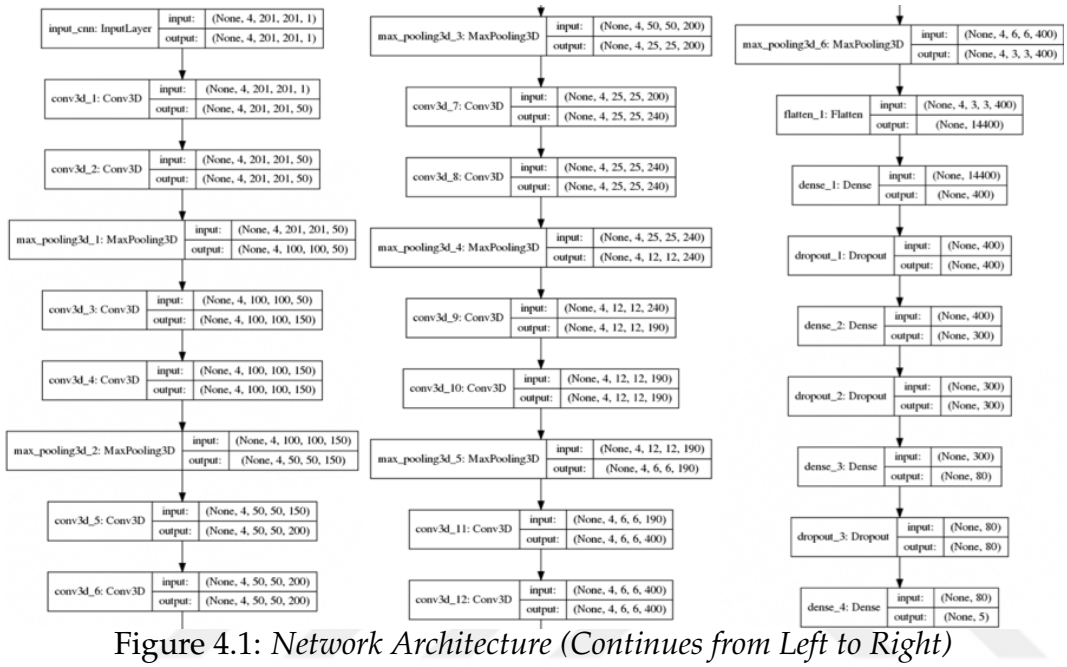


Figure 4.1: *Network Architecture (Continues from Left to Right)*

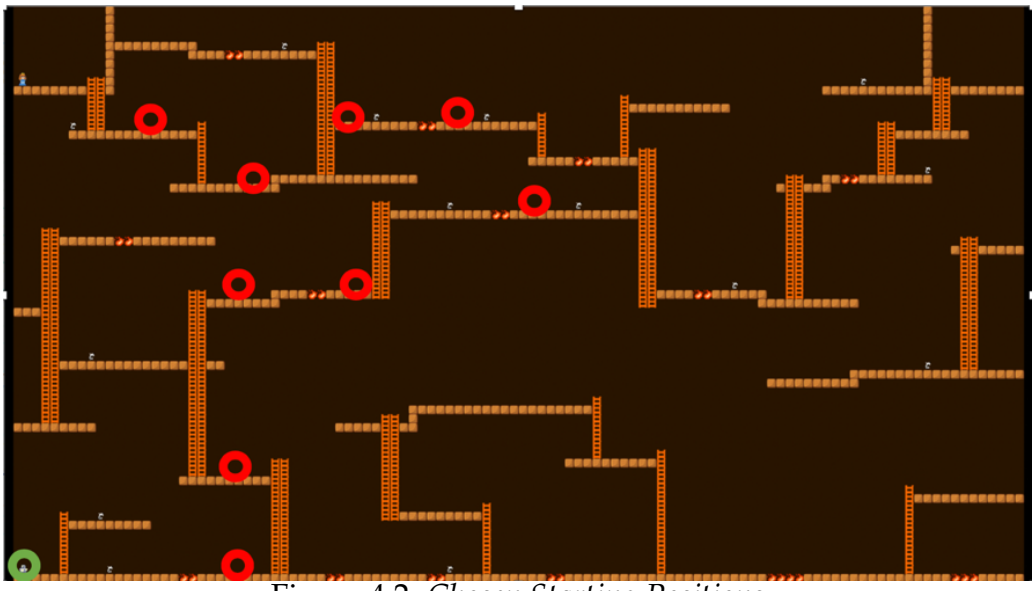


Figure 4.2: *Chosen Starting Positions*

Finally, in order to increase the states visited by the agent random starting positions was tried. Although it was optimal to use every state in the game as a potential starting position for the agent, this could not be done as this could

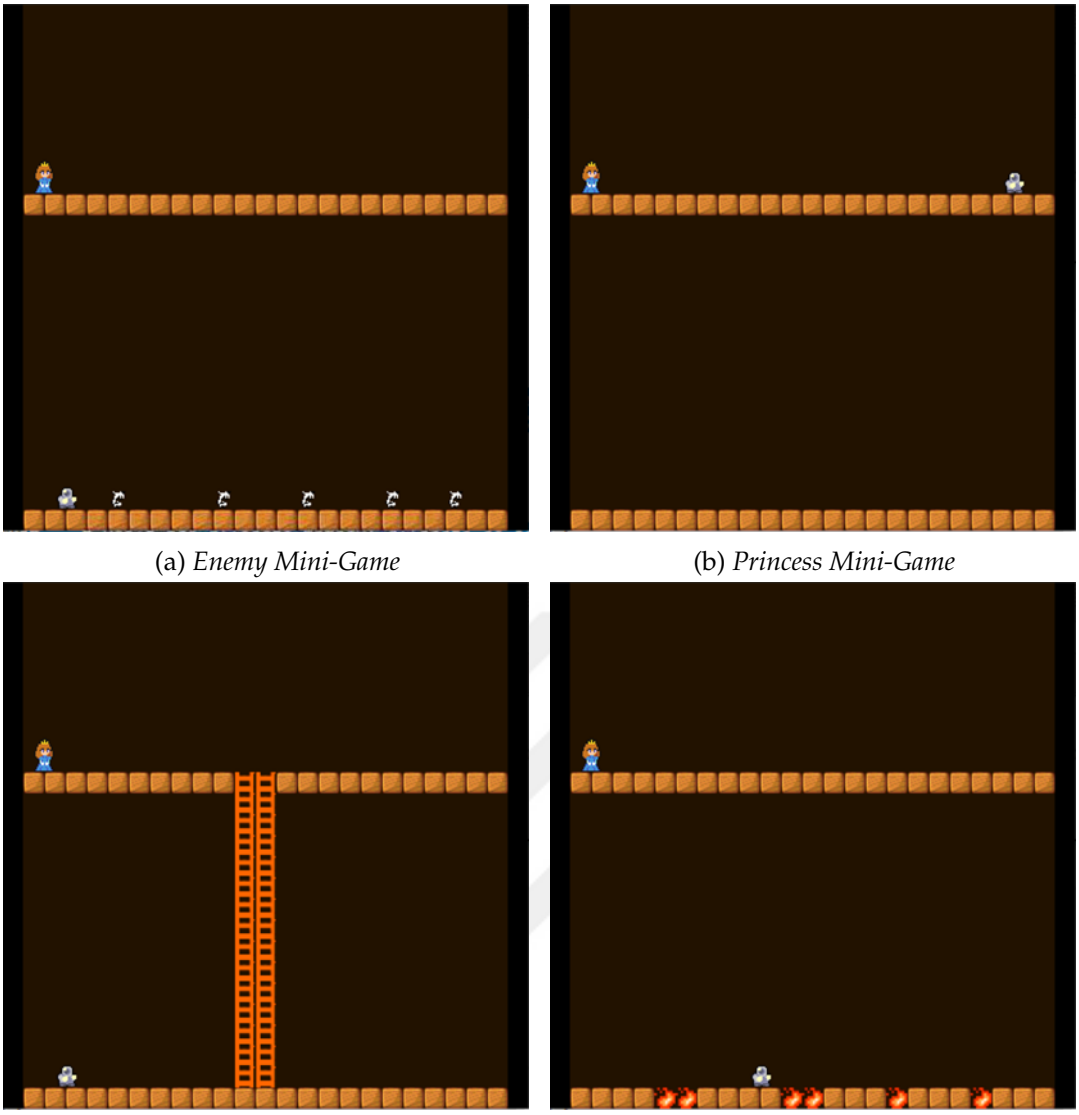
result in agent starting inside a wall or on a firepit. Due to this reason ten locations were determined, and these are randomly chosen. These locations are shown on Figure 4.2, green one being the default location used. All the locations chosen are on the path to the desired terminal state, the princess. This modification was tried with both of the state representations.

Making starting positions random did not make a difference either. This result is expected as DDQN is not built for cases where there is no reward from the environment. Trying to come up with a simple reward, that can be regarded as an intrinsic reward, ended in failure. There are algorithms that claim to perform well in environments without reward (Bellemare et al., 2016; Ostrovski, Bellemare, Van Den Oord, & Munos, 2017; Pathak et al., 2017; Tang et al., 2017), but these require extensive modification thus cannot be used to measure vanilla DDQN performance. In the following section modifications done to the algorithm in order to incorporate prior knowledge is documented with respective results.

4.2 DDQN Modifications

Using the compositionality concept (Lake et al., 2017) the original game environment is broken down to four more simple tasks each regarding a different entity in the game. These are ladder, pitfall, enemy and princess. For all these tasks, more simple environments are constructed which are given in Figure 4.3. Using the same network parameters, aside from the input layer, as in the previous section these four mini-games are used for training. The idea here was to use "flatten layer" outputs of the networks, concatenate them and use as the state representation for a network that would learn the original game in the end. One can say that the learned policies in these four mini-games are like prior knowledge for the network that is trained in the original game. For each mini-game reward is constructed in accordance with the task. For the enemy and pitfall tasks, dying as a result of colliding with the enemy resulted in a negative reward, whereas passing them successfully resulted in a positive reward. Episodes are ended when the agent died, or timeout is reached. For the princess task reaching the princess resulted in a positive reward that is discounted by a factor proportional to the time it took for the agent to reach it. Again, an episode is ended when the princess is reached, or the allotted time has expired. The Ladder case turned out to be trickier. A positive reward is given when the agent used the ladder for going up or down, and the episode was ended with a timeout.

Mini-games other than the ladder were learned by the agent successfully. These three were used instead of the originally planned four as state generators. The same reward functions as used in the previous section were tried, but the network failed to successfully play the game. Finally, the starting location of the player is made random to increase the states explored, although the starting position was fixed during the experiment given in Chapter 3. Nevertheless, the network still failed to finish the game.



(a) *Enemy Mini-Game* (b) *Princess Mini-Game*

(c) *Ladder Mini-Game* (d) *Pitfall Mini-Game*

Figure 4.3: *Mini-Game Environments*

A second set of trials were done using a different point of view. This time a network was constructed using the data collected during the experiments. Images were taken from the game using the keys pressed by the participants gathered during the experiment, and a network was trained to receive these images as inputs and output the keys pressed. Each key used in the game can only take the values 0 and 1, and because of this the problem could be treated as a classification problem. After it was observed that the trained network successfully completed the game, ways to use this network with reinforcement learning were evaluated since this kind of approach is not a reinforcement learning method.

In order to assess ways to use this network with reinforcement learning, a simplified version of the original problem is formed by cutting down the number of actions down to one, jumping. In this version the outputs of the other four actions are provided by the network trained with human data and the reinforcement agent only expected to learn jumping over pitfalls and ene-

mies. In addition to these, the agent also needed to jump at a point on the map where the player gets stuck due to a step shape shown on Figure 4.4. As CNN guaranteed the agent to be on the correct path to princess only an addition to the original reward structure is made; continuous jumping is penalized for agent to learn jumping at correct locations. For the state representation, a scaled down version of the whole game screen is used. Network provided previously is only changed to have only one output, whereas the other training parameters are kept the same. The agent successfully learned to jump and as a result reached the goal state.

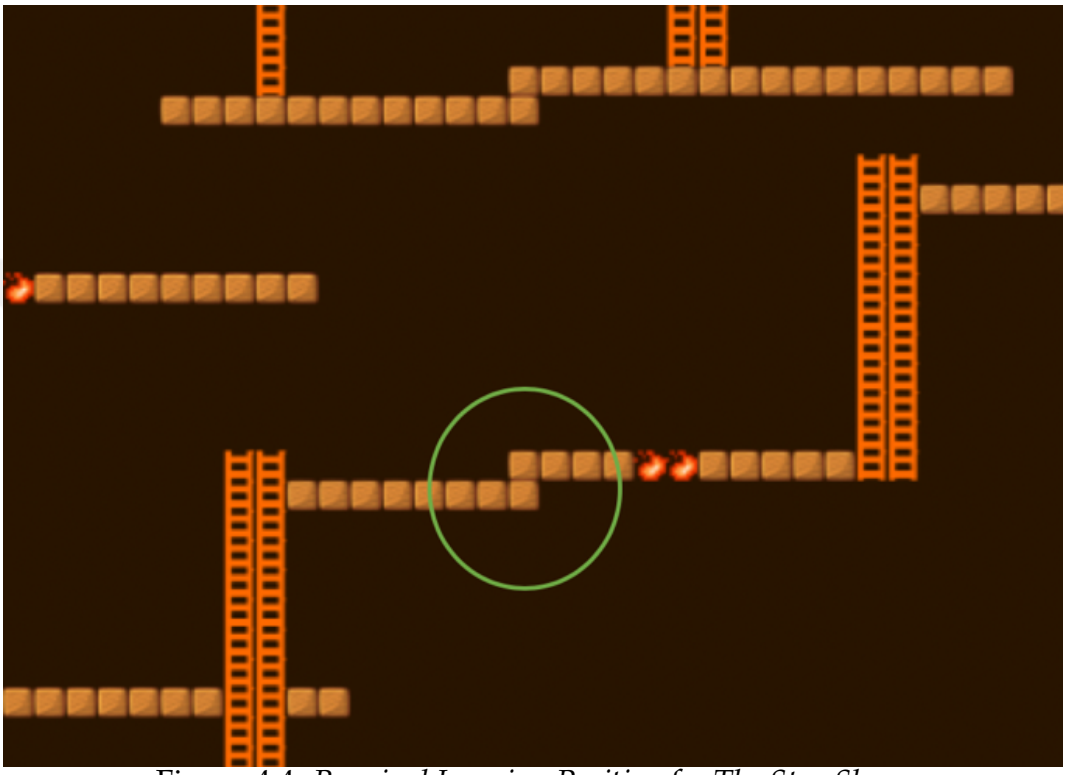


Figure 4.4: *Required Jumping Position for The Step Shape*

This simplified problem showed that with correct guidance the agent could learn a correct policy while keeping the original reward structure more or less the same. How to implement this strategy while keeping all the actions was the challenge at this point. The answer was that the obtained network could be used instead of the target network in the DDQN workflow. Normally updates between the target network and the q-network are done periodically in vanilla DDQN. In contrast to this approach, updates between the target network and the q-network were not done until the performance of the q-network exceeded the performance of the target network which was trained with human data for this trial. Better performance is decided using the completion time of the game which was the time it took the agent to reach the princess.

The algorithm changed in this way succeeded in finishing the game although there were some problems. First of all, as mentioned before, people playing the game directly tried to reach the princess, which means some states of the game was not explored. Consequently, the target network trained using

human data failed to complete the game if anything different is encountered. The difference could be a change in the starting position or a change in the map. This was most dramatically observed when the agent was intended to be tested in an easier setting. All the enemies and pitfalls were removed as shown in Figure 4.5. The network trained just ran by the ladder it should have climbed. When all the entities removed were put back, the trained network performed well and reached the princess as expected. This clearly is a sign of limited generalization performance of the agent. Although the agent was more robust to small changes, still there were limitations. The generalization problem may be tackled with more data generation by human demonstrators, but this would not be an efficient solution. The other problem was the lack of human-like behaviors. The agent finished the game perfectly sticking to the rules which was not the case for the human players. For example, human players after some point started to jump from ladders as a shortcut to cover a longer distance in shorter time. Parts of the data containing movements like these were filtered out for the target-network in order to see whether or not the reinforcement agent would show such behavior. Unfortunately, this was not the case.

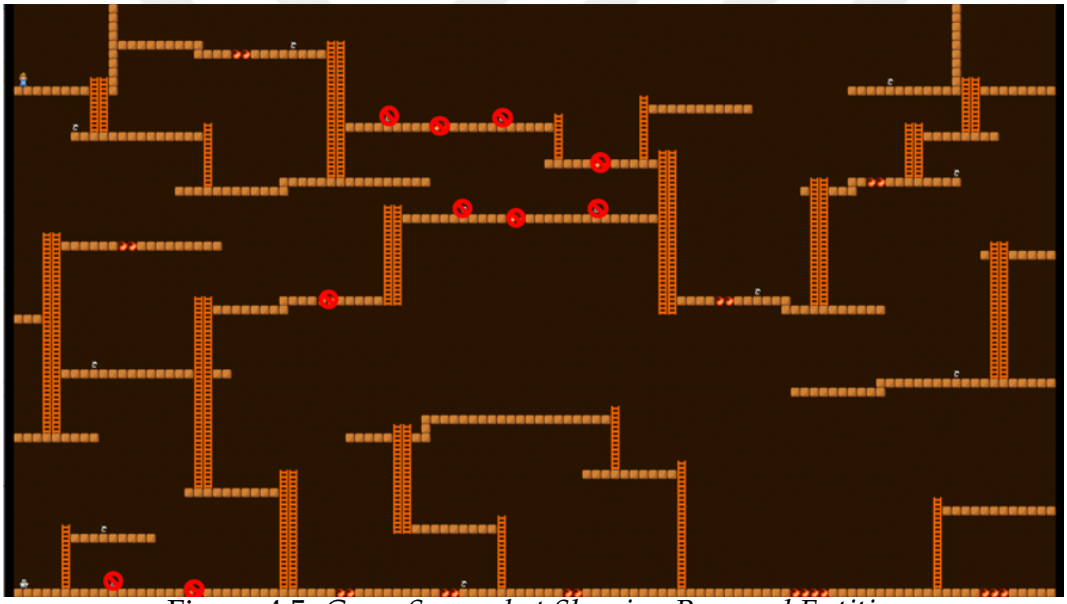


Figure 4.5: *Game Screenshot Showing Removed Entities*

The network architecture used in all the trials was nearly the same. As the state representations changed, the number of neurons in the input layers were adjusted accordingly. Two reasons can be mentioned for using the same network. One of the reasons was that the network architecture given in Figure 4.1 was optimized using a genetic algorithm code and successfully used in other reinforcement learning problems, making it a better candidate than a network constructed from scratch. The second reason was to make it easier to evaluate the modifications done by the algorithm. Even if the same network architecture was used, there is still an effect due to the architecture, but this effect is minimized by keeping it the same across different trials.

In order to observe what's happening to the input image passing through

different layers of the network, the output of the filters from three different layers is given in Figure 4.6. Here it is clearly seen that moving deeper, the outputs get more different than the game screen. Most of the information is lost mainly due to the pooling layers. Each filter certainly captured some information here, but it is not easy to say what is actually done by the network. This may be the biggest disadvantage of using a machine learning method as inner workings of the network is not transparent. If this was, for example, a rule-based architecture, selected rules could be evaluated, giving crucial data regarding how the task was actually done by the algorithm.

(a) *Layer 1* (b) *Layer 2* (c) *Layer 3*

Figure 4.6: *Filter Outputs for Different Layers*

CHAPTER 5

CONCLUSION AND DISCUSSION

Recently the popularity of reinforcement learning increased dramatically. Various algorithms are being developed frequently, both improving performance and making reinforcement learning applicable in different areas. Despite all this research activity, some points do not fully reach a solution. In addition, the similarity between human learning and reinforcement learning results attract interest from diverse areas of expertise. Within the scope of this study, in accordance with this diversity, the reasons underlying better performance of humans, observed by experiment done, were sought. At the same time, it was aimed to increase the reinforcement learning performance and to obtain agents showing human-like behaviors. In the previous chapters, relevant literature, work done, and results obtained are presented.

Although the definition of prior knowledge has a very broad meaning, it is clear that in most cases, people do not learn everything from scratch and use the knowledge they had previously to accomplish tasks. During the experiment, although none of the participants were told that the princess figure was the target, the participants who saw the princess identified her as a target and tried to reach her. This is not the case only in games. When most people see a baby in someone's arms, they will conclude that there is a kinship or intimacy between the person carrying the baby and the baby. However, there is no difference between the person carrying the baby and the humans all over the world from the perspective of a reinforcement learning algorithm. The relationship between the infant and the person carrying it, as observed in humans, can only be established during training. In the experiment described in Chapter 3, the prior knowledge effect was tried to be observed in a game environment. ANOVA analysis of the results obtained in the experiment showed that the differences observed between different versions of the game were not statistically significant. One of the main reasons for this is the low number of participants. In similar studies, services such as Amazon Mechanical Turk were used to increase the number of participants. The absence of such an available service required individual access to the participants and resulted in a low number of participants. In order to solve the shortage in the number of users, participants were asked to play all the games instead of playing a single version of the game. This also required a maximum of five minutes for each game to ensure that participants were not bored with the experiment. The statistics of the game completion times also caused a de-

terioration because no matter how difficult the game was to the person; the finishing time was recorded as a maximum of 300 seconds. Although face-to-face testing has led to a low number of participants, it can also be considered advantageous because it allowed close monitoring of the experiment. The observations made during the experiment and the comments of the participants showed that prior knowledge was a factor in game success. In some cases, it was observed that even the information obtained by the participants in the first game was used in a way that negatively affected the discovery of the environment in the subsequent games. Most participants stated that when they didn't play MG for the first time, even though there were more than one princess, they didn't even notice the other targets and tried to reach the target they saw in previous games. This situation can be explained not only with prior knowledge but also with attention and can be considered as a research subject in itself.

Within the scope of the study, a DDQN algorithm was implemented, and the game performance of both the vanilla form and the versions obtained as a result of various modifications were examined. Due to the fact that no reward information is provided by the environment, different reward functions that can be considered as internal rewards have been created and tested. In some cases, the state information had to be changed in addition to the reward. State information was obtained by using the screenshot of the game similar to the literature. As expected, the gaming environment without a reward has been a difficult task for the DDQN algorithm, and the agent failed to finish the game. It was observed that the reinforcement learning algorithms were sensitive to reward information and they can elicit behaviors beyond the expectations of the user due to the bugs in the reward function. When the reward is given in proportion to the difference between the two states, the agent was expected to explore the environment, but instead the agent always jumped exploiting the reward function. Algorithms that create intrinsic rewards based on human curiosity, the reward functions, that encourage the agent to explore the environment, are far more complicated compared to the application here. The difference between the two states only occurs as a result of the agent movement due to the fact that there is no object moving outside the agent in the game screen used in this study, and therefore it is considered that the reward derived from the difference between the two states was sufficient, but this proved to be wrong. In humans and even in animals, it is certain that the process is much more complicated but also practical. Although various animal behavioral experiments have demonstrated the existence of a reward concept, it remains unclear how exactly this system works for complex behavior. Studies on the behavior of humans, animals and biological systems related to these behaviors will certainly be useful for reinforcement learning.

Finally, a way to introduce prior knowledge to the reinforcement agent is sought, and the change in performance was observed together with any behavior similar to human players. In order to accomplish this task, the game is divided into more simple subtasks based on the concept of compositionality. Here two assumptions were made. First it was assumed that the agent could create a knowledge regarding the objects in the game. For example, by

playing the princess mini-game, the agent was expected to learn that princess was a positive object. The second assumption was that the agent could learn how to use certain objects, like using the ladder to climb up or down. The trials have not been successful, and in some cases, even mini-games have not been learned. This shows the difficulty of addressing prior knowledge in reinforcement learning algorithms. Moreover, it may also be said that the assumptions mentioned were not correct, but this cannot be certainly known due to the closed box nature of machine learning methods. If a rule-based method was being used, learned rules could be investigated to see whether or not expected prior knowledge was being learned by the agent. Unfortunately, machine learning methods does not provide this inside information, thus only the end result could be observed.

At this point, the data collected during the experiment was used. A convolutional neural network has been created by using the keys that the participant presses as input and screen image as output. This network has been utilized in two ways. First, the intermediate outputs of this network were used as the state representation for DDQN. Secondly, the trained network was used as a policy during training. DDQN can be used in this way because it is an off-policy method; hence, another policy can also be used to play the game during training. Here, in order to facilitate the problem, it is aimed primarily to learn the jumping movement. A positive reward for the jumping movement has not been defined, and the reward was given in relation to the change in game state. On the other hand, a negative reward for jumping continuously is added as the agent showed such behavior at first. Even though the problem was simplified, and the agent was only controlling one action, still the reward function used needed modification in order to prevent the agent from finding a bug and exploit it. As a result, it was seen that the DDQN algorithm learned to jump correctly, and the game could be completed. The result of using a policy derived from human data during training can, in fact, be considered as having a better exploration strategy. This is consistent with the experiment results. During the experiment, participants planned routes for the goals they set on the game screen and explored the game accordingly. They did not explore areas outside of this planned route. Here it can be assumed that prior knowledge was used in determining the goal and planning the route. For example, it is known from the beginning that the ladder is an object that can be used to go up and down. If the route to the goal involved a vertical movement beyond what is doable by jumping, players made use of the ladder. In reinforcement learning, on the other hand, this should be learned. A method like epsilon-greedy remains much weaker than exploration strategies based on prior knowledge. When epsilon-greedy is used, random movements are used to explore, but the extent to which the environment has been explored or whether important/relevant parts have been explored is not known when there is no reward. Building exploration strategies based on relevant prior knowledge may result in better performance. While it can be said that the exploration strategy of humans is efficient, there are also negative aspects. States explored by humans can be much lower as there is a clear goal and path determined. For example, in the game used in this study, if all the enemies had to be killed before the princess was reached, participants would

not be able to finish the game on their first try. A similar situation may also occur when the meanings of objects in the environment conflict with known meanings.

Returning to the original problem definition the solution found out to be usage of the CNN trained with human data as the target network. Here assumption was that the CNN being the target network could act as a tutor because the target network evaluates the actions chosen by the Q-network. Usage of CNN as the target network guaranteed to have a perfect tutor. The question here is; could introduction of human data through the target network be regarded as prior knowledge? Answer to this question can be given from two different viewpoints. If prior knowledge is purely defined to be the knowledge acquired by the humans through their own experiences, then the target network cannot be regarded as prior knowledge. On the other hand, humans could also acquire knowledge just by observing. From this point of view, it could be concluded that some kind of prior knowledge was introduced by the method used.

This present study has shown that prior knowledge has definitely an effect on game performance when there is no reward/score, but it is not easy to use this phenomenon in reinforcement learning algorithms. However, it is considered that if a way to accomplish this is found, more efficient and flexible reinforcement learning methods can be obtained. Obtaining agents that behave similarly to humans in complex environments is a much more difficult problem. Different disciplines contributing to each other and carrying out research in an interdisciplinary fashion can pave the way for progress in this regard. This work is positioned accordingly, and the problem studied is approached in many respects. In the future, conducting experiments, based on this study, with more participants and different game versions to collect data on people's exploration strategy may help to develop reinforcement agents with high performance and human-like behaviors.

REFERENCES

Baillargeon, R., Stavans, M., Wu, D., Gertner, Y., Setoh, P., Kittredge, A. K., & Bernard, A. (2012). Object Individuation and Physical Reasoning in Infancy: An Integrative Account. *Language Learning and Development*. doi: 10.1080/15475441.2012.630610

Bellemare, M. G., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., & Munos, R. (2016). Unifying count-based exploration and intrinsic motivation. In *Advances in neural information processing systems.*

Bellman, R. (1954). The Theory of Dynamic Programming. *Bulletin of the American Mathematical Society*. doi: 10.1090/S0002-9904-1954-09848-8

Botvinick, M., Ritter, S., Wang, J. X., Kurth-Nelson, Z., Blundell, C., & Hassabis, D. (2019). *Reinforcement Learning, Fast and Slow.* doi: 10.1016/j.tics.2019.02.006

Chollet, F. (2015). *Keras Documentation.*

DeepMind. (2019). AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. *DeepMind*.

Diamond, A. (1991). *Neuropsychological insights into the meaning of object concept development.*

Doshi-Velez, F., & Ghahramani, Z. (2011). A Comparison of Human and Agent Reinforcement Learning in Partially Observable Domains. *CogSci*, 1–6. Retrieved from `http://csjarchive.cogsci.rpi.edu/proceedings/2011/papers/0625/paper0625.pdf`

Dubey, R., Agrawal, P., Pathak, D., Griffiths, T. L., & Efros, A. A. (2018). Investigating Human Priors for Playing Video Games. *35th International Conference on Machine Learning, ICML 2018*, *3*, 2160–2168.

Everitt, T., & Hutter, M. (2019). Reward Tampering Problems and Solutions in Reinforcement Learning: A Causal Influence Diagram Perspective. *arXiv preprint arXiv:1908.04734*.

GoogleResearch. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. *Google Research*.

Hadfield-Menell, D., Milli, S., Abbeel, P., Russell, S., & Dragan, A. D. (2017). Inverse reward design. *Advances in Neural Information Processing Systems*, *2017-*

Decem(Nips), 6766–6775.

Hassabis, D., & Silver, D. (2017). *AlphaGo Zero: Learning from scratch.*

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*. doi: 10.1126/science.1127647

Ho, J., & Ermon, S. (2016). Generative adversarial imitation learning. In *Advances in neural information processing systems.*

Ibarz, B., Irving, G., Leike, J., Legg, S., Pohlen, T., & Amodei, D. (2018). Reward learning from human preferences and demonstrations in Atari. *Advances in Neural Information Processing Systems*, *2018-Decem*(2017), 8011–8023.

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*. doi: 10.1613/jair.301

Kingma, D. P., & Ba, J. L. (2015). Adam: A method for stochastic optimization. In *3rd international conference on learning representations, iclr 2015 - conference track proceedings.*

Kraebel, K. S., West, R. N., & Gerhardstein, P. (2007). The influence of training views on infants' long-term memory for simple 3D shapes. *Developmental Psychobiology*. doi: 10.1002/dev.20222

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*. doi: 10 .1145/3065386

Lake, B. M., Ullman, T. D., Tenenbaum, J. B., & Gershman, S. J. (2017). Building machines that learn and think like people. *Behavioral and Brain Sciences*, *40*(2012), 1–58. doi: 10.1017/S0140525X16001837

Lin, L. J. (1992). Self-Improving Reactive Agents Based on Reinforcement Learning, Planning and Teaching. *Machine Learning*. doi: 10.1023/A:1022628806385

Lobo, M. A., Kokkoni, E., de Campos, A. C., & Galloway, J. C. (2014). Not just playing around: Infants' behaviors with objects reflect ability, constraints, and object properties. *Infant Behavior and Development*. doi: 10.1016/j.infbeh.2014 .05.003

Mindermann, S., Shah, R., Gleave, A., & Hadfield-Menell, D. (2018). Active Inverse Reward Design. *CoRR*, *abs/1809.0*. Retrieved from `http://arxiv.org/abs/1809.03060`

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533. Retrieved from `http://dx.doi.org/10.1038/nature14236` doi: 10.1038/nature14236

Naz, K., & Epps, H. (2004). Relationship between color and emotion: a study of college students. *College Student J*.

Ostrovski, G., Bellemare, M. G., Van Den Oord, A., & Munos, R. (2017). Count-based exploration with neural density models. In *34th international conference on machine learning, icml 2017*.

Palan, M., Landolfi, N. C., Shevchuk, G., & Sadigh, D. (2019). Learning Reward Functions by Integrating Human Demonstrations and Preferences. *arXiv preprint arXiv:1906.08928*.

Pathak, D., Agrawal, P., Efros, A. A., & Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. *34th International Conference on Machine Learning, ICML 2017*, *6*, 4261–4270.

Pavlov, I. P., & Gantt, W. H. (1928). *Lectures on Conditioned Reflexes: Twenty-five Years of Objective Study of the Higher Nervous Activity (behaviour) of Animals*. International Publishers. Retrieved from https://books.google.com.tr/books?id=eaUAyAEACAAJ

Piaget, J., & Piaget, J. (2006). The development of object concept. In *The construction of reality in the child*. doi: 10.1037/11168-001

Puterman, M. L. (Ed.). (1994). *Markov Decision Processes*. Hoboken, NJ, USA: John Wiley & Sons, Inc. Retrieved from http://doi.wiley.com/10.1002/9780470316887 doi: 10.1002/9780470316887

Reynolds, G. D. (2015). Infant visual attention and object recognition. *Behavioural Brain Research*. doi: 10.1016/j.bbr.2015.01.015

Rosenfeld, A., & Tsotsos, J. K. (2018). Bridging Cognitive Programs and Machine Learning. , 1–9. Retrieved from http://arxiv.org/abs/1802.06091

Silver, D. (2015). *Lecture 2: Markov Decision Processes* (Tech. Rep.). London: University College London. Retrieved from http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching{_}files/MDP.pdf

Silver, D., & Hassabis, D. (2016). AlphaGo: Mastering the ancient game of Go with Machine Learning. *Google Research Blog*.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., ... Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*. doi: 10.1126/science.aar6404

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., ... Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature*. doi: 10.1038/nature24270

Skinner, B. F. (1938). *The behavior of organisms: an experimental analysis*. Oxford,

England: Appleton-Century.

Spelke, E. S. (1990). Principles of Object Perception. *Cognitive Science*. doi: 10.1207/s15516709cog1401_3

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning : an introduction*. Cambridge, Massachusetts London, England: The MIT Press.

Tang, H., Houthooft, R., Foote, D., Stooke, A., Chen, X., Duan, Y., ... Abbeel, P. (2017). Exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in neural information processing systems.*

Tasfi, N. (2016). *Pygame learning environment.* `https://github.com/ntasfi/PyGame-Learning-Environment`. GitHub.

Thorndike, E. L. (1911). Animal intelligence: Experimental studies.

Thrun, S., & Schwartz, A. (1993). Issues in Using Function Approximation for Reinforcement Learning. *Proceedings of the 4th Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*.

Tsividis, P. A., Pouncy, T., Xu, J. L., Tenenbaum, J. B., & Gershman, S. J. (2017). Human learning in atari. *AAAI Spring Symposium - Technical Report*, *SS-17-01* -, 643–646.

Van Hasselt, H. (2010). Double Q-learning. In *Advances in neural information processing systems 23: 24th annual conference on neural information processing systems 2010, nips 2010.*

Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double Q-Learning. *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, 2094–2100.

Watkins, C. J. C. H., & Dayan, P. (1992). Q-learning. *Machine Learning*. doi: 10.1007/bf00992698

Wilcox, T., Stubbs, J., Hirshkowitz, A., & Boas, D. A. (2012). Functional activation of the infant cortex during object processing. *NeuroImage*. doi: 10.1016/j.neuroimage.2012.05.039

Yin, H., & Pan, S. J. (2017). Knowledge transfer for deep reinforcement learning with hierarchical experience replay. In *31st aaai conference on artificial intelligence, aaai 2017.*