

PLAYING THE TURKISH TILE GAME *OKEY*
WITH DEEP REINFORCEMENT LEARNING



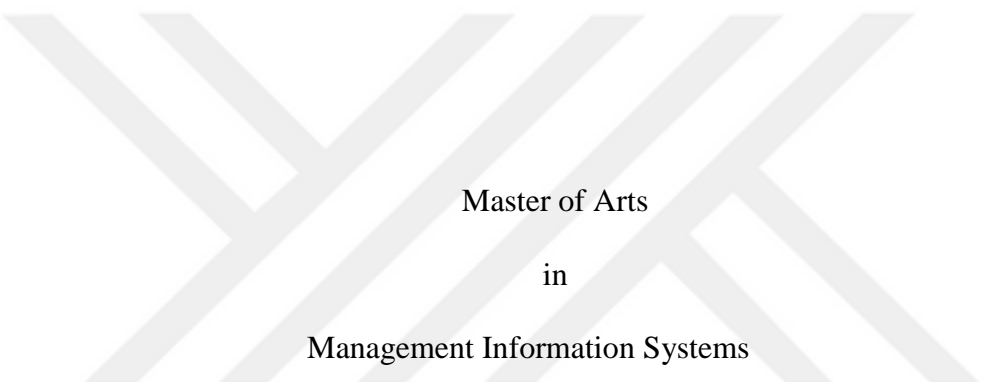
İLKE UYGUN

BOĞAZIÇI UNIVERSITY

2019

PLAYING THE TURKISH GAME *OKEY*
WITH DEEP REINFORCEMENT LEARNING

Thesis submitted to the
Institute for Graduate Studies in Social Sciences
in partial fulfillment of the requirements for the degree of



Master of Arts
in
Management Information Systems

by
İlke Uygun

Boğaziçi University

2019

Playing The Turkish Tile Game *Okey*

With Deep Reinforcement Learning

The thesis of İlke Uygun

has been approved by:

Assist. Prof. Ahmet Onur Durahim
(Thesis Advisor)

Prof. Aslı Sencer

Prof. ErKay Savaş
(External Member)

September 2019

DECLARATION OF ORIGINALITY

I, İlke Uygun, certify that

- I am the sole author of this thesis and that I have fully acknowledged and documented in my thesis all sources of ideas and words, including digital resources, which have been produced or published by another person or institution;
- this thesis contains no material that has been submitted or accepted for a degree or diploma in any other educational institution;
- this is a true copy of the thesis approved by my advisor and thesis committee at Boğaziçi University, including final revisions required by them.

Signature.....

Date13.09.2019

ABSTRACT

Playing the Turkish Tile Game *Okey*

With Deep Reinforcement Learning

Games are important test beds for machine learning studies for over the last decades. Significant progress has been made in games such as Checkers, Chess, Go and Poker with the help of deep neural networks used for function approximation within reinforcement learning algorithms. Agents were able to reach champion or superhuman levels by beating the top players of the world. This study focuses on the Turkish tile game *Okey* and aims to prove that agents can learn to play this game with the guidance of deep reinforcement learning. *Okey* has a unique setting where there is partially observable environment, stochastic nature and multiple players which are fully competitive. The study focuses on teaching a learning agent to play the game without any direct supervision, solely by receiving reward signals at each step for drawing and discarding tiles, with the help of stochastic policy gradients, actor-critic algorithm, prioritized experience replays which are explained thoroughly in this thesis. The learning agent plays against a random computer opponent in the custom Gym environment created for the *Okey* game as a two-player game version. Within the game framework, learning agent plays against an opponent that draws a tile from discarded tiles of the agent or from the center tile randomly, and always discards from the free tiles which makes it compelling enough for the learning agent. The results of the games through the experiments are reflected and win rates of the agent against the computer opponent can be considered as the achieved success of this study. Extensive research on the existing literature shows that this is the first study that uses reinforcement learning to play the game of *Okey*.

ÖZET

Türk Taş Oyunu *Okeyi*

Derin Pekiştirmeli Öğrenmeyle Oynamak

Geçtiğimiz yıllardan bu yana oyunlar makine öğrenmesi çalışmaları için önemli bir test yatağı olmaktadır. Satranç, Dama, Go ve Poker oyunlarında pekiştirmeleri öğrenme algoritmaları kapsamında derin yapay sinir ağlarıyla fonksiyon tahminlemeyle kayda değer ilerlemeler yapılmıştır. Yapay zekalar, oyunlarda dünyadaki en iyi insan oyuncularını yenerek şampiyon veya süper insan seviyelerine ulaşmıştır. Bu çalışma Türk taş oyunu *Okey*'e odaklanır ve derin pekiştirmeli öğrenmenin yönlendirmesiyle yapay zekanın bu oyunu öğrenebileceğini ispatlamayı amaçlar. *Okey*'in kısmi gözlemlenebilir ortamı, olasılıksal doğası ve birbirleriyle tam rekabet içinde olan oyuncularıyla kendine özgü bir yapısı vardır. Bu çalışma öğrenen bir yapay zekanın hiçbir doğrudan yönlendirme olmadan, sadece taş çekerken ve taş atarken her adımda ödül sinyalleri olarak, tez boyunca anlatılan olasılıksal davranış meyilleriyle, aktör-kritik algoritmasıyla, önceliklendirilmiş tecrübe tekrarlarıyla oyunu öğrenmesine odaklanmaktadır. Öğrenen yapay zeka, özel tasarlanmış 2 kişilik *Okey*'i Gym ortamında rastgele oynayan bilgisayar rakibine karşı oynar. Oyun çatısı içinde, öğrenen yapay zeka, yere atılan taşlardan ya da ortadaki taşlardan rastgele çeken ve her zaman elinde boşta olan taşlardan atan bilgisayar rakibine karşı oynar ve bu yapısı onu, öğrenen yapay zeka için yeterince zorlu kılar. Yapılan deneyler boyunca elde edilen sonuçlar bu çalışmada sunulmaktadır ve yapay zekanın rakibine karşı kazanma oranları bu çalışmanın elde ettiği başarı seviyesi olarak görülebilir. Literatürde yapılan kapsamlı araştırma sonucunda bu çalışma, pekiştirmeli öğrenme kullanılarak *Okey* oyununu oynatan ilk çalışma olarak gösterilebilir.

TABLE OF CONTENTS

| | |
|---|----|
| CHAPTER 1: INTRODUCTION | 1 |
| CHAPTER 2: LITERATURE REVIEW | 7 |
| CHAPTER 3: METHODOLOGY | 12 |
| 3.1 Reward function design | 18 |
| 3.2 Actor – Critic setting | 20 |
| 3.3 Greedy approach | 21 |
| 3.4 Prioritized experience replay | 22 |
| CHAPTER 4: RESULTS AND FINDINGS..... | 24 |
| CHAPTER 5: LIMITATIONS..... | 27 |
| CHAPTER 6: CONCLUSION AND FUTURE WORK | 30 |
| CHAPTER 7: BUSINESS IMPLICATIONS | 33 |
| REFERENCES..... | 36 |

LIST OF FIGURES

| | |
|--|----|
| Figure 1. RL agent-environment interaction diagram..... | 2 |
| Figure 2. The complete set of Okey tiles | 2 |
| Figure 3. An example Okey hand | 3 |
| Figure 4. An example Okey hand with free tiles | 14 |
| Figure 5. Action selection agent code | 16 |
| Figure 6. Neural network design for draw action | 17 |
| Figure 7. Neural network design for discard action..... | 18 |
| Figure 8. Agent action selection code for drawing tile | 22 |
| Figure 9. The process flow chart of the Okey game | 23 |

CHAPTER 1

INTRODUCTION

Games have been important test beds for artificial intelligence (AI) studies over the past decade. Significant progress has been made on Chess, Go, Texas Hold'em Poker, and Atari games to name a few examples. Silver and his colleagues (2016) from DeepMind's work AlphaGo in Go, Carnegie Mellon University with Facebook AI's Pluribus by Brown & Sandholm (2019) in No Limit Texas Hold'em Poker, portray promising results of reinforcement learning (RL) algorithms beating the top players and reaching superhuman level. This subset of the field of machine learning called reinforcement learning is getting more attention after these important advancements. Reinforcement learning differs from supervised learning, where there is labeled data set and training takes place with respect to these labels; like in image classification. It also differs from unsupervised learning where there are no labels in the input data and algorithms mainly find similarities in the data in order to group them into clusters. As described by Hu and Wellman (1999), supervised learning contains examples in the form of input and output pairs which are observed by the learning algorithm, and it tries to learn how to map them to each other. Environment provides the output values which guides the algorithm like a teacher or supervisor. However, in reinforcement learning, there is no outer guidance for the learning algorithm (agent) and it only learns by interacting with the environment and receiving observations (states) and rewards (see Figure 1). These observations and reward feedbacks guide the agent about which actions to abandon and which actions to keep doing in order to achieve the goal which is winning the game in gaming

context; even without knowing the game rules. So, the interesting part is that the agent learns to play with optimal strategies even without having to learn the game dynamics (although there are approaches for first figuring out the game dynamics using that insight to learn the optimal policy) and still being able to defeat the top players. This study will explain an attempt in teaching an agent the Turkish tile game *Okey* with model-free reinforcement learning methods.

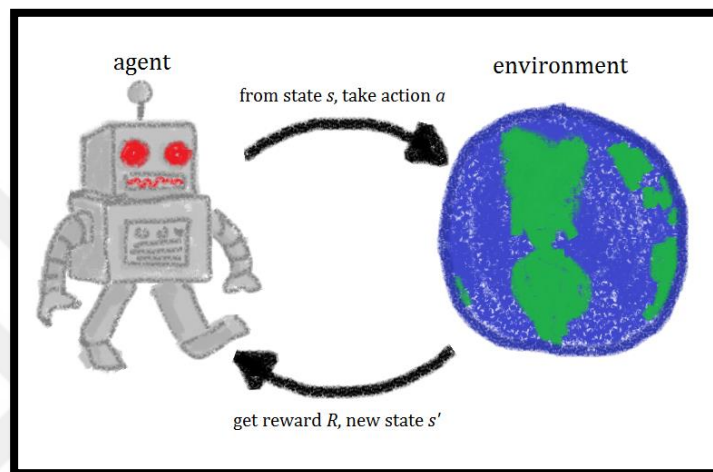


Figure 1. RL Agent-Environment interaction diagram
From Wikimedia Commons, licensed under the Creative Commons Attribution-Share Alike 4.0 International license.

Okey is a popular Turkish tile game generally played with 4 people but can also be played with 2 or 3 people. The game is played with 106 tiles, 2 stacks of 4 different colored tiles (i.e. red, black, blue, yellow), ranging from number 1 to 13 and

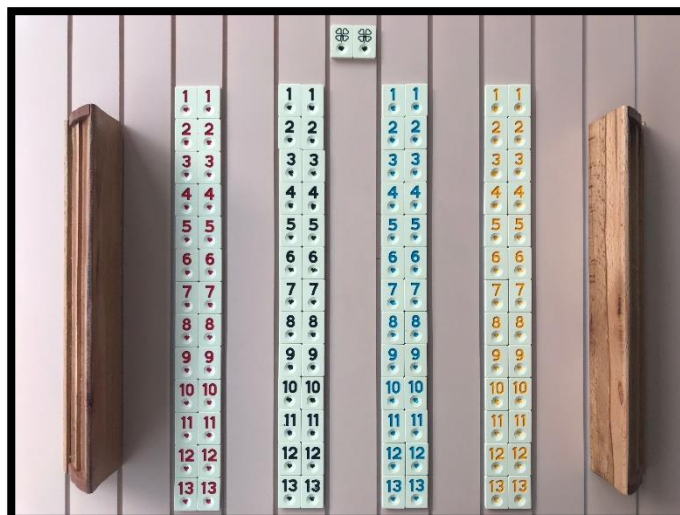


Figure 2. The complete set of Okey tiles

two false joker tiles (see Figure 2). To begin the game, the tiles are placed with values facing down and shuffled. One tile is placed in the center face-up and the one value greater than this tile's value indicates the joker (a.k.a. okey) tile. This wildcard tile symbolizes the name of the game and can be used in place of any tile in player's rack. The false joker tiles, however, can only be used as the real color and value of the joker tile only. For example, if the tile placed face-up is Yellow 5, false joker tiles will only be used as Yellow 6, and the Yellow 6 tiles will be used as wildcard joker tiles. The starting player takes 15 tiles and rest of the players take 14 tiles. The aim is to group or order these tiles in 3-, 4- or 5-tile series on your rack obtaining either tiles with the same rank with different colors like "Red 2, Blue 2, Yellow 2" or the same color with rank ordered tiles like "Blue 1, Blue 2, Blue 3, Blue 4" (see Figure 3). In contrast, players can also choose to form double pairs with the same tiles as in "Yellow 2, Yellow 2", "Black 5, Black 5" and so on but they must complete the hand with only double pairs, thus having seven such double pairs to win the game.

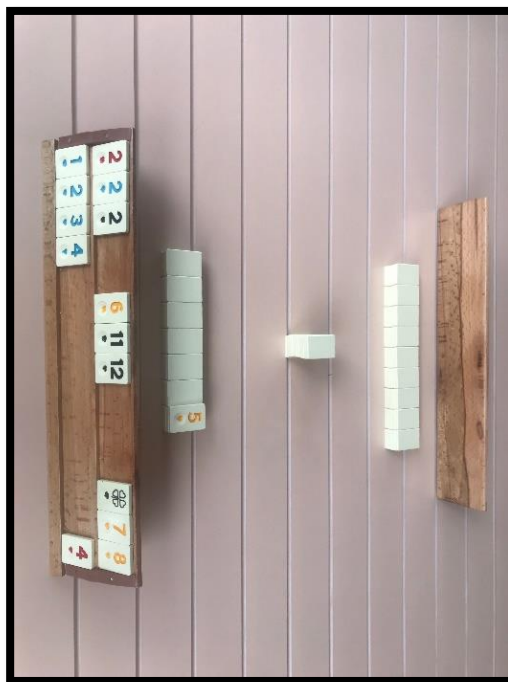


Figure 3. An example Okey hand

The starting player with 15 tiles begins by discarding a tile and each player takes turn anticlockwise by either drawing the discarded tile of the previous player or by drawing a tile from the center. The game goes on until a player finishes her/his hand by having no free tile left, therefore having all of tiles on her/his rack grouped/paired, indicating that s/he has won the episode. The game can also terminate at a state when there is no center tile left and no player has a finished hand (no free tiles left). If this is the case, the game is considered as a draw and a new game can be started, although in one version of the game, players continue to play by shuffling the discarded tiles faced down, forming new deck of center tiles. The winning players score is deducted by 2 points for a regular tile discard, 4 points for winning by discarding an Okey tile or 4 points again by winning with double pairs style in an episode. Usually game is played in episodes by discounting from 20 and until a player reaches 0 which indicates that s/he won the game.

This type of game environment is challenging in terms of reinforcement learning perspective for various reasons. First, there should be multiple agents in Okey, playing in a fully competitive fashion and in some versions additionally in a cooperative fashion. Secondly, the environment is partially observable because an agent cannot see the other agents' racks and the faced down center tiles. To name a few examples, there are similar games like Blackjack and Poker that contain partially observable settings. In Blackjack by Perez-Uribe & Sanchez (1998), players do not see one of the dealer's cards at the beginning and in Poker by Brown & Sandholm (2019), players do not have the chance to see each other's cards. There are also cooperative games like *Hanabi* by Bard and his colleagues (2019), where you do not see your own card and hold them facing to the other players and you communicate with the other players to try to figure out your own cards so that you can come up

with the best hand combination together with the other players while each player is trying to figure out what cards they might be holding. Lastly, the game is stochastic because of the randomness of the center tile drawn. Again, same randomness also holds for Blackjack and Poker because no player knows for sure which card will be dealt by the dealer. On top of these challenges of the environment, the extensive research done on the topic shows that, to the best of our knowledge, there is no attempt that has been made in applying reinforcement learning algorithms to the game Okey. Comprehensive examples from the existing literature on multi-agent reinforcement learning studies which influenced this study is shared in the Literature Review section.

With all these challenging characteristics of Okey, the goal of this study is to try teaching an agent to play Turkish tile game Okey by using the reinforcement learning algorithms. Can we implement an intelligent agent who can play Okey without any guidance, taking required actions wisely like a human would do?

In order to tackle this problem, deep reinforcement learning architectures including policy networks has been designed for each draw and discard decisions to be made by the Okey playing AI agent. Initially, the agent will start by taking actions according to the policy networks initialized randomly and then with the help of the observations received from the environment. Rewards obtained after taking an action in particular states were fed back to the system to improve the effectiveness of the decisions made in the next steps. Details of the methodology followed in this respect will be explained in Methodology section.

Aru and his colleagues (2017) state that, when an agent trains against another agent, this approach is called self-play and it is possible to come up with more

general strategies as a benefit. Usually, first the agents are trained with supervised learning because supervised learning is used to give example games as input. Afterwards, they train by self-play in order to increase the performance. For example, for the training of the AlphaGo by Silver and his colleagues (2016), the human expert moves were used as the initial training data. However, in this study it is preferred to train the agents with only self-play in order to find out adaptive strategies which is also the case in the follow up work for the improved version of AlphaGo, named AlphaZero, as presented in Silver and his colleagues (2017) and again in Silver and his colleagues (2018). We elaborated more on the outcome of the experiments performed in this study in Results and Findings section.

CHAPTER 2

LITERATURE REVIEW

Games can be classified into perfect information games and imperfect information games depending on the player's visibility of the game environment information. Perfect information game is the game in which the player can get all information about the state of the game, and imperfect information game is the game in which there is hidden information. For example, as Takaoka, Kawakami and Ooe (2017) illustrated, some examples for perfect information games are Chess, Go and Shogi, and in contrast, Rummikub, Mahjong and Bridge are the examples for imperfect information games. Okey being very similar to Rummikub game, is also an imperfect information game. Rummikub is also played with same racks and tiles but only difference is that first person to reach a pair of 30 points (each tile's face value indicating the point of a tile) melds the pair on to the center and players keep adding tiles to the center, similar to Scrabble in this sense, instead of keeping the ordered sets or groups in their rack secretly. So, the game still contains partial observability but in a different manner. In one of the key studies in reinforcement learning field, Heinrich and Silver (2016) stated that it is possible for each player to observe only his own state information, like a poker player sees his own private cards but doesn't know the other players' cards. Likewise, in Okey, a player can only partially observe the game environment, being able to see his/her own tiles, last discarded tiles of the other players and face-up tile in the center. Besides, the randomness in the tile that can be drawn from the center tiles and other players' actions bring stochasticity to the game.

While considering other players' decisions, it is also necessary to delve into the agents' interaction with each other when conducting a research, especially in multi-agent reinforcement learning (MARL) games. Buşoniu, Babuska and De Schutter (2011) implied that MARL techniques in games can be classified as fully cooperative, fully competitive or mixed stochastic games depending on the type of task targeted by the learning algorithm and how they address the learning agents' behaviours. In this manner, Okey is a type of game comprising of fully competitive players. A contrasting example can be given as Hanabi, which according to Bard and his colleagues (2019) is an imperfect information game played by two to five players which are in full cooperation, making the game like teamed version of solitaire. This is a game where players fully cooperate to maximize the shared reward. In Hanabi game, players only see the other players' hands and communicate with them to give clues about each other's hands to form the best combination of cards, consequently, to form the best possible cumulative hand in full cooperation.

Another important classification for games is made depending on the reward functions of the environment. Nowe, Vrancx and De Hauwere (2012) described that, the game is an identical payoff or common interest game when the same reward function is shared among all players, however, a zero-sum game when the total of all players's rewards sum up to 0. Two-player version of Okey can be considered as zero-sum game. According to Bard and his colleagues (2019), important two-player zero-sum games which contributed remarkable studies in the field of artificial intelligence by enabling computers to reach super-human skills are chess (Campbell, Hoane, & Hsu, 2002), checkers (Schaeffer, Lake, Lu, & Bryant, 1996), go (David Silver et al., 2016), backgammon (Tesauro, 1995) and two-player poker (Moravcik et al., 2017). Furthermore, Diddigi, Kamanchi and Bhatnagar (2019) stated that

information of the model of the game dynamics is not given to the players in majority of the two player zero-sum game settings in real life, and players try to utilize the states and rewards gathered from the environment, in order to come up with the optimal policies by just knowing the rules of the game only.

Another important factor affecting the complexity of the problem is the number of players, or learning agents, in the game because each agent adds its own variables to the joint state-action space. As a result, Foerster and his colleagues (2018) expressed that when the number of agents increase, the growth in the joint action space is exponential and it makes the learning very challenging. Moreover, Neto (2005) explained that there is stationary environment in single agent examples but in multi-agent scenario, each agent is changing the environment. This is a tough challenge because the agent is learning in a non-stationary environment with the possibility of other agents learning in a similar fashion as well. In parallel to that, Buşoniu, Babuska and De Schutter (2011) described that the best policy keeps changing with the changes in other agents' policies, which they defined as moving-target learning problem for each agent to deal with. So, this explains how dynamic state-space is being shaped after each turn of an agent. In single agent scenario, there is no such issue, and the problem is considerably easier to tackle. As it is stated in Wai, Wang and Hong (2018), there has been a considerable success achieved in single-agent reinforcement learning, but multi-agent reinforcement learning (MARL) is still being challenging because each agent interacts with each other while they are also interacting with the environment.

Reinforcement learning methods are chosen to be applied to try teaching a machine how to play Okey like a human player. According to Sethy, Patel and Padmanabhan (2015), reinforcement learning is the field of machine learning that

deals with agent figuring out how to behave in a certain situation by mapping situations to actions in order to maximize a numerical reward signal. The learner has no preconception of which actions to take, it is not told externally, but instead it has to decide on its own by trial and error considering the maximum reward it collects in the long run. Generally, this is a challenging and interesting approach because the actions taken may not affect only the immediate reward received but also the upcoming situations and therefore all subsequent rewards in the long run. So, RL agent must make the decisions and take actions accordingly, and receive observation and reward from the environment and update its beliefs on what is the correct action in a particular state of the game and this loop continues until the game ends.

In order to overcome the problem of making sequential decisions in a dynamic game environment, *deep* neural networks were implemented within reinforcement learning framework for making decisions, thus named deep RL. There are two moves to be made by the agent playing Okey game which are drawing a tile and discarding a tile. Endicott (2017) clarified the network setup attentively by stating that we should use our state as input to the neural network before making a move. In order to select the move with the help of the network, the resulting state from each possible move can be send to the network and the network will output the move with the highest probability to win with respect to the past moves. The state will contain the most recent observation of the agent either before it withdraws a tile or draws one. Technical details about how decision-making units are modeled as deep neural networks together with the reward design are provided in the Methodology section.

Deep learning and deep neural networks are two important terms that are used interchangeably where a neural network has more than one hidden layer, possibly many of them.

Lastly, there are other interesting and promising studies on RL, in other application domains as well. One example is by Zheng and his colleagues (2018) where they applied RL in a news recommendation system. They used Deep Q-Network to calculate the Q-value of a news article using the user features and context features representing the state of the environment are fed as the input. Based on the resulting Q-value, a list of news is recommended to a user. User's click on a certain news article is used as part of the reward agent received to improve the system. Another interesting application of RL is in finance domain. As it is stated in Srinivasan (2018), IBM built a complex trading system which makes financial trades using the power of reinforcement learning.

CHAPTER 3

METHODOLOGY

Can one implement intelligent agents who can play Okey without any human guidance and take required actions wisely in a manner like a human would do? To answer this research question, deep reinforcement learning methods are investigated and implemented, which corresponds to using deep neural networks as a function approximator within reinforcement learning algorithms. Specific to the game of Okey, there are two decisions to be made. First one is to decide on whether player should draw the tile that is discarded by the opponent player or from a tile from the center tiles. Second decision to be made is about which tile to be discarded from the player's hand tiles. As a result, a player is represented by a combination of two decision making entities, one for the draw action and the other for the discard action. We handle the problem of creating these entities at two different steps. The first step is to understand and implement the policy gradient algorithm and value functions for reinforcement learning, where the agent interacts with the environment by sending actions and receiving observations representing the state of the game as well as rewards. Policy and value networks get the current environment state that the agent is in as the input and output the action probability distribution and value of the state as the output, respectively. As it is prescribed in the study by Foerster and his colleagues (2018), each agent should learn independently from its own action and observation history when applying policy gradients to multiple agents in the simplest way possible. As Foerster and his colleagues suggest, the learning agent has an actor network corresponding to the policy network for each decision, draw and discard, and a critic network which embodies the value network to predict the value of the

state and return the error in actor's actions and value predictions. The details about the actor-critic setup is explained in detail under "3.2 Actor Critic Setting" section. In order to choose the draw and discard action, deep neural networks are designed for each action. Policy network is forwarded to calculate the probabilities of actions in order to choose the action either stochastically according the output probabilities or greedily where the action with the highest probability is selected. Then, agents are trained at each episode with the state, action and reward tuples to improve its decisions. Training corresponds to minimizing the loss function which is the term that agent tries to optimize to learn the best decisions to be made and tuning the action probabilities accordingly for the next episode. Multiple runs of the game have been executed in order to observe the improvements in the winning rates, net value gained and mean rewards after each game episode in the long run. Net value gained is the difference between the hand value for the final hand and the hand value for the initial hand obtained at the beginning of the game. Information about the hand value and rewards are detailed under "3.1 Reward Function Design" part.

In addition to considering the implementation details of the actor critic algorithm, different architectural parameters have been tested and detailed experiments has been conducted to evaluate greedy vs. stochastic action selection approaches, and utilization of prioritized experience replay. Greedy approach is where the agent always chooses the action with the highest probability, or exploits in other words, rather than randomly selecting an action according to the action probabilities returned from the decision network. Prioritized experience replay is also an important concept for reinforcement learning systems because it allows you to keep important samples, as $(state, action, reward, next\ state)$ tuple, and with a priority assigned to each sample which is the temporal difference error calculated for

these examples which is the difference between the estimated value of a state and bootstrapped estimation of that state. The details about how the temporal difference error is calculated are stated in “3.2 Actor Critic Setting” section and details about the prioritized experience replay are elaborated in “3.4 Prioritized Experience Replay” section. Crucial point is that experiences observed after each step of the discard action are stored in experience replay memory and selection of a predetermined sized batches of the most important samples are used at the end of each episode for training the discard networks of the agent. Also, the comparison of the results obtained with respect to the employed approaches stated here are evaluated in “Chapter 4: Results and Findings”.

For this research, custom Gym Environment of the game Okey is implemented where a randomly acting computer opponent plays against the learning agent which is the matter of subject for this study. The computer opponent picks to draw from discarded tiles or from the center tiles randomly but always discards a free tile where that tile does not belong to any three or four-tile groups. For example, in the Figure 4, “Red 1, 2, 3” is a three-tiled series and 7’s are four-tiled series and the rest of the tiles are free tiles which are of no use.



Figure 4. An example Okey hand with free tiles

To detail the steps of the research, it is important to start with the neural networks’ designs. There are three deep neural networks designed for the research

problem in this study: one network for discarding tile, one network for drawing tile which take the role of actor part of the learning agent. Actor determines the action probability distributions. Also, there is one network for the critic which acts as a judge for the actor about the discard action, generating value for the given input state.

The neural network for drawing a tile is a simple one and consists of one input layer of 107 nodes, one hidden layer of 512 nodes and one output layer of two nodes. 107 nodes at the input layer holds the hand tiles represented as an array of 52 indicators (13 values for each 4 different colors, Red, Black, Green and Yellow in order) where each indicator denotes the number of tiles at hand. For example, if the player has one “Red 1” tile in the hand tiles, the indicator at the first index of the hand tiles representation takes value 1 and otherwise it takes the value of 0 if the player doesn't have “Red 1” tile in its hand tiles. So, if the player has two “Red 1” tiles, then the first index will be 2 instead of 1, representing the count of that specific tile. As a result, sum of the values in hand tiles representation will add up to 14. Also, joker count is appended at the end as the 53rd item. So, hand representation for the state is always an integer array with 53 items.

Additional 53 nodes for the discard neural networks input layer is similarly an array of 53 integers allocated for representing the opponents' discarded tile lying on the ground, available for drawing by the learning agent. For example, if this discarded tile is a “Yellow 13” which is the last tile in the representation of all tiles, the array will contain 51 0's followed by a 1 as the last item. Also, 53rd item will be appended as 1 or 0 depending on this item being the joker or not. In total, this makes 106 nodes, where first 53 items represent the hand tiles plus the next 53 items represent the last discarded tile by the opponent. The last, 107th node contains the

number of the center tiles remaining which is available for drawing. These nodes in the input layer reflects the observation returned from the Okey environment before deciding on the action to be taken in drawing a tile. On the other hand, two nodes at the output layer symbolizes the possible actions for drawing either from the center tiles or drawing the discarded tile. First node (which gets inactive if the drawing is made from the center tiles and active if the discarded tile is drawn which corresponds to a value of 0 or 1 for the `draw_a` variable seen in Figure 5, accordingly) is designed for drawing the discarded tile. Second node gets active if the player draws from the center tiles (where variable `draw_a` gets a value of 1). Action probabilities (`draw_action_probs`) are output from the network and used for selecting the draw action taking these probabilities into account. Related source code for agent action selection for the draw is given as follows;

```
class Agent:
    def draw_tile(self, can_draw_from_discarded, can_draw_from_center, draw_a):
        if draw_a == 0 and can_draw_from_discarded:
            action = Actions.draw_discarded
        elif can_draw_from_center:
            action = Actions.draw_center
        else:
            action = Actions.can_not_draw
        return action
```

Figure 5. Action selection agent code

The flags `can_draw_from_discarded` and `can_draw_from_center` take values depending on the agents' observation's `discarded_tile` and `num_center_tiles_left` representations, respectively. If one of them reaches zero, the related boolean value is set to false which means that the agent cannot draw from discarded tiles or from center tiles. This is how first neural network works by inputting the last observation before drawing a tile and outputting the probabilities of either drawing from the

discarded tiles or from the center tiles which guides the agent to try to select the one considering the probabilities. It is also important to note that, in Figure 6 dark blue circles are symbolizing the input neurons and there are 107 of them. Light blue circles are the hidden nodes which consists of 512 neurons. Finally, two green neurons denote the nodes in the output layer which give the probabilities of drawing the discarded tile and drawing from the center tiles, respectively.

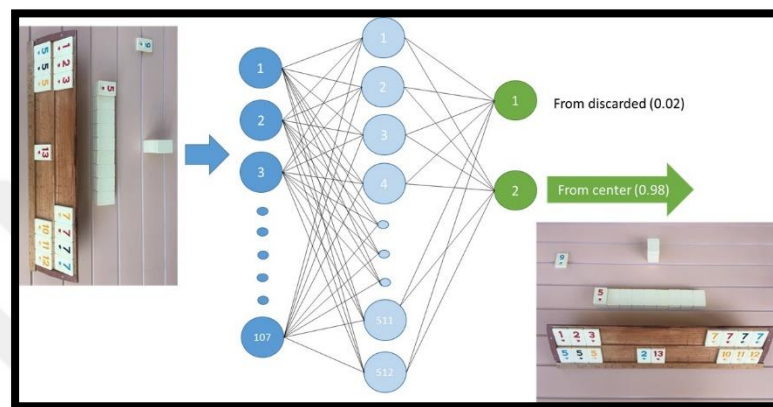


Figure 6. Neural network design for draw action

Furthermore, the structure of the neural network designed for the discarding tile action is similar to the draw network with minor differences. First, number of input neurons is 53 in discard network because only the hand tiles are given as the observation where the first 52 integers represent the counts of hand tiles plus 1 integer denoting the joker count. Second, there are 15 nodes at the output layer, containing 1 node per each existing tile in the agent's hand. This design guarantees that the network selects one of the tiles that the current hand contains. Also, there are three hidden layers in the discard network because the optimal target policy that is aimed to be learned is much more complex for the discard action as compared to the draw action. As it can be seen from Figure 7, integer representation of the tile at agent's hand are fed into the input layer and discard action probabilities are computed as the results of the output layer. Finally, the agent will choose to discard the tile with the highest probability returned from this neural network in the greedy

action selection approach or choose the one stochastically according to the selection probabilities. There is also another network that is used as critic which is explained in Section 3.2.

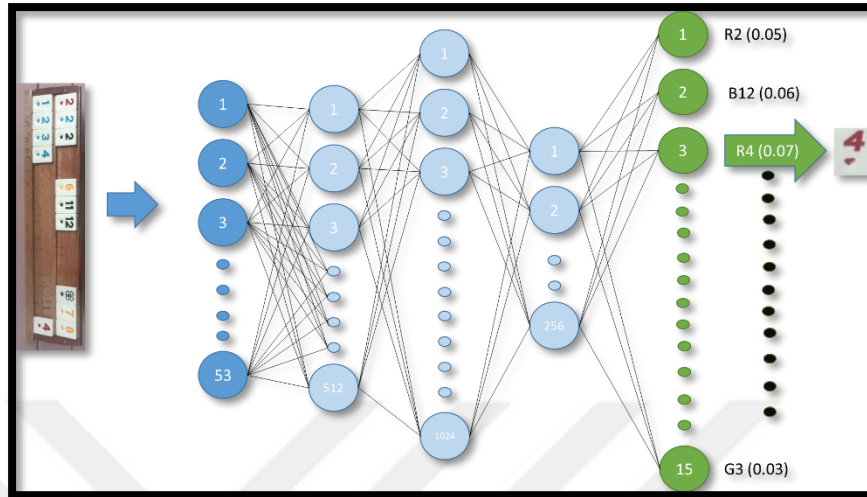


Figure 7. Neural Network Design for Discard Action

But the important question is how are the agents going to learn and improve their decisions when they start with random actions? What makes them to reinforce better game play just by playing with each other without any external guidance? How will this setting react when there is one randomly playing agent against a learning AI agent? To answer these crucial questions, next subsection elaborates on the reward function design which shapes the feedback loop and guides agents for improvement.

3.1 Reward function design

In order to design a good reinforcement learning framework, it is important to devise an effective reward mechanism within the environment. Initial attempt was to place a reward only at the end of the game which is the information as win or lose. So, if the agent wins the game, it receives a predetermined positive reward (i.e., 100), and gets the same but negative reward (i.e., -100) if the agent loses the game in that episode.

This design has the problem of learning too slow because agent must wait till the end of the episode in order to receive reward and discount this reward to all the actions taken throughout that episode which provides sparse rewards. Although this setting would have found the optimal action policies for the draw and discard, it requires much more training time and/or more processing power. To overcome this problem, partial reward scheme was designed to ease the sparse reward environment as follows.

The value of a given hand is calculated by considering the number of 3, 4 and 5-tile series at agent's hand tiles. Each 3-tile series (i.e. Black 3, Black 4, Black 5) are given 10 points, 4-tile series are given 15 points, and 5-tile series are given 20 points. So, the hand value is the sum of points obtained by taking tile series into consideration. Before each draw action, initial the hand value is calculated. The hand value is once more calculated after the drawing action is performed. The difference between this value and the initial value is returned as the reward. For example, if the agent draws a tile and makes a new 3-tile series as a result, 10 points is the total computed reward for this step. This is provided as an intermediary reward, which guides the agent in each step in finding the optimal draw policy.

Similarly, for each discarding action, value of the agent state can be calculated as the difference between the resulting hand value after taking the discard action and the hand value before that action. For example, -10 reward is given for each broken 3-tiled series, and -5 for each 4 or 5-tiled series, meaning a tile from the series is discarded. In addition to these, +5 reward is returned if the player discarded a free tile, meaning that it made the right decision by keeping all series in hand. So, as a result, with the help of rewards obtained at each step, our framework reinforces the learning of the agent.

Finally, when the episode ends, sum of these SAR triples (State, Action, Reward) collected at each turn from the agent actions are fed back to the network for training. After training the agent networks, a new episode starts and agent selects its action with the network trained with the experience obtained in previous episode.

It is also important to state that although the okey tile is valued the highest in the original game, the current reward design puts no emphasis on the uniqueness of the okey tile and it is valued as any other tile in a series. Similarly, opting for double series has no reward for the moment even though they have value more than regular series. These limitations are also referred in the Limitations chapter.

3.2 Actor – Critic setting

Actor – Critic algorithms play a vital role in reinforcement learning systems because they speed up the learning process significantly. The main idea is that there are actor networks which predict the action to be taken and the agent takes these actions and ends up in a new state, and receives new observation as a result. At this point, the critic comes into play to assess the value of the actor's states and assigns a score for both the state after the previous action is taken, and the resulting state after the current action is taken. For example, for the discard action, the steps taken in that respect are given as follows;

1. Critic takes the current state representation after the draw action is taken into its' network as input and outputs a score as the next state's value.
2. Reward is calculated as the sum of the reward after the discard plus the reward after the draw because there are two steps in between two consecutive discard actions, a draw followed by a discard.

3. Temporal difference target (td-target) is calculated as the reward obtained if the episode is done, or the reward obtained plus the discounted value of the next state where the discount factor is set as 0.99.
4. Critic predicts the current state's value score by taking the previous state representation as input.
5. Temporal difference error is calculated by subtracting the current state value from the td-target from Step 3.
6. Critic is updated with the previous state drawn from Step 4 and the td-target computed at Step 3 to minimize the loss and improve its' discard action selection policy.

So, critic network predicts scores for the two consecutive states after the discard action is taken by the actor network and updates itself with the resulting state and the bias resulted from its' predictions. Thus, for the next discard, the critic predicts with less bias and keep minimizing its' loss to be optimized for value score predictions for the states.

3.3 Greedy approach

Another approach that is tested through the experiments is the greedy approach where the agent opts for the action with the highest probability instead of randomly selecting considering the action probabilities. When the agent acts with the greedy approach, both in drawing a tile and discarding a tile, the output with the highest probability is selected as the action to be taken. Learning efficiency is tested by training agent networks both with the greedy and stochastic action selection approaches.

As an example, the agent action selection code for the tile drawing action is given as follows;

```
draw_action_probs = actor_draw.predict(std_0)
if greedy_action_selection_draw:
    draw_a = np.where(draw_action_probs[0] == np.amax(draw_action_probs[0]))[0][0]
else:
    draw_aprob_cum = np.cumsum(draw_action_probs)
    rndnum = np.random.uniform()
    draw_a = np.where(rndnum <= draw_aprob_cum)[0][0]
```

Figure 8. Agent action selection code for drawing tile

3.4 Prioritized experience replay

Different forms of experience replays are common methods used in the reinforcement learning applications because it is an effective way of reusing the previous knowledge to improve the agent. The idea is to hold the moves represented as (state, action, reward, next state) tuples attached with the temporal difference error explained in Actor-Critic Setting part as the priority value at each step and reuse a selected batch of them with the desired size for learning at the end of each episode. The important point for the prioritized experience replay is that memory is kept for the whole run and used at each episode in contrast to the agent SAR triples which are discarded at the end of each episode. For this reason, more important experiences meaning that the experiences with high td-errors, either positive or negative, regarding decisions made throughout the whole episodes are guiding the agent at each episode. This is parallel to real life experiences where a real upsetting or glowing memory is kept more importantly in our brain and reused more carefully when we are in a similar situation or “environment”. The setting of the prioritized experience replay buffer is to hold 20,000 memories in total and to discard the oldest when a new experience is added to the memory. A batch of 64 most important experience tuples, ones with highest priorities, are fed into the actor’s discard network at the end of each episode to try to reach the optimal action selection policy.

Training is carried out for 100 times at the end of each episode to improve the learning of the agent.

Results obtained from the experiments performed by considering the use of greedy action selection approach and utilization of the prioritized experience replay are stated in the next section.

Following is the process flowchart of the self-play reinforcement learning implemented in our research (Figure 9).

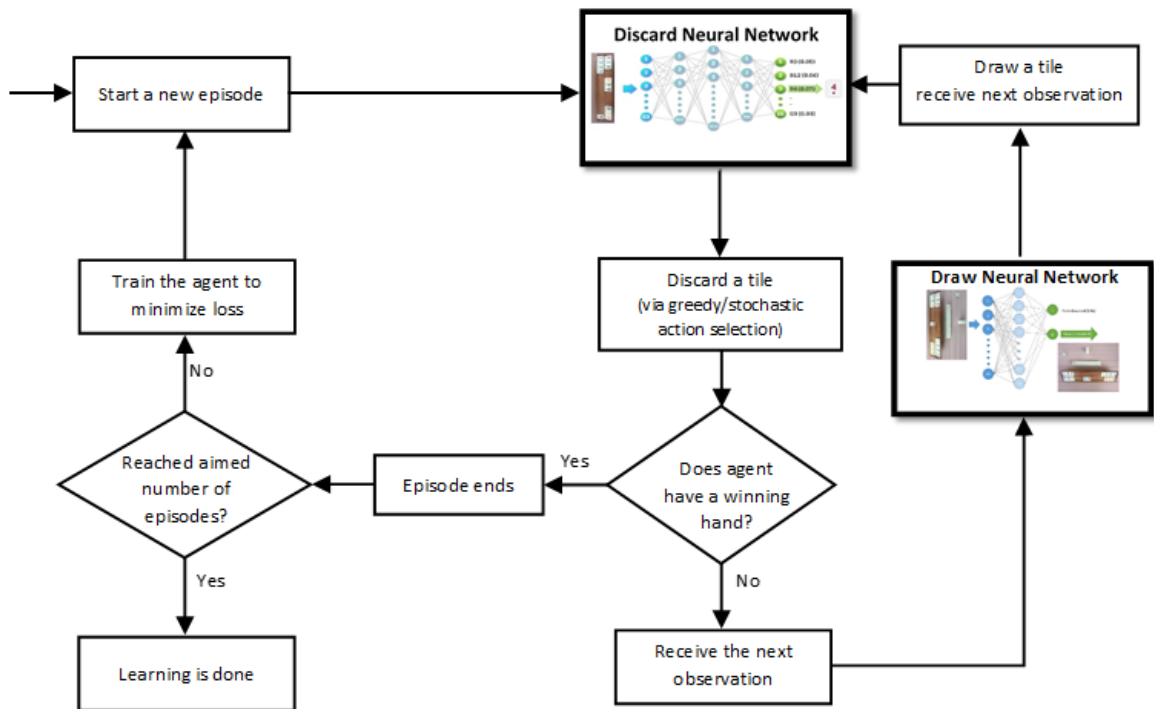


Figure 9. The process flow chart of the Okey

CHAPTER 4

RESULTS AND FINDINGS

Gathering the results of the experiments is interesting part of this research because the system runs for 100,000 episodes and the logs are observed throughout the process of running. The key factors to indicate the learning of the agents are mean draw reward, mean discard reward and the win rates for the runs. Majority of the episodes end by center tiles finishing which can be interpreted as draw. Computer opponent generally performed better than the learning agent because it always discarded intelligently from the free tiles similar to what a human player would have done, which means making no mistakes in keeping the series in hand. On the other hand, the agent managed to win some of the episodes and the details of the ratios of each party is shared in this section.

First of all, there are two different flags in place for the controlled experiments; utilizing a greedy action selection approach, and whether or not employing prioritized experience replays in training of the learning agent. As a result, $2^2 = 4$ unique runs were conducted to test every combination, each for 100,000 episodes. Each parameter has been explained in detail the previous Methodology chapter. The results are as below;

1. Discard greedily, prioritized experience replay

Agent Won: 16 (0.016%)

Draw: 68,436 (68.436%)

Comp. Opp. Won: 31,548 (31.548%)

Mean Draw Reward: 73.4

Mean Discard Reward: 133.7

2. Discard non-greedily, prioritized experience replay

Agent Won: 2,337 (2.337%)

Draw: 65,471 (65.471%)

Comp. Opp. Won: 32,192 (32.192%)

Mean Draw Reward: 68.3

Mean Discard Reward: 145.9

3. Discard greedily, no prioritized experience replay

Agent Won: **5,937 (5.937%)**

Draw: 48,983 (48.983%)

Comp. Opp. Won: 45,080 (45.080%)

Mean Draw Reward: 117.9

Mean Discard Reward: 407.4

4. Discard non-greedily, no prioritized experience replay

Agent Won: 1,356 (1.356%)

Draw: 54,417 (54.417%)

Comp. Opp. Won: 44,227 (44.227%)

Mean Draw Reward: 133.9

Mean Discard Reward: 390.7

As it can be concluded from the results of the experiments, highest winning rate and the highest mean discard reward is from the 3rd experiment where discarding action was taken greedily, without the aid of prioritized experience replay. Higher the mean discard reward means the agent performed better in holding on to the existing series discarding from the series belonging tiles as less as possible. 5.937% of the games for this run resulted as the learning agent winning which is the maximum value in between all experiments. Around 48% - 68% of games ended as a draw and this can be seen as a target for the agent win rate to steal from for further studies because the computer opponent will never be able to improve itself. It will be a major step if the learning agent can pass the 38.2% winning rate of the computer opponent which is the average for the computer opponent among the 4 experiments. Also, it is possible to reach stronger confidence level of results if the experiments can be conducted with higher episodes per run, much higher than 100 thousand episodes.

One last point that needs to be elaborated is the effect of the use of prioritized experience replay in conjunction with greedy action selection where an unexpected outcome is obtained. It is expected that the use of prioritized experience replay improves the learning rate of the agent, but in the experiment where it is used with greedy action selection, it is seen that it actually deteriorates the learning of the agent and agent wins lesser number of episodes. This situation may be observed due to the design of the architecture of the critic network or because the number of episodes is not enough for populating sufficient number of important examples. So, the use of prioritized experience replay needs further investigation.

CHAPTER 5

LIMITATIONS

One limitation of current Okey implementation is that playing with the paired tiles was not implemented. This type of strategy is discussed in the first chapter, where a player can try to collect pairs of exactly the same tiles and trying to reach to 7 pairs in order to win the game. To allow this type of game play learning, existing reward model must be expanded to support “paired tiles”. It is not easy to come up with the comparison of the value of the existing tile series and pairs. Furthermore, the environment state representation has to be refactored to specify that a certain player is aiming for pairs because other players has to be notified for this case. Also, finishing/winning with pairs has the same value as finishing by discarding the Okey tile which is another limitation of the current version. The case where the agent discards an Okey tile to have a finished hand, to win the game in other words, is not covered as well but the reward model can be thought as parallel to pairs strategy mentioned to extend the game to a more real-world scenario.

In addition, the game can be implemented with four players placed on sides of the square board where two players facing each other teamed together. This version of the game is called “*Eşli Okey*” in Turkish (can be translated as *Teamed Okey* or *Okey with Pairs*). For this type of the game, the reward model must be revisited to consider a team’s cumulative success. Also, the communication channels must be developed so that teamed agents can cooperate with each other. For example, one common scenario is when one agent stops aiming for a good hand and starts discarding the tiles strategically (discard like the next player’s discards) so that the next player, who is an opponent player, cannot draw the discarded tiles. As a

result, the opponent player, next to the agent aiming to prevent him/her from winning, is forced to draw from the center and left with only chance from drawing a random tile and therefore other teammate can try to finish before the opponent. These type of strategies and communications are hard to design but can be considered for extending the study for further. Mixed agent behavior algorithms can be searched to figure out a solution from similar games where there is both cooperation and competition.

Also, current game environment lacks to play for certain number of rounds when center tiles are finished, for example there can be an additional round to play by drawing discarded tiles of the opponent to have a minor extension to the episode length.

Another limitation is about the state representation provided to the learning agent. Current state representation for the discarded tile contains only the one hot encoded representation of the last tile but it will be more comprehensive to include a list of past discarded tiles within the current episode, i.e. last 10 tiles discarded by the opponent and include this within the observation seen by the learning agent so that the agent can decide considering more information from a greater memory. Besides, since all the discarded tiles can be observed by all the player, in addition to the discarded tile that can be drawn by the agent, all the discarded tiles can be incorporated into the state representation of the agent.

Moreover, current hand ordering logic does not allow for creating 5-tiled series. It is a minor limitation to the current game play because it allows up to 4-tiled series. Creating more than 4 tiled series is a rare case and usually the strategy is to split series when reaching to 6 tiles into 2 different 3-tiled series for increasing the

chances of expanding within more options but 5-tiled is a special case which should be allowed. This way, the win rates of the players can be improved slightly.

Lastly, there is another limitation in the game when trying to discard from 5 separate 3-tiled series. It is a rare case but with considering all the discarded tiles of the opponent or all the other players, the learning agent can decide better which 3-tiled series it needs to break and discard from it. Currently it doesn't follow discarded tiles carefully because no such observation is provided by the environment.



CHAPTER 6

CONCLUSION AND FUTURE WORK

The goal of this study was to try to figure out if an intelligent agent can be implemented for learning to play the popular Turkish tile game Okey just with self-play and with no human guidance. It is shown in the study that a deep reinforcement learning agent can learn to play a simplified version of the game successfully with the help of deep neural networks used in conjunction with reinforcement learning algorithms. Methodology section described how the simplified problem was dealt with in detail and how the reward function was modeled as well as other supporting algorithms and parameters like prioritized experience replay and greedy approach. Results and findings showed the progress of the experiments and success of the learning agent in each experiment. It can be concluded that this research provides valuable insights for implementing the learning agents for the Okey game. It can be considered as an initial attempt for applying deep reinforcement learning to Okey game and provides a new testing environment for further studies. This research can be extended by considering specifically the partially observable nature as well as the multi-agent cooperation and competition among the players that exists in the original version of the game.

In the multi-agent case, it will be crucial to understand how existence of the other agents as well as cooperation and competition among the players affect the learning capability of the agent. Because when there are multiple agents learning in the environment, there will be more information to observe, but at the same time there will be increased uncertainty due to extra non-observable hands of the opponents as well as allied player in the game.

Finally, apart from the limitations and improvements that can be made to the game environment, neural network or agent design, one future work can be planned to train the agents not by just self-play but also playing against human opponents. For example, game can be extended to 4 players and 3 human players can play against the learning agent which can help the agent to learn from other's decisions. Also, it can be beneficial for evaluation purposes because AI succeeding against human players is an important criterion for proving the strength of the system. Furthermore, a tournament can be found or organized in order to categorize the strengths of the players and try to beat all players to become the winner of the tournament. Unfortunately, there is no existing association or league of the game Okey as far as our research is concerned, but the study can be announced to related communities who can be interested. To help this somewhat formal evaluation process, official rules of the game Okey can be gathered (there is none yet according to our research) but it will be useful to revise the game environment with the light of such a rule set.

As another future work, design of the reward function will be reconsidered since the design in this study provides information to the agent which may lead it to learn and stuck at sub-optimal policy. Instead of leading the agent with the current reward design, a sparse reward function that only provides win/lose information to the agent can be implemented in order for the agent to find optimal policy.

To wrap up, this study has proven the fact that an intelligent agent can learn to play the Okey game without any human guidance or without even knowing the rule set of the game. Besides learning agent succeeds against an intelligent computer opponent just with the help of deep reinforcement learning algorithms. The deep reinforcement learning agent reached up to 5.937% winning rate and this level can be

considered as a basis for future studies to be conducted in reinforcement learning field considering the game Okey.



CHAPTER 7

BUSINESS IMPLICATIONS

Reinforcement learning studies generally focus on games at first because of their safe nature and easier to simulate environments. For these reasons, games serve as a good test bed for this research field but broader goal of course is to find practical applications that will serve for businesses as well as humanity itself. There are outstanding efforts in robotics, self-driving cars, industrial automation, healthcare and medicine, resource management for computers, personalized recommendations for targeted marketing, and automated financial stock trading to name a few areas.

First of all, self-driving car industry is a prominent application field of reinforcement learning. Although, the main focus is on sensor interpretation for recognizing road marking, traffic signs, pedestrians and other related objects, reinforcement learning is used in control systems of autonomous vehicles for supporting to select accelerator, brake and steering. According to Lytvynova's (2019) recent article on RL business applications, UK Company Wayve claims that they are the first company to develop a driverless car that operates with the help of reinforcement learning. Since autonomous driving is done in a partially observable environment, reinforcement learning algorithms and approaches developed in game environments like Okey can be benefited in self-driving car domain.

Secondly, robotics is as important application area for RL studies and production lines are heavily dependent on robots especially in big industries. And robotic applications keep gaining more importance in the era of fourth industrial revolution that we are in. Smart robots are needed to operate relentlessly to optimize

business operations and lower operating costs. For example, Lorica (2017) states that American company Bonsai develops AI for tuning machines used in complex industrial systems to replace expert human operators with the help of RL applications.

Healthcare is another important area of application for RL because according to Shortreed and her colleagues (2011), learning treatment policies in medical sciences share the same nature of receiving feedback based on actions taken. Clinical trials are being held while discussing the risks and ethical aspects of the implications and thus there are various potential applications of RL in medicine such as medication dosing, optimization of treatment policies for chronic diseases, usage of medical equipment.

Furthermore, as it is shown in the work by Mao and his colleagues (2016) that RL can be used to automatically learn to allocate and schedule computer resources to waiting jobs in order to optimize resource allocation and minimize the average job slowdown.

RL proves to be a good fit when there are dynamically changing streams of information like news. As it has been shown in Zheng and his colleagues (2018), RL can be used to recommend news articles to website visitors to increase the click rate.

Finally, there is vast amounts of effort spent in the financial sector for RL applications, especially to automate financial stock trading via creating intelligent trading agents. As it has been portrayed by Srinivasan (2018), software-giant corporate IBM built a complex system on their DSX platform which makes financial trades with the power of reinforcement learning. The model they developed is trained with historical stock price data using stochastic actions at each time step and the reward function is calculated based on the profit or loss for each trade. Similarly,

Lorica (2017) points out that, JPMorgan Chase uses an RL-based system for optimal trade execution. Their system aims to execute trading orders as fast as possible with the optimum price calculated. So, the insights gained from the current study may be utilized to enhance these systems.



REFERENCES

- Aru, J., Korjus, K., Vicente, R., Kuzovkin, I., Aru, J., Tampuu, A., ... Matiisen, T. (2017). Multiagent cooperation and competition with deep reinforcement learning. *Plos One*, *12*(4), e0172395. <https://doi.org/10.1371/journal.pone.0172395>
- Bard, N., Foerster, J. N., Chandar, S., Burch, N., Lanctot, M., Song, H. F., ... Bowling, M. (2019). *The hanabi challenge: A new frontier for AI research*. Retrieved from <http://arxiv.org/abs/1902.00506>
- Brown, N., & Sandholm, T. (2019). Superhuman AI for multiplayer poker. *Science*, *2400*(July), eaay2400. <https://doi.org/10.1126/science.aay2400>
- Buşoniu, L.; Babuska, R.; De Schutter, B. (2011). *A comprehensive survey of multiagent reinforcement learning*. *38*(2), 156–172. <https://doi.org/10.1007/978-94-007-1162-4>
- Campbell, M., Hoane, A. J., & Hsu, F. H. (2002). Deep Blue. *Artificial Intelligence*, *134*(1–2), 57–83. [https://doi.org/10.1016/S0004-3702\(01\)00129-1](https://doi.org/10.1016/S0004-3702(01)00129-1)
- Diddigi, R. B., Kamanchi, C., & Bhatnagar, S. (2019). *Solution of two-player zero-sum game by successive relaxation*. Retrieved from <http://arxiv.org/abs/1906.06659>
- Endicott, S. (2017). *Game applications of deep neural networks*. Retrieved from https://github.com/jeffheaton/t81_558_deep_learning/blob/master/README.md
- Foerster, J. N., de Witt, C. A. S., Farquhar, G., Torr, P. H. S., Boehmer, W., & Whiteson, S. (2018). *Multi-agent common knowledge reinforcement learning*. Retrieved from <http://arxiv.org/abs/1810.11702>
- Foerster, J. N., Farquhar, G., Afouras, T., Nardelli, N., & Whiteson, S. (2018). Counterfactual multi-agent policy gradients. AAAI 2974–2982. Retrieved from <https://arxiv.org/abs/1705.08926>
- Heinrich, J., & Silver, D. (2016). *Deep reinforcement learning from self-play in imperfect-information games*. Retrieved from <http://arxiv.org/abs/1603.01121>
- Hu, J., & Wellman, M. P. (1999). Multiagent reinforcement learning in stochastic games. *ICML 1999*. Retrieved from https://pdfs.semanticscholar.org/7ce1/4dbb9add4d9656746703babd00d8f765b22a.pdf?_ga=2.155249275.1723981294.1568874307-315415511.1564661226
- Lorica, B. (2017). Practical applications of reinforcement learning in industry. Retrieved from <https://www.oreilly.com/radar/practical-applications-of-reinforcement-learning-in-industry/>

- Lytvynova, K. (2019). Reinforcement learning explained: Overview, comparisons and applications in business. Retrieved from <https://www.topbots.com/reinforcement-learning-explained-business-applications/>
- Mao, H., Alizadeh, M., Menache, I., & Kandula, S. (2016). Resource management with deep reinforcement learning. *HotNets 2016 - Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, 50–56. <https://doi.org/10.1145/3005745.3005750>
- Moravcik, M., Morrill, D., Bard, N., Davis, T., Waugh, K., Johanson, M., & Bowling, M. (2017). DeepStack: Expert-level artificial intelligence in no-limit poker. *Science*, 513(May), 1–32. <https://doi.org/10.1126/science.aam6960.1>
- Neto, G. (2005). *From single-agent to multi-agent reinforcement learning: foundational concepts and methods*. Retrieved from <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.412.9511&rep=rep1&type=pdf>
- Nowe, Ann & Vrancx, Peter & De Hauwere, Yann-Michaël. (2012). Game theory and multi-agent reinforcement learning. 10.1007/978-3-642-27645-3_14.
- Perez-Uribe, A., & Sanchez, E. (1998). Blackjack as a test bed for learning strategies in neural networks. *IEEE International Conference on Neural Networks - Conference Proceedings*, 3(May), 2022–2027. <https://doi.org/10.1109/IJCNN.1998.687170>
- Schaeffer, J., Lake, R., Lu, P., & Bryant, M. (1996). Chinook: The world man-machine checkers champion. *AI Magazine*, 17(1), 21–29. <https://doi.org/10.1609/aimag.v17i1.1208>
- Sethy, H., Patel, A., & Padmanabhan, V. (2015). Real time strategy games: A reinforcement learning approach. *Procedia Computer Science*, 54, 257–264. <https://doi.org/10.1016/j.procs.2015.06.030>
- Shortreed, S. M., Laber, E., Lizotte, D. J., Stroup, T. S., Pineau, J., & Murphy, S. A. (2011). Informing sequential clinical decision-making through reinforcement learning: An empirical study. *Mach Learn.*, 84(1-2), 109–136. doi:10.1007/s10994-010-5229-0.
- Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., ... Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587), 484–489. <https://doi.org/10.1038/nature16961>
- Silver, D, Schrittwieser, J., Simonyan, K., Nature, I. A.-, & 2017, U. (2016). Mastering the game of go without human knowledge. *Nature*, 550(7676), 354.

- 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*, *362*(6419), 1140–1144. <https://doi.org/10.1126/science.aar6404>
- Srinivasan, A. (2018). Reinforcement learning: the business use case, part 2. Retrieved from <https://medium.com/inside-machine-learning/reinforcement-learning-the-business-use-case-part-2-c175740999>
- Takaoka, Y., Kawakami, T., & Ooe, R. (2017). A fundamental study of a computer player giving fun to the opponent. *Journal of Computer and Communications*, *06*(01), 32–41. <https://doi.org/10.4236/jcc.2018.61004>
- Tesauro, G. (1995). Temporal difference learning and td-gammon. *ICGA Journal*, *18*(2), 88–88. <https://doi.org/10.3233/ICG-1995-18207>
- Wai, H.-T., Yang, Z., Wang, Z., & Hong, M. (2018). *Multi-agent reinforcement learning via double averaging primal-dual optimization*. (NeurIPS). Retrieved from <http://arxiv.org/abs/1806.00877>
- Zheng, G., Zhang, F., Zheng, Z., Xiang, Y., Yuan, N. J., Xie, X., & Li, Z. (2018). DRN: A deep reinforcement learning framework for news recommendation. *[WWW2018]Proceedings of the 2018 World Wide Web Conference*, *2*, 167–176. <https://doi.org/10.1145/3178876.3185994>