DEEP REINFORCEMENT LEARNING IN ATARI GAMES

By

Andre Leonardo Angkawijaya

A Thesis
Submitted to the Faculty of Computing
President Univesity
in Partial Fulfillment of the Requirements
for the Degree of Bachelor of Science
in Information Technology

Cikarang, Bekasi, Indonesia

October 2019

DEEP REINFORCEMENT LEARNING IN ATARI GAMES

By

Andre Leonardo Angkawijaya

Approved:

_____

Dr. Tjong Wan Sen, S. T., M. T.
Thesis Advisor

ABSTRACT

Technologies development grew fast in this era. The flow of information has never become this fast in the human history. Data are precious as it will help the business to win the customer. Data are utilized to analyze the customer needs, enabling the company to know the customer behavior and they could also predict the upcoming trend in their business. Processing the big amount of data, clustering, and analyze them is one of what *Artificial Intelligence* (AI), a manmade intelligence, can do for human.

AI development does not stop in there. The autonomous self-driving car created by Tesla, the autonomous mechanical hands in the manufacturing companies, the smart Boston Dynamics robots, and a lot of creation was created and inspired by a popular machine learning algorithm, *Reinforcement Learning* (RL). Studying and implementing this machine learning algorithm should be able to enrich students' knowledge in AI and help them got the concept of AI in the complex real world problem.

TABLE OF CONTENTS

LIST OF TABLES

TABLE Page

LIST OF FIGURES

FIGURE Page

CHAPTER I

INTRODUCTION

The introduction consists of the Background, Problem Statement, Research Objectives, Scope and Limitation, Research Methodology and Thesis Outline. The Background explains the current condition of the technology and information accessibility and the main problem that is created by the accessibility. The Problem Statement describes concisely some issues that is solved by the application.

1.1 Background

To prosper the quality of human's life, human began to develop a man-made intelligence, or what we usually called *Artificial Intelligence* (AI). The long development goal of AI is to achieve the ability for the machine to think and act both rationally and humanly in solving any intellectual human task, which is called *Artificial General Intelligence* (AGI). In this era, the recent development of AI is in creating an AI that could both think and act rationally.

Google DeepMind and OpenAI are companies that shows AI potential in solving problem that can be trained in simulated environment. RL is utilized by these companies to train an expert agent that outperforms humans in game. The agent created by DeepMind, the *AlphaGo Zero*, is able to defeat the 18-time world champion Lee Sedol in the game of Go [7]. OpenAI agent is also able to defeat the three best Dota 2 player in

the world in 1v1 match and it puts a tough battle in five bots versus five players mode [5].

By this achievement, the author deducts that using game as the environment to train an agent is the first step in solving complex real-world problem. For example, researcher that would want to create a self-driving car could create an agent which excel in playing racing game. Strategic games also exist in the business such as the strategy to wins marketing, stock exchange, etc.

Seeing the potential of RL, in this study, the author implements *Reinforcement Learning* (RL) algorithms to create an agent that is excel in playing Atari games. The training results will be an agent that is excel in playing different kinds of Atari games environment.

1.2 Problem Statement

AI will become a powerful business tools that could help people to win at business. In order to studies the recent surge of AI trends in business, the author intent to studies and compares different kinds of reinforcement learning algorithm which can train an agent to learn and interact within the specified environment to reach a specified goal.

1.3 Research Objective

This study is conducted with the following objectives:

- To implement reinforcement learning as the machine learning algorithm in creating an agent that could outperform human in Atari games.
- To analyse the algorithm performance in the simulated environment to achieve the best result.
- To provide a reference material for students who are interested in this field that would become a base for a more suitable learning process to solve real world problem.

1.4 Scope and Limitation

This study was conducted using Q-learning algorithm (and Deep Q Network algorithms) in the Atari games environment. The aspects looked into are the implementation of the algorithm and the agent's performance in different environments.

1.5 Methodology

The author uses the Waterfall Model of software development process. In this model, the whole software development process is divided into four phases where the outcome of one phase will become the input for the next phase. The four phases in this model are:

- **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System Design** − the requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- **Implementation** − with inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

1.6 Thesis Outline

The thesis consists of seven chapters, which are:

1. Chapter I: Introduction

   Introduction consists of Thesis Background, Problem Statement, Research Objective, Scope and Limitation, Methodology, and Thesis Outline.

2. Chapter II: Literature Study

   Literature Study describes the theoretical basis of references and guidance in the thesis creation.

3. Chapter III: System Analysis

   System Analysis describes the analysis of the program – its behavior and function. It consists of System Overview, Hardware and Software Requirement, Use Case Diagram, Use Case Narrative, and Activity Diagram.

4. Chapter IV: System Design

   System Design describes the definition of the program's architecture, components, and modules. It defines User Interface Design, Physical Design, Data Design, and Class Diagram of the program.

5. Chapter V: System Implementation

System Implementation describes how the application is implemented. It consists of User Interface Development and Application Details.

6. Chapter VI: System Testing

   System Testing contains the testing documentation of the application. Included here are Testing Environment and Testing Scenarios, along with the results.

7. Chapter VII: Conclusion and Future Work

   This chapter contains conclusion of the research. It also describes possible future improvements in section Future Work.

CHAPTER II

LITERATURE STUDY

2.1 Artificial Intelligence

A man made intelligence. Similar with human, machine receives inputs, calculates, and then show the predictions of the input. carving an intelligence into machine needs an iteration of learning process which is called Machine Learning (ML). In this section, the author describes four popular ML methods which called Supervised, Unsupervised, Semi-supervised, and Reinforcement learning [4].

2.2 Machine Learning

Material previously published by the candidate may be included, although if used in the body of the thesis, it must be integrated into the text of the document. Such previously published material must have been originally written as part of the candidate's degree program.

2.2.1 Supervised and Unsupervised Learning

Supervised learning (SL) uses labelled/named data to trains the agents to predicts something, for example, the agent is trained with a labelled fruit images to be able to differentiates fruit's name when it receives a fruit image. In the Unsupervised Learning (UL), the agents are trained with an unlabelled data to find the pattern and classify the provided data. Market research, social network analysis, and data clustering are the

example of the agents who used this training method. Agents trained using Semi-supervised Learning have the same purposes as SL, however it receives both labelled and unlabelled data.

2.2.2 Reinforcement Learning

RL applies the trial and error learning method, where the agents learn the consequences of their known actions in a specific environment [1]. At the end of their actions, the state of the environment is evaluated. The agents are given a reward according to the environment's state which can be either positive or negative, the positive reward shown that the agent's actions satisfy our requirement whereas the negative reward do the opposites.

2.2.2.1 Reinforcement Learning Algorithm

OpenAI research in RL through various papers line up a nearly accurate taxonomy of algorithms in modern RL as shown by Figure 1 below. The author intend to implement the DQN which stands for Deep Q Network algorithm to train his agent.

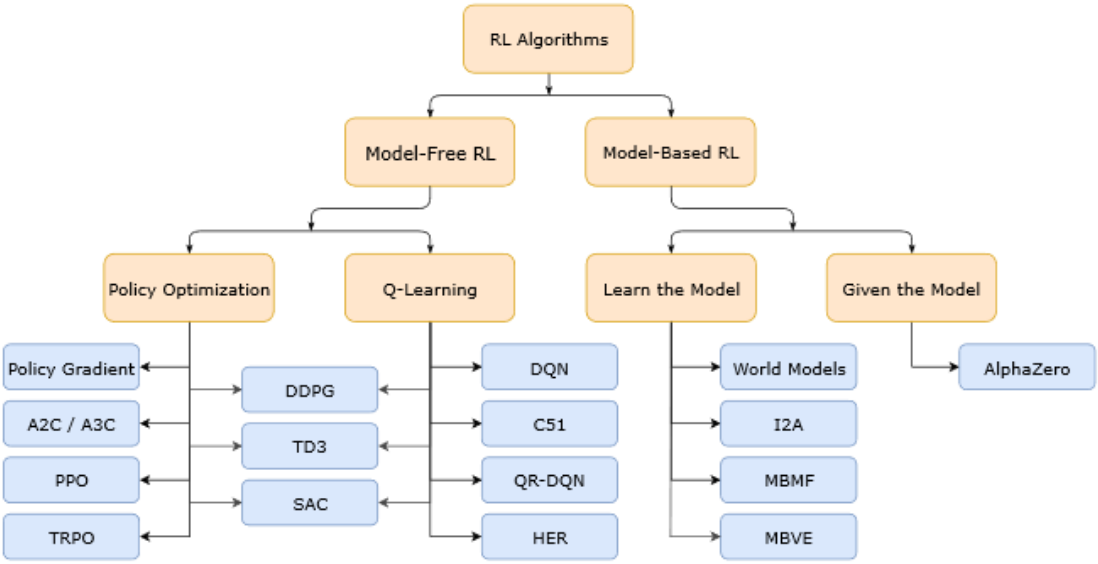

Figure 1. A non-exhaustive, but useful taxonomy of algorithms in modern RL. Reprinted from Part 2: Kinds of RL Algorithm, in OpenAI Spinning Up, 2018, Retrieved September 12, 2019, from https://spinningup.openai.com/en/latest/spinningup/rl_intro2.html.

2.3 Atari Games Environment

The uses of games environment as the means to simulated AI learning progress is a popular practice in training AI, especially to trains the AI that will take on a risky task. For example, the intelligence mechanical hand robot which have task on moving heavy or fragile object. If the training were to be conducted in real life, the business will only suffer a lot of damage through the training process. A lot of product or placeholder will be wasted just to train the hand's capabilities and the hand itself could be damaged through a lot of iterations of learning process.

Thus, game is utilized as it simulates real-life environment which can be used to trains AI to minimalize costs and risks in order to achieve greater task. The author utilizes M. G. Bellemare, Y. Naddaf, J. Veness, & M. Bowling (2013) creation of Arcade

Learning Environment (ALE) which is a dedicated simple object-oriented framework for hobbyists and AI researchers to developed AI agents using Atari games as shown by the Figure 2 below. Additionally, the author uses OpenAI Gym, which is an enhanced toolkit for creating an agent trained by RL which uses ALE.

Figure 2. Gym Retro screenshots collage showing Atari and Sega games environment. Reprinted from Gym Retro, in OpenAI, 2018, Retrieved from https://openai.com/blog/gym-retro/. Copyright 2018 by OpenAI. Reprinted with permission.

2.4 Related Works

The most popular paper that the author finds throughout his research in this topic is the paper called “Playing Atari with Deep Reinforcement Learning” by Mnih et al. [3] The team introduces the implementation of Q-Learning variant into Deep RL algorithm, which are Deep Q-Network and Deep Q-Network Best, to create a single agent that is capable to achieve the highest score from seven different Atari Games. In their report, Mnih’s team included their reinforcement learning algorithm which is shown in the Figure 3 below.

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Figure 3. The algorithm for Deep Q-Learning with experience replay by Mnih et al. (2013)

The Deep Q-Learning with experience replay algorithm will be the main algorithm that the author will try to implements and tweaks in his work. Additionally, the other related works that the author found is the bachelor thesis entitled Deep Q-Learning with Feature Exemplified in Atari Pacman by Meo [2]. He experiments between different training setup and algorithm to benchmark and retrieve the most satisfying performance of his algorithm.

Mnih and Meo's works are the base of the author's thesis. In his work, the author would like to take a single main algorithm and then compares it with different type of existing reinforcement learning algorithm. Then, the author would tweak different scenario in his setting and record it to search for the satisfying performance that his algorithm implementation can deliver.

CHAPTER III

SYSTEM ANALYSIS

3.1 System Overview

This thesis is intended to implement Deep Q Network Algorithm to train an agent in playing Atari game. The system will train the agent to the point that the agent is able to exploit the environment to achieve the best score as fast as possible. The main objective of the system is to create a system that were able to outperform human in playing games.

3.2 Hardware and Software Requirement

Listed below are the software and hardware that are needed to developed this application:

- Personal Computer

  A personal computer is where the application resides and be developed, from its earliest stage to its deployment. This application is developed on a PC running 64-bit Windows 10 education.

- JetBrains PyCharm Professional 2019.2.1

PyCharm is a Python Integrated Development Environment (IDE) that can be downloaded for free with a student email. Developed by a Czech company, JetBrains, PyCharm provides easy code analysis, jupyter notebook supports, an integrated unit tester, integration with version control systems (VCS), and supports web development with Django.

- Microsoft Office

  Microsoft Office application, specifically Microsoft Word is used in the making of the application's documentation.

- Python 3.7.4

  Python 3.7.4 is the programming language that is used in the development of the application. Some APIs listed below are used to support the development of the application:

  - Matplotlib

    A python 2D plotting library which can produce a variety of plot and histogram which are interactive and cross platform.

  - Tensorflow

    Google's open source machine learning platform that could deliver an easy yet robust machine learning model.

- Gym

  OpenAI's toolkit for developing and comparing a variety of reinforcement learning algorithm. Using Gym also allows the researchers to create and define their own environment to test their algorithm.

3.3 Functional Analysis

The system will be divided into a smaller functional elements which describes the overall system workflow. The system is divided into three sub parts which are preparing the environment, training the agent, and update the agent. The functional analysis diagram could be seen from the Figure 4 below.

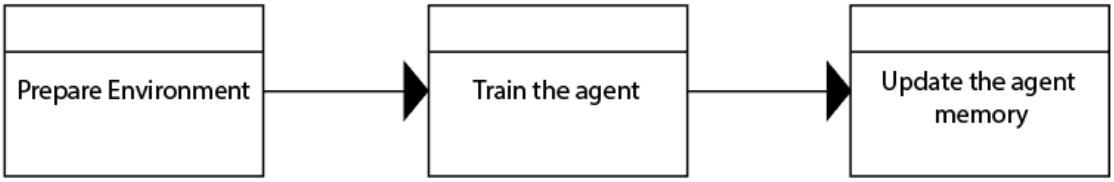

Figure 4. The Functional Analysis Diagram

3.4 Use Case Diagram

Use case diagram is a diagram describing dynamic behavior or technical concept of a system. It defines:

- Actors

The internal or external factors interacting with the system depicted by stickman

- Use Cases Nodes

  A sequence of actions that provide something of measurable value to an actor depicted by circle.

- Associations Arrow

  Dotted arrows which explains a node dependency toward another node.

- Unidirectional Associations Arrow

  Arrow which explains a node association toward another figures.

The application's use case diagram is shown in Figure 5.

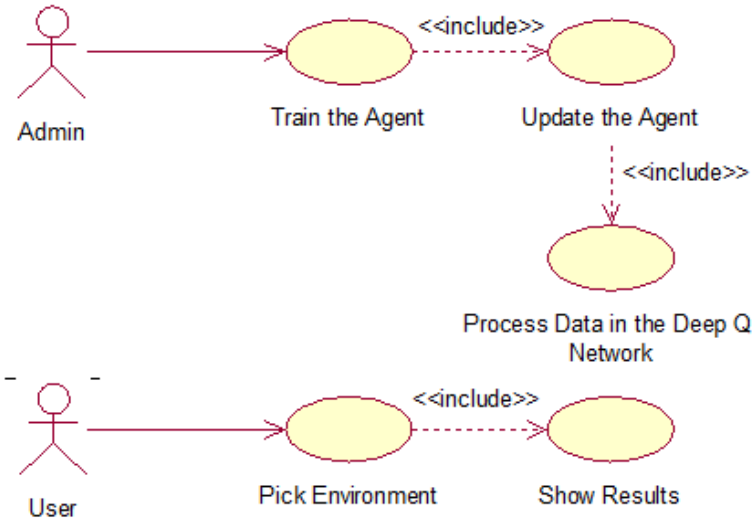

Figure 5. The Use Case Diagram

3.5 Use Case Narrative

A detailed textual representation of the sequence of events occurred during the interaction between an actor and the system in use case diagram will be explained in the narratives. The narrative is a table that will explains the prerequisites, conditions, expected results, alternative scenario, exception, goal, and post condition of a use case node. Use case narrative aims is to clarify the system's behaviors from the early stages.

Table 1. Use Case Narrative of Train the Agent Node

Use Case Name	Train the Agent
Goal in Context	The accuracy of the Agent is increased
Primary Actor	The admin
Secondary Actor	None
Precondition	The environment must exist
Trigger	The admin initializes the environment
Scenario	1. The admin initializes the environment 2. The admin starts the training process 3. The application train the agent to maximize the reward it could get 4. The training results are recorded and plotted into a chart
Alternate Scenario	None
Exception	The environment is not exist
Post Condition	The training data are saved

Table 2. Use Case Narrative of Process Data in Deep Q Network Node

Use Case Name	Process Data in Deep Q Network
Goal in Context	The environment data and the agent knowledge are updated
Primary Actor	The admin
Secondary Actor	None
Precondition	The agent is initialized
Trigger	The agent is taking an action that changes the environment
Scenario	1. The agent is taking an action that changes the environment 2. Both of the environment and agent data are processed into the Convolutional Neural Network 3. The agent knowledge base is updated from the retrieved data
Alternate Scenario	None
Exception	None
Post Condition	The agent got the knowledge from the past experience enables it to learn with more accuracy

Table 3. Use Case Narrative of Update the Agent Node

Use Case Name	Update the Agent
Goal in Context	The agent making a progress in every episode
Primary Actor	The admin
Secondary Actor	None
Precondition	The agent is initialized
Trigger	The training to a certain iteration is started by the admin
Scenario	1. The training is started by the admin 2. The agent is initialized with the default environment information 3. The agent takes random action in the environment 4. The action updated the environment which puts the agent in a new environment 5. The agent stores the knowledges and continue taking random action
Alternate Scenario	None
Exception	If the agent exploits the short-term reward and do not seeks a long-term reward, the agent would not get the best performance.

Post Condition	The agent made a progress in making a decision until it get the best performance or until it reaches the end of the iterations.

Table 4. Use Case Narrative of Pick Environment Node

Use Case Name	Pick Environment
Goal in Context	The environment is initialized
Primary Actor	The user
Secondary Actor	None
Precondition	The application is running
Trigger	The user chooses a specific environment from the combo box
Scenario	1. The user chooses an environment 2. The environment is initialized
Alternate Scenario	None
Exception	None
Post Condition	The environment is shown to the user

Table 5. Use Case Narrative of Show Results Node

Use Case Name	Show Results
Goal in Context	The training results is shown to the user through a chart
Primary Actor	The user
Secondary Actor	None
Precondition	The user has chosen a specific environment
Trigger	The environment is initialized
Scenario	1. The environment is shown to the user 2. The results of the agent's training are shown through a chart
Alternate Scenario	None
Exception	None
Post Condition	The result chart is shown to the user

3.6 Activity Diagram

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. The diagram defines both concurrent and sequential activities which is used by the developers as a lineup of the technical flow of their works. This sub chapter will explain the activities diagram that are used in building this application.

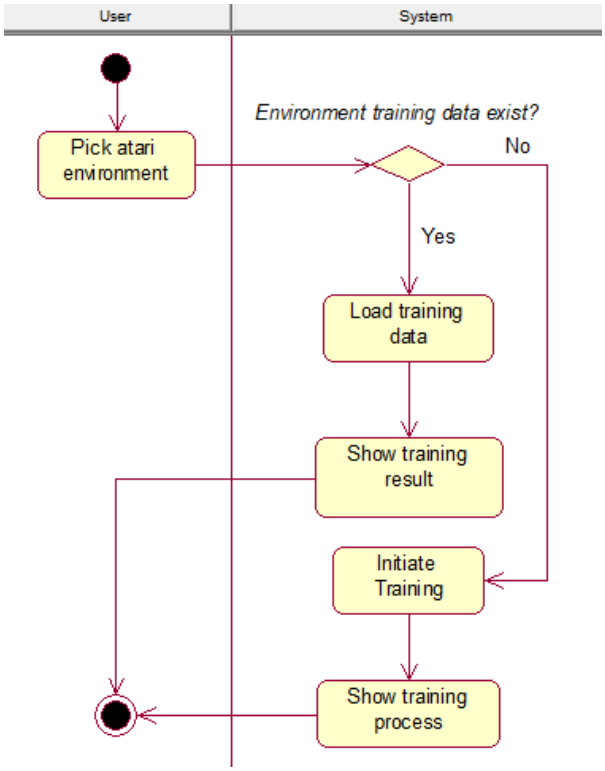

Figure 6. Pick Environment Activity Diagram

The Figure 6 above is derived from the *Pick Environment* node of the Use Case Diagram. The user who picks the environment will trigger the system to check whether a training data is exist in the environment. If the training already conducted, then the application will simply show the training result to the user. However, in the case where a training is not yet conducted, then the training will be initiated and the process will be visible to the user through the chart generated on the fly when the training is running. The *Initiate Training* node itself is explained in more detail through the Figure 7 below.

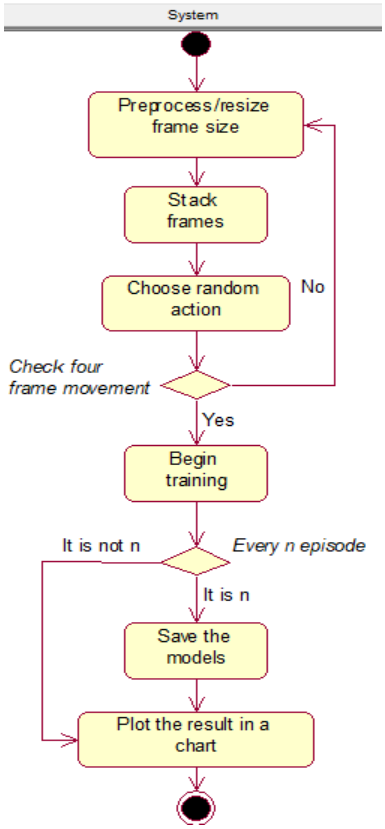

Figure 7. Initiate Training Activity Diagram

The system will be the sole actor of the *Initiate Training Activity Diagram*. To begin the training, the system will first *Preprocess/Resize* the environment's frame to delete meaningless details (extra spaces, scoreboard, etc.). Then, the next process is to *Stack Frames* so that the system could get the sense of motion. Next, the system will initiate the agent to take a *Random Action* to update the environment. If the system did not detect an update to the frames in *n-steps*, it will restart the process all over again from the *Preprocess* node. Otherwise, the system will *Begin the Training Session* that will be furtherly explained in the next section that is depicted by Figure 8. The last steps that will end the training session are to *Save* the trained model for each defined step and *Plot* the training results into the chart.

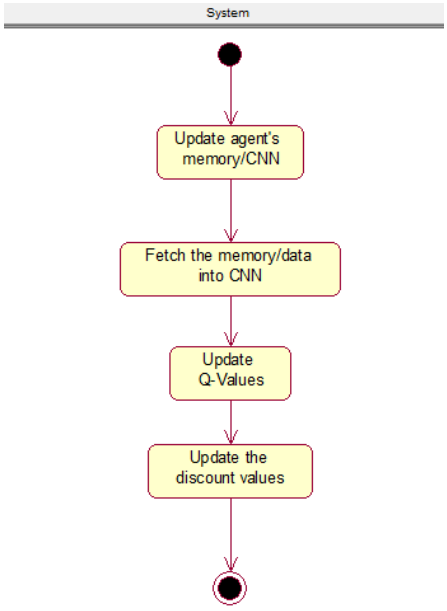

Figure 8. Begin Training Activity Diagram

The *Begin Training Activity Diagram* which is extended from the Figure 6 is the last activity diagram in this application. The training is started by initializing or *updating the agent's memory*. Then, the knowledge that the agent got will be *fetched into the Convolutional Neural Network (CNN)*. After that, the *Q-Values will be updated* by the new reward function and the *Discount Values* will also be updated by the CNN.

CHAPTER IV

SYSTEM DESIGN

System Design is the process of defining the architecture, components, modules, and data of a system to satisfy the specified requirements. It can be considered as the intermediate stage between System Analysis and Product Development where the analyses that have been made in the previous chapter are now visualized in detail. The application comprises of four sections: user interface design, physical design, data design, and class diagram.

4.1 User Interface Design

User interface design is one of the important components of any application, be it a desktop or a web application. The user interface will allow users to use the application easily. As a desktop application, this application will one interface for the sake of simplicity which is shown by Figure 9 below.

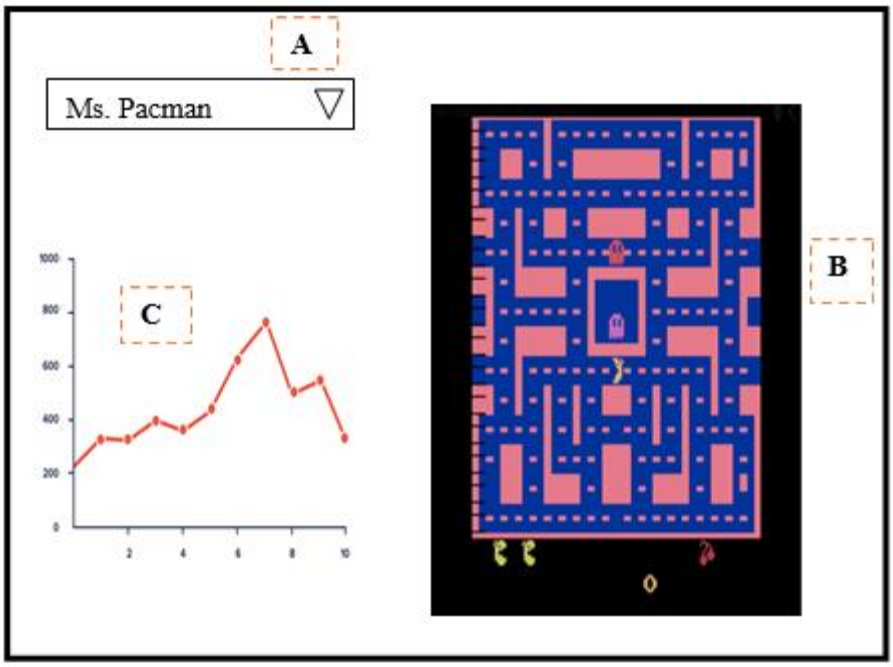

Figure 9. The User Interface Design

Shown by the figure are three main elements, that are represented by A, B, and C, of the User Interface design for this application. The user interface consists of the Atari environment chooser combo box, Atari environment UI, and a line chart which are respectively shown by the orange dotted-line box A, B, and C.

4.2 Physical Design

Physical Design defines the minimum requirements of software and hardware used in development process to ensure that the created application runs without problems. The Table 6 and 7 below show respectively the software and hardware requirements to runs this application.

Table 6. Software Requirements

No	Field	Description
1	Operating System	Windows 10 Education
2	Programming Language	Python
3	Program Development	JetBrains PyCharm Professional 2019.2.1
4	Documentation	Microsoft Word 2019

Table 7. Hardware Requirements

No	Field	Description
1	Processor	Minimum requirement: Intel Core i series
2	Memory	Minimum requirement: 2GB of RAM
3	Monitor	Minimum requirement: Resolution 1280x720
4	Hard Drive	Minimum requirement: Free space of 100 MB

4.3 Data Design

Database is not needed in this application as the application purpose is only to train the agent to be excel in a specific game environment. Therefore, there only exist three data that will be generated to save the checkpoint for the training, evaluation, and taken action in the format of checkpoint (.ckpt) file. Table 8 below depict the data design for the application.

Table 8. The Data Design

No	Filename	Content Description
1	training_checkpoint.ckpt	The training checkpoint for a defined number of iterations
2	evaluation_checkpoint.ckpt	The evaluation checkpoint for a defined number of iterations
3	next_action_checkpoint.ckpt	The next action checkpoint for a defined number of iterations

4.4 Class Diagram

A class diagram, as specified in UML, is a static diagram that describes the classes of a system. Classes here are classes in Object Oriented Programming context which displayed with its attributes, methods, and relations with other classes (if exists). Class diagram of this application is shown in Figure 10.

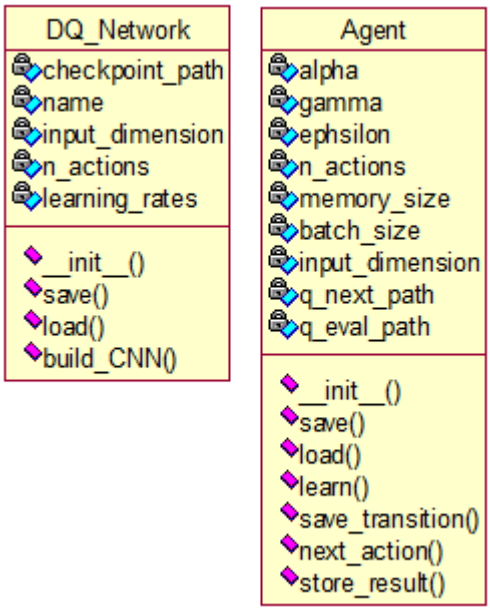

Figure 10. The Application Class Diagram

Two classes, *DQ_Network* and *Agent*, are used in this application. As shown by the figure above, no relation exist between classes in the diagrams because all of the classes are used in module. The further explanation of each class will be explained below:

a. DQ_Network Class

The DQ_Network Class main purpose is to build a Convolutional Neural Network sfor the agent. There exist five properties to define this class which are:

- checkpoint_path

  The *checkpoint_path* is used to store the path that will be utilized to either save or load the checkpoint file.

- name

  The *name* that will be referred from the agent's name.

- input_dimension

  The *input_dimension* is used to define the dimension of the input layer for the Convolutional Neural Network.

- n_actions

  The *n_actions* stands for a number of actions that the agent could take in the specific environment.

- learning_rates

  The *learning_rates* is a variable which hold a floating-point value from 0 to 1 that defines the learning speed of the agent.

Four methods of the *DQ_Network class* are explained below:

- __init__

The *__init__* method overwrite the class's default initializer. It is utilized to initialize the class's properties along with the TensorFlow session and the *build_CNN* method.

- build_CNN

  The *build_CNN* method is the main method of this class. The neural network is configured inside this method.

- save

  The *save* method is utilized to save the trained models in checkpoint (.ckpt) format.

- load

  The *load* method is utilized to load the previously trained models in checkpoint (.ckpt) format.

b. Agent Class

The *Agent Class* represent the agent that the neural networks trained. Agent stores the environment and the taken action to feed it to the neural networks to determine the next taken action and also evaluates the previous action. Hence, the agent will be able to create a better decision in taking an action. There exist nine properties that is created for the *Agent Class* which are:

- alpha

  The *alpha* variable indicates the learning rate, which is stored in floating-point format, that will be used in the Q-values calculation.

- batch_size

The *batch_size* indicates the

- epsilon

  The *epsilon* value stores a floating-point value between 0 and 1 to determine the agent characteristics to either explore or exploit the taken action.

- gamma

  The *gamma* value stores a floating-point value between 0 and 1 to determine the importance of the future reward that will be given to the agent.

- input_dimension

  The *input_dimension* stores the input layer dimension that will be feed to the network.

- memory_size

  The *memory_size* stores the memory size that the agent could hold.

- n_actions

  The *n_actions* stands for a number of actions that the agent could take in the specific environment.

- q_eval_path

  The *q_eval_path* stores the path string to save the checkpoint for the evaluation.

- q_next_path

  The *q_next_path* stores the path string to save the checkpoint for the next taken action training.

Seven methods in the *Agent Class* are explained below:

- __init__

  The *__init__* method overwrite the class's default initializer. It is utilized to initialize the class's properties.

- save

  The *save* method is utilized to save the trained models in checkpoint (.ckpt) format.

- load

  The *load* method is utilized to load the previously trained models in checkpoint (.ckpt) format.

- learn

  The *learn* method is the main method where the CNN will be run and the Q-values will be updated.

- save_transition

  The *save_transition* method is utilized to save the taken action and the environment states to the agent' memory.

- next_action

  The *next_action* method is used to determine whether the agent should explore or exploit the environment.

- store_result

  The *store_result* method is used to store the training result and plot itinto the chart.

REFERENCES

[1] Andrew, N. G. (n.d.). Part XIII Reinforcement Learning and Control [PDF document]. Retrieved from CS229 Stanford Edu Website: http://cs229.stanford.edu/notes/cs229-notes12.pdf

[2] Meo, R. (n. d.). *Deep Q-Learning with Feature Exemplified by Pacman* (Bachelor's Thesis, Hamburg University of Applied Sciences). Retrieved from http://edoc.sub.uni-hamburg.de/haw/volltexte/2018/4168/pdf/Roland_Meo_BA_Thesis.pdf

[3] Mnih et al. (2013, December 19). *Playing Atari with Deep Reinforcement Learning* [PDF]. Retrieved from https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf

[4] Musumeci et al. (2018). An Overview on Application of Machine Learning Techniques in Optical Networks [PDF document]. Retrieved from https://arxiv.org/pdf/1803.07976.pdf

[5] OpenAI Five. (n.d.). Retrieved from https://openai.com/five/

[6] Pfau et al. (2018). *Gym Retro screenshots collage showing Atari and Sega games environment*. [Screenshots collage]. Retrieved from https://openai.com/blog/gym-retro/

[7] The Google DeepMind Challenge Match. (n.d.). Retrieved from https://deepmind.com/alphago-korea