A NEOTERIC EVALUATION OF DEEP REINFORCEMENT LEARNING ALGORITHMS UTILIZING GAME CONCEPTS

2024
Master Thesis
COMPUTER ENGINEERING

Zabiullah ALI SHER

Thesis Advisor
Assist. Prof. Dr. Nehad T.A RAMAHA

A NEOTERIC EVALUATION OF DEEP REINFORCEMENT LEARNING ALGORITHMS UTILIZING GAME CONCEPTS

Zabiullah ALI SHER

Thesis Advisor
Assist. Prof. Dr. Nehad T.A RAMAHA

T.C.
Karabük University
Institute of Graduate Programs
Department of Computer Engineering
Prepared as
Master Thesis

KARABÜK
June 2024

I certify that in my opinion the thesis submitted by Zabiullah ALI SHER title "A NEOTERIC EVALUATION OF DEEP REINFORCEMENT LEARNING ALGORITHMS UTILIZING GAME CONCEPTS" is fully adequate in scope and in quality as a thesis for the degree of Master of Computer Engineering.

Assist. Prof. Dr. Nehad T.A RAMAHA .....
Thesis Advisor, Department of Computer Engineering

This thesis is accepted by the examining committee with a unanimous vote in the Department of Computer Engineering as a Master of Science thesis. June 14, 2024

Examining Committee Members (Institutions)		Signature
Chairman	: Asst. Prof. Dr. Alaa Ali HAMEED (IU)	…….
Member	: Asst. Prof. Dr. Nehad T.A RAMAHA (KBU)	…….
Member	: Asst. Prof. Dr. İsa AVCI (KBU)	…….

The degree of Master of Computer Engineering by the thesis submitted is approved by the Administrative Board of the Institute of Graduate Programs, Karabük University.

Assoc. Prof. Dr. Zeynep ÖZCAN .....
Director of the Institute of Graduate Programs

"I hereby declare that this thesis is the result of my own work and all information included has been obtained and expounded in accordance with academic rules and ethical policy specified by the institute. Besides, I declare that the statement, results, materials, not original to this thesis have been cited and referenced literally."

Without being bound by a particular time, I accept all moral and legal consequences of any detection contrary to the aforementioned statement.

Zabiullah ALI SHER

ABSTRACT

M. Sc. Thesis

A NEOTERIC EVALUATION OF DEEP REINFORCEMENT LEARNING ALGORITHMS UTILIZING GAME CONCEPTS

Zabiullah ALI SHER

Karabük University
Institute of Graduate Programs
Department of Computer Engineering

Thesis Advisor:
Assist. Prof. Dr. Nehad T.A RAMAHA
June 2024, 64 pages

This study developed an innovative Deep Reinforcement Learning (DRL) approach applied to the Chrome Dino Run. Unlike traditional methods using pixel-based images, this study transforms the state representation to include x and y coordinates and the width and height of the game obstacles. This departure aims to streamline learning, improve DRL efficiency, and reduce computational costs. By emphasizing essential features in the state representation, such as cacti and birds' precise coordinates, the model learns more efficiently and generalizes better across various game scenarios. Integrated with DRL algorithms like Deep Q-Network (DQN), the proposed method demonstrates competitive performance while being computationally more efficient.

The abstraction of the game state to essential elements, such as object coordinates, not only enhances the efficiency of the learning but also provides a more interpretable representation. This interpretability fosters a deeper understanding of the learned

policies, shedding light on the decision-making process of the DRL agent. By concentrating on key features directly influencing decision-making, the study suggests that abstract state representations simplify the learning process and enhance the generalization capabilities of the DRL model.

Keywords **:** Reinforcement Learning, Double DQN, Chrome Dino Run, Deep Neural Networks, Obstacle Coordinates, Exploration-Exploitation, Bellman Equation.

Science code **:** 92431

ÖZET

Yüksek Lisans Tezi

OYUN KAVRAMLARINDAN YARARLANAN DERİN PEKİŞTİRMELİ ÖĞRENME ALGORİTMALARININ NEOTERİK BİR DEĞERLENDİRMESİ

Zabiullah ALI SHER

Karabük Üniversitesi
Lisansüstü Eğitim Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı

Tez Danışmanı:
Dr. Öğr. Üyesi Nehad T.A RAMAHA
Haziran 2024, 64 sayfa

Bu çalışma, Chrome Dino Run'a uygulanan yenilikçi bir Derin Güçlendirme Öğrenme (DRL) yaklaşımı geliştirdi. Piksel tabanlı görüntüler kullanan geleneksel yöntemlerden farklı olarak bu çalışma, durum temsilini x ve y koordinatlarını ve oyun engellerinin genişlik ve yüksekliğini içerecek şekilde dönüştürüyor. Bu ayrılma, öğrenmeyi kolaylaştırmayı, DRL verimliliğini artırmayı ve hesaplama maliyetlerini azaltmayı amaçlamaktadır. Model, kaktüsler ve kuşların kesin koordinatları gibi durum temsilindeki temel özellikleri vurgulayarak daha verimli öğrenir ve çeşitli oyun senaryolarında daha iyi genelleme yapar. Deep Q-Network (DQN) gibi DRL algoritmalarıyla entegre edilen önerilen yöntem, hesaplama açısından daha verimli olmasının yanı sıra rekabetçi bir performans sergiliyor.

Oyun durumunun nesne koordinatları gibi temel unsurlara soyutlanması yalnızca öğrenmenin verimliliğini arttırmakla kalmaz, aynı zamanda daha yorumlanabilir bir

temsil sağlar. Bu yorumlanabilirlik, öğrenilen politikaların daha derinlemesine anlaşılmasını teşvik ederek DRL temsilcisinin karar verme sürecine ışık tutar. Karar almayı doğrudan etkileyen temel özelliklere odaklanan çalışma, soyut durum temsillerinin öğrenme sürecini basitleştirdiğini ve DRL modelinin genelleme yeteneklerini geliştirdiğini öne sürüyor.

Anahtar Kelimeler : Takviyeli Öğrenme, Çift DQN, Chrome Dino Koşusu, Derin Sinir Ağları, Engel Koordinatları, Keşif-Kullanım, Bellman Denklemi.

Bilim Kodu **:** 92431

ACKNOWLEDGEMENT

I extend my heartfelt gratitude to my beloved mother, whose unwavering support, encouragement, and sacrifice in every step of my life have been the cornerstone of my academic journey. Her boundless love and guidance have been my source of strength throughout this endeavor.

I would also like express my sincere appreciation to my advisor, Assist. Prof. Dr. Nehad T.A Ramaha, for his irreplaceable guidance, expert knowledge, and unwavering encouragement. His mentorship has been instrumental in shaping my research and academic growth, and I am truly grateful for his dedication and support.

Special thanks are owed to my kind-hearted and gentlemanly uncle, Ataeulah Afzali, for his generous economic support, which alleviated financial burdens and allowed me to focus wholeheartedly on my studies and research.

I am deeply grateful to Turkey and Karabük University for providing an enriching academic environment and essential resources for conducting this research. I extend my appreciation to all the professors and faculty members whose expertise and guidance have contributed to my academic development.

Additionally, I am thankful to all the professors, and my family who have provided assistance, feedback, and encouragement during the course of my master's thesis.

This thesis would not have been possible without the support and contributions of all those mentioned above. Thank you from the bottom of my heart.

LIST OF CONTENTS

Page

Page

Page

LIST OF FIGURES

Page

LIST OF TABLES

Page

ABBREVIATIONS

DRL : Deep Reinforcement Learning

RL : Reinforcement Learning

AI : Artificial Intelligence

DQN : Deep Q-Network

DDQN: Double Deep Q-Network

DNNs : Deep Neural Networks

CNNs : Convolutional Neural Networks

NNs : Neural Networks

DL : Deep Learning

CHAPTER 1

INTRODUCTION

1.1. BACKGROUND

Nowadays, deep learning (DL) and reinforcement learning (RL) concepts and algorithms are considered very important tools for developing intelligent systems. When these two powerful concepts are merged together, then we get the Deep Reinforcement Learning (DRL) paradigm that becomes very powerful in the artificial intelligence (AI) field [1,2]. After combining their principles, we make our machine capable of making intelligent decisions in complex environments. At its core, the model we create has neural networks (NNs) that can easily process the decision-making strategy; thus, it allows the agent to navigate very flexibly in passing through a complex and dynamic scenario.

Chrome Dino Run is a web browser-based game; a T-rex dinosaur navigates an infinite journey in a desert, facing obstacles. This game aims to jump over the cacti and pass through birds to survive and get great scores [3]. The agent we defined in this DRL system is responsible for acting as the player; the assigned player is to navigate through infinite obstacles that are randomly generated. Two fundamental actions are available in this game, such as jumping or doing nothing. Our agent in this dynamic and infinite environment aims to make itself capable of identifying and executing optimal strategies to pass through. It enables the agent to be flexible in navigating this diverse number of obstacles in this infinite environment and continuously maximize its overall scores. The DRL system we designed will transform the Chrome Dino Run game to compete itself as an AI game mastery.

1.2. MOTIVATION

The motivation we are breaking up in this research is to address the challenges of agents training in visually complex and dynamic situations; the prime characteristic in this case study is the environment of the Chrome Dino Run game, which is taken into action.

Image-based methods in DRL often struggle with some burden of high computations because of the high-dimensional input spaces [4]. Specifically, overcome tasks that are visually complex and dynamic. Here, the motivation is to contribute to this already developed algorithm that excels in the intuition of learning strategies and does more efficiently with high computational environments for the Chrome Dino Run game.

1.3. OBJECTIVE

This research's main goal is to push the DRL forward to deal with specific challenges that produce visually complex and dynamic environments [4,5], This research uses the Chrome Dino Run game environment as a testing ground to reach this goal. Therefore, this research is trying to achieve a bunch of unified and different ideas; each idea contributes to a deep exploration of the capabilities of DRL intuition. To achieve the desired goal, this research has the following sub-objectives:

1. To utilize an algorithm excelling the computational efficiency terms without compromising its performance, we use Double Deep Q-Network (DDQN) capabilities to adeptly navigate the Chrome Dino Run game. Successfully delivering the balance between learning efficiency and high performance is one of the crucial outcomes of the used algorithm achievement.

2. To explore the proposed algorithm generalization in real-time scenarios. Beyond gaming, we aim to explore the generalization of the proposed algorithm in real-time scenarios [6]. In this matter, we aim to ensure the adaptability of the DRL to all sorts of applications used for crucial decision-making in dynamic environments.

3. To highlight the proposed algorithm's potential contributions to the table beyond handling the specific Chrome Dino Run constraints in the game environment.

1.4. SCOPE OF THE STUDY

This study focuses on looking into different angles of DRL, particularly within the context of Chrome Dino Run game dynamics and visually complex environment. We will focus primarily on resolving the suggested DDQN algorithm and particularly how it works in the gaming environment, along with our contribution to this algorithm.

Our horizon widens here, demonstrating various ways of input representation, particularly in this study. Rather than the whole game frames, our decision is to focus on obstacle coordinates only, which means a deep dive into the different ways of making it happen, the advantages of this contribution, and facing limitations with this polished approach. This study aims to add some considerate contributions demonstrating a nuanced perspective dealing with DRL input representation strategies.

1.5. LIMITATION OF THE STUDY

While being a very good testing space for our DRL methodologies, the Chrome Dino Run game is an environment with straightforward approaches. The constraints of the game might not capture the complexity found in the real world’s more complex scenarios.

Although the elimination might need further research and technological enhancements, strategies are explored to tackle the problem. However, there are some issues that the suggested DRL system faces, like frame acquisition lag, which gets trickier, and the complexity of our agent responding consecutively.

Primarily, our research addresses challenges encountered within the gaming scene of Chrome Dino Run. Employing this in other games or real-world scenarios might need things to tweak the process to hold up.

CHAPTER 2

LITERATURE REVIEW

2.1. REINFORCEMENT LEARNING (RL)

RL is the core of AI, it represents to train an intelligent agent with adaptable capabilities in the dynamic environment, it has transformed in a powerhouse, showing off its flexibilities in all sort of applications [7], particularly in gaming world. Naturally its algorithm has the ability of picking up the optimal strategies by repetitive interactions in a complex environment covering a groundbreaking achievement.

In the RL world, the Q-learning update equation works as the basis for a repetitive fine-tuning strategy of the agent in choosing the action [8], we adapted it into the context of the Chrome Dino Run environment, this formula is guiding the agent's learning process. It allows the agent to figure out decisions with its interactions in the dynamic game environment. Here is the update equation which expressed:

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma \cdot max_{a'} Q(s',a') - Q(s,a)] \tag{2.1}$$

Here, $Q(s,a)$ is state-action pair of current Q-value, where the state s is our obstacle coordinates, and action a is the decision of agent. α is the agent's learning rate ($0 < \alpha \leq 1$), it is an essential parameter to determine the assigned weight of the new information. Each update is controlled by this magnitude, the smaller the value ensures the more conservative adjustments. r which illustrate the immediate reward that the agent obtained with an action a taken in a state s. This reward is given if the agent successfully navigates over the obstacles or a penalty (negative reward) for the agent's collisions. γ we could express this as the discount factor, that influences the balance between immediate and future rewards. In our Chrome Dino Run dynamic environment, the bigger γ would emphasizes the importance of long-term

consequences. Finally, $max_{a'}Q(s', a')$ says the maximum Q-values which is achieved in the state s' subsequent after the agent takes action a. This part covers the anticipation of agent's cumulative future rewards.

In this repetitive process update, the mentioned equation which embedded within the Q-learning algorithm, it allows our created agent to learn functional learning of optimal strategies in playing the Chrome Dino Run game, our agent adapt the behavior according to the experiences and rewards which accumulated in the training epochs iterations.

In the gaming world, the basis of the technology is the RL [7], it shows off its value across various gaming environment. The previous works have shown the RL algorithm adaptability like an expert that dominates the classic games, and breaks the limits in the world of video games. The achievements of the RL in game were solid proof in how it is an expert at controlling the tricky decisions and picking up learning nuanced behaviors.

2.2. DEEP REINFORCEMENT LEARNING (DRL)

We believe the transition from RL to DRL constitutes an important turning point in our AI environment along with broad consequences. However, it has characterized by the seamless deep neural networks (DNNs) integration [9], which then brings a new era of capabilities, and also pushing the boundaries of previously feasible consideration in to our AI world.

The algorithm of DRL is just an extended version of RL since it typically uses DNNs for function approximations such as Q-value and policy. Therefore, it can work with high-dimensional state spaces because it uses neural networks (NNs) for its prediction, which makes it ideal for tasks like images and audios, which are sensory raw inputs. Since then, the updated equation for the Q-value, particularly in DQN [10], is written below.

$$Q_{target} = \ rewards \ + \ (\gamma \cdot max_{a'} Q_{target\_next} \cdot (1 - dones)) \tag{2.2}$$

Here, the Q_{target} is belongs to target's Q-value for the current pair of state-action. Because with this value we aim to estimate by the time our agent is learning. Where, the $reward$ is immediate reward as obtained after taking the action within the current state. As if, successfully avoiding an obstacle would positive whereas, negative for colliding with an obstacle. Now, γ is our discount factor, a value between 0 and 1 provided to determine the future rewards importance. In order that, the trade-off-between immediate and future rewards influences. Consequently, $max_{a'} Q_{target\_nex}$ is the maximum Q-value among all action a' possibilities within the next state s'. In addition, it shows the estimated future reward for the best possible action of the agent. Finally, $(1 - dones)$ is binary indicator 0 or 1 that shows either the episode has terminated or not terminated. In fact, we could use to avoid future rewards consideration as if the episode has terminated.

2.2.1. Advancements Beyond Traditional RL

As if the basic RL algorithm perfectly thrives in some particular areas, we believe it hardly works with areas that involve high-dimensional input spaces. So, integrating DNN's potentials makes it possible to generalize a great learning process across huge states even though it provides a huge and flexible framework and introduces a paradigm shift in the RL concept, which we can call DRL.

Hence, there are a huge number of challenges in the high-dimensional input spaces [11], but the DRL concept stands out with its huge capabilities as an innovative light in the AI world. NNs bring depth and complexity to the learning process of the agent giving it the power to expose the complex relationship between data. This concept enables our agent to make wise decisions in a dynamic environment. So, this transformation of combining RL with the DNN concept forms the foundation to address the complexities that are encountered in the Chrome Dino Run game.
We decided to choose DRL because of its learning spatial pattern ability, it works flexible with any dynamic environment, if we talk about its decision-making policy, it

is very efficient. Hence, our objective is to select a sophisticated concept, to have the ability of learning techniques and also it should be consistent and customizable for our unique features of our chrome dino run game.

2.3. RELATED WORK

When we mix the DRL with gaming environments, the result it produces is fascinating and extraordinary. we believe that we could say that it outperforms frequency as a human-level performance in most certain environments. Thus, to leverage our DRL strategic decisions, if to be more specific, it is the DDQN algorithm for our Chrome Dino Run game challenges, which arises for tracking the record and handling its dynamic, and visually diverse gaming environments.

2.3.1. DRL Architectures in Gaming

In the gaming environment, I believe researchers have extensively worked with a variety of DRL architectures and here the goal was to achieve a human-level, or higher than that, superhuman performance. As we illustrated in Figure 2.1, here the CNNs have emerged a popular task [5,12], since it is a very efficient solution for visual processing of the input it receives from the game frames. Therefore, the visual illustration, reminds us in order that the capacity of NNs which is used to extract the hierarchical features and also recognize the patterns, this emphasizes the pivotal role that makes our agent successful in visually demanding games environment that we could use. Although, we could say that it could reinforces the significance of using the CNNs to explore along with our DRL architecture in the high-level gaming performance.

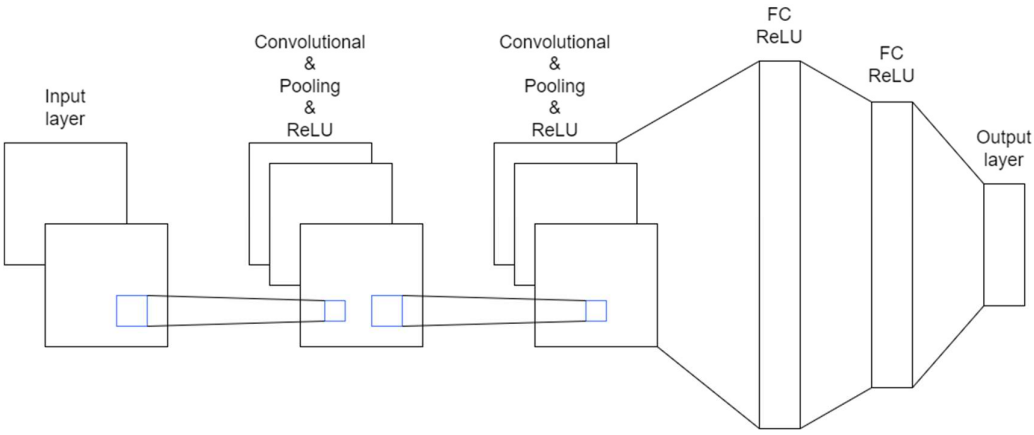


Figure 2.1. Illustrates the DRL process which is enhanced by the algorithm of CNNs, here game frame goes through the convolutional and pooling layers and also using the ReLU activation function, it extracts all hierarchical features [13].

Several demonstrations of DRL ability to perform better than humans across the game spectrum include the agents that master in the Atari games [14], navigate through complex 3D environments, and make themselves higher in real-time game strategies. Therefore, these achievements show the generalization and flexibility of the DRL algorithm, so these capabilities make it a potential candidate for addressing challenges that are exposed by dynamic and visually rich scenarios in the game.

So now, the technique of transfer learning, which is exemplified in Figure 2.2, can play a vital role in enhancing the strength of the agent's generalizing ability across distinct gaming environments. Here, the concept includes training a CNN agent on an exactly different game and then transferring the knowledge acquired to definitely another game [15]. Hence, this method effectively leverages the learned policy, which we can significantly apply to another game and reduce the training time so that it could excel at a drastic gaming task where it could adapt swiftly to a totally new environment.

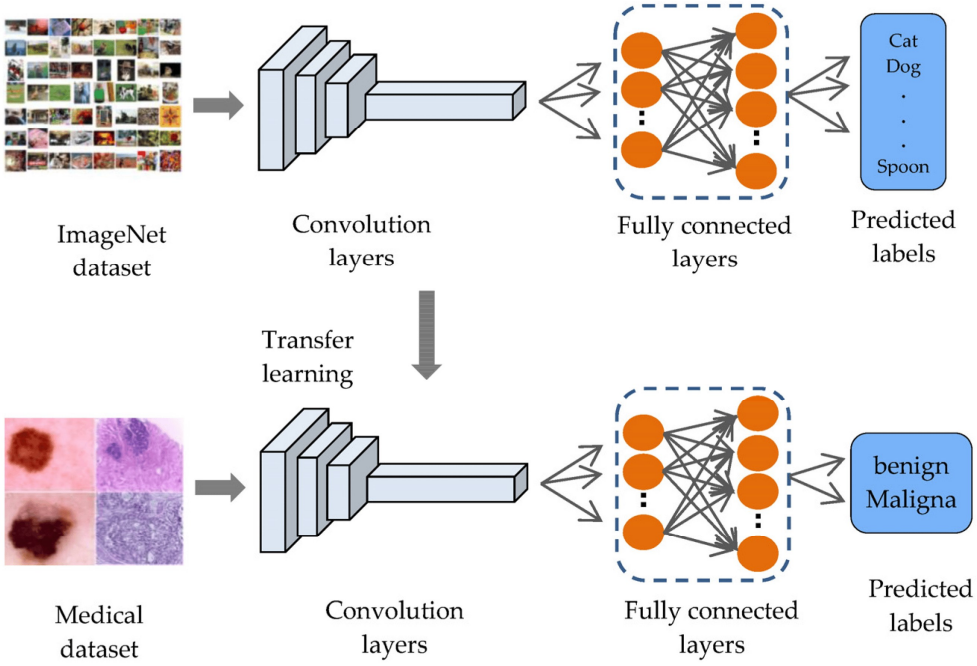


Figure 2.2. A pretrained CNNs illustration of transfer learning, an initial training dataset is transferred to a medical dataset showing off CNNs flexibility and efficiency [16].

Demonstration of transfer learning basically give the model a capacity to generalize its knowledge within different scenarios, since this method leads our agent to more efficient and effective character in the new environment.

2.3.2. Challenges and Opportunities

Hence, DRL successfully doing very well in game scenarios, but there are challenges that remained, to be specific it is in balancing the concept of exploration and exploitation, sometimes there are challenges in sparse rewards, we believe there are challenges in ensuring the robustness of learned policies. Of course, there are researches going on that focuses to resolve these problems and also find options for further refinement and innovations in gaming environments of DRL application.

2.3.3. Chrome Dino Run Previous Approaches

Although, this game generates new challenges regularly based on its infinite and high-dimensional state space. Hence, this game, with its continuously changing environment and as well as its real-time decision-making requirements, it needs flexible and adaptive learning strategies. Perceiving each aspect of its mechanics is very critical to develop an effective DRL strategies.

In 2020, Divyanshu Marwa [17] applied DRL to the Chrome Dino Run game with the algorithm of DDQN to train the agent that she designed, and the state representation is the entire game frame that is rendered for the agent, so to extract features from the image, she used CNN's algorithm. Training lasted for 8 hours across 2647 episodes, during which the agent could achieve a maximum score of 2800.

In a similar endeavor, to play the Chrome Dino Run game Lustina Ivanova in 2021 [18], she used the standard DQN algorithm with the help of reinforcement learning, represented the entire game frame as the state, and used the CNN algorithm for nuanced feature extraction of the image. Her agent training lasted for about 24 hours and 1,980,000 steps, with a result of 258 maximum scores.

Table 2.1. Previous approaches methods and other details on Chrome Dino Run Game.

Reference	**Divyanshu Marwa** [17]	**Lustina Ivanova** [18]
Year	2020	2021
Algorithm	Double Deep Q-Network (DDQN)	Deep Q-Network (DQN)
Method	Convolutional Neural Network	Convolutional Neural Network
Episode	2,647	N/A
Time	8 hours	24 hours with 1,980,000 time steps
Max Score	2,800	258

In contrast to the traditional image-based state representation, we used the DDQN algorithm to design our system and also obstacle coordinates (x, y, w, h) for the state representation of our agent. Unlike the abovementioned methods [17,18], we did not use the CNN algorithm for image feature extractions. We trained our agent for almost 3 hours and 2000 episodes so that our agent achieved a maximum score of 2064. Our

method that we used is explained in clear detail in Chapter 4. This approach proved to be not only effective in achieving competitive performance but also efficient in terms of computation, owing to the simplified state representation based on obstacle coordinates.

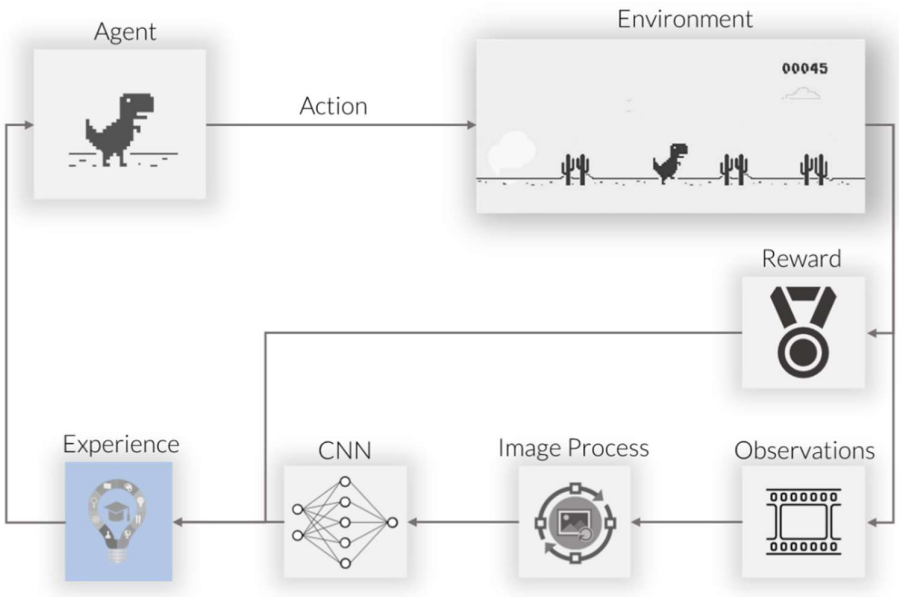


Figure 2.3. Snapshots serve as the state through the CNN for feature extractions [19].

A very large amount of procedure has been discovered, based on the previous attempts when addressing this game's challenges [17,18,20]. Therefore, all the traditional attempts on this were just using CNNs frequently giving the entire game frame as its input where it aims encompassing an effective visual information [3]. But the problem here is all of the above strategies requires a huge amount of computation so that is it faces inefficiencies in computation, this prompts that we should explore some kind of alternative representation of the state. Where we clearly illustrated visually in Figure 2.3, we explained in detail the intricacies and limitations of previous approaches in Chapter 3.

In previous approaches [23], [24], because they used the entire game frame as the state representation, which caused the reward methods to be straightforward. Typically, if the agent crashes into an obstacle, the agent will receive a negative reward, and if the

agent successfully passes through an obstacle, then the agent will receive a positive reward. This reward method is directly tied to the outcome of the game.

In our approach, because we used obstacle coordinates as the state representation, so that it allows us to design a more nuanced reward method. Since using the alternative state representation, I have more control over the reward mechanism, that allows me to make more customized and efficient reward methods. In Chapter 4 of this research, we discussed our reward methods to play the Chrome Dino Run game, which highlights its efficiency and effectiveness to guide our agent through the learning process.

CHAPTER 3

PROBLEM STATEMENT

If we go through the Chrome Dino Run game environment demanding, and think it within the framework of DRL, so we could get spot point of the problem where it needs a very critical evaluation of problem in the hand of DRL. There by, this section of thesis, extracts the complexities encountered and the difficulties which is found, so that we can identify the limitations in the existing approaches where we can re-structure the state representation in an easy and computation efficient way.

3.1. CHALLENGES IN HIGH-DIMENSIONAL SPACES

As we get deeper into DRL architecture, where the gaming environments, including our exampled game (Chrome Dino Run), this will introduce some kind of challenges, which are very critical, in case of managing its high-dimensional state spaces. Hence, this high computational load [12,21], where we are processing the entire game frames is going to lack the learning process time for the agent. Perceiving this problem, will prompts a way of exploring an alternative state representation, where the main goal should be the relevant extraction of features that the agent needs and this will maintain a manageable computational burden.

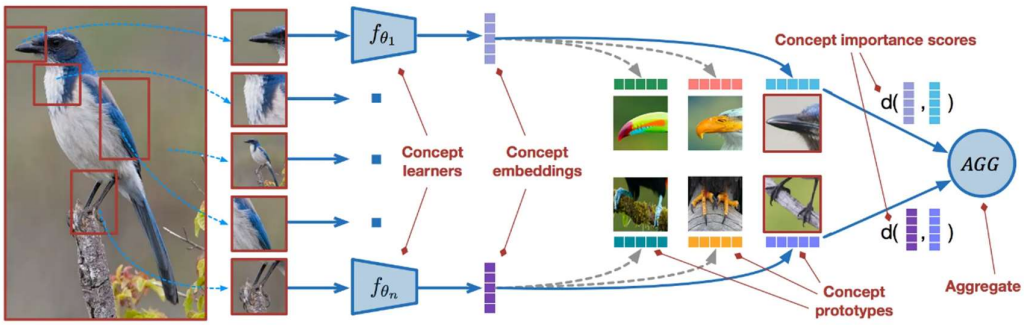


Figure 3.1. Illustrates the complexity and richness of high-dimensional space [22].

3.1.1. Challenges in Chrome Dino Run

Our exampled game environment, provides some kind of challenges, where it makes difficult than other DRL applications. As explained above this environment involves, infinite and also high-dimensional state space [12], so that our agent needs to make decision in the real-time that adds some layers of complexities. We are required to train the agent very efficiently that could navigate this dynamic environment, and agent should have a comprehensive knowledge of those challenges, which we are going to explore in the subsequent chapters.

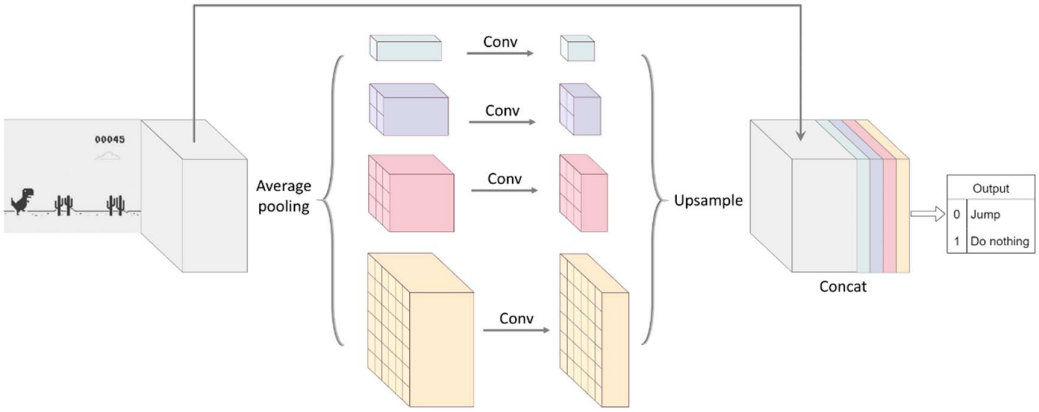


Figure 3.2. Architecture of Dino Chrome Run CNN [20]

The requirement of real-time decision-making really depends on the complexity of the game, as illustrated in Figure 3.2, where it consists pooling and convolutional layers, so that it processes the input from the game environment frames. And the final output which we are getting would be the agent's decision, either jump or do nothing. By the time the obstacles expose itself in the screen dynamically, so here our agent should be very careful on splitting second decision, and here it should provide a sense of urgency to its learning process. So, this means that to train a proficient agent that perceives the time-sensitive navigation.

3.2. LIMITATIONS OF EXISTING APPROACHES

In fact, with the existing approach on the Chrome Dino Run game, especially when we are using CNNs, and pass the entire game frame as input introduces computational

inefficiencies for the agent. However, processing a large number of visual data consecutively for the agent in the real-time can strain the learning process in a serious situation. Hence, this part of our thesis sheds some lights on the limitations for the traditional methodologies which are existed, and we are going to establish our own innovative strategies of modifying the representation of the state.

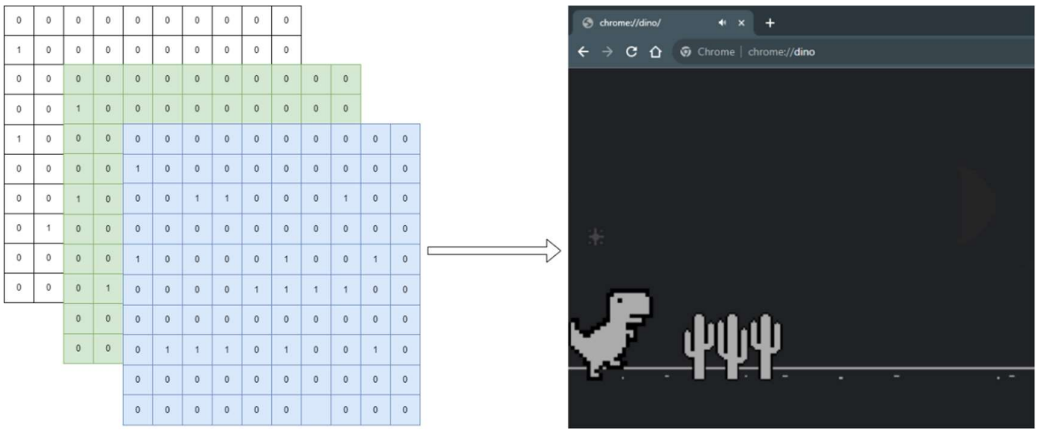


Figure 3.3. Illustrates our image-based state representation which is traditionally used in our exampled game environment.

As compared to the association of computational inefficiencies, where the process gets the entire game frame as its input, as illustrated in Figure 3.3, the existing approaches [23], which are demonstrated in our Chrome Dino Run game, are currently struggling with real-time learning problems. This pressure, which is caused by the large amount of visual data processing, will act as a catalyst for our unique tactics, which are explored in the subsequent sections, where we are only focusing on the modification of state representation to just enhance the efficiency and effectiveness of the agent.

Our motivation is the need to achieve greater efficiency, and we also need to enhance the learning dynamics, so here our approach gets inspiration from the previous efforts done by researchers, but with a unique carving path. We used DRL techniques; to be specific, we used the DDQN algorithm in our methodology. The difference here is that we strategically focused on obstacle coordinates as a very straightforward state representation. Hence, this is our intentional move that aims to improve the flexibility of the algorithm and how it could act in a dynamic and visually diverse environment for our example problem that renders to the algorithm.

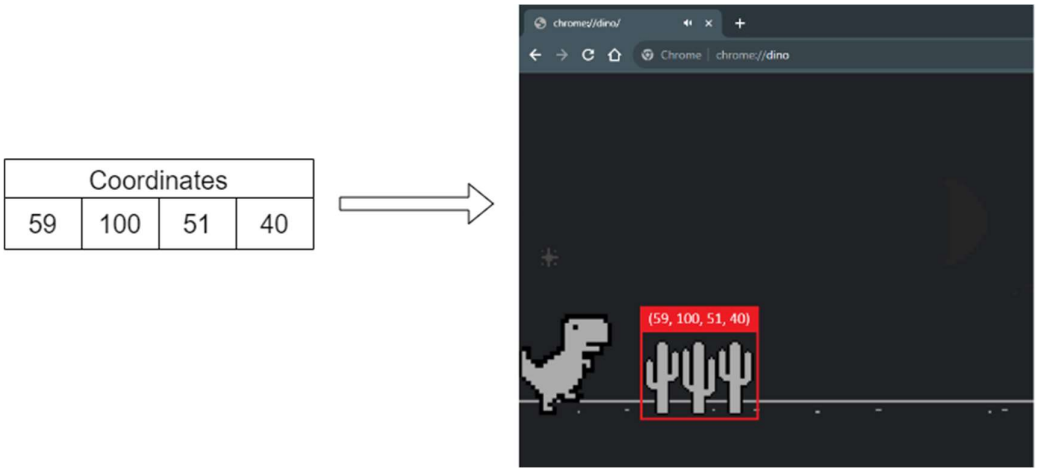


Figure 3.4. Illustration of state representation as coordinates (x, y, width, height).

Moreover, we have carefully reviewed the previous approaches of the DRL algorithm in games and decided to contribute a distinct approach of state representation to the Chrome Dino Run game. Indeed, we acknowledge that the value that is passed as input to CNNs is the entire game frame, so that our methodologies give the CNNs a more streamlined input that is precisely needed by our agent. In this research, we prioritize obstacle coordinates as our main focus, using the DRL approaches along with this contribution. Therefore, we deliberately chose the underscored commitment just to enhance the adaptability of our algorithm so that it could behave smoothly in the dynamic and visually diverse challenges that the game renders, as depicted in Figure 3.4.

3.3. RATIONAL FOR STATE REPRESENTATION MODIFICATION

Through our journey designing this DRL system, the main concern is the pursuit to create a carefully balance between the information richness and the efficiency of computation, so that requires us to understand state representation that we render to our agent. Therefore, the abovementioned challenges and limitations that we identified clears the concern that why we must subtly modify the source that how our agent should perceive and process its environment adeptly. This material explains the basis for our decision to change state representation, which in turn provides theoretical support for developing and implementing the algorithm that we propose as shown on Figure 3.4. Finding a better way to learn is inseparable from adjusting strategically

how the agent interacts with and comes to understand Chrome Dino Run's various states.

By the time we go through the chapters, we are going to move our focus on unfolding the state representation, give a depth detail of our consideration, and mainly describe the innovative strategies that we contributed into the DRL algorithm. Therefore, as we explored the complexities of the state representation, our goal is to illuminate a streamlined path moving through efficient and effective learning in the world of Chrome Dino Run dynamics.

CHAPTER 4

METHODOLOGY

In this chapter, we are going to deep dive into our contribution methods, which are for using the DRL algorithm we make our agent master on the Chrome Dino Run game. We are going to start off our journey by concentrating on the DRL in depth, and we will shed light on the critical role of the DRL in our intelligent agent training and how to interact with dynamic like environments. Based on the foundation we designed, we are going to navigate through the very detailed considerations involved and we restructure the state representation. The approach that we have chosen, is mainly gets its basis around obstacle coordinates, so this basis, on which we aimed, shows an intentional approach to speeding up the learning process.

4.1. PROGRAMING LANGUAGES AND FRAMEWORKS

Python pulls off all the tricks in our grand plan implementation, and makes sure the entire system's frameworks and libraries do their responsibilities in an organized manner. Each of these technologies comes together, creating a powerful dynamic system, it is crafted to tackle the thrown challenges of the Chrome Dino Run game.

4.1.1. Python

The programming language Python, lies at the heart of our setup, managing the entire workflow of our code by versatility and expressiveness, like a real expert in the real-world. It makes the perfect pick by its readability and ease of use, the way it crystal clears complex functionalities allows us to articulate. Whether diving into the complexity of RL or navigating into the game interaction, with an elegance and finesse python spin the narration and tells the story [24].

4.1.2. Native Python Libraries

With mixing all frameworks which are explained in this section, the old perfect Python libraries also show off their versatility in this research. Covering the tricks of random number or other useful features from these libraries, contributes to our flexible and rich implementation of the system. Each native library in this system plays a unified role, working efficiently in a group to create stack of functionalities and methodologies in sync together [24].

4.1.3. Selenium

Selenium library is acting in the center spotlight, connecting our Python code with JavaScript in the browser, playing the role of a bridge. To chat smoothly with Chrome Dino Run game it makes seamless interaction with a paramount role. Leading the code line as an action in the browser, selenium is the manager for the journey of our agent through the digital realm, grabbing screenshot from canvas, and restarting the game to reset the environment for the agent altogether interactions [25].

4.1.4. NumPy

NumPy, in Python it is the numerical operations expert, it could be said that NumPy is the backbone that does all the calculations to help our agent making its decision, and navigates the numerical turns and complexities smoother in the RL world. Handling arrays and metrices efficiently is just our agent needs to smoothly prowess the computation in decision-making [26].

4.1.5. Pillow and OpenCV

The team up of Pillow and OpenCV libraries dons are the artisans, and are visual elements fine-tuning of our system. The library expert in computer vision OpenCV, to process the image in a smooth way, makes the agent sure that the images are handled efficiently and smoothly [27], while the Pillow library, takes the preprocessing

screenshot of the game frame from canvas [28]. Together, they get ready the entire visual input for our learning agent, makes sure all ok.

4.1.6. PyTorch

In the deep learning terms, the emergence of PyTorch could be like an expert, a dynamic computational graph is expected which can syncs up seamless alignment of the Chrome Dino Run environment [29]. PyTorch can easily bring the DDQN model to the agent's life easily, which our agent learns and adapts and use its game wise in each timestep in its life.

4.2. OVERVIEW OF DEEP REINFORCEMENT LEARNING

The intention of training our agent was that it could play the Chrome Dino Run game smoothly, so here we should select our RL algorithm very carefully. Therefore, an algorithm which barely noticeable to manage the complexities of the state representation and also it is already built in PyTorch library is the DDQN algorithm, this would be a very good example to lighthouse our virtual journey to lead.

Since the Chrome Dino Run game has a dynamic environment, so that we have to choose a pursuit reinforcement learning algorithm for our agent's training, and that should be a very careful selection. Thus, after pivotal research, we came up with the DDQN algorithm [30], which is the best lighthouse that guides our agent through the virtual journey it has. This algorithm exposes its sophisticated potentials within the competitive environment of the library of PyTorch, so that it could provide a very powerful framework that our agent could easily learn optimal strategies by the time it starts interacting with the complex environment of the Chrome Dino Run game.
Meanwhile, we are going through the equation that the learning process of our model incorporates with both, immediate reward and as well as estimates of future rewards, altogether encapsulated. Hence, it will help the agent's adaption of its Q-value over the episode which shown below:

$$Q_{target} = r + (\gamma Q_{target\_next}(1 - \ dones)) \tag{4.1}$$

Here, when we mentioned Q_{target} it is the updated Q-value for our current state-action pair, r in this concept is the immediate reward which is obtained within the environment, γ is the representation of discount factor, that influences the weight of the future reward that we assigned, similarly Q_{target_next} this guy just tells us the Q-value which obtained for the next state, and this will reflect an estimate of the future reward for our agent, finally $dones$ is an indicator of binary values (0 or 1) it shows whether the environment's episode is terminated.

4.2.1. Double Deep Q-Network (DDQN)

We picked DDQN over regular DQN because DQN has a very prevalent issue known as overestimation bias, which usually affects the learning stability and performance of our agent, and DDQN eliminates this common issue, which is its foundation. Since we have to develop two unique Q-networks for our DDQN method, which is the notion of this algorithm, one of the Q-networks Q_{local} is only for present state Q-values and the other Q_{target} is just for future estimation Q-values. As a result, this crucial approach is a unique one that decreases bias overestimation while also improving the overall agent learning process.

Moreover, the inspiration of DDQN roots from the Bellman Equation, and that is a very basic reinforcement learning concept. As for the action function $Q(s, a)$ value, we express the Bellman Equation as follows:

$$Q(s, a) = R(s, a) + \gamma \sum_{s'} P(s' \mid s, a) max_{a'} Q(s', a') \tag{4.2}$$

The Bellman Equation is the main concept for our RL algorithm, where it encapsulates the Q-values $Q(s, a)$ dynamics, it expects the cumulative future rewards where it takes action a in the state s. Therefore, the above equation composes in many components with complex structure, $R(s, a)$ is our immediate reward it obtains this reward after it execute an action a in a state s, respectively γ which is our discount factor in concept

it just determines the future reward significance, $P(s'|s,a)$ is our probability transition it points to likelihood subsequent state s' transition if we give the current state s and also the current action a. Hence, this equation has the intention to optimize the decision-making, it calculates the maximum Q-value $max_{a'}Q(s',a')$ where a state s' succeeds in all sides of feasible action a'. We are going to use this carefully selected formulation that forms the basis of our agents understanding and managing of our RL scenarios.

Moreover, into the learning path of the agent our algorithm guide works as a foundation and with consistency. Hence, the Bellman Equation uses two distinct networks [31], this helps us the durable learning route with an insurance. While our local Q-network makes immediate decisions, and the target Q-network make next state decision, both these creates a full understanding of game environment.

4.3. DRL ALGORITHM SELECTION

The method we used for our reinforcement learning roots from the Bellman Equation which works as the leader for our Q-learning environment [32]. Specifically, this Bellman Equation controls for our Q-value updates so that our agent that navigates through dynamic environment in the Chrome Dino Run game is essential. We used the Bellman Equation in our implementation context like this:

$$Q(s,a) = R(s,a) + \gamma max_{a'}Q(s',a') \tag{4.3}$$

Here, $Q(s,a)$ is our Q-value when takes action a in a particular state s, we show our immediate reward as $R(s,a)$, respectively γ will represent our discount factor along these parts the $max_{a'}Q(s',a')$ will store the maximum Q-value for our subsequent state, and s' stores the next state.

Basically, in numerous cases, the silent lying of the bias caused by exaggerating haunts the RL. This conventional DDQN architecture is also very robust. However, it can sometimes be entrapped by the allure of inflating its Q-values especially when there is

noise and uncertainty abound. DDQN, a dedicated protagonist, opens new doors to our journey towards learning that are free from these dangers of bias it can do this.

4.3.1. Mean Squared Error (MSE) Loss Function

The loss function used in our environment is the MSE loss function which is expressed mathematically $MSE(y, \hat{y}) = \frac{1}{n}\sum_{i=1}^{n} (y_i - \hat{y})^2$, so basically this function calculates the average squared difference between each $\hat{y}_i$ which is the predicted value and our y_i that is the true target value [33,34]. The loss function we used is particularly useful for problems like regression, here the goal of this function in our project is to minimize the difference between values that are predicted against the value which are the actual values.

In our contribution, we used mse_loss function, and this function seamlessly integrated with computational graph library of PyTorch context, so that this method gives us the facility of backpropagation of the errors by the time our agent starts training. Therefore, our described mse_loss function will help us in the training process that it sets the stage, where it shows its crucial role that how it guides our neural network to push it towards of getting an optimal parameter adjustment. Moreover, we used mse_loss function in our contribution as like this:

$$loss = mse\_loss(Q_{expected}, Q_{target}) \tag{4.4}$$

In the above formula, $Q_{expected}$ is the Q-values that our local Q-network predicted, and Q_{target} is the Q-values which derived from the bellman equation that we used. Therefore, mse_loss function gets the predicted and also the expected Q-values and then calculates the mean squared difference between them, so that measuring the expectation and also the actual outcome of the agent during the learning process.

4.3.2. Soft Update

The architecture of the reinforcement learning we designed and contributed, here in this contribution the concept of soft update plays a pivotal role that stabilizes our agent learning process and also it helps the agent to enhance its adaptability in the dynamic world of Chrome Dino Run game. Which we expressed the soft update formula like this:

$$\theta_{target} = \tau \cdot \theta_{local} + (1 - \tau) \cdot \theta_{target} \tag{4.5}$$

In this formula θ_{target} this guy is our target model and θ_{local} is the parameters that we extract from the local model, τ this guy is the parameter that we defined and assigned its value, based on this value our target model gets updated from local model. Therefore, through this formula our target model gets integrated gradually from local model parameters, and also it ensures a very smooth and controlled transition in to our target model. Basically, the concept of soft update just prevents the target from sudden change of transition, in short, this concept encourages a strong stability to learning process and helps our DRL system in overall efficiency. Moreover, this concept helps our agent to minimize the overestimation bias, and the concept of overestimation bias is very common in DQN algorithm, so that this concept is really valuable that helps our agent a smoother integration by the time it starts training.

4.3.3. Epsilon-Greedy Action

In our DRL architecture particularly in decision-making, the concept of epsilon-greedy is a pivotal strategy [35], because it helps our agent to achieve a delicate balance between exploration and also exploitation. So that, our agent gets guide from this strategy to select its action during the learning process, and also it helps the agent to interact with the environment a very carefully approach.

$$\begin{cases} \arg\max(Q\text{—}values)\,, with\ probability\ 1- \in \\ random\ action, \qquad with\ probability \in \end{cases} \tag{4.6}$$

Where, $\in$ is the probability of the agent's exploration, it helps the agent select exploration over the exploitation. We can describe the Q-values as the estimated

expected future rewards that we have already recorded in each action. This useful approach helps our agent to exploit with a high-value actions that is in its memory and also maintain it exploration strategy based on this concept to adapt the dynamic nature of the environment and learn what to do.

4.3.4. Adam Optimizer

Optimizing our neural network parameters in the world of DRL architecture, it is a pivotal aspect because it influences our agent's learning process [36]. Here, in this contribution, the Adam Optimizer works as a pillar in this landscape, because it combines two very useful concepts like RMS prop and Momentum that helps the neural network parameters to modify its learning rate dynamically. Therefore, the mathematical formulation of this concept is just computes the first and second moment estimates by including them, hence it helps the agent to have an adaptive and efficient parameter during its training process.

Here, the process that updates the learning rate, it initializes the moment vectors m and v, it computes the gradients, and also loops through to update the estimates. The biased-corrected $\widehat{m}$ and $\hat{v}$ which is the first and second moment estimates is an account for the time-dependent nature of our algorithm. Finally, to update the parameters, this optimizer uses learning rate, the estimation of the correct moment, and also a constant which is a very small, this helps the optimizer to prevent numerical instability.

4.4. TRAINING PROCESS

In this contribution of DRL, we set 20 epochs of a revolutionary journey where each epoch contains 100 episodes to our agent experience the dynamic environment of this learning process. This is a well-thought plan that shapes, sharps and update the experience of our intelligent agent to bypass numerous obstacles which is offered by Chrome Dino Run game environment.

$$T_{agent} = \sum_{i=1}^{20} \sum_{j=1}^{100} (i \cdot j) \tag{4.7}$$

So that, T_{agent} is our agent that transforms itself across 20 epochs where each epoch is comprising of 100 episodes journey to be master in this game environment.

Moreover, every era in the training program is equivalent to a chapter in a story that is drafted. This show unfolds from epoch to epoch over 100 episodes with 20 epochs. Therefore, at each 100 episodes the chapter carefully avoid its title, through these episodes, we see the challenges and hardships that his handler encounters in this ever-changing world of Chrome Dino Run game. We will witness that as the agent devotes itself to the ambiguities within game rules, its understanding of them becomes more sophisticated and it draws gradually closer to optimal decision-making, throughout this period the agent himself subtly weaves through the web of actions that constitutes Chrome Dino Run.

It interacts with events, it draws lessons from observation, and it reads the context in ensembles of arias coexisting somewhat like plain song. Evaluation is often made in no way depending on game action, such as applause for success and attacks after an error to which something has led the score. These things greatly influence one's subsequent choices. Its and we can talk about being both sides of the agent's brain at once. Underlying context that creates such null concepts. The rewards, whether it is a round applause or foul play in the game, prompt notes of rhythm. Importantly, these changes where subsequent action will take place, a harmony of interplay like this keeps up the rate at which we learn, with each new thing learned fitting into an evolving we of knowledge.

In the Chrome Dino Run game, the learning process of our agent we can say that is a dynamic interaction over these episodes because our state, its action, and all its subsequent rewards will make its structure with decision-making. Let's put s as our state, a would be our action, r is the reward in each time step. Therefore, the rule that our agent uses to learn an optimal policy π would map the states to our actions, so that our agent goal is to maximize the cumulative expected rewards over time.

$$\pi^*(s) = \arg\max_a Q^*(s,a) \tag{4.8}$$

Here in this formula, $\pi^*(s)$ is our optimal policy where $Q^*(s,a)$ we can say that is our optimal action-value function if our agent take an action a in some kind of state s so this will represent the expected cumulative reward. When we loop through the episodes so the agent updates its understanding over time, because if the policy is updated so that our agent could navigate over the complexity of the game environment very effectively.

4.5. EVALUATION METRICS

For our contribution to this DRL architecture, we assessed our agent, which involves numerous measures, so each of these measures provides its own perspective on what it is capable of. So, in this evaluation, the main goal was that our agent should obtain the highest score during this planned journey of 2000 episodes. Numerous measures that are integrated into this evaluation demonstrate how proficient our agent is to navigate this dynamic and complex environment of the Chrome Dino Run game.

We evaluated our agent through a comprehensive analysis to check its performance across all these metrics and see how it provides insight into its level of expertise in the dynamic environment of the Chrome Dino Run game.

4.5.1. Maximum Score

For our intelligent agent, a very high pivotal metrics is that how it performs to achieve a maximum score during this 2000 episodes. So that, our trained agent did its very best demonstration and ability where it achieved a remarkable maximum score that is 2064, so with demonstration our agent showed off its ability of navigating the game environment adeptly.

4.5.2. Average Score

Over the entire duration, our agent's average score shows its consistent performance in the game environment that how it performs. Therefore, if we notice the average score, it received which is 525 means that our agent demonstrated with a robust ability that makes itself very effective in decision-making across a very diverse range of game scenarios.

4.5.3. Training Time

The performance of our agent and also its efficiency is the crucial part of its training process, where our agent successfully completed the 2000 episodes for the total time of 187 minutes and 56 seconds. Therefore, we can say that our algorithm provides an intuition of computational efficiency for the agent in training pipeline.

4.5.4. Exploration-Exploitation Strategy

Our epsilon-greedy strategy controls the agent's exploration-exploitation compromises. Since, we initially assigned the parameter for ϵ as 1 to explore more, and also, we assigned the decay rate of the ϵ to 0.995. By the time our agent explores so the ϵ gets smaller to minimum reduced amount, which assigned the minimum decay value to 0.01, so that the decay rate ensures our agent to shift forward and of exploitation gradually until it gains more experience.

In this chapter of metrics, we explained that how our agent collectively illustrates its adaptability against the environment, how it performed based on its learning efficiency, and also how our agent achieved the capabilities of decision-making strategies. Finally, how our agent formed with a robust foundation to make itself successful in the training methodology that we assigned.

CHAPTER 5

STATE REPRESENTATION

The game we chose to work with is Chrome Dino Run, which is a dynamic environment where the obstacles are generated randomly, so our agent must determine them and make its decision very carefully. So that, the DDQN algorithm finds its forts here because of its adaptability and also its capacity for learning, where an agent can learn from each of its interactions in the dynamic environment seamlessly. In such a dynamic environment, DDQN gives our agent the courage to maneuver where it can pass through the challenges that our agent faces during the learning process in the Chrome Dino Run game.

Our agent faces a dilemma of complexities in the environment of the game, so state representation will play its own pivotal role in choosing it carefully because it shapes the learning path for our agent. Hence, in this chapter, we dive deeper into the very tiny difference in our state representation for our agent in the Chrome Dino Run game based on the contribution we have made to this DRL solution. Firstly, the exploration in this chapter will be the examination of traditional image-based state representation methods, which are already done in this field, and here we are going to perceive the strengths, limitations, and also the complexities they provide in computation. Respectively, in the next step, we are going to introduce our own innovative design of state representation, which is a kind of move away from a conventional approach. This will set a full, detailed stage justification that we represented the modified state. We are going to render the strategies that enhance agent efficiency, adaptability, and also the learning performance of our agent. Finally, our journey as we present the state representation complexities gradually exposes itself in the subsequent sections, and we are also going to illuminate in detail the path towards a more effective approach as well as the tiny differences in approaches that we have used in the training of our agent in the Chrome Dino Run game dynamic environment.

5.1. TRADITIONAL IMAGE-BASED STATE REPRESENTATION

The discussion of traditional approaches in the DRL solutions, the foundation of training an agent is the state representation, which is image-based. We often get this kind of state representation in gaming scenarios, which are highly rich in visual data environments [37–39]. In this section, we are going to delve into methodologies that overload the agent with the entire game frame as input, which is then the state for our agent, and we are also going to expose the challenges that our agent bears that are associated with these conventional strategies. So that as we shown in Figure 5.1, the process of traditional image-based state representation, we get the raw game frame from the game environment as our input [12,18,32], which then it goes through convolutional and also pooling layers for extraction of hierarchical feature and finally we get our state representation value.

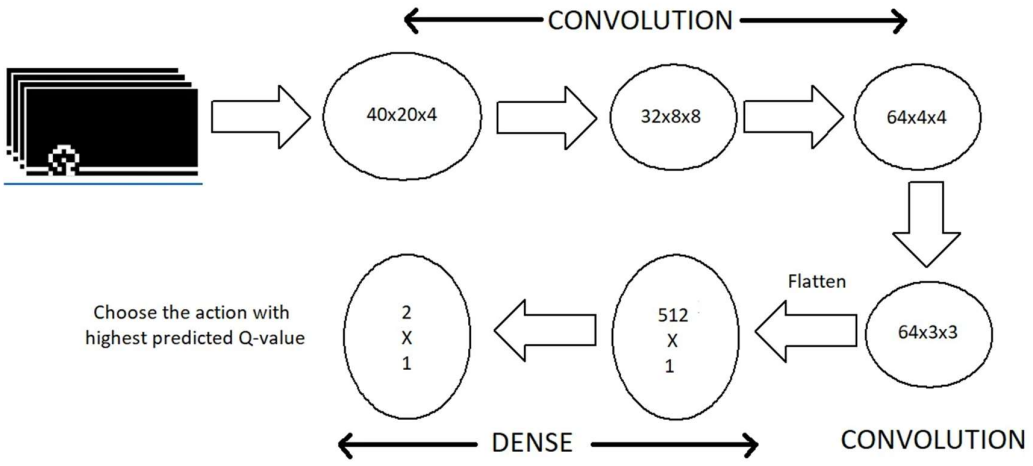


Figure 5.1.Illustration of CNN process employed in traditional image-based state representation [40].

5.1.1. Convolutional Neural Networks (CNNs)

The ever-used approaches algorithm to have an effective state representation for our agent is utilizing CNN algorithm which is clearly and visually illustrated in the Figure 5.1, well the CNN naturally involves methods like processing the game frames through layers [41], and these layers enable the agent to extract hierarchical features and also all its complex patterns from this raw image which is visually presented as input.

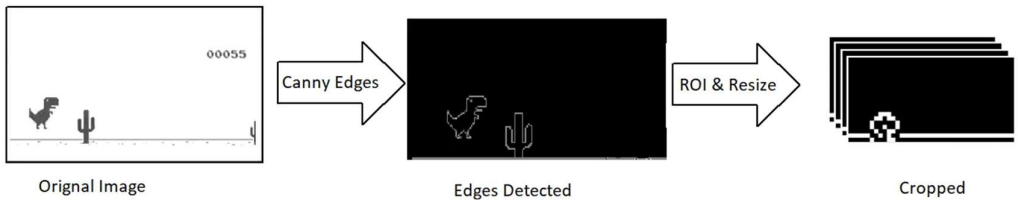


Figure 5.2. Illustration of processing the original image through edge detection and then send to CNN utilization [3].

5.1.2. Computational Inefficiencies

When utilizing CNNs and then processing the whole game frame that comes out of the game canvas, as seen in Figures 5.1 and 5.2, this conventional approach leaves our agent vulnerable to computational inefficiencies [42]. The processing of a large amount of visual data is really straining the real-time learning process of our agent, which can make it very difficult for our agent to achieve the best training efficiency.

The above section discussed about the limitations that our agent would face in traditional image-based state representation approaches and it demands a high computation and overfitting of visual features, this prompts us to design a revaluation contribution on input representation for CNNs because this help the agent to be more efficient and adaptive during training strategies.

5.2. PROPOSED STATE REPRESENTATION

As we deep dived into the traditional image-based state representation and perceived how it incurs a high computational challenge, now is the time that we should introduce our own novel state representation approach, which is only customized for the Chrome Dino Run game environment, and we know that this game is dynamic in nature and also rich in visual data complexities. Our state representation that we proposed has a very specific goal, and that goal is to streamline the learning process for our agent. It also offers our agent a more efficient and targeted means by which our agent can interpret and navigate this dynamic game.

5.2.1. State Representation Design

As we decided to design the state representation in tiny different way, I believe this state takes the main stage of this DRL architecture and also this will reshape the leaning trajectory of our agent in the DRL system. We select to render the state representation as the obstacle coordinates (x, y, width, height), and extracting the coordinate needs a very careful preprocessing pipeline so we have carefully stepped a sequence of process that converts our raw visual data into this meaningful and purposeful state representation. If we assume that S is our state and o is respectively our obstacle then we can structure our equation as tuple that is shown below:

$$S = (x_o, y_o, w_o, h_o) \tag{5.1}$$

Our state representation is very carefully defined with the game's obstacle coordinates in our DRL system, so here in this equation, each of the element contributes with each other's providing crucial information for the understanding of the entire game environment. Specifically, x_o is the x-coordinate, y_o is the y-coordinate, w_o is the width, and finally h_o is the height of our obstacle, each representing their own meaning in this equation, respectively. We designed this structure, and it serves as a foundational representation for the pipeline or our preprocessing. This concept facilitates an integration which is seamless in the game data and also provides a meaningful and informative format for our learning algorithm.

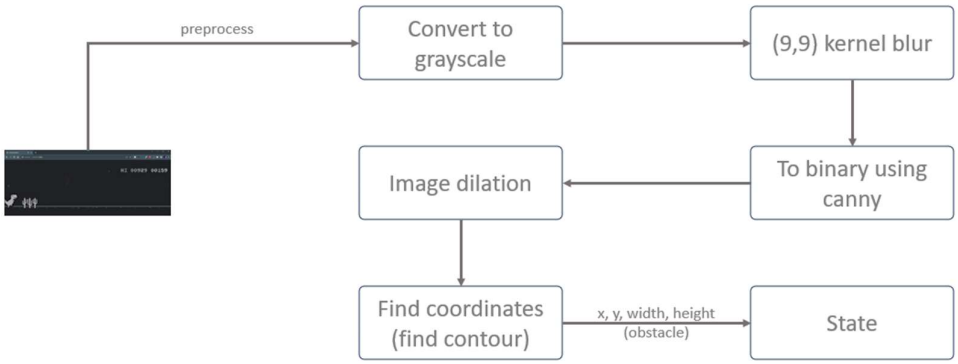


Figure 5.3. Our image preprocessing journey that before rendering it as the state it goes through this processing step to finally outputs the coordinates.

In the Chrome Dino Run game environment we start off taking a screenshot from canvas of the game, since at the backend the language JavaScript so we get a byte-encoded image view then we convert this byte-encoded image into NumPy array then we convert this array image into grayscale which gradually removes the colors to expose some tiny differences in the underlying contrast which clearly shown in Figure 5.3, then the grayscale image is rendered to the blur function to make it smoother the imperfections and emphasize some vital features of the image then it goes through the canny edge detection process to enhance the definition of the obstacles basically it sharpens the contours of the image the dilation process highlights the key elements in focus.

Finally, we reached at the end of the process the function 'findContours' that identifies the contours of the image so we can that, this contours delicately outline the boundaries of the obstacle in the image that come out of the game canvas the 'boundingRect' function then subsequently gives the x, y, w, h which is the encapsulation of the dimension of the obstacle, all these steps which are described are visually illustrated in Figure 5.3.

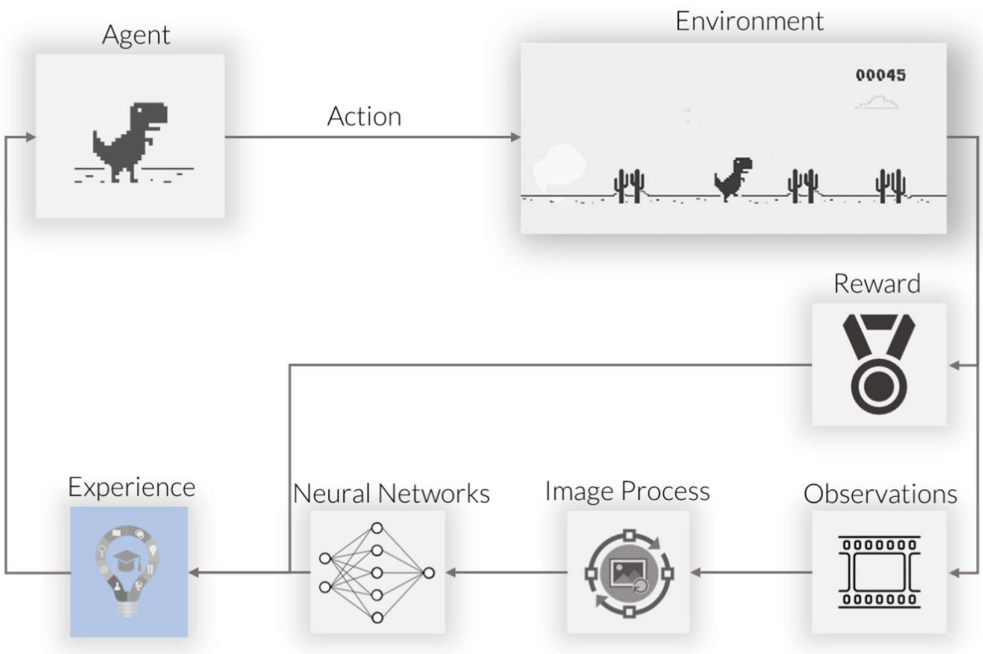


Figure 5.4. Overview of the state representation process.

5.2.2. Advantages of State Representation

In our contribution to this DRL system, we used Neural Networks (NNs) after the preprocessed image and extracted coordinates as the state representation rather than CNNs and inputting the entire game frame. Unlike the previous approaches that inputs image as the state representation into the CNNs and then extracts the visual patterns, our approach just concentrates on the important information what our agent needs to navigate through the environment which is the coordinate of the obstacle (x, y, w, h) which is clearly illustrated in Figure 5.4. our novel approach is pretty straightforward where it gets only the useful information form the image which is obstacle coordinates and render it as the state representation for the agent this method does not compute overhead associated image processing that helps our agent in learning process time. The exemplified environment gives as a dynamic problem that we can navigate it with a unique perspective way which is why we use NNs algorithm that can optimize and handles our input structure effectively.

We use obstacle coordinates that our agent uses as its state representation is a very careful decision for the game and this decision exposes a variety of advantages in this DRL solution that we have. Indeed, these advantages give our agent the efficiency ability and also computational lightness that our agent performs in the environment.

Figure 5.5. A snapshot of the Chrome Dino Run game environment, that shows how it give the precise coordinates and also the dimensions of the obstacle in the game.

We used a diverse range approach just to express the state representation in our DRL solutions is it illustrated very good in Figure 5.5. so that, it shows how we divert the image-based state representation to more concise and very short information which needed by our agent. I believe that image preprocessing, features extraction and reduction of the dimension is a method that every research needs to construct it in the gaming environment because visually rich data is a very huge computational problem.

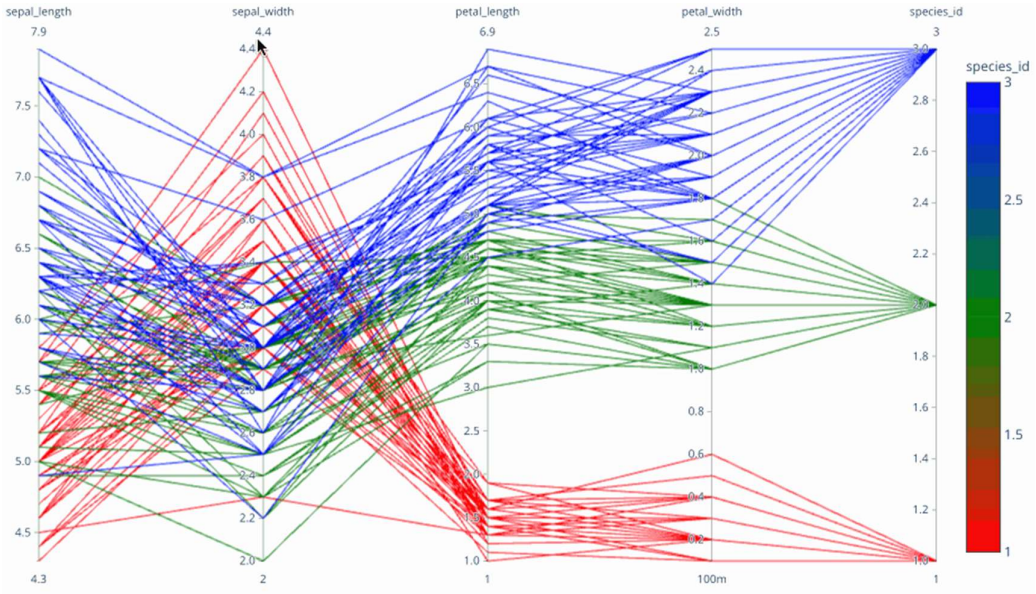


Figure 5.6. This visually image-based state representation illustrates the high-dimensional state space representation [43].

As we investigated about state representation the method that we undertake is inseparably connected with the undefine problem that we receive from Chrome Dino Run game environment. As shown in Figure 5.6, that our strategy and approach make balance between computing efficiency and it gives the agent a rich information that it needs to navigate the environment.

CHAPTER 6

EXPERIMENTAL SETUP

Before we tested and developed the Chrome Dino Run game, we carefully configured our experimental setup for our DRL solution. Therefore, in this chapter, we dig into the appropriate setup and also the complexities of our environment, how we collect systematic relevant data for our agent, how we tuned up our hyperparameters for fast learning, and finally how we implemented the model from the PyTorch library, which is the real backbone that forms all the computational exploration for our agent. We made each section like a key component for easy understanding of how we setup our experimentation. All these collections are done because of their contribution to making our agent intelligent in a robust evaluation framework.

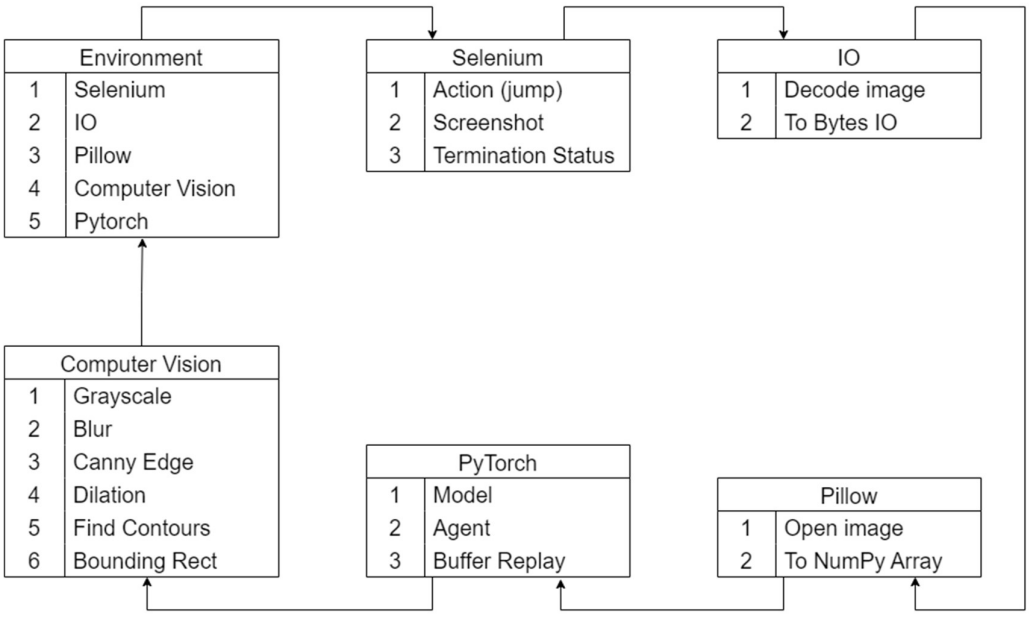


Figure 6.1. This illustration shows the entire overview of our solution process that how the data flow through which components in our Chrome Dino Run game.

6.1. ENVIRONMENT CONFIGURATION

Firstly, we are going to understand our in-depth of the inner workings of our DRL solution, including its complexities. We integrated the component of a versatile library, Selenium, into the environment of the Chrome Dino Run game in its core architecture because this component interacts between the Chrome Dino Run environment's JavaScript and our Python methods. This library serves as the core method of our DRL solution and enables some pivotal functionalities like capturing snapshots, restarting the game, and monitoring the game termination status in the real-time.

The play starts off when our game module takes a screenshot from the game's canvas, and preprocess some of the stages, and renders the result to our DRL system.

Now that we get the image from our game module so the attention turns itself to the vision module, where our main focus is the computer vision skills. Here, we employed the great and powerful OpenCV library that makes our efforts easy by extracting the intricate complexities of the image that was received from the game module, and at the end, it gives us the coordinates of the obstacle (x, y, w, h). This module expresses our Chrome Dino Run game's complexities renders a materialized and smooth parameter which is the coordinates.

So that as we receive the smooth transmission of our image to coordinates makes our agent a master of the decision-making in the environment and this the highest and important point of the mentioned visual journey. This kind of state representation along with the associated rewards give our agent a journey that can ensure a detailed and precise rendering of the game world. Our DRL system exemplifies itself including its refined design and efficiency based on smooth exchange of state representation.

6.1.1. Game Module

When we start off our system, the first module that essentially takes place is the game module, so the module makes itself ready to capture the essence of Chrome Dino Run and renders. Our game module's primary role is to act as the eyes of the agent because

it takes the screenshots from the dynamic and ever-changing environment of the game. We can say that the screenshots it captures for our agent are the beginning strains of the game environment story for our DRL system, so our game module gives raw visual data and sets the groundwork for the succeeding acts.

Within the game module, we can say that the method $grab\_screen$ is taking on a pivotal responsibility in the game module as well as in our entire system because the responsibility this method has is like the lens of our agent through which we can capture the environment. Basically, this method adeptly captures the current game status and gives us a byte-string array type that contains the entire visual information of the game encapsulated. In short, the byte-string array that we receive from this method is a visual data state of the game snapshot that provides a raw material through which we preprocess subsequent stages for decision-making. The retrieval of this method is quick, and makes our system ensures that it receives the most up-to-date visual data as a byte string.

Respectively, after the $grab\_screen$ method, the next method that takes center stage in our system is our act method, because this method translates our agent's decision that it takes into tangible movements in the Chrome Dino Run game environment. Moreover, the role of this method in the game module is to transfer the appropriate action from our system to the Chrome Dino Run game environment, coordinating such activities as jumping or withholding the action.

In the abovementioned module, there is a property named $is\_terminated$ that is the sensor of our system, or we could say that this property is acting like the sensor for our system it provides a binary signal to our system informing about the game termination, whether the game has concluded, and also signals our agent that either it has done a successful navigation or an obstacle collision going through these challenges of the game.

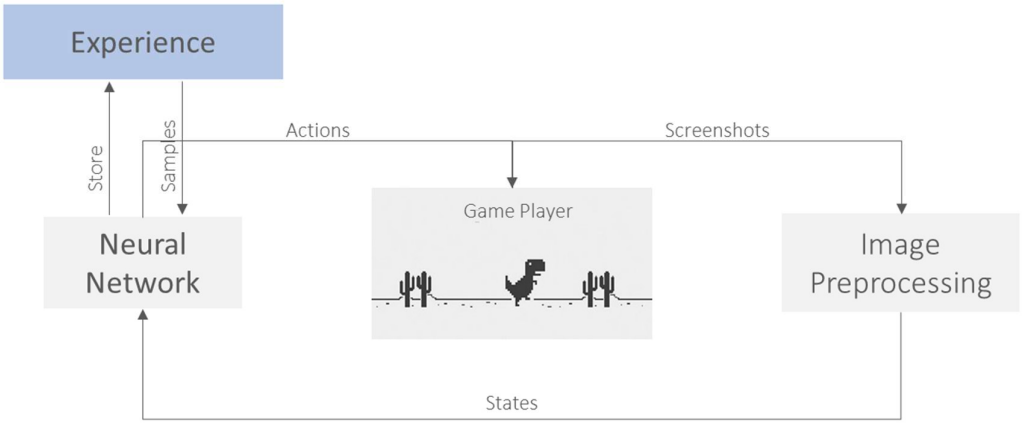


Figure 6.2. Shortly it illustrates the interaction flow within our system. The agent signals an action to the game and takes back a screenshot from the game. Then the screenshot will go to preprocess in the image processing stage then we get the result and send it to the neural network as the state.

6.1.2. Vision Module

When we transition our focus towards the vision module here the screenshots that our game module captured as the byte-string array serves as a canvas and our vision module should unfold its intricate details smoothly. Within this module we used some sophisticated algorithms from OpenCV library through which this module skillfully extracts the elusive coordinates of obstacles from visual composition of the image. Therefore, we can say that coordinates we received is like a complicated strokes in the environment so it easily unlocks the door of understandable complexities which embedded naturally into the fabric of our game environment.

This module starts off with an initial step that is $conv2gray$ method that takes the image as its input, basically this method processes the captured game frame what it receives it just transforms that into a grayscale image. Therefore, this conversion make it sample for further step because the image is now a single-channel representation of the image.

When one steps forward, after our image is converted into grayscale, so the next method that takes places is the $blur$ method that takes an image as the input and give the image a very smooth effect. In this method we apply a Gaussian blur to the image

since this method renders the image with lesser noise and also with an enhanced quality overall, and make the image ready for further processing.

Respectively, then $canny$ method comes to the play, so this method takes the preprocessed image from the $blur$ method then this method applies its embedded Canny edge detection algorithm to the received image. Basically, when the Canny edge detection applied the image contours makes itself sharpen because the edges should be identified clearly and also highlights essential key features before it renders the image to the next step.

After the abovementioned process, the step come to the $findContours$ method which is an OpenCV module. This method identifies and also extracts the contours which are presented in the image that comes in as processed through the above methods. This method provides a very detailed representation of the key features in the image and also it provides the structures within the image of the game.

To get the end result as the coordinate of the game image, we utilized the $boudingRect$ method which is an OpenCV method. Since, this method gives as the final state which are the coordinates of the game as (x, y, width, height) encapsulates our state and gives us.

6.1.3. Replay Buffer Module

The headed module is responsible for just doing behind-the-scenes stuff, or we can say it is a backstage actor, just like a coordinator of memories that assumes to boost our agent's performance. Moreover, it takes on the role of saving and preserving the experiences by the time our agent just passed and experienced them, including states, actions, rewards, next states, and finally the terminals. Our agent saves these experiences in its memory because they will serve as an archive for its learning journey by the time it tries to learn the environment. This gives our agent an array of experiences from various positions that helps the agent to draw during the training process.

6.1.4. Agent

Our agent which is the last arrangement in our DRL solution modules we arm it with coordinates of obstacle which our vision module already extracted utilizing this these coordinates it enters into the limelight to make decision during its journey based on its past experiences and define its trajectory within the game. The coordinates which are extracted by the above-mentioned module helps our agent as its state representation as it takes action navigating the Chrome Dino Run game landscape. Our agent has a binary option that comes out of its neurons which is 0 based on this our agent's decision would be 'do nothing' and if its 1 so the neurons of our agent signal to our agent you should 'jump' and this becomes a concluding note in this complex composition.

6.2. DATA FLOW

I believe that the interaction and communication between numerous components within our DRL solution complexities that they arrange is our data flow. Therefore, the main data that all the components try to make it is the coordinates of the obstacle, because this obstacle's coordinates just try to shape our state representation, and this representation guides our agent through its learning path in this complex environment.

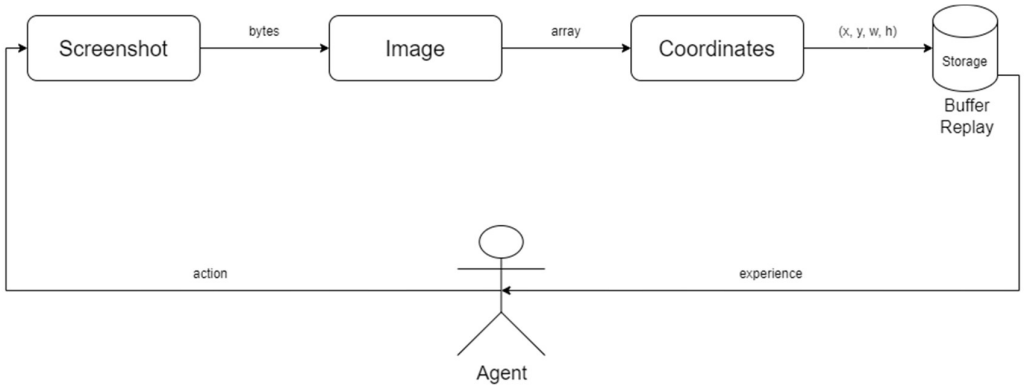


Figure 6.3. It illustrates that our agent starts its journey by a screenshot then extract the coordinates and utilize that as its state representation and it saved in its experience memory data flow.

As we visually showcased our data flow diagram in Figure 6.3, from the overall perspective of our components, the game module takes the first and center stage in our DRL solution because it provides the game environment screenshots continuously, and these screenshots serve as our agent's visual information for its entire journey. Respectively, as we go forward, our vision module takes the responsibility, and tries to enter into the limelight, and starts some preprocessing routes meticulously. As the responsibility is transferred to our vision module, this module applies a variety of complex algorithms that we took from the computer vision OpenCV library and extract the encoded obstacle coordinate (x, y, w, h) features from the image that is delivered by the game module.

The extracted coordinates from the image influence the neural pathway as an extended choice for our agent because this choice enhances the efficiency of our agent during the learning process and also our agent will have a precise choice in decision-making. The data flow that we structured very precisely will act as a well-rehearsed performance that will reinforce our agent to be intelligent in this efficient system, and it will also highlight that how important is the state representation that we have chosen, which is the obstacle coordinates, is and how it guides and forces our agent to perform efficiently in this DRL system.

6.3. HYPERPARAMETER TUNING

In our DRL system, the overall components and modules that are living or trying to show themselves are very carefully structured. Similarly, the DDQN algorithm that we selected in our architecture is not only some layer arrangement; we fine-tuned its hyperparameters in a precise way. In this DDQN algorithm, our NN concept just exposes some stages of layers, including the input layer that is responsible for capturing the substance of the coordinates that we extracted very carefully and the hidden layers that we give the responsibility of extracting the tiny differences of features following this layer. At the end of this path, we have an output layer that just gives our agent action that is more suitable to perform based on the current state that our agent faced. Moreover, the architecture that we designed will help our agent to navigate through the game environment with poise and purpose.

Table 6.1. Configuration of our Hyperparameters constants tuning for our DRL system's DDQN algorithm.

Constant	Value
Buffer Size	6e5
Update Every	4
Batch Size	1024
Gamma	0.99
Tau	1e-2
LR (Learning Rate)	1e-3

For optimal performance of our DRL solution we carefully mixed up our DDQN hyperparameters based on the algorithmic experimentation. Each constant which are mentioned in Table 6.1, acts a vital role in this complex environment and for performance term and nuanced decision-making these constants are important. To make sure that our agent performs a precise and also a balance learning process refining these parameters are crucial.

6.3.1. Learning Rate (LR)

In the concept of our DRL solution, learning rate controls the size of the steps and then it updates the parameters of the model based on these steps and how it receives its rewards to predict. We used this important parameter to train our algorithm for our Q-learning concept. Moreover, if we assign a higher learning rate so it allows our algorithm to update its parameter larger but the concern here is that it overshoots and converges bigger. Likewise, if the assigned value is lower so that more conservative update and here the issue would be slowing down the learning process.

Thus, to appropriately select a learning rate for our algorithm so we have to have more experimentation on the tuning our learning rate value. Hence, very high value selection for our learning rate results a fluctuation and very low value selection for our learning rate our algorithm will get stuck in local minima. Observing the mentioned issues, we used an advanced optimization algorithm named Adam that makes the learning rate flexible during the training because it checks out the historical information of our gradients in the algorithm and do the needs for the learning rate. Therefore, this kind

flexibility in adjusting the learning rate during training time will improve the training process in our DRL solution.

6.3.2. Updating the Q-Network

When we update the Q-network for our agent continuously, so we can say that is the key of our training process, for our agent. Therefore, our agent makes its understanding, that where and how it should act by the time our agent garnered its rewards during this training process. Our Q-network will strength itself during the training process and makes out an ever-evolving masterpiece because our agent is now a composition of strategies that it learned and flexibility it gained during the training process.

Moreover, updating the Q-network will traverse our agent through episodes and epochs, and after the training is done then we get a piece of intelligence out of our agent. When we tun the parameters, to be flexible with game environment, after that when we update the Q-network then our DRL solution shapes itself with intricacies components. Finally, we get an agent with a new emerge that is not only a learner but a masterpiece with expertise, and also our agent really ready for the challenge that the Chrome Dino Run game gives it, and I believe our agent will navigate through it with its finesse and strategies that it gained during the training.

6.3.3. Soft Update Parameter

In terms of our DRL solution the soft update we used to update the parameters of our target network very gradually over the time of training, because an abrupt update will cause some issues to our target network and we denoted the soft update with parameter called ‘tau’ (τ). Thuse, we embedded two kinds networks in our DRL solution which are our Q_{target} and the other one the Q_{local} rather than updating the Q_{target} with Q_{local} parameters directly per iteration, so we used this soft update τ concept to update our target network with a tiny number of parameters from our local network.

Hence, the soft update parameter helps our agent to update its target network with a more stable and also change very slowly within its algorithm because it helps our agent in avoiding oscillations during training. Generally, we set our τ value with very small number may be between 0.001 and 0.005 because it makes our algorithm ensures to update very slow and very steady its Q_{target} parameters from Q_{local} parameters.

6.3.4. Exploration-Exploitation

The concept of exploration over exploitation in our system tries to make a right balance between finding a new action that may have greater reward for our agent which the exploration concept and the knowledge that our agent currently has and based on this knowledge it can maximize its cumulative reward which is the concept of exploitation. Here balancing this concept really, a tough challenge because if the agent explores too much then it won't be able to gain it cumulative rewards for some optimal actions. Likewise, if our agent continues with the exploitation or the knowledge it currently has will lead our agent premature or we can say with less knowledge.

Table 6.2. Exploration and Exploitation parameters for Fine-Tuning.

Parameter	**Initial Value**
Epsilon	1
Epsilon Decay	0.995
Epsilon Minimum Decay	0.01

We select the concept epsilon-greedy, in this concept our agent chooses a random action, or we can say the exploration concept if the probability is ϵ, likewise it continues with its current knowledge it has and also it knows that this action is appropriate in this state if the probability is $1 - \epsilon$.

6.4. MODELS

In our system, our goal is to design an architecture through which our agent should achieve a high and precise flexibility in performance, when playing the game. Hence, we carefully chose PyTorch library to design our model for our agent in this system

which is very powerful and rich in machine learning environment. As shown in Figure 6.4, the layers in our model are customized where each layer handle various aspects of its responsibilities in the learning process of our system for our agent.

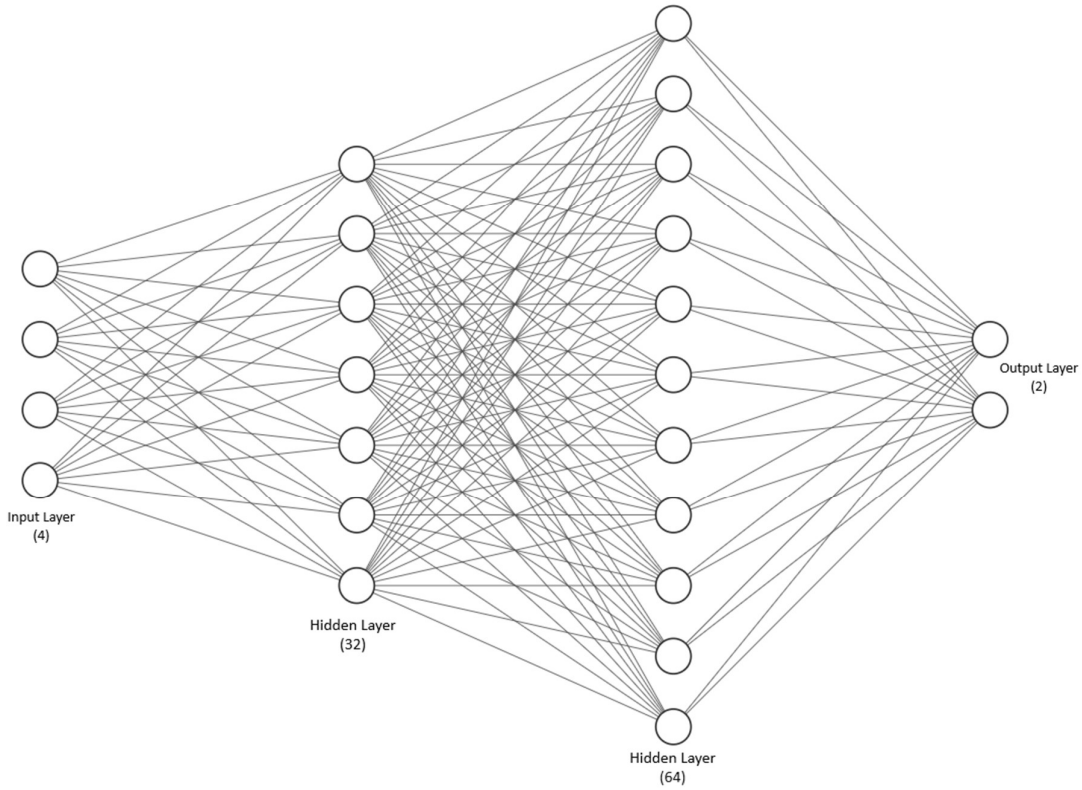


Figure 6.4. Architecture of our PyTorch neural network that we represented in this visual diagram.

Respectively, our input layer which is the initial layer in our model, we designed this layer that has the capacity of 4, and this number is the size of our state that comes as an input to our first layer which is represented like (x, y, w, h). When it receives the state as its input with the mentioned size so following forward it outputs a data representation of 16 in size that provides the feature space as its intermediate representation for the subsequent layers, that builds on top of this layer.

When take a look into our hidden layers, so, our first hidden layer receives a size 16 data representation from the input layer and then converts this intermediate data representation in a size of 32 feature space for second hidden layer which requires a data representation of 32 in size. Respectively, our second hidden layer receives the

data and then refine that data features in an input size of 64 feature space which then send to output layer.

Finally, we got to the final layer, which is the output layer appropriately, so our last layer takes a dimension of 64 feature space, and give us a size of 2, which is aligned with the number of actions that our agent tries to learn in the Chrome Dino Run game environment. I believe, this layer is doing a very crucial responsibility, because it determines the optimal action for our agent based on what it leaned features from its past.

Initially, the layers we have here in this system are linear, to introduce none-linear functionality and make our model enable to capture complex patterns, basically we have utilized to all the layers, is the concept of Rectified Linear Unit (ReLU) activation function and we exclude the last layer which the output layer from ReLU activation function. We decided to employ ReLU to our layers reflects a strategic choice that enables the capacity of capturing intricate patterns and enhance the model with none-linear system within the game environment.

We used the PyTorch $argmax$ function for our action selection. Generally, this function selects the value that has the highest Q-value probability, and this is going to be a very important factor for our agent in this system for decision making.

CHAPTER 7

RESULTS AND ANALYSIS

This chapter is held, to explain and also show off the thorough study that we have done on the Chrome Dino Run game, and how our agent that we trained it, performs in this environment. In this chapter, our main focus would be on some key metrics, like the maximum score that our agent achieved during it then training, average score, and finally the training time that our agent past, by this key metrics, we can assess the efficiency of our agent among a variety of configurations and variations that we had refined.

7.1. EXPERIMENTAL SETUP

A very quick review of experimental setup. We use PyTorch and implement the DDQN algorithm for our experiments. Our model that we structured has an input layer with 4 dimensions as input and this layer outputs 16 dimensions to the next layer which the hidden layer. In this architecture the first hidden layer has the input 16 and outputs back 32 dimensions to the next hidden layer. The second hidden layer takes an input of 32 feature space and outputs 64 dimensions to output layer. Finally, our output layer gives a 2-dimensional information which are the actions for our agent. Respectively, we applied ReLU activation function to all our layers excluding the output layer which uses the $argmax$ function for high Q-value production.

7.2. BASELINE PERFORMANCE

The journey of our agent continued for 187 minutes and 56.8 seconds in its time. During its journey in this training, it achieved a maximum score 2064 which was very good, and if we count its average score from the overall of its training process so our

agent gained 525 score in average. If we come to its exploration-exploitation trade-offs, we set the parameter of our epsilon to 1 and it was decaying during this training process with the decay rate of 0.995 exponentially, until it reached for the minimum value which was 0.01, so this was our baseline performance configurations.

7.3. ALTERNATIVE CONFIGURATIONS

We examined various experimentation by changing the hyperparameters setting and also the model's itself architecture, just to assess the different setting's impact and also its variations. All we summarized in Table 7.1, including the performance of our agent under different configurations, and also, we highlighted the changes like maximum rewards and the time of the training for each episode.

Table 7.1. Variation of configurations on our coordinate-based state representation.

Configuration	TAU (τ)	Learning Rate (LR)	Hidden Layers	Hidden Layer Size	Epsilon Decay
Base Line	0.01	0.001	2	32, 64	0.995
A	0.001	0.0005	1	64	0.998
B	0.01	0.001	1	8	0.995
C	0.1	0.001	0	4	0.995
D	0.01	0.001	2	32, 64	0.99
E	0.1	0.01	1	8, 4	0.995

We focused mainly just on coordinate-based state representation, and also Table 7.1, clearly explains every configuration that we applied in our paper and tells everything. In the abovementioned table we described our experiment in each row, and columns in this table points to the specific configuration of the parameters with their respective sizes in each experiment that we have done. Therefore, we have provided a clear transparency, and we mentioned the overall experiments and its setup describing all its configurations, from these experiments that we have done in this research, gives us the understanding of how we configure the parameters based on which environment. This exploration within the context of coordinate-based state representation of Chrome Dino Run is nicely understandable.

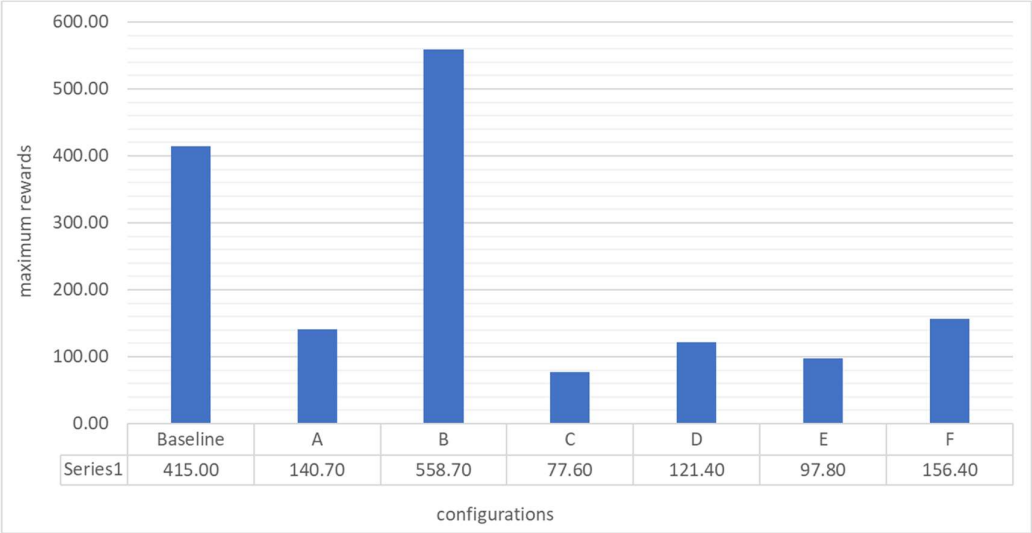


Figure 7.1. Maximum rewards across different configuration visual representation.

Based on the variations we brought in parameter configurations where we depicted in Table 7.1, and visually illustrated in chart based in Figure 7.1, altogether points to the maximum rewards variation that our agent gained in different environments. This concept provides, that how our agent performed under various circumstances and the impact of these configuration variations on our agent like layers structure and learning rate in the Chrome Dino Run environment. Figure 7.1, the visual representation as the chart represents the comparative analysis of experiment outcomes.

In general, this average reward term describes the number of average rewards, that our agent received while it is in training. This may consist of several episodes and epochs, particularly in the Chrome Dino Run game environment. Therefore, we extracted the average rewards of our agent, because it evaluates its performance and how agent became intelligent.

Table 7.1 along with Figure 7.2, consecutively show the average reward of our agent which it gained during its training across different configuration that we have made. We uniquely identified each configuration, since we could spot them and do some comparative analysis on them, to check the performance of our agent under these different experimental settings for our Chrome Dino Run game environment.

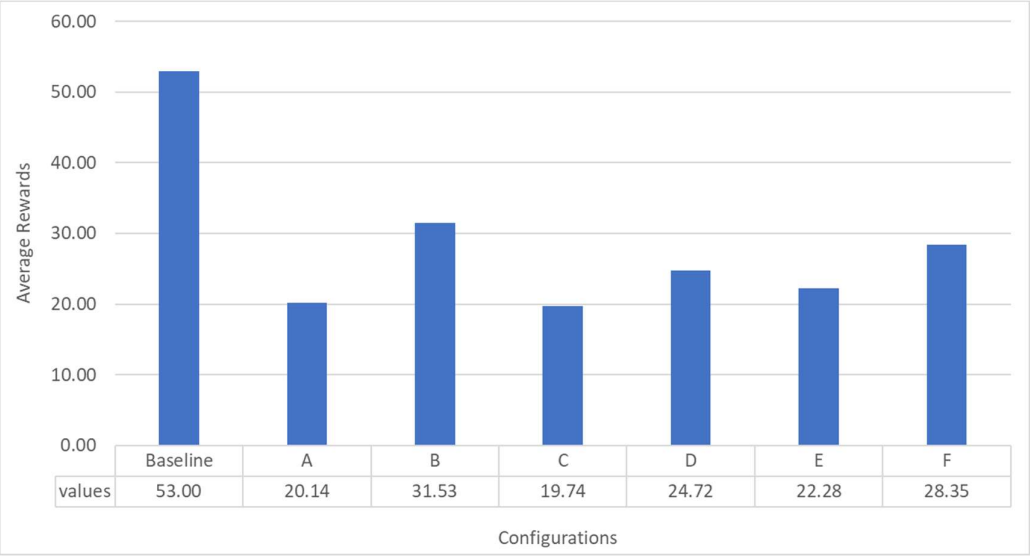
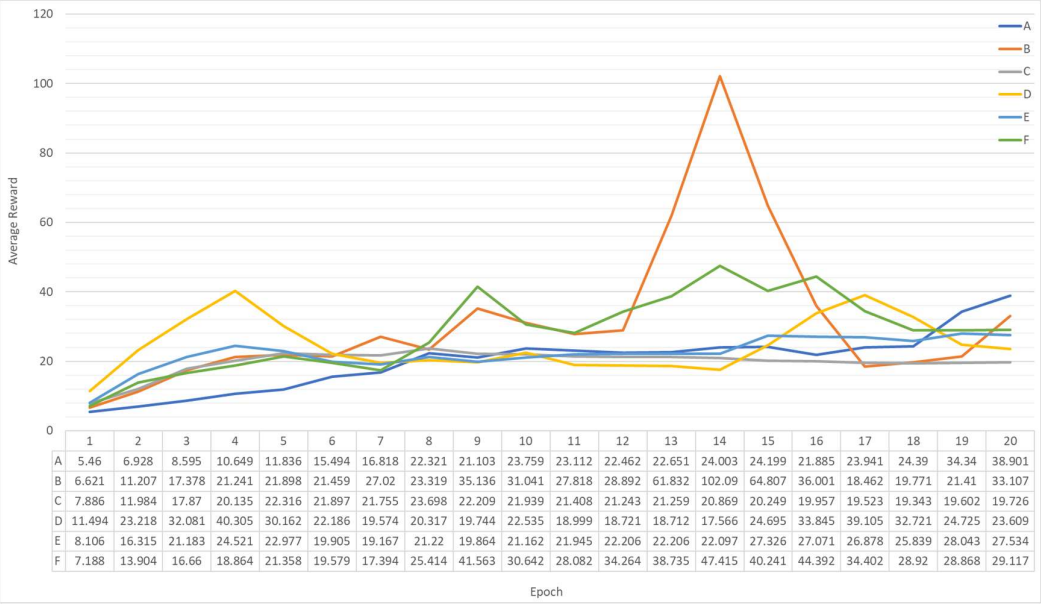


Figure 7.2. Average rewards for different configuration for our system.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
A	5.46	6.928	8.595	10.649	11.836	15.494	16.818	22.321	21.103	23.759	23.112	22.462	22.651	24.003	24.199	21.885	23.941	24.39	34.34	38.901
B	6.621	11.207	17.378	21.241	21.898	21.459	27.02	23.319	35.136	31.041	27.818	28.892	61.832	102.09	64.807	36.001	18.462	19.771	21.41	33.107
C	7.886	11.984	17.87	20.135	22.316	21.897	21.755	23.698	22.209	21.939	21.408	21.243	21.259	20.869	20.249	19.957	19.523	19.343	19.602	19.726
D	11.494	23.218	32.081	40.305	30.162	22.186	19.574	20.317	19.744	22.535	18.999	18.721	18.712	17.566	24.695	33.845	39.105	32.721	24.725	23.609
E	8.106	16.315	21.183	24.521	22.977	19.905	19.167	21.22	19.864	21.162	21.945	22.206	22.206	22.097	27.326	27.071	26.878	25.839	28.043	27.534
F	7.188	13.904	16.66	18.864	21.358	19.579	17.394	25.414	41.563	30.642	28.082	34.264	38.735	47.415	40.241	44.392	34.402	28.92	28.868	29.117

Figure 7.3. Average rewards per epoch for different configurations.

Figure 7.3 displays the average rewards that the agent earned for each period under various conditions, as shown by Table 7.1. This visual depiction makes it easier to analyze agent performance in-depth during training by showing the patterns and changes linked to a particular test configuration in the Chrome Dino Run environment.

7.4. LEARNING CURVES

To journey with our agent, passing through Chrome Dino Run game complexities, and face the challenges that come to our agent, the learning curves are nicely explained and illustrated in Figure 7.4. Curves that are created, comes with the goal of showing the fundamental progress of our agent in the training episodes, so that it could offer us its visual representation that it tracked down during the learning time steps. Learning curves illustrates fluctuations, peaks, and plateaus as our agent performs by navigating the dynamic game environment.

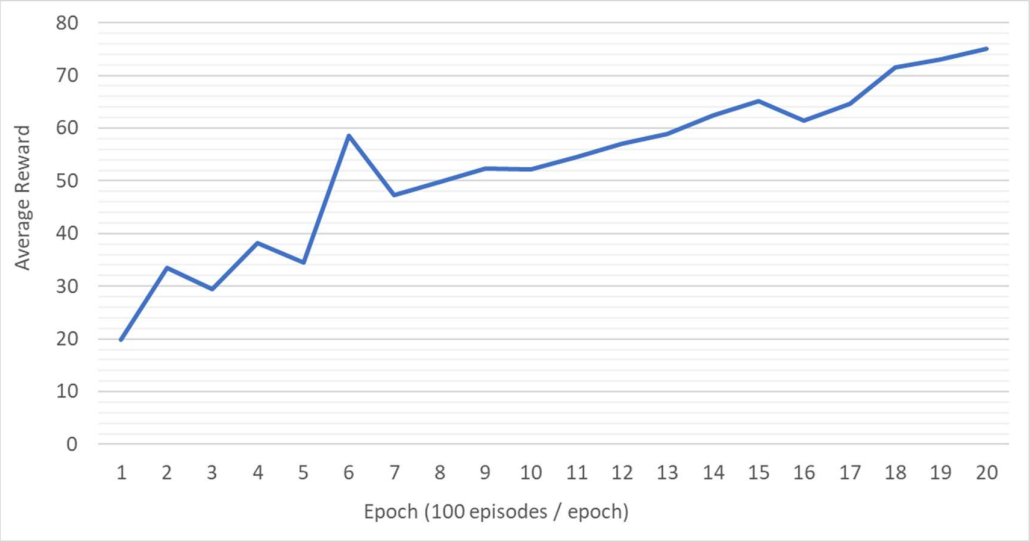


Figure 7.4. Our baseline experiment learning curves journey.

Each curve gives us insights into the gradual improvements and adaptation of our agent that passed through the challenges of the Chrome Dino Run game, which posed, and also it offers us the expanded view of how our agent grows in performance. Figure 7.4 clearly illustrates the learning curves on average rewards that our agent received in each epoch over the course of training episodes.

In Figure 7.5, we illustrated the pinnacle of achievement, that our agent recorded receiving the maximum reward in each epoch, facing the challenges and passing through during its training process. As it shown visually, it mostly highlights the peak moments that our agent performed in that epoch, so that provides the extensive

overview of our agent that how it uses its ability and capacity to navigate through the obstacles and also excel within that dynamic environment of the Chrome Dino Run game. Initially our agent is in exploratory point that really does not know about decision-making, so that learning curves narrates the proficiency of our agent in some challenging path in the dynamic environment.

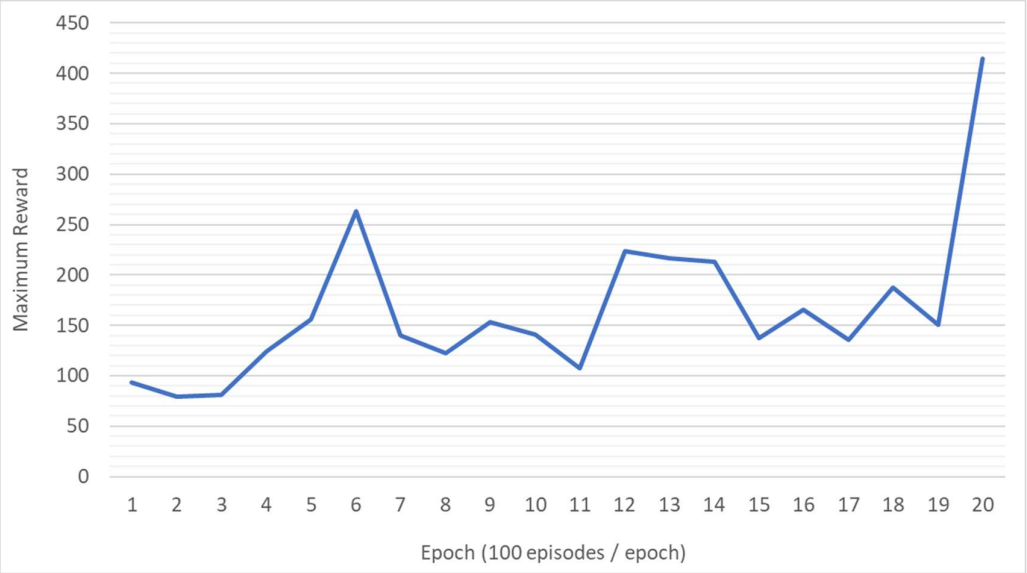


Figure 7.5. Our baseline experiment maximum rewards and it is shown per epoch.

In this comprehensive analysis, learning curves, as showcased in Figure 7.4 and Figure 7.5, provide a detailed insight into the dynamic evolution of the agent's performance throughout the training process. The curves not only depict the raw scores achieved by the agent but also uncover intricate trends, shedding light on crucial aspects such as convergence patterns or episodes marked by fluctuations. This through examination of learning curves enriches our understanding of the agent's learning trajectory, offering valuable insights into the nuance of its adaptation to the challenges posed by the Chrome Dino Run environment.

7.5. GENERALIZATION TO DIFFERENT SCENARIOS

In this research, we mostly focused on generalizations within the variety of scenarios in the case of critical aspects to assess the adaptability and robustness of our model.

The Chrome Dino Run game gives us some multitude of challenges, such as obstacle variations and the configuration of some dynamic changes in game speed.

We carefully conducted several experiments to see how our agent uses its learned policies by examining them into some different situations. The mentioned situations could include random obstacle patterns; that could also be the environment with an absolutely different layout; and it also adds some complexity by the time the player gets more scores to the game. Our goal in this study is to investigate the model that we structured. Now, this model should use its ability to generalize its experience that it perceived to make decisions beyond some setting that our agent encounters during the training.

CHAPTER 8

DISCUSSION

In this chapter we going to discuss about the examination of our methods and also about interpretation of the data that we collected and shed some light on our experimental research specifically that what we have within the Chrome Dino Run game environment.

8.1. INTERPRETATION OF RESULTS

The essential role here is that we have to interpret our methods for its understanding and its limitations efficiencies. We have examined our model's performance and made metrics for it, the examination factors that we got here, such as maximum rewards, average rewards, and the learning curves we made for our agent's performance. We identified our agent's patterns, trends, and anomalies so that we could provide an in-depth narrative about our agent to capture model from its core perceive its behavior from the overall training process.

Different experiments that we have done with different configurations will reveal some invisible patterns that influence our agent to make itself successful in the Chrome Dino Run environment. Therefore, when we fine-tuned our hyperparameters noticeably, we got a discernible improvement in scores; the convergence accelerated, and this is an evident when we fine-tune the specific parameter. As an instance, we can say that if we bring a small change in the learning rate parameter or if we bring change to the structure of our NN will have a very significant impact on the agent's learning dynamics.

8.2. INSIGHTS INTO STATE REPRESENTATION INFLUENCE

Our discussion mainly covers the influence of state representation that we designed and how it impacts on the learning dynamics of our model. After a lot of investigations, we found out that transitioning from traditional image-based state representation to a more efficient way like coordinate-based state representation approach will impact better on the model that we designed. Therefore, through all these meticulous investigations that we made on the experimental configurations, we perceived that how the state representation we chose has the impact on the ability of our model and how it navigates through the complex gaming environment.

The data we are collecting from the game environment support our initial theories that apply on the exploration-exploitation trade-off. The trends we've noticed highlight how crucial it is to finely tune key parameters, finding that sweet spot where we delicately balance exploring new strategies and making the most of what we've already learned. As we dig into these findings, we get a better grasp of how our agent tackles the game's tricky parts. This gives us some cool ideas for sprucing up and fine-tuning the DRL system down the road.

8.3. IMPLICATIONS FOR FUTURE RESEARCH

These findings are just the tip of the iceberg not just about what we have learned today. We are excited about what they could unlock the bigger picture of DRL research and where do these findings lead us will be the new research avenues that it opens up. As we discover the areas to level up like potential extensions, and unexplored dimensions or the stuff we have not looked into yet gives some thrilling future endeavors. They are not just about improving DRL models but about making them absolute adaptability, efficiency, and generalization capabilities in the dynamic world with all-around awesomeness to the forefront.

With these talks, we can think of our agent as a super player who can learns by playing millions of games. That is the dream of our agent in gaming, and we are digging deep

to explore its potential for advancing our research in the exciting domain of DRL applied to the gaming environment.

CHAPTER 9

CONCLUSION

After all this digging, we have finally unearthed the treasure summary of oud study, in this final chapter, we tie up all emphasizing key findings, contributions and the problems we have solved and the exciting possibilities for future research.

9.1. SUMMARY OF FINDINGS

It is not just about the data and conclusion of our research; it captures the essence of what we learned and we can think of it the overall summary of our journey. We go back to the good stuff such as highlighting what significance we have achieved, the challenges we faced and what interesting things we noticed during this extensive experimentation that we have done with the Chrome Dino Run game environment. Rewinding and rewatching our highlights back give us all about what we did and this analysis is our way of putting a bow on the package, and what doors it unlocks for our future explorations.

9.2. CONTRIBUTIONS OF THE STUDY

We made the impact of our research clear and relatable, showcasing the groundbreaking contributions that take the whole field of DRL applied to gaming scenarios to the next level. Whether we talk about new ways of the state representation, making the enhancements to the algorithm, or diving deep into the complex details of the Chrome Dino Run game, we lay out a unique value that our research brings to the existing body of knowledge. In this section, we give credit to how our study pushes the boundaries and how our study can help boost our grasp on using DRL principles.

9.3. LIMITATIONS AND FUTURE WORK

Although, our research found some roadblocks but it isn't flawless, thus it gives some awesome ideas that can inspire future research efforts. We took a close look at the limitations and challenges of our study, we may not have all the answers, but we identified areas where we can improve. We hope that our work can serve for us as a guide for future research to build on our work and unravel the hidden secrets.

Selenium is basically a tool that helps us to interact with Chrome Dino Run game in the browser, and sometimes our agent experiences a delay problem when taking screenshots using this library, so we need to speed it up to make the interaction feel natural and instant for our agent [44]. Basically, our agent takes screenshots from the game to observe what is happening, so that if those screenshots take too long, it makes our agent to react slowly. We can improve this issue by utilizing parallel processing or asynchronous methods since our agent can see the game very clearly and will react as quickly as possible.

Challenge that makes our agent stuck frequently in a loop, and also in this situation, our agent won't be able to learn since it gets infrequent rewards [45,46], which we generally call it sparse rewards in the game environment, which opens up an opportunity for our future investigations. Therefore, making the system more flexible, we should explore different ways of improving our agent through a nicely reward system. Doing that, we should investigate reward engineering strategies and also, through extra environmental cues [47–49], we can implement advance reward shaping methods. This leads our agent to an efficient and flexible learning and also enables our agent in grasping some hidden behaviors to improve the overall performance of the system.

BIBLIOGRAPHY

1. Plaat, A., "Deep Reinforcement Learning, a textbook", (2022).

2. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D., "Human-level control through deep reinforcement learning", ***Nature 2015 518:7540***, 518 (7540): 529–533 (2015).

3. "Build an AI to Play Dino Run", **https://blog.paperspace.com/dino-run/** (2024).

4. "2018 {IEEE} Conference on Computational Intelligence and Games, {CIG} 2018, Maastricht, The Netherlands, August 14-17, 2018", (2018).

5. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., Van Den Driessche, G., Graepel, T., and Hassabis, D., "Mastering the game of Go without human knowledge", ***Nature 2017 550:7676***, 550 (7676): 354–359 (2017).

6. Kirk, R., Zhang, A., Grefenstette, E., and Rocktäschel, T., "A Survey of Zero-shot Generalisation in Deep Reinforcement Learning", (2021).

7. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D., "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play", ***Science***, 362 (6419): 1140–1144 (2018).

8. Hu, H., Mirchandani, S., and Sadigh, D., "Imitation Bootstrapped Reinforcement Learning", (2023).

9. Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D., "Continuous control with deep reinforcement learning", ***4th International Conference On Learning Representations, ICLR 2016 - Conference Track Proceedings***, (2015).

10. Huang, Y., "Deep Q-networks", ***Deep Reinforcement Learning: Fundamentals, Research And Applications***, 135–160 (2020).

11. Li, S., Hu, X., and Du, Y., "Deep Reinforcement Learning and Game Theory for Computation Offloading in Dynamic Edge Computing Markets", ***IEEE Access***, 9: 121456–121466 (2021).

12. Aparecido Breve, F., "From Pixels to Titles: Video Game Identification by Screenshots using Convolutional Neural Networks", (2023).

13. Tian, X., Li, B., Gu, R., and Zhu, Z., "Reconfiguring multicast sessions in elastic optical networks adaptively with graph-aware deep reinforcement learning", ***Journal Of Optical Communications And Networking, Vol. 13, Issue 11, Pp. 253-265***, 13 (11): 253–265 (2021).

14. Guo, X., Singh, S., Lee, H., Lewis, R. L., and Wang, X., "Deep Learning for Real-Time Atari Game Play Using Offline Monte-Carlo Tree Search Planning", ***Neural Information Processing Systems***, (2014).

15. Wang, Z., Fan, D., Jiang, X., Triantafyllou, M. S., and Karniadakis, G. E., "Deep reinforcement transfer learning of active control for bluff body flows at high Reynolds number", ***Journal Of Fluid Mechanics***, 973: A32 (2023).

16. Do, S., Song, K. D., and Chung, J. W., "Basics of deep learning: A radiologist's guide to understanding published radiology articles on deep learning", ***Korean Journal Of Radiology***, 21 (1): 33–41 (2020).

17. Marwah, D., Srivastava, S., Gupta, A., and Verma, S., "Chrome Dino Run using Reinforcement Learning", (2020).

18. "How to Play Google Chrome Dino Game Using Reinforcement Learning | by Iustina Ivanova | Deelvin Machine Learning | Medium", **https://medium.com/deelvin-machine-learning/how-to-play-google-chrome-dino-game-using-reinforcement-learning-d5b99a5d7e04** (2023).

19. Torrado, R. R., Bontrager, P., Togelius, J., Liu, J., and Perez-Liebana, D., "Deep Reinforcement Learning for General Video Game AI", ***IEEE Conference On Computatonal Intelligence And Games, CIG***, 2018-August: (2018).

20. "Build an AI to Play Dino Run", **https://blog.paperspace.com/dino-run/** (2023).

21. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. A., "Playing Atari with Deep Reinforcement Learning", ***ArXiv.Org***, (2013).

22. Adriaenssens, A. J. C., Raveendranathan, V., and Carloni, R., "Learning to Ascend Stairs and Ramps: Deep Reinforcement Learning for a Physics-Based Human Musculoskeletal Model", ***Sensors***, 22 (21): 8479 (2022).

23. Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D., "Rainbow: Combining Improvements in Deep Reinforcement Learning", (2017).

24. "Welcome to Python.Org", **https://www.python.org/** (2024).

25. "WebDriver | Selenium", **https://www.selenium.dev/documentation/webdriver/** (2024).

26. "NumPy", **https://numpy.org/** (2024).

27. "OpenCV - Open Computer Vision Library", **https://opencv.org/** (2024).

28. "Pillow (PIL Fork) 10.2.0 Documentation", **https://pillow.readthedocs.io/en/stable/** (2024).

29. "PyTorch", **https://pytorch.org/** (2024).

30. Van Hasselt, H., Guez, A., and Silver, D., "Deep Reinforcement Learning with Double Q-Learning", ***AAAI Conference On Artificial Intelligence***, 2094–2100 (2015).

31. Sewak, M., "Deep Q Network (DQN), Double DQN, and Dueling DQN", ***Deep Reinforcement Learning***, 95–108 (2019).

32. Oroojlooyjadid, A., Nazari, M. R., Snyder, L. V., and Takáč, M., "A Deep Q-Network for the Beer Game: Deep Reinforcement Learning for Inventory Optimization", ***Manufacturing & Service Operations Management***, 24 (1): 285–304 (2021).

33. Hodson, T. O., Over, T. M., and Foks, S. S., "Mean Squared Error, Deconstructed", ***Journal Of Advances In Modeling Earth Systems***, 13 (12): e2021MS002681 (2021).

34. Gilroy, E. J., Hirsch, R. M., and Cohn, T. A., "Mean square error of regression-based constituent transport estimates", ***Water Resources Research***, 26 (9): 2069–2077 (1990).

35. Geman, S., Bienenstock, E., and Doursat, R., "Neural Networks and the Bias/Variance Dilemma", ***Neural Computation***, 4 (1): 1–58 (1992).

36. Heskes, T., "Bias/Variance Decompositions for Likelihood-Based Estimators", ***Neural Computation***, 10 (6): 1425–1433 (1998).

37. Hafez, M. B. and Wermter, S., "Continual Robot Learning using Self-Supervised Task Inference", ***IEEE Transactions On Cognitive And Developmental Systems***, (2023).

38. Ramaha, N. T. A., Mahmood, R. M., Hameed, A. A., Fitriyani, N. L., Alfian, G., and Syafrudin, M., "Brain Pathology Classification of MR Images Using Machine Learning Techniques", ***Computers 2023, Vol. 12, Page 167***, 12 (8): 167 (2023).

39. Alhamid, M. and Ramaha, N. T. A., "Unveiling Alzheimer's Disease via MRI: Deep Learning Approaches for Accurate Detection", ***International Journal Of Advanced Natural Sciences And Engineering Researches (IJANSER)***, 7 (10): 418–422 (2023).

40. "How I Built an AI to Play Dino Run | by Ravi | Acing AI | Medium", **https://medium.com/acing-ai/how-i-build-an-ai-to-play-dino-run-e37f37bdf153** (2024).

41. Indolia, S., Goswami, A. K., Mishra, S. P., and Asopa, P., "Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach", ***Procedia Computer Science***, 132: 679–688 (2018).

42. Volokitin, V. D., Vasiliev, E. P., Kozinov, E. A., Kustikova, V. D., Liniov, A. V., Rodimkov, Y. A., Sysoyev, A. V., and Meyerov, I. B., "Improved vectorization of OpenCV algorithms for RISC-V CPUs", (2023).

43. Porotti, R., Tamascelli, D., Restelli, M., and Prati, E., "Coherent transport of quantum states by deep reinforcement learning", ***Communications Physics 2019 2:1***, 2 (1): 1–9 (2019).

44. Gutiérrez-Moreno, R., Barea, R., López-Guillén, E., Araluce, J., and Bergasa, L. M., "Reinforcement Learning-Based Autonomous Driving at Intersections in CARLA Simulator", ***Sensors 2022, Vol. 22, Page 8373***, 22 (21): 8373 (2022).

45. Dawood, M., Dengler, N., De Heuvel, J., and Bennewitz, M., "Handling Sparse Rewards in Reinforcement Learning Using Model Predictive Control", ***Proceedings - IEEE International Conference On Robotics And Automation***, 2023-May: 879–885 (2023).

46. Lucia, S. and Karg, B., "A deep learning-based approach to robust nonlinear model predictive control", ***IFAC-PapersOnLine***, 51 (20): 511–516 (2018).

47. Yi, M., Xu, X., Zeng, Y., and Jung, S., "Deep imitation reinforcement learning with expert demonstration data", ***The Journal Of Engineering***, 2018 (16): 1567–1573 (2018).

48. Abdulrazzaq, M. M., Ramaha, N. T. A., Hameed, A. A., Salman, M., Yon, D. K., Fitriyani, N. L., Syafrudin, M., and Lee, S. W., "Consequential Advancements of Self-Supervised Learning (SSL) in Deep Learning Contexts", ***Mathematics 2024, Vol. 12, Page 758***, 12 (5): 758 (2024).

49. Bahar, K. and Ramaha, N. T. A., "Exploring Somali Sentiment Analysis: A Resource-Light Approach for Small-scale Text Classification", ***International***

Conference On Applied Engineering And Natural Sciences, 1 (1): 620–628 (2023).

CURRICULUM VITAE

Zabiullah ALI SHER graduated from Hakim-e-Nasir Khusraw Balkhi High School in Kabul, Afghanistan. Following his high school education, he pursued higher studies at RANA University, where he obtained a bachelor's degree in Software Engineering in 2013. During his undergraduate studies, Zabiullah demonstrated a keen interest in the practical applications of software engineering principles. For his thesis, he developed a comprehensive system named "Business Book," designed to maintain and calculate all insurance policies for the Insurance Corporation of Afghanistan. This project not only showcased his technical skills but also addressed a significant need within the insurance sector in Afghanistan.

Building on his academic and practical experiences, Zabiullah decided to further his education and expertise by pursuing a master's degree in Computer Engineering at Karabuk University in Turkey. At Karabuk University, he continued to expand his knowledge and skills, focusing on advanced topics in computer engineering. For his master's thesis, Zabiullah conducted research on "A Neoteric Evaluation of Deep Reinforcement Learning Algorithms Utilizing Game Concepts." This research delved into the innovative applications of deep reinforcement learning, leveraging game concepts to evaluate and enhance these algorithms.

Zabiullah's academic journey reflects a strong commitment to leveraging technology to solve real-world problems and contribute to the development of critical sectors in his home country and beyond. Through his undergraduate and graduate studies, he has consistently demonstrated a dedication to advancing his field and applying his knowledge to practical and impactful projects.