USING TASK-BASED VISUAL ATTENTION FOR CONTINUALLY IMPROVING THE PERFORMANCE OF AUTONOMOUS GAME AGENTS

OTONOM OYUN AJANLARININ PERFORMANSINI SÜREKLİ İYİLEŞTİRMEK İÇİN GÖREV TABANLI GÖRSEL DİKKAT KULLANIMI

EREN ULU

PROF. DR. TUNCA DOĞAN

Supervisor

ASSOC. PROF. DR. UFUK ÇELIKCAN

2nd Supervisor

Submitted to

Graduate School of Science and Engineering of Hacettepe University

as a Partial Fulfillment to the Requirements

for the Award of the Degree of Doctor of Philosophy

in Computer Engineering

December 2023

ABSTRACT

USING TASK-BASED VISUAL ATTENTION FOR CONTINUALLY IMPROVING THE PERFORMANCE OF AUTONOMOUS GAME AGENTS

EREN ULU

Doctor of Philosophy, Computer Engineering

Supervisor: Prof. Dr. Tunca Doğan

2nd Supervisor: Assoc. Prof. Dr. Ufuk Çelikcan

December 2023, 116 pages

Recent developments in the field of machine learning have led to the widespread acceptance of Deep Reinforcement Learning (DRL) techniques, which are a subset of machine learning, in the realm of digital intelligence. DRL allows agents to make sequential decisions and adapt their behavior through interactions with their environment, making it particularly suited for tasks that involve decision-making and learning from experience. This increasing utilization of DRL has opened new avenues for enhancing the capabilities of digital agents, enabling them to tackle complex challenges such as autonomous game playing, robotic control, and optimizing resource allocation. These advancements in DRL hold great promise for revolutionizing the ways in which intelligent agents operate in various domains more efficiently."

DRL has been effectively performed in various complex video game environments. In many game environments, DeepMind's baseline Deep Q-Network (DQN) game agents performed at a level comparable to that of humans. However, these DRL models require many

experience samples to learn and lack the adaptability to changes in the environment and handling complexity.

This thesis focusses on the specific domain of video game playing agents that have garnered significant attention due to adaptive decision-making. The study delves into the application of DRL techniques to develop and enhance game playing agents. In the first part of the thesis, we proposed Attention-Augmented DQN(AADQN) game agent by incorporating a combined top-down and bottom-up visual attention mechanism into the DQN game agent to highlight task-relevant features of input. Our AADQN model uses attention mechanis that dynamically teaches a DQN game agent how to play a game by focusing on the most task-related information. In the evaluation of our agent's performance across eight games in the Atari 2600 domain, which vary in complexity, we demonstrate that our algorithm surpasses the baseline DQN agent. Notably, our model can achieve greater flexibility and higher scores at a reduced number of time steps.

In the second section of this thesis, we address the limitations associated with employing Auxiliary Functions (AF) in DQN game agents. We investigate uxiliary strategies in some games in the Atari 2600 domain environments by integration of auxiliary functions and exploring methods to enabling more efficient and robust learning, ultimately contributing to the advancement of DQN game agent in complex and dynamic gaming environments

We demonstrate that our methods are effective in addressing the inherent inefficiency and inflexibility issues that plague the DQN, thereby marking a significant advancement in the realm of DQN game agents. By investigating the integration auxiliary functions and attention mechanism with DQN algorithms, this thesis show what can be achieved in performance improvement in autonomous game playing in Atari game envrionments. The findings and insights from this thesis are expected to contribute not only to the field of artificial intelligence but also to the broader community of gamers and developers, offering new perspectives on the creation of sophisticated and responsive game agents.

Keywords: Deep Reinforcement Learning(DRL), Deep Q-Network(DQN), Auxiliary Functions(AF), Top-Down and Bottom-up Visual Attention, Attention-Augmented DQN(AADQN)

ÖZET

OTONOM OYUN AJANLARININ PERFORMANSINI SÜREKLİ İYİLEŞTİRMEK İÇİN GÖREV TABANLI GÖRSEL DİKKAT KULLANIMI

EREN ULU

Doktora, Bilgisayar Mühendisliği

Danışman: Prof. Dr. Tunca Doğan

Eş Danışman: Doç. Dr. Ufuk Çelikcan

Aralık 2023, 116 sayfa

Son zamanlardaki makine öğrenmesi ilerlemeleri, dijital zeka alanında makine öğrenmesinin bir alt kümesi olan Derin Takviyeli Öğrenme (DTO) tekniklerinin benimsenmesini beraberinde getirmiştir. DTO, ajanların çevreleriyle etkileşim yoluyla ardışık kararlar almasını ve davranışlarını uyarlamasını sağlayarak deneyimden öğrenme ve karar verme gerektiren görevler için özellikle uygundur. DTO'nun bu artan kullanımı, dijital ajanların yeteneklerini geliştirmek için yeni olanaklar açmış ve otonom oyun oynama, robot kontrolü ve kaynak tahsisi optimizasyonu gibi karmaşık zorluklarla başa çıkmalarını sağlamıştır. DTO'daki bu ilerlemeler, akıllı ajanların çeşitli alanlarda daha verimli bir şekilde nasıl çalıştığına dair yeni bir umut taşımaktadır.

Bu tez, adaptif karar verme nedeniyle önemli ilgi toplayan video oyunu oynama ajanlarının belirli bir alanına odaklanmaktadır. Çalışma, DTO tekniklerinin oyun oynama ajanlarının geliştirilmesi ve iyileştirilmesi için uygulanmasını ele almaktadır.

DTO, çeşitli karmaşık video oyunu ortamlarında etkili bir şekilde gerçekleştirilmiştir. Birçok oyun ortamında, DeepMind'in temel olarak kullandığı Derin Q-Ağı (DQN) oyun ajanları, insan seviyesinde bir performans sergilemiştir. Bununla birlikte, bu DTO modelleri öğrenmek için birçok deneyim örneğine ihtiyaç duymakta ve çevredeki değişikliklere uyum sağlamada ve karmaşıklığı yönetmede esneklikten yoksundur.

Tezin birinci bölümünde, DQN oyun ajanına hem yukarıdan aşağıya hem de aşağıdan yukarıya bir görsel dikkat mekanizmasının birleşik olarak entegre edildiği Dikkat-Artırılmış DQN (DADQN) oyun ajanı önerilmektedir. DADQN modelimiz, girdinin görevle ilgili özelliklerini vurgulamak için dikkat mekanizmasını kullanmaktadır. Bu sayede DQN oyun ajanına bir oyunda nasıl oynanacağını dinamik olarak öğretir. Atari 2600 oyun ortamındaki sekiz oyunun performans değerlendirmesinde, karmaşıklık seviyesi farklı olan oyunlarda, algoritmamızın temel DQN ajanını geride bıraktığını gösteriyoruz. Özellikle, modelimiz daha az zaman adımında daha büyük esneklik ve daha yüksek puanlar elde edebilmektedir.

Bu tezin ikinci bölümünde, DQN oyun ajanında Yardımcı Fonksiyonların (YF) kullanımıyla ilişkili sınırlamalar ele alınmaktadır. Atari 2600 oyun ortamındaki bazı oyunlarda yardımcı fonksiyonların entegrasyonu ve daha verimli ve sağlam öğrenmeyi mümkün kılan stratejiler araştırılmaktadır. Bu çalışmalar, karmaşık ve dinamik oyun ortamlarında DQN oyun ajanının gelişimine katkıda bulunmayı amaçlamaktadır.

Bu yöntemlerimizin, DQN'yi etkileyen içsel verimsizlik ve esneklik sorunlarını ele almada etkili olduğunu gösteriyoruz, bu da DQN oyun ajanı alanında önemli bir ilerlemeyi işaret ediyor. Yardımcı fonksiyonların ve dikkat mekanizmasının DQN algoritmalarıyla entegrasyonu üzerine yapılan araştırmalar, Atari oyun ortamlarında otonom oyun oynama performansının nasıl iyileştirilebileceğini göstermektedir. Bu tezin bulguları ve içgörüleri, sadece yapay zeka alanına değil, aynı zamanda geniş bir oyunsever ve geliştirici topluluğuna da katkı sağlaması beklenmektedir. Sofistike ve tepki veren oyun ajanlarının yaratılmasında yeni perspektifler sunmaktadır.

Keywords: **Derin Takviyeli Öğrenme (DTO), Derin Q-Ağı (DQN), Yardımcı Fonksiyonların (YF), Yukarıdan Aşağıya Görsel Dikkat, Aşağıdan Yukarıya Görsel Dikkat**

ACKNOWLEDGEMENTS

I am deeply grateful to my advisor, Assoc.Prof.Dr. Ufuk Çelikcan, for his guidance and support throughout this thesis. His support and direction have played a crucial role in shaping my research. He has given me the freedom to explore my ideas and delve deep into my research. I consider myself extremely fortunate to have had the opportunity to collaborate with Prof.Dr. Tolga Çapın. He has created a welcoming research environment, where open and honest discussions are encouraged. I would like to extend my gratitude to Asst.Prof.Dr. Bora Çelikkale for our meaningful discussions held during our time working together in the office. These discussions have had a profound impact on shaping my thoughts and have greatly influenced my thinking.

CONTENTS

Page

TABLES

FIGURES

ABBREVIATIONS

Abbreviations

DRL	:	**D**eep **R**inforcement **L**earning
RL	:	**R**inforcement **L**earning
DQN	:	**D**eep **Q**-**N**etwork
DNN	:	**D**eep **N**eural **N**etworks
CNN	:	**C**onvolutional **N**eural **N**etwork
LRP	:	**L**ayer-wise **R**elevance **P**ropagation
FDP	:	**F**eature **D**ecomposed **P**ixel
AADQN	:	**A**ttention **A**ugmented **D**eep **Q**-**N**etwork
LSTM	:	**L**ong **S**hort-**T**erm**M**emory
SAM	:	**S**aliency **A**ttentive **M**odel
RNN	:	**R**ecurrent **N**eural **N**etwork
FCN	:	**F**ully **C**onnected **N**etwork
AF	:	**A**uxiliary **F**unction
ADF	:	**A**uxiliary **D**istance **F**unction
PER	:	**P**rioritized **E**xperience **R**eplay
ALE	:	**A**arcade **L**earning **R**environment
VAM	:	**V**isual **A**ttention **M**echanism
DDQN	:	**D**ouble **D**eep **Q**-**N**etwork
NN	:	**N**eural **N**etworks
VQA	:	**V**isual **Q**uestion **A**nswering
TDA	:	**T**op **D**own **A**ttention
BUA	:	**B**ottom **U**p **A**ttention
PBA	:	**P**reliminary **B**ottom-up **A**ttention
BAR	:	**B**ottom-up **A**ttention **R**efinement
SL	:	**S**upervised **L**earning

NLP	:	**N**atural **L**anguage **P**rocessing
ASR	:	**A**utomatic **S**peech **R**ecognition
SSL	:	**S**emi **S**upervised **L**earning
GANs	:	**G**enerative **A**dversarial **N**etworks
MDP	:	**M**arkov **D**ecision **P**rocesses
TN	:	**T**arget **N**etwork
PN	:	**P**rediction **N**etwork
AR	:	**A**uxiliary **R**ewards
PF	:	**P**article **F**ilter
ER	:	**E**xperience **R**eplay
TD	:	**T**emporal **D**ifference
DNNA	:	**D**ueling **N**eural **N**etwork **A**rchitecture
DDRA	:	**D**istributed **D**eep **R**einforcement **E**Architecture
DMAE	:	**D**istance **M**ean **A**bsolute **E**rror
A3C	:	**A**synchronous **A**dvantage **A**ctor **C**ritic
ML	:	**M**achine **L**earning
MFRL	:	**M**odel **F**ree **R**einforcement **L**earning
MBRL	:	**M**odel **B**ased **R**einforcement **L**earning
MCTS	:	**M**onte **C**arlo **T**ree **S**earch
VBRL	:	**V**alue **B**ased **R**einforcement **L**earning
PBRL	:	**P**olicy **B**ased **R**einforcement **L**earning
PPO	:	**P**roximal **P**olicy **O**ptimization

ABBREVIATIONS

Symbols

I	:	input frame
$\boldsymbol{\mathcal{F}}$	:	set of feature maps
$\boldsymbol{V}$	:	importance vector
V_j	:	features' importance value
$\boldsymbol{F_j}$	:	specific feature map
$\boldsymbol{\omega_i}$	:	particle vector
$\boldsymbol{A}$	:	attention vector
D	:	number of feature maps
$\boldsymbol{H}$	:	saliency map
p_j	:	probability of particle
R_t	:	reward
$Q(s_t, \hat{\omega}_i)$	:	state value of particle
$\hat{\omega}_i^j$	:	normalized particle
$\mathbf{P_j}$	:	feature decomposed pixel
H_j	:	feature saliency value
W_j	:	refinement vector
F'	:	refined feature map

1. INTRODUCTION

Reinforcement Learning (RL) has proven highly successful in addressing numerous tasks, including Atari games within the Arcade Learning Environment (ALE) [4], where the sequence of environmental observations serves as the basis for determining decisions [5]. RL algorithms process environmental data to learn a policy that chooses the best action to maximize cumulative reward [6]. During RL, the agent interacts with its environment to arrive at different states by performing actions that cause the agent to obtain positive or negative rewards. Nevertheless, the limited adaptability of RL approaches poses challenges when dealing with complex tasks. Consequently, developing methods that enable the application of RL to complex environments is a significant research problem [6]. The goal is to enhance the capabilities of RL algorithms to effectively handle intricate tasks, allowing for more robust and efficient learning in complex scenarios. The combination of Deep Neural Networks (DNNs) and Q-Learning [7] led to the Deep Q-Network (DQN) algorithm [5, 8], which has been used in various works to develop models for complex tasks. These agents demonstrate remarkable success, surpassing human-level performance and outperforming baseline benchmarks [9].

However, the DQN algorithm can suffer inefficiency and inflexibility, which can limit its performance [9, 10]. It is vulnerable in complex environments regarding data efficiency, as there is an infinite number of possible experiences in such environments, and there is a need to process many states, requiring high computational power [11]. The DQN algorithm has received criticism for its need for more flexibility, specifically when adapting to changes in the environment or incorporating new tasks. In such cases, the algorithm typically requires a complete restart of the learning process, which can be time-consuming and inefficient [10]. It also has certain limitations in terms of generalization compared to regular Neural Networks (NNs) [12].

Researchers have developed a range of extensions to the original DQN algorithm to address these issues, such as Double Deep Q-Network (DDQN) [13]. Moreover, various studies

have suggested employing of Visual Attention Mechanism (VAM) [14–17], which allows the network to focus on specific regions of an input image rather than processing the entire image at once [14].

The attention mechanism can be implemented in various ways, like the utilization of Convolutional Neural Networks (CNNs) for image feature extraction, Recurrent Neural Networks (RNNs), and Visual Question Answering (VQA) models that process the textual question input [18–20]. The VQA attention mechanism serves the purpose of selectively directing attention toward image regions that are crucial for answering a question. The attention mechanism effectively prioritizes the attended regions by assigning importance scores or weights to different image regions based on their relevance to the question. As a result, these regions are granted a higher degree of significance in the overall analysis [18–23].

In the first part of this thesis, we address the aforementioned limitations of the baseline DQN agent and propose an Attention-Augmented Deep Q-Network (AADQN) by extending DQN with a dual attention mechanism to highlight task-relevant features of the input. The dual attention mechanism unifies bottom-up and top-down visual attention mechanisms within the AADQN model, as illustrated in Figure 1.1. This way, AADQN allows the agent to efficiently concentrate on the most relevant parts of the input image. The Top-Down Attention (TDA) incorporates particle filters, enabling dynamic identification of task-related features. The integration of particle filters provides a regularization effect, effectively mitigating overfitting and enhancing generalization. The Bottom-Up Attention (BUA) comprises two components: a Preliminary Bottom-up Attention (PBA) module and a Bottom-up Attention Refinement (BAR) module. The PBA extracts the feature importances, which TDA subsequently utilizes to initialize the particle filters. BAR then refines attention by considering the saliency value of the features.

AADQN differs from the previous work using visual attention with DQN [14–17] by refining top-down focus through bottom-up attention. The collaboration of both attention mechanisms enables more accurate decision-making and improves the model's robustness

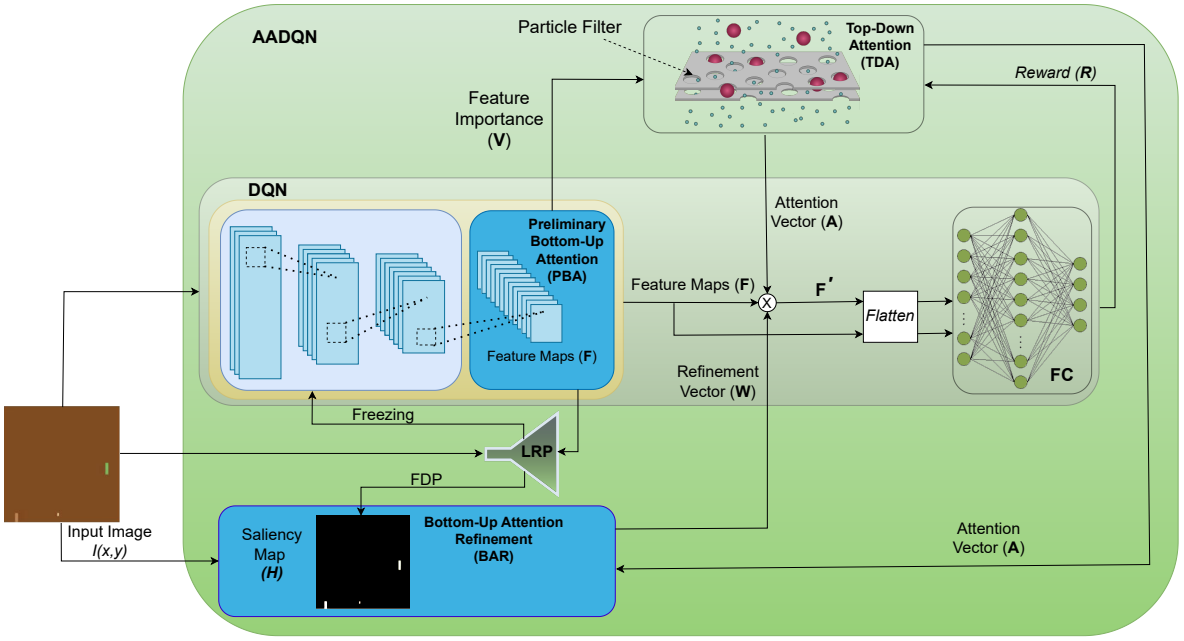


Figure 1.1 Our proposed Attention-Augmented Deep Q-Network extends DQN with a particle filter-enhanced dual attention mechanism as follows: The top-down attention (TDA) mechanism employs particle filters to compute the attention vector. The bottom-up attention mechanism comprises two components: the preliminary bottom-up attention (PBA) and the bottom-up attention refinement (BAR). PBA determines the feature importance for initializing the particle states. BAR enhances focus by considering the saliency map of the input. Layer-wise Relevance Propagation (LRP) determines the feature decomposed pixels of the input and freezes the unimportant neurons on CNN.

to variations, reducing its sensitivity to noise. It is also fundamentally different from the VQA-based RL approaches, which aim to align visual and textual information to make decisions and are typically applied to tasks where understanding and reasoning about visual and textual data are essential, such as answering questions about the content of an image [18–23]. Our dual attention mechanism, enhanced by a particle filter, can be applied to various tasks without relying on textual information, enabling AADQN to adaptively focus on different regions of the input.

1.1. Scope Of The Thesis

We will explore the theoretical underpinnings, implementation strategies, and empirical results of incorporating Auxiliary Functions (AFs) and attention mechanisms into the DQN framework. These mechanisms accelerate the convergence of DQN but also significantly improve its ability to navigate complex environments. Through analytical investigation and extensive experimentation, this thesis aims to provide a comprehensive assessment of the potential benefits and trade-offs associated with these enhancements.

1.2. Contributions

In this research, we cover the inefficiency and inflexibility deficiencies of DQN by proposing a novel, simple, and efficient DRL approach. With the incorporation of auxiliary functions and attention mechanisms with baseline DQN, this work effectively addressed these inherent limitations of DQN. Through extensive experimentation proposed methods have proven to be an effective approach in tackling the efficiency and flexibility challenges of DQN in the domain of game-playing agents.

Overall, the main contributions of this thesis can be summarized as follows:

- The AADQN introduces a novel particle filter-based attention approach to DQN, integrating bottom-up and top-down attention mechanisms. This unified approach handles the complexity of the environment by extracting essential task-related features, thereby enhancing efficiency and improving overall performance.
- AADQN enhances the flexibility performance of DQN by freezing the unimportant or irrelevant units of CNN's inner layers during the decision-making process. These units, identified by low relevance scores assigned by AADQN, undergo transfer learning to optimize the model's robustness and flexibility.
- Offering an auxiliary function mechanism for enhancing the learning efficiency of DQN. By guiding the agent's exploration by providing supplementary reward

feedback, facilitating faster learning, improving exploration, reducing overestimation, and making DQN more robust and adaptable, ultimately resulting in improved performance.

1.3. Organization

This thesis is organized as follows: Chapter 1 presents the research introduction and contributions. Then, the background of reinforcement learning and an overview of the related work are presented in chapters 2 and 3, respectively. The proposed algorithm and algorithmic efficiency analysis are presented in Chapter 4. These are followed by the experiment results and ablation study reported in Chapter 5. Finally, Chapter 6 and 7 discussed the results and concludes the thesis respectively.

The organization of the thesis is as follows:

- Chapter 1 presents our motivation, contributions, and the scope of the thesis.
- Chapter 2 provides the background of RL, Q-Learning, and Deep Q-Learning
- Chapter 3 gives a literature studies
- Chapter 4 introduces the proposed methods and important components
- Chapter 5 demonstrates experimental results and ablation study
- Chapter 6 provides discussion, insights, and interprets the results
- Chapter 7 states the summary of the thesis, conclusion, limitations, and possible future directions.

2. BACKGROUND OVERVIEW

In recent years, machine learning has witnessed remarkable advancements, with notable progress in both Supervised Learning (SL) and RL techniques. SL algorithms aim to learn connections between input data and the corresponding labels by leveraging annotated datasets. This approach has found success in various domains, such as image classification, Natural Language Processing (NLP), and Automatic Speech Recognition (ASR). On the other hand, RL focuses on training agents to make consecutive decisions by engaging with an environment and obtaining feedback in the form of rewards. RL algorithms learn through trial and error, exploring different actions and optimizing their strategies to maximize cumulative rewards. The fundamental concept underlying RL is Q-learning, a popular algorithm that estimates the optimal action-value function, commonly referred to as the Q-function. Q-learning has proven effective in solving various RL problems. However, the applicability of Q-learning to complex, high-dimensional environments has been limited. To address this challenge, the emergence of Deep Reinforcement Learning (DRL) has revolutionized the field by combining RL with Deep Neural Networks (DNN). One prominent example of DRL is the Deep Q-Network (DQN), which utilizes a Deep Neural Network (DNN) as a function approximator to calculate the Q-value. DQN has demonstrated remarkable capabilities in learning directly from raw sensory inputs, such as images, and has achieved significant breakthroughs in challenging tasks, including playing Atari 2600 games at a human-level performance. In this section, we offer a summary of these fundamental concepts, including supervised learning, reinforcement learning, Q-learning, deep reinforcement learning, and the seminal Deep Q-Network algorithm.

2.1. Machine Learning (ML)

Within the realm of machine learning, supervised and unsupervised learning stand as two fundamental approaches. Supervised learning is a learning paradigm in which an algorithm learns from labeled examples. In this methodology, a dataset is presented, consisting of input data matched with corresponding output labels. The goal is to train a model that can

accurately map new, unseen inputs to the correct output label. The model learns to generalize patterns from the labeled data, allowing it to generate predictions for unseen instances. Frequently employed algorithms in supervised learning encompass decision trees, support vector machines, and neural networks. On the other hand, unsupervised learning involves learning patterns from unlabeled data. In contrast to supervised learning, unsupervised learning lacks access to explicit output labels. Instead, the algorithm attempts to discover clusters within the data. Common unsupervised learning techniques involve clustering algorithms such as k-means. In recent years, there has been growing interest in combining supervised and unsupervised learning techniques, such as Semi-Supervised Learning (SSL) and Generative Adversarial Networks (GANs). These hybrid approaches aim to leverage the strengths of both paradigms to overcome data labeling challenges and enhance learning performance.

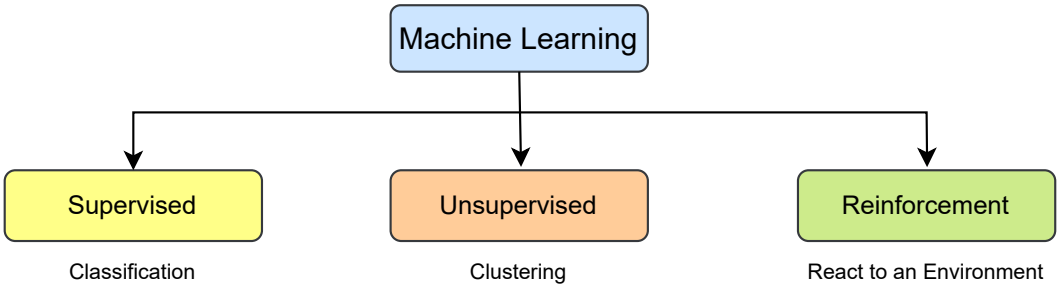


Figure 2.1 Machine Learning

Another prominent subfield of machine learning is RL which focuses on the concept of an agent learning how to engage with an environment to accomplish a particular goal. Unlike supervised and unsupervised learning, RL does not rely on labeled data or predefined patterns. Instead, the RL agent learns by engaging in trial and error, receiving feedback in the form of rewards or penalties contingent on its actions. The RL paradigm draws inspiration from how humans and animals learn by interacting with their surroundings. The agent explores the environment, takes action, and observes the resulting state changes and rewards. Through this iterative process, The agent endeavors to learn an optimal policy that maximizes the cumulative rewards over the long term. The Q-learning algorithm stands out

as one of the most widely utilized reinforcement learning algorithms, which estimates the value of state-action pairs and uses an iterative process to optimize the policy.

2.1.1. Reinforcement Learning Applications

Reinforcement Learning (RL) has found diverse applications across a wide range of fields, showcasing its versatility and potential for solving complex decision-making problems. Here are some examples of different application areas where RL has been successfully employed:

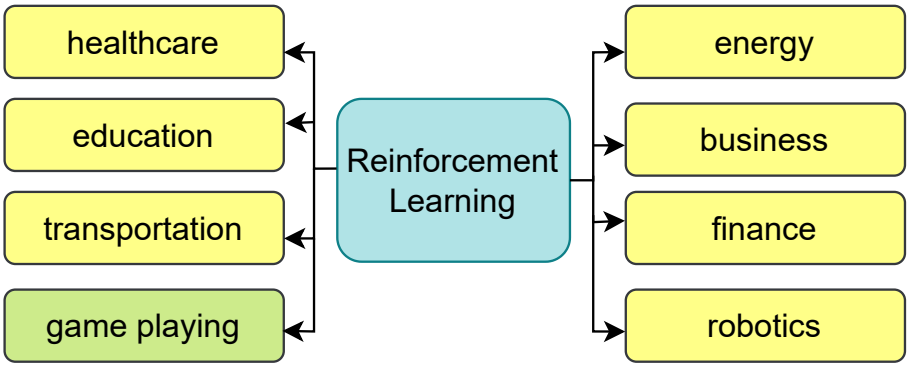


Figure 2.2 Reinforcement Learning Applications

RL has made significant breakthroughs in game playing, surpassing human performance in various domains. It has been applied to classic board games like chess and Go, as well as digital games, enabling agents to learn complex strategies and tactics. RL agents can explore different moves, and optimize their gameplay through trial and error learning. Through RL, game playing agents engage in a process of continuous learning and decision-making, driven by the pursuit of maximizing long-term rewards. These agents explore the game environment, engaging in actions and obtaining feedback in the form of rewards or penalties, which guide their learning process. RL algorithm, enable agents to assess the value of various game states or combinations of states and actions, progressively improving their decision-making abilities. In the realm of digital games, RL has enabled game playing agents to conquer complex virtual worlds. From classic arcade games to modern first-person shooters, RL agents have proven their ability to learn and master diverse game mechanics, adapt to dynamic environments, and achieve high scores that rival or surpass human players.

By employing techniques similar to deep neural networks, RL agents can directly learn from raw pixel inputs. This thesis explores the utilization of Reinforcement Learning (RL) in the realm of game playing, aiming to investigate the effectiveness and potential of RL algorithms in mastering complex games. By leveraging RL techniques, we seek to develop intelligent game playing agents capable of autonomous learning and optimal decision-making. The objective is to analyze the performance of RL algorithms in different game environments, assess their ability to adapt to dynamic games. This research contributes to the advancement of RL in game playing and practical implications of employing RL algorithms to enhance game playing capabilities. Through empirical evaluations we aim to provide valuable insights into the application of RL in game playing, laying the groundwork for future advancements in this exciting field. The impact of RL in game playing extends beyond mere entertainment. It serves as a testbed for developing and refining RL algorithms, pushing the boundaries of AI research. The insights gained from RL in game playing have far-reaching implications, finding applications in other domains such as robotics, and decision-making systems.

2.2. Reinforcement Learning

In the ever-changing landscape of artificial intelligence and machine learning, reinforcement learning stands out as a cornerstone of autonomous decision-making and problem-solving. Based on the wider domain of machine learning, reinforcement learning is a framework that enables intelligent agents to learn and adapt to complex environments by interactions with their surroundings. It forms the basis for developing agents capable of making sequential decisions and enhancing their behavior gradually.

2.2.1. Objective: Maximize the Cumulative Reward

RL draws inspiration from the idea of learning through trial and error, akin to the learning processes observed in humans and animals acquire new skills and adjust to different situations. RL uses Markov Decision Processes (MDP) as a foundational model for these

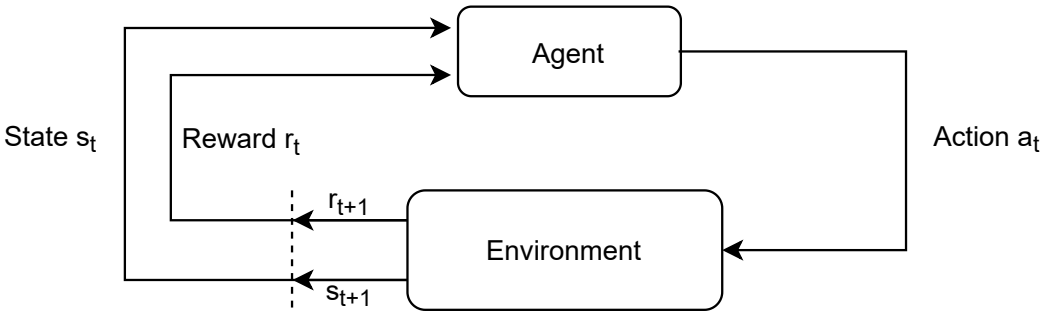


Figure 2.3 RL scenario

problems. It offers a structured and mathematical framework for creating intelligent agents that can make informed decisions, learn from their experiences, and maximize their overall rewards.

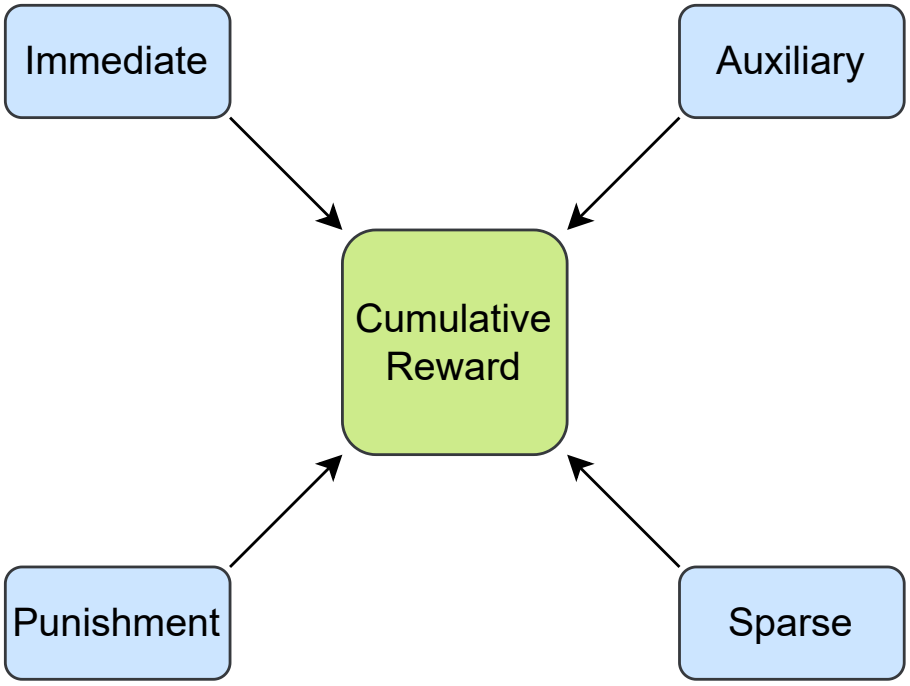


Figure 2.4 Maximize the cumulative reward

RL plays a role in various applications, including autonomous robotics, game-playing, supply chain optimization, and financial portfolio management. It is also found in practical applications in real-world situations, such as the ability to make sequential decisions in dynamic and uncertain environments.

2.2.2. Variety of Reinforcement Learning Methods

Reinforcement Learning (RL) encompasses a diverse set of algorithms that aim to enable intelligent decision-making in dynamic environments. Two prominent categories within RL are model-free and model-based algorithms, each with its own unique approach to learning and decision-making.

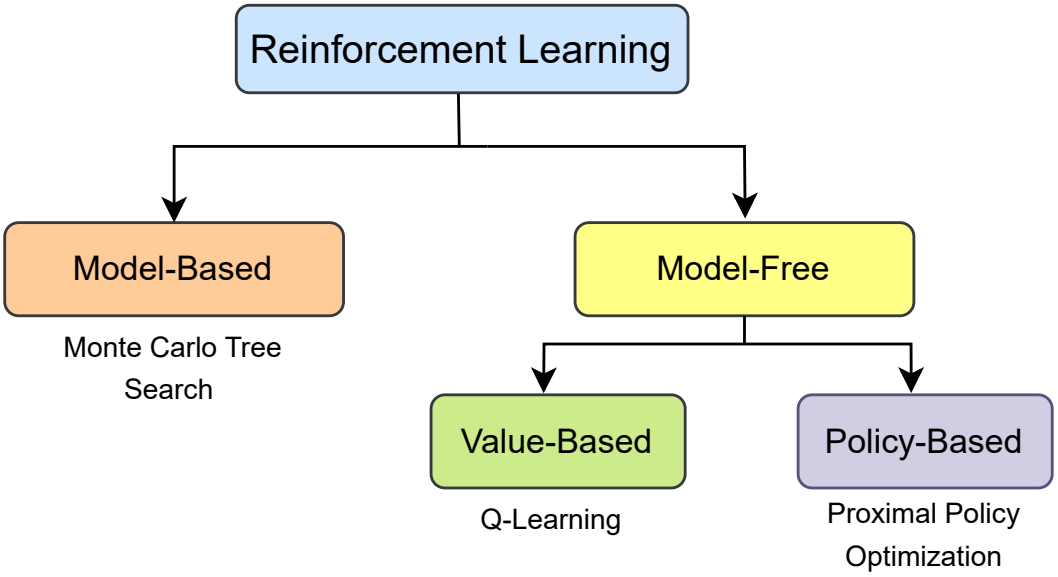


Figure 2.5 Reinforcement Algorithms

Model-Free RL (MFRL) algorithms learn directly from interacting with the environment, without explicitly constructing a model or representation of the environmental dynamics. These algorithms focus on optimizing the decision-making process by iteratively updating policies or value functions based on observed rewards and states. Model-free methods excel in scenarios where the environment is complex, uncertain, or difficult to model accurately. They rely on trial and error learning, exploring the environment to discover the best actions to take in different states. Model-free algorithms, such as Q-learning, SARSA, and Deep Q-Networks (DQN), serve as examples in this category.

In contrast, Model-Based RL (MBRL) algorithms explicitly construct a model or representation of the dynamics within the environment. This model captures the probabilities of transitioning between states and the anticipated rewards that linked to various actions.

By learning and utilizing this model, the agent has the capability to simulate or predict the outcomes of its actions and make decisions accordingly. Model-based methods aim to achieve more efficient decision-making by planning and optimizing actions using the learned model. These algorithms can be advantageous in situations where an accurate model can be learned and where planning can be performed effectively. Model-based algorithms, exemplified by Monte Carlo Tree Search (MCTS), fall into this category.

Both model-free and model-based RL algorithms have their strengths and weaknesses. Model-free methods offer simplicity and flexibility, as they do not require an explicit model and can handle complex environments. However, they may require more exploration and data collection to discover optimal policies. Model-based methods, on the other hand, enable more efficient planning and decision-making by utilizing a learned model but rely on the accuracy of the model and may be computationally demanding.

The selection between model-free and model-based RL approaches is contingent upon the distinct characteristics of the problem domain, the availability of data or prior knowledge, and the trade-offs between exploration and exploitation. Researchers and practitioners often select the most suitable approach based on the nature of the environment and the desired balance between learning efficiency and decision-making performance.

2.2.3. Model-free: Policy-based and Value-based

In the field of model-free RL, two fundamental categories of algorithms are value-based and policy-based methods, each with its own distinctive approach to learning and decision-making.

Value-Based RL (VBRL) algorithms concentrate on estimating the value of various states or combinations of states and actions. The algorithms aim to learn a value function, such as the Q-function. This function represents the expected cumulative rewards that an agent can obtain by following a particular policy. Value-based methods aim to determine the most favorable values for states or state-action pairs by continuously adjusting the value

function using observed rewards and states. Through iterative updates, these methods seek to optimize the estimation of values that represent the expected cumulative rewards associated with different states or state-action pairs. After the value function has been learned, it can be employed to derive an optimal policy. This is achieved by selecting actions that maximize the estimated value associated with each state or state-action pair. By utilizing the learned value function, the agent can make decisions that are expected to yield the highest cumulative rewards in a given environment. Prominent examples of value-based algorithms encompass Q-learning, SARSA, and Deep Q-Networks (DQN). These algorithms are widely recognized and have contributed significantly to the field of reinforcement learning.

Indeed, Policy-Based Reinforcement Learning (PBRL) algorithms take a different approach by directly learning a parameterized policy, which serves as a mapping from states to actions. Rather than estimating the values of states or state-action pairs, PBRL algorithms focus on optimizing the policy itself. Rather than focusing on determining the values of states or state-action pairs, these methods aim to find the best policy that maximizes the expected cumulative rewards. Policy-based methods employ iterative updates to the policy using observed rewards and states, with the goal of finding the policy that maximizes the expected cumulative rewards. These algorithms are capable of handling both discrete and continuous action spaces and have the ability to learn stochastic policies, which assign probabilities to different actions.

One notable example of a policy-based algorithm is REINFORCE (REward Increment = Nonnegative Factor × Offset Reinforcement × Characteristic Eligibility). REINFORCE utilizes the policy gradient method to optimize the policy parameters. By estimating the gradient of the expected cumulative rewards with respect to the policy parameters, REINFORCE updates the policy in the direction that maximizes the rewards. This algorithm is particularly suitable for problems with high-dimensional action spaces and has been applied successfully in various domains.

Another example is Proximal Policy Optimization (PPO), which is a state-of-the-art policy optimization algorithm. PPO aims to strike a balance between sample efficiency and stability

in policy updates. It utilizes a trust region approach to ensure that policy updates do not deviate too far from the previous policy, preventing abrupt and unstable changes. PPO has shown strong performance in a wide range of tasks and has become a popular choice for policy optimization in recent years.

These policy-based algorithms, along with various other approaches like Trust Region Policy Optimization (TRPO), Actor-Critic methods, and evolutionary algorithms, contribute to the rich landscape of policy-based reinforcement learning. They offer effective strategies for directly optimizing policies, enabling agents to learn complex decision-making strategies and achieve high performance in diverse environments.

Value-based and policy-based methods have distinct characteristics and trade-offs. Value-based methods often excel in environments with large state or action spaces as they focus on estimating the value of states or state-action pairs. They are effective in scenarios where finding an optimal policy is the primary objective. Policy-based methods, conversely, are suitable for continuous action spaces and can directly learn stochastic policies. They can handle both exploration and exploitation more naturally and can be effective in scenarios where finding the optimal policy is challenging.

In practical application, the decision between value-based and policy-based approaches is influenced by the specific characteristics of the problem domain, the nature of the state and action spaces, and the desired balance between exploration and exploitation. Researchers and practitioners often select the most appropriate approach based on the characteristics of the task and the objectives of the RL problem at hand.

2.3. Q-Learning

Q-learning is a fundamental reinforcement learning algorithm employed to train an agent in making decisions within an environment with the aim of maximizing its cumulative long-term rewards. In Q-learning, the agent learns through interactions with the environment (Figure 2.6). The environment is commonly modeled as a Markov Decision Process (MDP), comprising a set of states, actions, transition probabilities, and associated rewards. The

objective of the agent is to acquire an optimal policy, characterized as a mapping from states to actions that maximizes the expected cumulative reward over time.

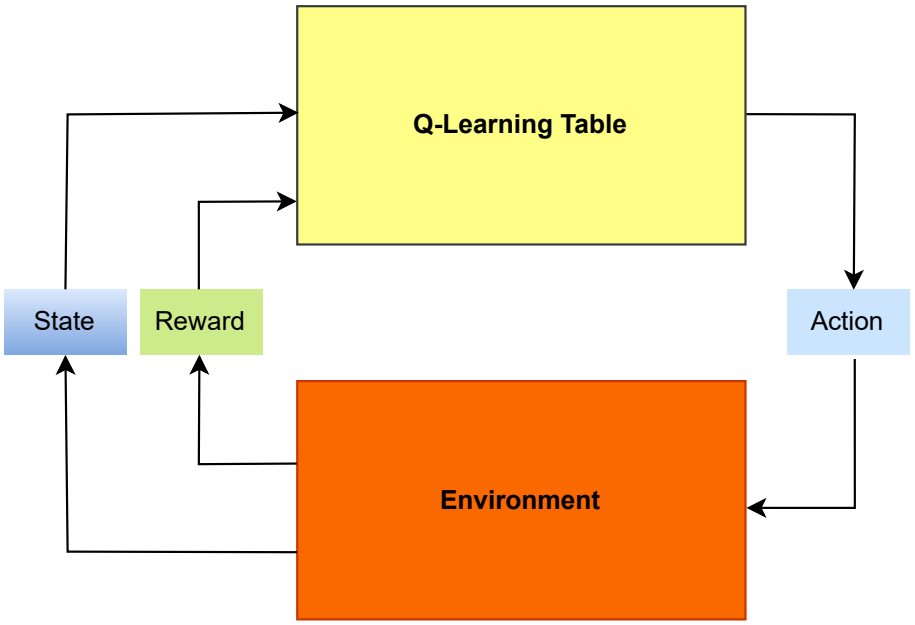


Figure 2.6 Q-Learning

Central to Q-learning is the Q-value, a measure that signifies the anticipated cumulative reward for executing a specific action in a given state. The Q-value undergoes iterative updates influenced by the agent's experiences in the environment. The agent explores the environment by taking actions, observes the resultant state and reward, and adjusts the Q-values accordingly. The agent aims to execute a sequence of actions that ultimately yields the maximum total reward, often referred to as the Bellman equation, which we will formalize as part of the learning process: [24]:

$$Q(s,a) = R(s,a) + \gamma maxQ(s',a') \tag{1}$$

According to the Bellman equation, Q-learning learns a function that receives the value for each action by receiving state and action and updates the Q-values. It iteratively updates the Q-values for state-action pairs based on the immediate reward received and the maximum Q-value achievable in the next state. As the agent progressively explores and interacts more

with the environment, the Q-values converge to their optimal values, signifying the best actions to take in each state.

In the above; the Q-value for a state s when taking action a is calculated as the sum of the immediate reward $R(s, a)$ and the highest Q-value achievable from the subsequent state s'. The variable γ, referred to as the discount factor, governs the extent to which future rewards contribute to this calculation. $Q(s', a)$ again depends on $Q(s", a)$ which will then have a coefficient of gamma squared. Thus, the Q-value is contingent on the Q-values of future states, as illustrated here:

$$Q(s, a) \leftarrow \gamma Q(s', a) + \gamma^2 Q(s", a)...\gamma^n Q(s^n, a) \tag{2}$$

Modifying the value of gamma will decrease or increase the impact of future rewards. The agent executes sequences of actions that will yield the maximum total reward. With experience, it will converge to the optimal policy. The Q-value is formalized as follows [24]:

$$Q(s, a) \leftarrow Q(s, a) + \alpha\big[R(s, a) + \gamma max Q(s', a') - Q(s, a)\big] \tag{3}$$

where α is the learning rate, and $\big[R(s, a) + \gamma max Q(s', a') - Q(s, a)\big]$ is called the Temporal Difference (TD) error.

2.3.1. Markov Decision Processes (MDP)

Markov Decision Processes (MDPs) serve as mathematical frameworks utilized to model sequential decision-making problems within a stochastic environment.

An MDP is defined by a tuple (S, A, P, R, γ), where:

- S represents the range of potential environmental states.

- A denotes the set of available actions that the agent can take.
- P defines the state transition probabilities. It determines the likelihood of transitioning between states upon taking a specific action.
- R captures the reward assigned to the agent immediately when moving from one state to another.
- γ The discount factor (gamma) influences the relative significance of future rewards in comparison to immediate rewards.

MDPs (Markov Decision Processes) are built on the assumption of the Markov property. This property asserts that the future state and reward solely rely on the present state and action, irrespective of the past sequence of states and actions. This property simplifies the modeling process and allows for efficient algorithms to be developed. The primary goal is to discover an optimal policy that maximizes the expected cumulative rewards over time. The objective is to identify the policy that guides the agent's actions in a way that leads to the highest possible total reward as it interacts with the environment over multiple time steps. By finding the optimal policy, the agent can make informed decisions to maximize its long-term rewards and achieve its objectives in the given MDP.

Q-learning is indeed a value-based reinforcement learning (RL) algorithm that is commonly employed to learn an optimal policy within Markov Decision Processes (MDPs). Q-learning seeks to approximate the value of state-action pairs through the use of the Q-function. This function estimates the anticipated cumulative rewards that an agent can achieve by selecting a particular action within a given state. During the exploration of the environment, the algorithm faces a trade-off between exploration and exploitation. It selects actions based on this trade-off, gradually updating the Q-values towards their optimal values. Through an iterative process, Q-learning converges to these optimal Q-values, enabling the agent to choose actions that maximize the expected cumulative rewards.

The relationship between MDPs and Q-learning is that Q-learning is a specific algorithm designed to solve MDPs. MDPs provide the formal framework for modeling sequential

decision-making problems, while Q-learning is an algorithmic approach to learn an optimal policy within the MDP framework. By leveraging the Q-function, Q-learning provides a means to approximate the potential value associated with taking specific actions in particular states, enabling the agent to make informed decisions based on these value estimates.

In summary, Markov Decision Processes (MDPs) serve as a mathematical framework for modeling sequential decision-making problems. Within this framework, Q-learning is a specific value-based reinforcement learning algorithm used to learn an optimal policy. Q-learning achieves this by estimating the Q-values, which represent the expected cumulative rewards for state-action pairs. The algorithm iteratively updates the Q-values based on observed rewards and state transitions, enabling the agent to gradually converge towards an optimal policy for solving MDPs.

2.4. Deep Q-Learning

Reinforcement learning has showcased notable success across diverse domains, encompassing robotics, game playing, and autonomous vehicles. It has been used to train agents that can defeat world champions in board games like chess and Go, master complex video games, and even control real-world robotic systems.

Despite its successes, RL faces challenges such as sample inefficiency, exploration-exploitation trade-offs, and generalization to new tasks. Researchers continue to explore advancements in RL algorithms, including deep reinforcement learning and combining RL with deep neural networks to overcome these challenges and push the boundaries of what RL can achieve.

Deep Q-Learning merges traditional Q-Learning with deep neural networks in reinforcement learning. Its objective is to empower agents to discover the best actions in intricate tasks. Utilizing a neural network to estimate the Q-function, which predicts the anticipated total reward for each action in a specific state, allows Deep Q-Learning to effectively navigate environments with extensive state spaces.

In Q-Learning, the agent is familiar with the anticipated reward for each action in every step. This is akin to equipping the agent with a practical reference table, enabling it to determine precisely which action to undertake. (Figure 2.6). According to this table, the agent executes a series of actions designed to yield the maximum total reward over time. However, if this reference guide becomes excessively long, it transforms into a table with millions of cells, leading to a loss of control. This presents challenges related to both memory requirements and feasibility. The memory needed to store and update the table would grow with the expanding number of states and the time required to explore each state for constructing the table would become impractical. Thus, the methodology revolves around estimating Q-values through machine learning models, particularly employing a neural network (Figure 2.6). This conceptualization paved the way for the creation of DeepMind's advanced deep Q-network (DQN) algorithm.

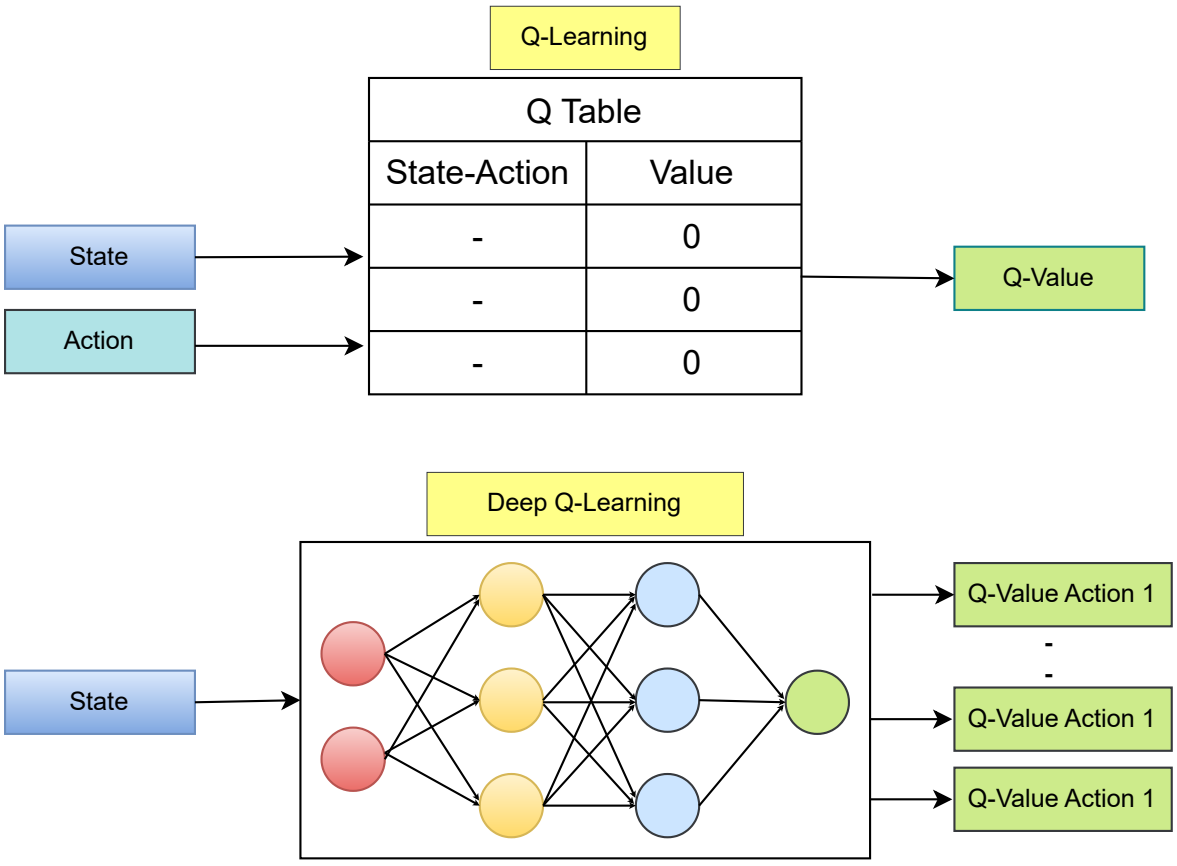


Figure 2.7 Deep Q-Learning

In response to the limitations of fundamental Q-learning, such as slow convergence and the need for manual feature engineering, Mnih et al. proposed a DQN [5] to utilize deep neural networks to approximate the learning function, allowing for more complex and efficient learning directly from raw sensory inputs. The DQN architecture (Figure 2.10) uses a CNN to process the input image representing the game state and produces Q-values for all available actions. The CNN's convolutional layers extract important features, generating a feature map. This feature map is subsequently flattened and inputted into a Fully Connected Network (FCN), responsible for computing the Q-values for the actions in the current state.

2.5. Target/Predict Network

In supervised neural networks, the learning process involves a fixed target at each step, in which the network's parameters are updated contingent on the disparity between the predictions and the ground truth labels. However, in Reinforcement Learning (RL), the learning paradigm differs significantly. In RL, there is no predetermined fixed target that the agent aims to match. Instead, the target itself is estimated, and this estimation is subject to change during the learning process.

The absence of a fixed target in RL is a fundamental characteristic that sets it apart from supervised learning. In reinforcement learning, the agent engages with an environment by taking actions and receiving feedback in the form of rewards (Figure 2.8). The agent's objective is to maximize its cumulative rewards over time. To achieve this, RL algorithms employ a trial-and-error approach, exploring different actions and learning from the subsequent consequences.

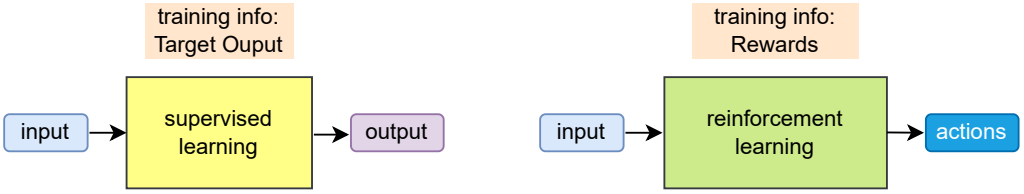


Figure 2.8 Target in Reinforcement Learning

However, changing the target during the training process leads to unstable learning. To resolve this issue, DQN uses a dual neural network architecture, referred to as the prediction and target network, in which there are two Q-networks with identical structures but different parameters (Figure 2.9). Learning is mainly achieved by updating the prediction network at each step to minimize the loss between its current Q-values and the target Q-values. The target network is created as a replica of the prediction network, but with less frequent updates. Typically, the weights of the prediction network are copied to the target network every n steps. This way, the target network serves to maintain stability and to prevent the prediction network from overfitting to the current data by keeping the target values fixed for a window of n time steps.

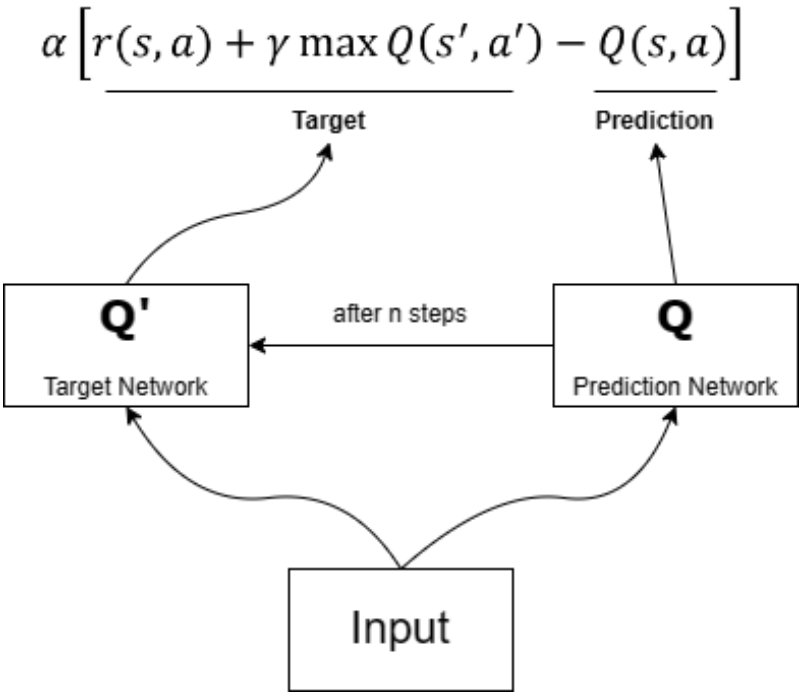


Figure 2.9 Prediction and Target Network

Target Network (TN) keeps the values of weight coefficients constant during training, and at the end of each n step, all the weight coefficients related to the Prediction Network (PN) are copied to this network. In other words, all target network parameters are fixed within n time intervals. The loss function is the difference between the predicted value and the target value. The max value for the next state is received from the target network.

2.6. Architecture of DQN:

In this work, the image of the game screen is the state of the environment. Therefore, we only feed the image of the game screen as an input to the DQN. As shown in Figure 2.10, DQN includes a convolutional neural network, which uses this input image (game state) as an input, and outputs the Q value of all the actions in the game state. The convolutional layers extract features from the image and produce a feature map. Next, the algorithm flattens the feature map and feeds this flattened feature map as an input to the fully connected network. The fully connected network takes this feature map as an input and returns the Q value of all the actions in the state (Figure 2.10).

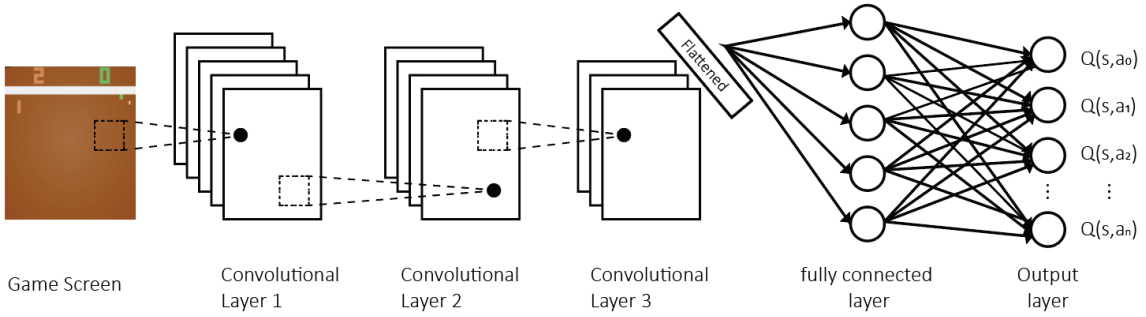


Figure 2.10 DQN Architecture

2.7. Auxiliary Rewards (AR):

In RL, the main focus is on maximizing rewards. However, in many fascinating scenarios, these rewards are rarely received, which creates challenges in terms of what and how to learn when they are not available. Auxiliary Rewards (AR) refers to an additional objective that is incorporated into the learning process to assist the main RL task. The purpose of an auxiliary reward is to provide supplementary information or guide the learning process to improve performance or accelerate convergence. The Auxiliary Reward (AR) can manifest in various forms, contingent on the specific reinforcement learning problem. It can be designed to capture additional information about the environment, provide extra supervision signals, or guide the agent's exploration and decision-making. In reinforcement learning, the reward signal plays a pivotal role in steering the agent's behavior. By designing an auxiliary reward, additional rewards can be provided to steer the agent towards desired behaviors. This can help in shaping the agent's policy and accelerating the learning process. Another application of auxiliary reward is in the context of multi-task learning. When dealing with multiple related tasks, auxiliary rewards can be used to learn features that are beneficial for all tasks. By jointly optimizing the main task and auxiliary tasks, the agent can improve performance across all tasks. Auxiliary rewards can also be employed as regularization or exploration mechanisms. For instance, in exploration, An auxiliary reward can direct the agent to explore unvisited states or take actions that provide informative insights for learning. This can help in discovering new strategies and improving sample efficiency.

2.8. Particle Filter (PF):

Particle filter [25, 26] is a technique used to represent the probability distribution of the state using a collection of particles. The algorithm is based on randomly generated particles; and at every time step, the particles are moved through the state model, and new particles are generated. The weights of the particles are updated, and the distribution of the state is estimated using a set of weighted samples.

The steps of the Particle Filter Algorithm include the following: [1]:

1. Randomly generate particles, each of which is a binary vector that represents the combination of feature maps.
2. Predict the next state of the particles: Move the particles to a new state based on reward feedback.
3. Update the weighting of the particles with normalization.
4. Get rid of very unlikely particles and put in more copies of the likely ones. Figure 4.1 demonstrates the process of re-sampling.

3. RELATED WORK

Reinforcement learning algorithms, such as DQN [5], have shown remarkable success in learning to play Atari 2600 games directly from raw pixel inputs. While DQN [5] can learn to play games effectively, it can suffer from instability and inefficiency in learning [9, 10].To address these shortcomings, several modifications have been proposed to the original DQN algorithm [9, 11, 13, 27, 28]. These modifications aim to improve the learning stability and efficiency of DQN by introducing various enhancements, such as prioritized experience replay, dueling network architectures, asynchronous updates, and distributional value estimation. In this section, we review and analyze these related works to provide a comprehensive understanding of the advancements made in addressing the instability and inefficiency issues of the DQN algorithm.

3.1. Prioritized Experience Replay (PER):

The Prioritized Experience Replay (PER) method, proposed by Schaul et al. [27], prioritizes experience replay based on the importance of the sample, so that it replays critical transitions more frequently, leading to more efficient learning. The PER method is an enhancement to the standard Experience Replay (ER) mechanism used in RL algorithms, particularly in the context of deep Q-networks (DQNs). ER is a technique commonly employed in RL, where the agent's experiences, consisting of observed states, actions, rewards, and next states, are stored in a replay buffer. During training, the agent selects a batch of experiences from this buffer to update its Q-network. This technique helps to eliminate the temporal correlation between consecutive experiences, reducing the bias introduced by sequential data. In PER, the replay buffer is augmented with a priority value associated with each experience. The priority reflects the importance of the experience in terms of its potential impact on learning. The higher the priority, the more likely an experience will be sampled during training. The priority of an experience is typically calculated based on the Temporal Difference (TD) error. This error indicates the disparity between the anticipated Q-value and the target Q-value. This prioritization mechanism enables the agent to concentrate on

learning from experiences that contribute to improving its performance. During the training process, experiences are selected from the replay buffer based on their priorities. However, to balance the exploration-exploitation trade-off, a stochastic element is introduced. Instead of always selecting the experiences with the highest priorities, experiences are sampled probabilistically, giving lower-priority experiences a chance to be selected as well.

The use of PER has been shown to enhance the learning efficiency and effectiveness of DQN [27]. By prioritizing experiences that lead to significant learning progress and focusing on important transitions, the agent can converge faster and achieve better performance.It's worth noting that there are different variations and implementations of prioritized experience replay, with different ways to calculate and update priorities. Overall, PER is a technique that improves the sample efficiency and learning effectiveness of RL algorithms by selectively prioritizing experiences with higher learning potential during the training process.

3.2. Dueling Neural Network Architecture (DNNA):

A Dueling Neural Network Architecture (DNNA), which is introduced by Wang et al. [13], separates the state value function and the action function, resulting in more stable learning and better performance than the state of the art on the Atari 2600 domain.

The DNNA is a variation of Deep Neural Networks (DNNs) commonly used in RL, particularly in the context of value-based methods such as Deep Q-Networks (DQNs). It aims to improve the efficiency and stability of learning by explicitly separating the estimation of the value and action functions of states. In value-based methods, such as Q-learning, the Q-function is employed to estimate the value of taking a particular action in a given state. The Q-function directly combines the state value and action information into a single output. However, this can make it challenging to accurately estimate the value and action separately, especially when they have different dynamic ranges. The dueling architecture addresses this challenge by decoupling the estimation of the value and action functions and estimates these functions independently. Value function focuses on estimating the state value, representing how advantageous it is to be in a particular state irrespective of the action taken. The other

action function estimates the advantages of different actions in a given state. The key idea behind the dueling architecture is that by estimating the value and action separately, the network can learn more efficiently and generalize better across different actions and states. Overall, the dueling neural network architecture is a technique in RL that improves learning efficiency and stability by decoupling the estimation of value and action functions. This architecture has found successful applications across various domains and has contributed to advancements in the DQN algorithm.

3.3. Distributed Deep Reinforcement Architecture (DDRA):

Several works have presented a distributed architecture for DRL [11, 28, 29]. Such architectures distribute the learning process across multiple parallel agents, which enables more efficient exploration of the state space and faster convergence [11]. As the complexity of tasks and the size of the neural networks increase, training on a single machine may become time-consuming. Distributed architectures address these challenges by distributing the workload across multiple machines or processors, allowing for efficient resource utilization. Multiple instances of the environment are created, each running on a separate machine or processor. These instances interact with the agent generating more diverse experiences and increasing the overall data throughput. Overall, DDRA shows the training acceleration and sample efficiency improvement on the DQN algorithm [11, 28].

These modifications collectively form the foundation for advanced algorithms which combines several of these enhancements to achieve performance enhancements over DQN [9].

3.4. Auxiliary Functions (AF):

Another group of work has aimed to learn additional Auxiliary Functions (AFs) with denser training rewards to improve the sample efficiency problem [30]. In RL, AFs are often used to provide additional learning signals to the agent during training. These signals are not the primary reward but are designed to help the agent learn more effectively. One common

approach is to introduce auxiliary tasks or auxiliary losses in combination with the DQN architecture [31]. Auxiliary tasks or auxiliary losses, are additional learning objectives incorporated into a RL model to improve its performance. These auxiliary functions are designed to provide additional supervision or regularization signals to guide the training of the main task. By providing additional supervisory signals, they guide the model to focus on specific aspects of the input data. While the primary reward signal can be sparse and delayed (only received at the end of the game), the auxiliary function provides a task-based auxiliary reward signal that can be more frequent (e.g., received after every time step). Auxiliary rewards help the DQN agent learn more efficiently, improve its exploration strategy, and potentially achieve better overall performance.

3.5. Attention Mechanisms (AM):

Several studies have used attentional mechanisms to improve the performance of their models [15, 16, 32–34]. Some of these use bottom-up attention, allowing the agent to selectively focus on different segments of the input, regardless of the agent's task. Others have applied attention to DRL problems in the Atari game domain [14, 15, 35]. Additionally, several others have explored attention by incorporating a saliency map [36] as an extra layer [37] that modifies the weights of CNN features.

Studies that use the basic bottom-up saliency map [36] as an attention mechanism in RL have used many hand-crafted features as inputs [38]. Yet, these models show an inability to attend to multiple input information with sufficient importance simultaneously. Top-down attention mechanisms can also be used to improve the performance of DQNs by allowing the agent to selectively attend to relevant parts of the input based on its current tasks [39].

Most of the previous DRL studies that use attention mechanisms are generally based on back-propagation learning [16, 34], which is actually not ideal to be used by DRL [32, 37] as it can lead to inflexibility [40]. Few other works have proposed to learn attention without back-propagation [32, 37]. Yuezhang et al. infer that attention from optical flow only applies to issues involving visual movement [37]. The Mousavi attention mechanism uses a

bottom-up approach, which is directed by the inherent characteristics of the input information regardless of the reward [32]. In the first part of this thesis, we incorporate a unified bottom-up and top-down visual attention mechanism into the DQN model to improve the game agent's training stability and efficiency. In the second part, we help the DQN agent learn faster to play the game by providing additional task-based feedback to the agent using auxiliary rewards.

4. PROPOSED METHOD

In this section, we present a novel proposed method that addresses the inefficiency issues of the DQN algorithm by incorporating attention mechanisms and auxiliary functions. The first part of our proposed method focuses on leveraging attention mechanisms to enhance the performance of DQN. Attention mechanisms allow the agent to selectively attend to relevant features or regions of the input, enabling more efficient exploration and utilization of the available samples. By incorporating attention into the DQN framework, our objective is to enhance the agent's capacity to concentrate on the most informative states and actions, thus mitigating the sample inefficiency problem.

In the second part of our proposed method, we introduce auxiliary functions to further enhance the inefficiency of DQN. Auxiliary functions provide additional learning objectives that guide the agent's to prioritize and promote the learning of more informative actions. By incorporating auxiliary functions into the DQN architecture, we aim to encourage the agent to learn more task related meaningful representations, leading to improved sample efficiency.

Through a series of experiments and comparisons with the standard DQN algorithm, we demonstrate the efficacy of our proposed method in addressing the inefficiency challenges. The integration of attention mechanisms and auxiliary functions enhances DQN sample efficiency, enabling more effective and rapid learning and finally improves the performance of DQN.

4.1. Attention-Augmented DQN Algorithm (AADQN)

Our proposed AADQN approach enhances the DQN architecture by incorporating bottom-up and top-down attention mechanisms. This integration empowers the game agent to selectively attend to task-relevant features while disregarding irrelevant features, thereby reducing the complexity of the task at hand.

As illustrated in Figure 1.1, the CNN part of the model takes the present state of the game as input I and extracts the set of feature maps $\boldsymbol{\mathcal{F}} = \{\boldsymbol{F_j} : 1 \leq j \leq D\}$. Using $\boldsymbol{\mathcal{F}}$, PBA defines the feature importance vector $\boldsymbol{V} = [V_1, V_2, \ldots, V_D]$. The TDA mechanism then generates particle vectors $\boldsymbol{\omega_i} : 1 \leq i \leq m$, each with D dimensions representing the D feature maps, and converts them to the attention vector $\boldsymbol{A} = [A_1, A_2, \ldots, A_D]$. Next, the BAR mechanism refines attention vector $(\boldsymbol{A})$ by considering the saliency map $\boldsymbol{H}$.

After calculating the CNN mid-layer relevance scores by LRP rules, AADQN freezes the irrelevant units on the mid-layers. Freezing the irrelevant neurons on the mid-layers improves the model's robustness and flexibility in complex environments.

The overall flow of AADQN is given in Algorithm 1. In the following, we break down this process and describe the AADQN modules in detail.

Algorithm 1: Attention-Augmented DQN (AADQN)

Initialize DQN network;

Initialize particles: $\boldsymbol{\omega_i} : 1 \leq i \leq m$;

while *stopping criterion not met* **do**

  get input frame: $I(x, y)$;

  extract CNN feature maps;

  calculate features' importance value: $V_j = \frac{1}{2}(\hat{F}_j + \hat{E}_j)/\chi_j^2$;

  calculate the prior probability of each particle:

  $p_j = \frac{V_j - \min(V_1, V_2, ..., V_D)}{\max((V_1, V_2, ..., V_D)) - \min(V_1, V_2, ..., V_D)}$;

  update all particle states based on prior probability;

  particle state normalization: $\hat{\omega}_i^j = \frac{\omega_i^j}{\sum_{j=1}^{D} \omega_i^j}$;

  get reward R_t from DQN;

  calculate state value of particles: $Q(s_t, \hat{\omega}_i)$;

  calculate likelihood probability of particles:

  $P(R_t|\omega_i) \propto \exp\left(-(\epsilon_i - \min\left(Q(s_t, \hat{\omega}_i)\right))\right)$;

  likelihood normalization: $p(\omega_i) = \frac{P(R_t|\omega_i)}{\sum_{i=1}^{m} P(R_t|\omega_i)}$;

  attention update: $\boldsymbol{A_j} = \frac{1}{m} \sum_{i=1}^{m} \omega_i^j / \sum_{j=1}^{D} \frac{1}{m} \sum_{i=1}^{m} \omega_i^j$;

  saliency map ($\boldsymbol{H}$) extracted from input and normalized in $[0, 1]$: ;

  calculate relevance score of neurons: $\sum_i \phi_l^{i \leftarrow j} = \phi_{l+1}^j$;

  extract features FDP information: $\mathbf{P_j} = I(x, y) \quad s.t. \quad \phi_j(x, y) > \bar{\phi}_j$;

  calculate feature saliency value: $H_j = \frac{1}{|P_j|} \sum_{x,y \in \mathbf{P_j}} H(x, y)$;

  extract the refinement vector: W_j;

  refine feature maps: $F^{'} = F \odot (W \odot A)^{\Uparrow}$;

  if $A_j > \theta$ **then**

    freeze irrelevant neurons corresponding to the j-th feature map;

  end

end

4.1.1. Preliminary Bottom-Up Attention (PBA)

PBA determines the feature importance vector $\boldsymbol{V}$ of the set of feature maps $\boldsymbol{\mathcal{F}}$, which is then utilized by the TDA mechanism to initialize the particles' states.

To generate feature importance $\boldsymbol{V}$, we first compute the normalized mean activation value $\hat{F}_j$ of each feature map $\boldsymbol{F_j}$ in $\boldsymbol{\mathcal{F}}$, where $1 \leq j \leq D$. The mean activation value of a feature map gives a measure of the activity within a CNN, s.t., it serves to assess the level of activity in a feature map.

To select the most informative features, we calculate the entropy of each feature map $\boldsymbol{F_j}$ using the Shannon entropy formula [41] given by

$$E_j = -\sum_{i=1}^{B} b_i \log_2 b_i \tag{4}$$

where B is the number of feature value ranges, and b_i is the probability of observing a feature value in the ith range [41].

The entropy E_j is then also normalized:

$$\hat{E}_j = \frac{E_j}{\sum_{i=1}^{D} E_i} \tag{5}$$

Features with uniform distribution are not desired in CNNs, as they do not provide the network with the ability to learn complex patterns and features in the input data. To this end, the chi-square (χ^2) Test [42] given by

$$\chi_j^2 = \sum_{i=1}^{K^2} \frac{(O_i - \mu_i)^2}{\mu_i} \tag{6}$$

is used to improve the feature selection, where the χ^2 metric is used to assess the uniformity of the activation values. Here, O_i represents the activation values of the feature maps, while

μ_i corresponds to the activation values that are uniformly distributed. A higher value of χ^2 indicates the lower importance of the feature in determining the output.

The evaluation of feature importance typically relies on activation values [43, 44] or, in certain cases, the entropy of feature maps [45, 46]. We use both metrics to achieve better performance across diverse environments and enhance the assessment of feature importance, as in

$$V_j = \frac{\frac{1}{2}(\hat{F}_j + \hat{E}_j)}{\chi_j^2} \tag{7}$$

V_j makes up the feature importance vector $\boldsymbol{V}$, which is used in TDA to initialize the particle states.

4.1.2. Top-Down Attention (TDA)

The TDA mechanism helps the agent focus on task-specific information by assigning weights to feature maps through the attention vector $\boldsymbol{A} = [A_1, A_2, \ldots, A_D]$, where A_j represents the attention weight of the feature map $\boldsymbol{F_j}$. This mechanism uses a particle filter that generates a set of particles $\boldsymbol{\Omega} = \{\boldsymbol{\omega_i} : 1 \leq i \leq m\}$ to estimate the distribution of the attention vector for the present task. Each particle is a D-dimensional binary vector $\boldsymbol{\omega}$ that estimates the probability distribution of the feature maps with respect to importance. In a particle vector $\boldsymbol{\omega_i}$, the jth element ω_i^j corresponds to the feature map $\boldsymbol{F_j}$ and indicates whether that feature is useful (‘1’) or not (‘0’) for the task at hand.

This binary representation provides a selection of essential features [1, 25, 26, 47].

The algorithm iteratively updates the particles using the reward feedback. Then, re-samples the particles (Figure 4.1) and generates a new sample set. Finally, the distribution of the feature maps is estimated.

Combining gradient descent and particle filter provides the following advantages:

- Tracing the local gradient within the particle filter makes it easier to find minimum points while using a reduced number of particles. [48].
- Improving the gradient descent method by introducing multiple hypotheses helps prevent getting stuck in local minimum points [49].

Similar to DQN [5], we apply $D = 64$ feature maps in its final CNN layer. To estimate the distribution of these feature maps, the number of particles required depends on the distribution complexity. Increasing the number of particles generally enhances the accuracy of distribution estimation. However, there is no universally optimal number of particles that applies to all scenarios. It is common to employ a sufficiently large number of particles to ensure reliable estimates [1, 47]. In our case, $m = 250$ particles are utilized to process the set of 64 feature maps.

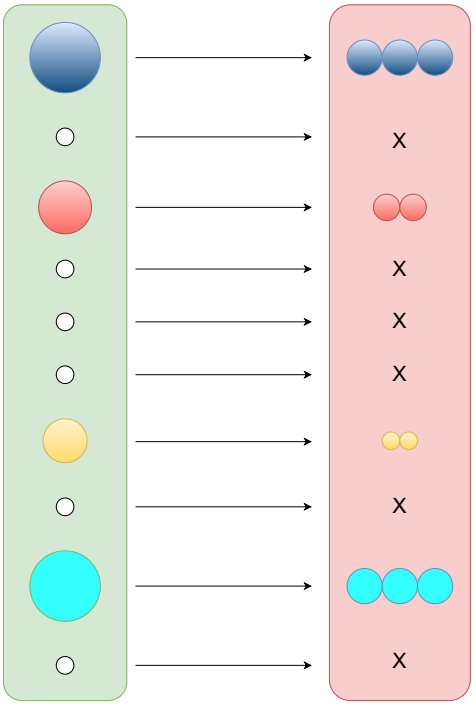


Figure 4.1 The size of each particle corresponds to its weight. In the re-sampling process, some particles are selected multiple times, while others are not chosen, as indicated by the 'x' symbol. Re-sampling removes particles with very low probabilities (white particles) and replaces them with particles that have higher probabilities. A distinct color is assigned to each particle and its corresponding resampled particles [1]

In order to generate the particle set $\boldsymbol{\Omega}$ that TDA uses to compute the attention vector $\boldsymbol{A}$, we first normalize the importance value V_j of features $\boldsymbol{F_j}$ across all feature maps and then convert to the probability. By implementing this approach, it guarantees that the most active feature will possess a certain probability equal to 1 as follows:

$$p(V_j) = \frac{V_j - \min(\boldsymbol{V})}{\max(\boldsymbol{V}) - \min(\boldsymbol{V})} \tag{8}$$

Then, the binary ω_i^j are generated randomly using a Bernoulli distribution, i.e., $p(\omega_i^j)$ has a probability of $p(V_j)$ for $\omega_i^j \leftarrow 1$ and a probability of $1 - p(V_j)$ for $\omega_i^j \leftarrow 0$. In this way, TDA initializes the particles by focusing on the most informative features with regard to the feature importance V_j, so that the features with higher V_j are more likely to be attended to than those with lower V_j.

To calculate the particles' R_t likelihood, i.e., the immediate reward that is output from DQN's given state, first, the initialized particle state ω_i is normalized as follows:

$$\hat{\omega}_i^j = \frac{\omega_i^j}{\sum_{j=1}^{D} \omega_i^j} \tag{9}$$

After updating the state of each particle as described above, we calculate the error between the predicted particle state Q-value $(Q(s_t, \hat{\omega}_i))$ and the R_t, which is returned from DQN as follows:

$$\epsilon_i = (R_t - Q(s_t, \hat{\omega}_i))^2 \tag{10}$$

where $\hat{\omega}_i$ is the normalized particle.

In the TDA process, rewards are determined based on normalized particles. A good particle state has a stronger predictive ability $Q(s_t, \hat{\omega}_i)$ for the target reward R_t. The main objective in this context is to identify the particle state that achieves the highest accuracy among a given set of particles, with the aim of minimizing the error in (Equation (10)). By minimizing the

squared error, the algorithm seeks to find the particle state that closely aligns with the desired target reward, which is used in the next time step.

Particles are updated based on the likelihood of the immediate reward R_t, which is proportional to the following error value, s.t.,

$$P(R_t|\omega_i) = exp(-(\epsilon_i - \min(\epsilon_1, \epsilon_2, ..., \epsilon_m))) \tag{11}$$

Once the likelihoods are calculated, $p(\omega_i)$ are found as follows:

$$p(\omega_i) = \frac{P(R_t|\omega_i)}{\sum_{i=1}^{m} P(R_t|\omega_i)} \tag{12}$$

After that using $p(\omega_i)$, the particles are randomly selected again with replacement, and the distribution is adjusted for the next step.

Finally, the attention vector **A** is reset with the normalized mean of the particle states as in

$$A_j = \frac{\bar{\omega}^j}{\sum_{j=1}^{D} \bar{\omega}^j} \tag{13}$$

where $\bar{\omega}^j$ is given by

$$\bar{\omega}^j = \frac{1}{m} \sum_{i=1}^{m} \omega_i^j \tag{14}$$

4.1.3. Bottom-Up Attention Refinement (BAR)

BAR enhances the focus by considering the saliency values of the essential features. It aims to improve agent performance by increasing the attention weights of the essential features ($A_j >= \theta$) (Equation (17)) with lower saliency values H_j. While bottom-up methods can be useful in identifying salient features, they are not solely effective in identifying task-related features since they can miss much crucial task-related information due to noise or complexity.

As a result, our approach improves the learning ability of the agent while considering this attention refinement, allowing for improved capture of task-related information.

Traditional saliency prediction methods rely on low-level features like color, contrast, and texture [50, 51], but they struggle to capture the full range of factors influencing visual saliency maps [52]. BAR utilizes the Saliency Attentive Model (SAM) by Cornia et al. [52], which uses a convolutional long short-term memory to enhance saliency predictions iteratively.

We quantify the relevance score of Feature Decomposed Pixels (FDPs) with respect to specific features within the CNN using LRP [3]. These relevance scores are used to select the specific feature map-related pixels. These pixels are determined by whose relevance scores are greater than the average value of the corresponding pixels' relevance scores.

BAR obtains the saliency value $H(x, y)$ corresponding to a specific pixel (x, y) from the saliency map ($\boldsymbol{H}$) that is generated by the SAM model [52].

To calculate the specific feature saliency value H_j according to the pixel information using the saliency map ($\boldsymbol{H}$), we first normalize the saliency map within the range of $[0, 1]$. Then, BAR calculates the saliency value of a specific feature map denoted as j by considering only pixels with average relevance scores higher than the average relevance of the corresponding feature map.

$$\mathbf{P_j} = I(x, y) \quad s.t. \quad \phi_j(x, y) > \bar{\phi}_j \tag{15}$$

Here, $\mathbf{P}$ corresponds to the pixels that have a higher relevance score than the average relevance value ($\bar{\phi}$) of that specific feature map:

$$H_j = \frac{1}{|P_j|} \sum_{x,y \in \mathbf{P_j}} H(x, y) \tag{16}$$

The refinement vector $\boldsymbol{W}$ is obtained by considering the feature saliency value H_j using

$$W_j = \begin{cases} 1 + e^{-\alpha H_j} & \text{if } A_j >= \theta \\ 1 & \text{if } A_j < \theta \end{cases} \tag{17}$$

where θ threshold is defined as the average value of the attention vector $\boldsymbol{A}$.

The attention vector $\boldsymbol{A}$ is refined by multiplying element-wise with the refinement vector $\boldsymbol{W}$ as $(W \odot A)$. Then, each refined element is replicated to align with the dimensions of a feature map, ensuring that the same attentional value is applied uniformly across all spatial locations in the feature map $(W \odot A)^{\Uparrow}$. Finally, this process re-weights the feature maps, amplifying feature maps with attentional weights as (Equation (18)):

$$\boldsymbol{F}' = \boldsymbol{F} \odot (\boldsymbol{W} \odot \boldsymbol{A})^{\Uparrow} \tag{18}$$

Here $\odot$ corresponds to the Hadamard product, and $\Uparrow$ shows the upscaling the refined attention vector by replication

4.1.4. Layer-Wise Relevance Propagation (LRP) -Based Transfer Learning:

AADQN employs a transfer learning scheme to enhance the agent's flexibility by reducing features and improving adaptation to noise and complexity. This scheme is based on LRP by Saraee et al. [3].

LRP propagates the output of the network backward through its CNN layers, assigning relevance scores to each neuron in each layer based on its contribution to the output, as illustrated in Figure 4.2, according to the LRP rule as in [2], which satisfies

$$\sum_i \phi_l^{i \leftarrow j} = \phi_{l+1}^j \tag{19}$$

where ϕ_l^i is the relevance of neuron i at layer l, and ϕ_{l+1}^j is the relevance of neuron j at the next layer $l+1$. According to the LRP method, the relevance value of the feature map is redistributed in the lower layers, and $\phi_l^{i \leftarrow j}$ is defined as the share of ϕ_{l+1}^j to neuron i in the lower layer l. Back-propagation continues until relevance scores are extracted for all neurons, including the input layer.

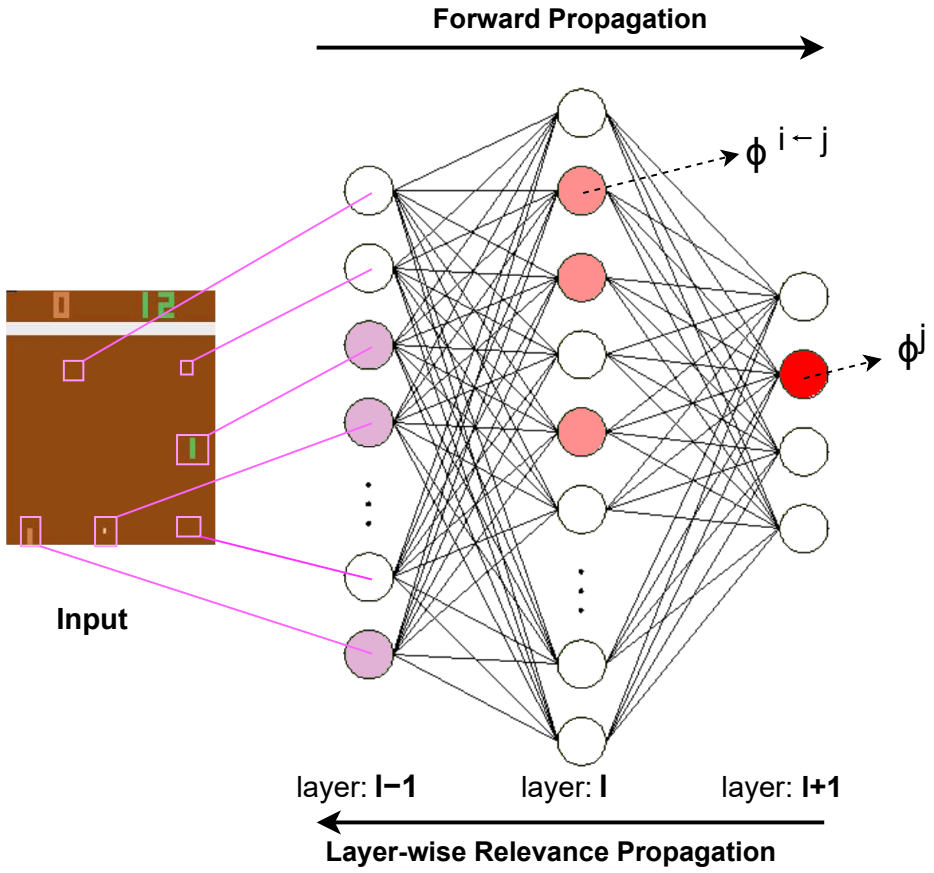


Figure 4.2 The backward process of the LRP in the propagation of features' importance relevance value across the middle layers. ϕ^j is the relevance value of neuron (j) at the layer $(l+1)$. According to the LRP method, the relevance value of the feature map is redistributed in the lower layers, and $\phi^{i \leftarrow j}$ is defined as the share of ϕ^j to neuron i in the lower layer l [2, 3].

As described in Section 4.1.3. above, the relevance scores for the input layer are then used in BAR to highlight the FDP information in the input data that leads to the output of the feature maps.

When computing the final layer feature maps of the CNN, multiple internal CNN layers are employed. The vast number of parameters within the mid-layers of the network enables the capture of intermediate features at various levels of abstraction [53]. By harnessing these intermediate features, especially through dual attention mechanisms, the models' generalization capabilities can be enhanced [18]. To avoid unrecoverable information loss [54], AADQN freezes the unimportant neurons of the inner layers [54, 55], which improves the model computing efficiency. Moreover, the model becomes most robust to noise and can improve flexibility in complex environments.

4.2. Auxiliary Distance Function (ADF):

In DQN, the agent typically receives observations, in the form of images or frames, as input. Our ADF aims to encourage the DQN agent to focus on changes occurring in consecutive frames and learn to exploit these changes for decision-making. The key idea behind the ADF is to compute the pixel-wise difference between consecutive frames and use it to calculate a separate learning signal to mitigate the main reward sparsity. By incorporating this additional signal, the DQN agent can learn to recognize and exploit changes in the environment, which can be valuable for tasks that require tracking moving objects, detecting dynamics, or understanding temporal patterns.

The ADF finds the frame difference of two consecutive frames by subtracting the pixel values of one frame from the other. This subtraction operation results in an image that highlights regions where changes have occurred. Then, update the auxiliary reward based on Distance Mean Absolute Error (DMAE) that finds the absolute distances of the agent with changes area by comparing the previous frame distance value. ADF updates the auxiliary distance reward if the current distance is less than the agent's previous distance value. In such cases, the auxiliary distance reward is increment by a constant value of 0.1, and in the vice versa case, it is reduced by the same value. The relative importance or weighting of the auxiliary reward compared to the primary task is a crucial consideration. To combine the auxiliary distance reward with the DQN main reward to provide contributions during

training, the auxiliary distance reward added to the main task reward corresponds to the DQN agent action. By jointly optimizing the main task reward by auxiliary distance reward, the agent is encouraged to pay attention to changes in the environment and learn to exploit them effectively.

5. EXPERIMENTAL RESULTS

In our work, we focus on DRL game agent's inefficiency and inflexibility problems. In this context, we investigate the importance of applying a visual attention mechanism to the performance of the game agent. To assess the performance of our model, we address the following questions:

- How does AADQN's game-play average score compare with DQN?
- How is AADQN's learning stability in comparison to DQN?

To compare the two algorithms, we have implemented our approach on the Atari 2600 environment in the OpenAI Gym environment [4]. From Atari 2600, we selected eight games (Pong, Wizard of Wor, SpaceInvaders, Breakout, Asterix, Seaquest, Beam Rider, Qbert) (Figure 5.1) with varying complexities and difficulty levels.

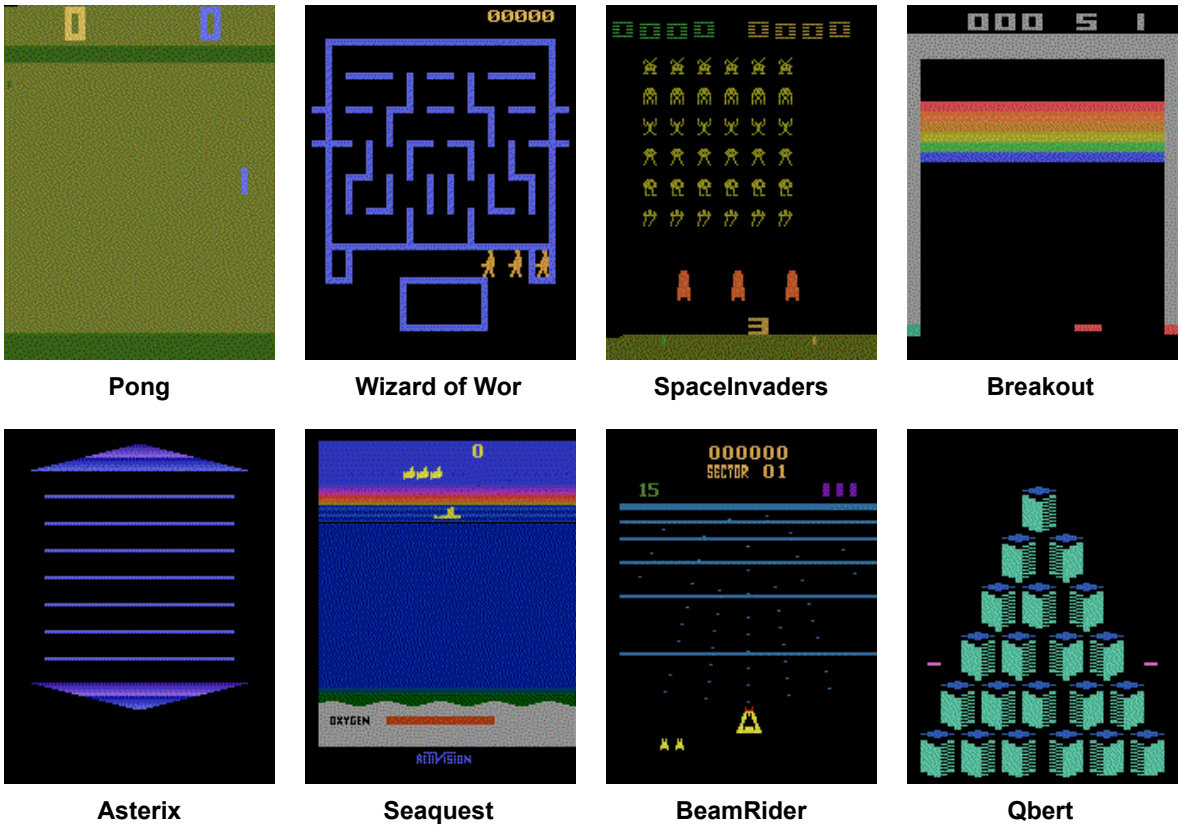


Figure 5.1 Pong, Wizard of Wor, SpaceInvaders, Breakout, Asterix, Seaquest, Beam Rider, Qbert.

It's challenging to precisely categorize the complexity of different OpenAI Gym Arcade Learning Environments, as complexity can be subjective and depends on various factors such as game dynamics and objectives. We try to categorize these environments based on basic features. Certainly, the complexity of solving these games using DRL can be influenced by the dynamics of the game and the state space complexity. The following table 5.1 provides a general attempt to categorize these environments based on their complexity for solving with DQN. We consider factors such as the complexity of the state space and the difficulty of learning effective strategic decision-making policies through DRL methods like DQN:

	Game Type	Game Dynamics	Objective	Environment Complexity	Visual Style
PONG	table tennis sports game where players control paddles to hit a ball back and forth	paddle and ball with two-player interaction	Score points by hitting the ball past the opponent's paddle	simple with two paddles and a ball	Simple 2D graphics with paddles and a ball
Breakout	brick-breaking game where the player controls a paddle to bounce a ball, breaking bricks at the top of the screen.	Paddle and ball dynamics with a focus on breaking bricks.	Break as many bricks as possible, aiming for a high score	relatively simple complexity with brick patterns and ball physics	2D graphics with bricks, paddles, and a ball.
Spaceinvaders	shooting game where the player controls a spaceship to defend against descending alien invaders.	Shooting aliens descending from the top of the screen.	Defend against and defeat descending alien invaders	Moderate complexity with descending alien formations	2D graphics with spaceships and aliens
Q-bert	puzzle game where the player controls a character that hops between cubes, changing their colors.	involving hopping on cubes to change their colors	Change the colors of cubes while avoiding enemies	Moderate complexity with cube-hopping and enemy avoidance	2D isometric graphics with cubes and characters
Asterix	Guide character Asterix to avoid lyres while collecting other objects	horizontal movement and vertical movement provide. lyres and other useful objects	Score points by collecting useful objects Collect as many useful objects as possible	Moderate complexity introducing challenges in navigation and decision-making	2D graphics with involves guiding the character to collect useful objects while avoiding lyres
Wizardofwor	maze-based shooting game where players navigate through a dungeon, shooting monsters	Maze navigation and shooting monsters	Navigate the maze, shoot monsters, and achieve a high score	Moderate complexity with maze navigation and shooting	2D graphics with maze walls, monsters, and shooting
Seaquest	involves underwater exploration where the player controls a submarine, rescues divers, and avoids or attacks sea creatures.	Underwater navigation, diver rescue, and enemy avoidance dynamics	Rescue divers, avoid enemies, and achieve a high score	Moderate complexity with underwater navigation and enemy interactions.	2D underwater graphics with submarines and sea creatures
Beamrider	space-themed shooting game where the player controls a spaceship moving forward on a grid.	Space-themed shooting and dodging dynamics	Shoot enemies and navigate through space	Moderate to high complexity with space-themed shooting and dodging	2D space-themed graphics with spaceships and enemies

Table 5.1 Experiment environments basic features

These features provide a starting point for categorizing and understanding the characteristics of each game in the OpenAI Gym Arcade Learning Environments. Each task is explained in the appendix section with more details. All experiments were run on a workstation with an Intel i7-8700 CPU and an Nvidia GTX-1080Ti GPU. The source code for the AADQN algorithm is available at the paper's GitHub page, `https://github.com/celikcan-cglab/AADQN`, accessed on 20 October 2023.

5.1. Comparison with Baseline DQN

In this section, we offer a comparative analysis with respect to DQN in the context of the game average score and time step. The time-step analysis includes the number of steps needed to attain a specific average score performance threshold. Table 5.2 provides the comparison of training the AADQN and DQN benchmark agents in average scores achieved on the eight tasks at the same final time spent for each task, and Figure 5.2 illustrates the progress of the two agents in terms of achieved scores until the final time step. In this plot, the y-axis reflects the average score, and the x-axis corresponds to the time steps. Given the extensive training process with over a million time steps for each environment, attempting to show every individual time step on the plot would be impractical. To address this challenge, the plot is generated by dividing the entire training duration into 10 equal ranges. Within each range, an average of $50k$ samples is taken for eight games, providing a simplified yet meaningful representation of performance trends.

In two relatively simpler games, Pong and Breakout, we have observed an increasing average score for both algorithms with increasing time steps. Furthermore, AADQN outperformed the DQN algorithm throughout the entire process. Until 25×10^5 time steps, both algorithms performed similarly, but after this point, AADQN started to outperform DQN. This is likely due to the fact that in simple environments, learning is easier for both networks, and the game's high-level task is easier to learn due to the simplicity of these game environments.

Game	DQN Avg. Score	AADQN Avg. Score	Improvement by AADQN (%)	Time Step
Pong	18.88	20.64	9.32%	$1 \times 10e7$
Wizard of Wor	791.51	5004.19	532.23%	$2 \times 10e8$
SpaceInvaders	1299.33	2999.47	130.84%	$3 \times 10e7$
Breakout	192.13	299.85	56.06%	$2 \times 10e6$
Asterix	5021.27	11999.97	138.98%	$5 \times 10e7$
Seaquest	6001.04	15000.1	149.95%	$3 \times 10e7$
Beamrider	7503.97	12999.49	73.23%	$3 \times 10e8$
Qbert	3997.46	9999.47	150.14%	$5 \times 10e7$
average	3103.19	7290.39	134.93%	$8 \times 10e7$

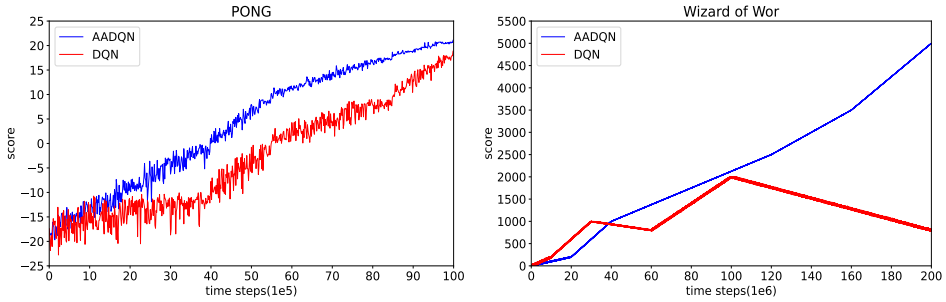
Table 5.2 Average score and improvement comparison

In the other more complex games, we have observed a better game score performance of AADQN after a certain point of game experience. The difference between AADQN and DQN grows with increasing time steps of learning. For example, in the Wizard of Wor game environment, DQN shows a better performance than our approach until 30×10^6 time steps. This is because particles in the AADQN algorithm in the initial steps were not yet different from random states. However, after (30×10^6) steps, AADQN has a better performance than the DQN method throughout the entire experiment. This clearly shows that when the particles are close to the desired state, which represents a target configuration of the features map, AADQN reaches the optimal policy earlier than DQN. We have observed similar behavior in other complex game environments (SpaceInvaders, Seaquest, Beamrider, and Qbert).

The decrease in score performance of the baseline DQN method during learning is also noteworthy in these complex environments, whereas the AADQN score has an increasing

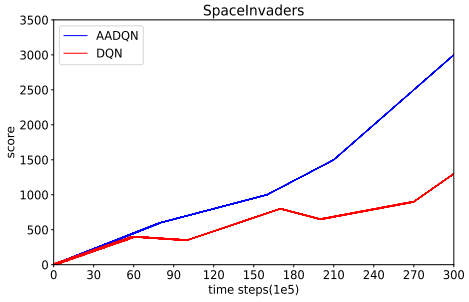
trend throughout the experiment. For example, in the Wizard of Wor game, at different learning steps (e.g., 30 and 100×10^6), DQN's score actually decreased. These complex environments, including Wizard of Wor, can be exhibit non-stationary behavior, implying that the most effective strategy could vary as time progresses. DQN assumes a stationary environment, and if this assumption is violated, the algorithm may struggle to adapt, leading to a decrease in performance. Moreover, DQN faces a challenge in exploration-exploitation trade-offs. In some stages of training, the agent may prioritize exploration and try different actions to learn about the environment. This exploration can lead to a decrease in the average score. If the exploration strategy is not well-tuned, the agent may not explore enough to discover optimal policies.

Overall, these findings provide empirical evidence supporting the claim of faster learning in AADQN with respect to DQN.

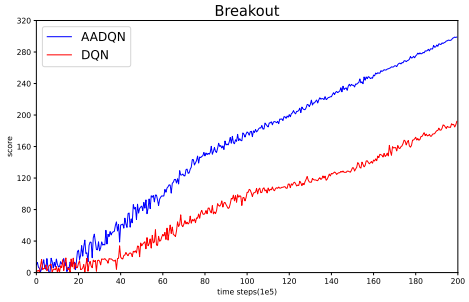


(a) Image 1

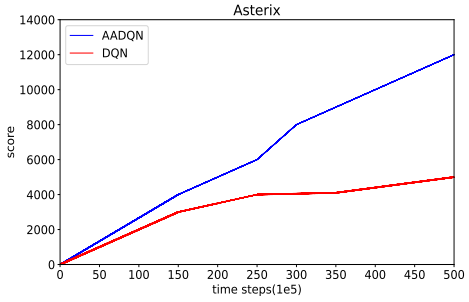
(b) Image 2



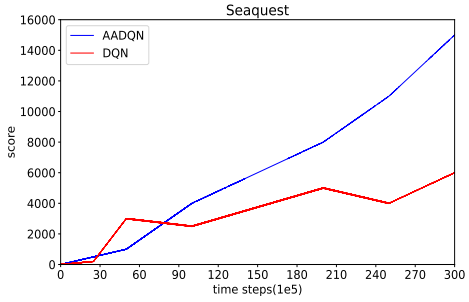
(c) Image 3



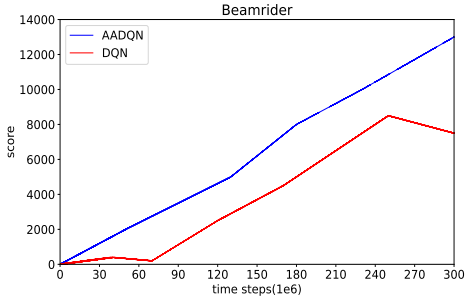
(d) Image 4



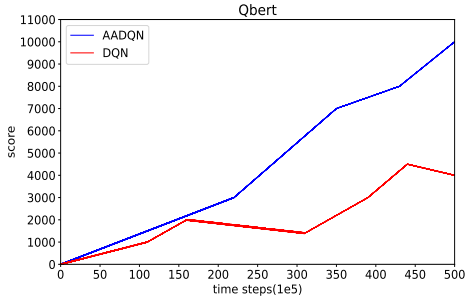
(e) Image 5



(f) Image 6



(g) Image 7



(h) Image 8

Figure 5.2 AADQN and DQN data efficiency comparison on eight Atari games. The x-axis shows the total number of training time steps. The y-axis shows the average score.

5.2. Comparison with other algorithms

In this section, we extend the performance comparison of AADQN to other state-of-the-art algorithms in addition to DQN. The results are given in Table 5.3, where we provide the final state of each analyzed algorithm for the eight games under consideration at the same specific time step to ensure an equitable comparison.

We extend and improve the performance comparison of AADQN by including other state-of-the-art RL algorithms, in addition to DQN (Deep Q-Network). We expand our analysis by incorporating a value-based Dueling DQN algorithm and a policy-based A3C (Asynchronous Advantage Actor-Critic) algorithm. By comparing the performance of these algorithms, we gain a more comprehensive understanding strengths and weaknesses of AADQN in game playing scenarios. To ensure a fair and equitable comparison, we present the results by providing the final state of each analyzed algorithm for the eight games considered at the same specific time step. By including the value-based Dueling DQN algorithm, we explore the potential benefits of its architecture, which separates estimating the values of states and the advantages of actions, it possible to achieve more precise approximation values. This comparison helps us evaluate whether the AADQN algorithm, which incorporates attention mechanisms and action conditioning, outperforms or is comparable to the most advanced or cutting-edge value-based DQN approach. Additionally, by incorporating the policy-based A3C algorithm, we investigate the effectiveness of policy gradient methods in game playing. This algorithm utilizes asynchronous training and parallelism to improve exploration and exploit the advantages of policy optimization. Comparing AADQN with A3C provides insights into the performance differences between these two distinct RL paradigms and their suitability for various game scenarios. By expanding the performance comparison to include these additional algorithms, we aim to provide a more comprehensive evaluation of AADQN's capabilities and its relative performance against other state-of-the-art RL approaches. The outcomes derived from this analysis will contribute to our understanding of the strengths, limitations, and potential

applications of AADQN and guide future research in game playing and RL algorithm development.

The results reveal that there are specific environments where AADQN falls slightly behind the other most advanced or state-of-the-art methods or techniques. For instance, the asynchronous actor-learner architecture (A3C) of A3C [28], which utilizes parallel agents, achieves a higher average score in the Wizard of Wor game. A3C is recognized for its ability to efficiently explore diverse actions and policies in intricate environments, making it valuable in the "Wizard of Wor" game. Similarly, the dueling network architecture (Dueling DQN) by Wang et al. [13], which separates value and advantage function estimation for Q-value computation, exhibits better performance in the Qbert game. This separation allows Dueling DQN to excel in situations where distinguishing between the value and advantage of different actions is critical, such as in the Qbert game. Nonetheless, in other games, AADQN outperforms these methods. Overall, we can conclude that AADQN exhibits relatively better performance in complex game environments.

Game	Random Play [5] Avg. Score	A3C [28] Avg. Score	Dueling DQN [13] Avg. Score	DQN Avg. Score	AADQN Avg. Score	Improvement by AADQN (%)	Time Step
Pong	-20.7	-15	14.5	18.88	**20.64**	9.32%	1×10^7
Wizardofwor	563.5	**6000**	5666	791.51	5004.19	−16.59%	2×10^8
SpaceInvaders	148	480	864	1299.33	**2999.47**	130.84%	3×10^7
Breakout	1.7	30.0	164.5	192.13	**299.85**	56.06%	2×10^7
Asterix	210	2224	2857	5021.27	**11999.97**	138.98%	5×10^7
Seaquest	68.4	850	9000	6001.04	**15000.1**	66.66%	3×10^7
BeamRiders	363.9	7600	**15112**	7503.97	12999.49	−13.97%	3×10^8
Qbert	163.9	8668	**15714**	3997.46	9999.47	−36.36%	5×10^7
average	187	3,230	6,174	3,103	7,290	**41.86%**	3×10^7

Table 5.3 Average scores comparison across eight games.

5.3. General Comparison with Fairness-Aware Constraints

In this section, we're taking a comprehensive comparison with other studies in the literature, to understand how our AADQN model's performance aligns with that of relevant studies. We're comparing normalized results, checking out how our scores compare to those in similar studies.

It's important to mention that we're aware of the time step constraints in our comparison. In this section the other studies that we're looking at didn't report info about the time steps when they hit their highest scores. And we had to stop our model before reaching the maximum score due to hardware constraints. So we have a limitation in our study which it had to be halted before reaching the maximum and we're being upfront about this aspect. Hence, to

provide relatively fair comparison, we're specifically focusing on normalized results. Our goal is to give a fair assessment of our AADQN model, comparing it to a bunch of other studies. We want to be honest about a limitation: we had to stop training our model before it could reach the maximum score for each task because of some hardware limits. Even with this challenge, we're excited to share insights into how well our model is doing and contribute to the ongoing discussions in the research community.

To obtain summary of the results across games, we normalize the score for each game as follows:

$$NormalizedScore = \frac{MaximumRawScore - RandomPlayScore}{|HumanPlayScore - RandomPlayScore|} \times 100 \qquad (20)$$

In our comprehensive performance comparison of agents, we employed the human-normalized score as a benchmark. The human and random scores align with those defined by Wang, providing a baseline for our evaluation. The agent's score determined by considering its maximum raw score. To assess the overall performance of the agent, we employed both the mean and median calculations of the normalized score across the eight games. This approach capturing both the average and central tendency of the agent's normalized performance and ensuring a robust assessment across the eight diverse gaming scenarios. Overall, this normalization process involves accounting for variations in the scoring systems of individual games, ensuring a fair and standardized representation of performance. By applying this normalization technique, we aim to derive comprehensive insights that facilitate a more understanding of our model's performance across gaming scenarios. We provide a comprehensive overview of results reported in various studies in table 5.4, drawing upon relevant references.

	PONG	Breakout	Spaceinvaders	Q-bert	Asterix	Wizardofwor	Seaquest	Beamrider
Random [29]	-20.7	1.7	148	163.9	210	563.5	68.4	363.9
Human [29]	14.6	30.5	1,668.7	13455	8503	4757	42,054.7	16,926.5
DQN [5]	20.2	428.1	2,869	13,890	7756	5,412	6,596	8,465
AADQN	20.64	299.85	2999.47	9999.47	11,999.97	5004.19	15000.1	12,999.49
DDQN [29]	20.9	418.5	2,525.5	15,088.5	17,356.5	7,492.0	16,452.7	13,772.8
Tuned-DDQN [27]	19.1	371.6	9063.0	11277.0	31907.5	7451.0	39096.7	31181.3
Prior-DDQN [9]	20.7	381.5	7,696.9	18,802.8	41,268.0	10,373.0	44,417.4	22,430.7
DUEL-DQN [9]	**21.0**	345.3	6,427.3	19,220.3	28,188.0	98,209.5	50,254.2	12,164.0
Distrib-DQN [9]	18.9	548.7	6,368.6	15,035.9	395,599.5	11,824.5	3,275.4	15,002.4
Rainbow [9]	19.0	379.5	12,629.0	18,397.6	280,114.0	14,631.5	19,176.0	21,768.5
PRIOR. DUEL [29]	20.9	366.0	15,311.5	18,760.3	375,080.0	12,352.0	931.6	**30,276.5**
PRIOR. [28]	18.9	371.6	9063.0	11277.0	31907.5	7451.0	39096.7	26172.7
A3C [28]	11.4	**766.8**	**23846.0**	21307.5	22140.5	18082.0	2355.4	24622.2
C51 [29]	20.9	748	5,747	**23,784**	**406,211**	**949,604**	**266,434**	14,074
Gorila [11]	18.30	402.20	1883.41	7089.83	6433.33	13731.33	13169.06	3822.07
NoisyNet-A3C [56]	**21**	401	1,280	19,418	35,045	16,143	984	12,819
NoisyNet-Dueling [56]	**21**	283	7,227	27,543	28,957	9,790	23,373	19,163

Table 5.4 Maximum Raw scores across eight games for each algorithm that obtained the highest score during training. We report the published scores for AADQN, and the results for other studies reported from given references.

	PONG (%)	Breakout (%)	Spaceinvaders (%)	Q-bert (%)	Asterix (%)	Wizardofwor (%)	Seaquest (%)	Beamrider (%)
DQN	115	1480	178	103	90	115	15	48
AADQN	117	1035	187	74	142	105	35	76
DDQN	117	1447	156	112	206	165	39	80
Tuned-DDQN	112	1284	586	83	382	164	92	**186**
Prior- DDQN	117	1354	496	140	495	233	105	133
DUEL- DQN	**118**	1226	413	143	337	2328	**119**	71
Distrib-DQN	111	1896	409	111	4767	268	7	88
Rainbow	111	1310	821	137	3375	335	45	129
PRIOR- DUEL	117	1264	997	139	4520	281	2	180
PRIOR.	111	1295	586	83	382	164	92	155
A3C	90	**2653**	**1559**	159	264	417	5	146
C51	117	2591	368	177	**4895**	**22633**	63	82
Gorila	109	1389	114	52	75	314	31	20
NoisyNet A3C	117	1386	74	144	420	371	2	75
NoisyNet-Dueling	117	976	465	**206**	346	220	55	113

Table 5.5 Normalized score for eight games.

Through the normalization of results presented in the table 5.5, we establish a standardized basis for evaluating the results. To facilitate a comprehensive assessment, we augment the presentation by including both the mean and median normalized scores across a diverse set of eight games in the table 5.6. This enhances the robustness of our evaluation process and provides a clearer understanding of the performance of algorithms across varied eight gaming scenarios

The evaluation of overall agent performance is presented through the mean and median of the normalized scores across eight games. The comparison between algorithms is outlined in the following table 5.6, featuring both median and mean scores. The percentage improvement, relative to the DQN baseline in terms of median score, is reported in the last column. Notably, all agents demonstrate an enhancement in both mean and median normalized scores. In each game, the score is determined by selecting the maximum performance, and AADQN consistently outperforms the DQN baseline across all eight games. This superiority is evident in both the normalized results in the table 5.5 and the mean/median scores in the table 5.6. The improved performance of AADQN is also noticeable when compared to its corresponding DQN baseline in all eight environments, as well as in the case of Gorilla. Throughout these comparisons, it's important to acknowledge that all AADQN scores are values attained before reaching the maximum score for each game. This limitation arose due to the necessity of early termination of AADQN runs owing to hardware constraints. We

	Mean (%)	Median (%)	Improvement On Median (%)
DQN	268	109	-
AADQN	221	129	18
DDQN	290	136	24
Tuned-DDQN	361	175	60
Prior- DDQN	384	186	70
DUEL- DQN	594	240	120
Distrib-DQN	957	189	73
Rainbow	782	236	116
PRIOR- DUEL	937	230	111
PRIOR.	358	159	45
A3C	661	211	93
C51	**3865**	**272**	**149**
Gorila	263	92	-15
NoisyNet A3C	323	130	19
NoisyNet-Dueling	312	213	95

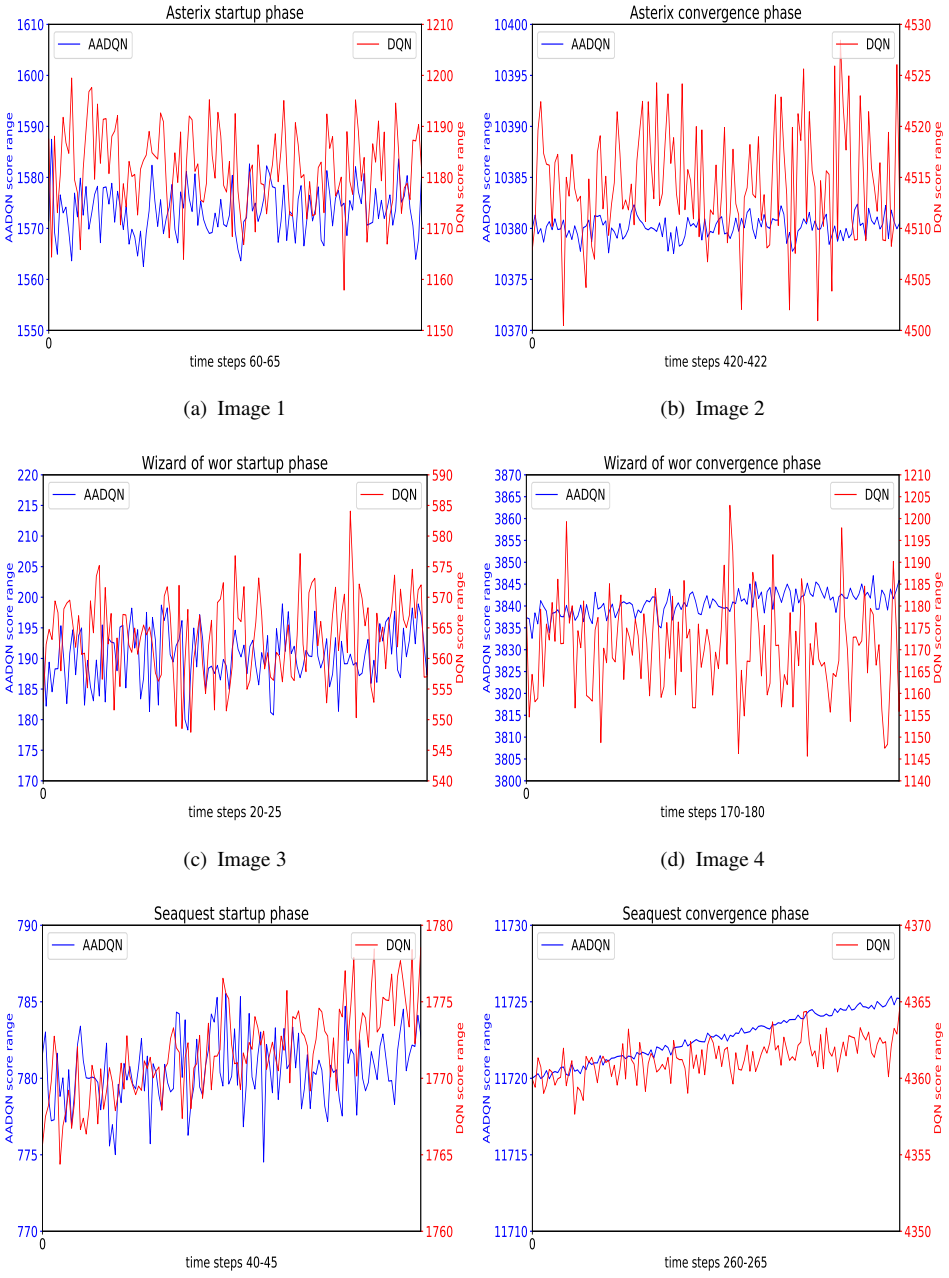
Table 5.6 Mean and median normalized scores.

believe that if we could have trained AADQN for the same number of timesteps required to reach the maximum score, our results would have shown further improvement. The assumption is that additional training time could have enhanced the performance of our algorithm, leading to even better outcomes Furthermore, it's worth noting that information regarding the time step at which the maximum score was achieved in other studies is not reported.

5.4. Comparison of Learning Stability

Figures 5.3 and 5.4 demonstrates the fluctuation rate in the average game scores of the agent during the start-up and convergence phases for the eight games under consideration. For the sake of a clear comparison between DQN and AADQN, the two plots are overlaid in each sub-figure so that they are shown with the same scale but different ranges (e.g., in the Asterix game, the AADQN scores are in the range [1550–1610], whereas DQN is the range [1150–1210]).

It is seen that the AADQN and DQN methods have different start-up and convergence phase fluctuation behaviors. Significant fluctuations in the start-up phase for both algorithms indicate instability or inconsistent learning. In the convergence phase, AADQN's average score stabilizes and mitigates the fluctuation compared to the DQN algorithm. This point indicates that the agent has converged to a relatively optimal policy and is consistently performing well. As shown in Figure 5.2, in the Pong game, AADQN becomes stabilized sooner than the baseline DQN. Similar stability behavior in the start-up and convergence phases are observed in all games, regardless of game complexity.



(a) Image 1 (b) Image 2

(c) Image 3 (d) Image 4

(e) Image 5 (f) Image 6

Figure 5.3 Comparison of the start-up (left) and convergence (right) phases between AADQN and DQN.

(a) Image 7 (b) Image 8

(c) Image 9 (d) Image 10

(e) Image 11 (f) Image 12

Figure 5.4 Comparison of the start-up (left) and convergence (right) phases between AADQN and DQN.

5.5. Algorithmic Efficiency

The time required for training DQN can vary significantly depending on the hardware. So, in this section, we explain the computational efficiency of AADQN in terms of total time steps to ensure a fair and equitable comparison with the baseline. Our framework is based on DQN together with particle-filter-based top-down and saliency-based bottom-up attention. So, the computation overhead on top of DQN comes from these attention mechanisms. The particles can be sampled in parallel, which adds a single particle computation to the system. However, the attention mechanism decreases the total time steps required to obtain the same average score with a focus on significant feature maps. Similarly, the saliency maps can be extracted parallel to the DQN forward pass. LRP relevance scores can be calculated parallel to the DQN backward pass, which requires no additional time steps. In Figure 5.2, the time-step comparisons of DQN and AADQN are given.

5.6. Ablation Study

In order to delve deeper into the intricate workings of our proposed model and elucidate the pivotal components contributing to its performance, we conduct an ablation study. This examination involves the selective removal of specific components from our model architecture, allowing us to discern their impact on the overall efficacy of the system. By systematically removing and evaluating each constituent part, the ablation study provides a comprehensive understanding of the model's robustness and sheds light on the key factors driving its success. The insights gained from this analysis not only contribute to the refinement of our current model but also offer valuable guidance for the design and optimization of future iterations. In the subsequent sections, we present the findings of this ablation study, dissecting the contributions of various components and elucidating their influence on the model's overall performance.

5.6.1. Attention Mechanism Ablation

Within the intricate architecture of our AADQN model, a synergy of innovation is achieved by combining the foundational framework of the baseline DQN with a novel attention component. This attention component, a fusion of bottom-up and top-down attention mechanisms, plays a pivotal role in enhancing the model's adaptability and performance. Here, we analyze the contribution of this attention-based component to our AADQN model performance by selectively removing its constituent parts. Through this examination, we aim to unravel the specific impact of the attention mechanism components on the AADQN model's learning capabilities and overall efficacy. This exploration not only deepens our understanding of the intricate interplay between attention mechanisms and traditional DQN components but also sheds light on the necessity and significance of these components in the pursuit of optimal performance. The following sections present the outcomes of our ablation study on the important components, providing invaluable insights for the refinement of our AADQN model.

5.6.2. Bottom-up and top-down Components

In our AADQN model, we have taken an approach to integrate both bottom-up and top-down attention mechanisms, with the objective of capturing task-specific information that may elude the bottom-up attention component during training. Previous studies have shed light on the limitations of bottom-up saliency-based methods, emphasizing their tendency to overlook critical task-related details. To address this concern, we have designed the top-down attention component in our AADQN model to play a central role in focusing on task-specific details that may have been disregarded or received insufficient attention by the bottom-up component.

The top-down attention component assigns attention weights to task-specific details. Our hypothesis posits that the attention mechanism aims to uncover task-based information that could be overlooked by the bottom-up attention but holds significance when attended by the

top-down attention component. The overarching goal of this attention mechanism is to refine and optimize its performance by adjusting the attention weights assigned to these crucial details.

To summarize, if the top-down attention component assigns attention weights higher than the mean attention value to any task-specific details that were disregarded by the bottom-up module, the attention mechanism endeavors to highlight these details through the refinement of attention weights.

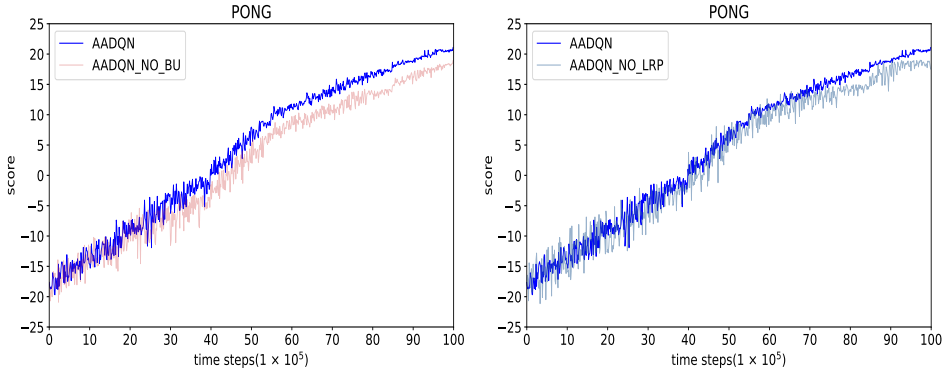
Through the intricate interplay of bottom-up and top-down attention, our AADQN model strives to enhance its ability to prioritize task-relevant information, resulting in an overall improvement in the performance of the attention mechanism. This integration of bottom-up and top-down attention mechanisms in our AADQN model provides a robust framework for capturing task-specific details that may be missed by bottom-up saliency-based methods, thereby contributing to a more effective and comprehensive attention mechanism.

In this section of the thesis, our focus was on conducting an ablation study analysis of the bottom-up and top-down components. The purpose of this analysis was to gain an understanding of the individual contributions and effectiveness of these components within the overall AADQN framework.

By removing or disabling each attention component independently, we aimed to assess their impact on the performance and functionality of the model. This approach allowed us to evaluate the specific roles played by the bottom-up and top-down components in capturing and attending to relevant information. When we remove the top-down attention, it's crucial to also remove the bottom-up component because the bottom-up module refines the top-down results. Without the top-down module, we can't make use of the bottom-up component. As a result, if we remove both the top-down and bottom-up components, our model essentially goes back to the DQN baseline. This means we don't have any ablation analysis for removing top-down attention.

5.6.3. Ablation Studies Experiments

As shown in Figure 5.5, we specifically selected the Pong game as the evaluation environment for our ablation study analysis. Due to the hardware constraints, our ablation study was limited to the simplest task among the eight games. And our analysis is constrained to the Pong game environment. The limitations in computational resources have led us to focus exclusively on the Pong game. It enables us to identify the effectiveness of the bottom-up and LRP components.



(a) AADQN with-out Bottom-up (b) AADQN wiht-out LRP

Figure 5.5 Buttom-up and LRP components ablation analysis

5.6.4. Bottom-Up Attention Ablation Analysis Results

This section, delve into the outcomes of the ablation study focused on the bottom-up attention mechanism. We specifically look at the impact of removing the bottom-up attention from the attention mechanism, highlighting any observed changes in the model's performance. The Figure 5.6, illustrates the results of the model performance in this case, and represents how these changes affect the overall performance of the model.

We observed that when removing the bottom-up (BU) component, the model's performance remained nearly same with very slight decrease in performance. This outcome was particularly evident in a straightforward game like Pong, where the game play involves just one ball without any complexity. In such uncomplicated environments, it seems that the

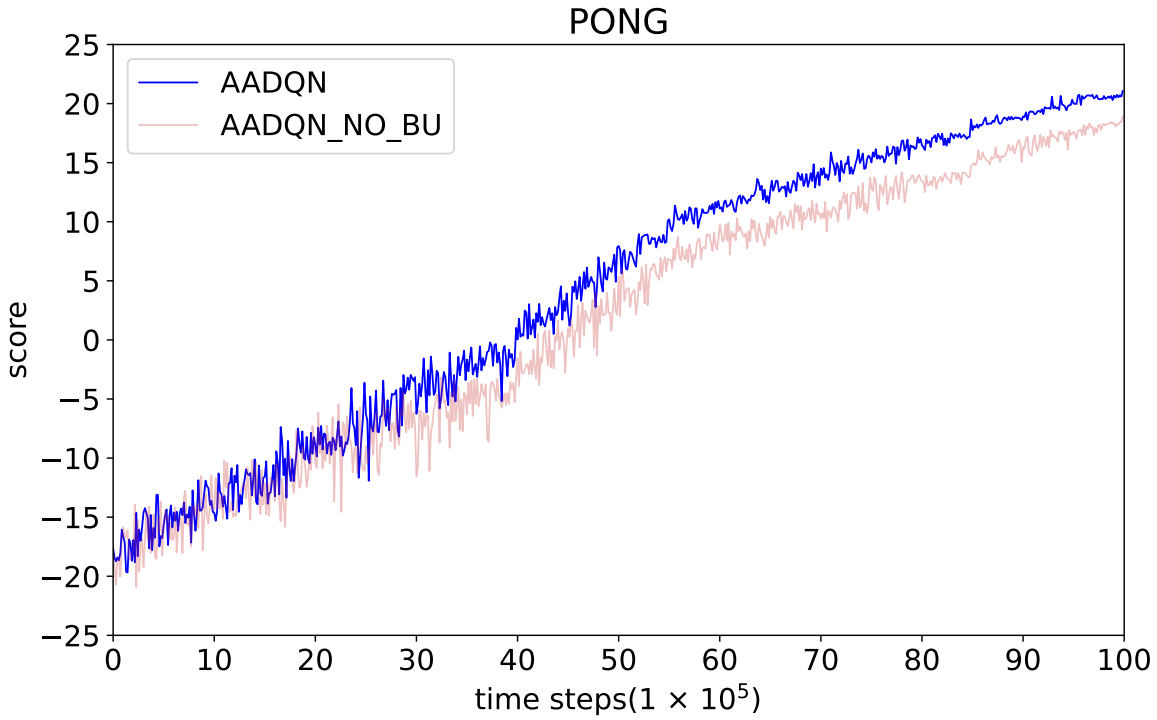


Figure 5.6 Buttom-up(BU) Ablation Analysis

feature importance value alone is adequate for justifying the bottom-up attention mechanism role. As depicted in the figure, the absence of bottom-up attention refinement component did not result in any significant alteration in the model's performance.

5.6.5. LRP-Based Transfer Learning Ablation Analysis Results

This section explores the findings from the ablation study concentrating on the using Layer-wise Relevance Propagation (LRP) to freeze irrelevant CNN neurons in a DQN architecture during training. LRP is a technique commonly employed for interpreting the decisions made by neural networks, particularly in understanding the importance of individual neurons. We use LRP to select which neurons contribute more significantly to the decision-making process. Freezing irrelevant neurons improves the stability of the model and lead to a more efficient use of computational resources.

The Figure 5.7 focus on the results of the LRP freezing mechanism, investigating the impact of removing the freezing and shedding light on how these changes affect the model's behavior and overall performance.

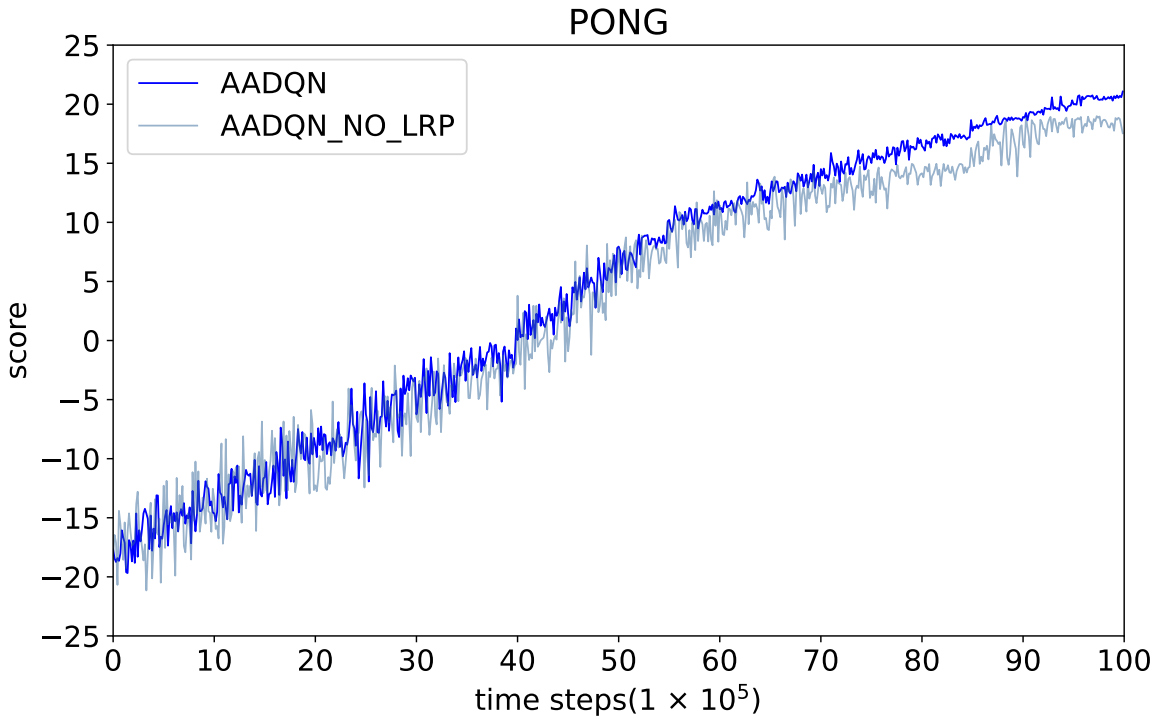


Figure 5.7 LRP Freezing Ablation Analysis

We notice that, after removing the LRP freezing, there is an increase in fluctuations. Additionally, shows a slight decrease compared to the AADQN performance throughout the training process. Hence, LRP-based freezing for irrelevant neurons in DQN providing slightly efficiency and stability advantages.

6. DISCUSSION

This study explores the field of Deep Reinforcement Learning (DRL), focusing on advancing video game-playing agents. DRL encompasses the process of training agents to engage with and navigate an environment and make sequential decisions without labeled data, primarily using the Q-learning concept. The integration of Reinforcement Learning with Deep Neural Networks, known as Deep Q-network (DQN), addresses challenges in dealing with complex environments, including the application of convolutional neural networks to process raw inputs like images. Despite DRL's success in various domains, it faces limitations, such as sample inefficiency and challenges adapting to complex tasks.

To improve the Deep Q-Network (DQN) algorithm's stability and efficiency, previous studies introduces several advancements, including Prioritized Experience Replay (PER), Dueling Neural Network Architecture (DNNA), Distributed Deep Reinforcement Architecture (DDRA), Auxiliary Functions (AF), and Attention Mechanisms (AM). PER prioritizes critical experiences for replay based on their importance, DNNA separates the estimation of value and action functions for stability, and DDRA distributes the learning process across multiple parallel agents. AFs provide denser training rewards, and attention mechanisms enhance model performance in various contexts.

This thesis introduces a unified bottom-up and top-down visual attention mechanism to address inefficiency issues in the DQN algorithm. Bottom-up attention allows agents to selectively focus on distinct regions or components of the input, while top-down attention provides task-based reward feedback to accelerate the learning process. Additionally, the thesis introduces an Auxiliary Distance Function (ADF) to enhance DQN's efficiency by addressing sparsity in main rewards feedback. ADF guides the DQN agent to focus on changes between consecutive frames, combining auxiliary distance rewards with the main reward during training to alleviate sparsity.

The effectiveness of the proposed method is showcased through experimental evaluations and comparisons with standard DQN and other algorithms. The integration of attention

mechanisms and auxiliary functions successfully enhances DQN sample efficiency, facilitating more effective and rapid learning. The evaluation considers average game scores, learning stability, and algorithmic efficiency to showcase the overall improvement of the proposed method on the DQN algorithm.

6.1. Advancements in Deep Reinforcement Learning for Video Game Agents

This thesis makes a contribution to the field of Deep Reinforcement Learning (DRL) by introducing the Particle Filter-based Attention-Augmented Deep Q-Network (AADQN), a novel attention mechanism aimed at enhancing efficiency in learning tasks. A unique feature of the AADQN model is its incorporation of Particle Filters, a method that represents the probability distribution of the state using randomly generated particles. In the AADQN framework, Particle Filters are employed to capture the distribution of pertinent features, providing an innovative approach to attention mechanisms. Additionally, the AADQN stands out by combining top-down and bottom-up attention mechanisms, further enhancing its capabilities in efficient processing and leveraging relevant information for decision-making. The incorporation of distance auxiliary functions serves as an enhancement to the Deep Q-Network (DQN), resulting in improvements in learning efficiency.

6.2. Effectiveness of Attention Mechanism

The introduction offers a comprehensive summary of the present state of Reinforcement Learning (RL), acknowledging its initial achievements in tasks such as Atari games. However, it emphasizes that RL still faces significant hurdles when it comes to tackling complex tasks and adapting to new environments. While the Deep Q-Network (DQN) algorithm has shown success, it suffers from inefficiency and inflexibility problems. These inefficiency and inflexibility issues, necessitate the introduction of innovative methods like the proposed Attention-Augmented DQN (AADQN).

The AADQN is a game agent designed to overcome these limitations. AADQN incorporates a dual visual attention mechanism, encompassing both top-down and bottom-up attention, to enhance the recognition of task-related features. Extensive evaluations across eight Atari 2600 games reveal that AADQN outperforms the baseline DQN agent, demonstrating heightened flexibility and achieving improved scores within a reduced time-step.

6.3. Comprehensive Insights from AADQN vs. DQN:

AADQN consistently outperforms DQN in terms of performance across a range of game environments, including both simpler games like Pong and Breakout, as well as more complex ones like Wizard of Wor, SpaceInvaders, Seaquest, Beamrider, and Qbert. While both algorithms initially perform similarly in simpler games, AADQN exhibits a noticeable improvement in performance beyond a certain point. In complex games, AADQN outshines DQN by approaching the optimal policy at a faster rate.

One of the key advantages of AADQN over DQN lies in its ability to handle non-stationary environments more effectively. DQN struggles with maintaining optimal performance in such dynamic settings, resulting in a decline in its scores over time. In contrast, AADQN excels by consistently delivering increasing scores as it adapts to the changing dynamics of the environment.

The empirical results clearly demonstrate that AADQN surpasses DQN in terms of both initial and performance over extended periods or in the long run across a diverse set of games. This highlights the effectiveness of incorporating attention mechanisms into the DQN framework, enabling AADQN to better capture and leverage important information during the learning process. These findings indicate the potential of AADQN as a robust and flexible algorithm for reinforcement learning tasks, particularly in dynamic game environments.

6.3.1. Enhancing Computational Performance

The computational efficiency of AADQN in this thesis is evaluated regarding the overall duration or total time steps. The incorporation of attention mechanisms into DQN serves a crucial role in optimizing the learning process, contributing to a more efficient training paradigm. A notable element in assessing computational efficiency is the reduction in total time steps achieved through the use of attention mechanisms. The attention mechanisms, including both top-down and bottom-up attention components, work together to refine the agent's focus on relevant information within the environment. These mechanisms effectively filter out less critical details, allowing the agent to focus on the most informative aspects of the input data. The impact of attentional mechanisms on computational efficiency is profound. Rather than relying on exhaustive exploration or processing of the entire input space, AADQN uses attention to selectively attend to salient features. This selective attention significantly speeds up the decision-making process, allowing the agent to reach optimal actions more quickly. We do not consider the additional computational overhead from attention mechanisms. Becuase, the parallelizability of the attention mechanisms components enhances efficiency without adding extra burden. For example, particle filters, a crucial element in top-down attention, can be sampled in parallel, without imposing additional computational load. This parallel processing helps in decreasing the overall computational cost compared to the baseline DQN. As a result, the computational efficiency of AADQN is a synergistic result of components within its architecture. The decrease in the total number of time steps highlights how the attention mechanisms contribute to more efficient learning, ultimately improving the overall effectiveness of the training process. This efficiency not only speeds up how quickly the agent learns but also makes AADQN a more practical and adaptable solution for addressing challenges in complex and dynamic environments in reinforcement learning scenarios.

6.3.2. Stability and Robustness Analysis of AADQN

The decline in the average score performance of the DQN algorithm, particularly in environments like Space Invaders, Seaquest, or Beamrider at specific training intervals, can be attributed to two primary factors: sparse rewards and non-stationarity. In environments with sparse rewards, the agent receives infrequent or delayed positive feedback, posing a challenge for the DQN agent to associate its actions with favorable outcomes. If the agent encounters a prolonged lack of positive reinforcement, it may struggle to effectively update its policy, leading to a temporary dip in performance.

Environments like Space Invaders and Seaquest exhibit non-stationarity, indicating that the optimal policy can change over time. DQN, however, operates under the assumption of a stationary environment. When this assumption is violated, the algorithm encounters difficulties in adapting to the evolving optimal policy, resulting in fluctuations in the agent's performance.

In the context of reinforcement learning, non-stationarity refers to scenarios where the optimal policy— the most effective set of actions for an agent—shifts over time. This shift is not indicative of the environment itself undergoing changes but rather suggests that the optimal strategy for the agent may vary due to factors like the evolution of enemy behavior or dynamic elements within the environment. For instance, in games like Space Invaders, the behavior of enemies may undergo alterations as the game progresses, with adversaries becoming more aggressive, faster, or employing different attack patterns. This changing behavior necessitates adaptation in the learned policy to effectively navigate the evolving challenges presented by the non-stationary environment.

AADQN can help address these challenges in non-stationary environments, such as those seen in games like SpaceInvaders. By incorporating a novel attention mechanism, combining both top-down and bottom-up attention, this dual attention allows the agent to selectively focus on relevant features or dynamics in the environment. By emphasizing dynamic areas where changes occur, AADQN can adapt more quickly to evolving dynamics in

non-stationary environments. Moreover, AADQN employs Particle Filters to represent the probability distribution of the state that can enhance attention mechanisms by capturing the distribution of relevant features. This helps the agent effectively respond to changes in the environment, promoting adaptability in non-stationary scenarios.

The reduced fluctuation observed in the start-up and convergence phases of AADQN compared to DQN can be attributed to the mechanisms integrated into DQN. AADQN demonstrates more stable learning during convergence, indicating consistent performance once converged. Significant fluctuations in DQN's start-up phase suggest instability or inconsistent learning. Here are potential reasons for the lower fluctuation: AADQN employs a dual attention mechanism, combining both top-down and bottom-up attention. This selective attention allows the agent to concentrate on pertinent features and ignore less critical or less significant information during the initial phases of learning. The attention mechanisms guide the agent in making more informed decisions, reducing the early-stage randomness in actions and, consequently, fluctuation. Also, leveraging particle filters, AADQN can reduce uncertainty after the start-up phase, leading to a more stable learning process.

In summary, AADQN's attention mechanisms, particle filter-based top-down attention, and integration with bottom-up provide a learning process that is more adaptable and stable. This allows the agent to navigate non-stationary environments more effectively by focusing on relevant changes, improving learning efficiency, and efficiently learning during the start-up and convergence phases, resulting in reduced fluctuation.

6.4. Comparative Evaluation of AADQN in the Literature

In our comparative evaluation of AADQN with other prominent studies such as A3C and Dueling DQN, we observed varying performance in different video game environments. In particular, AADQN showed superior overall performance in complex game environments, highlighting its effectiveness in tackling challenging tasks.

Looking at specific games, we found that A3C and Dueling DQN performed exceptionally well in Wizard of Wor, highlighting their efficiency in exploring different actions. These

algorithms demonstrated the ability to navigate through complex game scenarios and make informed decisions. On the other hand, Dueling DQN outperformed in Qbert due to its separation of value and benefit functions. This separation enabled Dueling DQN to better estimate the values of different actions, leading to more effective decision making and ultimately higher performance in the Qbert game. The comparative evaluation not only highlights the strengths and weaknesses of each algorithm, but also underscores the significance of taking into account the unique attributes or characteristics of the game environment when choosing a suitable deep reinforcement learning approach. AADQN's robust performance on a variety of complex games suggests its potential as a versatile and effective algorithm in a wide range of video game scenarios. To summarize, these findings enhance our comprehension of the strengths and limitations of different deep reinforcement learning algorithms in video game environments. They also offer valuable insights into the appropriateness or suitability of AADQN for improving the performance of video game agents, particularly in complex and dynamic environments. Further research and experimentation can build on these findings to advance the domain or application of deep reinforcement learning in video game agents, leading to improvements in gameplay performance and intelligent decision-making.

6.5. Effectiveness of Auxiliary Distance Function Integration

In the second phase of this thesis, the investigation shifts towards the integration of the Auxiliary Distance Function (ADF) into DQN gaming agents, with the goal of promoting more efficient and robust learning in gaming environments. By incorporating the ADF into the DQN framework, we aim to address the inefficiency challenges associated with traditional DQN algorithms and improve the performance of autonomous game agents. To evaluate the impact of ADF on the early-stage learning of DQN, we introduce the Auxiliary Distance Reward augmented DQN (ADFDQN). ADFDQN shows mixed results in different game environments. In some simple dynamic tasks, ADFDQN shows improved learning efficiency compared to baseline DQN. The inclusion of ADF allows the agent to better understand and exploit the distances between objects or relevant features in the environment,

facilitating more effective decision making. However, in the early stages of complex environments such as Wizard of Wor, the performance of ADFDQN is not as effective as desired. The complexity and dynamism of these environments can make it difficult for ADFDQN to capture and use distance information optimally. As a result, its performance in these particular scenarios falls short of the baseline DQN. Interestingly, in environments without significant dynamics, such as Qbert, the performance of ADFDQN is exhibits inferior performance compared to the baseline DQN. This result suggests that the inclusion of ADF may not be beneficial in environments where dynamics play a minor role and other factors, such as pattern recognition or strategic decision making, take precedence. Overall, the integration of ADF into the DQN framework represents an approach to improving learning efficiency in certain dynamic gaming environments. However, its effectiveness is context dependent, with performance outcomes varying across tasks and environments. These findings shed light on the interplay between distance-based auxiliary functions and the complexity of game environments, yielding valuable insights that can guide future research endeavors in the field of autonomous gaming. By extending the capabilities of DQN and exploring the integration of auxiliary distance reward in initial stage of the DQN, this thesis contributes to the understanding of DQN techniques deeply in autonomous gaming. These findings form the basis for further research in DRL and refinement of algorithms to improve the efficiency, adaptability and robustness of game-playing agents.

7. CONCLUSION

The DQN algorithm has made significant contributions to the field of DRL by demonstrating its capability to learn directly from high-dimensional raw sensory inputs. However, the DQN algorithm exhibits certain inefficiencies that limit its performance and hinder its applicability in complex and challenging environments primarily stem from the lack of sample efficiency. In response to the inefficiency problems of the DQN algorithm, this thesis has proposed a novel method that addresses these limitations and significantly enhances the overall performance and sample efficiency of DQN. By incorporating a combination of techniques, including bottom-up and top-down visual attention mechanism, the proposed method tackles the challenges of inefficient learning of DQN in open-ai gym environments. Through experimentation and evaluation on various game environments, the findings illustrate the effectiveness of the proposed method compared to the standard DQN algorithm.

7.1. Key Findings of Attention-Augmented Deep Q-Network (AADQN)

Motivated by the lack of efficiency and flexibility of standard DQN, this thesis proposed a new attention-augmented DQN model (AADQN), which integrates bottom-up and top-down attention mechanisms, surpasses DQN performance in shorter steps, and demonstrates adaptability in intricate environments.

While the bottom-up method identifies basic lower-level features, it may miss task-related information due to noise or complexity by itself. The particle-filter-based top-down attention mechanism addressed this limitation and enhanced DQN's performance. Additionally, by utilizing LRP, unimportant neurons on CNN were identified and frozen, improving DQN robustness.

Leveraging a combination of particle filter and gradient descent impacts the determination of gradient direction in a way that causes time-step basis acceleration during the training process and improves the average score by about 134.93% across the eight game environments. The particle-filter-based attention reduces fluctuation and provides more reliable convergence,

together with alleviating the necessity for extensive training data that are typically required by baseline DQN.

When comparing with other studies, it is worth noting that in certain complex games like 'Wizard of Wor' and 'Qbert', AADQN was outperformed by Asynchronous Advantage Actor Critic (A3C) and Dueling DQN algorithms, respectively. However, if we exclude these two games from our analysis, we can generally observe that AADQN performs relatively well in complex game environments.

In future research, it would be valuable to conduct more extensive comparisons with other methods, such as Gorila and Double DQN, and provide a more comprehensive analysis involving other game environments in varying complexity. Furthermore, future research in this field could explore the impact of clustering the extracted feature map of a CNN to reduce the dimensionality of the attention vector. This clustering technique may have the potential to enable game agents to attend to multiple areas simultaneously, thereby enhancing their ability to focus on multiple regions of interest at once. Furthermore, inhibitory/negative signals of LRP can also be taken into account, which might further enhance its contribution by refining its functionality.

7.2. Key Findings of Auxiliary Distance Function Augmented DQN (ADFDQN)

The findings of this research provide valuable insights into mitigating the inefficiency issues of DQN and pave the way for more efficient DQN algorithms. This section summarizes and highlights the importance of auxiliary functions in addressing the inefficiency limitation of the baseline DQN. After successfully implementing ADF-augmented DQN across eight different environments, we have concluded that ADF improves the learning efficiency of the DQN during the early stages of specific tasks. However, notable challenges arose in complex tasks, such as 'Wizardofwar,' where the DQN agent confronted numerous dynamics, some of which were not directly relevant to the task at hand. Consequently, this complexity led to potential misdirection of the DQN agent in the early phases of training.

Furthermore, in environments devoid of dynamics like Qbert, we observed that the results were even worse than the baseline DQN, highlighting a potential limitation of the ADF approach in static environments. This section compared the average score performance of the early-stage performance of the Auxiliary Distance Function Augmented DQN algorithm with the standard DQN algorithm. Keep in mind that the relative performance may vary depending on the specific task and the choice of auxiliary tasks used. Additionally, note that comparing early-stage performance may not necessarily reflect the long-term performance of the algorithms, as their learning dynamics might differ over time. Therefore, it's crucial to consider the long-term performance as well if you want to make a comprehensive assessment of the algorithms.

7.3. Limitations of the Study

We introduced AADQN, an attention-augmented Deep Q-Network that integrates both bottom-up and top-down attention mechanisms into the DQN baseline architecture. This innovative approach yields improvements over traditional DQN baseline methods when applied to video games in the OpenAI Gym environments. Our method achieves two significant breakthroughs. Firstly, AADQN mitigates the exponential increase in the need for high samples when tackling complex tasks. Secondly, it partially addresses the stability limitations inherent in DQN, providing a more stable behavior with reduced fluctuations and minimizing the risk of losing learned policies during training. Despite these advancements, AADQN does have limitations. Looking ahead, our future plans involve expanding the applicability of our algorithm to more intricate settings, such as its integration into real-world systems. Additionally, while this study focused on evaluating AADQN across eight arcade environments out of nearly 60 tasks, there is room for extending the evaluation to other game environments in future research. In each game, AADQN consistently outperforms the DQN baseline by selecting the maximum performance. Notably, all AADQN scores represent values achieved before reaching the maximum score for each game due to the early termination of AADQN runs caused by hardware constraints. We believe that if we had been able to train AADQN for the same number of timesteps required to reach the maximum score, our results would have demonstrated even further improvement.

It's important to emphasize that, in this study, our examination of auxiliary comparison is limited to the initial stages due to hardware constraints, and we are unable to present the long-term average scores until the completion of training. Exploration of both early and long-term performance could significantly contribute to a deeper understanding of the algorithm's capabilities and its effectiveness over extended training periods.

7.4. Future Research Recommendations

Looking ahead, an exciting avenue for future research AADQN model proposes exploring the idea of training the AADQN model on a specific eight OpenAI Gym games, like Breakout, using the same architecture. This could potentially enable fine-tuning the model for another environments. The concept here is to utilize the knowledge acquired from training on one game and apply them to improve the performance and adaptability of the model when transitioning to a different gaming environment. Investigating the potential of transfer learning in this context not only expands the usability of the AADQN model but also contributes to a more in-depth understanding of how well it can adapt and perform efficiently across a variety of gaming scenarios. This recommendation encourages further exploration into the details of transfer learning within the AADQN framework, paving the way for advancements in its ability to generalize and its potential deployment in diverse environments.

In future research, it would be valuable to expand the scope of comparisons by including a broader range of methods, such as General Reinforcement Learning Architecture such as Gorila. These algorithms have demonstrated promising results in various RL domains and could provide insightful benchmarks for evaluating the performance of the proposed approach. A more comprehensive analysis involving additional game environments with varying levels of complexity would also contribute to a better understanding of the algorithm's generalizability and scalability across different problem domains.

Moreover, future research in this field could explore the potential impact of clustering techniques applied to the extracted feature map of a Convolutional Neural Network (CNN). By clustering the feature map, it might be possible to reduce the dimensionality of the attention vector, potentially enhancing the game agent's ability to attend to multiple areas simultaneously. This approach could offer a more efficient and effective way for agents to allocate attention across different regions of interest, leading to improved decision-making and performance in complex game environments with diverse visual stimuli.

Additionally, investigating the integration of negative signals within the Layer-wise Relevance Propagation (LRP) framework could be a promising avenue for future research. By incorporating these signals into the LRP process, it may be possible to refine the functionality and contribution of LRP, potentially leading to improved interpretability and understanding of the model's decision-making process. This could provide valuable insights into the inner workings of the RL agent and enhance its ability to explain its actions and decisions in a more transparent and interpretable manner.

Irrelevant neurons might become noticeable or apparent only in certain or specific situations. Neurons that were irrelevant in one phase of training might become important in other phases, and freezing them could impede the model's ability to adapt. Moreover, the freezing of neurons might affect the generalization ability of the model. Future research can be consider a freezing strategy, rather than freezing of all irrelevant neurons, identify and freeze those that consistently demonstrate low relevance across a range of various different environments.

Exploring the performance of the Auxiliary Distance Function Augmented DQN algorithm in comparison to the standard DQN algorithm, both in the early learning phases and during convergence, holds potential for shedding light on learning dynamics and the benefits of integrating auxiliary tasks. As we delve into future research, a comprehensive exploration of both early and long-term performance could further enhance our understanding of the algorithm's capabilities.

By addressing these avenues for future research, we can further advance the field of RL in game playing and deepen our understanding of the capabilities and limitations of different algorithms. These endeavors have the potential to drive innovation, foster new insights, and pave the way for the development of more robust and intelligent game playing agents.

7.5. Future Avenues for Enhanced Ablation Studies

It's important to note that limitation in ablation study emphasizing the non-complex task training. Moreover, Conducting the ablation study on the simplest task within the constrained scenario serves as a necessary starting point for future research. We acknowledge the

importance of evaluating the model ablation analysis across a broader range of environments. This baseline analysis serves as a guiding reference for subsequent studies aiming to expand the ablation study to encompass more diverse environments, enabling a comprehensive evaluation of the model's capabilities.

Despite the hardware limitations that confined our ablation study, we offer recommendations for extending ablation studies in the future. By addressing these limitations, future research endeavors can build upon our findings, contributing to a deeper understanding of the intricate relationships within attention components.

7.6. Auxiliary Distance Function (ADF)

Due to the limitations imposed by our hardware, we were unable to thoroughly explore auxiliary functions throughout the entire training process. This constraint forced us to focus solely on the early stages of model training. However, in the interest of transparency and providing comprehensive information, we have included this constrained experiment in the recommendations section of the thesis.

Comparing the performance of the Auxiliary Distance Function Augmented DQN algorithm with the standard DQN algorithm in the early stages of training can provide insights into the initial learning dynamics and potential benefits of incorporating auxiliary tasks.

The choice and design of an auxiliary function depend on the RL problem and the desired learning goals and can take different forms depending on the specific problem. It requires careful consideration and experimentation to determine the most effective auxiliary function that complements the main RL task and enhances learning performance. The objective of this section is to evaluate the efficacy or effectiveness of incorporating the ADF in enhancing the performance of the DQN during the initial phases of the training process across eight experimental environments. By introducing the ADF as an additional component to the DQN algorithm, we seek to evaluate its impact on the agent's learning capabilities in the early stage of these diverse environments. Our agent tries to maximize ADF feedback rewards

simultaneously by DQN reward. Experiment results, show how the inclusion of the ADF influences the DQN agent's performance during its early stages.

We conduct a comparative analysis concerning DQN in terms of the game average scores in the early stages of each task. The average score analysis includes the number of steps required to achieve a certain average score in the initial phase of the learning journey. Table 7.1 provides the comparison of training the ADF augmented DQN (ADFDQN) and DQN baseline agent based on average scores achieved on the eight tasks at the early stages for each task. We do not show the progress of the two agents in long-term average scores until the end of the training. At the defined evaluation intervals, record the average score for both algorithms. Execute the training process for the same number of time steps, ensuring that both algorithms are trained for an equal duration. Table 7.1 just demonstrates the average score of the initial stages performance comparison between both algorithms. Experiment results in Table 7.1 show that ADFDQN is not effective in the early stage of complex environments like Wizardofwor. But improves the learning efficiency of the DQN during the early stages of some simple dynamic tasks. Furthermore, in environments with a lack of dynamics like Qbert, the performance is worse than the baseline DQN,

	DQN Baseline		**DQN Augmented ADF**	
Game	$TimeSteps$	$Avg.Score$	$TimeSteps$	$Avg.Score$
Pong	$\approx 1 \times 10^6$	-16.7	$\approx 7 \times 10^5$	**-13.9**
Wizard of Wor	$\approx 5 \times 10^6$	50	$\approx 5 \times 10^6$	31
SpaceInvaders	$\approx 1 \times 10^6$	33	$\approx 1 \times 10^6$	26
Breakout	$\approx 1 \times 10^6$	5.5	$\approx 6 \times 10^5$	**6.5**
Asterix	$\approx 3 \times 10^6$	300	$\approx 2 \times 10^6$	**320**
Seaquest	$\approx 1 \times 10^6$	40	$\approx 8 \times 10^5$	**55**
Beamrider	$\approx 5 \times 10^6$	25	$\approx 3 \times 10^6$	**28**
Qbert	$\approx 1 \times 10^6$	45	$\approx 1 \times 10^6$	17

Table 7.1 DQN vs. ADFDQN Average score comparison

Although our exploration was limited to the initial training phases, we still obtained valuable insights from this analysis. These insights lay the groundwork for future investigations and highlight the importance of expanding these experiments beyond the early training stages. In this section of the thesis, we discuss the implications of this limitation and propose directions for future research. By extending the analysis to encompass later stages of training, can gain a more comprehensive understanding of the dynamics and impact of auxiliary functions.

In summary, due to hardware constraints, our exploration of auxiliary functions was confined to the early stages of training. Despite this limitation, the insights gained from this analysis are valuable and provide a starting point for further research. We emphasize the need to extend these experiments beyond the initial training stages to gain a more comprehensive understanding of how auxiliary functions influence the model.

REFERENCES

[1] Jos Elfring, Elena Torta, and René van de Molengraft. Particle filters: A hands-on tutorial. *Sensors*, 21(2):438, **2021**.

[2] Seul-Ki Yeom, Philipp Seegerer, Sebastian Lapuschkin, Alexander Binder, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. Pruning by explaining: A novel criterion for deep neural network pruning. *Pattern Recognition*, 115:107899, **2021**.

[3] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, **2015**.

[4] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, **2013**.

[5] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, **2015**.

[6] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, **2018**.

[7] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, **1992**.

[8] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, **2013**.

[9] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*. **2018**.

[10] Will Dabney, Mark Rowland, Marc Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32. **2018**.

[11] Arun Nair, Praveen Srinivasan, Sam Blackwell, Cagdas Alcicek, Rory Fearon, Alessandro De Maria, Vedavyas Panneershelvam, Mustafa Suleyman, Charles Beattie, Stig Petersen, et al. Massively parallel methods for deep reinforcement learning. *arXiv preprint arXiv:1507.04296*, **2015**.

[12] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International conference on machine learning*, pages 1352–1361. PMLR, **2017**.

[13] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, **2016**.

[14] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. *Advances in neural information processing systems*, 27, **2014**.

[15] Ivan Sorokin, Alexey Seleznev, Mikhail Pavlov, Aleksandr Fedorov, and Anastasiia Ignateva. Deep attention recurrent q-network. *arXiv preprint arXiv:1512.01693*, **2015**.

[16] Anthony Manchin, Ehsan Abbasnejad, and Anton van den Hengel. Reinforcement learning with attention that works: A self-supervised approach. In *International conference on neural information processing*, pages 223–230. Springer, **2019**.

[17] Lennart Bramlage and Aurelio Cortese. Generalized attention-weighted reinforcement learning. *Neural Networks*, 145:10–21, **2022**.

[18] Siyu Lu, Yueming Ding, Mingzhe Liu, Zhengtong Yin, Lirong Yin, and Wenfeng Zheng. Multiscale feature extraction and fusion of image and text in vqa. *International Journal of Computational Intelligence Systems*, 16(1):54, **2023**.

[19] Siyu Lu, Mingzhe Liu, Lirong Yin, Zhengtong Yin, Xuan Liu, and Wenfeng Zheng. The multi-modal fusion in visual question answering: a review of attention mechanisms. *PeerJ Computer Science*, 9:e1400, **2023**.

[20] Gangyan Zeng, Yuan Zhang, Yu Zhou, Xiaomeng Yang, Ning Jiang, Guoqing Zhao, Weiping Wang, and Xu-Cheng Yin. Beyond ocr+ vqa: Towards end-to-end reading and reasoning for robust and accurate textvqa. *Pattern Recognition*, 138:109337, **2023**.

[21] Zhiyang Ma, Wenfeng Zheng, Xiaobing Chen, and Lirong Yin. Joint embedding vqa model based on dynamic word vector. *PeerJ Computer Science*, 7:e353, **2021**.

[22] Xi Zhang, Feifei Zhang, and Changsheng Xu. Reducing vision-answer biases for multiple-choice vqa. *IEEE Transactions on Image Processing*, **2023**.

[23] Wenfeng Zheng, Lirong Yin, Xiaobing Chen, Zhiyang Ma, Shan Liu, and Bo Yang. Knowledge base graph embedding module design for visual question answering model. *Pattern recognition*, 120:108153, **2021**.

[24] Yuxi Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, **2017**.

[25] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. **1960**.

[26] Rudolph E Kalman and Richard S Bucy. New results in linear filtering and prediction theory. **1961**.

[27] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, **2015**.

[28] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, **2016**.

[29] Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *International conference on machine learning*, pages 449–458. PMLR, **2017**.

[30] Muhammad Rizki Maulana and Wee Sun Lee. Ensemble and auxiliary tasks for data-efficient deep reinforcement learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 122–138. Springer, **2021**.

[31] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, **2016**.

[32] Sajad Mousavi, Michael Schukat, Enda Howley, Ali Borji, and Nasser Mozayani. Learning to predict where to look in interactive environments using deep recurrent q-learning. *arXiv preprint arXiv:1612.05753*, **2016**.

[33] Ruohan Zhang, Zhuode Liu, Luxin Zhang, Jake A Whritner, Karl S Muller, Mary M Hayhoe, and Dana H Ballard. Agil: Learning attention from human for visuomotor tasks. In *Proceedings of the european conference on computer vision (eccv)*, pages 663–679. **2018**.

[34] Alexander Mott, Daniel Zoran, Mike Chrzanowski, Daan Wierstra, and Danilo Jimenez Rezende. Towards interpretable reinforcement learning using attention augmented agents. *Advances in Neural Information Processing Systems*, 32, **2019**.

[35] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *2015 aaai fall symposium series*. **2015**.

[36] Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20(11):1254–1259, **1998**.

[37] Liu Yuezhang, Ruohan Zhang, and Dana H Ballard. An initial attempt of combining visual selective attention with deep reinforcement learning. *arXiv preprint arXiv:1811.04407*, **2018**.

[38] Samuel Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. Visualizing and understanding atari agents. In *International conference on machine learning*, pages 1792–1801. PMLR, **2018**.

[39] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, **2016**.

[40] David Silver, Hado Hasselt, Matteo Hessel, Tom Schaul, Arthur Guez, Tim Harley, Gabriel Dulac-Arnold, David Reichert, Neil Rabinowitz, Andre Barreto, et al. The predictron: End-to-end learning and planning. In *International Conference on Machine Learning*, pages 3191–3199. PMLR, **2017**.

[41] Jielei Wang, Ting Jiang, Zongyong Cui, and Zongjie Cao. Filter pruning with a feature map entropy importance criterion for convolution neural networks compressing. *Neurocomputing*, 461:41–54, **2021**.

[42] Mary L McHugh. The chi-square test of independence. *Biochemia medica*, 23(2):143–149, **2013**.

[43] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, **2016**.

[44] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1529–1538. **2020**.

[45] Cheonghwan Hur and Sanggil Kang. Entropy-based pruning method for convolutional neural networks. *The Journal of Supercomputing*, 75:2950–2963, **2019**.

[46] Mohammadreza Soltani, Suya Wu, Jie Ding, Robert Ravier, and Vahid Tarokh. On the information of feature maps and pruning of deep neural networks. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 6988–6995. IEEE, **2021**.

[47] M Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on signal processing*, 50(2):174–188, **2002**.

[48] Patcharin Kamsing, Peerapong Torteeka, and Soemsak Yooyen. An enhanced learning algorithm with a particle filter-based gradient descent optimizer method. *Neural Computing and Applications*, 32:12789–12800, **2020**.

[49] Daniel Grest and Volker Krüger. Gradient-enhanced particle filter for vision-based motion capture. In *Workshop on human motion*, pages 28–41. Springer, **2007**.

[50] Stas Goferman, Lihi Zelnik-Manor, and Ayellet Tal. Context-aware saliency detection. *IEEE transactions on pattern analysis and machine intelligence*, 34(10):1915–1926, **2011**.

[51] Jianming Zhang and Stan Sclaroff. Saliency detection: A boolean map approach. In *Proceedings of the IEEE international conference on computer vision*, pages 153–160. **2013**.

[52] Marcella Cornia, Lorenzo Baraldi, Giuseppe Serra, and Rita Cucchiara. Predicting human eye fixations via an lstm-based saliency attentive model. *IEEE Transactions on Image Processing*, 27(10):5142–5154, **2018**.

[53] Elham Saraee, Mona Jalal, and Margrit Betke. Visual complexity analysis using deep intermediate-layer features. *Computer Vision and Image Understanding*, 195:102949, **2020**.

[54] Yang He, Xuanyi Dong, Guoliang Kang, Yanwei Fu, Chenggang Yan, and Yi Yang. Asymptotic soft filter pruning for deep convolutional neural networks. *IEEE transactions on cybernetics*, 50(8):3594–3604, **2019**.

[55] Dor Livne and Kobi Cohen. Pops: Policy pruning and shrinking for deep reinforcement learning. *IEEE Journal of Selected Topics in Signal Processing*, 14(4):789–801, **2020**.

[56] Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, **2017**.

Appendices

Appendix 1: Experiment Environments

The OpenAI Gym Atari environment is a widely used benchmark for evaluating reinforcement learning algorithms in the domain of video games. This appendix offers a comprehensive depiction of the Atari environment and its specific games that were utilized in this thesis. Using the OpenAI Gym Atari environment in this thesis allowed the algorithm to receive observations (game frames) as input and select actions to interact with environment. The environment also provided rewards and termination signals to evaluate the algorithm's performance. The Atari environment consists of a collection of classic Atari 2600 games, such as Pong, Breakout, Space Invaders, and many more. These games are ideal for evaluating reinforcement learning algorithms due to their rich and diverse dynamics, complex visual inputs, and challenging gameplay. Each game is treated as a separate task, requiring the agent to learn appropriate actions to maximize its rewards.

In this thesis, a subset of Atari games (Pong, Breakout, SpaceInvaders, Asterix, Seaquest, Beamrider, Qbert, Wizarofwar) was selected to investigate the performance of proposed algorithm. The chosen games were carefully chosen to represent a range of complexities. These games include both simple and visually intricate environments, enabling a thorough analysis of the algorithm under various conditions. To conduct experiments, the OpenAI Gym Atari environment provided a standardized interface for interacting with the games. This interface allowed the algorithm to receive observations (game frames) as input and select actions to be executed within the games. The environment also provided rewards and termination signals to evaluate the agent's performance.

Detailed information about the Atari games used, including their specific features can be found in the following sections of this thesis appendix.

1.Pong

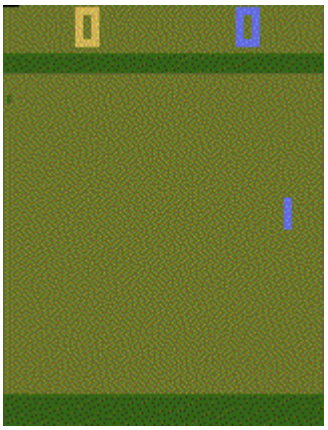


Figure 7.1 Pong Atari environment

Pong is a two-player table game where the objective is to score points by hitting a ball back and forth between two paddles. The game is played on a rectangular grid, with each player controlling a paddle that can move up and down along the side of the screen. The ball moves horizontally across the screen, bouncing off the paddles and the walls at the top and bottom. By using the Pong environment, this thesis aimed to investigate how the reinforcement learning algorithm could learn effective strategies to play simple games without complexity.

2.Breakout

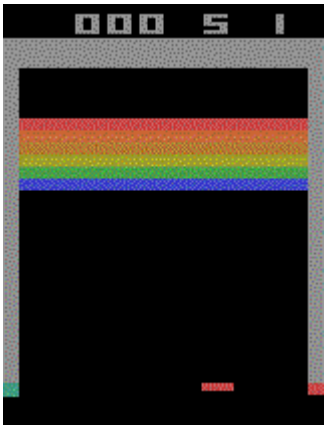


Figure 7.2 Breakout Atari environment

The Breakout game is a single-player game where the player maneuvers or controls a paddle positioned at the lower section of the screen, bouncing a ball off the paddle to break a wall of bricks located at the upper part of the screen. The goal is to eliminate or remove all the bricks while preventing the ball from falling below the paddle. The player earns points for each brick destroyed, and the game becomes progressively more challenging as the number of bricks decreases.By utilizing the Breakout environment, this thesis aimed to investigate how the proposed algorithm could learn effective strategies to clear the bricks, anticipate the ball's trajectory, and make precise paddle movements.

3.Qbert

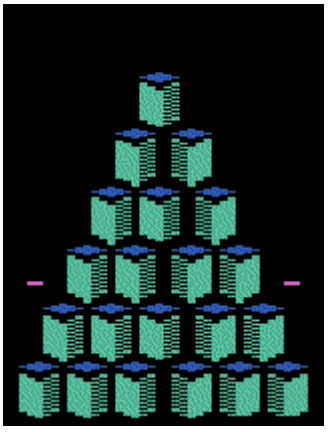


Figure 7.3 Qbert Atari environment

The Qbert game is a single-player game where the player controls a character named Qbert, who must hop on a pyramid of cubes to change their colors. The objective is to change the color of all the cubes while avoiding various enemies that roam the pyramid. Qbert must strategically navigate the pyramid, avoiding hazards and enemies. By utilizing the Qbert, this thesis aimed to investigate how the proposed algorithm could effectively navigate the pyramid, change cube colors, and avoid enemies in order to maximize its score. The algorithm's ability to learn optimal strategies, adapt to changing enemy behavior, and make strategic decisions in the dynamic and challenging world of Qbert was examined.

4.Seaquest

Figure 7.4 Seaquest Atari Environment

The Seaquest game is an underwater rescue mission game where the player controls a submarine navigating the depths of the ocean. The objective is to save divers stranded at various depths while battling against hostile sea creatures while avoiding collisions with obstacles and enemies. In this thesis, the Seaquest environment was utilized to assess the performance of the suggested algorithm when compared with some more complexity. We aimed to investigate the algorithm's ability to learn long-term planning, adapt to changing situations, and optimize its actions in the dynamic and challenging world of Seaquest.

5.Asterix

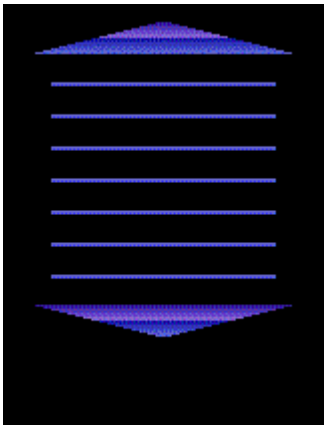


Figure 7.5 Asterix Atari Environment

The Asterix is a classic video game that player controls Asterix, the main character, and help Asterix to navigate and collect treasures while avoiding obstacles and enemies. Asterix can defeat enemies by jumping on their heads or by using a limited supply of magic potions. The game features different types of enemies. In the context of our thesis, we are using the Asterix game to provide a description of the proposed methods performance about enemy behaviors.

6.Beamrider

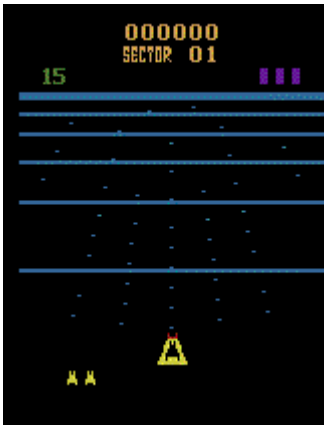


Figure 7.6 Beamrider Atari Environment

The Beamrider game is a single-player space shooter game where the player controls a spaceship that moves horizontally at the bottom of the screen. The objective is to destroy enemy ships and other obstacles while navigating through a series of space lanes. The player must avoid colliding with enemy projectiles and obstacles while aiming to achieve high scores. By utilizing the Beamrider environment, this thesis aimed to investigate how proposed algorithm could learn effective strategies to destroy enemy ships and navigate through the challenging space lanes to maximize its score. The algorithm's ability to adapt to changing enemy formations, make strategic decisions, and optimize its actions in dynamic gameplay of Beamrider was examined.

7.Wizard of Wor

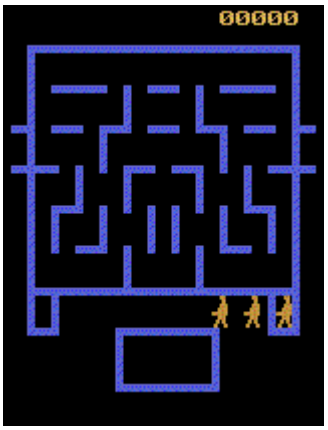


Figure 7.7 Wizard of Wor Atari Environment

The Wizard of Wor game is a classic maze-based shooter game that offers both single-player and two-player cooperative gameplay. In the Wizard of Wor game, players control characters who navigate through maze. The objective is to eliminate enemy creatures while avoiding their attacks. The game provide challenging layout of mazes and enemy configurations. Players can shoot to defeat the enemy, but they must be careful to avoid being hit by enemy fire. In the context of this thesis, investigate the proposed method performance by considering gameplay strategy about enemy behavior, evaluating its ability to navigate the complex maze and timing to defeat.

8.SpaceInvaders

Figure 7.8 SpaceInvaders Atari Environment

In Space Invaders, the player controls a spaceship positioned at the bottom of the screen, moving horizontally to dodge enemy projectiles and firing back at rows of descending alien invaders. The objective is to eliminate all the aliens before they reach the bottom of the screen. The aliens gradually advance and increase their speed. The player can take cover behind destructible barriers to shield themselves from enemy fire, but the barriers also gradually deteriorate and can be destroyed by both the player and the aliens. This thesis discusses the performance of the suggested algorithm engaging in SpaceInvaders, evaluating its ability to dodge enemy projectiles, strategically eliminate aliens, and potentially optimize its gameplay strategy.