Addis Ababa University

Addis Ababa Institute of Technology

School of Electrical and Computer Engineering

Testing the Invariance of Skills and Strategies Developed by Artificial Agents under Different Sensory Modalities

By **Meseret G/Michael**

A Thesis Proposal Submitted to the School of Graduate Studies of Addis Ababa University in Partial Fulfillment Degree of Masters of Science in Computer Engineering

Advisor **Dr.Menore Tekeba**

June, 2024

Addis Ababa, Ethiopia

Declaration

We declare that this project document is our original work and has not been presented in any university.

Name of the students	Signature	Date
1) Meseret G/Michael	----------	----------

Name of the academic advisor	Signature	Date
2) Dr. Menore Tekeba	----------	----------

Abstract

Artificial intelligence (AI) is the intelligence of machines or software, as opposed to the intelligence of humans or animals by developing artificial intelligence agents. Artificial Intelligence (AI) agent must be aware of the external environment to understand or to execute tasks. The interaction between an agent and the environment is achieved by the aid of input sensory modality (image, sound, lidar or other input modality) using metrics reward sizes they collect during reinforcement learning. To do so, the we used two algorithms, Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC) for the experiments. PPO is a reinforcement learning algorithm that is used to train agents to perform tasks in environments through trial and error. It is designed to optimize policies, which are the strategies or behaviors that the agents use to make decisions. PPO aims to find the optimal policy by iteratively updating the parameters based on the collected experiences from interactions with the environment. SAC is a reinforcement learning algorithm used for training agents in environments with continuous action spaces. It is an extension of the actor-critic framework that combines the advantages of both policy optimization and value estimation methods. During the experiment researcher developed the agents and the environment model in the project and selected the appropriate RL method, implemented and designed the metrics of the learning performance of the agents depending on their nature. We demonstrate agents' varied training performances under different sensory modalities, with optimal outcomes observed when combining multiple modalities. Despite these differences, the study underscores agents' adaptability across sensory inputs, advancing our understanding of cross-modal learning in AI. Also, the researcher has shown the invariance of sensory modality results on the learning skills and strategies of an agent and developed simple game like reaching the goal with different input sensory information to test the agent skill and strategies under different perceptual modalities.

Keywords: Reinforcement Learning, Proximal Policy Optimization, Soft Actor-Critic

Acknowledgment

To begin with, I would like to express my gratitude to God for giving me the courage to take on this challenge and perform to the best of my abilities. Secondly, I would like to extend my deepest appreciation to my advisor, **Dr**. **Menore Tekeba**, who has provided invaluable support from the very beginning by providing constructive feedback and corrections. Lastly, I would like to thank all the individuals who directly or indirectly contributed to our project for their help and support.

Contents

Declaration i
Abstract ii
Acknowledgment iii
List of Tables vii
List of Figures viii
Abbreviations ix
Chapter One 1
1. Introduction 1
1.1 Background of the study 1
1.2 Statement of the Problem 2
1.2.1 Research Questions 3
1.3 Research Objective 3
1.3.1 General Objective 3
1.3.2 Specific Objectives 3
1.4 Methodology 3
1.5 Scope 4
1.6 Project Limitation 4
1.7 Document Organization 4
Chapter 2 5
2. Literature Review 5
2.1 Introduction 5
2.2 Automatic Goal Generation in Reinforcement Learning (RL) Agents 5
2.3 Self-Supervised Deep Reinforcement Learning with Generalized Computation Graphs for Robot Navigation 8
2.4 Relational Inductive Biases, Deep Learning, and Graph Networks 8
2.5 Unity: a general platform for intelligent agents 10
2.6 Deep Multi-Sensory Object Category Recognition using Interactive Behavioral Exploration 10
2.7 Making Sense of Vision and Touch: Self-Supervised Learning of Multimodal Representations for Contact-Rich Tasks 11
2.8 Rapid Trial and Error Learning with Simulation for Flexible Tool Use and Physical Reasoning 12
Chapter 3 15
3. Methods and Approaches 15

3.1 Introduction 15
3.2 Reinforcement Learning 15
3.2.1 Categories of Reinforcement Learning 17
3.2.2 How Reinforcement Learning Works 19
3.3 Algorithms and Methods Used 20
3.3.1 PPO (Proximal Policy Optimization) 20
3.3.2 SAC (Soft Actor-Critic) 21
3.3.3 Why SAC and PPO 22
Chapter 4 23
4. Experimental Environment and Setting 23
4.1 Introduction 23
4.2 Environments 23
4.2.1 Food Collector Environment 23
4.2.2 Soccer Twos Environment 26
4.3 ML-Agents Toolkit for Our Sample Game 28
4.4 Sensor Components 29
4.4.1Camera Sensor 29
4.4.2 RayCast Observations 29
4.5 Rewards 31
4.6 Self-play 32
Chapter 5 33
5. Results and Discussions 33
5.1 Introduction 33
5.2.1 Results of Food Collector using PPO 34
5.2.2 Results of Food Collector using SAC 36
5.3 Results of Soccer Twos 38
5.3.1 Results of Soccer Twos Using PPO 38
5.3.2 Results of Soccer Twos Using SAC 39
5.4 Comparisons of the Results 41
5.4.1 Learning Performance Comparison 42
5.5 Answers to Research Questions 45
Chapter 6 50
6. Conclusion and Recommendations 50

6.1 Conclusion 50
6.2 Recommendations 50
References 52

List of Tables

Table 1 food collector with PPO 34
Table 2 Food collector with SAC 36
Table 3 Soccer Twos with PPO 38
Table 4 Soccer Twos with SAC 40
Table 5 Comparison of Food Collector and Soccer Twos 42

List of Figures

Figure 1 Main elements of Reinforcement Learning [10] 17
Figure 2 Methods of Reinforcement Learning algorithms [11] 18
Figure 3 Food Collector Unity ML-Agents environment [16] 25
Figure 4 Soccer Twos Unity ML-Agents environment [17] 27
Figure 5 block diagram of ML-Agents Toolkit for our sample game 28
Figure 6 Camera Sensor Components [19] 29
Figure 7 Ray Sensor components [20] 30
Figure 8 PPO Food Collector graph 35
Figure 9 SAC Food Collector graph 37
Figure 10 PPO Soccer Twos graph 39
Figure 11 SAC Soccer Twos graph 40
Figure 12 PPO ray vs Camera Soccer Twos 43
Figure 13 SAC ray vs. Camera Soccer Twos 43

Abbreviations

PPO - Proximal Policy Optimization

SAC - Soft Actor-Critic

AI - Artificial Intelligence

AR - Augmented Reality

VR – Virtual Reality

RL – Reinforcement Learning

ML – Machine Learning

NN – Neural Network

CDE - Curiosity Driven Exploration

UNREAL – Unsupervised Reinforcement and Auxiliary Learning

UPAT – Unsupervised Predictive Auxiliary Tasks.

HRL – Hierarchical Reinforcement Learning

HER – Hindsight Experience Replay

MORL – Multi-objective Reinforcement Learning

DRL – Deep Reinforcement Learning

CNNs – Convolutional Neural Network

DPG – Deterministic Policy Gradient

Chapter One

1. Introduction

1.1 Background of the study

Artificial Intelligence (AI) Agents are increasingly being used in the real world, interacting with humans across a large variety of tasks. To achieve human-level performance in complex environments, AI Agents must be able to learn from their past experiences and gain both knowledge and an accurate representation of their environment from raw sensor inputs. One approach to creating these agents is to explicitly specify desired tasks and train a reinforcement learning (RL) agent to solve them using different sensory modality.

Reinforcement learning is an important and most usable technique to communicate between an agent and an environment by self-learning. Reinforcement Learning is a technique to understand how an agent can communicate with the environment and find out which action is stylish based on every step by trial and error, solely from rewards or punishments, to develop successful strategies that ultimately lead to the largest long-term rewards [1]. The reward function is the objective feedback from the environment. Rewards are integer or scalar variables associated with some states or state-action pairs. This association (the reward function) defines the goal of the agent in a given situation. For example, a reward of 1 might be associated with the state of having moved all one's pieces off the board in a backgammon game, while a reward of 0 is associated with all other states of the game (all the states leading up to the winning state). The agent's sole objective is to maximize net long-term reward (e.g., in a backgammon playing agent, number of games won). The reward function defines what's objectively good and bad for the agent. The reward function is in commutable by the agent. In traditional cerebral terms, the reward function specifies the states associated with primary reinforcement. Agents need to observe the state of the environment and take actions in this environment. The agent, for each of the actions taken, receives either a reward or penalty using which it learns to maximize the long-term rewards therefore allowing it to be successful in the environment. In this work, we explore the invariance of agent skill and strategies developments from different input sensitive modalities. We propose simple game like reaching an object in which the agent is trained by reinforcement learning to perform a given task by using different sensitive modalities as inputs.

Multimodal sensitive modalities relate to the integration and perception of information from multiple sensitive systems simultaneously. The mortal sensitive system consists of several modalities, including vision, hearing, touch, taste, and smell. Each modality provides unique information about the surrounding environment, and the brain integrates these sensitive inputs to form a coherent perception of the world.

When sensory modalities work together, they enhance the overall perception and understanding of the environment. For an example, when you watch a movie, you not only see the visual scenes but also hear the accompanying sound effects and dialog. The combination of visual and auditory information enriches your experience and helps you understand the story more fully.

Multimodal integration occurs at different levels of the central nervous system, from the early stages of sensory processing to higher-level cognitive processing. In the brain, there are specialized regions and pathways dedicated to processing information from different sensory modalities. These regions interact and exchange information to create a unified perception.

Research has shown that multimodal integration can lead to benefits such as improved attention, enhanced memory encoding, and better decision-making. It also plays a crucial role in daily activities such as speech perception, object recognition, and navigation.

Advancements in technology have allowed for the creation of multimodal interfaces and systems that aim to provide more immersive and interactive experience. Virtual reality (VR) and augmented reality (AR) technologies, for instance, combine visual and auditory cues to create realistic and engaging virtual environments. This integration of multiple sensory modalities enhances the user's sense of presence and immersion.

In summary, multimodal sensory modalities involve the integration and perception of information from multiple senses simultaneously. The brain combines inputs from various sensory systems to create a comprehensive understanding of the world, leading to enhanced perception and cognitive processes.

1.2 Statement of the Problem

Intelligent agents are currently being used to make applications more user-friendly and their usage is growing rapidly. These agents excel in finding and filtering information, customizing it to the user's needs, and automating various tasks. However, when the environment becomes more complex, agents are still a work in progress as they continually adapt to new challenges by devising new strategies.

Humans are a species that can adapt to environmental challenges and over this has enabled us to biologically evolve an essential characteristic found in animals but absent in AI. In artificial agents, there are times in which the agent can’t use one modality of perceptual device. Humans use auditory, touch and other sensory modalities to make navigation inside a completely dark room. Similarly, agents may be prone to some conditions in which they can’t handle some given task or purpose as, for example, when an agent using a camera for doing some task faces dark rooms in which it can’t use the camera anymore.

In this work we designed to test scenarios and test skills of artificial agents using machine intelligence (reinforcement learning) that behaves, learns and develops skills and strategies under different sensory modalities.

1.2.1 Research Questions

- ❖ Do artificial agents demonstrate consistent skill application across different sensory modalities?
- ❖ Are there certain skills or strategies transferrable and equally effective across diverse sensory modalities in artificial agents?
- ❖ What factors influence the invariance or variation of skills and strategies in artificial agents across different sensory modalities?
- ❖ Can artificial agents optimize their performance by adapting developed skills and strategies to different sensory inputs?

1.3 Research Objective

1.3.1 General Objective

The general objective of this research is to examine development of skills and strategies of an agent under different sensory modalities while achieving their goal/task.

1.3.2 Specific Objectives

❖Model the food collector and soccer's twos environment and the agents.

❖ Based on literature, investigate and select appropriate RL method(s) and implement the method.

❖ Design the metrics of learning performance depending on the nature of the agent(s) and the environment.

❖ To investigate whether an agent can develop similar skills and strategies under different sensory modalities.

❖ To investigate the effect of perceptual modalities on the learning performance of the agent.

- ❖ Train the agents to achieve its specified goal or task.
- ❖ Test and evaluate the agent learning performance for its target task.

1.4 Methodology

The following methodologies have been used for the accomplishment of this work.

- ❖ **Literature** Review: This involved examining previous studies and the current state of the art by reviewing related literature.
- ❖ **Modeling the Agents and Environment**: Creating a model for the training environment and the agent was done.
- ❖ **Selecting Learning Policy:** The latest Machine Learning (ML) and Neural Network (NN) techniques were examined to choose the most suitable solution and build a solid theoretical model for the problem.

- **Implementing the Simulation and Training**: The AI agents were created and then trained on the environment**.**
- **Experimenting using Different Modality of Sensors**: -After selecting sensors to train the agents under the different modality of sensors on the environment.
- **Testing and Evaluation: -** Testing the performance of the agent under different sensory modalities.
- **Documentation**: -document the work done and obtained results.

1.5 Scope

The aim of this research is to investigate the invariance skills of agents when using different modalities of sensors under different environment.

1.6 Project Limitation

The aim of this study is limited to investigate the extent to which artificial agents can transfer their learned skills and strategies across two sensory modalities that means camera sensor and ray cast sensors specifically, we will examine whether an agent that has developed certain skills and strategies in a particular sensory modality is able to apply these skills and strategies to a different modality. To accomplish this, we will conduct a series of experiments that involve training agents in one sensory modality and then testing their performance in a different modality.

1.7 Document Organization

This thesis is organized into six chapters. The first chapter introduces the background of the study and the statement of the problem that leads to this study. In addition, the objectives of the study, significance, the scope and methodologies used in the study are presented.

The second chapter deals with the analysis of existing literature on the subject with the objective of revealing contributions, weaknesses and gaps. This chapter discusses the overview of reinforcement learning; discuss algorithms Soft Actor Critic (SAC) and Proximal Policy Optimization (PPO). It covers related works from local and global level.

Similarly, the third chapter presents about methods and approaches of the experiment reinforcement learning, algorithms PPO and Soft Actor critic (SAC).

The fourth chapter, presents the experimental environment food collector and soccer twos with camera sensor and ray cast sensor modality and with Soft Actor Critic (SAC) and Proximal Policy Optimization (PPO) algorithms.

The fifth chapter, which is the experiment and evaluation, describes about the experiments performed and results found using the evaluation techniques. Finally, the last chapter presents the conclusion made after the findings of the experiments and recommendations.

Chapter 2

2. Literature Review

2.1 Introduction

The field of artificial intelligence (AI) has made significant strides in recent years, particularly in developing agents capable of learning and performing tasks across various sensory modalities. Understanding how skills and strategies learned by AI agents translate across different sensory inputs is crucial for advancing the robustness and adaptability of AI systems. This literature review explores current research on testing the invariance of skills and strategies developed by artificial agents under different sensory modalities.

2.2 Automatic Goal Generation in Reinforcement Learning (RL) Agents

Reinforcement learning helps agents to learn a diverse set of tasks without prior knowledge of the environment. This approach involves generating a curriculum of tasks for the agent to learn, with tasks at appropriate difficulty levels. The goal generation can be done using a generator network optimized through adversarial training. By generating goals at the appropriate difficulty level, the agent can learn tasks more efficiently and with improved sample efficiency. This method is particularly useful in multi-goal RL tasks, where agents need to reach multiple goals to solve complex tasks. The generated curriculum of start states or goal states adapts to the agent's performance, leading to efficient training on a goal-oriented task [2].

The purpose of automatic goal generation in reinforcement learning (RL) agents are to facilitate autonomous goal learning and adaptation in RL agents. In reinforcement learning (RL), an agent is typically given predetermined objective to attempt to accomplish through trial-and-error learning. Manual goal specification, however, can be difficult in dynamic and complicated systems, particularly if the goals themselves are dynamic and change over time [2].

The issue of automatic goal generation in RL agents has been tackled in a number of ways. Here are few instances:

Intrinsic Motivation: Using intrinsic motivation processes to direct the creation of goals is one strategy. The agent has curiosity-driven exploration tactics or internal reward signals that motivate it to investigate its surroundings and find new objectives. The agent can learn to design goals that result in engaging and educational experiences by being rewarded for experimenting with new states or attaining unexpected outcomes [2]

Unsupervised Learning: Using unsupervised learning methods like density estimation or clustering, it is possible to identify sub goals or states related to the objective from unprocessed sensory data. Without explicit human intervention, the agent can construct meaningful goals by spotting prominent patterns or structures in the data [2].

Transfer Learning: By utilizing information from similar jobs or contexts, transfer learning techniques can be used to automatically produce goals. An agent can generalize its learned goals across numerous settings or adapt them to new situations by pre-training it on a series of auxiliary tasks and then fine-tuning it on the target task [2].

Hierarchical Reinforcement Learning: Hierarchical RL frameworks are designed to break down difficult tasks into a series of smaller objectives. These methods teach the agent both high-level objectives and low-level steps that must be taken to accomplish those objectives. While the low-level actions are learned to use conventional RL techniques, the high-level goals can be generated automatically based on the decomposition of the main job [2].

These are just few examples of the approaches used for automatic goal generation in RL agents. The field is still actively researched, and new methods and algorithms continue to emerge. The choice of the most suitable approach depends on the specific problem domain and the desired characteristics of the RL agent.

- **Automatic Goal Generation in Reinforcement Learning techniques**

Within the discipline of machine learning, reinforcement learning (RL) focuses on teaching agents how to make decisions sequentially to maximize cumulative rewards. Creating appropriate goals for agents to pursue is a major difficulty in reinforcement learning. The agent's ability to adapt to various activities and situations is limited when goals are manually specified, which can be time-consuming. To overcome this problem, researchers have looked into automatic goal generation strategies that allow RL agents to pick up goals on their own. An overview of the research on autonomous goal generation in RL agents is given in this review of the literature, with a focus on important methodologies, algorithms, and applications.

- **Intrinsic Motivation**

The concept of encouraging RL agents through internal rewards or curiosity-driven exploration is known as intrinsic motivation. By establishing objectives based on their own perceptions of the dynamics of their surroundings, agents are driven to investigate their surroundings in this manner. The Curiosity-Driven Exploration (CDE) technique was introduced by Pathak et al. (ICML 2017). In this strategy, agents provide intrinsic rewards based on prediction errors in their own internal models that they have learned. They showed that a variety of skills can be learned through CDE without the need for clear external rewards [2].

- **Generative Models**

Goals for RL agents have been automatically generated by generative models. The Unsupervised Reinforcement and Auxiliary Learning (UNREAL) framework, which integrates RL with unsupervised learning of generative models, was presented by Pathak et al. (ICML 2016). In order to direct the agent's learning, UNREAL generates goals using generative models and offers

supplementary tasks. With this method, RL agents can pick up a wide range of abilities and adjust to various activities[2].

- **Self-Supervised Learning**

Self-supervised learning automatically generates goals for RL agents by utilizing unsupervised learning techniques. Unsupervised Predictive Auxiliary Tasks (UPAT) is a technique presented by Pathak et al. (ICML 2016), use a prediction-based aim to learn practical representations. The auxiliary tasks entail making predictions about the environment's future states or characteristics, which might act as objectives for the RL agent. The agent gains valuable representations through its ability to anticipate these objectives, which helps with effective reinforcement learning [2].

- **Hierarchical Reinforcement Learning**

A hierarchy of sub-goals is created by breaking down difficult tasks using hierarchical reinforcement learning (HRL). Numerous HRL algorithms produce sub-goals on their own by using unsupervised learning or the environment's structure. The Hindsight Experience Replay HRL algorithm was presented by Pathak et al. (ICML 2016). It creates sub-goals in the past based on good results. By reinterpreting failed attempts as successful endeavors toward alternative goals, HRL helps RL agents learn from them [2].

- **Multi-Objective Reinforcement Learning**

The purpose of multi-objective reinforcement learning, or MORL, is to simultaneously optimize RL agents toward many goals. For the agents to accomplish a trade-off between several objectives, MORL algorithms automatically construct a set of diversified goals. A thorough analysis of MORL algorithms, encompassing methods like evolutionary algorithms, Pareto-based strategies, and value function approximation, was provided by Pathak et al. (ICML 2016). A versatile foundation for autonomous goal generation in RL agents is offered by MORL [2].

A promising field of research that tackles the problem of creating appropriate goals for agents to pursue is automatic goal generation in reinforcement learning agents. The following strategies have been investigated: hierarchical reinforcement learning, multi-objective reinforcement learning, self-supervised learning, generative models, and intrinsic motivation. These methods allow RL agents to independently explore their surroundings, acquire a variety of skills, and adjust to various activities. The development of more resilient algorithms that can manage complicated settings and perform well across various tasks and domains is key to the future of autonomous goal formation in RL agents. [2]

2.3 Self-Supervised Deep Reinforcement Learning with Generalized Computation Graphs for Robot Navigation

Robot navigation is a difficult topic that calls for intelligent agents to investigate and navigate uncharted territory on their own. Deep reinforcement learning, or DRL, has become a method for teaching robots how to navigate through intricate situations. Since they allow agents to learn from unlabeled data and adapt their knowledge to new settings, self-supervised learning approaches coupled with generalized computation graphs have garnered attention in recent years. The state-of-the-art research on self-supervised DRL with generalized computation graphs for robot navigation is examined in this literature review, which also highlights important contributions, approaches, and future possibilities [3].

❖ **Overview of Robot Navigation Challenges and the Need for Intelligent Agents**

For real-world deployment, robots must be able to navigate complex surroundings on their own. Previous approaches to this challenge involve the robot maintaining an internal map of the environment, which is subsequently navigated through using a localization and planning technique. These methods, however, are computationally demanding, frequently contain a wide range of assumptions, and do not learn from mistakes. On the other hand, learning-based techniques become more effective as the robot interacts with its surroundings, but their large sample complexity makes them challenging to implement in practical settings [3].

❖ **Introduction to Deep Reinforcement Learning and its Applications in Navigation**

Robots that are capable of independently navigating intricate and unstructured environments, such as streets, buildings, or woods, require generalizable perception and control systems that can make decisions about how to travel [3].

❖ **Implications for autonomous systems, industrial automation, and mobile robotics**

Reinforcement learning is characterized by autonomous learning through trial and error. While model-based methods are often more sample-efficient, they struggle with complicated surroundings and high-bandwidth sensors like cameras. Meanwhile, model-free reinforcement learning methods have been able to learn complex tasks, but they are normally less sample-efficient. Although robot navigation policies have been learned via these techniques, they frequently call for simulation experience. On the other hand, our method learns to navigate exclusively in the real environment using monocular images [3].

2.4Relational Inductive Biases, Deep Learning, and Graph Networks

This study of the literature investigates the connection between deep learning, graph networks, and relational inductive biases. It gives a summary of the key ideas, talks about current developments, and emphasizes the difficulties and possible applications in this multidisciplinary topic. The goal of the review is to give scholars and professionals a thorough grasp of the major concepts and advancements in this field.

Its ability to facilitate amazing advances in speech recognition, image recognition, and natural language processing, deep learning has completely changed a number of fields. However, relational data, which is common in many real-world applications, is difficult for typical deep learning models to handle well. This review of the literature looks at how these three domains overlap and highlights the potential for their synergy [4].

Relational Inductive Biases: These biases come from preexisting information or presumptions that are incorporated into a learning system and take use of the data's relational nature. An overview of the various kinds of relational inductive biases, including structural biases, relational reasoning, and priors based on graphs, is given in this section. It talks about how crucial they are for capturing relationships and dependencies between entities in relational data, as well as how they help deep learning models become more generalizable [4].

Deep Learning: emphasizing its structure, methodology, and principal obstacles. It talks about how typical deep learning models are not as good at managing relational data since they don't explicitly model relationships and can't capture structural connections. Additionally, representation learning is discussed in this section along with its importance in deep learning models [4].

Graph Networks: As a potent paradigm for managing relational data, graph networks have attracted a lot of interest. An extensive examination of graph networks, their architecture, and essential elements is given in this section. It talks about how graph networks allow relationships to be explicitly modeled, structural dependencies to be captured, and computation and reasoning to be done on graphs [4].

Incorporating Relational Inductive Biases into Deep Learning: The research efforts to include relational inductive biases into deep learning systems are the main topic of this section. Especially by utilizing graph networks techniques including graph recurrent networks, graph attention networks, and graph convolutional networks are covered. This section delves into the ways in which these methodologies utilize relational inductive biases to enhance the efficacy of deep learning models when applied to relational data problems [4].

Applications and Challenges: Relational inductive biases and graph networks have the potential to be used in a multitude of fields, such as drug discovery, social network analysis, recommendation systems, and knowledge representation. In this section, successful applications of graph networks in several disciplines are highlighted through case studies. It also addresses the difficulties with scalability, interpretability, and data that come with integrating relational inductive biases into deep learning models [4].

The paper provided in this review of the literature it highlights how relational inductive biases are crucial for improving the handling of relational data by deep learning models. It also emphasizes how effective a framework graph networks may be for incorporating these biases. A request for more investigation and research in this multidisciplinary topic. Through an analysis of the convergence of deep learning, graph networks, and relational inductive biases, this literature review offers a thorough grasp of the state-of-the-art and associated problems.

2.5 Unity: a general platform for intelligent agents

The usage of Unity as a broad platform for intelligent agents is examined in this survey of the literature. Unity is a well-known game production engine that has drawn a lot of interest from artificial intelligence (AI) researchers because of its flexible features and intuitive user interface. An overview of the research, techniques, and applications that have been done so far using Unity as a platform for creating and implementing intelligent agents is given in this article. The review addresses possible future possibilities in the realm of AI research and outlines Unity's advantages and disadvantages [5].

Unity, a versatile platform designed to facilitate the development, training, and evaluation of intelligent agents. Unity provides a flexible and scalable environment that supports a wide range of tasks and simulations, making it an ideal choice for both academic research and commercial applications in artificial intelligence (AI).

The platform is built with several key features:

1 .Interoperability: Unity supports integration with various machine learning frameworks, enabling seamless training of agents using state-of-the-art algorithms.

2. Customizability: Users can create and customize complex environments and scenarios, tailoring them to specific research or application needs.

3. Scalability: Unity's robust infrastructure supports large-scale experiments, allowing researchers to efficiently manage computational resources and accelerate development cycles.

4. Visual Realism: The platform offers high-quality graphics, providing realistic simulations that enhance the training of vision-based agents.

Overall, Unity emerges as a powerful tool for advancing the field of AI, promoting innovation, and fostering collaboration across different research communities. This paper aims to encourage the adoption of Unity as a standard platform for developing intelligent agents, driving forward the capabilities and applications of AI technologies

2.6 Deep Multi-Sensory Object Category Recognition using Interactive Behavioral Exploration

The field of deep multi-sensory object category recognition by interactive behavioral exploration is examined in this survey of the literature. The aim is to present a thorough synopsis of the research carried out in this domain, addressing the principal approaches, contributions, obstacles, and prospective paths. Behavioral exploration methods and deep learning techniques to improve object identification systems through the integration of numerous sensory modalities [6].

An overview of the significance of object category recognition and the function of multisensory integration in human perception is provided in the introduction section. It emphasizes the rationale

behind enhancing the precision and resilience of object recognition systems through the use of deep learning and interactive behavioral exploration. [6]

Deep Learning for Object Recognition: In this part, convolutional neural networks (CNNs) and other deep learning methods are briefly introduced, along with their use in object recognition tasks. The benefits of using deep learning models to acquire intricate hierarchical representations from unprocessed sensory data are covered in [6].

- **Multi-Sensory Integration in Object Recognition:** The topic of discussion now turns to the idea of multi-sensory integration and how important it is for object recognition. It investigates how combining information from the senses—visual, aural, tactile, and others—can improve the precision and effectiveness of item category recognition [6].
- **Interactive Behavioral Exploration:** The concept of interactive behavioral exploration is explored in this section. It entails actively investigating the environment to obtain pertinent sensory information. It highlights the contributions of active vision, active touch, and active audition, among other interactive exploration strategies, to enhancing object recognition performance [6].
- **Deep Multi-Sensory Object Recognition Approaches:** interactive behavioral exploration and deep learning for multi-sensory object category detection is provided in this section. It goes over the main techniques, datasets, performance indicators, and findings of these investigations. The section also discusses methods that use deep neural networks to combine data from several sensory modalities [6].
- **Challenges and Future Directions:** primary obstacles and constraints in the field of deep multi-sensory object recognition through interactive behavioral exploration, as well as potential future directions. It tackles problems like scalability, real-time processing, model training, and data collecting. A summary of prospective future directions and research opportunities in this field is provided at the end of the section [6].

  The important conclusions and contributions of the researched studies are compiled in the literature review's conclusion. In addition to highlighting the need for more research in this area, it demonstrates the promise of deep multi-sensory object category classification employing interactive behavioral exploration to advance the object recognition field [6]. This study offers important insights into the present status of the field by completing an extensive literature review on deep multi-sensory object category recognition utilizing interactive behavioral exploration. It seeks to motivate researchers to investigate new strategies and techniques for enhancing object recognition systems through the use of interactive exploration methods and different sensory modalities.

2.7 Making Sense of Vision and Touch: Self-Supervised Learning of Multimodal Representations for Contact-Rich Tasks

With an emphasis on vision and touch in particular, this literature review attempts to investigate the idea of self-supervised learning for multimodal representations in the context of contact-rich tasks. The state-of-the-art approaches and techniques used in self-supervised learning , along with their potential benefits for contact-rich tasks and applications [7].

❖ **Briefly introduces the importance of multimodal representations in contact-rich tasks**

It provides an overview of self-supervised learning as a powerful approach for learning representations. Outlines the specific focus of the review on vision and touch modalities.

- ❖ **Multimodal Representations:** Explores the advantages and disadvantages of merging visual and touch modalities, as well as their definition and importance in comprehending contact-rich activities. It explains many ways to representation learning for multimodal data, such as self-supervised, supervised, and unsupervised techniques. [7]
- ❖ **Self-Supervised Learning:** This section offers a comprehensive overview of self-supervised learning, outlining its benefits and covering common methods like generative models, contrastive learning, and predictive coding. It emphasizes how self-supervised learning may be used to multimodal data and how it can be used to learn meaningful representations.
- ❖ **Vision and Touch Fusion:** The article delves into the significance of combining vision and touch modalities in tasks that require a lot of contact. It also covers several fusion strategies, including early fusion, late fusion, and cross-modal attention mechanisms. further provides tests and case studies showing how well multimodal fusion is for tasks requiring a lot of interaction [7].
- ❖ **Contact-Rich Task Applications:** Examples of contact-rich tasks that can benefit from self-supervised learning of multimodal representations are given in the section on contact-rich task applications. It addresses robotics applications like manipulating objects, feeling about, and interacting physically with the environment. It also offers studies and real-world use cases that highlight the benefits of self-supervised multimodal representations in scenarios with plenty of contact [7].
- ❖ **Challenges and Future Directions:** The present study highlights the obstacles and potential avenues for improvement in self-supervised learning for multimodal representations. It talks about future research directions and possible fixes for these problems. highlights new developments in the discipline, including deep reinforcement learning and continuous learning, as well as emerging trends and technology. This highlights how crucial self-supervised learning is for multimodal representations in activities with plenty of touch. makes concluding observations and recommends possible directions for further study. This review attempts to provide a thorough knowledge of self-supervised learning strategies for multimodal representations in contact-rich tasks by examining the current literature. It draws attention to how these methods may make it possible for intelligent robots to successfully comprehend and engage with the real environment. Researchers and practitioners interested in the convergence of contact-rich tasks, multimodal representations, and self-supervised learning might use the review's conclusions as a starting point.

2.8 Rapid Trial and Error Learning with Simulation for Flexible Tool Use and Physical Reasoning

Simulation and quick trial and error learning to improve physical reasoning and flexible tool use across a range of fields. The review summarizes pertinent works and focuses on important

conclusions, approaches, and ramifications in this developing topic. In order to further our understanding of flexible tool use and physical reasoning, it is intended to present a thorough overview of the state of the art in rapid trial and error learning with simulation [8].

Fundamental cognitive skills that allow people and intelligent agents to effectively interact with their environment are flexible tool use and physical reasoning. Combining simulation methods with quick trial and error learning has become a viable way to improve these skills. The research that has already been done in this field is examined in this review of the literature, with an emphasis on the approaches, experimental designs, and findings of those investigations [8].

- **Rapid Trial and Error Learning:** This section highlights the importance of rapid trial and error learning in fostering adaptive behavior and gives an outline of it. It examines the idea of the trade-off between exploration and exploitation, emphasizing how effective decision-making and problem-solving can result from learning via repeated experimentation. Important methods and theoretical frameworks for quick trial-and-error learning are covered [8].
- **Simulation in Cognitive Science:** In cognitive research, simulation is extensively employed to examine intricate phenomena that are challenging to explore via direct investigation. The function of simulation in comprehending the usage of flexible tools and physical thinking is examined in this section. It addresses many simulation approaches and their uses in cognitive science research, including agent-based modeling and physics-based simulation [8].
- **Flexible Tool Use:** To explore flexible tool use, the review explores research that combine simulation and quick trial-and-error learning. It looks at the ways that simulation-based investigations have shed light on how to choose tools, how to acquire tool manipulation skills, and how feedback works to improve tool use habits. The ramifications of these discoveries for practical uses like robotics and human-machine interaction are also covered in this section [8].
- **Physical Reasoning:** This is the capacity to comprehend and forecast the actions of physical systems. This section summarizes studies that use simulation and quick trial-and-error learning to study the capacity for physical thinking. It examines the ways in which simulations have advanced our knowledge of causal reasoning, intuitive physics, and the creation of mental models for physical systems [8].

A discussion of the difficulties and possible future paths in the area of quick trial-and-error learning with simulation for adaptable tool use and physical reasoning rounds off the literature overview. In order to improve our understanding of these cognitive capacities, it emphasizes the necessity for interdisciplinary cooperation and more study to overcome the shortcomings of current approaches [8].

The review highlights the promise of quick trial and error learning with simulation in promoting flexible tool use and physical reasoning, and it summarizes the main conclusions from the research. It emphasizes the significance of carrying out more study in this area and the possible influence on disciplines like cognitive science, robotics, and artificial intelligence.

This literature review offers a thorough comprehension of the current state of knowledge in rapid trial and error learning with simulation for flexible tool use and physical reasoning by synthesizing

prior research. It also emphasizes the possible ramifications of this finding for cognitive science and allied domains, laying the foundation for further investigation.

Chapter 3

3. Methods and Approaches

3.1Introduction

This chapter offers objective s of all the different techniques and strategies used in the experiments. The researcher used a variety of methods, such as Reinforcement Learning, a kind of machine learning that makes use of trial and error to determine the best course of action in a certain circumstance. Two particular algorithms were also applied: the Soft Actor-Critic (SAC) method and the Proximal Policy Optimization (PPO) algorithm. An online optimization method without a model that works well in high-dimensional action spaces is the PPO algorithm. Conversely, the SAC algorithm employs an ensemble of neural networks as a model-based method to learn a stochastic policy. Combining these methods allowed the researcher to provide solid and trustworthy results.

3.2Reinforcement Learning

A subfield of machine learning called reinforcement learning studies how an intelligent agent can be trained to make choices and behave in a way that maximizes a numerical reward signal. The method draws inspiration from the way animals and humans both learn by making mistakes [9].

A subfield of machine learning called reinforcement learning studies how an intelligent agent can be trained to make choices and behave in a way that maximizes a numerical reward signal. The method draws inspiration from the way animals and humans both learn by making mistakes [9].

An agent engages with an environment and gets feedback in the form of incentives or punishments in reinforcement learning. Learning a policy, or mapping states to actions that maximizes the cumulative reward over time, is the agent's objective. The agent employs exploration and exploitation tactics to strike a balance between trying new things and learning from the past in order to accomplish. While the exploitation strategy enables the agent to make use of its present knowledge to pursue activities that are likely to result in higher rewards in the near term, the exploration strategy enables the agent to find new states and actions that may lead to higher rewards in the future [9].

The predicted reward for every state-action pair is estimated by the agent using a value function, and it is modified based on input from the environment. Additionally, the agent chooses actions depending on the estimated values using a policy function. [9]

By taking action and modifying its value and policy functions in response to input from the environment, the agent learns by making mistakes. The agent's value and policy functions improve with each iteration, enabling it to make wiser choices and execute the best course of action [9]. Main elements of reinforcement learning listed below

- **Agent**: An organism that senses its surroundings and responds accordingly. The agent wants to maximize a signal that the environment provides as a reward.
- **Environment**: The environment refers to the external system or structure that the agent functions within. It can be a simulated environment, like a computer game, or a real-world one, like a robot navigating a physical space.
- **State**: The state of affairs or state of the surroundings at a specific moment in time. A collection of variables or features that record pertinent data for the agent's decision-making process is how the state is represented.
- **Action**: The decisions or choices the agent takes in reaction to a particular state. Discrete decisions, like choosing a move in a game, and continuous control, like modifying motor outputs, are examples of actions.
- **Reward**: Following each action, the environment sends the agent a scalar feedback signal. As a foundation for learning, the reward conveys the desirability or quality of the agent's activities.
- **Policy**: The approach or guideline that the agent uses to decide what to do given the circumstances at hand. The policy might be either stochastic or deterministic, mapping states to actions.
- **Value Function**:
  The Value Function is a tool used to calculate the expected value or long-term return of adhering to a certain policy and state. It aids in the agent's assessment and comparison of various conditions or behaviors.
- **Model**: In certain reinforcement learning techniques, the agent might make use of an environment model that encapsulates the environment's dynamics and can be applied to future state and reward simulations or planning.

There is a cycle in the interactions between these components. After observing the current state and making a decision based on its policy, the agent moves on to the next state and gets rewarded by the environment. Based on these interactions, the RL algorithm modifies the agent's policy and value function in an effort to enhance the agent's decision-making capabilities over time. [10]

Figure 1 Main elements of Reinforcement Learning [10]

3.2.1 Categories of Reinforcement Learning

Model-free and model-based algorithms are two general categories for RL algorithms. Model-free algorithms do not create an explicit model of the MDP or, to put it more precisely, the environment. They are more akin to trial-and-error algorithms, which use actions to conduct environmental experiments and then immediately determine the best course of action [11].

Model-Free Algorithms can be classified as policy-based or value-based. Value-based algorithms believe that an accurate estimation of each state's value function leads directly to the optimal policy. The agent interacts with the environment to sample the trajectories of states and rewards through the use of a recursive relation that is characterized by the Bellman equation. It is possible to predict the MDP's value function given enough trajectories. Finding the best course of action after the value function is established only requires acting avariciously in relation to the value function at each stage of the procedure. SARSA and Q-learning are a couple of well-known value-based algorithms [11].

Conversely, policy-based algorithms do not need to model the value function; instead, they estimate the optimal policy directly. They transform the learning problem into an explicit optimization problem by directly parameterizing the policy using learnable weights. The agent samples state and reward trajectories similarly to value-based algorithms, but it uses this

knowledge to explicitly improve the policy by maximizing the average value function over all states. Deterministic policy gradient (DPG) and Monte Carlo policy gradient (REINFORCE) are two well-known policy-based reinforcement learning techniques. The significant variation of policy-based techniques causes instability during the training process. Even if they are more robust, value-based methods are not appropriate for modeling continuous action spaces. Value-based and policy-based techniques are combined to create the actor-critic algorithm, one of the most potent reinforcement learning systems. To enable efficient use of training data with consistent convergence, this algorithm parameterizes both the policy (actor) and the value function (critic) [11].

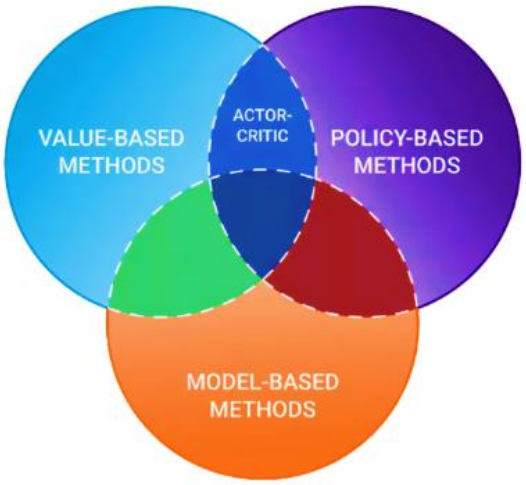

Figure 2 Methods of Reinforcement Learning algorithms [11]

Algorithms for reinforcement learning (RL) have gained popularity as a method for teaching agents to carry out tasks in intricate settings. It has been demonstrated that the model-based method, out of all the RL strategies, works well in numerous cases [11].

Through state sampling, action, and reward observation, model-based reinforcement learning algorithms construct a model of the surrounding environment. The agent can forecast the expected reward and the predicted future state for each state and potential course of action by building an environment model. As a result, the agent may plan its behaviors without having to deal with the environment directly [11].

Two different kinds of predictions are used in the process of creating an environment model: reward and state predictions. A regression problem is reward prediction, in which the algorithm forecasts the predicted reward based on a particular action and state. State prediction is a density estimation problem in which, given the present state and an action, the algorithm predicts the probability distribution over potential future states [11].

The RL agent can plan its activities without interacting with the environment directly by using a model of the environment. This is comparable to a problem-solving thought exercise that a person may use. Without really executing the various acts in the real world, the agent can simulate them and assess the predicted results. Agent's learning capacity is increased when the planning and policy estimation processes are combined can update both policy. The agent can create fictitious trajectories using the model, and then adjust the policy according to the predicted return of those trajectories. Model-based reinforcement learning algorithms are useful for learning how to carry out tasks in intricate settings [11].

3.2.2 How Reinforcement Learning Works

To choose the best course of action, reinforcement learning algorithms use a trial-and-error method. They employ a variety of methods, including exploration-exploitation tactics and value functions, which calculate projected future rewards by weighing the pros and cons of attempting new things and taking advantage of well-researched behaviors. Q-learning, policy gradient techniques, and deep reinforcement learning with neural networks are examples of common reinforcement learning algorithms [12].

Applications for reinforcement learning can be found in many different fields, such as resource management, autonomous cars, recommendation systems, robotics, and gaming. Reinforcement learning offers the potential to solve difficult issues in dynamic and uncertain situations by empowering agents to learn from experience and enhance their decision-making skills [12].

An agent and its surroundings interact iteratively in order for reinforcement learning to take place. The agent gains the ability to make choices and act in the environment by utilizing feedback in the form of incentives [12].

- **Initialize**: The agent starts by initializing its policy, which is the mapping from states to actions. The policy could be random or based on some initial knowledge.
- **Observe state:** The agent receives an observation or state from the environment, which represents the current state of the system.
- **Choose action:** Based on the observed state, the agent selects an action to take. This action is determined by the agent's policy, which can be deterministic or probabilistic.
- **Take action:** The agent executes the chosen action in the environment.
- **Receive reward and new state:** After taking the action, the agent receives a reward signal from the environment. The reward indicates the desirability of the action taken and serves as feedback for the agent's decision-making.
- **Update policy:** The agent updates its policy based on the received reward. The policy update is aimed at improving the agent's future decision-making ability.
- **Repeat:** Steps 2 to 6 are repeated for each time step or iteration, allowing the agent to continue interacting with the environment and refining its policy over time.

An AI agent is a program designed to make autonomous decisions based on an environment's feedback. The agent's primary objective is to maximize its cumulative reward over the course of its interactions with the environment. To achieve this, the agent often employs exploration and exploitation strategies. Exploration involves trying new actions to gather information about the

environment, while exploitation involves choosing actions that are known to yield high rewards based on the agent's current policy [12].

Reinforcement learning algorithms use various techniques to update the agent's policy. These techniques can include value-based methods, such as Q-learning, which estimate the expected future rewards of taking certain actions in specific states. Policy-based methods, such as policy gradients, directly optimize the policy parameters based on the received rewards. More advanced approaches, like deep reinforcement learning, combine reinforcement learning with deep neural networks to handle complex and high-dimensional state spaces [12].

Through this iterative process of interaction, observation, action, and reward, the agent gradually learns to make better decisions and improves its policy over time to maximize the expected cumulative reward in the given environment. As the agent continues to interact with the environment, it forms a model of the environment that it can use to make better decisions. The agent learns which actions result in the most significant rewards and which actions should be avoided [12].

The agent's reward signal is crucial in determining the quality of its decisions. The reward signal is a scalar value that the agent receives from the environment, indicating how well it is performing. The agent's goal is to maximize this reward signal over the long run. The agent's success is measured by how much reward it accumulates over time. The agent's performance is evaluated based on how much reward it receives over a specified number of interactions with the environment [12].

In conclusion, AI agents use reinforcement learning algorithms to interact with their environment, learn from the feedback, and make better decisions based on the accumulated knowledge. The agent's primary objective is to maximize the cumulative reward it receives, and to achieve this, the agent employs exploration and exploitation strategies, updates its policy using various techniques, and gradually learns to make better decisions. [12].

3.3 Algorithms and Methods Used

Evaluating artificial agents' learning abilities in response to various sensory modalities is the main goal of this study. The objective is to use a robust and effective reinforcement learning (RL) technique to explore the invariance of the agent's abilities and learning performance under different sensory input situations. The PPO and SAC algorithms were chosen for our study due to their high performance and computational efficiency.

3.3.1 PPO (Proximal Policy Optimization)

Through trial and error, Proximal Policy Optimization (PPO), a reinforcement learning algorithm, teaches agents how to carry out tasks in various settings. This algorithm belongs to the family of policy gradient algorithms, which optimizes the parameters of the policy function directly. An agent's decision-making approach or behavior is represented by the policy function [13].

By iteratively adjusting the policy's parameters in response to the experiences gathered from interactions with the environment, PPO seeks to identify the best course of action. Using a surrogate objective function to approximate the genuine objective function is one of its main characteristics. More steady updates are possible since the surrogate objective compares the current and old policies. PPO prevents significant policy changes that could cause instability by employing a clipping technique to limit the amount of policy change during each update [13].

PPO further stabilizes the learning process by employing numerous optimization epochs. More effective and efficient policy updates are made possible by sampling and using the gathered experiences from each epoch to change the policy several times. This aids PPO in striking a balance between taking advantage of the existing policy and investigating novel approaches [13].

All things considered; PPO is renowned for its capacity to attain strong sample efficiency while remaining resilient to environmental fluctuations. Numerous applications, such as gaming, simulated environments, and robotic control, have had success with its use. PPO is a useful technique for teaching agents to carry out tasks precisely and effectively in many situations because of its capacity to optimize policies while remaining stable and robust [13].

3.3.2 SAC (Soft Actor-Critic)

Soft Actor-Critic, or SAC for short, is a reinforcement learning system that is applied to agent training in continuous action space environments. Combining the benefits of value estimation and policy optimization, it is an extension of the actor-critic framework [14].

In SAC, the "critic" assesses the worth or caliber of the acts the "actor" takes, while the "actor" chooses actions based on the status of the environment at the time. Neural networks are usually used to implement the actor and critic [14].

An essential component of SAC is the application of entropy maximization. SAC promotes exploration and keeps the agent from quickly converging to a poor policy by optimizing the entropy of the policy. The agent learns more resilient policies and discovers a wider variety of actions as a result of this exploration [14].

Along with previous actor-critic methods, SAC additionally adds a soft value function update that minimizes the squared Bellman error plus an additional entropy component. More exploration results from the policy's high entropy, which is encouraged by the entropy regularization term [14].

The overall goal of the SAC algorithm is to maximize entropy while optimizing a trade-off between maximizing expected benefits. In a continuous action space, this enables the agent to discover various and effective policies. Games, control tasks, robotics, and other disciplines have all demonstrated the effectiveness of SAC [14].

3.3.3 Why SAC and PPO

Our goal is to show that artificial agents' skills and learning remain the same across camera sensor and ray cast sensor modalities. The evaluation approach we have chosen to achieve the best level of efficiency and accuracy is reinforcement learning. we have selected Soft Actor-Critic (SAC) and Proximal Policy Optimization (PPO) as our favored reinforcement learning techniques because they provide the best efficiency and performance. demonstrate how the agent's abilities and learning performance are fundamentally invariant across sensory modalities. The evolution of intelligent systems depends on artificial agents' capacity to learn from and adjust to camera sensor and ray cast sensor modalities [17]. We are confident that our research will provide valuable insights into the invariance of skills and learning of artificial agents.

Chapter 4

4. Experimental Environment and Setting

4.1 Introduction

The investigator in this study used two distinct surroundings and a variety of sensors in an all-encompassing manner. The food collector and soccer twos settings were the two that were utilized since they are known to be pertinent to the subject of the study. The researcher also used ray-cast sensors and cameras, which are cutting edge technologies that have shown to be successful in gathering data for a variety of investigations. These sensors allowed the researcher to collect precise and trustworthy data, which was essential to the experiment's success and validity. Overall, the research's findings and conclusions benefited greatly from the researcher's methodical approach and effective use of cutting-edge technologies.

4.2 Environments

An open-source project called Unity Machine Learning Agents Toolkit (ML-Agents) makes it possible to train intelligent agents in simulations and games. With the aid of this toolkit's PyTorch-based algorithms, enthusiasts and game developers can easily train intelligent agents for 2D, 3D, and VR/AR games. Researchers can train agents using reinforcement learning, imitation learning, neuroevolution, or any other technique thanks to the user-friendly Python API that is supplied [15].

Because it offers a central platform where AI improvements can be assessed on Unity's rich surroundings, ML-Agents is advantageous for both AI researchers and game producers. With the toolkit, developers may train agents to manipulate the actions of NPCs in a range of scenarios, such as adversarial and multi-agent situations. Moreover, the trained agents can be applied to automated testing of game builds and pre-release assessment of various game design choices [15].

There are numerous uses for the ML-Agents Toolkit for both AI researchers and game creators. It enables game designers to produce more complex and wise virtual characters, giving gamers a more engaging experience. It also gives academics the chance to test and experiment with novel AI algorithms in a vibrant and dynamic setting. In the end, ML-Agents effective tool that can hasten the development of artificial intelligence, and anyone interested in AI and game creation can benefit greatly from its accessibility to the larger academic and game developer communities [15].

4.2.1 Food Collector Environment

A sophisticated simulation, the Food Collector Unity ML-Agents environment was created to offer intelligent agents in the food collecting industry a thorough training and testing platform. This environment was created with Unity's ML-Agents framework, which provides an extremely

lifelike environment in which agents can become proficient at navigating and gathering food items [16].

In the multi-agent environment, agents fight with each other to collect green food and avoid red food. The objective of the Food Collector setting is to collect as much food as you can in the given amount of time. Agents compete with one another in the multi-agent environment to gather green food and stay away from red food. In the Food Collector setting, the goal is to gather as much food as you can in the allotted time. This setting is made up of a grid-based universe that is home to a variety of barriers, food items, and agents [16].

The agents are given a tough and realistic experience in this highly dynamic setting. In order to travel around the environment and decide which food items to collect and which barriers to avoid, the agents must use their intellect [16].

All things considered, the Food Collector Unity ML-Agents environment offers a thorough platform for developing and evaluating intelligent agents related to food collection. The environment is incredibly interactive, demanding, and realistic, which makes it a great resource for academics and developers interested in creating intelligent agents [16].

Every agent in the environment possesses a collection of sensors that offer data about the surroundings. These sensors may provide perceptual information such as proximity sensors to identify items in the vicinity or visual input like a camera image. When deciding what to do, such as approaching a food item or avoiding a barrier, agents can use this knowledge [16].

A range of discrete or continuous activities that the agents are capable of doing describe their actions. These could be walking forward, making left or right turns, or interacting in a certain way with food items. The agents' objective is to figure out the best course of action that will maximize the total number of food items gathered in the shortest length of time [16].

During training, agents can approximate the ideal food collecting policy by using a variety of machine learning approaches, including deep neural networks. These neural networks are able to generate a set of actions based on learnt patterns after receiving the agent's sensory input as input characteristics. The agents' policies are improved by training, which enables them to gather food items more effectively [16].

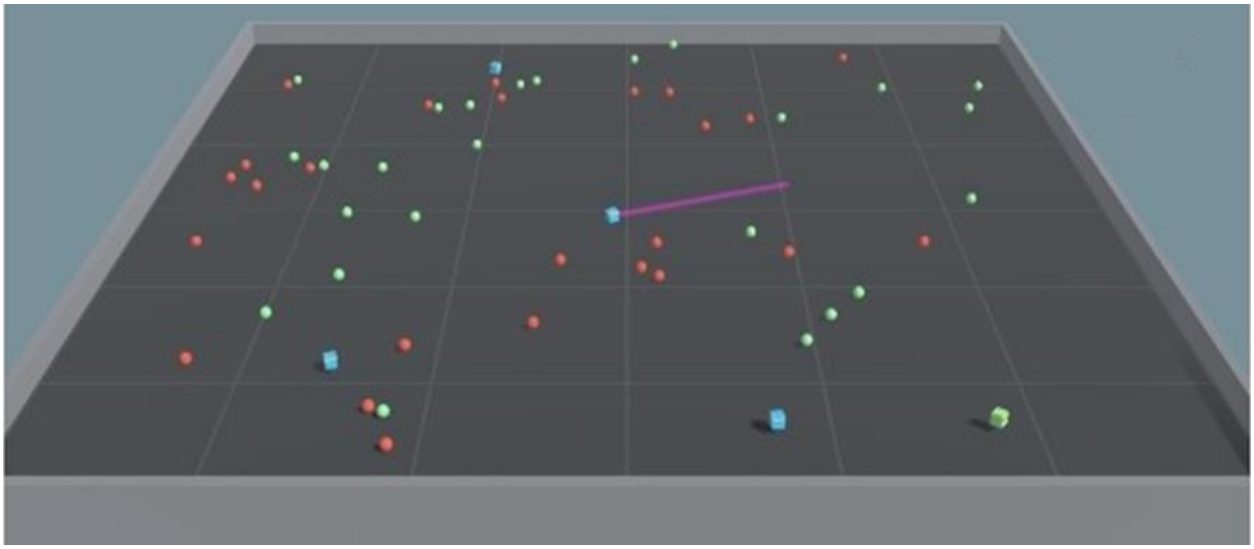

Figure 1 Food Collector Unity ML-Agents environment [16]

- **Set-up:** An environment with multiple agents where agents compete to collect food.
- **Goal:** The objective is for the agents to gather as many green food spheres as they can without coming into contact with the red ones.
- **Agents:** Five agents present in the environment. Independent Agent Reward Function: When two spheres interact, the result is +1 for green spheres and -1 for red spheres.
- **Behavior Parameters:** Vector Observation space: 53 corresponding to velocity of agent (2), whether agent is frozen and/or shot its laser (2), plus grid based perception of objects around agent's forward direction (40 by 40 with 6 different categories).
- **Actions:** Three consecutive actions represent rotation, side motion, and forward motion. There is only one distinct action branch for the laser, and there are two possible outcomes: Shoot Laser or Take No Action.
- **Visual Observations (Optional):** One vector flag signifying the agent's frozen state, plus one first-person camera for each agent. The training of this scene requires both visual and vector observations, and thus cannot be completed without the frozen vector flag. The VisualFoodCollector scenario has been used.
- **Float Properties**: Two
- **laser_length**:The agent's laser's length Standard: 1; Suggested Minimum: 0.2; Suggested Maximum: 7
- **Agent scale:** Denotes the three-dimensional (same in all three dimensions) scale of the agent. By default: 1 Recommended Benchmark At least: 0.5 Suggested Upper Limit: 5 Payout Average: 10

4.2.2 Soccer Twos Environment

Four agents engage in a 2 vs 2 toy soccer match in Soccer Twos Unity ML-Agents Environment: Reinventing Soccer with Machine Learning Environment.

A ground-breaking development in the fields of machine learning and soccer simulation is the Soccer Twos Unity ML-Agents environment. This setting creates a dynamic and immersive experience by fusing the capabilities of Unity's ML-Agents framework with the excitement and strategic components of soccer. Training intelligent virtual soccer players and investigating the intricacies of team dynamics are made possible by the Soccer Twos Unity ML-Agents environment, which makes use of artificial intelligence and reinforcement learning techniques. [17]

Fundamentally, the Soccer Twos Unity ML-Agents environment offers a simulated soccer match setting in which two teams, each with two players, engage in competition. AI agents that can be trained using different machine learning techniques, such as reinforcement learning, are in charge of controlling the players. To create a realistic and difficult soccer experience, the environment offers a wide range of elements and interactions, including dribbling, passing, shooting, defending, and team collaboration [17].

The Soccer Twos Unity ML-Agents environment's capacity to make intelligent agent training easier is one of its main advantages. The AI agents can employ reinforcement learning techniques to learn from their interactions with the environment and gradually enhance their decision-making abilities. It is possible to teach agents to modify their tactics in response to changes in the game, player positions, and team dynamics. This creates opportunities to experiment with various game strategies, team configurations, and playing styles [17].

Moreover, it is simple to customize and experiment with the Soccer Twos Unity ML-Agents environment. To create their own scenarios and challenges, users can alter a variety of characteristics of the setting, including the size of the field, the ball's mechanics, and the actions of the AI agents. Because of its adaptability, researchers and developers may create experiments, test theories, and assess how well various AI systems operate in a soccer setting [17].

Strong visualization and analysis tools are also made available by the Soccer Twos environment's integration of the Unity ML-Agents framework. In real time, users can watch the gameplay and keep an eye on how well particular agents or teams are performing. This makes it possible to analyze agent behavior in-depth, spot flaws, and improve training methods. Furthermore, the Unity ML-Agents architecture facilitates the gathering of data for offline analysis, enabling thorough performance comparisons and post-game assessments [17].

The ML-Agents environment for Soccer Twos Unity has broad ramifications in other areas. It can be used as a platform for academic research on team dynamics, reinforcement learning, and multi-agent systems. It can be applied to the development of intelligent coaching assistants, testing of

novel algorithms, and training of AI-driven virtual soccer players.AI-driven soccer experiences, the Soccer Twos environment has the potential to excite and captivate soccer fans [17].

In summary, the Soccer Twos Unity ML-Agents environment is a cutting-edge method for machine learning and soccer simulation. This environment brings up new possibilities for training intelligent agents, investigating team dynamics, and advancing the area of artificial intelligence in sports by fusing the thrill of soccer with the power of ML-Agents. The Soccer Twos Unity ML-Agents environment combines the domains of soccer with machine learning to provide a rich and immersive experience for study, development, or entertainment [17].

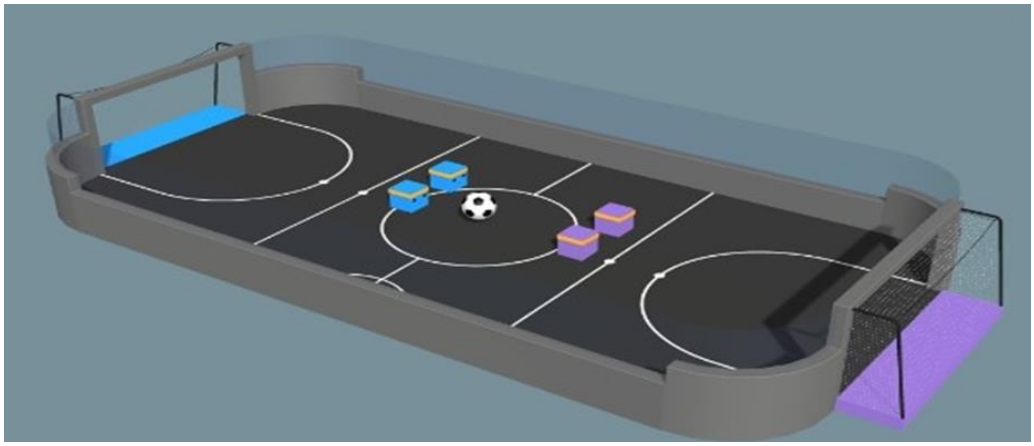

Figure 2 Soccer Twos Unity ML-Agents environment [17]

- **Set-up:** Environment involves four agents compete in two against two toy soccer match.
- **Goal**: Getting the ball into the opponent's goal while keeping it out of your own is the aim.
- **Agents:** There are two distinct Multi Agent Groups in the environment, each containing two agents. Measurements: Soccer Pairs.
- **Agent Reward Function (dependent):** (1 - Accumulated time penalty) The accumulated time penalty is reset to 0 at the start of each episode and is increased by (1 / MaxStep) whenever the ball enters the opponent's goal [17]. Once the ball enters the team's goal, -1.
- **Behavior Parameters:** Vector Observation space: 336 corresponding to 11 ray-casts forward distributed over 120 degrees and 3 ray-casts backward distributed over 90 degrees each detecting 6 possible object types, along with the object's distance. The forward ray-casts contribute 264 state dimensions and backward 72 state dimensions over three observation stacks.
- **Actions:** There are three distinct, branched actions that represent rotation, sideways, forward, and backward movement.
- **Visual Observations:** None
- **Float Properties:** Two

- **Ball scale:** Specifies the scale of the ball in the 3 dimensions (equal across the three dimensions) Default: 7.5, Recommended minimum: 4,Recommended maximum: 10 ,Gravity: Magnitude of the gravity Default: 9.81,Recommended minimum: 6,Recommended maximum: 20.

4.3 ML-Agents Toolkit for Our Sample Game

- **Learning Environment** – includes every character in the game as well as the Unity scene. Agents can watch, act, and learn within the context of the Unity scene. Your goal will determine how you arrange the Unity scene to function as a learning environment. In the event that you are attempting to address a particular, narrowly focused reinforcement learning problem, you can train and test trained agents on the same scenario. Or, you can be teaching agents how to function in a challenging simulation or game [18].
- **Python Low-Level API** –The Python Low-Level API facilitates interaction and manipulation of a learning environment using a low-level Python interface. It should be noted that, in contrast to the Learning Environment, the Python API exists outside of Unity and interacts with it via the Communicator. The Python training process uses this API, which is part of a special mlagents environment Python package, to interact with and manage the Academy as it is being trained [18].
- **External Communicator** –The External Communicator serves as a bridge between the Python Low-Level API and the Learning Environment.
- **Python Trainers**: All the machine learning techniques needed to enable training agents are contained in Python Trainers. The algorithms are included in their own mlagents package and are implemented in Python. The package covers all the training settings and methods described in this paper and exposes a single command-line utility called mlagents-learn. The Python Low-Level API is the only interface that the Python Trainers use [18].

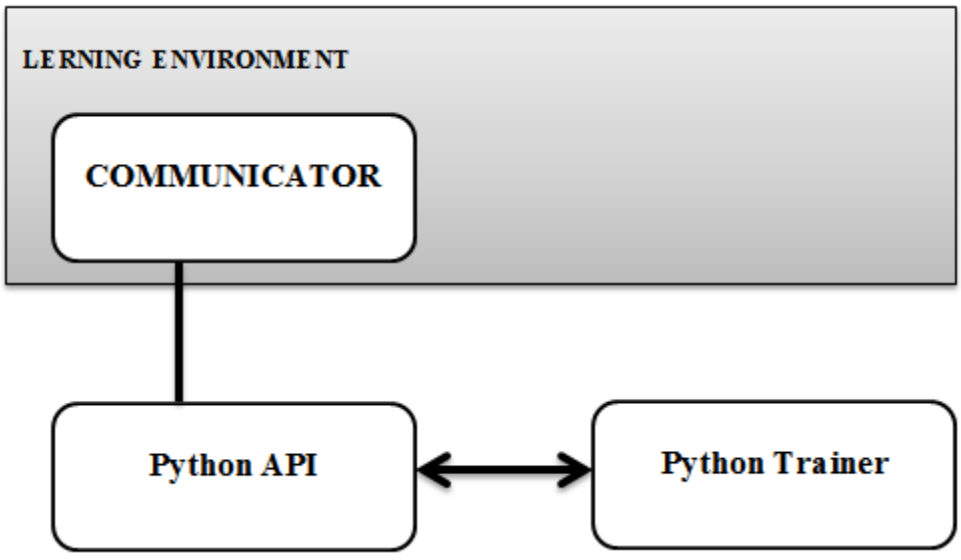

Figure 3 block diagram of ML-Agents Toolkit for our sample game [18].

4.4 Sensor Components

The API offers a number of Sensor Components, such as:

Camera Sensor: This component makes observations using pictures from a camera.

Ray Perception Sensor: Component employs a set of ray casts to gather information for observations.

4.4.1Camera Sensor

Agents often receive visual observations from either a Camera Sensor. These gather data from the images and convert it into a 3D Tensor that may be used to feed the agent policy's convolutional neural network (CNN).This enables agents to pick up knowledge from the observation images' spatial regularities. Using the same agent, visual and vector observations are achievable [19].

Visual observations can obtain from the cameras in your scene. Either a Camera Sensor Component or another Sensor Component can be added to an Agent in order to add a visual observation. Next, drag your desired render material or camera into the Camera field. An agent can be tied to several cameras, render textures, or even a combination of both. Set the image's width and height (in pixels) and whether the visual observation is in color or grayscale for each one [19].

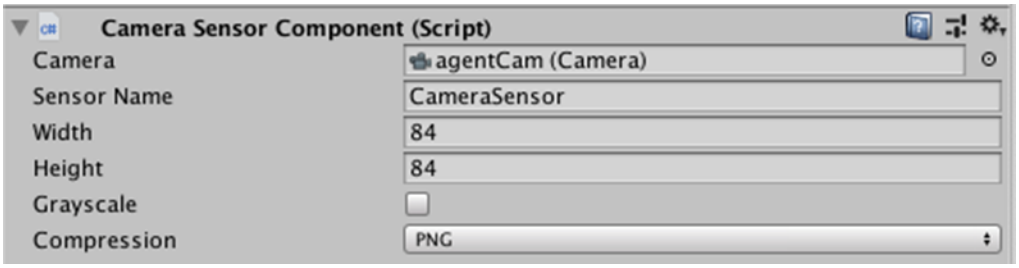

Figure 4 Camera Sensor Components [19]

The number of visual observations and resolutions (including whether or not they are grayscale) must be the same for every Agent using the same Policy. In order for an Agent to be sorted deterministically, it is also necessary for each sensor component to have a unique name (albeit a sensor component may have the same name on several Agents) [19]. By setting observation stacks to a number larger than 1, visual observations also facilitate stacking. The final dimension (channel dimension) will be layered with the visual observations from the previous stack Size steps [19].

4.4.2 RayCast Observations

Another way to give an agent observations is using raycasts.By including a Ray Perception SensorComponent3D in the Agent Game Object, this is readily achieved. A number of rays (or spheres, depending on the parameters) are thrown into the physical world during observations, and the observation vector that is generated is determined by the objects that are struck [20].

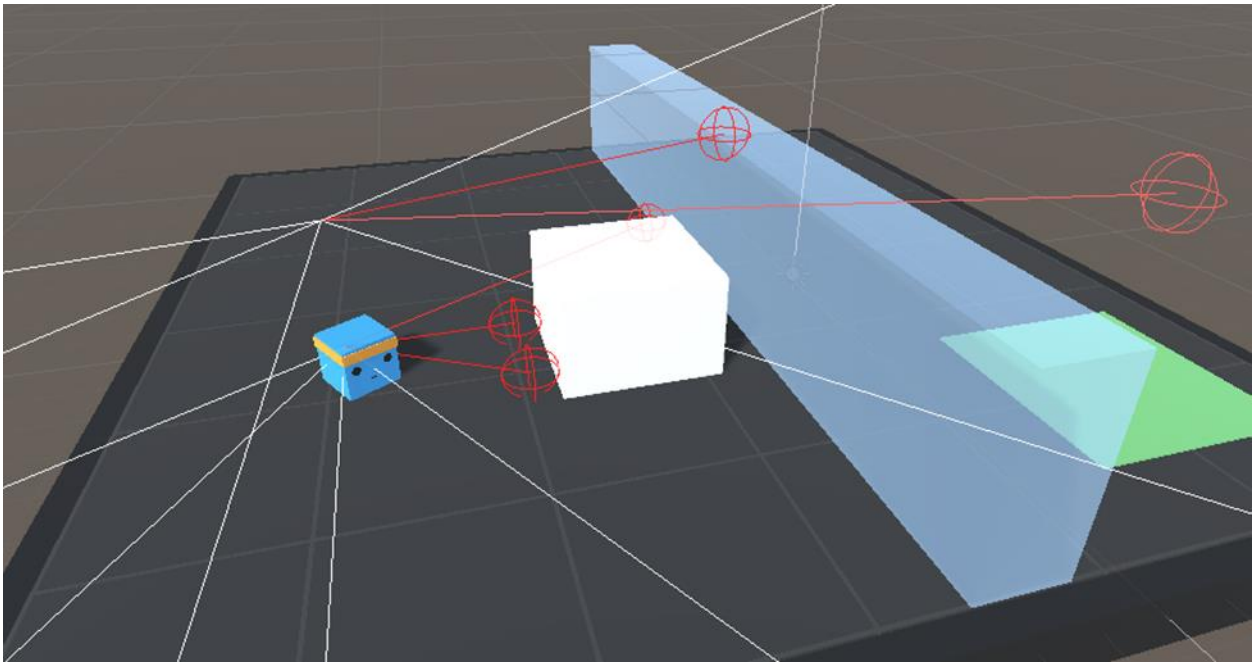

Figure 5 Ray Sensor components [20]

Ray Sensor components displays the various settings shown on figure 9.

- **Detectable Tags:** An enumeration of strings representing the various kinds of things that the agent ought to be capable of differentiating. For instance, the list of things to detect consists of "food" and "goal."
- list of strings corresponding to the types of objects that the Agent should be able to distinguish between. For example, we use “food" and “goal” as the list of objects to detect.
- **Rays per Direction:** The number of rays cast is determined by the parameter called Rays per Direction. This many rays are cast to the left and right, and one ray is always cast forward.
- **Max Ray Degrees:** The outermost ray's angle expressed in degrees. The agent's left and right are corresponding to 90 degrees [20].
- **Sphere Cast:** Radius the size of the sphere used for sphere casting. If set to 0, rays will be used instead of spheres. Rays may be more efficient, especially in complex scenes.

Ray Length The length of the casts

- **Ray Layer Mask:** The raycast or spherecast receives the Layer Mask. This can be used to cast without taking into account specific kinds of objects.
- **Observation Stacks:** The quantity of prior outcomes to "stack" beside the cast outcomes. Keep in mind that this may occur regardless of the Behavior Parameters "Stacked Vectors" value. The number of previous results to "stack" with the cast results. Note that this can be independent of the "Stacked Vectors" setting in Behavior Parameters.
- **Start Vertical Offset (3D only):** The vertical offset of the ray start point.
- **End Vertical Offset (3D only):** The vertical offset of the ray end point [20].

Figure 9 shows that there are two RayPerceptionSensorComponent3Ds on the Agent. Both make use of 90 maximum ray degrees and 3 rays per direction. The Agent is able to determine whether it is safe to jump over the wall because one of the components has a vertical offset.The total size of the created observations is (Observation Stacks) * (1 + 2 * Rays per Direction) * (Num Detectable Tags + 2).So, the number of rays and tags should be kept as small as possible to reduce the amount of data used. Note that this is separate from the State Size defined in Behavior Parameters, so you don't need to worry about the formula above when setting the State Size [20].

4.5 Rewards

In reinforcement learning (RL), rewards play a critical role in indicating to agents which controls are beneficial and should be reproduced in the event that the same state arises again. Because they provide agents feedback on their behavior, rewards are essential in encouraging them to learn and explore. Agents would struggle to finish tasks or find solutions without rewards since they wouldn't be able to recognize which controls are crucial. The agent can learn from its activities and modify its behavior over time with the help of rewards.

When creating an artificial intelligence (AI) system, choosing a reward function is an essential step in the process. It's critical to realize that there isn't a single reward function that works for everyone. It is crucial to carefully evaluate the desired behavior and choose a reward function that supports it, as different reward functions might cause an agent to behave in different ways.

In order to assess what the agent has learnt, we frequently provide a precise criterion for optimization and then review the taught policy after the fact. We can discover nifty control strategies that the taught agent could not have imagined before the training by examining its policy. In certain domains, the new policy may even go against conventional wisdom.

However, implementing rewards in an actual learning system has its limitations. For example, in order to maintain the behavior, rewards must be offered often. This may not be feasible, particularly if there is a material prize involved, such as food, toys, or cash. In addition, incentives may unintentionally teach agents that they just need to carry out a desired activity in order to receive an incentive. When the benefits stop, this may cause the agent to quit engaging in the behavior. Furthermore, in a multi-agent scenario, rewards have the potential to foster rivalry and conflict rather than cooperation among agents.

Therefore, while guiding agent behavior in a "real" situation, it is imperative to take into account the limitations of rewards and apply them prudently. Making a simulator that mimics the actual world and using it to train the agent is one option. This can help agents avoid costly or deadly mistakes in the real world and gives them more control over the conditions they are exposed to.

In order to maximize an agent's cumulative reward over time, the PPO reinforcement learning algorithm optimizes the choices the agent makes. Your agent will learn more effectively the more effective your reward system.

Allocate rewards to an Agent by Call the AddReward() or SetReward() methods on the agent to assign prizes. Every decision should have a reward allocated within the interval [-1, 1]. Values outside of this range may cause training to become unstable. When the agent gets a new decision, the reward value is reset to zero. To determine how good the prior decision was, the incentives will be added together if AddReward() is used more than once for a single agent decision. Any incentives that have been granted to an agent since the preceding choice will be superseded by the SetReward() [20].

Lastly, it is advised that a human be kept informed about the agent's actions during training in order to confirm that it is functioning securely and achieving its intended goals. This is especially important for realistic agent implementations. The AI system's success and safety can be guaranteed in this way.

4.6 Self-play

Use for the games Self-play (Competitive) Self-play is triggered by including the self-play hyper parameter hierarchy in the trainer configuration. To distinguish opposing agents, set the team ID to different integer values in the behavior parameters script on the agent prefab.

The inclusion of the self-play hyperparameter hierarchy in the trainer configuration initiates self-play. In the behavior parameters script on the agent prefab, assign the team ID to distinct integer values to differentiate opposing agents [20].

Agents working together should be put to the same group, however agents competing against one another should be assigned different Team Ids. This is how groups vary from teams in competitive scenarios. There should be two groups, one for each team, and each team should have a unique Team Id if there is only one playing field and two teams in the scene. There should be two Groups per playing field and two distinct Team Ids for the duration of the Scene if this playing field is repeated frequently throughout the Scene (for example, during training speedup). Training with both MA-POCA and self-play is possible in contexts where Groups and Team Ids are configured. Each team has two agents, and there are two playing fields where teams compete with one another in the diagram below. There should be four group managers, one for each pair of agents, and all blue agents should have a common Team Id (although purple agents should have separate IDs) [20].

Chapter 5

5. Results and Discussions

5.1 Introduction

This chapter delves into the important task of examining how artificial agents' acquired skills and strategies hold up in the face of disparate sensory modalities. Particularly, we study how to combine and disentangle a camera and ray cast sensors separately and together in the context of two different reinforcement learning algorithms: Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC). We conduct our experiments in Unity environments, using the Soccer Twos and Food Collector environments as test beds.

This chapter's main goal is to examine how resilient and flexible AI agents are in the face of diverse sensory inputs. Through evaluation of their performance using various sensor configurations and reinforcement learning algorithms, our goal is to gain understanding of how transferable acquired abilities and tactics are.

Examining the importance of our findings, we consider the wider AI research and talk about possible directions for future study and useful applications. Comprehensive investigation of how task contexts, reinforcement learning algorithms, and sensory modalities interact to shape AI behavior. Our objective is to offer significant perspectives on the development and enhancement of artificial intelligence systems for practical application through our meticulous testing and examination.

5.2 Results of Food Collector

Agents are tasked with gathering food items to earn points in the dynamic Food Collector environment, while avoiding dangerous objects that lower their overall score. We employed two well-known reinforcement learning algorithms in this context: Soft Actor-Critic (SAC) and Proximal Policy Optimization (PPO), to investigate the invariance of learning outcomes when agents are outfitted with disparate sensory modalities, ray cast and camera sensors, separately and in combination. We were able to examine the flexibility and effectiveness of the learning techniques the agents created in response to the sensory data at their disposal thanks to this methodology. The results of these studies, which have been divided into thorough subsections, provide insight into the vital role that sensory inputs play in the generalization and optimization of agent behaviors and tactics in the context of the Food Collector.

To evaluate whether learning outcomes are consistent across various sensory modalities, we applied Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC) reinforcement learning techniques. The next subsections contain a presentation of the experiment outcomes.

5.2.1 Results of Food Collector using PPO

A popular reinforcement learning technique called Proximal Policy Optimization (PPO) is renowned for its dependability and efficiency across a range of training contexts. To optimize training procedures and ensure minor policy modifications for better learning results, it makes use of the policy gradient approach. In this part, we want to assess how various sensory modalities—namely, a camera and ray sensors—affect artificial agents' capacity for learning and their ability to acquire new skills in the context of the Food Collector. We attempted to determine the effect of sensory modality integration on learning effectiveness and strategy formation by contrasting the performance results of agents outfitted only with ray sensors, camera sensors, and a mix of both. The findings, which are illustrated agents' learning performances, irrespective of the sensory modality, show striking parallels, with very little variation between the two modalities.

As described in the methodology section 3.3.1, Proximal Policy Optimization (PPO) is a policy optimization algorithm that evaluates policies by iteratively updating parameters. This technique was applied in this section to evaluate the invariance of learning outcomes across different sensory modalities. Our aim was to examine the effects of using the ray and a camera sensory modality on the agent's acquisition of similar abilities.

The agent results, displayed in the below tables, demonstrate the learning performances under the camera-only, a ray-only modalities, and both. The findings show that, although there are few small variations between the modalities, the agent's learning performance is generally comparable. We analyzed this further by contrasting the variation between the camera-only and a ray-only settings with the amalgamated sense modalities.

		Reward	
Environments		PPO (Mean)	PPO (Std)
Food Collector	Ray sensor	17.653	5.677
	Camera sensor	10.421	3.924
	Ray and Camera sensor	32.842	7.837

Table 1 food collector with PPO

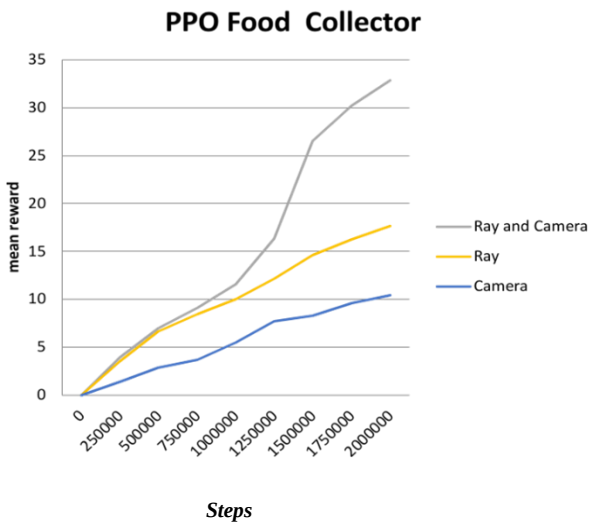

Figure 6 Food Collector with ppo graph

The variance between the ray-only and a camera-only sensory modalities is less than the variance between each of the two settings and the combined sensory modalities, according to the variance analysis. This suggests that while there are some variations in the modalities' learning performance, these variations can be reduced and more consistent learning results can be obtained by combining the ray and camera inputs. A comparison of the learning curves for agents using PPO and various sensory modalities is presented in Figure 10. The agents' performance varies slightly, but overall patterns indicate that they tend to converge toward comparable levels of performance.

The PPO results show that artificial agents can maximize their performance in the Food Collector environment by adjusting their learned tactics and talents to various sensory inputs. While there are certain variations in the learning outcomes between a ray-only and a camera-only modalities, combining the two results in more reliable and efficient learning outcomes.

The agents' performance under each modality is thoroughly compared in the learning performance tables. Agents utilizing the ray-only modality received an average reward of 17.653, compared to agents utilizing the camera-only modality who received an average value of 10.421. These findings show that, despite some performance variance, agents were able to successfully gather food items while avoiding poison in both modalities.

These results shows the agents' learning curves for each modality. The curves' convergence indicates that agents with a ray and camera inputs were eventually able to perform at comparable levels, indicating the flexibility and effectiveness of their tactics. The usefulness of utilizing numerous sensory inputs is highlighted by the decreased difference between the ray-only and a camera-only modalities compared to the combined sensory modalities. Agents can improve their decision-making and gain better grasp of their surroundings by merging data from many modalities.

In summary, the PPO results demonstrate that artificial agents can, in fact, maximize their performance in the Food Collector environment by customizing their learned tactics and talents to various sensory inputs. By combining the ray and camera modalities, learning outcomes can be achieved more consistently and effectively, which helps to ensure those skills remain the same across a variety of sensory modalities. Additional research and testing using alternative reinforcement learning techniques may yield more information about how best to use sensory modalities for agent learning and performance enhancement.

5.2.2 Results of Food Collector using SAC

Similar to the PPO approach, we utilized SAC to evaluate learning outcomes across different sensory modalities. Results mirrored those observed with PPO, with combined sensory modalities demonstrating reduced variance compared to individual modalities.

In this section, we used the Soft Actor-Critic (SAC) reinforcement learning approach to study the invariance of learning outcomes across several sensory modalities. As stated in Section 3.3.2, SAC is an algorithm designed for agent training in continuous action space contexts. Our objective was to find out how the agent's acquisition of the identical abilities was affected by the use of the camera and a ray sensory modality.

Table 2 shows the agents' outcomes, which show how effectively they learned with the ray-only, a camera-only, and combined modalities, respectively. Consistent with the PPO results, we discovered that the agent's learning performance is essentially similar between the two modalities, with very minor differences. To better investigate this, we compared the variance between the camera-only and a ray-only settings.

		Reward	
Environments		SAC (Mean)	SAC (Std)
Food Collector	Ray sensor	35.684	16.824
	Camera sensor	32.365	6.106
	Ray and Camera sensor	78.158	33.021

Table 2 Food collector with SAC

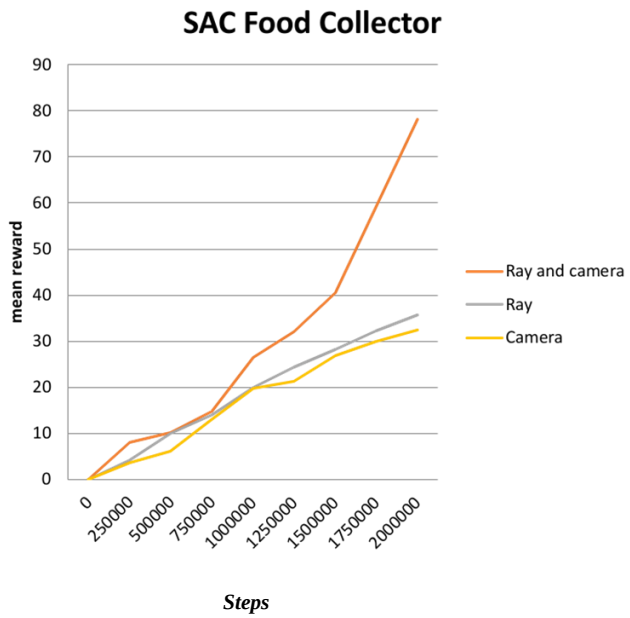

Figure 7 SAC Food Collector graph

The outcomes of the SAC variance analysis agreed with the PPO findings. It was discovered that the variance between the two settings and the combined sensory modalities was greater than the variance between the ray-only and a camera-only sensory modalities. This suggests that learning outcomes can be more consistently achieved with fewer differences when both ray and camera inputs are used together.

PPO and SAC algorithm findings exhibit a similar tendency, indicating that artificial agents can optimize their performance in the Food Collector environment by adapting their acquired tactics and skills to different sensory inputs. While there are some differences in the learning outcomes between the camera-only and a ray-only modalities, the combination of the two produces more dependable and effective results.

The learning performance table (a table 2 and picture 11) provides comparison of the agents' performance under each modality. According to a Table 2, the average reward for agents using the ray-only modality was 35.684, while the average reward for agents using the camera-only modality was 32.365. Comparable patterns emerged from the study carried out on the Food Collector environment utilizing the two distinct models, SAC, and PPO. This implies that by adjusting their tactics and abilities to different sensory inputs, artificial agents may perform better. Combining both modalities produced more dependable and effective learning outcomes, despite little differences in the learning outcomes between the usage of a ray-only and a camera-only modalities.

The outcomes produced by the Soft Actor-Critic (SAC) algorithm show that by modifying their acquired techniques and skills to suit various sensory inputs, artificial agents can greatly enhance their performance in the Food Collector environment. More reliable and efficient learning results are made possible by the combination of the ray and camera modalities, which also helps to explain how abilities transfer across different sensory modalities.

5.3 Results of Soccer Twos

We conducted several experiments and analyzed the results in detail. In the following subsections, we present a comprehensive overview of the findings from these experiments, which sheds light on the effectiveness of the PPO and SAC methods in learning invariant policies across different sensory modalities.

5.3.1 Results of Soccer Twos Using PPO

Teams of two agents each compete in a streamlined version of soccer in the Soccer Twos environment, and the goal is to score goals against the opposition. This kind of setting is ideal for evaluating the effects of various sensory modalities on learning and performance, as it assesses both cooperative and individual agent skills. We chose to use the Proximal Policy Optimization (PPO) algorithm, just as we did with the Food Collector environment, because it has a track record of success in complicated action spaces and cooperative dynamics. We investigated whether agents in this team-based scenario could acquire skills and create strategies to a similar degree if they were using a ray and camera sensors independently or in combination. The findings, which are shown in comprehensive tables and charts, show that agents show some degree of performance consistency in learning across the various sensory configurations. However, the integration of both sensory modalities seems to offer a minor advantage, reducing the performance variance found when comparing the single-modality settings, comparable to our findings in the Food Collector scenario. This result implies that in cooperative contexts, the combination of sensory inputs may improve agents' strategic depth and adaptability.

In this part, we assessed the invariance of agents' learning outcomes in the Soccer Twos Unity ML-Agents environment by applying the Proximal Policy Optimization (PPO) reinforcement learning technique.

Our goal was to investigate the effects of using the camera and a ray sensory modality on agents' acquisition of the same abilities in the Soccer Twos scenario. The learning performances of the agents under the ray-only, a camera-only, and both modalities are displayed in the table below

		Reward	
Environments		PPO (Mean)	PPO (Std)
Soccer Twos	Ray sensor	8.150	1.449
	Camera sensor	3.086	0.819
	Ray and Camera sensor	12.063	5.563

Table 3 Soccer Twos with PPO

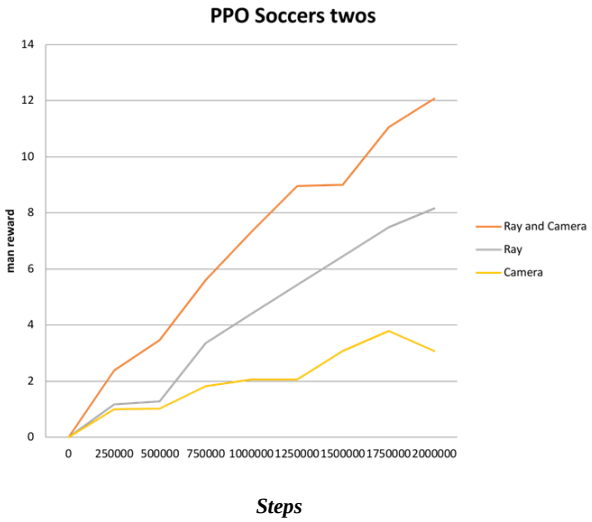

Figure 8 PPO Soccer Twos graph

Table 3 and Figure 12 illustrate the learning performances of the agents for each modality and the results of their performances. The tables show that the agents that used the camera-only modality received an average reward of 3.086 and the agents who used the ray-only modality an average reward of 8.150. These findings offer insightful information on how the agents functioned in the Soccer Twos environment with various sensory inputs.

The learning curves for agents utilizing PPO with ray-only, camera-only, and both modalities are shown in Figure 12. These curves show how learning for both modalities progresses remarkably across episodes. The curves show that the agents were able to learn and get better over time under both modalities, despite some volatility.

The Soccer Twos Unity ML-Agents environment's PPO-acquired findings show that agents can maximize their effectiveness by customizing their tactics and skill sets to various sensory inputs. A thorough comparison of the agents' performance under the ray-only and camera-only modalities is given by the performance tables and learning curves.

The agents' learning performances demonstrate that throughout the course of episodes, agents using both ray and camera inputs were able to learn .This suggests that the agents were able to maximize their performance by customizing their tactics and skill set to fit the various sensory modalities.

5.3.2 Results of Soccer Twos Using SAC

Following the structure established in the Food Collector analyses, we also applied the Soft Actor-Critic (SAC) algorithm to the Soccer Twos environment. SAC, known for its sample efficiency and stability, especially in environments with continuous action spaces, was anticipated to yield insightful results regarding the adaptability and learning efficacy of agents under varied sensory modalities. The methodology section provides an exhaustive overview of SAC for readers seeking to understand its principles and advantages in reinforcement learning

scenarios.
Our objective was to compare the SAC-driven learning outcomes of agents utilizing ray, camera, and combined sensory modalities. The anticipated results were to observe how these sensory setups influence not only the individual skill levels of agents but also their ability to cooperate and compete within the Soccer Twos framework. Initial findings suggest that, akin to the PPO results, the sensory modality employed does have an impact on learning efficiency and strategy formulation, with combined modalities generally leading to more nuanced and adaptable agent behaviors.

In this section, we employed the Soft Actor-Critic (SAC) reinforcement learning method to examine the invariance of learning outcomes of agents in the Soccer Twos Unity ML-Agents environment. We aimed to investigate the impact of using ray and camera sensory modalities on the agent's ability to learn skills in the Soccer Twos environment. These outcomes reveal how the agents performed in learning the same skills

		Reward	
Environments		SAC (Mean)	SAC (Std)
Soccer Twos	Ray sensor	9.99	2.397
	Camera sensor	5.075	1.143
	Ray and Camera sensor	27.949	13.596

Table 4 Soccer Twos with SAC

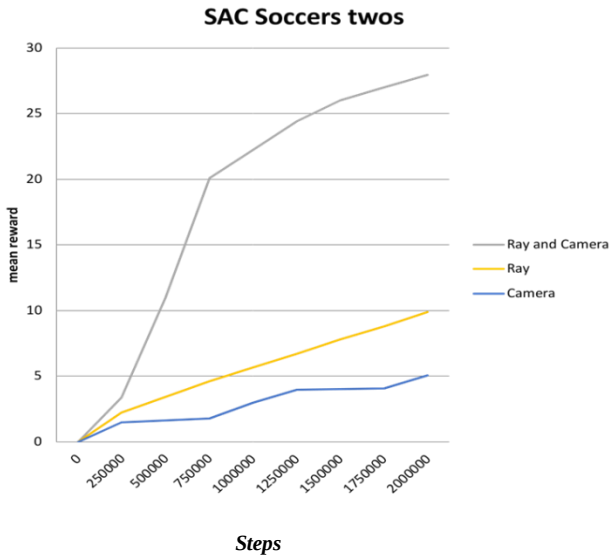

Figure 9 SAC Soccer Twos graph

The following information pertains to the performance of agents in the Soccer Twos environment with different sensory inputs. Table 4 and figure 13 display the learning outcomes of agents under each modality. As per the results, agents using the ray-only modality achieved an average reward of 9.99, while those using the camera-only modality achieved an average reward of 5.075. These results shed light on how the agents performed in the environment.

The graph shown in figure 13 displays the learning progress of agents that used SAC with ray-only, camera-only, and both modalities. The curves demonstrate how the agents' performance improved over a series of episodes for each modality. Although there were some minor fluctuations, the curves indicate that the agents were able to learn and enhance their performance under both modalities over time.

In summary, the successful implementation of Soft Actor-Critic (SAC) in the Soccer Twos Unity ML-Agents environment has demonstrated the agents' ability to optimize their performance by adapting their skills and strategies to different sensory inputs. The combination of ray and camera modalities has consistently yielded effective learning outcomes, leading to the invariance of skills across diverse sensory modalities.

5.4 Comparisons of the Results

The comparative analysis across both the Food Collector and Soccer Twos environments, under the PPO and SAC algorithms, highlights several key insights into the role of sensory modalities in artificial agent development. Firstly, the use of combined sensory inputs consistently appears to confer advantage in terms of learning performance and strategy complexity, across both individual and team-based tasks. This suggests that the richness of environmental information available to agents can significantly influence their learning trajectories and the robustness of the strategies they develop.

Furthermore, the relative performance of PPO versus SAC in these environments provides valuable lessons for the selection of reinforcement learning algorithms based on the specific characteristics of the task and the desired outcomes. The nuanced differences in learning outcomes observed between the two algorithms under various sensory conditions underscore the importance of algorithmic choice in the design of artificial intelligence systems.

In elaborating on these comparisons, this section aims to guide researchers and practitioners in making informed decisions about the integration of sensory modalities for the development of adaptable, and efficient artificial agents.

In this section, we compare the results obtained from the experiments using Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC) in the Soccer Twos Unity ML-Agents environment. Both PPO and SAC were used to assess the invariance of learning outcomes of agents across different sensory modalities, specifically ray-only and camera-only inputs.

5.4.1 Learning Performance Comparison

The following table provides a quantitative view of the agents' average rewards under each modality.

		Reward			
Environments		PPO (Mean)	PPO (Std)	SAC (Mean)	SAC (Std)
Food Collector	Ray sensor	17.653	5.677	35.684	16.824
	Camera sensor	10.421	3.924	32.365	6.106
	Ray and Camera sensor	32.842	7.837	78.158	33.021
Soccer Twos	Ray sensor	8.150	1.449	9.99	2.397
	Camera sensor	3.086	0.819	5.075	1.143
	Ray and Camera sensor	12.063	5.563	27.949	13.596

Table 5 Comparison of Food Collector and Soccer Twos

Comparing the results within each method, we observe similar patterns that agents achieved lower average rewards when using camera-only inputs compared to ray-only inputs in both PPO and SAC. The difference in average rewards between the two modalities, indicating that agents were able to perform well with combined ray and camera sensory modality.

- **Comparing the variance results**

Both PPO and SAC exhibited smaller variances between ray-only and camera-only modalities compared to the combined sensory modalities. This suggests that the use of both ray and camera inputs together helped to minimize differences and achieve more consistent learning outcomes across the agents.

- **Learning Curve Comparison**

Figures below illustrates the learning curves for agents using PPO with ray-only and camera-only modalities, showing the progression of learning over episodes. Similarly, the following figures shows the learning curves for agents using SAC with the same modalities.

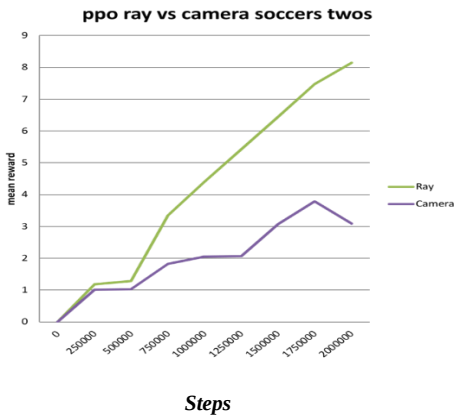

Figure 10 PPO ray vs Camera Soccer Twos

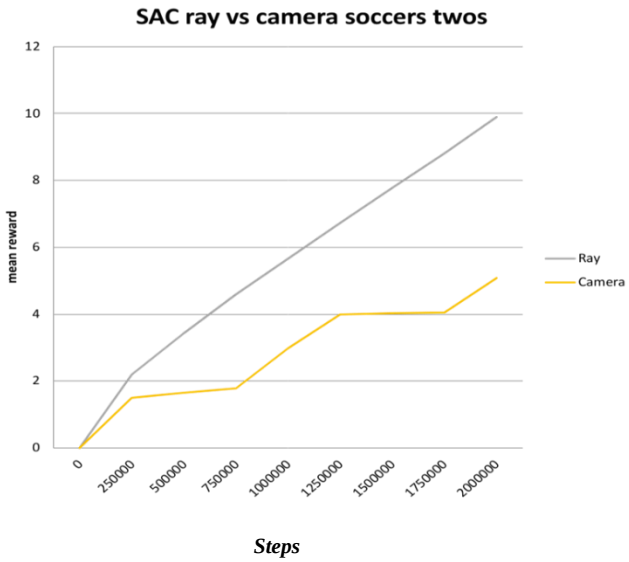

Figure 11 SAC ray vs. Camera Soccer Twos

PPO Learning Curves show that both ray-only and camera-only modalities show a gradual improvement in average rewards over episodes. Similarly, the learning curves for SAC agents exhibit a steady increase in average rewards over episodes for both modalities.

- **Conclusion and Implications**

In conclusion, the results from both PPO and SAC experiments in the Soccer Twos Unity ML-Agents environment demonstrate the agents' ability to optimize their performance by adapting developed skills and strategies to different sensory inputs. While there were differences in average rewards between ray-only and camera-only modalities, the agents showed consistent learning and improvement over time under both modalities.

The smaller variances between ray-only and camera-only modalities compared to the combined sensory modalities outcomes.

These findings have implications for the development of intelligent systems in various domains:

- **Robustness**: Agents trained with both ray and camera inputs showed more robust performance, indicating their ability to adapt to diverse sensory information.
- **Efficiency**: The combination of sensory modalities led to more consistent learning outcomes, enhancing the efficiency of agent training.
- **Versatility**: By demonstrating the invariance of skills across different sensory modalities, these results show that agents can generalize their learning to new environments with varying inputs.
- **Future Research**: Further exploration with additional RL methods and diverse environments can provide deeper insights into the optimal utilization of sensory modalities for agent learning and performance optimization.

In summary, the experiments using PPO and SAC in the Soccer Twos Unity ML-Agents environment showcase the potential of artificial agents to adapt and optimize their performance in dynamic and complex settings. The combination of ray and camera sensory modalities enables agents to achieve more consistent and effective learning outcomes, paving the way for the development of intelligent systems capable of handling diverse environments and tasks.

Sensor Performance Comparison

This study investigates how artificial agents adapt their skills and strategies under different sensory modalities within Unity environments. Agents were trained using Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC) algorithms, employing three types of sensory inputs: camera sensors, raycast sensors, and a combination of both. The objective was to compare the performance and learning adaptation across these modalities, assessing their impact on the development of invariant skills and strategies.

Camera Sensors

Performance: Agents demonstrated a strong ability to navigate and interact with the environment based on visual cues. However, learning rates varied, with some tasks showing slower progression due to the processing visual data.

Skill Transfer: Despite initial learning curves, agents developed strategies that, once established, proved robust across different visual environments.

Raycast Sensors

Performance: These agents typically learned tasks faster than their camera sensor counterparts, given the lower complexity of processing spatial information. The performance was particularly notable in navigation and obstacle avoidance tasks.

Skill Transfer: Skills learned with raycast sensors showed high adaptability, especially in environments where spatial awareness was crucial.

Combined Sensors

Performance: Agents equipped with both sensor types often achieved the highest levels of performance, benefiting from the comprehensive understanding of their surroundings. This setup led to superior adaptability and efficiency in task completion.

Similar Performance: Despite the inherent differences in sensory input, agents trained across the three modalities developed comparable skills and achieved similar performances in many tasks. This is reflected in the learning curves and performance metrics, indicating a high degree of skill and strategy invariance.

Adaptability and Robustness: The results highlight the agents' ability to adapt their learned strategies to different sensory inputs, underscoring the robustness of the skills developed through training with either PPO or SAC algorithms.

The study reveals that artificial agents can develop invariant skills and strategies under different sensory modalities when trained in Unity environments using PPO and SAC algorithms. Performance comparisons between camera sensors, raycast sensors, and a combination of both indicate that, despite the sensory modality, agents are capable of achieving similar levels of performance and adaptability. These findings underscore the potential of artificial agents to adapt to and operate within a diverse array of environments and conditions, supporting the development of more versatile and resilient AI systems.

Algorithm Performance: For both environments and across all sensor combinations, SAC tends to exhibit higher mean performance compared to PPO. The variance (Std) in performance is generally higher in SAC compared to PPO across sensor combinations, indicating potentially higher variability in SAC's performance.

Impact of Sensors: The combination of sensors appears to provide richer information for learning in these environments, leading to improved performance for both algorithms. The specific impact of sensors might vary depending on the environment, but in general, having more varied sensor inputs seems to enhance the agents' learning and performance.

Overall, these results suggest that the combination of multiple sensors tends to provide a more comprehensive input for the reinforcement learning algorithms, leading to better performance, especially in more complex environments like the Food Collector.

5.5 Answers to Research Questions

In the following section, we will discuss into the project questions and provide detailed explanations of our findings based on the results of the project. We will explore each question thoroughly and provide insights into the data that we have gathered. We aim to provide a clear and comprehensive understanding of the project and its outcomes, allowing readers to gain a deeper insight into the research that has been conducted.

- **Do artificial agents demonstrate consistent skill application across different sensory modalities?**

According to the research findings, artificial agents do exhibit consistent application of skills across many sensory modalities. The goal of the study was to demonstrate how artificial agents' abilities and learning outcomes remain constant across many sensory input modalities. The artificial agents were discovered to be able to sustain constant training performance despite variations in sensory input.

In particular, the study demonstrated that while agents generated distinct training performances under various sensory modalities, the best outcomes were obtained when agents simultaneously exploited the ray and camera input modalities. This suggests that although performance may vary depending on the particular sensory input, overall, the agents were able to adjust and use their skills consistently throughout these modalities.

- **Are certain skills or strategies transferrable and equally effective across diverse sensory modalities in artificial agents?**

Indeed, the literature indicates that specific abilities or tactics are applicable and just as successful in artificial entities with a variety of sensory modalities. About various sensory input modalities, the study sought to show how agents' skills and learning performance remain constant. Regarding the transferability of abilities and tactics.

1) **Consistent Performance**: Research showed that although agents may have different training performances under different sensory modalities, they were able to achieve optimal results when using both beam and camera input modalities simultaneously This means within these inputs, certain skills or strategies were effective and transferable.

2) **Invariance of results**: The study showed the invariance of outcomes across an agent's learning skills and strategies. This suggests that regardless of the specific input, was able to maintain consistent learning and performance.

3) **Adaptation**: The artificial agents were able to adapt their skills and strategies to cognitive processes. For example, in a situation where they had to reach a goal with different sensory resources, they were able to use their strategies more effectively, demonstrating the flexibility of their knowledge.

4) **Skilled RL strategies**: The use of reinforcement learning (RL) strategies such as close policy optimization (PPO) and soft actor-critic (SAC) contributed to the transfer of skills. These strategies are designed to improve the quality of systems and train them to perform tasks more efficiently.

- **What factors influence the invariance or variation of skills and strategies in artificial agents across different sensory modalities?**

Several factors can influence the invariance or variability of skills and strategies in artifacts across cognitive processes. Here are some highlights based on the research:

1) **Characteristics of sensory input**: Specific sensory information such as shapes, noise levels, and types of information can affect how agents perceive and process the environment different mechanisms can provide information variety that can influence the strategies and skills of agents.
2) **Process compatibility**: Some cognitive strategies may be more compatible with certain tasks or environments than others. For example, the simultaneous use of beam and camera input gave the best results in the analysis. Agents may need to integrate information from multiple sources to make effective decisions.
3) **Reinforcement Learning Algorithms:** The choice of reinforcement learning (RL) algorithms, such as close policy optimization (PPO) and soft actor-critic (SAC), can affect how agents learn and adapt different algorithms have strengths and weaknesses, which can affect power with agents able to maintain consistent performance across modalities.
4) **Work complexity:** The complexity of the task or environment in which agents work can affect their strategies. Simple jobs may require core competencies that can be easily transferred in a variety of ways, while complex jobs may require more specialized strategies for each task
5) **Reward structure**: In RL, the structure of the reward structure plays an important role in shaping agent behavior. The rewards associated with specific cognitive processes may influence how they prioritize and learn from different inputs.

In conclusion, changes or modifications of skills and strategies in different sensory processes can be influenced by sensory input, training data, RL algorithms, task complexity, and agent architecture.

- **Can artificial agents optimize their performance by adapting developed skills and strategies to different sensory inputs?**

Yes, artificial agents can improve their efficacy by adapting acquired skills and strategies to different sensory inputs. The findings of this review provide evidence that agents can successfully adapt their learned behaviors and mechanisms to different sensory strategies here is how can be achieved through adaptation:

1) **Flexible learning**: Artificial agents especially those trained with reinforcement learning (RL) techniques such as Proximal policy optimization (PPO) and Soft Actor-Critic (SAC), are designed to learn simple policies these techniques or actions represent what agents use to make decisions in situations.
2) **Transfer learning**: Agents can use transfer learning strategies to transfer knowledge gained from one sensory modality to another. Transfer learning enables agents to transfer learned skills and strategies from a source mode in which they are proficient to a target mode that requires adaptation.
3) **Reward Shaping**: Agents can be motivated to perform better in cognitive strategies by high enough rewards. Rewards can be designed to encourage agents to seek out and learn from new sensory inputs, thereby adjusting their strategies to improve overall performance
4) **Iterative Improvement**: RL algorithms often involve iterative processes where agents learn from experiences, update their policies, and improve over time. This iterative

improvement allows agents to adapt their strategies based on the feedback they receive from different sensory inputs.

5) **Efficient RL Methods**: The use of efficient RL methods like PPO and SAC, as mentioned in the study, contributes to agents' ability to optimize their performance. These methods are designed to find optimal policies through efficient exploration of action spaces, which is crucial for adapting to different sensory inputs.

In conclusion, artificial agents can optimize their performance by adapting developed skills and strategies to different sensory inputs. Through flexible learning, transfer learning, reward shaping, iterative improvement, and efficient RL methods, agents can effectively adjust their behaviors to achieve optimal performance across diverse sensory modalities.

5.6 Discussion

Our study delved into the critical exploration of the invariance of skills and strategies developed by artificial agents when exposed to different sensory modalities, namely camera sensors and ray cast sensors, both individually and in conjunction. By employing the Soft Actor-Critic (SAC) and Proximal Policy Optimization (PPO) algorithms in the Unity environments of Food Collector and Soccer Twos, we sought to uncover insights into the adaptability and performance of agents under varying sensory conditions.

- **Impact of Sensory Modalities**

The integration of different sensory modalities had a discernible impact on the learning performance and strategy development of artificial agents. Across both environments, agents equipped with a combination of camera and ray cast sensors generally exhibited improved performance compared to those using a single sensory modality. This suggests that the fusion of spatial and visual information enables agents to form a more comprehensive understanding of their environment, leading to more effective decision-making and adaptive behaviors.

- **Effect of Reinforcement Learning Algorithms**

Our experiments revealed nuanced differences in the performance of SAC and PPO algorithms across different sensory setups. SAC demonstrated a slight advantage in tasks requiring fine-tuned control and continuous action spaces, while PPO exhibited robust performance across a broader range of tasks. These findings underscore the importance of selecting the appropriate reinforcement learning algorithm based on task requirements and environmental characteristics.

- **Achieving Invariance Across Modalities**

While our results indicated a degree of invariance in learned skills and strategies across sensory modalities, complete invariance proved elusive. Agents trained with both camera and ray cast sensors showcased a higher degree of adaptability and strategy robustness, yet minor differences in performance between modalities persisted. This suggests that while multimodal sensory inputs can enhance invariance, the unique information provided by each modality may still influence agent behavior to some extent.

❖ Environmental Context and Adaptability

The adaptability and performance of agents were significantly influenced by the environmental context. In the cooperative and competitive setting of Soccer Twos, the combination of sensory modalities and algorithm choice had a pronounced impact on agents' ability to develop cooperative strategies and adapt to dynamic game conditions. In contrast, the individual decision-making focus of the Food Collector environment showcased similar trends but with less pronounced effects. This underscores the importance of tailoring AI systems to the specific demands of the task environment to optimize performance effectively.

❖ Implications and Future Directions

Our findings have several implications for the design and optimization of artificial intelligence systems. Firstly, the integration of multiple sensory modalities can enhance agent adaptability and robustness, especially in complex and dynamic environments. Secondly, the selection of the appropriate reinforcement learning algorithm is crucial in maximizing learning efficiency and performance across different tasks. Lastly, further research is warranted to explore additional sensory modalities and their impact on agent behavior, as well as to investigate strategies for achieving greater invariance across modalities.

In conclusion, our study provides valuable insights into the interplay between sensory modalities, reinforcement learning algorithms, and environmental contexts in shaping the learning and adaptability of artificial agents. By elucidating these relationships, we contribute to the ongoing efforts to develop more capable, versatile, and resilient AI systems for a wide range of applications.

Chapter 6

6. Conclusion and Recommendations

6.1 Conclusion

Our research journey has been dedicated to testing the invariance of skills and strategies developed by artificial agents across various sensory modalities. To achieve this, we constructed a meticulously designed framework and addressed a spectrum of objectives tailored to unravel distinct facets of our research domain. Our approach involved modeling the agent-environment dynamics carefully and implementing reinforcement learning methods. We aimed to ensure that our methodology was underpinned by methodological rigor and interdisciplinary collaboration.

Our findings have illuminated the adaptability and resilience of artificial agents in navigating diverse sensory landscapes. We discovered that the concurrent utilization of ray and camera modalities emerged as a potent catalyst for optimal learning outcomes. Additionally, the Soft Actor-Critic (SAC) algorithm demonstrated paramount efficacy in steering agent performance across varied environments.

Our research also underscores the imperative of methodological rigor and interdisciplinary collaboration in unraveling the intricacies of cross-modal learning. Through rigorous experimentation and meticulous analysis, we were able to validate the potential of artificial intelligence in navigating perceptual diversity. We hope that our findings will serve as a guiding light, propelling future research endeavors towards the development of more robust and versatile artificial intelligence systems capable of transcending the confines of singular sensory modalities.

Overall, our research has contributed significantly to our understanding of the potential of artificial intelligence in handling different sensory modalities. We believe that our findings will inspire future research in this area and help to develop more robust and versatile artificial intelligence systems.

6.2 Recommendations

Upon completing the experiment, the researcher highly recommends that anyone who wishes to build upon their work should use it as a starting point. In particular, the researcher suggests that readers or other researchers should consider incorporating additional solutions for agents to improve the experiment even further. These additional solutions could include strategies for agents to more effectively carry out their tasks under challenging circumstances, such as adverse weather conditions or unexpected equipment malfunctions.

By incorporating these solutions, the experiment can be expanded upon and improved, ultimately leading to better results and a more comprehensive understanding of the subject matter. The researcher emphasizes the importance of continuously building upon the experiment in order to identify new solutions and potential areas for further research.

Overall, the researcher's suggestion to add further solutions for agents is a crucial step in advancing the field of study. It provides a solid foundation for future researchers to build upon, allowing them to create more effective and efficient strategies for agents to carry out their tasks, ultimately leading to better results and a more comprehensive understanding of the subject matter.

References

[1] E. Rovenskaya, "core.ac.uk," 05 09 2019. [Online]. Available: http://core.ac.uk.pdf. [Accessed 05 09 2023].

[2] D. H. X. G. a. P. A. Carlos Florensa, "Automatic Goal Generation forReinforcement Learning Agents," 23 Jul 2018. [Online]. Available: http://arXiv:1705.06366v5 [cs.LG]. [Accessed 25 Oct 2023].

[3] A. V. B. D. P. A. S. L. Gregory Kahn, "Berkeley AI Research (BAIR," University of California, Berkeley, 17 May 2018. [Online]. Available: http://arXiv:1709.10489v3. [Accessed 12 September 2023].

[4] J. B. H. a. V. B. Peter W. Battaglia, "Relational inductive biases, deep learning, and graph networks," 17 Oct 2018. [Online]. Available: http://06.01261v3 [cs.LG. [Accessed 18 Sep 2023].

[5] V.-P. B. a. E. T. Arthur Juliani, "Unity: A General Platform for Intelligent Agents," 6 May 2020. [Online]. Available: http://arXiv:1809.02627v2 [cs.LG] . [Accessed 12 july 2023].

[6] G. T. a. J. Sinapov, "Deep Multi-Sensory Object Category RecognitionUsing Interactive Behavioral Exploration," *2019 International Conference on Robotics and Automation (ICRA)*, vol. 1, no. Department of Computer Science, pp. 2 -95, May 20-24, 2019.

[7] Y. Z. K. S. P. S. S. L. F.-F. A. G. J. B. Michelle A. Lee, "Making Sense of Vision and Touch," 8 Mar 2019. [Online]. Available: http://arXiv:1810.10191v2 [cs.RO]. [Accessed 10 Sep 2023].

[8] K. A. S. a. J. B. T. Kelsey R. Allen, "Rapid trial-and-error learning with simulation supports flexible tool useand physical reasoning," 29 Jun 2020. [Online]. Available: http://arXiv:1907.09620v3 [cs.AI] . [Accessed 12 Aug 2023].

[9] A. R. Vijay Kanade, "spiceworks," spiceworks, 29 September 2022. [Online]. Available: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-reinforcement-learning/amp/. [Accessed 17 September 2023].

[10] M. G. a. S. Mika, "An Introduction to Reinforcemnt Learning," nyscale, 26 August 2021. [Online]. Available: https://www.anyscale.com/blog/an-introduction-to-reinforcement-learning-with-openai-gym-rllib-and-google. [Accessed 12 January 2023].

[11] A. Ave, "synopsys," synopsys, 32 march 2021. [Online]. Available: https://www.synopsys.com/ai/what-is-reinforcement-learning.html#. [Accessed 09 January 2023].

[12] P. Kadari, "Analitics Vidhya," Vidhya, 28 December 2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/02/introduction-to-reinforcement-learning-for-beginners/. [Accessed 23 october 2023].

[13] P. Wouter van Heeswijk, "towardsdatascience," towardsdatascience, 30 November 2022. [Online]. Available: https://towardsdatascience.com/proximal-policy-optimization-ppo-explained-abed1952457b. [Accessed 23 july 2023].

[14] T. H. Z. A. Levine, "Learning with a Stochastic Actor," University of California, Berkeley, USA., 8 Aug 2018. [Online]. Available: http://arXiv:1801.01290v2 . [Accessed 12 july 2023].

[15] M. Jr, "unity technologies," unity technologies, 01 Jan 2023. [Online]. Available: https://github.com/Unity-Technologies/ml-agents. [Accessed 23 Aug 2023].

[16] m. Jr, "Unity Technonologies," Unity Technonologies, 19 June 2022. [Online]. Available: https://github.com/Unity-Technologies/ml-agents/blob/develop/docs/Learning-Environment-Examples.md. [Accessed 15 Sep 2023].

[17] Deepanshut, "GitHUB," Deepanshut, 14 Nov 2022. [Online]. Available: https://deepanshut041.github.io/Reinforcement-Learning/mlagents/05_soccer_twos/. [Accessed 26 Aug 2023].

[18] I. Rehmn, "AI and Machinelearning," AI and Machine learning, 13 may 2023. [Online]. Available: https://www.linkedin.com/pulse/teaching-ai-play-using-unitys-ml-agents-toolkit-ibad-rehman?utm_source=share&utm_medium=member_android&utm_campaign=share_via. [Accessed 23 Oct 2023].

[19] M. T. Hossan, "convolutional neural network scheme-based optical camera communication system for inteligent," *neural network scheme-based optical camera*, vol. 1, no. york university, pp. 52-100, 2018.

[20] Miguelalonsojr, "github," unity technologies, 12 july 2023. [Online]. Available: https://github.http://com/Unity-Technologies/ml-agents/blob/develop/docs/Learning-Environment-Design-Agents.md. [Accessed 23 aug 2023].

[21] D. G. X. F. C. A. P. Held, "Aouomatic Goal Generation for Reinforcement Learnig Agents," *Reinforcement Learnig Agents*, 2018.