Scientific Research Publishing

Journal of Software Engineering and Applications, 2022, 15, 417-435
https://www.scirp.org/journal/jsea
ISSN Online: 1945-3124
ISSN Print: 1945-3116

Reinforcement Learning Toolkits for Gaming: A Comparative Qualitative Analysis

Mehdi Mekni, Charitha Sree Jayaramireddy, Sree Veera Venkata Sai Saran Naraharisetti

Tagliatela College of Engineering, University of New Haven, West Haven, USA
Email: mmekni@newhaven.edu, cjaya1@unh.newhaven.edu, snara8@unh.newhaven.edu

How to cite this paper: Mekni, M., Jayaramireddy, C.S. and Naraharisetti, S.V.V.S.S. (2022) Reinforcement Learning Toolkits for Gaming: A Comparative Qualitative Analysis. *Journal of Software Engineering and Applications*, **15**, 417-435.
https://doi.org/10.4236/jsea.2022.1512024

Received: October 6, 2022
Accepted: December 27, 2022
Published: December 30, 2022

Copyright © 2022 by author(s) and Scientific Research Publishing Inc.
This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).
http://creativecommons.org/licenses/by/4.0/

Open Access

Abstract

Historically viewed as a niche economic sector, gaming is now projected to exceed a global annual revenue of $218.7 billion in 2024, taking advantage of recent Artificial Intelligence (AI) advances. In recent years, specific AI techniques namely; Machine Learning (ML) and Reinforcement Learning (RL), have seen impressive progress and popularity. Techniques developed within these two fields are now able to analyze and learn from gameplay experiences enabling more interactive, immersive, and engaging games. While the number of ML and RL algorithms is growing, their implementations through frameworks and toolkits are also extensive too. Moreover, the game design and development community lacks a framework for informed evaluation of available RL toolkits. In this paper, we present a comprehensive survey of RL toolkits for games using a qualitative evaluation methodology.

Keywords

Game Design & Development, Machine Learning, Reinforcement Learning, Deep Learning

1. Introduction

Computer gaming is a growing market showing a global revenue increase of 8.7% from 2019 to 2021 to reach $218.7 billion in 2024 [1]. Many games have multiple non-player characters (NPCs) who play with the player, against them or take a neutral position within the game. They play an essential part in video games to increase the player experience and should therefore be supplied with a fitting behavior by creating a fitting Artificial Intelligence (AI) for them [2]. They can take multiple roles like providing a challenge for the player to fight against or representing a trusted ally with whom they fought many battles [3]. It is therefore important, that the field of game design and development finds new

ways to build their intelligence and let them play their role inside the game [4].

There are different AI techniques in use in modern computer games. Especially ever since the 21st century, various sorts of video games, online or offline have undergone rapid changes with the development of artificial and computational intelligence [5]. The roots of AI application in game design and development can be traced back to the 1950s when Claude Shannon (The Information Theory) and Alan Turing (Theory of Computation) began to write AI logic for chess programs [6]. In 1997, the famous computer "*Deep Blue*" which represented the pinnacle of AI techniques beat the chess Master Garry Kasparov in a publicized match [6].

It is widely accepted that decision making and pattern recognition are basic skills for humans; however, it can be challenging for computers. Sequential decision-making is a core topic in Machine Learning (ML). Moreover, a sequence of decisions taken to achieve a given goal in an environment evolves the concept of Reinforcement Learning (RL). The ability to let the AI decide on its own is a fascinating concept, and it is progressively being worked on in every field including gaming [7].

Yannakakis and Togelius [8] identified various research areas standing out within the application of AI in the gaming field. Their work aimed to offer a higher-level overview of AI applications in gaming and was more about the interactions among these applications as well as the influences they had on each other. One critical limitation of this work is that it does not capture the recent advances in ML and RL and hence does not provide a current source to study AI applications in game design and development. More recently, Shao *et al.* [9] provided a survey of the progress of Deep RL methods and compared their main techniques and properties. A major shortcoming of this study is that it exclusively focuses on Deep RL and leaves the scientific community without current state-of-the-art of ML and RL applications specific to game design and development.

Motivated by the quality of the work presented in [6] [7] [8], we aim to address the existing limitations associated with outdated studies and incomplete analysis of trending ML and RL techniques in the field of game design and development. This paper presents an insight into AI implementation in game development with an emphasis on ML and RL toolkits. It proposes a comprehensive evaluation framework using a qualitative comparison to support the community of game developers. In this study, we examine the applications of ML and RL toolkits in gaming, their challenges, as well as their trends.

The remainder of this paper is organized as follows; Section 2 provides an overview of the evolution of the global gaming industry. Section 3 introduces the fundamental concepts of ML and its sub-fields. Section 4 details the state-of-the-art of available ML and RL toolkits. Section 5 presents our qualitative evaluation methodology articulated around a specific set of technical criteria. Section 6 outlines the key evaluation analysis findings. Finally, Sections 7 and 8 discuss this study and conclude with future work.

2. Gaming Industry

The early years of the gaming industry date back to the 1970s with the introduction of arcade machines and game consoles [10]. As computer components became more affordable, companies began to explore such market opportunities in game design and development [11]. Video games are a generic term for all types of digital games, played and used on some type of screen. This includes arcade machines, handheld devices, game consoles (*i.e.*, Xbox, PlayStation, Game Boy), and computer games [12]. Stanford University in the USA hosted the first gaming tournament in 1972 giving rise to competitive video games [13]. Following attempts to increase the popularity of gaming were made during the 1980s and 1990s with the organization of national tournaments and world championships. Companies such as Atari or Nintendo used these events as a marketing tool to promote their video games, while fostering a gaming culture [14].

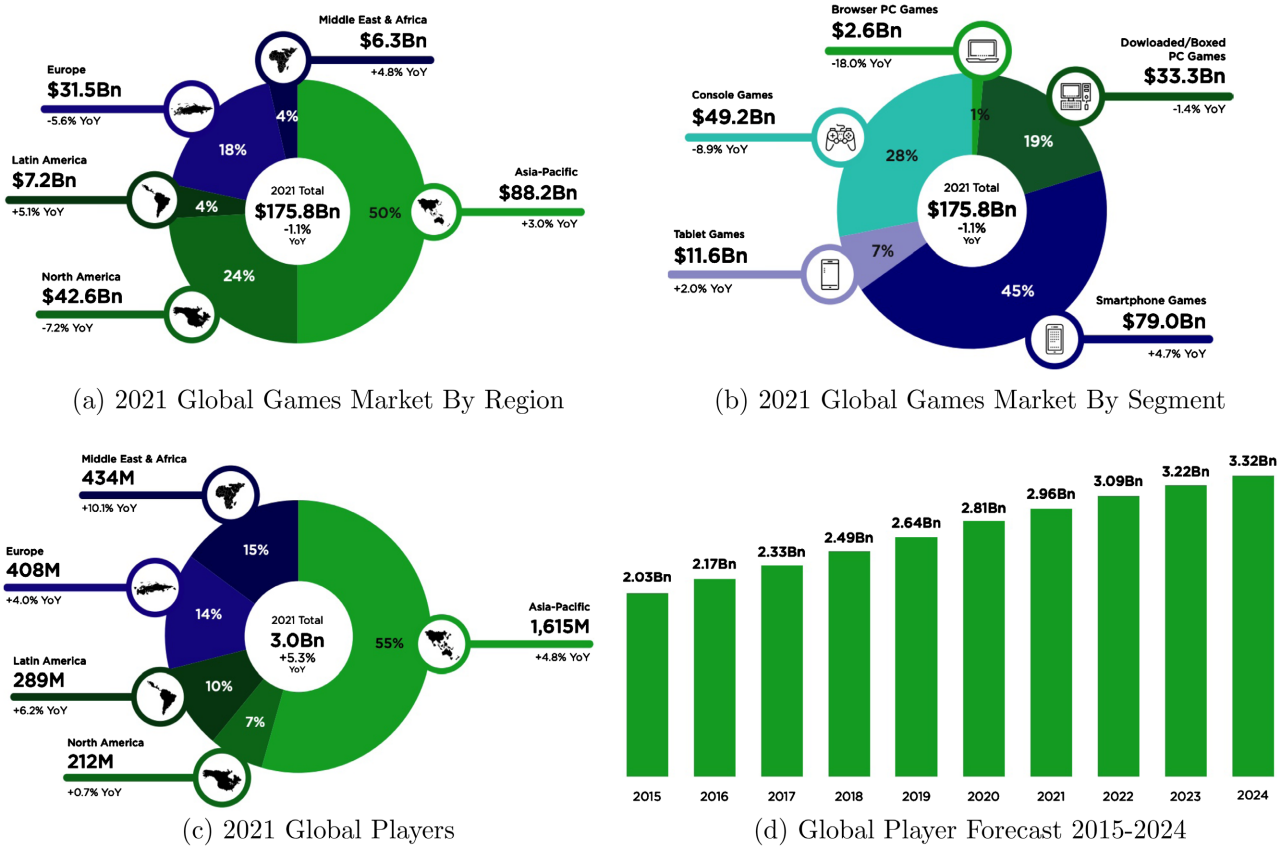
During the 1990s, with the development of the internet and further multiplayer capabilities, video games experienced significant growth, making it possible not only to connect but to compete with external players [15]. Further multiplayer tournaments began proliferating, as well as the tournament organizations across the globe (*i.e.*, Cyberathlete Professional League (CPL) and the AMD Professional Gamers League (PGL) in the USA, the Deutsche Clanliga (DeCL) in Germany, among many others in different countries and over the years) [16]. Asia-Pacific is easily the world's biggest region by games revenues, with $88.2 billion in 2021 alone, making up 50.2% of all game revenues. With its contribution of $45.6 billion, China is by far the primary driver here. North America remains 2021's second-biggest region, boasting game revenues of $42.6 billion (mainly from the U.S.) (See **Figure 1(a)**).

The recent pandemic has had a profound impact on game development and publishing in terms of delays, which are affecting revenues across the board in 2021' mostly on the console side but also on PC. Compared to mobile, console and PC games tend to have bigger teams, higher production values, and more cross-country collaborations (See **Figure 1(b)**).

There will be close to 3.0 billion players across the globe in 2021. This is up +5.3% year on year from 2020, showcasing that 2020's gaming boom has led to a lasting increase in players, with room for further growth (See **Figure 1(c)**).

Looking ahead, the global number of players will pass the 3-billion milestone next year in 2022. This number will continue to grow at a +5.6% of the compound annual growth rate (2015-2024) to 3.3 billion by 2024 (See **Figure 1(d)**).

Along with the growth of the global gaming industry and advancements in AI research, the need to figure out tough problems in existing game design and development using current benchmarks for designing, developing and training AI models (See **Figure 1**) has also increased. However, as these challenges are "solved," the need for novel interactive environments, engaging gameplay, and smart NPCs arises. Yet, creating such environments is often time-intensive and requires specialized computational and AI domain knowledge.



(a) 2021 Global Games Market By Region

(b) 2021 Global Games Market By Segment

(c) 2021 Global Players

(d) Global Player Forecast 2015-2024

Figure 1. An overview of the global gaming market over [1].

In the following section, we introduce the fundamental ML concepts aiming at boosting the game design and development field.

3. Machine Learning Concepts

3.1. Machine Learning

ML is the art of making computer programs learned from experience. A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E [17]. For example, task T can be playing checkers, experience E is playing thousands of checkers games, and the performance P is the fraction of games won against human opponents. We can divide the learning problems into three classes:

- Learning is called supervised if the experience E takes the form of a labeled dataset (x,y), the task is to learn a function that maps x to y,
- Learning is called unsupervised if E takes the form of an unlabeled dataset. The task is to learn underlying structure,
- Reinforcement learning (RL) is when the experience E takes the form of state-action pairs and corresponding rewards. The task is to maximize future rewards over a number of time steps.

Tasks are usually described in terms of how ML should process a data item

(*i.e.*, an example). If the desired behavior is to assign the input data item to one category among several, this is a classification task, e.g., object recognition. Other examples of tasks are machine translation, transcription, anomaly detection, etc [18] [19].

3.2. Reinforcement Learning (RL)

Reinforcement Learning (RL) is particularly interesting for playing games since its task involves interaction with an environment, by committing actions and receiving rewards for these actions [20]. In RL, the experience is a set of episodes. Each episode is a sequence of tuples (State, Action, Reward, Next State), the performance measure is the discounted total reward, and the task basically consists of playing (**Figure 2**). A more sophisticated description of playing is adopting a policy that maps states of the game to actions. If this mapping takes the form of a neural network, a deep one, we refer to Deep Reinforcement Learning (DRL).

3.3. Deep Reinforcement Learning (DRL)

Given an agent that interacts with an environment through percepts (observations) and actions, the goal of reinforcement learning is to find an optimal policy π^* that maximizes the expected total sum of rewards the agent receives during a run, while starting from an initial state $s_0 \in S$ [22]. Usually, the performance of a given policy π is evaluated as shown in Equation (1):

$$\text{eval}(\pi \mid s_0) = \mathbb{E}_{\rho(\pi)}\left[\sum_{t=0}^{\tau} \gamma^t r(s_{t+1})\right] = \mathbb{E}_{\rho(\pi)}\left[\mathcal{R}_0 \mid s_0\right] \tag{1}$$

where γ is a discount factor, and the expectation is over all the possible runs (or traces) allowed by the policy π. $\mathcal{R}_0$ is the total reward for $t = 0$. Among the most popular algorithms to reach an optimal policy in this context are value iteration and Q-learning. Value iteration assumes that the reward model and the transition model are known a priori. Q-learning actively learns a utility function for (State, Action) pairs [9] as detailed in Equations (2) and (3):

$$Q(s_t, a_t) = \mathbb{E}\left[\mathcal{R}_t \mid (s_t, a_t)\right] \tag{2}$$

$$\pi^*(s) = \arg\max_a Q(s, a) \tag{3}$$

By combining these ideas from reinforcement learning with the recently re-invented neural networks a new set of algorithms emerges and is dubbed

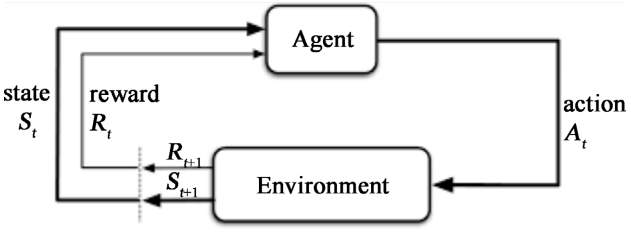


Figure 2. Classic agent-environment loop [21].

DRL. One of the seed contributions in this area is **value learning**. In [23], a Convolutional Neural Network (CNN) was trained to play Atari with a variant of Q-learning. The CNN approximates the utility function of Q-learning based on raw pixels for input and an estimation of future reward as output. The loss function for value learning is in Equation (4):

$$\mathcal{L} = \mathbb{E}\left[\left\|Q_{\text{real}}(s,a) - Q_{\text{predicted}}(s,a)\right\|\right] \tag{4}$$

where $Q_{\text{predicted}}(s,a)$ is the output of the neural network and $Q_{\text{real}}(s,a)$ is the actual Q value is in Equation (5):

$$Q_{\text{real}}(s,a) = r + \gamma Q(s',a') \tag{5}$$

Another approach is **policy learning**, where the policy is learned directly through training a neural network and without passing through value learning. Policy learning is shown to be very successful at addressing challenges of (1) large or continuous action space such as in self-driving, and (2) stochastic transition and reward models. Policy learning is based on a set of policy gradient methods with the goal of learning a probability distribution over the actions given a state $P(a \mid s)$. The training is performed through continuous running of episodes and simply increasing the probability of actions that resulted in high reward, and decreasing the probability of actions that resulted in low reward. The loss function in Equation (6):

$$\mathcal{L} = \mathbb{E}\left[-\log P(a \mid s)\mathcal{R}\right]. \tag{6}$$

3.4. Applications in Gaming

ML, RL and DRL are heavily used in gaming to develop not only competitive agents but also collaborative agents and NPCs. Alpha Go beated the top human player at Go in 2016. DeepMind introduced AlphaZero in 2017, a single system that taught itself through self-play how to master the games of chess, shogi (Japanese chess), and Go [24] [25]. MuZero, a general-purpose algorithm, was able to master Go, chess, shogi and Atari without needing to be told the rules, thanks to its ability to plan winning strategies in unknown environments [26]. A summary of these algorithms as per [27] is depicted in **Table 1**.

Similarly, AlphaStar, a multi-agent RL system, was developed to play StarCraft

Table 1. Evolution of DRL for playing board games.

DRL	Domain				Knowledge		
	Go	Chess	Shogi	Atari	Human Play	Domain Knowledge	Known Rules
ALpha Go [28]	⊗				⊗	⊗	⊗
Alpha Go Zero [29]	⊗						⊗
Alpha Zero [24]	⊗	⊗	⊗				⊗
Mu Zero [26]	⊗	⊗	⊗	⊗			

II at Grandmaster level [30]. OpenAI developed Dota 2 AI agents, called OpenAI Five, and made them learn by playing over 10,000 years of games against themselves. The agents demonstrated the ability to defeat world champions in Dota 2 [31]. Using the same RL model as OpenAI Five boosted with additional techniques, OpenAI trained a pair of neural networks to solve the Rubik's Cube with a human-like robot hand. Facebook and Carnegie Mellon built the first AI-based game that beats pros in 6-player poker [32].

4. Reinforcement Learning Toolkits

4.1. Unity ML-Agents

The Unity Machine Learning Agents Toolkit (ML-Agents) is an open-source project that enables games and simulations to serve as environments for training intelligent agents [33] [34]. The training of agents is performed using ML techniques including reinforcement learning, imitation learning, and neuroevolution [35]. There are 3 main kinds of objects in a learning environment in Unity ML-Agents:

- **Agent:** Each Agent can have a unique set of states and observations, take unique actions within the environment, and receive unique rewards for events within the environment. An agent's actions are decided by the brain it is linked to.
- **Brain:** Each Brain defines a specific state and action space, and is responsible for deciding which actions each of its linked agents will take.
- **Academy:** Each environment contains a single academy which defines the scope of the environment, in terms of *engine configuration*, *frameskip*, and *global episode length*.

With the Unity ML-Agents toolkit, a variety of training scenarios are possible, depending on how agents, brains, and rewards are connected. Despite the lack of detailed studies on Unity ML-Agents, a few games have been implemented using Unity and its ML-Agents package where the training has been done using reinforcement learning including imitation learning and self-play. **Figure 3** illustrates the Unity ML-Agents Learning Environment [36]. An AI-based agent has been implemented in *Connect*4 game using Unity ML-Agents [37]. The agent training was performed using the Proximal Policy Optimization (PPO) algorithm. Moreover, a RL model using Hierarchical Critics (RLHC) algorithm has been implemented in Unity ML-Agents which performance was compared with the PPO model using two different competitive games—*Soccer* and *Tennis* [38].

4.2. OpenAI

OpenAI is a research lab whose mission is to ensure that artificial general intelligence benefits all of humanity [21]. OpenAI provides various tools to support applications of RL and ML in scientific research and game design and development.

4.2.1. OpenAI Gym

Gym is an open-source toolkit for developing and comparing reinforcement

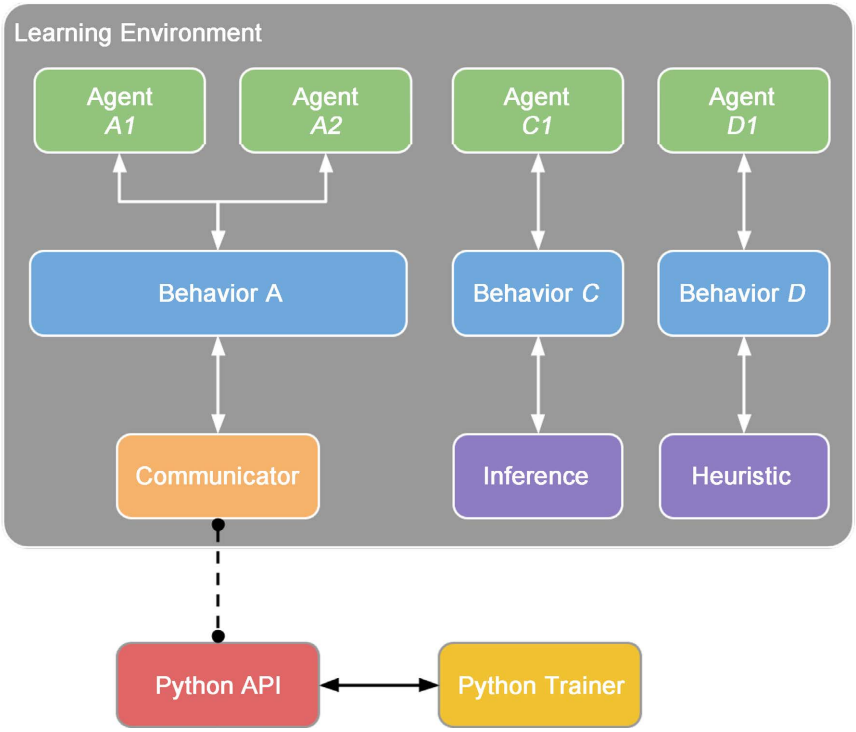


Figure 3. Unity ML-agents learning environment [36].

learning algorithms [39]. The OpenAI Gym toolkit encompasses a collection of tasks, called environments, including Atari games, board games, as well as 2D and 3D physical simulations for serious games [40]. It is used to train agents by implementing and comparing various ML and RL algorithms using shared interfaces. Therefore, OpenAI Gym is mainly used for standardization and benchmarking purposes.

4.2.2. OpenAI Safety Gym

Safety Gym is a suite of environments and tools for RL agents with safety constraints implemented while training. While training the RL agents, safety is not much focus, but in certain aspects, safety is an important concern and is to be considered. To address the safety challenges while training the RL agents and to accelerate the safe exploration research, OpenAI introduced Safety Gym. It consists of two components:

- An environment builder for creating a new environment by choosing from a wide range of physics elements, goals and safety requirements.
- Provides a suite of pre-configured benchmarks environments to choose from.

Safety Gym uses the OpenAI Gym for instantiating and interfacing with the RL environments and MuJoCo physics simulator to construct and forward-simulate each environment [41].

4.2.3. OpenAI Baselines

OpenAI Baselines is a set of high-quality implementations of RL algorithms. These algorithms make it easier for the research community to replicate, refine,

and identify new ideas, and create baselines to build research on top of. Such algorithms include Deep Q-Network (DQN) and its variants, Actor Critic using Kronecker-Factored Trust Region (ACKTR), Advantage Actor Critic (A2C), and Asynchronous Advantage Actor Critic (A3C) [42].

4.2.4. OpenAI Universe

OpenAI universe is an extension of the gym. It provides the ability to train and evaluate agents in a wide range of simple to real-time complex environments. It has unlimited access to many gaming environments. Using Universe, any program can be turned into a Gym environment without access to program internals, source code, API's as universe works by launching the program automatically behind a virtual network computing remote desktop. With support from EA, Microsoft Studios, Valve, Wolfram, and many others, openAI has already secured permission for Universe AI agents to freely access games and applications such as Portal, Fable Anniversary, World of Goo, RimWorld, Slime Rancher, Shovel Knight, SpaceChem, Wing Commander III, Command & Conquer: Red Alert 2, Syndicate, Magic Carpet, Mirror's Edge, Sid Meier's Alpha Centauri, and Wolfram Mathematica.

4.2.5. OpenAI Gym Retro

OpenAI Gym Retro enables the conversion of classic retro games into OpenAI Gym compatible environments and has integration for around 1000 games. The emulators used in OpenAI Gym Retro support Libretro API which allows the creation of games and supports various emulators [43]. It is useful primarily as a means to train RL on classic video games, though it can also be used to control those video games using Python scripts.

4.3. Petting Zoo

Petting Zoo is a python library for conducting research in multi-agent environments. Petting Zoo is a multi-agent version of OpenAI Gym. What OpenAI Gym has done with single agent reinforcement learning environments, Petting Zoo was developed with the goal of doing the same with multi-agent environments. PettingZoo's API, while inheriting many features of OpenAI Gym, is unique amongst Multi Agent Reinforcement Learning (MARL) APIs. PettingZoo models environments as Agent Environment Cycle (AEC) games, in order to be able to cleanly support all types of multi-agent RL environments under one API and to minimize the potential for certain classes of common bugs. Petting Zoo includes 63 default environments [44].

4.4. Google Dopamine

Dopamine is a TensorFlow based research framework for the fast prototyping of reinforcement learning algorithms. Dopamine supports multiple agents like DQN, SAC and these are implemented using JAX which is a Python library for high-performance ML research. Dopamine supports Atari environments and

OpenAI's MuJoCo environments [45].

5. Evaluation Methodology

In this study, we propose a qualitative evaluation methodology that uses a set of eleven specific technical criteria (See the following subsections). Each candidate ML/RL toolkit introduced in Section 1 is evaluated based on the following qualitative data collection techniques: 1) Game design and development experts interviews; 2) Technical experimentation and observations; and 3) Documentation including scientific publications and technical reports.

5.1. Portability

Portability in ML/RL toolkits is the usability of the same toolkit in different environments. The pre-requirement for portability is the generalized abstraction between the toolkit logic and its interfaces. When a ML/RL toolkit with the same functionality is developed for several environments, portability is the key issue for development cost reduction.

5.2. Interoperability

Interoperability refers to the capability of different ML/RL toolkits to communicate with one another and with game engines freely and easily. Toolkits that are interoperable exchange information in real-time, without the need for specialized or behind-the-scenes coding.

5.3. Performance

The training speed of agents in a ML/RL depends on the complexity and analysis of the algorithm used to train that agent. Booth *et al.* provide a comparison study of different algorithms including PPO in ML-Agents and A2C, ACKTR and PPO2 algorithms of OpenAI Baselines [46].

5.4. Multitask Learning

Multi-task learning is an ML/RL approach in which we try to learn multiple tasks simultaneously, optimizing multiple loss functions at once. Rather than training independent models for each task, we allow a single model to learn to complete all of the tasks at once. In this process, the model uses all of the available data across the different tasks to learn generalized representations of the data that are useful in multiple contexts.

5.5. Multi-Agent Environments

An environment might contain a single agent or multiple agents. In the case of multiple agents, each agent might have a different set of actions to perform and the agents might need interaction between them as the training goes on [47]. This requires a different training methodology from training a single agent (see **Figure 4**).

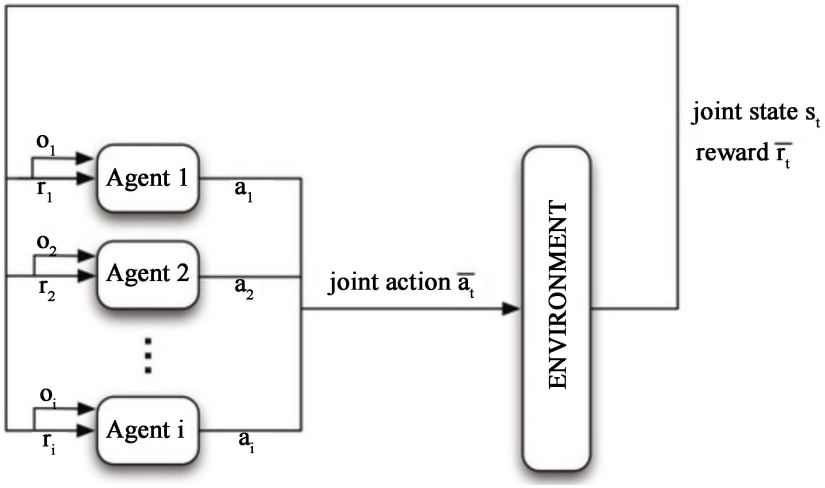


Figure 4. Multi-Agent Model [47].

5.6. Usability

Usability is a measure of how well a specific user in a specific context can use a ML toolkit to design and develop games effectively, efficiently and satisfactorily. Game designers usually measure a toolkit design's usability throughout the development process' from wireframes to the final deliverable' to ensure maximum usability.

5.7. Documentation and Support

ML/RL toolkit documentation is written text or illustration that accompanies toolkits or is embedded in the source code. The documentation either explains how the toolkit operates or how to use it. Documentation is an important part of game design and development when using ML/RL toolkits. Types of documentation include; 1) *Requirements*—Statements that identify attributes, capabilities, characteristics, or qualities of a toolkit, 2) *Architecture/Design*—Overview of the toolkit design and includes relations to an environment and construction principles to be used, 3) *Technical*—Documentation of code, algorithms, interfaces, and APIs, 4) *End user*—Manuals for the end-user, administrators and support staff, and 5) *Marketing*—How to market the product and analysis of the market demand.

5.8. Learning Strategies

The learning strategies are the different techniques ML/RL toolkits and frameworks used to train the agents in game design and development. These strategies are translated through machine learning algorithms including:

- *Naïve Bayes Classifier Algorithm* (Supervised Learning—Classification) based on Bayes' theorem and classifies every value as independent of any other value. It allows predicting a class/category, based on a given set of features, using probability.
- *K Means Clustering Algorithm* (Unsupervised Learning—Clustering) is a type of unsupervised learning, which is used to categorize unlabelled data,

i.e., data without defined categories or groups. The algorithm works by finding groups within the data, with the number of groups represented by the variable K. It then works iteratively to assign each data point to one of K groups based on the features provided.

- *Support Vector Machine Algorithm* (Supervised Learning—Classification) analyzes data used for classification and regression analysis. It essentially filters data into categories, which is achieved by providing a set of training examples, each set marked as belonging to one or the other of the two categories. This algorithm then works to build a model that assigns new values to one category or the other.
- *Linear Regression* (Supervised Learning/Regression) is the most basic type of regression. Simple linear regression allows us to understand the relationships between two continuous variables.
- *Logistic Regression* (Supervised learning—Classification) focuses on estimating the probability of an event occurring based on the previous data provided. It is used to cover a binary dependent variable that is where only two values, 0 and 1, represent outcomes.
- *Artificial Neural Networks* (Reinforcement Learning) comprise "units" arranged in a series of layers, each of which connects to layers on either side. ANNs are essentially a large number of interconnected processing elements, working in unison to solve specific problems.
- *Random Forests* (Supervised Learning—Classification/Regression) is an ensemble learning method, combining multiple algorithms to generate better results for classification, regression and other tasks. Each individual classifier is weak, but when combined with others, it can produce excellent results. The algorithm starts with a "decision tree" (a tree-like graph or model of decisions) and an input is entered at the top. It then travels down the tree, with data being segmented into smaller and smaller sets, based on specific variables.
- *Nearest Neighbours* (Supervised Learning) The K-Nearest-Neighbour algorithm estimates how likely a data point is to be a member of one group or another. It essentially looks at the data points around a single data point to determine what group it is actually in.

5.9. Reward Strategy

Reward functions describe how the agent "ought" to behave. It is an incentive mechanism that tells the agent what is correct and what is wrong using reward and punishment. The goal of agents in RL is to maximize the total rewards. Sometimes we need to sacrifice immediate rewards in order to maximize the total rewards. Reward strategy depends on the parameters a game developer setup during the creation of a game environment.

5.10. Precision and Recall

Precision is one indicator of a machine learning model's performance - the qual-

ity of a positive prediction made by the model. Precision refers to the number of true positives divided by the total number of positive predictions (*i.e.*, the number of true positives plus the number of false positives). It helps us to measure the ability to classify positive samples in the model. Precision and recall are two important model evaluation metrics. While precision refers to the percentage of relevant results, recall refers to the percentage of total relevant results correctly classified by ML/RL algorithm. Recall helps measure how many positive samples were correctly classified by the model.

5.11. Visual Observations

Visual observation extends ML/RL toolkits to allow both novice and expert game developers to quickly and easily build and deploy highly accurate and explainable ML/RL models for agents in games using image-based data. Observation gathers data through visual or technological means. Visual observation is 'direct' allowing game developers to witness the agents' behaviors firsthand in their environment.

6. Evaluation Analysis

Table 2 illustrates the outcomes of the proposed qualitative evaluation analysis with respect to the technical criteria detailed in Section 5. It is important to note that OpenAI is an open-source platform and Unity is a commercial platform. Nevertheless, Unity offers its ML Agent as an open-source toolkit. With respect to the proposed set of eleven technical criteria, it is obvious that Unity ML-Agents toolkit provides full support for most of these criteria with some limitations with regard to *Multitask Learning* and *Learning strategies*. On the other hand, OpenAI including its various tools, Petting Zoo, and Google Dopamino suffer from a critical lack of *Visual Observations* support. Moreover, OpenAI and its tools fail to fully support *Multi-Agent Environments*.

OpenAI Gym and Unity ML-Agents underlying software architectures are very similar and both provide comparable functionalities to game developers. In the scientific community, OpenAI has larger popularity compared to Unity ML-Agents as it is developed with the intent of developing, analyzing and comparing reinforcement learning algorithms whereas Unity's main purpose is to develop

Table 2. Overview of the evaluation of reinforcement learning toolkits.

	Unity ML-Agents	OpenAI				Petting Zoo	Dopamine
		Gym	Safety Gym	Universe	Gym Retro		
Portability	●	●	●	●	●	●	●
Interoperability	●	●	◐	◐	◐	●	◐
Performance	●	●	◐	●	●	●	●
Multitask Learning	○	●	○	●	◐	◐	◐
Multi-Agent Environments	●	○	○	○	○	●	○
Documentation & Support	●	●	●	◐	●	●	◐
Learning Strategies	◐	◐	◐	◐	◐	◐	◐
Reward Strategy	●	●	●	●	●	●	●
Visual Observations	●	○	○	○	○	○	○

●(Fully supported) ◐(Partially supported) ○(Not supported)

and produce enterprise-level games. OpenAI Gym and Unity ML-Agents have been used widely for implementation of RL algorithms in recent years. OpenAI Gym doesn't restrict itself to gaming, and has been used in various streams like telecommunications, optical networks and other engineering fields. Because of its wide range of options, OpenAI Gym has been used more widely than Unity ML-Agents to perform research and establish ML/RL models related benchmarking results.

The training of the game agents can be performed both in Gym and Unity. However, Gym only supports reinforcement learning for training the agents, whereas with ML-Agents, it is possible to train the games using reinforcement learning, imitation learning, and curriculum learning. The comparison between Unity ML-Agents PPO and OpenAI Baselines' PPO2 has proved this latter has scored 50% higher while training 14% slower. The Actor Critic using Kronecker-Factored Trust Region (ACKTR) algorithm and the Advantage Actor Critic (A2C) algorithm of the OpenAI Baselines trained 33% faster than Unity ML Agent [46].

Unity has a rich visual platform which is most helpful in building the environments even with a little programming experience. It has components designed for each asset and can be easily configured. On the other hand, OpenAI Gym is compatible with Tensorflow and provides rich graphs. To train more robust agents that interact at real-time to dynamic variations of the environment such as changes to the objects' attributes, Unity provides randomly sampled parameters of the environment during training (also called *Environment Parameter Randomization*). This technique is based on *Domain Randomization* which enables training agents by randomized rendering. Unity ML-Agents also allow the use of multiple cameras for visual observation. This enables agents to learn and integrate information from multiple visual streams.

7. Discussion

Unity ML-Agents offers a rich visual interface to create environments and place assets. It provides a rich panel or well established algorithms as part of ready to use default environments as illustrated in **Table 3**. Consequently, it offers more usability for game developers. Moreover, the abundant technical and functional documentation, case studies, tutorials, and technical support increase the popularity of this platform among the game design and development community. The OpenAI Gym platform allows users to compare the performance of their ML/RL algorithms. In fact, the aim of the OpenAI Gym scoreboards is not to design and develop games, but rather to foster scientific community collaboration by sharing projects and enabling meaningful ML/RL algorithm benchmark [21].

On the one hand, Unity ML-Agents toolkit allows multiple cameras to be used for observations per agent. This enables agents to learn to integrate information from multiple visual streams. This technique leverages CNN to learn from the input images. The image information from the visual observations that are provided by the Camera Sensor is transformed into a 3D Tensor which can be fed

Table 3. Default environments in unity ML-agents.

Environments	Description	Algorithms
3DBall: 3D Balance Ball	A balance-ball task, where the agent balances the ball on it's head.	PPO & SAC
GridWorld	A multi-goal version of the grid-world task. Scene contains agent, goal, and obstacles. The agent must navigate the grid to the appropriate goal while avoiding the obstacles.	PPO & SAC
PushBlock	A platforming environment where the agent can push a block around.	Imitation & PPO & SAC
Wall Jump	A platforming environment where the agent can jump over a wall.	PPO & SAC
Crawler	A creature with 4 arms and 4 forearms.	Imitation & PPO & SAC
Worm	A worm with a head and 3 body segments.	PPO & SAC
Food Collector	A multi-agent environment where agents compete to collect food.	PPO & SAC
Hallway	Environment where the agent needs to find information in a room, remember it, and use it to move to the correct goal.	Imitation & PPO & SAC
Soccer Twos	Environment where four agents compete in a 2 vs. 2 toy soccer game.	MA-POCA
Strikers Vs. Goalie	Environment where two agents compete in a 2 vs. 1 soccer variant.	PPO & SAC
Walker	Physics-based Humanoid agents with 26 degrees of freedom. The agents must move its body toward the goal direction without falling.	PPO & SAC
Pyramids	Environment where the agent needs to press a button to spawn a pyramid, then navigate to the pyramid, knock it over, and move to the gold brick at the top.	Imitation & PPO & SAC
Match 3	Simple match-3 game. Matched pieces are removed, and remaining pieces drop down. New pieces are spawned randomly at the top, with a chance of being "special".	PPO
Sorter	The Agent is in a circular room with numbered tiles placed randomly. The agent visits all the tiles in ascending order.	PPO
Cooperative Push Block	Similar to Push Block, the agents are in an area with blocks that need to be pushed into a goal.	MA-POCA
Dungeon Escape	Agents are trapped in a dungeon with a dragon, and must work together to escape. The goal is to unlock the dungeon door and leave.	MA-POCA

into the CNN of the agent policy. This allows agents to learn from spatial regularities and terrain topology in the observation images. In addition, it is possible to use visual and vector observations with the same agent with Unity ML-Agents. This powerful feature provides access to vector observations such as raycasting, real time visualization, and parallelization. Such a feature is designed with the intent of rapid AI agents implementation in video games, not for scientific research. This hinders its application to more realistic, complex and real-world use cases and serious games.

OpenAI Gym lacks the ability to configure the simulation for multiple agents. In contrast, Unity ML-Agents supports dynamic multi-agent interaction where agents can be trained using RL models through a straightforward Python API. It also provides MA-POCA (MultiAgent POsthumous Credit Assignment), which is a novel multi-agent trainer.

Petty-zoo provides a multi-agent policy gradient algorithm where agents learn a centralized critique based on the observations and actions of all agents. However, it suffers from a performance limitation when dealing with large-scale multi-agent environments. In fact, the input space of Q grows linearly with the number of agents N [48].

Finally, Table 4 provides a summary of common algorithms used in reinforcement learning gaming toolkits. Unity ML-Agents does not support multi-task learning. However, it offers multiple interacting agents with independent reward signals sharing common Behavior Parameters. This technique offers game developers the ability to mimic multitask learning by implementing a single agent model and encoding multiple behaviors using *HyperNetworks*.

8. Conclusions and Future Work

In this paper, we provided an overview of the main ML and RL toolkits for game design and development. OpenAI and its rich suite of tools provide a solid option for AI-based agent implementation and training with respect to a large panel of supported RL algorithms. Yet, Unity ML-Agents remains a recommended toolkit for rapid AI-based game development using limited yet pre-trained RL models. The proposed qualitative evaluation methodology used a set of specific technical criteria. Each candidate toolkit has been evaluated based on the following qualitative data collection techniques including interviews, observations, and documentation. Qualitative methodologies provide contextual data to explain complex issues by explaining the "why" and "how" behind the "what." However, the limitations of such a methodology include the lack of generalizability, the time-consuming and costly nature of data collection in addition to the difficulty and complexity of objective data analysis and interpretation.

Table 4. Common algorithms used in reinforcement learning gaming toolkits.

Algorithms	OpenAI Gym	Unity ML-Agents	Google Dopamine
A2C	Yes		
ACER	Yes		
ACKTR	Yes		
DDPG	Yes		
DQN	Yes		Yes
GAIL	Yes	Yes	
PPO	Yes	Yes	
HER	Yes		
SAC	Yes	Yes	Yes
C51	Yes		Yes
Rainbow	Yes		Yes
IQN	Yes		Yes

To address the limitations of our qualitative evaluation approach, our future work will focus on empirical and quantitative evaluations to verify, validate, and confirm our qualitative findings. A mixed method design with both qualitative and quantitative data will involve statistical assessments of existing RL toolkits to measure complexity, CPU and memory usage, scalability, and other relevant software quality attributes.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Newzoo (2021) Global Games Market Report 2021.

[2] Tazouti, Y., Boulaknadel, S. and Fakhri, Y. (2022) Design and Implementation of ImALeG Serious Game: Behavior of Non-Playable Characters (NPC). In: Saeed, F., *et al.*, Eds., *Advances on Smart and Soft Computing*, Springer, Berlin, 69-77. https://doi.org/10.1007/978-981-16-5559-3_7

[3] Yannakakis, G.N. (2012) Game AI Revisited. *Proceedings of the 9th Conference on Computing Frontiers*, Caligari, 15-17 May 2012, 285-292. https://doi.org/10.1145/2212908.2212954

[4] Yohanes, D.N. and Rochmawati, N. (2022) Implementasi Algoritma Collision Detection dan A*(A Star) pada Non Player Character Game World of New Normal. *Journal of Informatics and Computer Science*, **3**, 322-333. https://doi.org/10.26740/jinacs.v3n03.p322-333

[5] Frank, A.B. (2022) Gaming AI without AI. *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*. https://doi.org/10.1177/15485129221074352

[6] Lyle, D., *et al.* (2022) Chess and Strategy in the Age of Artificial Intelligence. In: Lai, D., Ed., *US-China Strategic Relations and Competitive Sports*, Springer, Berlin, 87-126. https://doi.org/10.1007/978-3-030-92200-9_5

[7] Sweetser, P. and Wiles, J. (2002) Current AI in Games: A Review. *Australian Journal of Intelligent Information Processing Systems*, **8**, 24-42.

[8] Yannakakis, G.N. and Togelius, J. (2014) A Panorama of Artificial and Computational Intelligence in Games. *IEEE Transactions on Computational Intelligence and AI in Games*, **7**, 317-335. https://doi.org/10.1109/TCIAIG.2014.2339221

[9] Shao, K., Tang, Z., Zhu, Y., Li, N. and Zhao, D. (2019) A Survey of Deep Reinforcement Learning in Video Games.

[10] Palma-Ruiz, J.M., Torres-Toukoumidis, A., Gonzalez-Moreno, S.E. and Valles-Baca, H.G. (2022) An Overview of the Gaming Industry across Nations: Using Analytics with Power Bi to Forecast and Identify Key Influencers. *Heliyon*, **8**, e08959. https://doi.org/10.1016/j.heliyon.2022.e08959

[11] Bornemark, O. (2013) Success Factors for e-Sport Games. *Umeå's 16th Student Conference in Computing Science*, 1-12.

[12] Gonzalez-Moreno, M.S.E., Montalvo, J.A.C. and Palma-Ruiz, J.M. (2019) La industria cultural y la industria de los videojuegos. In: *Juegos y Sociedad: Desde La Interaccio'N a la Inmersion Para el Cambio Social*, McGraw Hill, New York, 19-26.

[13] Li, R. (2017) Good Luck Have Fun: The Rise of eSports. Simon and Schuster, New York.

[14] Borowy, M., *et al.* (2013) Pioneering eSports: The Experience Economy and the Marketing of Early 1980s Arcade Gaming Contests. *International Journal of Communication*, **7**, 2254-2275.

[15] Saiz-Alvarez, J.M., Palma-Ruiz, J.M., Valles-Baca, H.G. and Fierro-Ramırez, L.A. (2021) Knowledge Management in the eSports Industry: Sustainability, Continuity, and Achievement of Competitive Results. *Sustainability*, **13**, Article No. 10890. https://doi.org/10.3390/su131910890

[16] Scholz, T.M., Scholz, T.M. and Barlow (2019) eSports Is Business. Springer, Berlin. https://doi.org/10.1007/978-3-030-11199-1

[17] Jorda, M.I. and Mitchell, T.M. (2015) Machine Learning: Trends, Perspectives, and Prospects. *Science*, **349**, 255-260. https://doi.org/10.1126/science.aaa8415

[18] Bertens, P., Guitart, A., Chen, P.P. and Perianez, A. (2018) A Machine-Learning Item Recommendation System for Video Games. 2018 *IEEE Conference on Computational Intelligence and Games*, Maastricht, 14-17 August 2018, 1-4. https://doi.org/10.1109/CIG.2018.8490456

[19] Vondrek, M., Baggili, I., Casey, P. and Mekni, M. (2022) Rise of the Metaverse's Immersive Virtual Reality Malware and the Man-in-the-Room Attack & Defenses. *Computers & Security*, **238**, Article ID: 102923. https://doi.org/10.1016/j.cose.2022.102923

[20] Tucker, A., Gleave, A. and Russell, S. (2018) Inverse Reinforcement Learning for Video Games.

[21] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J. and Zaremba, W. (2016) Openai Gym.

[22] Duryea, E., Ganger, M. and Hu, W. (2016) Exploring Deep Reinforcement Learning with Multi q-Learning. *Intelligent Control and Automation*, **7**, 129-144. https://doi.org/10.4236/ica.2016.74012

[23] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M. (2013) Playing Atari with Deep Reinforcement Learning.

[24] Silver, D., Hubert, T., Schrittwieser, J., *et al.* (2018) A General Reinforcement Learning Algorithm That Masters Chess, Shogi, and Go through Self-Play. *Science*, **362**, 1140-1144. https://doi.org/10.1126/science.aar6404

[25] Samara, F., Ondieki, S., Hossain, A.M. and Mekni, M. (2021) Online Social Network Interactions (OSNI): A Novel Online Reputation Management Solution. 2021 *IEEE International Conference on Engineering and Emerging Technologies*, Istanbul, 27-28 October 2021, 1-6. https://doi.org/10.1109/ICEET53442.2021.9659615

[26] Schrittwieser, J., Antonoglou, I., Hubert, T., *et al.* (2020) Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model. *Nature*, **588**, 604-609. https://doi.org/10.1038/s41586-020-03051-4

[27] Andrew, A.M. (1999) Reinforcement Learning: An Introduction by Richard S. Sutton and Andrew G. Barto, Adaptive Computation and Machine Learning Series, MIT Press (Bradford Book), Cambridge, Mass., 1998, xviii+ 322 pp, ISBN 0-262-19398-1, (Hardback, £ 31.95). *Robotica*, **17**, 229-235. https://doi.org/10.1017/S0263574799211174

[28] Silver, D., Huang, A., Maddison, C.J., *et al.* (2016) Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, **529**, 484-489. https://doi.org/10.1038/nature16961

[29] Silver, D., Schrittwieser, J., Simonyan, K., *et al.* (2017) Mastering the Game of Go without Human Knowledge. *Nature*, **550**, 354-359.

https://doi.org/10.1038/nature24270

[30] Arulkumaran, K., Cully, A. and Togelius, J. (2019) Alphastar: An Evolutionary Computation Perspective. *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, Prague, 13-17 July 2019, 314-315. https://doi.org/10.1145/3319619.3321894

[31] Berner, C., Brockman, G., Chan, B., *et al.* (2019) Dota 2 with Large Scale Deep Reinforcement Learning.

[32] Sweeney, N. and Sinclair, D. (2012) Applying Reinforcement Learning to Poker. *Computer Poker Symposium*, Quebec, 314-315.

[33] Nandy, A. and Biswas, M. (2018) Machine Learning Agents and Neural Network in Unity. In: Nandy, A. and Biswas, M., Eds., *Neural Networks in Unity*, Springer, Berlin, 69-111. https://doi.org/10.1007/978-1-4842-3673-4_3

[34] Jayaramireddy, C.S., Naraharisetti, S.V., Nassar, M. and Mekni, M. (2023) A Survey of Reinforcement Learning Toolkits for Gaming: Applications, Challenges and Trends. In: Arai, K., Ed., *Proceedings of the Future Technologies Conference*, Springer, Berlin, 165-184. https://doi.org/10.1007/978-3-031-18461-1_11

[35] Lanham, M. (2018) Learn Unity ML-Agents-Fundamentals of Unity Machine Learning: Incorporate New Powerful ML Algorithms Such as Deep Reinforcement Learning for Games. Packt Publishing Ltd., Birmingham.

[36] Juliani, A., Berges, V.-P., Teng, E., *et al.* (2018) Unity: A General Platform for Intelligent Agents.

[37] Baby, N. and Goswami, B. (2019) Implementing Artificial Intelligence Agent within Connect 4 Using Unity3D and Machine Learning Concepts. *International Journal of Recent Technology and Engineering*, **7**, 193-200.

[38] Cao, Z. and Lin, C.-T. (2021) Reinforcement Learning from Hierarchical Critics. *IEEE Transactions on Neural Networks and Learning Systems*, 1-8. https://doi.org/10.1109/TNNLS.2021.3103642

[39] Borovikov, I., Harder, J., Sadovsky, M. and Beirami, A. (2019) Towards Inter-Active Training of Non-Player Characters in Video Games.

[40] Silver, T. and Chitnis, R. (2020) Pddlgym: Gym Environments from Pddl Problems.

[41] Ray, A., Achiam, J. and Amodei, D. (2019) Benchmarking Safe Exploration in Deep Reinforcement Learning. Vol. 7.

[42] Dhariwal, P., Hesse, C., Klimov, O., *et al.* (2022) OpenAI Baselines.

[43] Nichol, A., Pfau, V., Hesse, C., Klimov, O. and Schulman, J. (2018) Gotta Learn Fast: A New Benchmark for Generalization in rl.

[44] Terry, J., Black, B., Grammel, N., *et al.* (2021) PettingZoo: Gym for Multi-Agent Reinforcement Learning. *Proceedings of the* 20*th International Conference on Autonomous Agents and Multiagent Systems* (*AAMAS* 2021), London, 3-7 May 2021, 441-470.

[45] Castro, P.S., Moitra, S., Gelada, C., *et al.* (2018) Dopamine: A Research Framework for Deep Reinforcement Learning. http://arxiv.org/abs/1812.06110

[46] Booth, J. and Booth, J. (2019) Marathon Environments: Multi-Agent Continuous Control Benchmarks in a Modern Video Game Engine.

[47] Nowe, A., Vrancx, P. and Hauwere, Y.-M.D. (2012) Game Theory and Multiagent Reinforcement Learning. In: Wiering, M. and Otterlo, M., Eds., *Reinforcement Learning*, Springer, Berlin, 441-470. https://doi.org/10.1007/978-3-642-27645-3_14

[48] Lowe, R., Wu, Y., Tamar, A., *et al.* (2017) Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments.