Train Multiplayer First-Person Shooter Game Agents

Subjects: Computer Science, Artificial Intelligence

Contributor: Pedro Almeida , Vítor Carvalho , Alberto Simões

Artificial Intelligence bots are extensively used in multiplayer First-Person Shooter (FPS) games. By using Machine Learning techniques, we can improve their performance and bring them to human skill levels.

reinforcement learning unity first-person shooter games

1. Introduction

Recently, many breakthroughs have been made in the creation of Artificial Intelligence (AI) agents powered by Reinforced Machine Learning [1]. New platforms have made Reinforcement Learning (RL) more accessible [2], new algorithms trained agents more capably, and new training architectures created new ways to train those agents.

Video games have always been used as a benchmark for testing new AI techniques, and as the years pass, more and more games are solved by AI algorithms, as is the case with the classic games of Go [3], Chess [4], and Shogi [4]. RL has proved to be developed enough to play 2D arcade games such as Breakout, Pong, and Space Invaders at a human level from scratch [5], but it is still being developed for facing complex 3D environments. However, it has been proven that it is possible to apply RL to more simple First-Person Shooter (FPS) games [1][6].

FPS games are one of the more popular genres for developing RL agents in these 3D environments due to its nature of placing the camera on the agent's perspective, thus facilitating the use of visual recognition, and more closely mimicking what a real-life robot would require when functioning in a real environment [1].

2. First-Person Shooters (FPS)

FPS games are a sub-genre of action games that are played from the first-person point of view that usually involve one or more ranged weapons and allow the player to fully navigate a 3D environment. The major focus of games like these are usually combat, although they can also have narrative and puzzle elements to them. They allow the player to freely control their character's movement, aim, and shooting, often in fast-paced and intense scenarios [7].

Many of the games in this genre have a multiplayer component, where players can play against each other or against AI-controlled opponents in various formats such as duels, free-for-all, or team-based modes.

Games in this genre include Doom (Id Software, 1993), Counter-Strike (Valve, 2000), Halo (Bungie, 2001), and Call of Duty (Infinity Ward, 2003).

3. Machine Learning (ML)

The field of ML focuses on developing programs that learn how to perform a task, as opposed to the traditional approach of developing programs with hardcoded rules on how to perform a task. With ML techniques, a program can adapt to changes in its environment without needing manual changes [8].

A good example of how ML thrives is in problems that are too complex for traditional methods, such as the spam filter [9]. An ML program analyses words in emails flagged as spam, finding patterns and learning by itself how to identify future spam mail. If the spam filter was run through the traditional programming approach, the designers would have to update the program each time the spam mail changed patterns.

3.1. Neural Networks

A neural network's purpose is to simulate the mechanism of learning in biological organisms [10]. Nervous systems contain cells referred to as "Neurons", which are connected to one another through axons and dendrites, and these connections are referred to as synapses. The strength of the synapses changes in response to external stimulation, and these changes are how learning takes place in living organisms.

This process is simulated in artificial neural networks, which also contain "neurons", in the form of computation units [11]. These neurons are organised into three main types of layers: input, hidden, and output layers. Data are fed into the network through the input layer, and they propagate through the hidden layers where computations occur. The output layer then produces the network's predictions or results [12].

In modern times, neural networks are becoming more and more popular in multiple areas and many organisations are investing in them to solve their problems. Neural networks can be found in a variety of places, which include computing, science, engineering, medicine, environmental, agriculture, mining, technology, climate, business, arts, and nanotechnology, among others [10].

3.2. Deep Learning

A subfield of ML, Deep Learning refers to the use of artificial neural networks with multiple layers in their networks, which can better process high levels of raw inputs. These Deep Learning neural networks can be commonly found being used in modern uses of neural networks such as image processing programs like face recognition and image generation, smart assistants such as Siri/Alexa, suggestion algorithms, and many more. Most of these state-of-the-art programs require inputting large amounts of data into the neural network, and as such, they are classified as Deep Learning [13].

4. Reinforcement Learning (RL)

RL is a subfield of ML that focuses on teaching an agent to make sequential decisions in an environment to maximise its long-term rewards. It is inspired by how humans and animals learn through interactions with the world. RL places an agent in an environment, carrying sensors to check its state, and gives it a set of actions that it can perform, as seen in **Figure 1**. The agent then tries out those actions by trial-and-error, so that it can develop its control policy and maximise rewards based on its performed actions [14].

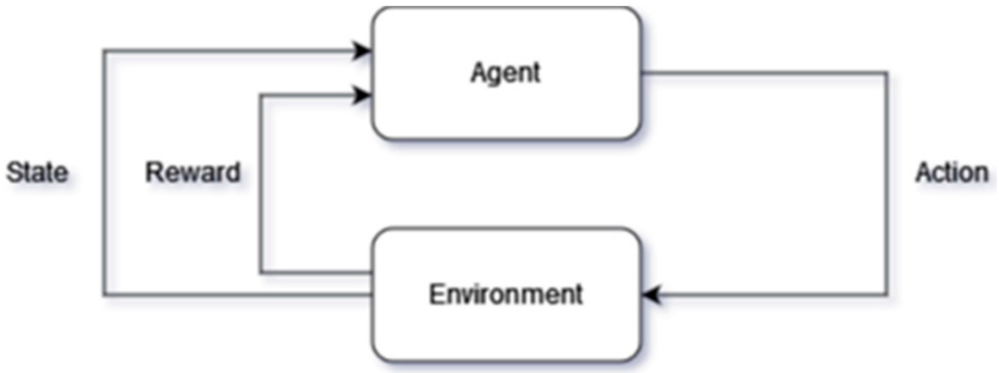

Figure 1. Agent interaction with the environment in RL.

RL is different from both Supervised and Unsupervised Learning, as it does not receive any pre-labelled data and it also is not trying to find a hidden structure, but instead working its way to maximising the reward value [15].

RL is made up of several components such as the agent, the environment, the actions, the policy, and the reward signal.

There is also a deep version of RL, called Deep Reinforcement Learning (DRL) [16].

Reinforcement Learning Components

When constructing an RL scenario, there are many components that one should keep in mind [15]:

- RL Agent

The agent is the entity that is being trained on the environment, with various training agents contributing to designing and refining the control policy. The agent monitors the state of the environment and performs actions.

- RL Environment

The environment is the space that the agent is in and can interact with and changes according to the agent's actions. It sends back feedback to the agent in the form of a reward signal.

- Actions of the RL Agent

The actions are the choices available to the agent—the agent selects actions based on the control policy, which influences the environment and generates a reward signal.

- RL Policy

The control policy is a map of the agent's action selection—it represents the behaviour or strategy of an agent in an environment. Moreover, it defines the mapping between states and actions, indicating what action the agent should take when it is in a particular state. The goal of RL is to find an optimal policy that maximises a notion of cumulative reward signal or value over time.

- RL Reward Signal

The reward signal is a numeric value that defines the goal for the agent. When the agent performs certain actions, reaches goals, or makes mistakes, the environment sends the agent a reward value, which can be positive, negative, or zero.

5. Deep Reinforcement Learning (DRL)

DRL is achieved by combining Deep Learning techniques with RL. While RL considers the problem of an agent learning to make decisions by trial-and-error, DRL incorporates Deep Learning into the solution, which allows the input of large quantities of data, such as all the pixels in a frame, and still manages to decide which action to perform [16]. In **Figure 2**, we can see how the added Deep Neural Network (DNN) works with RL.

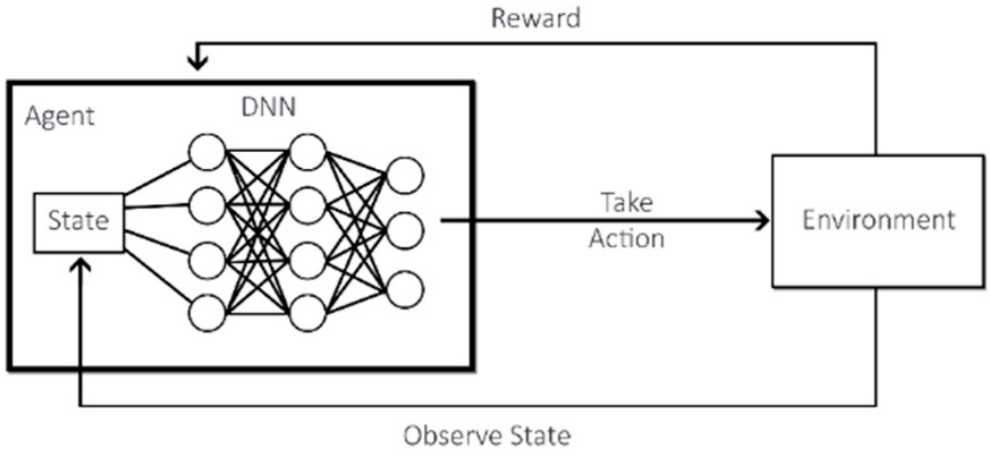

Figure 2. DRL agent interaction with the environment.

6. Training Architectures

A training architecture is how the designer trains their agents; agents can be trained alone versus traditional AI, against themselves, or even against or with other agents. They can also be trained using Curriculum Learning [17] and with Behaviour Cloning [18].

6.1. Single-Agent Reinforcement Learning

Single-Agent RL is a branch of ML that focuses on the interaction between an agent and its environment. In single-agent RL environments, there is one agent learning by interacting with either just the environment without AI or against traditional AI agents [19], as is the case in [5], where the agent learns to play many Atari arcade games.

6.2. Multi-Agent Reinforcement Learning

Multi-agent RL focuses on scenarios where numerous agents learn and interact in a shared environment. As shown in **Figure 3**, each agent is an autonomous entity that observes the environment, takes actions, and receives rewards based on its own actions and the actions of other agents. It can take the form of multiple scenarios, be cooperative with each other or competitive with each other in a one-vs.−one scenario or a team-vs.-team scenario [19].

Figure 3. How multi-agent RL has multiple agents each controlling one player, acting independently from each other but still contributing to the same policy.

6.3. Self-Play

Self-play is a technique often used in RL that involves having RL agents playing against themselves to improve performance. As seen in **Figure 4**, a single agent acts as all players, learning from the outcomes of its own actions. Self-play has been successfully applied in [4], where researchers used this method to develop their Chess- and Shogi-playing AI.

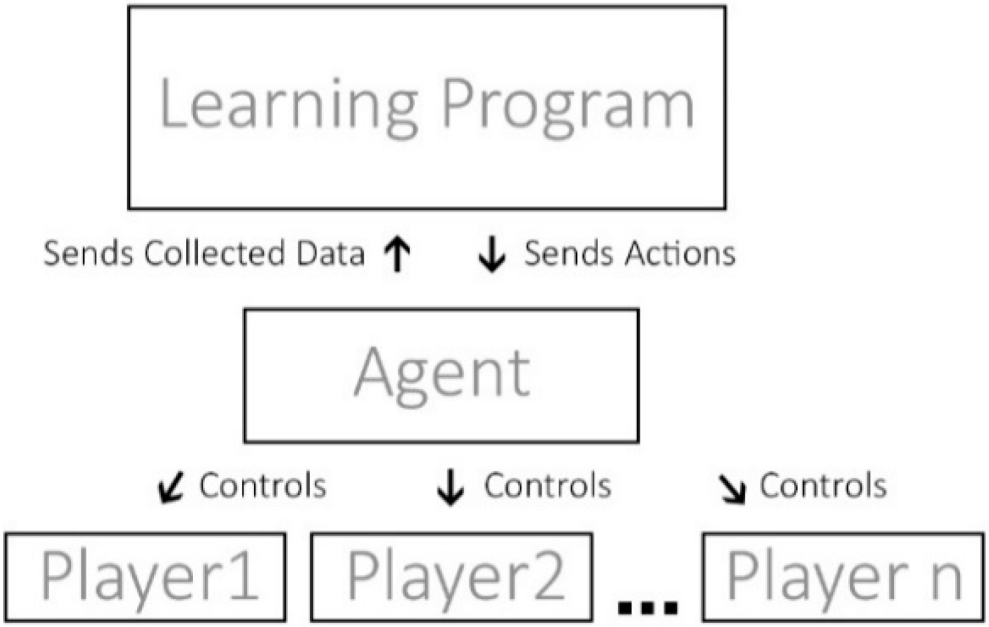

Figure 4. How self-play puts an agent in control of various players.

6.4. Behaviour Cloning

A form of imitation learning, Behaviour Cloning involves capturing the actions of a human performer and inputting them into a learning program. The program will then output rules that help agents reproduce the actions of the performer [18]. In video games, this usually means having a human player play in the designed environment, where their actions are recorded and then used to train the agent's policy. The more diverse the recording data, the better.

6.5. Curriculum Learning

Curriculum Learning architecture mimics human training by gradually increasing training difficulty. In Supervised Learning, this means increasing the complexity of training datasets; while in RL, it means increasing the complexity of the environment and task that the agent is required to perform [17].

In practical terms, this means that, for example, if one is training an agent on how to jump over a wall, they might want to start by having a wall with no height, and as the agent accumulates reward, the wall starts getting taller, as shown in **Figure 5** [20][21]. At the beginning of the training, the agent has no prior knowledge of the task, so it starts exploring the environment to learn a policy and randomly tries out things. It will eventually reach the goal, understand its objective, and progressively improve at jumping over higher and higher walls [20].

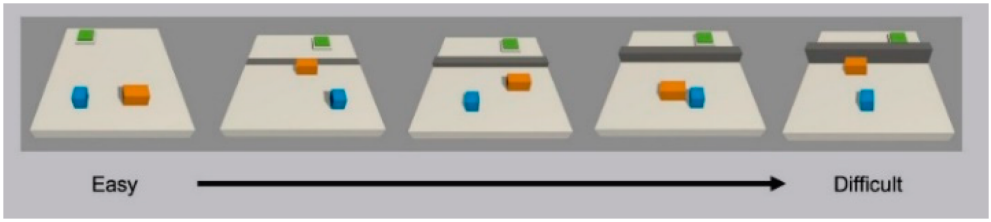

Figure 5. A Curriculum Learning environment where the agent must jump over a progressively higher wall; blue and orange objects are 2 agents, the grey object is the wall, and in green is the agents' target.

7. Unity

The Unity engine is a cross-platform multimedia platform made for creating 3D and 2D games, simulations, and other experiences [22][23]. Unlike other previously mentioned platforms, Unity is a standalone general platform, meaning that users can freely create their own environments with many more customised parameters than the alternatives. Unity contains its own physics engine and dedicated tools to create commercial 3D and 2D games, as well as a tool to create RL agents—the ML-agents toolkit [2]. Furthermore, the Unity's in-engine editor is easy and fast to use, allowing for quick prototyping and development of environments [23].

7.1. Unity's Features

Nvidia PhysX engine integration—Unity comes out of the box integrated with the PhysX physics software created by Nvidia allowing users to simulate complex state-of-the-art physics, mimicking real world interactions [23].

Simple to use, yet flexible—compared to alternative AI research platforms such as ViZDoom and DeepMind Lab [23], Unity's interface is simpler and easier to use. As seen in **Figure 6**, it allows the user to control all the environment's aspects either through its menus or programmatically. As Unity is a standalone engine meant for game development instead of a modified open-source engine such as ViZDoom, it allows much better control of its game environment [23].

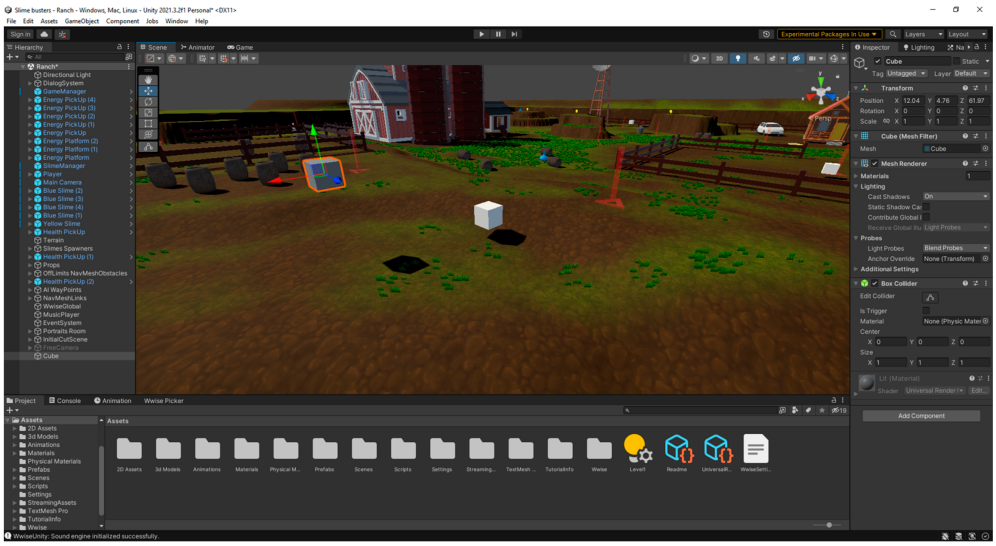

Figure 6. The Unity engine's interface, with a scene being worked on. On the right is the selected object's properties, on the left, the list of objects in the scene, and below, the list of assets in the whole project.

7.2. ML-Agents Toolkit

The ML-agents toolkit is an open-source project that allows researchers and developers to use environments created in the Unity engine as training grounds for ML agents by connecting via a Python API [2]. The toolkit features support training single-agent, multi-agent cooperative, and multi-agent competitive scenarios with the use of several RL algorithms such as Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC) [2].

References

1. McPartland, M.; Gallagher, M. Reinforcement Learning in First Person Shooter Games. IEEE Trans. Comput. Intell. AI Games 2011, 3, 43–56.
2. Unity Team. The ML-Agent's Github Page. Available online: https://github.com/Unity-Technologies/ml-agents (accessed on 20 June 2023).
3. Silver, D.; Huang, A.; Maddison, C.; Guez, A.; Sifre, L.; Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of Go with deep neural networks and tree search. Nature 2016, 529, 484–489.
4. Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. Science 2018, 362, 1140–1144.
5. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.; Veness, J.; Bellemare, M.; Graves, A.; Riedmiller, M.; Fidjeland, A.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. Nature 2015, 518, 529–533.
6. Glavin, F.; Madden, M. Learning to Shoot in First Person Shooter Games by Stabilizing Actions and Clustering Rewards for Reinforcement Learning. In Proceedings of the 2015 IEEE Conference on Computational Intelligence and Games (CIG), Tainan, Taiwan, 31 August–2 September 2015; pp. 344–351.
7. Elias, H. First person shooter: The subjective cyberspace. In Proceedings of the ISEA2008, Singapore, 25 July–3 August 2008.
8. Marsland, S. Machine Learning: An Algorithmic Perspective, 2nd ed.; CRC Press: Boca Raton, FL, USA, 2014.
9. Géron, A. Hands-On Machine Learning with Scikit-Learn and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2017.
10. Abiodun, O.; Jantan, A.; Omolara, A.; Dada, K.; Mohamed, N.; Arshad, H. State-of-the-art in artificial neural network applications: A survey. Heliyon 2018, 4, e00938.
11. Aggarwal, C. Neural Networks and Deep Learning: A Textbook, 1st ed.; Springer: Cham, Switzerland, 2018.

12. Goodfellow, I.; Bengio, Y.; Courville, A. Deep Learning; MIT Press: Cambridge, MA, USA, 2016.
13. Ketkar, N.; Moolayil, J. Deep Learning with Python: Learn Best Practices of Deep Learning Models with PyTorch, 2nd ed.; Apress: Berkeley, CA, USA, 2021.
14. Mitchel, T. Machine Learning; McGraw-Hill: New York, NY, USA, 1997.
15. Sutton, R.; Barto, A. Reinforcement Learning—An Introduction, 2nd ed.; The MIT Press: Cambridge, MA, USA, 2018.
16. François-Lavet, V.; Henderson, P.; Islam, R.; Bellemare, M.; Pineau, J. An introduction to deep reinforcement learning. Found. Trends® Mach. Learn. 2018, 11, 219–354.
17. Soviany, P.; Ionescu, R.; Rota, P.; Sebe, N. Curriculum Learning: A Survey. arXiv 2022, arXiv:2101.10382.
18. Sammut, C. Behavioral Cloning. In Encyclopedia of Machine Learning and Data Mining, 2nd ed.; Sammut, C., Webb, G., Eds.; Springer: Boston, MA, USA, 2017; pp. 120–124.
19. Serafim, P.; Nogueira, Y.; Vidal, C.; Neto, J. Evaluating competition in training of Deep Reinforcement Learning agents in First-Person Shooter games. In Proceedings of the 17th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames), Foz do Iguaçu, Brazil, 29 October–1 November 2018; pp. 117–11709.
20. Adamsson, M. Curriculum Learning for Increasing the Performance of a Reinforcement Learning Agent in a Static First-Person Shooter Game. Master's Thesis, KTH University, Stockholm, Sweden, 2018.
21. Juliani, A. Introducing ML-Agents Toolkit v0.2: Curriculum Learning, New Environments, and More —Unity blog. 8 December 2018. Available online: https://blog.unity.com/community/introducing-ml-agents-v0-2-curriculum-learning-new-environments-and-more (accessed on 20 June 2023).
22. Unity Team. Unity Engine's Official Site. Available online: https://unity.com/ (accessed on 20 June 2023).
23. Juliani, A.; Berges, V.; Teng, E.; Cohen, A.; Harper, J.; Elion, C.; Goy, C.; Gao, Y.; Henry, H.; Mattar, M.; et al. Unity: A General Platform for Intelligent Agents. arXiv 2020, arXiv:1809.02627.