# Game Brains: NPCs Intelligence Using Neural Network Brains

Mosaddiq Billah[1], Aanoora Seher, Ahmad Bahar, Muniba Ashfaq, Abdullah Hamid
Computer Systems Engineer (UET Peshawar)
*Correspondence:20pwcse1863@uetpeshawar.edu.pk

**Citation |** Billah. M, Seher. A, Bahar. A, Ashfaq. M, Hamid. A, "Game Brains: NPCs Intelligence Using Neural Network Brains", IJIST, Special Issue. pp 74-82, May 2024
**Received |** May 04, 2024, **Revised |** May 09, 2024, **Accepted |** May 16, 2024, **Published |** May 21, 2024.

This paper aims to develop the foundational knowledge about the Unity game development engine embedded with AI for the development of a hyper-casual game that has intelligent NPCs, which operate strategically in the environment. The targeted audience comes in the class of those who are pursuing their career in the niche of AI game development and enhancing the gaming experience for single-player game users. Using Unity Engine and Python, Curriculum learning and self-learning experiments were conducted to test the AI game. Moreover, in this paper, different reinforcement learning methods have been discussed, which have been implemented in the game that produces the optimal results for the behavior of NPCs. Hence, this paper tends to represent a glimpse into the future perspective of the gaming industry in hyper-casual gaming platforms.

**Keywords:** Unity; Reinforcement Learning; AI Games; NPCs, and Agents.

**Introduction:**

In traditional games, the behavior of NPCs (Non-Player Characters) follows a scripted, generic standard flow, leading them to perform actions in a predictable manner. Consequently, users disengage from such games after a while since they anticipate the patterns of the NPCs' movements. However, this paper tends to introduce AI to the platforms of hyper-casual games, since AI is mostly found in high-level games such as Red Dead Redemption 2. Our paper aims to develop a hypercausal game, which is a lightweight game with fewer mechanics that would have intelligent NPCs, using Unity Engine.

Unity is a cross-platform game engine, used for 2D and 3D games. It supports a variety of platforms such as desktops and mobiles. Furthermore, Unity is one of the most popular engines among the other game development engines due to its flexibility, efficiency, and convenience. The gaming engine comprises various tools that are convenient tools for modifying your project. The feature of real-time play mode with smart previews allows you to monitor the modifications instantly [1]. Unity engine supports the deployment of its projects on different operating systems such as Mac, Linux, and Windows, along with artist-friendly tools by allowing developers to develop efficient games [2]. Additionally, the navigation system in Unity enables NPCs to logically move around the environment of the game [3]. However, this feature only limits the movement of the NPCs or agents to make decisions in terms of movement around obstacles in the environment. On the scripting aspect of the Unity engine, it supports C#, which is an object-oriented programming language developed by Microsoft. There are two ways to design C# scripts in the Unity engine. The first one is object-oriented design, which is referred to as the traditional approach and is used by the majority of developers. The second is data-oriented design, which is also supported by Unity [4]. AI games have been on the rise since 1960 when games such as "SpaceWars" introduced AI in the field of game development. In contemporary times, games like Red Dead Redemption 2 and Grand Theft Auto 6 have been found to be popular due to their AI-trained NPCs.

The paper has been divided into four sections. Section 1 elaborates on the introduction, where the problem statement, literature review, recent advancements, and objectives of the research have been elaborated. Section 2 covers the methodology and experiments implemented for the research. Section 3 covers the Results and Discussion to present the result of the RL algorithm and discuss the objectives of the research. Section 4 covers the conclusion.

**Research Paper Objectives:**

- To study Unity and AI for the development of innovative AI hypercasual games.
- To examine the best optimal Reinforcement Learning methods for the agents.
- To outline the integration of AI with Unity

This paper tends to represent a glimpse into the future perspective of the gaming industry in hyper-casual gaming platforms.

**Methodology:**

The following methodology is adopted in this paper in which data is gathered from different sources, which relates to the domain of this paper. The collection of data for this work has been gathered from the following tools:

- Unity Engine
- C#
- Python (Anaconda Software)

**Training Environment:**

The environment was created for the hyper-casual game to test the reinforcement methods. The environment is based on a football game where two teams play against each other by scoring the ball into the located goals. The important feature used in the game is "ray cast" which collects data. Ray cast is used to detect the object from a distance by the agent to make

intelligent decisions based on the values collected. The agent receives the reward after completing their desired objective of scoring a goal.

**Reinforcement Learning Methods:**

Multiple RL methods exist in this case for the purpose of training these agents, whereas the following RL methods were implemented in order to attain the aim of leading it to an AI game where the agents or NPCs operate intelligently. Furthermore, the training of the neural network was implemented parallel to enhance the training process.

The following methods were utilized for the agents:

- Proximal Policy Optimization (PPO)
- Random network distillation
- Behavior Cloning (BC)
- Curiosity-driven exploration by self-supervised prediction
- Generative Adversarial Imitation Learning (GAIL)


**Figure 1:** Training Environment (Soccer field)

**Collecting Training Data:**

Statistics were collected from the feature of ray-cast, which is a component in Unity that detects objects from a distance when it is implemented on an object. It retains information about an object through an invisible laser, which is visible in the editor mode of Unity Engine.

**Agent's Behavior Parameters:**

In this game, the NPCs or agents were assigned various ray-cast lasers for detecting various objects. Agent or NPCs, for observation space, have been assigned with 11 ray-casts forward and 3 ray-casts backward in order to collect data for the training model in order to make intelligent decisions through reinforcement learning.

**Experiments:**

- **Curriculum Learning:**

In this experiment, the AI agents are trained using a curriculum approach, where they start by learning from simpler scenarios and gradually progress to more complex ones. Initially, the AI agents may compete against each other in basic soccer scenarios, with the difficulty level increasing over time. In general, it is a player versus an AI agent.

- **Self-Learning:**

This experiment focuses on training AI agents to play soccer against each other. All players in the game are controlled by AI algorithms, which learn from their interactions and experiences within the simulated environment. In general, it is an AI agent versus an AI agent.

**Result and Discussion:**

In this section of the proposed paper, the research objectives have been discussed. Clear themes have been identified to assist in discussing the research goals effectively in order to attain the main aim of the proposed paper.

**To Study Unity and AI for the Development of Innovative AI Hyper Casual Games:**

Unity 3D offers several features, which has not been discussed yet. Unity supports asset tracking, scripting, processing, and physics which eases the development time for developers for their projects. According to a paper [5], the Unity engine supports 27 diverse platforms and devices in an environment that is user-friendly development. Regarding AI in the game industry, it is a broad topic. AI comprises machine learning and decision-making abilities specifically for the behavior of agents in the game. It is a significant aspect of contemporary game development. Developing an AI game can have an immersive impact on gameplay [6].

The first AI game, called the "Spacewar!", was developed at MIT in the early 1960s [6]. It was a multiplayer space combat genre game that had AI-controlled opponents. Nevertheless, AI games evolved over the years since then for the purpose of enhancing the gaming experience. During those days, PCs did not exist as large mainframes or computers were mostly found in institutions. In recent times, large gaming organizations have started to implement AI into their games such as Red Dead Redemption 2, which is a popular game. But it must be taken into notice that such games are mostly high-level and require specific consoles or upgraded PCs.

In this case, hypercasual games, such as Candy Crush game or Subway Surfers are known for their simple features and mechanics, convenient gameplay, and minimum design, enabling them to be accessible to a broad audience of the gaming genre. To convey AI games to a broad audience of such genres, hypercasual games are mostly played by people. By introducing AI to the platform of such games, it can uplift the gaming industry to a greater extent.

**To Examine the Best Optimal Reinforcement Learning Methods for the Agents:**

According to a paper, there are various training tools for the evaluation of reinforcement learning algorithms [1]. There is a reinforcement learning algorithm known as the NEAT or Neuro Evolution of Augmenting Topologies. NEAT has various applications in several domains, including game development. However, the shortcoming of NEAT, which is compared to the reinforcement learning methods implemented in this paper, is that it does not inherit a memory system. It focuses on evolving the structure and linkage of neural networks instead of having a mechanism for remembering information over the period. However, in this paper as mentioned above, the following reinforcement learning methods have been utilized for the training agents in the game environment:

**Proximal Policy Optimization (PPO):**

The default core method is the Proximal Policy Optimization (PPO) reinforcement algorithm, which was first presented by [7]. By limiting policy updates to a limited range of $[1 - \varepsilon, 1 + \varepsilon]$, PPO is renowned for its stable and user-friendly architecture. This prevents huge policy changes from gradient ascent, which can be damaging. Deviations of rt (θ) from 1 in policy changes are penalized by the objective function. Mathematically, the function of PPO is represented by;

$$L^{CLIP}(\theta) = \hat{E}t\left[(rt(\theta)\hat{A}t, , clip(rt(\theta), 1 - \varepsilon, 1 + \varepsilon)\hat{A}t\right]$$

- ε is a hyperparameter, (usually 0.1 or 0.2).
- Êt is the expectation over timesteps.
- rt is the probability ratio of new and old policies.
- Ât is the estimated advantage at time t.
- θ is the policy parameter.

The Proximal Policy Optimization (PPO) reinforcement algorithm is a stable and user-friendly method that is ideal for training agents in-game environments since it makes use of the clipped objective function.

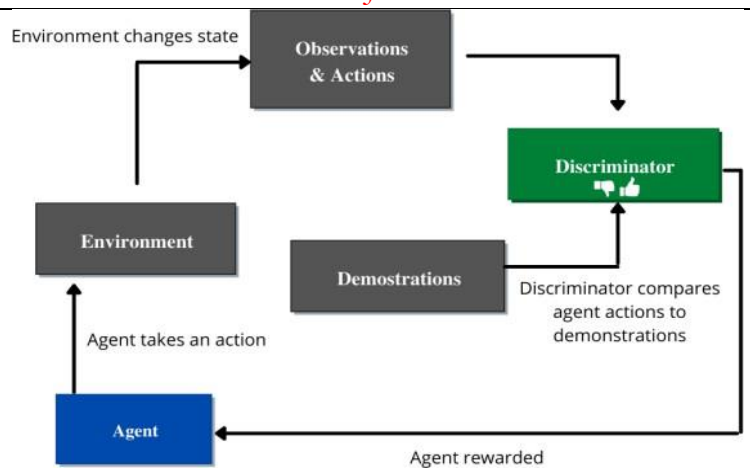


**Figure 2:** Generative Adversarial Imitation Learning

**Generative Adversarial Imitation Learning (GAIL):**

Among model-free imitation algorithms, Generative Adversarial Imitation Learning (GAIL) is unique in that it performs better than previous model-free techniques, particularly in complex contexts [8]. This approach depends more on learning from the environment and requires more contact with it than model-based approaches because it is model-free. The method is ad hoc action exploration to find out which activities lead to a policy shift in favor of the expert's policy.

With GAIL, agents can learn behavior without explicitly knowing the reinforcement signal by watching the expert's activities. The expert records and saves its actions in a demonstration file with associated states and actions. With this framework, behavior may be directly extracted from the given example.

**Curiosity-Driven Exploration by Self-Supervised Prediction:**

According to [9], the reports have observed that in real-world settings, rewards are frequently insufficient or nonexistent. In such situations, agents might be encouraged to explore their surroundings and learn new abilities that are necessary to accomplish their goals by using their curiosity as a motivator. Here, the agent's curiosity is defined as its ability to predict the results of its actions.

According to [9], the reports have explained that curiosity-driven exploration encourages agents to explore more effectively, lowering the requirement for significant contact with the environment to accomplish objectives. This method also allows the agent to forecast how its actions will turn out in scenarios that haven't been witnessed before. One notable benefit of curiosity-driven exploration is its capacity to motivate agents to discover new places in games.

**Behavior Cloning (BC):**

The need to accurately mimic human behavior in intelligent entities is addressed by behavior cloning, a concept Hussein [10] introduced. This method allows behavior-cloned agents to mimic human behavior in similar circumstances. Autonomous vehicles, helper robots, and computer-human communication are just a few of the domains in which it finds use. As a basic aspect of behavior cloning, imitation learning depends on insights from the environment and the agent's actions.

Imitation learning has intrinsic limitations, despite its usefulness. It is contingent upon the availability of expert demonstrations of a high caliber, and the agent's competence is limited to what people can demonstrate. Cloning behavior is very useful for agents that need to closely mimic the examples given. One disadvantage of such agents is that they are not able to improvise when choosing what to do.

Behavior cloning is a useful method for giving agents human-like behavior, and it has applications in many different fields. Although it works well for faithful imitation, it has

limitations in terms of the caliber of expert demonstrations and the incapacity of agents to perform at levels higher than those of humans.

**Random Network Distillation (RND):**

Burda [11] introduced Random Network Distillation, which modifies the reinforcement learning techniques by adding an exploration incentive. The bonus is ascertained by calculating the prediction error of a neural network (NN) feature acquired from observations using a fixed, randomly initialized NN. This strategy serves as a directed exploration tool with the primary goal of addressing the problem of scant rewards. Essentially, the agent is rewarded for discovering new things when interacting with the environment or for venturing into locations that haven't been explored before.

Setting itself apart from a lot of curiosity-based approaches, the RND methodology shows persistence in the face of obstacles like becoming stuck when subjected to random noise, like static on a TV screen. The method's dependence on an exploration bonus instead of an absolute prediction error accounts for this robustness. As a result, RND shows great promise as an algorithm, especially for agents whose job it is to navigate complicated environments where the observation data obtained is highly noisy.

In reinforcement learning, Random Network Distillation presents a fresh viewpoint on exploration and efficiently handles situations with plenty of noise and scant rewards. Because of its unique methodology, it is positioned as a viable algorithm for agents exploring complex and noisy scenarios. With the training of each algorithm, PPO gave a positive response with higher accuracy in terms of reinforcement learning, where the agents strategically operated throughout the game. Other algorithms gave a response, while PPO turned out to have a higher accuracy of 95% as compared to the other RL algorithms, which were less in terms of their accuracy and response by the agents in the environment.

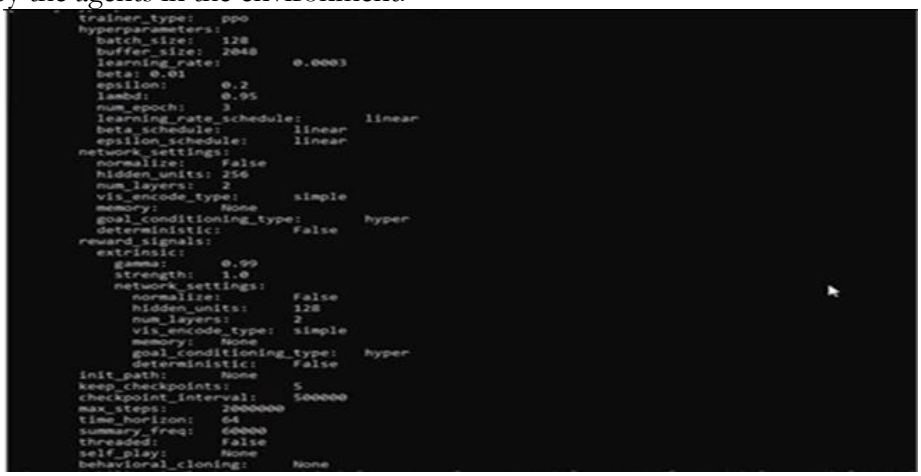


**Figure 3:** Training Behavior of PPO algorithm in the Soccer Game using Anaconda Prompt integrated with the Unity Engine's file project

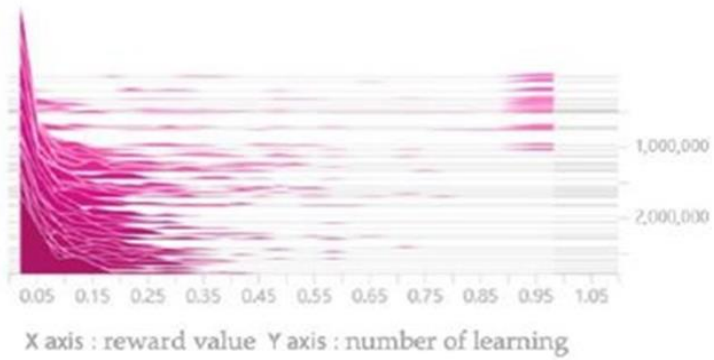


**Figure 4:** Cumulative-reward-value graph using the PPO algorithm

**To examine the Integration of AI with Unity:**

Unity cross-platform has the feature of supporting integration with other applications. Although, it has a library of AI that supports the Navigation Mesh, as discussed in the above section, which is for agents to move intelligently in the environment by avoiding obstacles. Unity contains the API that allows the integration of Python with the agents in the environment by assigning them the algorithm depending upon the purpose.

The Unity Machine Learning API package in Unity enables the developers to integrate Python with Unity Editor in order to train the behaviors of agents (See Figure Below) **[14]**. The API package contains the following three components. The components are agent, python API, and trainer. The agent is the character model in the game to which the model has been assigned. It communicates the training model through the Python API via a communicator in Unity. Hence, the agent utilizes the communicator in order to connect with the trainers through the Python API.

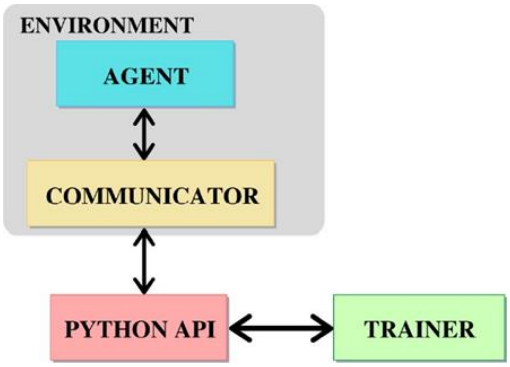


**Figure 5:** API Framework Model:

**Conclusion:**

To conclude, the paper seeks to enhance the gaming experience for users by using hyper-casual game platforms with intelligent NPCs or agents. With the discussed methodology and discussion, the algorithm utilized for the game is PPO due to its stability and policy improvement. Furthermore, it can be observed that introducing AI to hyper-casual games will not only uplift the game industry but encourage other independent developers to enter the domain of game development due to its modern perks of introducing AI with optimal reinforcement algorithms. Three objectives were formulated to attain these goals such as to study Unity and AI for the development of innovative AI hyper-casual games, to examine the best optimal Reinforcement Learning methods for the agents, and to outline the integration of AI with Unity. Associated with these objectives, three questions were also formulated for the same purpose of attaining these goals what are the benefits of AI games? To what extent AI games can shape the gaming industry? Which Reinforcement Learning methods are optimal for the game agents to operate intelligently in the game?

Hence, AI games are the future of the gaming industry and it has been rising since the 1960s with the development of the "Space Wars!" game to contemporary games like Red Dead Redemption 2. It will start appearing as a common feature in every game in the future.

**Acknowledgement:**

The success and final outcome of this project required a lot of guidance and assistance from many people and we are extremely fortunate to have this all along the completion of our project. We want to convey our sincere gratitude and appreciation to our supervisor, Dr. Muniba Ashfaq, and our advisor, Engineer Abdullah Hamid, for the significant assistance and support during this effort. Finally, we want to thank our parents and professors for helping and motivating us with our projects and academic endeavors. We acknowledge our University, University of Engineering and Technology Peshawar, for providing us with a feasible environment in order to work on our project.

**Author's Contribution:**

**Mosaddiq Billah:**

- **Conceptualization:** Developed the initial idea for creating a soccer theme-based hypercasual game.
- **Writing:** Drafted the initial version of the project proposal and technical documentation outlining the game's architecture and development strategy.
- **Literature Review:** Conducted comprehensive review of literature on AI gaming and Unity development.

**Aanoora Seher:**

- **Data Collection:** Assisted in gathering relevant datasets and resources necessary for training and testing the AI algorithms.
- **Data Analysis:** Performed statistical analysis on game data to identify patterns, trends, and performance metrics for evaluating the AI's effectiveness.
- **Data Interpretation:** Analyzed experimental results to evaluate the AI's behavior, identify strengths and weaknesses, and propose improvements for enhancing gameplay.

**Ahmad Bahar:**

- **Unity Development:** Developed the initial prototype of the game and implemented the script logic for the agents and game objects.
- **Graphic Design:** Designed the User Interface of the game and the training environment for the agents in the game.

**Muniba Ashfaq:**

- **Supervision:** Supervised the entire research project, providing guidance and feedback on project milestones, research methodology, and documentation.

**Abdullah Hamid:**

- **Assistance:** Provided hands-on assistance throughout the project, offering technical support and troubleshooting for various aspects of game development.
- **Mentorship:** Offered mentorship by sharing knowledge of advanced concepts and features applicable to the game development process, guiding the team in implementing innovative ideas and best practices.

**Conflict of Interest:**

The authors declare that there is no conflict of interest regarding the publication of this paper. We have no financial or personal association with organizations or individuals that could influence the objectivity or interpretation of the project findings. This research was conducted with academic integrity and with the objective of achieving the primary goals of the research.

**Project Details:**

This research was conducted as part of the final year project for the Bachelor of Science in Computer Systems Engineering program at the University of Engineering and Technology Peshawar. The project was completed during the academic year of 2023-2024. The project did not incur any cost as the resources were utilized by the university, including software tools and guidance from faculty mentors. The completion date for the project was May 1st, 2024.

**References:**

[1] A. Juliani et al., "Unity: A General Platform for Intelligent Agents," Sep. 2018, Accessed: May 06, 2024. [Online]. Available: https://arxiv.org/abs/1809.02627v2

[2] A. Canossa, "Interview with Nicholas Francis and Thomas Hagen from Unity Technologies," Game Anal., pp. 137–142, 2013, doi: 10.1007/978-1-4471-4769-5_8.

[3] C. Becker-Asano, F. Ruzzoli, C. Hölscher, and B. Nebel, "A Multi-agent System based on Unity 4 for Virtual Perception and Wayfinding," Transp. Res. Procedia, vol. 2, pp. 452–455, Jan. 2014, doi: 10.1016/J.TRPRO.2014.09.059.

[4] T. N. Malete, K. Moruti, T. S. Thapelo, and R. S. Jamisola, "EEG-based Control of a 3D Game Using 14-channel Emotiv Epoc+," Proc. IEEE 2019 9th Int. Conf. Cybern. Intell. Syst. Robot. Autom. Mechatronics, CIS RAM 2019, pp. 463–468, Nov. 2019, doi: 10.1109/CIS-RAM47153.2019.9095807.

[5] "Using the Unity Game Engine to Develop SARGE: A Case Study." Accessed: May 06, 2024. [Online]. Available: https://www.researchgate.net/publication/265284198_Using_the_Unity_Game_Engine_to_Develop_SARGE_A_Case_Study

[6] J. Wexler, "Artificial Intelligence in Games: A look at the smarts behind Lionhead Studio's 'Black and White' and where it can and will go in the future".

[7] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. K. Openai, "Proximal Policy Optimization Algorithms," Jul. 2017, Accessed: May 04, 2024. [Online]. Available: https://arxiv.org/abs/1707.06347v2

[8] J. Ho and S. Ermon, "Generative Adversarial Imitation Learning," Adv. Neural Inf. Process. Syst., pp. 4572–4580, Jun. 2016, Accessed: May 06, 2024. [Online]. Available: https://arxiv.org/abs/1606.03476v1

[9] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-Driven Exploration by Self-Supervised Prediction," IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work., vol. 2017-July, pp. 488–489, Aug. 2017, doi: 10.1109/CVPRW.2017.70.

[10] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation Learning," ACM Comput. Surv., vol. 50, no. 2, Apr. 2017, doi: 10.1145/3054912.

[11] Y. Burda, H. Edwards, A. Storkey, and O. K. Openai, "Exploration by Random Network Distillation," 7th Int. Conf. Learn. Represent. ICLR 2019, Oct. 2018, Accessed: May 06, 2024. [Online]. Available: https://arxiv.org/abs/1810.12894v1